|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

|  |
| --- |
| Capstone Project Document |
| PET DATING |
| **VERSION 1.0** |
| |  |  | | --- | --- | |  | | | **Supervisor** | **Phan Duy Hùng** | | **Group Members** | **Lê Đức Chính – SE04365**  **Trịnh Bá Tân – SE04700**  **Lê Văn Sơn – SE04807**  **Dương Ngọc Duy – SE05121** | | **Capstone Project code** | **PD** | |

Ha Noi, May 20, 2019

**CHAP 1: INTRODUCTION**

* 1. Propose

This chapter provides an overview of the capstone project. It includes the project information, the background of the capstone project, the problem and our proposal to solve the problem.

* 1. Project Information

|  |  |
| --- | --- |
| Project Information | |
| Project name | Pet Dating |
| Project code | PD |
| Group name | Orange Wave |
| Product type | Mobile Application |
| Timeline |  |

* 1. The People

Supervisor:

|  |  |  |  |
| --- | --- | --- | --- |
| Full name | Phone | E-Mail | Title |
| Phan Duy Hùng |  | HungPD2@fe.edu.vn | Supervisor |

Member:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Full name | Student ID | Phone | E-Mail | Title |
| Lê Đức Chính | SE04365 | 0395052756 | Chinhldse04365@fpt.edu.vn | Leader |
| Trịnh Bá Tân | SE04700 | 0347766759 | Tantbse04700@fpt.edu.vn | Member |
| Lê Văn Sơn | SE04807 | 0975637302 | Sonlvse04807@fpt.edu.vn | Member |
| Dương Ngọc Duy | SE05121 | 0989188497 | Duydnse05121@fpt.edu.vn | Member |

* 1. Background

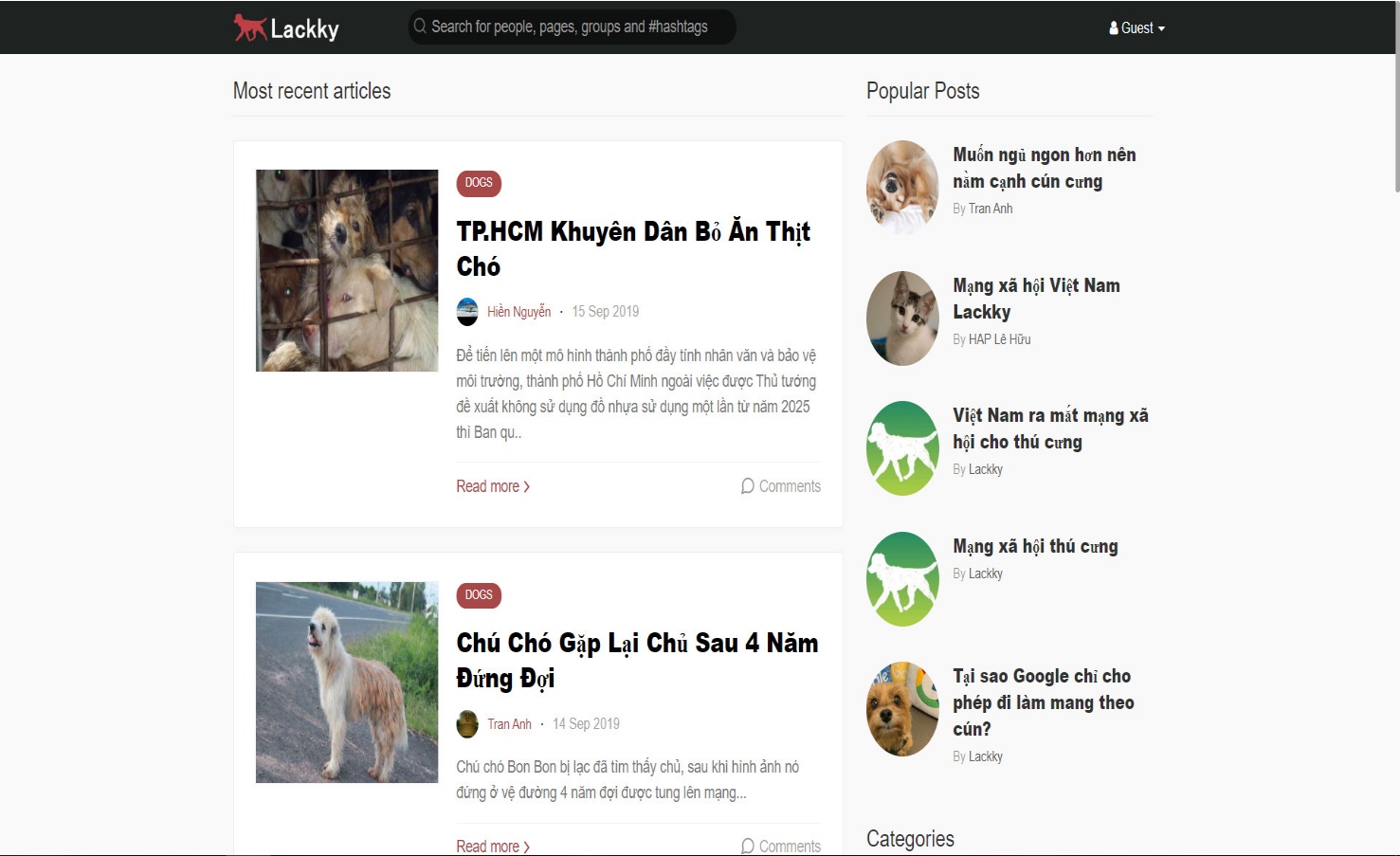
In 21st century, people are interested in pet. According to the survey of Q&ME, 40% Vietnamese often wear clothes for their pet and many people remember and celebrate birthdays for these little friends. Each family has at least a pet.

* 1. The Problem

Everybody cares about the life of pet such as food, clothes, morale and mates. In Viet Nam, People have trouble finding mate for your pet. A few groups in Facebook support about that but it is not convenient and maybe cost a lot of time in some case. Currently, the groups mostly support a kind of pet like doggie dog, pitbull dog, British shorthair cat, Scottish Fold cat, ….

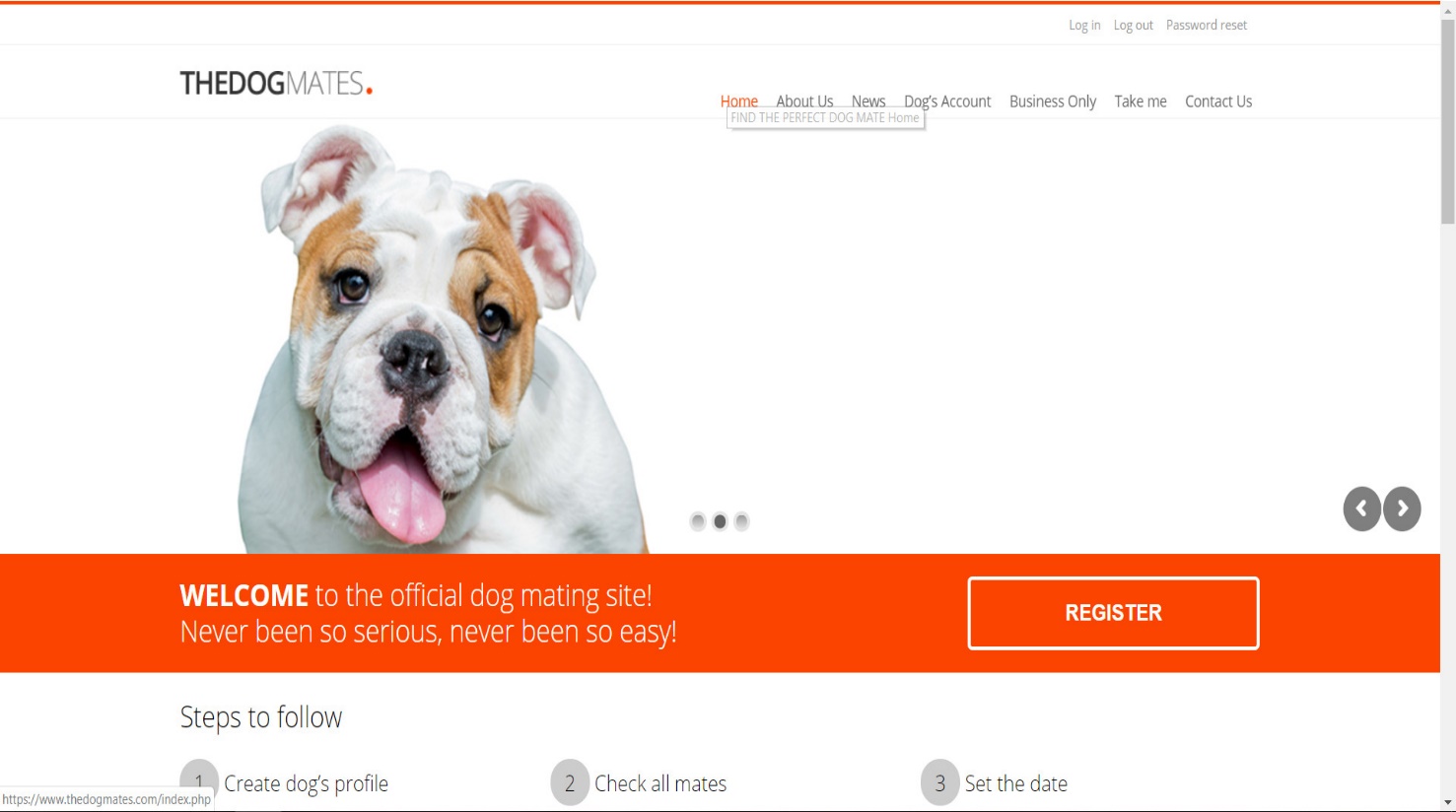
* 1. Literature Review

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



*Figure 1: lackky.com*

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



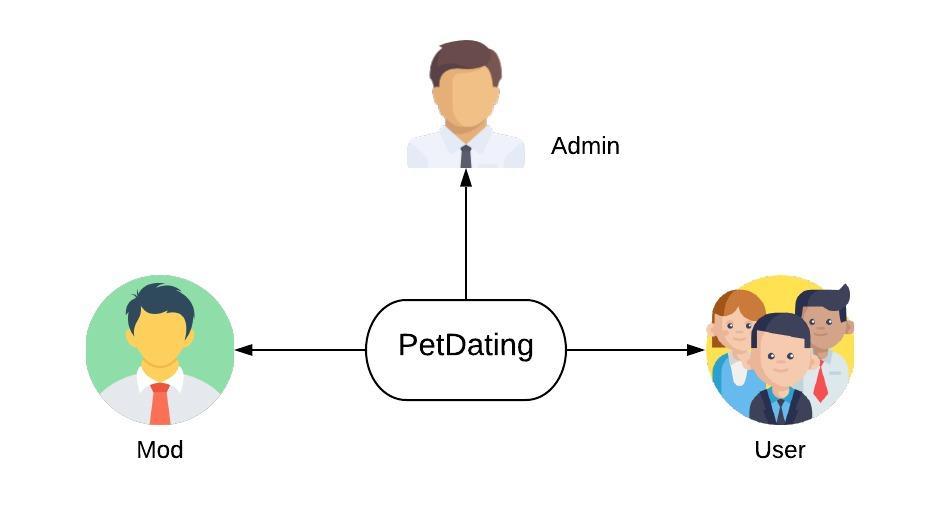
*Figure 2: the dogmates.com*

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| * Have multiple kind of dogs. * Vote for dog | * Not popular in Viet Nam. * Complex in registration and approval. * Serve only for dog. |

* 1. The Proposal
     1. The idea

We want to build a mobile application where everyone can join to the worlds of 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; post status. In addition, everyone can connect and chat about pets. It helps making pets dating becomes easy, safe and enhance the community. Our idea has two part: mobile application for user and website for manager.

* + 1. The proposal



*Figure 3: Role in PD system*

* + 1. System functions

There are three roles and responsibilities of each role on PetDating:

|  |  |
| --- | --- |
| **Role** | **Responsibilities** |
| Admin | People who manage the PD system and has all authority on system |
| Mod | Somebody who manage the PD system. But with less authority than administrator |
| Users | People who join and use PD application |

The system has the following roles and main functions:

* Admin:

+ Login and Logout to website.

+ Authorize to moderator.

* Mod:

+ Login and Logout to website.

+ Review reports from user.

* Users

+ Login and Logout to mobile application.

+ Allow users to join PD, view user profile and modify your profile.

+ Allow users to create, modify your pet profile and view the pet profile. The pet profile defines the name, sex, age, … of pets.

+ Allow users to upload images, post status and set privacy such as private or public.

+ Users can use dating function to find mates for pets. There are two types of finding mates:

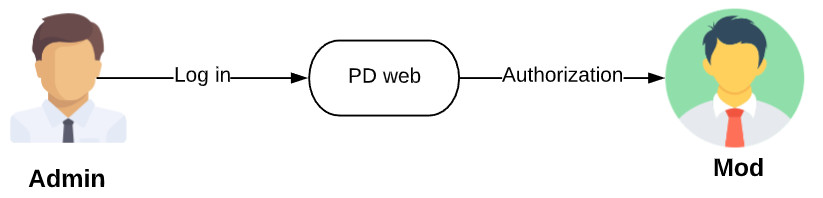
* + Find mates by automation: Application suggests pets base on your pet that the pet has selected.
  + Find mates by manual: Application suggests pets base on your choose.

+ Users can use chatting system to connect to other users.

+ Allow users to like or report the posts.

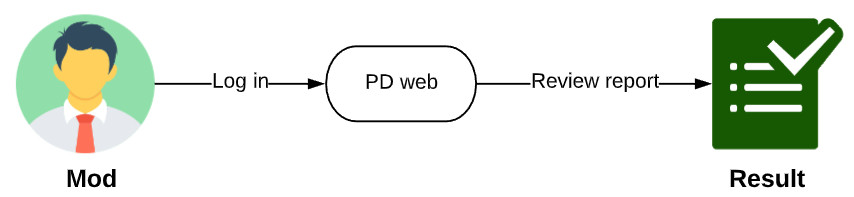
+ Allow users to feedback about system.

* + 1. Business flow
* **Authorization process**

****

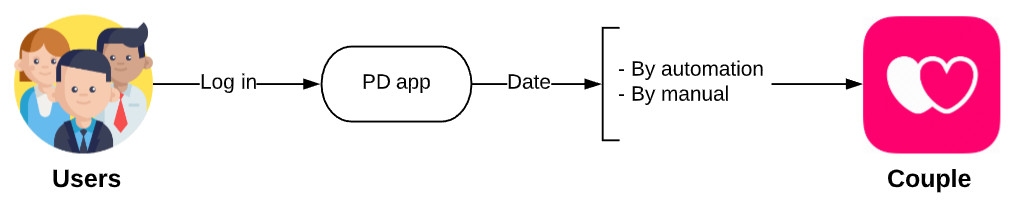
*Figure 4: Process of authorization on PD*

* **Review report process**

****

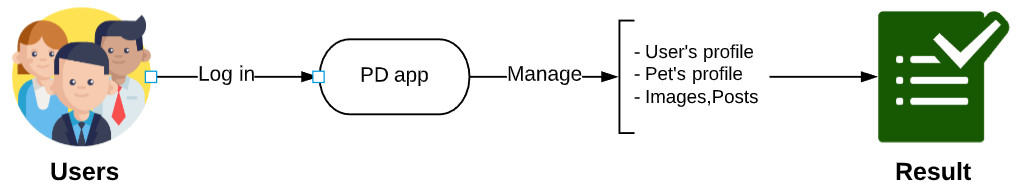
*Figure 5: Process of review report on PD*

* **Dating process**



*Figure 6: Process of dating on PD*

* **User management process**



*Figure 7: Process of user management process on PD*

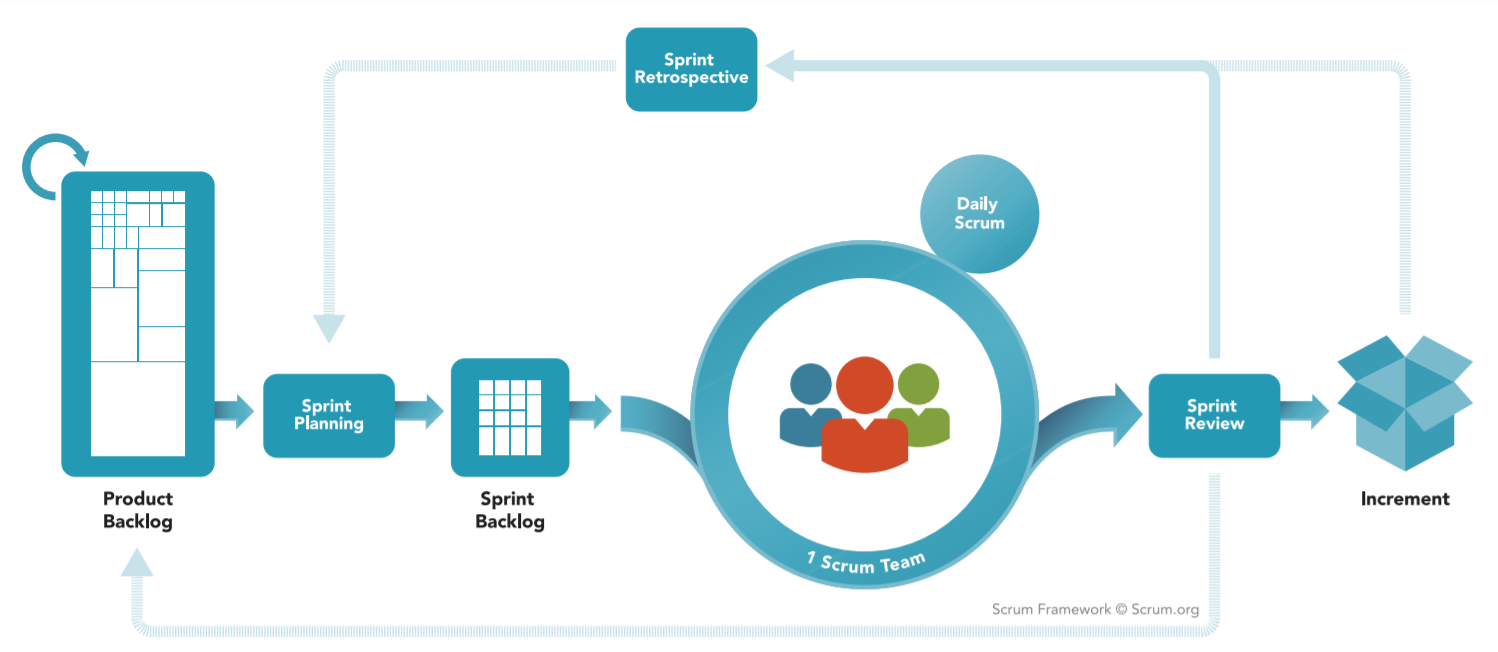
# **CHAP 2: SOFTWRE PROJECT MANAGEMENT PLAN (SPMP)**



## 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.

## Software Process Model



*Agile Model – SCRUM*

### 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.*

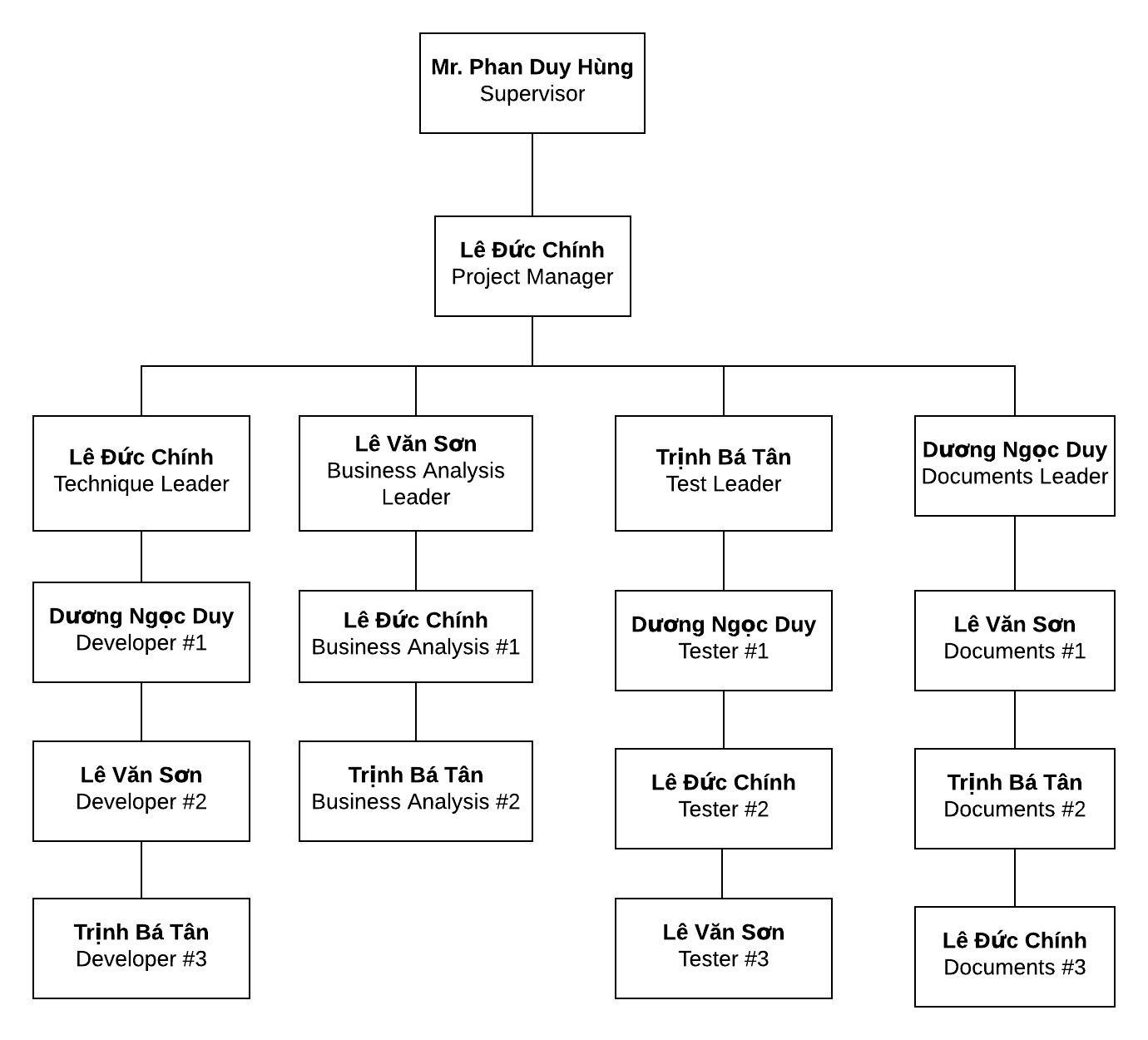
### Advantages and disadvantages of SCRUM

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

*Table X: Advantages and disadvantages of SCRUM*

## Project Organization

### Organization structure



*Figure X: Organization structure*

### 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. | Lê Đức Chính |
| **Business analyst team** | | |
| BA leader | * Elicit and analyze requirement. * Define scope and create SRS template. * Design entity relationship diagram. | Lê Văn Sơn |
| BA #1 | * Define business process flow and object state. | Lê Đức Chính |
| BA #2 | * Capture and specific describe use case. | Trịnh Bá Tân |
| **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. | Lê Đức Chính |
| Dev #1 | * Lead chatting function. * Design a part of database. * Design and code chatting function. * Code others screen. * Design mockup for application. | Dương Ngọc Duy |
| Dev #2 | * Design and code dating function. * Code others screen. * Design mockup for application. | Lê Văn Sơn |
| Dev #3 | * Design and code chatting function. * Code others screen. * Design mockup for application. | Trịnh Bá Tân |
| **Tester** | | |
| Test leader | * Create template testing documents. * Define test strategy, create test plan and defect log temple. | Trịnh Bá Tân |
| Test #1 | * Implement test case and log defect. | Dương Ngọc Duy |
| Test #2 | * Implement test case and log defect. | Lê Đức Chính |
| Test #3 | * Implement test case and log defect. | Lê Văn Sơn |
| **Documentation** | | |
| Docs leader | * Prepare all the documentation relating to the project. * Give format in documentation. | Dương Ngọc Duy |
| Docs #1 | * Check spelling errors and grammar errors. | Lê Văn Sơn |
| Docs #2 | * Check spelling errors and grammar errors. | Trịnh Bá Tân |
| Docs #3 | * Check spelling errors and grammar errors. | Lê Đức Chính |

*Table X: Roles and responsibilities*

## Tools And Techniques

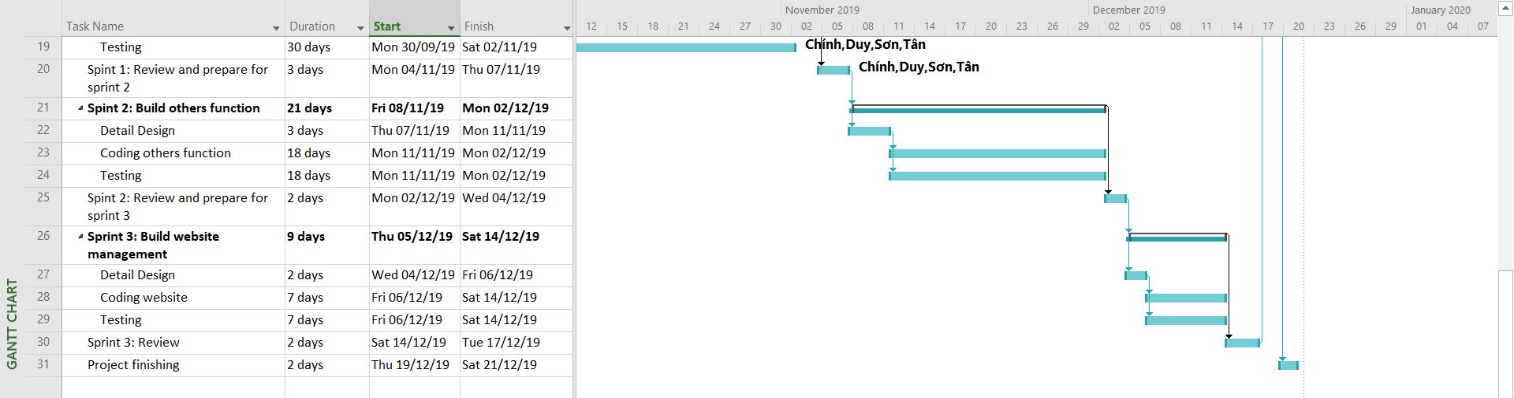
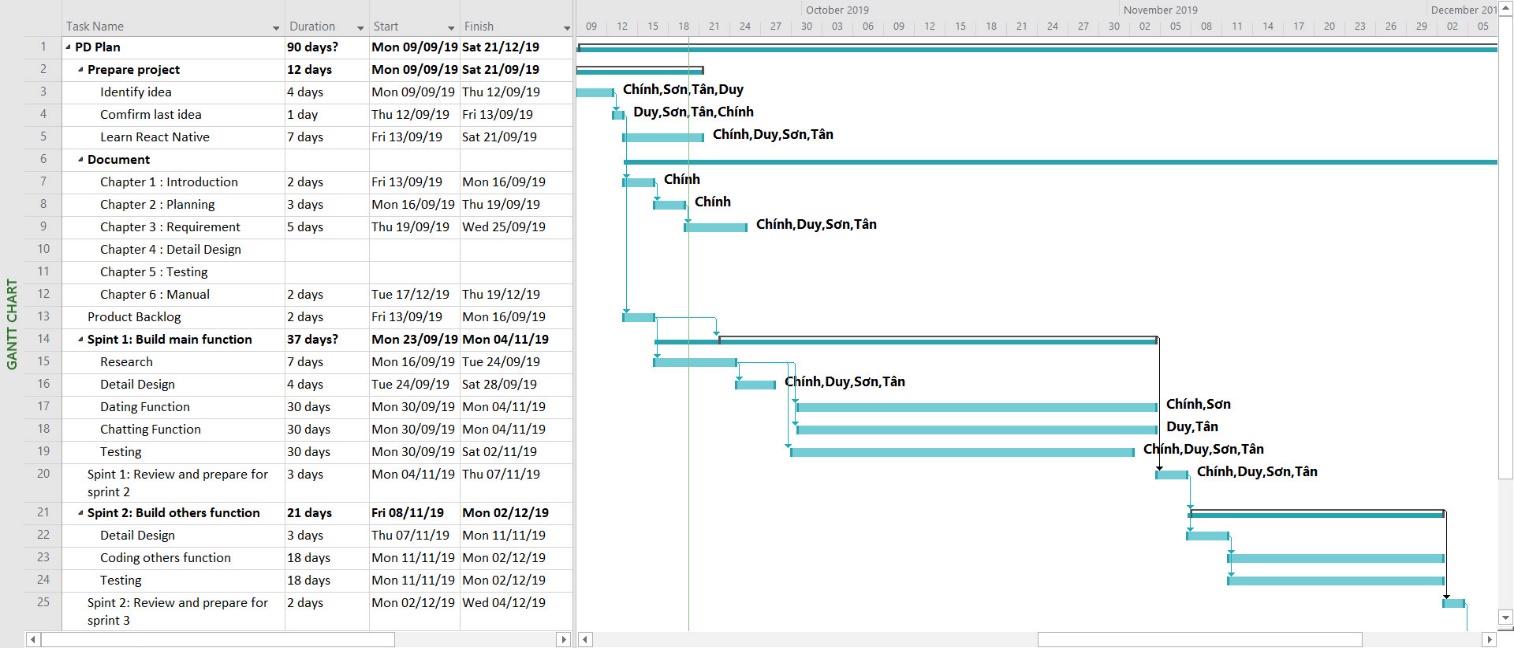
|  |  |  |
| --- | --- | --- |
| **Tool and techniques** | **Version** | **Description** |
| Microsoft Project | 2019 | Management plan |
| Trello | Online | Control task, backlog |
| Microsoft Office | 2019 | Documentation tool includes: Words, Excel, PP, … |
| GIT kraken | latest | Control source code |
| Facebook | Online | Communication tool |
| Lucidchart | Online | Diagram, UML, chart tool |
| Visual Studio Code | 1.34 | Text editor for coding |
| NodeJS | 8.9.4 | Environments |

*Table X: Tools and techniques*

## Project Management Plan

### Project schedule

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



*Figure X: Project Planning*

### Meeting minutes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Name/Code** | PetDating | **Date of meeting** | |  |
| **Conductor** | Lê Đức Chính | **Location** | |  |
| **Topic of meeting** |  | | | |
|  |  | | | |
| **Attendees:** | | | | |
| **Name** | **Role** | | **Attendance** | |
| Phan Duy Hùng | Supervisor | | Present | |
| Lê Đức Chính | PM | | Present | |
| Trịnh Bá Tân | Team member | | Present | |
| Lê Văn Sơn | Team member | | Present | |
| Dương Ngọc Duy | Team member | | Present | |
|  | | | | |
| **Meeting objective:** | | | | |
|  | | | | |

### Risk management plan

**\*Risk identification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Risk description** | **Contingency/Mitigation plan** | **Probability** | **Impact** |
| 1 | Poor solving conflict management | Working in group team building | High | Medium |
| 2 | Poor defining scope, ambiguous definition | Meeting with BA expert | Medium | High |
| 3 | Change Requirement/Objective during the project | Meeting and redefine objective with each team member. | Medium | High |
| 4 | Fail for estimating iteration time, inadequate change | Doing overnight work to keep process continuing | High | Medium |
| 5 | Conflicts among team members | - Transfer problems to whole team to resolve it.  - Create happy and friendly environment among team members. | Low | Medium |
| 6 | Program testing is not fully covered | - Meeting with Test leader  - Meeting with supervisor | Medium | Medium |
| 7 | Application can be crash | Improve performance for application | Low | Medium |

*Table X: Risk identification*

**\*Probability - Impact matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Probability | High |  | R1, R4 |  |
| Medium |  | R6 | R2, R3 |
| Low |  | R5, R7 |  |
|  |  | Low | Medium | High |
|  |  | Impact | | |

### Communication plan

Project report and meeting:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Activity** | **Stakeholder** | **Time** | **Description** |
| 1 | Weekly report | * Team member * Supervisor | Friday | * 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 |

*Table x: Project report and meeting*

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. |

*Table X: Project communication*

### 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 |

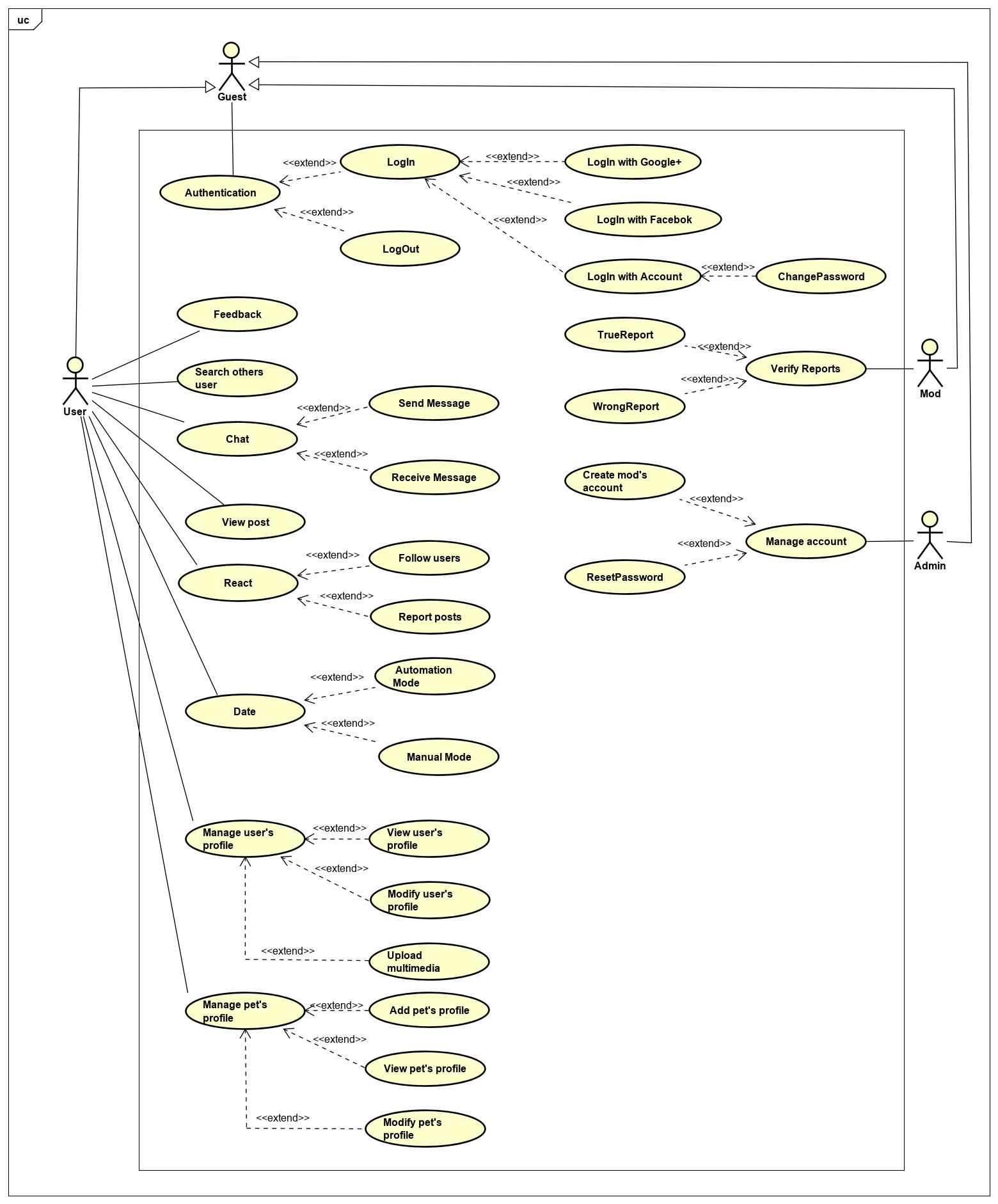
Table X: Coding Convention

**CHAP 3: SOFTWARE REQUIREMENT SPECIFICATION**

* 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.

* 1. Functional Requirement
     1. Use case diagram



*Figure X: Use case diagram*

* + 1. Business rules

|  |  |
| --- | --- |
| **ID** | **Description** |
| B01 | Account's email address must be valid. |
| B02 | Account's password must be at least 8 characters in length and must contain at least 1 uppercase letter, 1 lower case letter, and 1 digit. |
| B03 | Account's password must not be stored as plain text. Instead it must be hashed using a secure hash algorithm. |
| B04 | Admin/Mod must provide their account's email address and password when logging into the website. |
| B05 | Only Admin account can create mod’s account. |
| B06 | Only Admin account can reset password of mod’s account. |
| B07 | The field must be filled by characters consist of alphabet and numbers. |
| B08 | Max character is 240. |
| B09 | Image file size must not exceed 5MB. |
| B10 | Image file type must be either JPG or PNG. |
| B11 | Google+ account or Facebook account must be valid. |
| B12 | When changing password, new password must not be the same as the current password. |
| B13 | Follow the terms of user. |
|  |  |
|  |  |
|  |  |

* + 1. Use case list

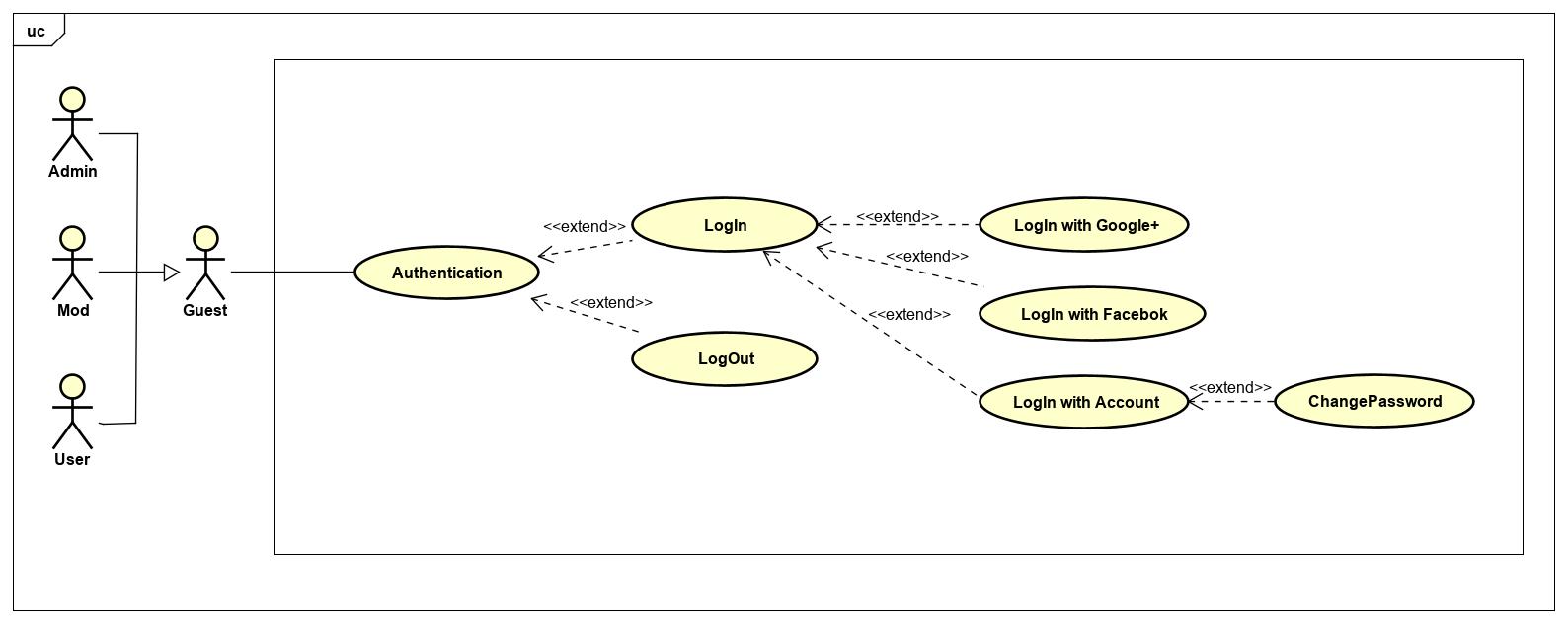
|  |  |  |
| --- | --- | --- |
| **Actors** | **Inherits** | **Description** |
| Guest |  | Everyone who want to use PetDating. |
| User | Guest | Everyone who has an account on the PetDating system and used it. |
| Admin | Guest | People who has responsibilities about mod’s account. |
| Moderator | Guest | A group of people who review reports from User. |

*Table X: Actors description*

|  |  |  |
| --- | --- | --- |
| **ID** | **Actor** | **Name** |
| GU-UC-1.0 | Guest | Login with Google+ or Facebook |
| GU-UC-2.0 | Login with Account |
| GU-UC-3.0 | Logout |
| GU-UC-4.0 | Change Password |
| AD-UC-1.0 | Admin | Create mod’s account |
| AD-UC-2.0 | Reset Password |
| MD-UC-1.0 | Moderator | True Report |
| MD-UC-2.0 | Wrong Report |
| US-UC-1.0 | User | Feedback |
| US-UC-2.0 | Search others user |
| US-UC-3.0 | Send message |
| US-UC-4.0 | Receive message |
| US-UC-5.0 | View post |
| US-UC-6.0 | Follow user |
| US-UC-7.0 | Report post |
| US-UC-8.0 | Automation Mode |
| US-UC-9.0 | Manual Mode |
| US-UC-10.0 | View user’s profile |
| US-UC-11.0 | Modify user’s profile |
| US-UC-12.0 | Upload multimedia |
| US-UC-13.0 | Add pet’s profile |
| US-UC-14.0 | View pet’s profile |
| US-UC-15.0 | Modify pet’s profile |

*Table X: Use case list*

* + 1. Use case specification
       1. **Guest**



*Figure X: Use case diagram of Guest actor*

**\*Login with Google+ or Facebook**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-1.0** **SPECIFICATION** | | | |
| **Use case ID** | GU-UC-1.0 | **Use case version** | V1.0 |
| **Use case name** | Login with Google+ or Facebook | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 25/09/2019 | **Priority** | High |
| **Primary actor** | Guest | **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 PETDATING”, “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:**  **GU1.0-E1 – Cannot connect with Google API**  System displays error message.  **GU1.0-E2 – Cannot connect with Facebook API**  System displays error message.    **Business Rules: B11** | | | |
|  | | | |

**\*Login with Account**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-2.0** **SPECIFICATION** | | | |
| **Use case ID** | GU-UC-2.0 | **Use case version** | V1.0 |
| **Use case name** | Login with Account | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 25/09/2019 | **Priority** | High |
| **Primary actor** | Guest | **Secondary actor** | N/A |
| **Description** | Allows Admin/Mod login to PD system in website. | | |
| **Pre-condition** | Admin or Mod 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/Mod goes to login page |  | | **2.** |  | Website displays the details, which includes the following:  Title: Đăng nhập vào hệ thống PD  TextInput: Tài khoản  TextInput: Mật khẩu  Button: Đăng nhập | | **3.** | Admin/Mod fills account and password then clicks “Đăng nhập” button. |  | | **4.** |  | System generates cookies and save them. | | **5.** |  | Website navigates to homepage. |   **Alternative Flows: Admin/Mod logged and not yet log out.**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Admin/Mod goes to homepage |  | | **2.** |  | Website checks cookies then navigates to login page if not exist cookies. |   **Extension Flows: N/A**  **Exception:**  **GU2.0-E1 – Cannot connect with Account API**  System displays error message.  **GU2.0-E2 – Wrong account or password**  System displays error message and requests login again.    **Business Rules: B1, B2, B4** | | | |
|  | | | |

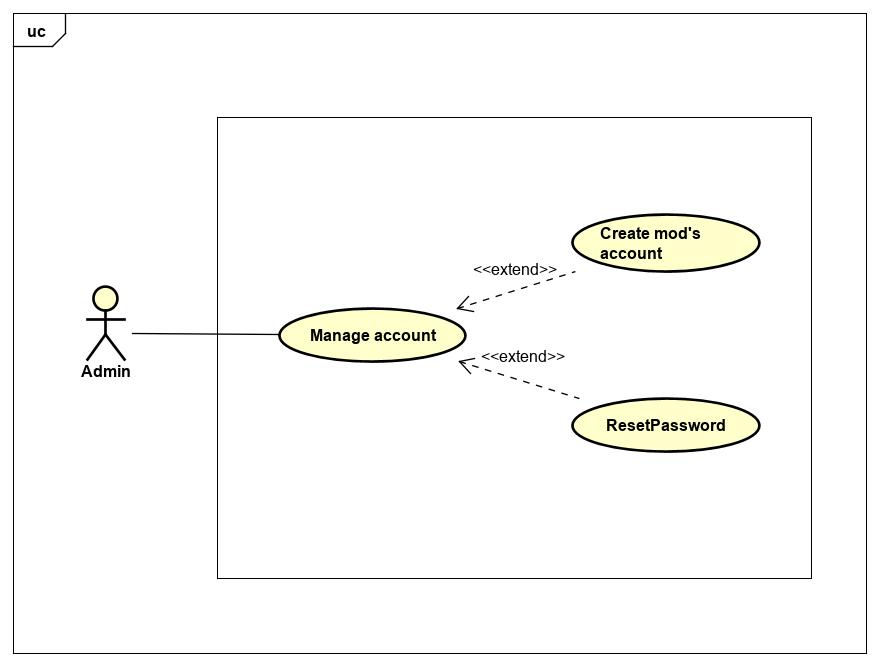
**\*Logout**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-3.0** **SPECIFICATION** | | | |
| **Use case ID** | GU-UC-3.0 | **Use case version** | V1.0 |
| **Use case name** | LogOut | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 25/09/2019 | **Priority** | High |
| **Primary actor** | Guest | **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: valid for mobile application**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | User clicks top right icon on top tab bar |  | | **2.** |  | Application displays “Account Management” screen | | **3.** | User clicks “Đăng xuất” to logout system |  | | **4.** |  | Application navigates to Login screen |   **Alternative Flows: N/A**  **Extension Flows: valid for website**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Admin/Mod clicks “Đăng xuất” on top right of website. |  | | **2.** |  | Website navigates to Login page. |   **Exception: N/A**  **Business Rules: N/A** | | | |
|  | | | |

**\*Change Password**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-4.0** **SPECIFICATION** | | | |
| **Use case ID** | GU-UC-4.0 | **Use case version** | V1.0 |
| **Use case name** | Change Password | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 25/09/2019 | **Priority** | High |
| **Primary actor** | Guest | **Secondary actor** | N/A |
| **Description** | Allows admin/mod to change their password. | | |
| **Pre-condition** | Admin/Mod logged in to system. | | |
| **Post-condition** | Password has been changed. | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Clicks “Thông tin cá nhân” on top right of website. |  | | **2.** |  | Website directs to “Thông tin cá nhân” page | | **3.** | Clicks “Đổi mật khẩu” |  | | **4.** |  | Website directs to “Đổi mật khẩu” interface | | **5.** | Fills all require field and presses the “Xác nhận” button |  | | **6.** |  | Shows sucess message and directs “Thông tin cá nhân” page |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception: N/A**  **Business Rules: B02, B12** | | | |
|  | | | |

* + - 1. **Admin**



*Figure X: Use case diagram of Admin actor*

* + - * 1. Manage account

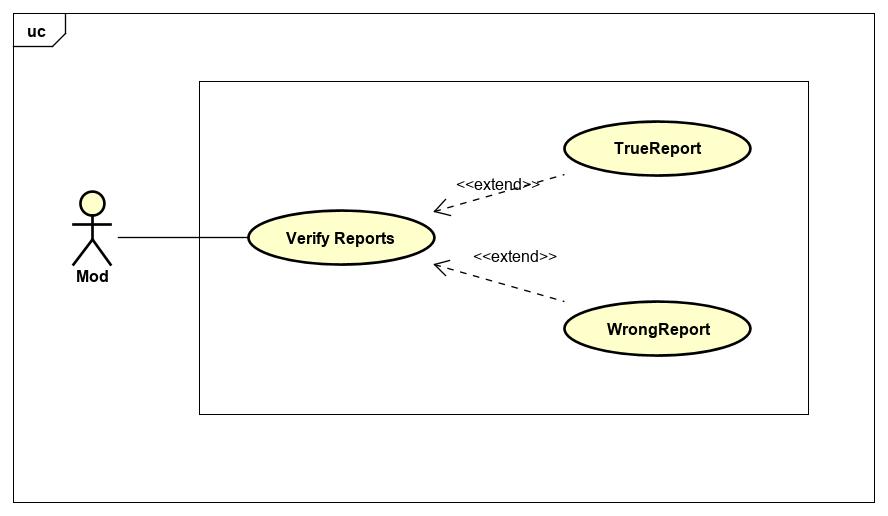
**\*Create mod’s account**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE AD-UC-1.0** **SPECIFICATION** | | | |
| **Use case ID** | AD-UC-1.0 | **Use case version** | V1.0 |
| **Use case name** | Create mod’s account | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 26/09/2019 | **Priority** | Medium |
| **Primary actor** | Admin | **Secondary actor** | N/A |
| **Description** | Administrator creates account for moderator. | | |
| **Pre-condition** | Admin has logged into system by admin account. | | |
| **Post-condition** | N/A | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Admin clicks “Quản trị”. |  | | **2.** |  | Website displays management page. | | **3.** | Admin clicks “Thêm mới” button |  | | **4.** |  | Website navigates to newmod page. | | **5.** | Fills all information of moderator and click “Thêm” button |  | | **6.** |  | Shows success message and redirect to management page. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **AD1.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: B05** | | | |
|  | | | |

**\*Reset Password**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE AD-UC-2.0** **SPECIFICATION** | | | |
| **Use case ID** | AD-UC-2.0 | **Use case version** | V1.0 |
| **Use case name** | Reset Password | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 26/09/2019 | **Priority** | Medium |
| **Primary actor** | Admin | **Secondary actor** | N/A |
| **Description** | Administrator resets password for moderator’s request. | | |
| **Pre-condition** | Admin has logged into system by admin account. | | |
| **Post-condition** | Password has been changed to default. | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Admin clicks “Quản trị”. |  | | **2.** |  | Website displays management page. | | **3.** | Admin clicks “Khôi phục mật khẩu”. |  | | **4.** |  | Website navigates to reset page. | | **5.** | Fills mod’s account that need reset password. Then clicks “Khôi phục” |  | | **6.** |  | The password of this account has been reset. Shows success message and redirect to management page |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **AD2.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: B06** | | | |
|  | | | |

* + - 1. **Moderator**

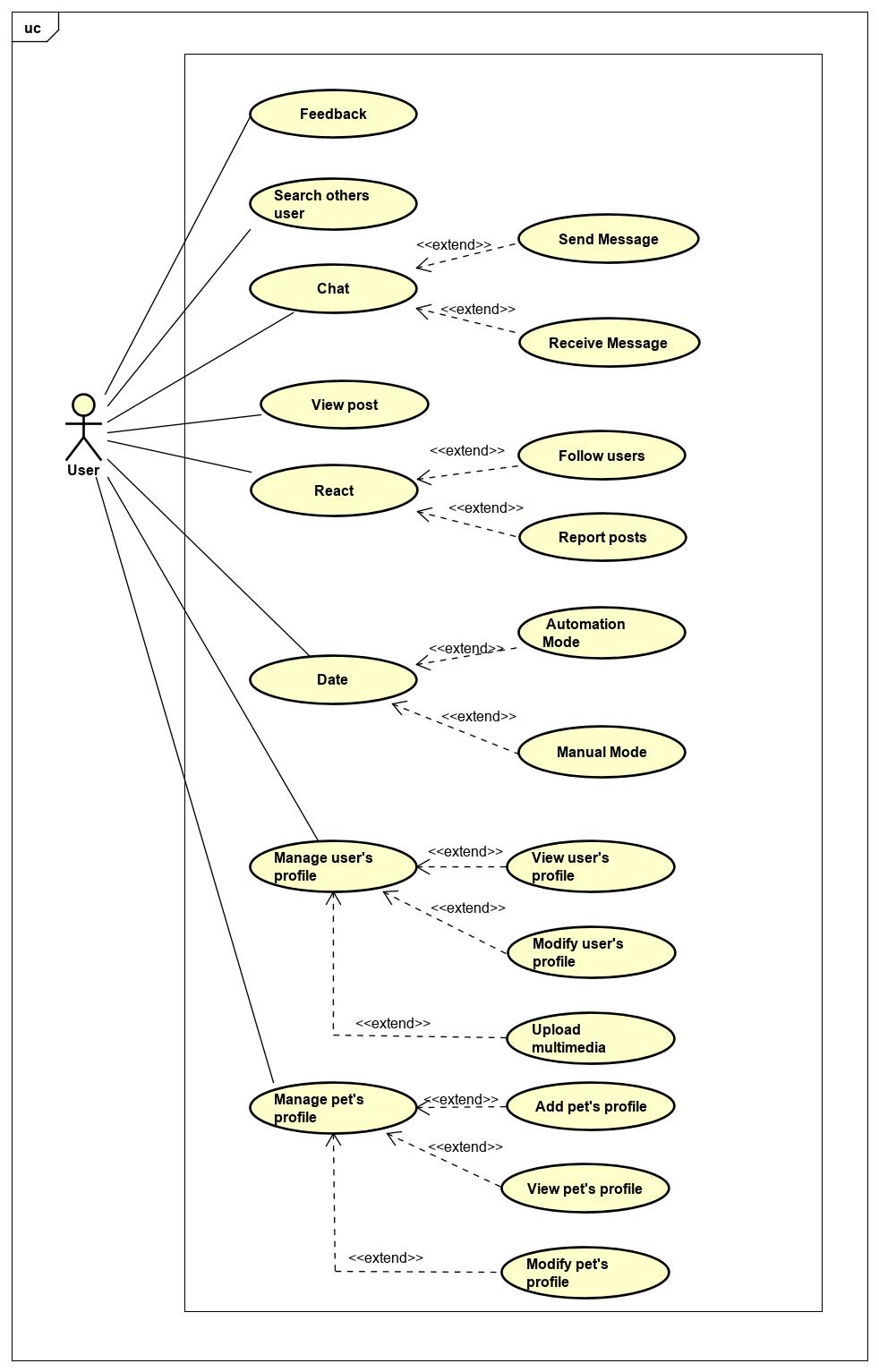


*Figure X: Use case diagram of Mod actor*

* + - * 1. Verify Reports

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE MD-UC-1.0** **SPECIFICATION** | | | |
| **Use case ID** | MD-UC-1.0 | **Use case version** | V1.0 |
| **Use case name** | Verify Report | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 26/09/2019 | **Priority** | Medium |
| **Primary actor** | Moderator | **Secondary actor** | N/A |
| **Description** | Handling reports from User | | |
| **Pre-condition** | **MD-1.0 – PRE1** Mod has logged in the system by moderator account.  **MD-1.0 – PRE2** At least report has been summited. | | |
| **Post-condition** |  | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Mod clicks “Xác minh báo cáo” |  | | **2.** |  | Website displays report page. | | **3.** | Mod clicks id of reports. |  | | **4.** |  | Website display information about report that include:  Image or video  Owner of post | | **5.1** | Mod verify and deliver result. Clicks “Vi phạm” button |  | | **6.1** |  | The system checks the number of violations of the violator. If the number less than 3, alert message to violator. If the number greater than 3, ban account. |   **Alternative Flows:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **5.2** | Mod verify and deliver result. Clicks “Không vi phạm” button |  | | **6.2** |  | Website redirect to report page. |   **Extension Flows: N/A**  **Exception:**  **MD1.0-E1 – Cannot communicate with API server**  System displays error message.    **Business Rules: B13** | | | |
|  | | | |

* + - 1. **User**



*Figure X: Use case diagram of User actor*

* + - * 1. Feedback

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-1.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-1.0 | **Use case version** | V1.0 |
| **Use case name** | Feedback | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | Low |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User feedbacks about system. | | |
| **Pre-condition** | User has logged into application. | | |
| **Post-condition** |  | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | User clicks top right icon on top tab bar |  | | **2.** |  | Application displays “Account Management” screen | | **3.** | User clicks “Phản hồi” to feedback about system. |  | | **4.** |  | Application links to other mail apps. | | **5.** | User uses own mail app to send feedback. |  |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **Business Rules: N/A** | | | |
|  | | | |

* + - * 1. Search others user

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-2.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-2.0 | **Use case version** | V1.0 |
| **Use case name** | Search others user | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | Medium |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User search others user. | | |
| **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 “News Feed” screen, which includes:  Search bar  New post  List of posts | | **3.** | User clicks search bar then fills keyword and clicks search button on bottom right of keyboard |  | | **4.** |  | Application displays list of users which have same keyword. | | **5.** | User clicks the right user name. |  |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US2.0-E1 – Cannot communicate with API server**  System displays error message.  **US2.0-E2 – User does not exist and not match**  System displays nothing.  **Business Rules: N/A** | | | |
|  | | | |

* + - * 1. Chat

**\*Send message**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-3.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-3.0 | **Use case version** | V1.0 |
| **Use case name** | Send message | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **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 message icon in receiver profile. |  | | **4.** |  | Application displays “Send Message” 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 “Chat” screen. | | **3.** | User clicks new message icon. |  | | **4.** |  | Application displays “Send Message” screen. | | **5.** | User fills receiver name and enter message then clicks icon send. |  | |  |  | Application sends message to receiver. |   **Extension Flows: N/A**  **Exception:**  **US3.0-E1 – Cannot communicate with API server**  System displays error message.  **US3.0-E2 – User does not exist or empty**  System cannot send message.  **Business Rules: N/A** | | | |
|  | | | |

**\*Receive message**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-4.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-4.0 | **Use case version** | V1.0 |
| **Use case name** | Receive message | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **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.** | User click message icon on top tab bar |  | | **2.** |  | Application displays “Chat” screen. | | **3.** | User clicks sender name to read message. |  | | **4.** |  | Application displays chatting screen with sender. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US4.0-E1 – Cannot communicate with API server**  System displays error message.    **Business Rules: N/A** | | | |
|  | | | |

* + - * 1. View post

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-5.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-5.0 | **Use case version** | V1.0 |
| **Use case name** | View post | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | Medium |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User views posts of others user. | | |
| **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 “News Feed” screen, which includes:  Search bar  New post  List of posts | | **3.** | User roll down to read posts. |  |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US5.0-E1 – Cannot communicate with API server**  System displays error message.    **Business Rules: N/A** | | | |
|  | | | |

* + - * 1. React

**\*Follow user**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-6.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-6.0 | **Use case version** | V1.0 |
| **Use case name** | Follow user | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **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 target profile. |  | | **2.** |  | Application displays target profile. | | **3.** | User clicks follow icon in target profile. |  | | **4.** |  | Application displays success message. |   **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** | | | |
|  | | | |

**\*Report post**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-7.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-7.0 | **Use case version** | V1.0 |
| **Use case name** | Report post | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | Medium |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User report posts of others user. | | |
| **Pre-condition** | User has logged into application. | | |
| **Post-condition** |  | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | Do US-UC-5.0 then clicks report icon. |  | | **2.** |  | Application displays list of report types. | | **3.** | User chooses the type then submit. |  | | **4.** |  | Application displays success message. |   **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** | | | |
|  | | | |

* + - * 1. Date

**\*Automation Mode**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-8.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-8.0 | **Use case version** | V1.0 |
| **Use case name** | Automation Mode | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User selects automation mode for dating. | | |
| **Pre-condition** | User has logged into application. | | |
| **Post-condition** |  | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | User clicks dating icon on top tab bar. |  | | **2.** |  | Application displays “Dating” Screen | | **3.** | User chooses pet that want to dating. |  | | **4.** | User clicks “Kết nối” button. |  | | **5.** |  | Application display list of users valid. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US8.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: N/A** | | | |
|  | | | |

**\*Manual Mode**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-9.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-9.0 | **Use case version** | V1.0 |
| **Use case name** | Manual Mode | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User selects manual mode for dating. | | |
| **Pre-condition** | User has logged into application. | | |
| **Post-condition** |  | | |
| **Trigger** |  | | |
| **Main flow:**   |  |  |  | | --- | --- | --- | | **No.** | **Actor events** | **System respond** | | **1.** | User clicks dating icon on top tab bar. |  | | **2.** |  | Application displays “Dating” Screen. | | **3.** | User chooses pet that want to dating. |  | | **4.** | User clicks “Chế độ” then choose “Thủ công”. |  | | **5.** | User fills and select all required information then clicks “Kết nối” button. |  | | **6.** |  | Application display list of users valid. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US9.0-E1 – Cannot communicate with API server**  System displays error message.  **p**  **Business Rules: B07, B08** | | | |
|  | | | |

* + - * 1. Manage user’s profile

**\*View user’s profile**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-10.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-10.0 | **Use case version** | V1.0 |
| **Use case name** | View user’s profile | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User views 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:  Ảnh đại diện  Thông tin cá nhân  Lưu trữ: ảnh, video, …  Các bài viết |   **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** | | | |
|  | | | |

**\*Modify user’s 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** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **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:  Ảnh đại diện  Thông tin cá nhân  Lưu trữ: ảnh, video, …  Các bài viết | | **4.** | User clicks “Chỉnh sửa trang cá nhân”. |  | | **5.** |  | Application navigates to “Modify Profile” screen. | | **6.** | User fills and select all required information then clicks “Chỉnh sửa” button. |  | |  |  | Application displays success message. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US11.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: B07, B08** | | | |
|  | | | |

**\*Upload multimedia**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-12.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-12.0 | **Use case version** | V1.0 |
| **Use case name** | Upload multimedia | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User upload multimedia to own 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 “News Feed” screen, which includes:  Search bar  New post  List of posts | | **3.** | User clicks “Thêm mới”, select images or videos and then click “Xác nhận” |  | | **4.** |  | Application uploads multimedia to system. | | **5.** |  | Application navigates to “News Feed” screen. |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US12.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: B07, B08, B9, B10** | | | |
|  | | | |

* + - * 1. Manage pet’s profile

**\*Add pet’s profile**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-13.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-13.0 | **Use case version** | V1.0 |
| **Use case name** | Add pet’s profile | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **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 “New Pet” screen. | | **5.** | User fills and select all required information then clicks “Thêm mới” button. |  | | **6.** |  | 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** | | | |
|  | | | |

**\*View pet’s profile**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-14.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-14.0 | **Use case version** | V1.0 |
| **Use case name** | View pet’s profile | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User views 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 avatar of pet to view pet’s profile |  | |  |  | Application navigates to “Pet Profile” screen |   **Alternative Flows: N/A**  **Extension Flows: N/A**  **Exception:**  **US14.0-E1 – Cannot communicate with API server**  System displays error message.  **Business Rules: N/A** | | | |
|  | | | |

**\*Modify pet’s profile**

|  |  |  |  |
| --- | --- | --- | --- |
| **USECASE US-UC-14.0** **SPECIFICATION** | | | |
| **Use case ID** | US-UC-14.0 | **Use case version** | V1.0 |
| **Use case name** | View pet’s profile | | |
| **Author** | Le Duc Chinh | | |
| **Date** | 27/07/2019 | **Priority** | High |
| **Primary actor** | User | **Secondary actor** | N/A |
| **Description** | User views 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-14.0 then click “Sửa đổi thông tin” |  | | **2.** |  | Application navigates to “ModifyPet Profile” screen | | **3.** | User fills and select all required information then clicks “Sửa đổi” button. |  | |  |  | 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: N/A** | | | |
|  | | | |

* 1. Non-functional Requirement
     1. Security
* The system must be use Google authentication (login with Google) or Facebook authentication (login with Facebook) for login to the system. So, this is an absolute guarantee of account information security.
* Admin/Mod does not access to database of user.
* Your information can hide in private mode.
  + 1. Availability
* The system active 24/24.
  + 1. Usability
* User interface should be friendly and easy to use.
* Application supports Android and IOS.
* The guideline is clearly and easy to use.
* The system is easy to deploy.
  + 1. Performance
* The mobile application work well in Android and IOS
* The web application work well with Chrome, Firefox.

# **CHAP 4: SOFTWARE DESIGN**



## Purpose

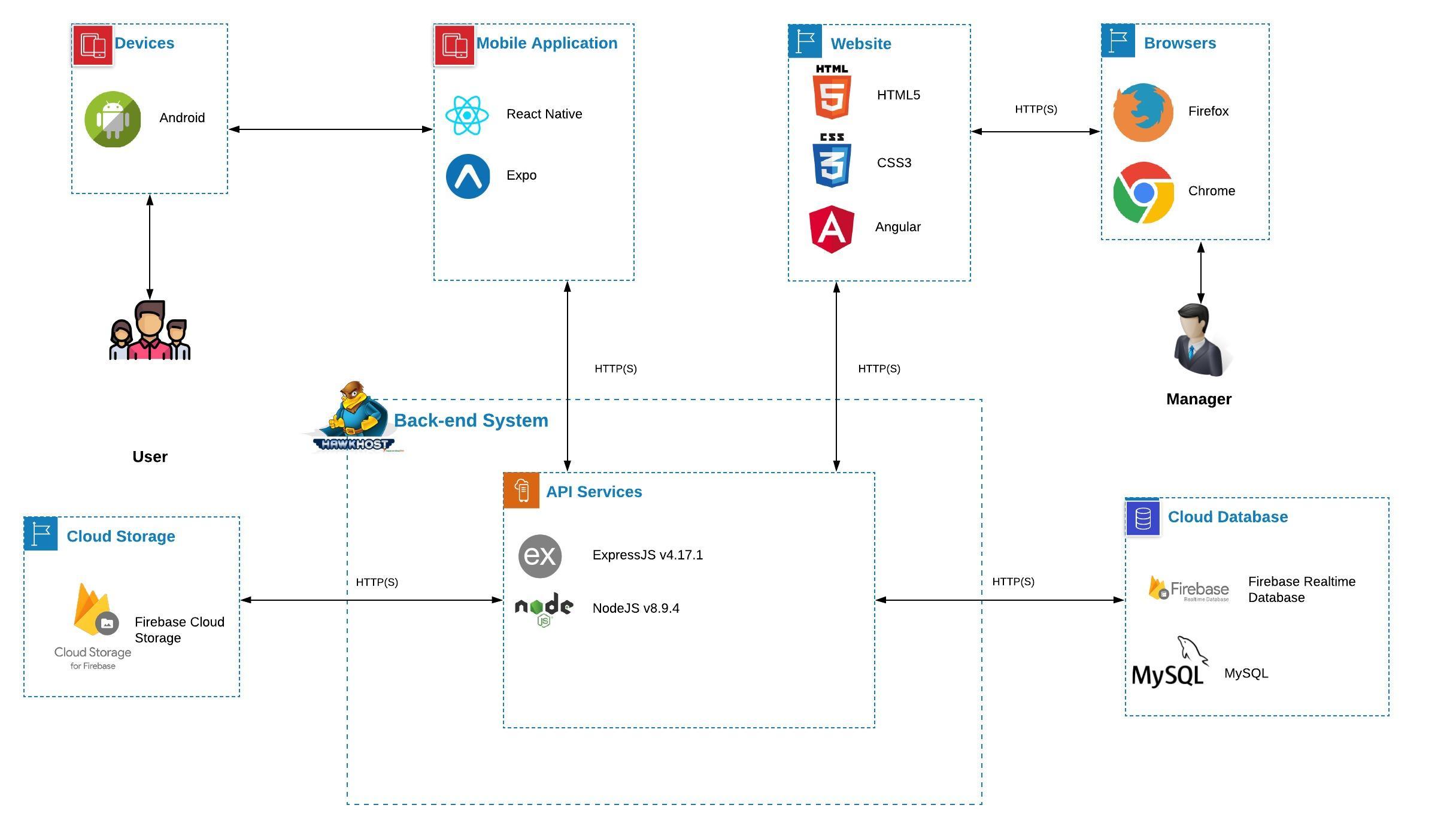
This chapter is to give the developer team an overview of what the system’s architecture  
is, and how they should be implemented. This chapter consists of:

* Overview of system architecture
* Database Design
* Detailed Design

## Overview of System Architecture



### Diagram



*Figure X: PetDating system architecture*

### Component Explanation

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

#### Deployment and Hosting

##### *Hawkhost*



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

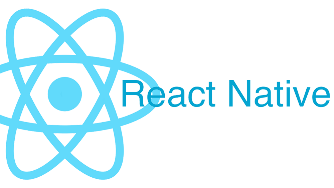
##### *Firebase Cloud Storage*



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

#### Mobile Application components

##### *React Native*



**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**.**

##### *Expo*



**Expo** is a toolchain built around React Native to help you quickly start an app. It provides a set of tools that simplify the development and testing of React Native app and interface and services that are usually available in third-party native React Native components. **We use Expo to build an application that available for Android based on React Native.**

#### Website components

##### *HTML5 and CSS3*



Using **HTML5 and CSS3** provides advantages to businesses that develop and deploy web content and web applications, to create refined and accurate web pages and web systems that **operate across devices, operating systems and web browsers**.

##### *Angular*



**Angular** is a platform for building mobile and desktop web applications. **We use Angular for font-end website.**

#### API Services components

##### *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.**

##### *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.**

#### Database components

##### *Firebase Realtime Database*



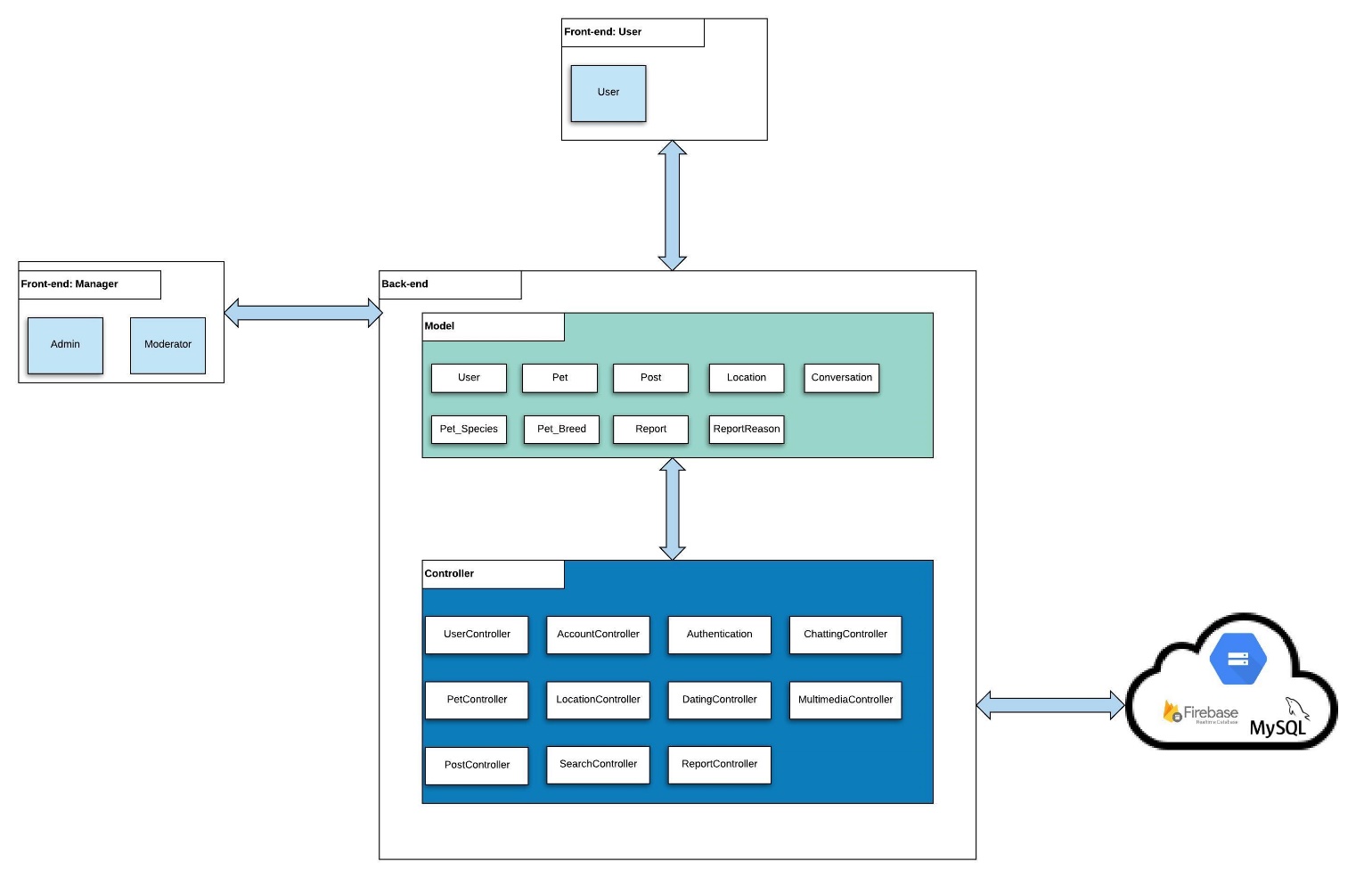
The **Firebase Realtime Database** is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. **We use it to work with chatting function.**

##### *MySQL*



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

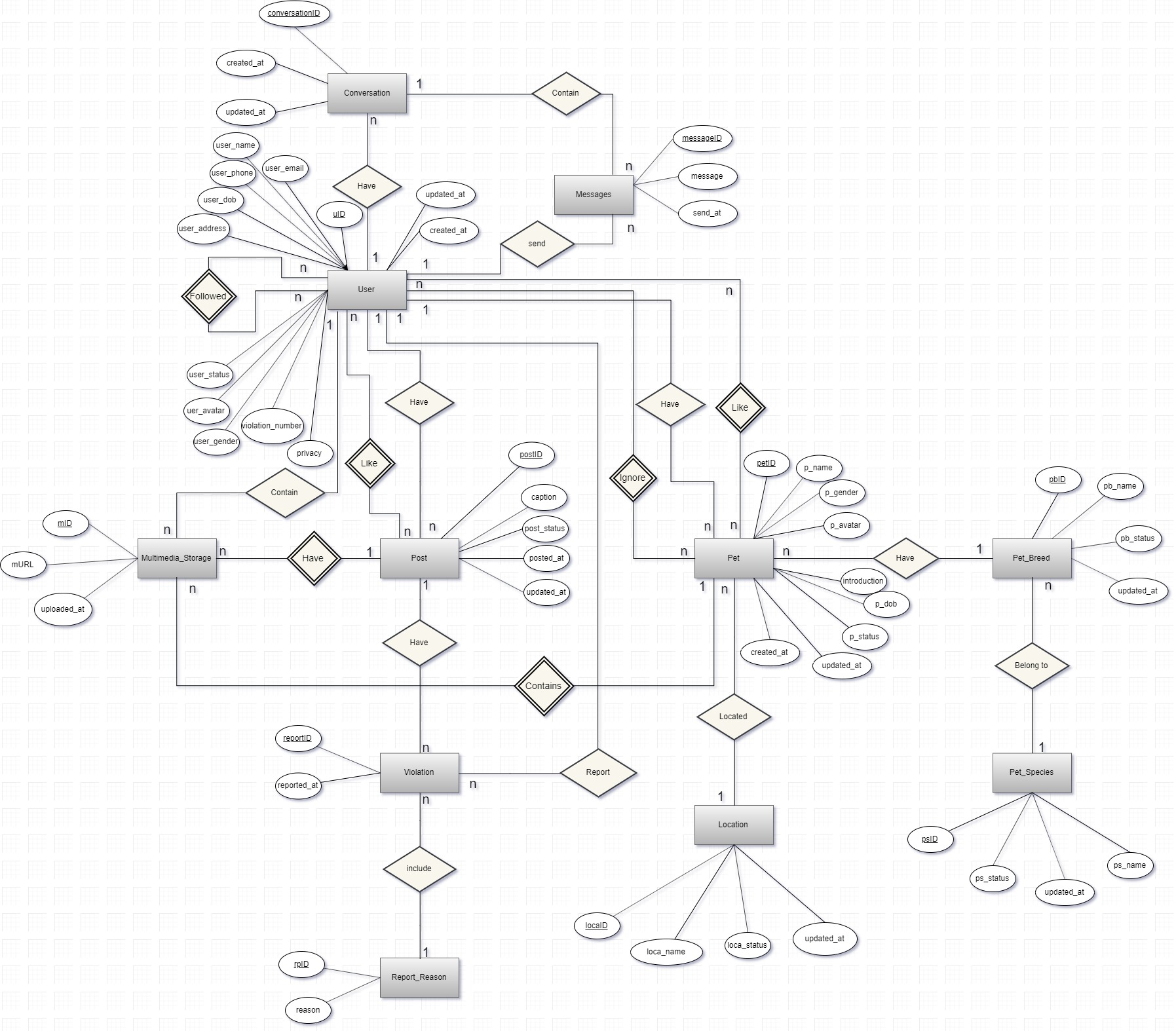
## Architecture Layer Design



*Figure X: PetDating layer design*

## Database Design

### Entity relationship diagram

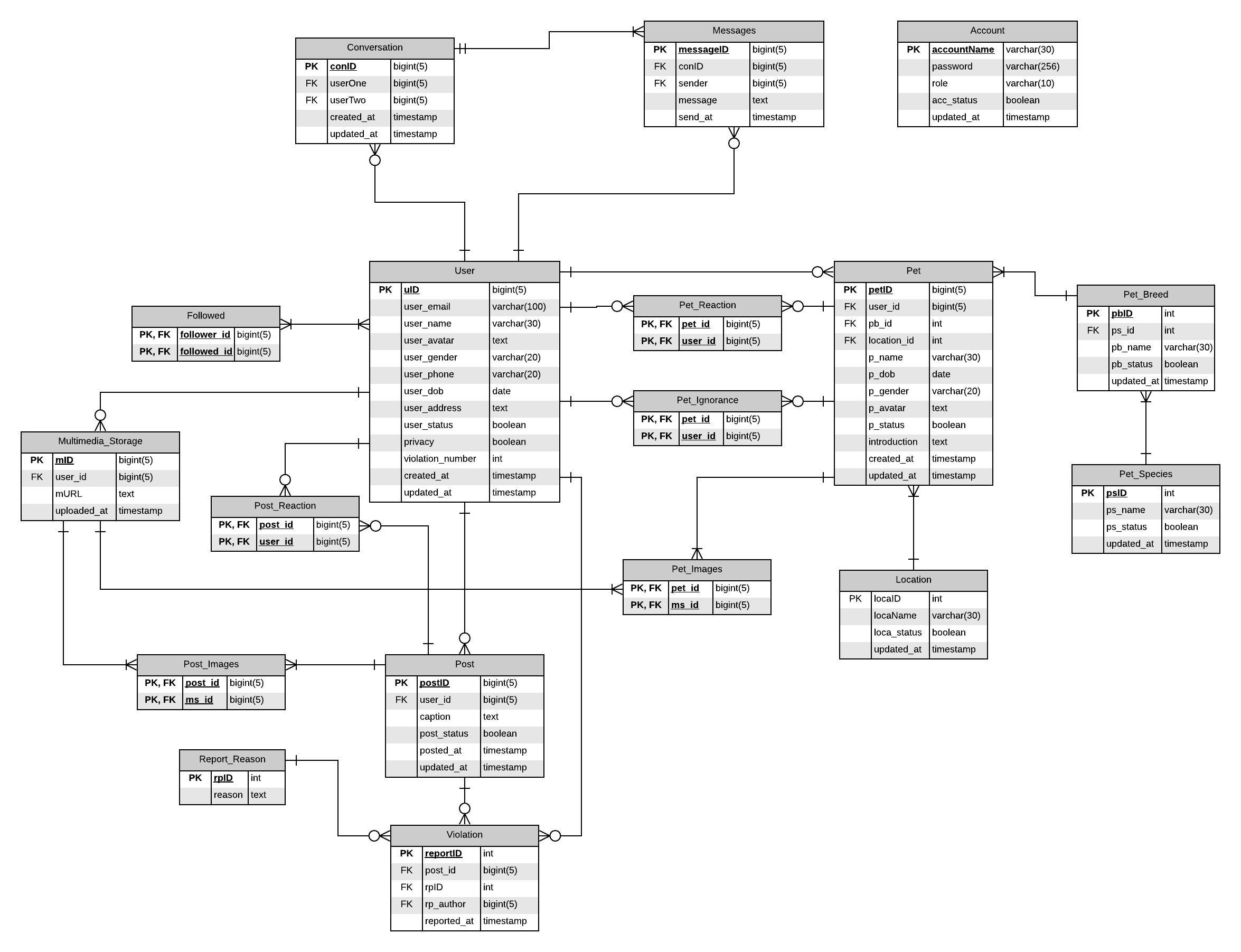


*Figure X: Entity relationship diagram*

|  |  |  |
| --- | --- | --- |
| **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 |
| violation\_number | number of violations (max 2) |
| 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 |
| created\_at | datetime that the pet was created |
| updated\_at | datetime that the pet was updated |
| Pet\_Breed | **pb\_id** | id of pet breed |
| pb\_name | specific name of pet breed |
| pb\_status | status of pet breed |
| updated\_at | datetime that the pet breed was updated |
| Pet\_Species | **ps\_id** | id of pet specie |
| ps\_name | specific name of specie |
| ps\_status | status of specie |
| updated\_at | datetime that the specie was updated |
| Location | **loca\_id** | id of pet location |
| loca\_name | specific name of location |
| loca\_status | status of location |
| updated\_at | datetime that the specie was updated |
| Post | **postID** | id of the post |
| caption | caption of the post |
| post\_status | status of post |
| posted\_at | datetime that the post was publicized |
| updated\_at | datetime that the post was updated |
| Violation | **reportID** | id of the report |
| reported\_at | datetime that the report was issued |
| Report\_Reason | **rpid** | id of the reason |
| reason | content of the reason |
| Multimedia\_Storage | **mID** | id of multimedia |
| mURL | link to multimedia |
| uploaded\_at | datetime that the multimedia was uploaded |
| Conversation | **conversationID** | id of conversation |
| created\_at | datetime that the conversation was created |
| updated\_at | datetime that the last message in conversation was updated |
| Messages | **messageID** | id of message |
| message | content of message |
| send\_at | datetime that the message was send |

*Table X: ERD description*

### Database diagram



*Figure X: Database diagram*

**Database Explanation:**

**\*User:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **uID** | bigint(5) | no | id of user  **PRIMARY KEY** |
| 2 | user\_email | varchar(100) | No | email of user  **UNIQUE** |
| 3 | user\_name | varchar(30) | No | name of user |
| 4 | user\_avatar | text | No | avatar of user |
| 5 | user\_gender | varchar(20) | No | gender of user (Nam; Nữ; Không công khai) |
| 6 | user\_phone | varchar(20) | Yes | phone of user |
| 7 | user\_dob | date | No | birthdate of user |
| 8 | user\_address | text | Yes | address of user |
| 9 | user\_status | boolean | No | status of user  1: Activated - Default  0: inActivated |
| 10 | privacy | boolean | No | privacy of user  1: Công khai - Default  0: Riêng tư |
| 11 | violation\_number | int | No | number violation of user(Default: 0). |
| 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** | bigint(5) | No | id of pet  **Primary key** |
| 2 | *user\_id* | bigint(20) | 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 |
| 4 | *location\_id* | int | No | location of pet  Foreign key refernces to locaID in “Location” table |
| 5 | p\_name | varchar(30) | No | name of pet |
| 6 | p\_dob | date | No | birthdate of pet |
| 7 | p\_gender | varchar(20) | No | gender of pet (Đực; Cái) |
| 8 | p\_avatar | text | No | avatar of pet |
| 9 | p\_status | boolean | No | status of pet  1: Activated - Default  0: inActivated |
| 10 | introduction | text | Yes | introduction about pet |
| 11 | created\_at | timestamp | No | datetime that the pet was created  Default: CURRENT\_TIMESTAMP |
| 12 | 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 |

**\*Pet\_Species:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **psID** | int | No | id of pet’s species  **Primary key** |
| 2 | ps\_name | varchar(30) | No | name of specie |
| 3 | ps\_status | boolean | No | status of specie  1: Activated - Default  0: inActivated |
| 4 | updated\_at | timestamp | No | datetime that the specie 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 | locaName | varchar(30) | No | name of location |
| 3 | loca\_status | boolean | No | status of location  1: Activated - Default  0: inActivated |
| 4 | updated\_at | timestamp | No | time that the location was updated. |

**\*Conversation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **conID** | bigint(5) | No | id of conversation  **Primary key** |
| 2 | *userOne* | bigint(5) | No | id of user one who join in a conversation  Foreign key references to uID in “User” table |
| 3 | *userTwo* | bigint(5) | No | email 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 |

**\*Messages:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **messageID** | bigint(5) | No | id of message  **Primary key** |
| 2 | *conID* | bigint(5) | No | id of conversation  Foreign key refernces to conID in “Conversation” table |
| 3 | *sender* | bigint(5) | No | user\_id of user who send message  Foreign key references to uID in “User” table |
| 4 | message | text | No | content of message |
| 5 | send\_at | timestamp | No | time that the message was send  Default: CURRENT\_TIMESTAMP |

**\*Post:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **postID** | bigint(5) | No | id of post  **Primary key** |
| 2 | *user\_id* | bigint(5) | No | user\_id of user who posted this caption  Foreign key refernces to uID in “User” table |
| 3 | caption | text | No | caption of post |
| 4 | post\_status | boolean | No | status of post  1: Available - Default  0: Delete |
| 5 | posted\_at | timestamp | No | datetime that the post was publicized  Default: CURRENT\_TIMESTAMP |
| 6 | updated\_at | timestamp | No | datetime that the post was updated  Default: CURRENT\_TIMESTAMP  ON UPDATE CURRENT\_TIMESTAMP |

**\*Multimedia\_Storage:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **mID** | bigint(5) | No | id of multimedia  **Primary key** |
| 2 | *user\_id* | bigint(5) | No | user\_id of user who has multimedia  Foreign key refernces to uID in “User” table |
| 3 | mURL | int | No | number like of post |
| 4 | uploaded\_at | timestamp | No | datetime that the multimedia was uploaded  Default: CURRENT\_TIMESTAMP |

**\*Violation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **reportID** | int | No | id of violation  **Primary key** |
| 2 | *post\_id* | bigint(5) | No | id of post that the user want to report about violation  Foreign key references to postID in “Post” table |
| 3 | *rpID* | int | No | if of reason that the user reported  Foreign key references to rpID in “ReportReason” table |
| 4 | *rp\_author* | bigint(5) | No | user\_id of user who reported  Foreign key refernces to uID in “User” table |
| 5 | reported\_at | timestamp | No | time that the post was reported  Default: CURRENT\_TIMESTAMP |

**\*Report\_Reason:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **rpID** | int | No | id of report’s reason  **Primary key** |
| 2 | reason | text | No | contain of reason |

**\*Followed:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***follower\_id*** | bigint(5) | No | id of follower  **Primary key**  Foreign key refernces to uID in “User” table |
| 2 | ***followed\_id*** | bigint(5) | No | id of user who has followed  **Primary key**  Foreign key refernces to uID in “User” table |

**\*Post\_Reaction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***post\_id*** | bigint(5) | No | id of post  **Primary key**  Foreign key references to postID in “Post” table |
| 2 | ***user\_id*** | bigint(5) | No | id of user who has reacted about post  **Primary key**  Foreign key refernces to uID in “User” table |

**\*Post\_Images:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***post\_id*** | bigint(5) | No | id of post  **Primary key**  Foreign key references to postID in “Post” table |
| 2 | ***ms\_id*** | bigint(5) | No | id of image that contained in this post  **Primary key**  Foreign key refernces to mID in “Multimedia\_Storage” table |

**\*Pet\_Images:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***pet\_id*** | bigint(5) | No | id of pet  **Primary key**  Foreign key references to petID in “pet” table |
| 2 | ***ms\_id*** | bigint(5) | No | id of image that contained in this post  **Primary key**  Foreign key refernces to mID in “Multimedia\_Storage” table |

**\*Pet\_Reaction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***pet\_id*** | bigint(5) | No | id of pet  **Primary key**  Foreign key references to petID in “Pet” table |
| 2 | ***user\_id*** | bigint(5) | No | id of user who has reacted about pet  **Primary key**  Foreign key refernces to uID in “User” table |

**\*Pet\_Ignoreance:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | ***pet\_id*** | bigint(5) | No | id of pet  **Primary key**  Foreign key references to petID in “Pet” table |
| 2 | ***user\_id*** | bigint(5) | No | id of user who has ignored about pet  **Primary key**  Foreign key refernces to uID in “User” table |

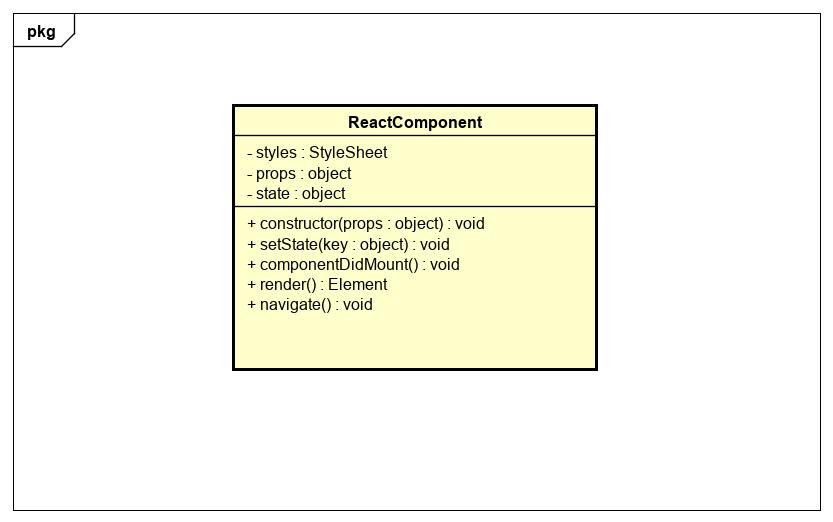
**\*Account:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 1 | **accountName** | varchar(30) | No | name of manager’s account  **Primary key** |
| 2 | password | varchar(256) | No | password of account |
| 3 | role | varchar(10) | No | Role of manager : Admin or Moderator |
| 4 | acc\_status | boolean | No | status of account  1: Available - Default  0: InAvailable |
| 5 | updated\_time | timestamp | No | time that the account was updated  Default: CURRENT\_TIMESTAMP  ON UPDATE CURRENT\_TIMESTAMP |

## Detailed Design

### ReactComponent

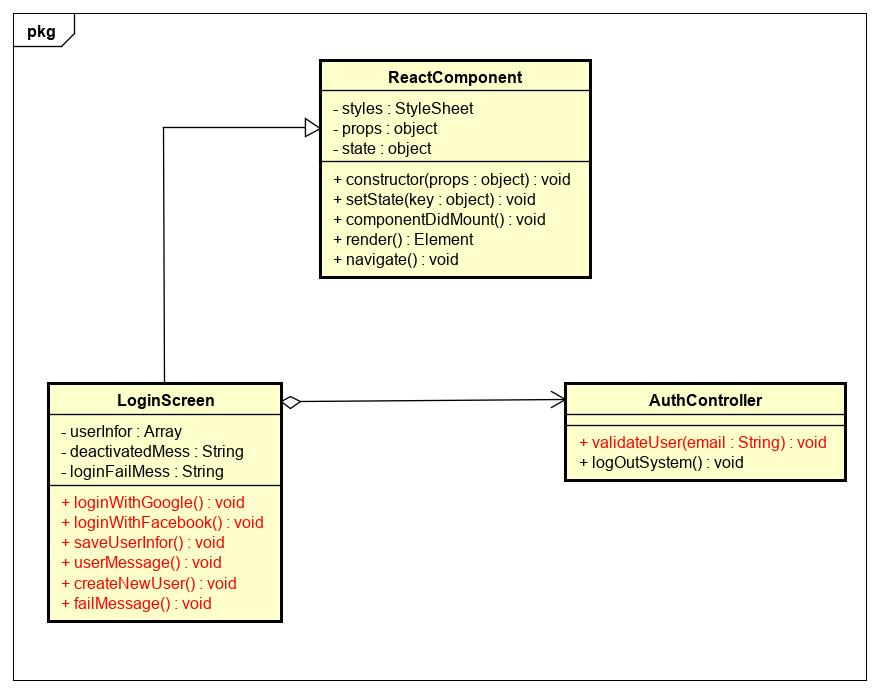
**\*ReactComponent class**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | ReactComponent | | | **Type** | Class |
| **Description** | A class that represents a React component to be rendered on React Native | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| styles | StyleSheet | private | Includes style of this class | | |
| props | object | private | Properties of the component | | |
| state | object | private | State of the component | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| constructor(props : object) | void | public | Create a React Native component | | |
| setState(key : object) | void | public | Update the state of the component | | |
| componentDidMount() | void | public | Handle the event when the component is loaded firstly. | | |
| render() | Element | public | Render the components on the application | | |

### Login

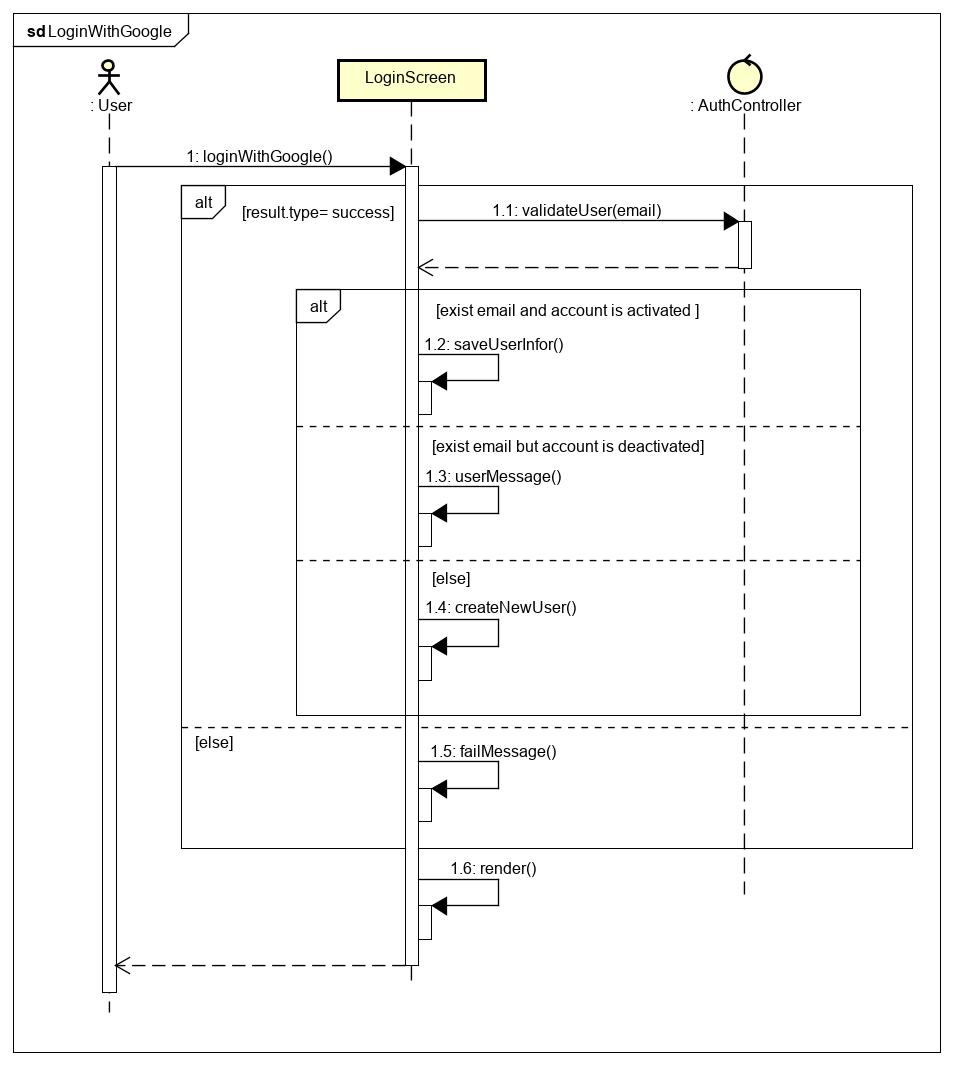
**\*Login class diagram**



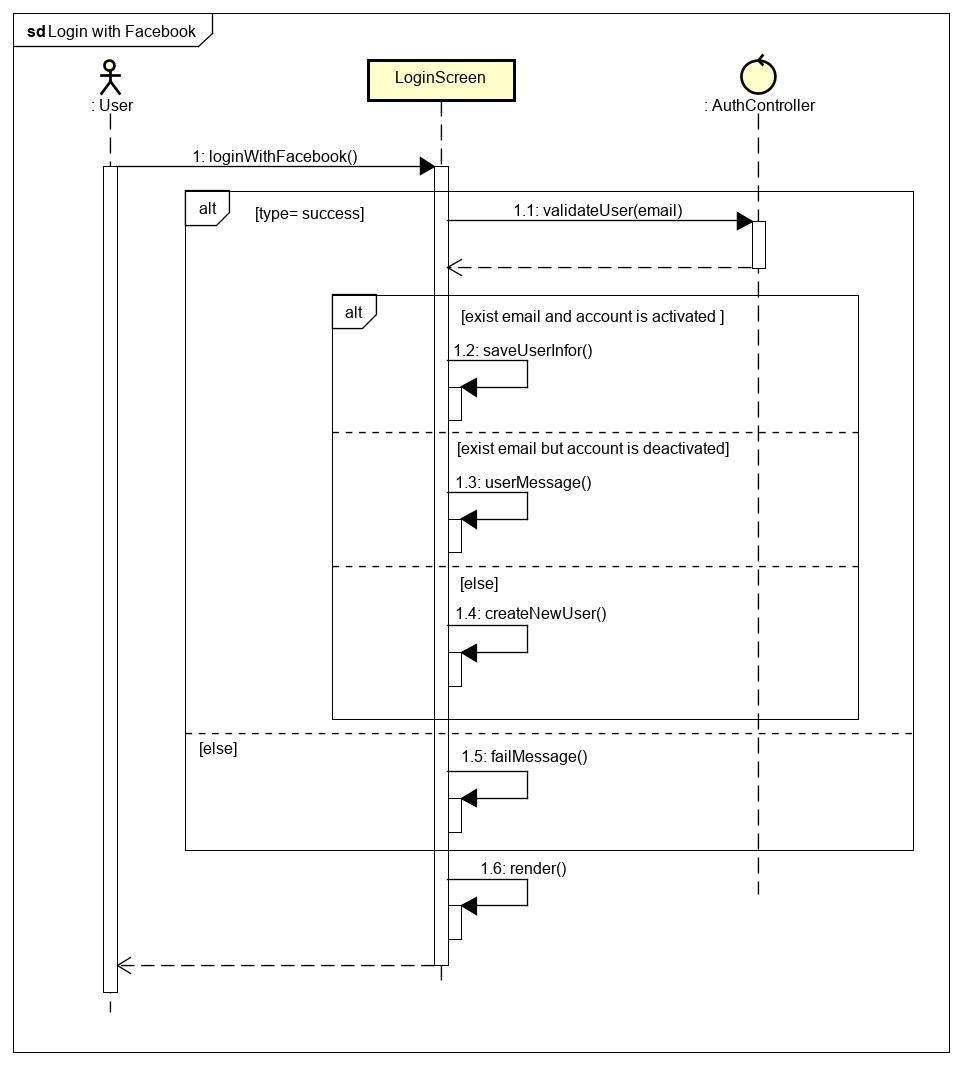
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | AuthController | | | **Type** | Class |
| **Description** | Controller class to control authentication. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **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** | | |
| userInfo | Array | private | Store user informration after logged in successfully | | |
| deactivatedMess | String | private | The message of deactivated user | | |
| loginFailMess | String | private | The message while user logged fail | | |
| **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 | | |
| failMessage() | void | public | Display fail message | | |

**\*loginWithGoogle sequence diagram**

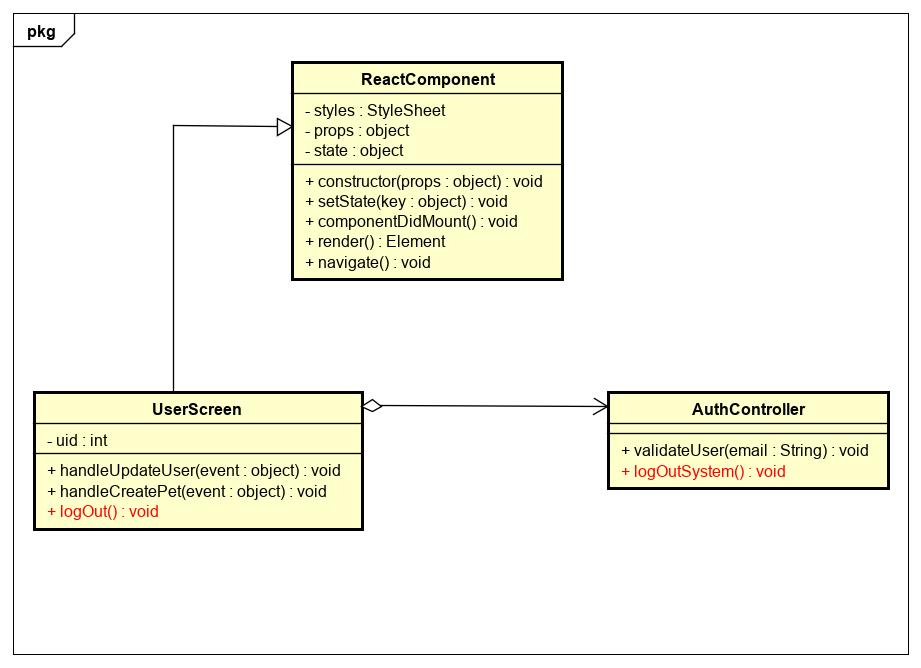


**\*loginWithFacebook sequence diagram**



### Logout

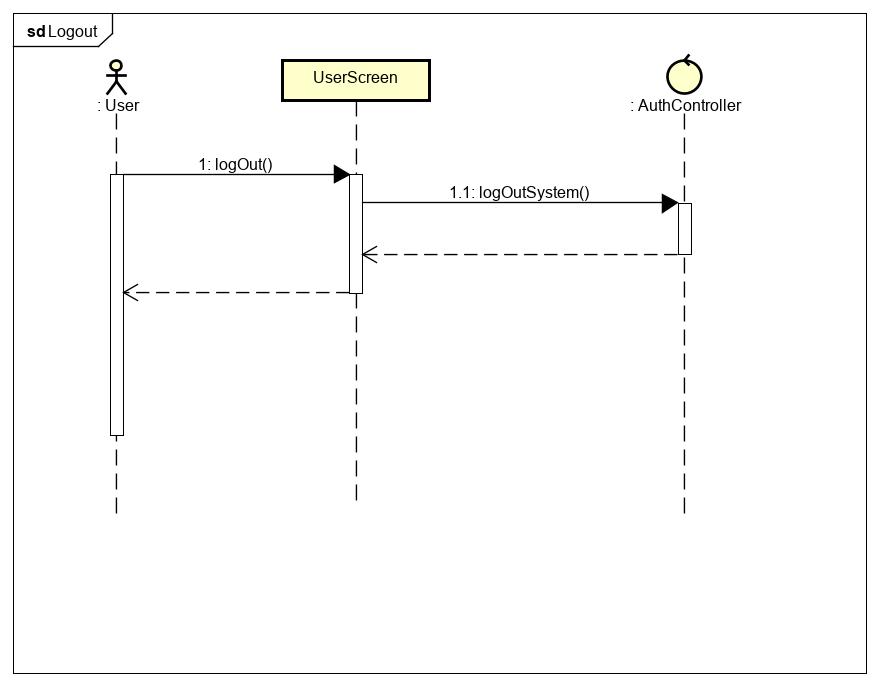
**\*Logout class diagram**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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. | | |
|  |  |  |  | | |

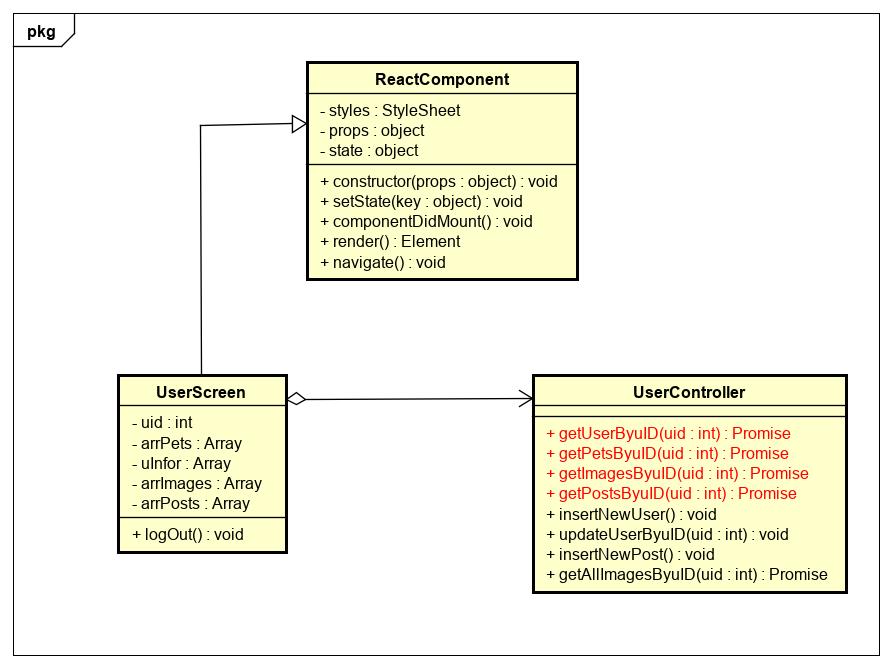
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserScreen | | | **Type** | Class |
| **Description** | A class that displays user screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| logOut() | void | public | Handle logout from application. | | |
|  |  |  |  | | |

**\*Logout sequence diagram**



### View UserScreen

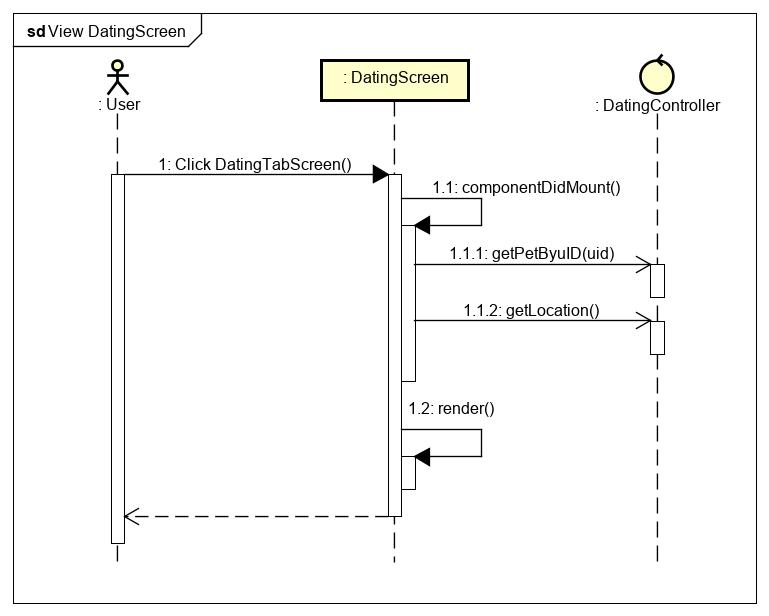
**\*View UserScreen class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserController | | | **Type** | Class |
| **Description** | Controller class to control user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getUserByuID(uid) | Promise | public | Get information of user | | |
| getPetsByuID(uid) | Promise | public | Get Pets Information that owned by user | | |
| getImagesByuID(uid) | Promise | public | Get images that user uploaded and limit 6 | | |
| getPostsByuID(uid) | Promise | public | Get all posts that user created | | |

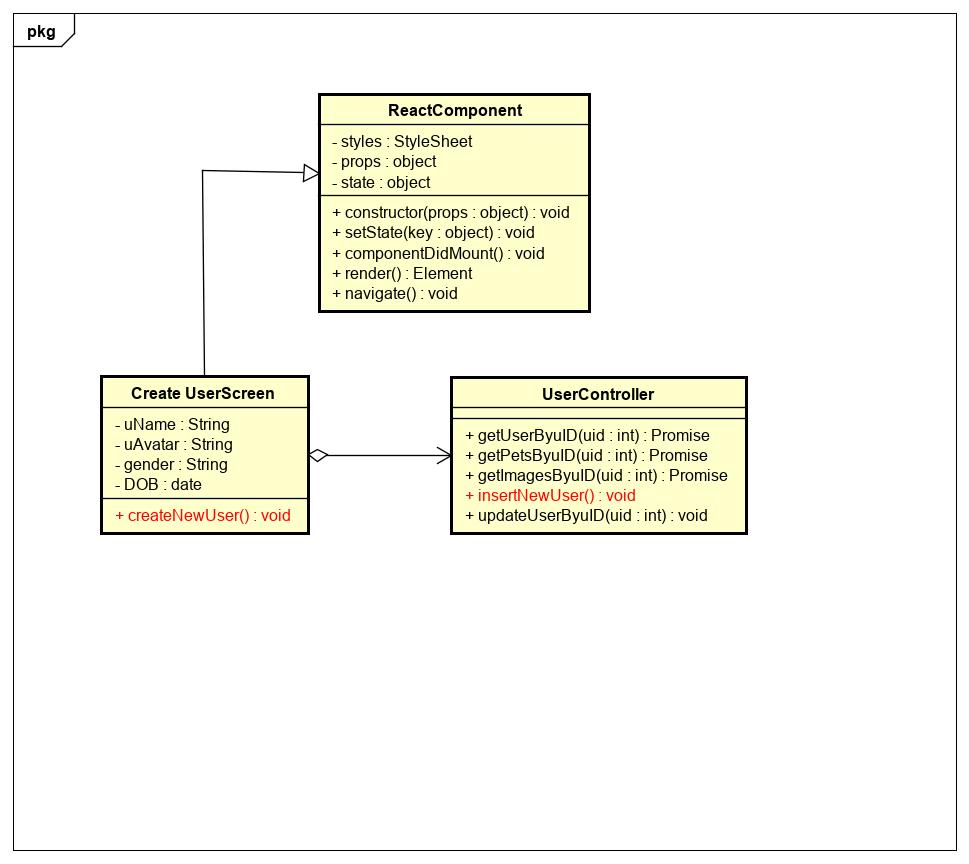
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserScreen | | | **Type** | Class |
| **Description** | A class that displays user screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| uid | int | private | Id of user who logged in application | | |
| arrPets | Array | private | Array of pets that owned by user | | |
| uInfor | Array | private | Information of user | | |
| arrImages | Array | private | Array of images that uploaded by user | | |
| arrPosts | Array | private | Array of posts that created by user | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |

**\*View UserScreen sequence diagram**



### Create User

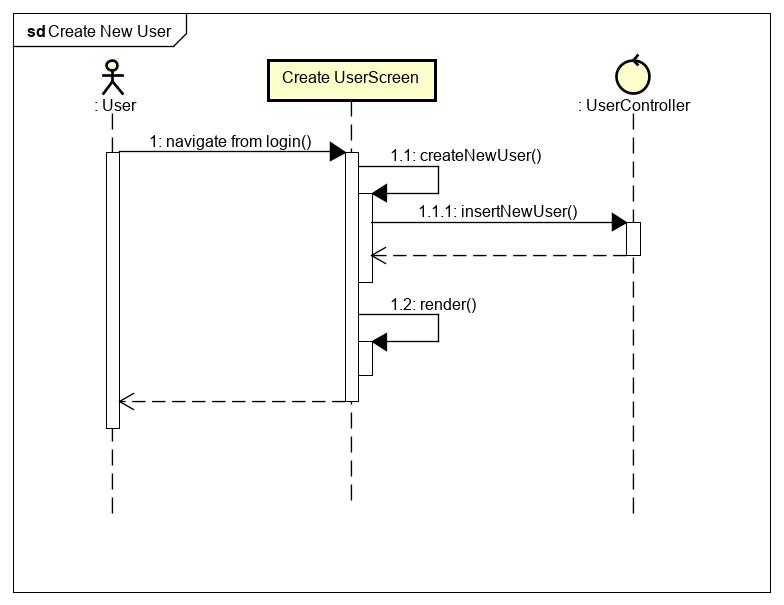
**\*Create New User class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserController | | | **Type** | Class |
| **Description** | Controller class to control user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| insertNewUser | void | public | Insert new user to database | | |
|  |  |  |  | | |

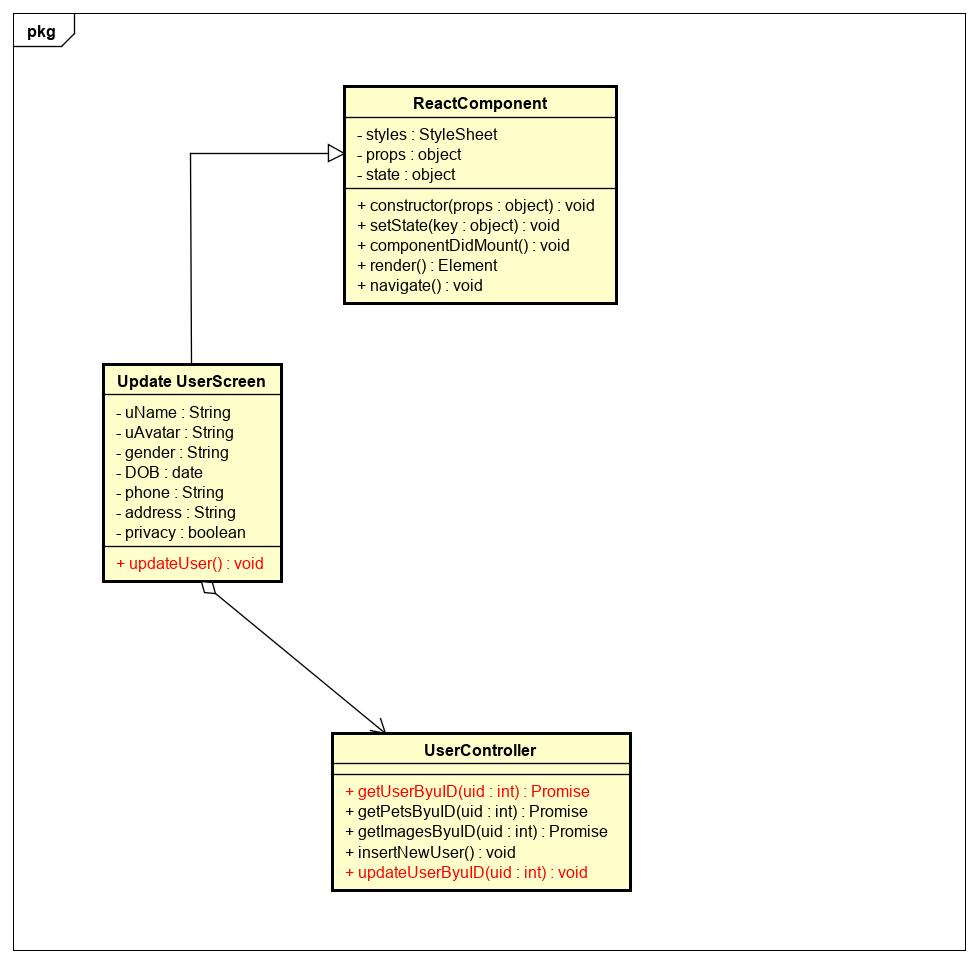
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | Create UserScreen | | | **Type** | Class |
| **Description** | A class that displays create user screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| uName | String | private | Nickname of user | | |
| uAvatar | String | private | Avatar of user | | |
| gender | String | private | Gender of user | | |
| DOB | date | private | Birthdate of user | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| createNewUser() | void | public | Handle create new user | | |
|  |  |  |  | | |

**\*Create New User sequence diagram**



### Update User

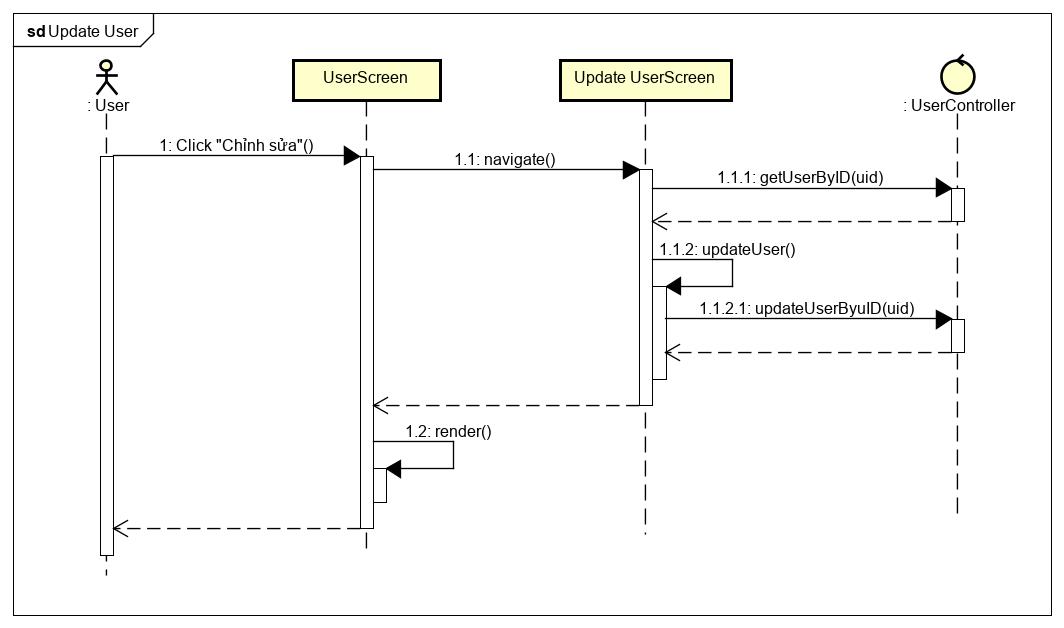
**\*Update User class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserController | | | **Type** | Class |
| **Description** | Controller class to control user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| updateUserByuID(uid) | void | public | Update user in database by user id | | |
| getUserByuID(uid) | void | puiblic | Get information of user by user id | | |
|  |  |  |  | | |

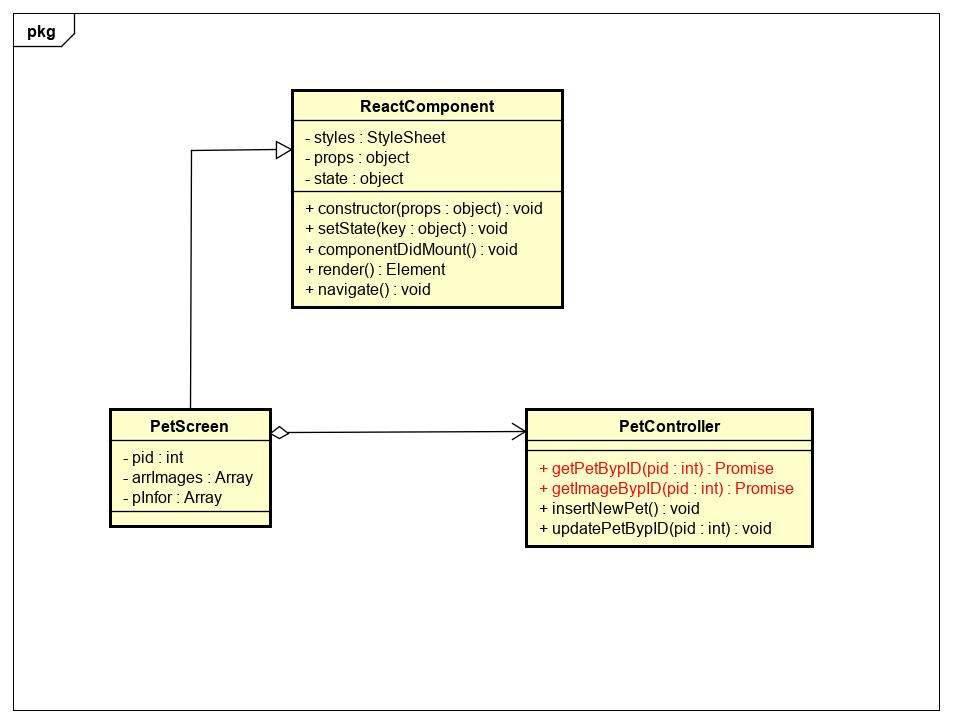
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | Update UserScreen | | | **Type** | Class |
| **Description** | A class that displays update user screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| uName | String | private | Nickname of user | | |
| uAvatar | String | private | Avatar of user | | |
| gender | String | private | Gender of user | | |
| DOB | date | private | Birthdate of user | | |
| phone | String | private | Phone number of user | | |
| address | String | private | Address of user | | |
| privacy | boolean | private | Privacy of user | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| updateUser() | void | public | Handle update user | | |
|  |  |  |  | | |

**\*Update User sequence diagram**



### View Pet

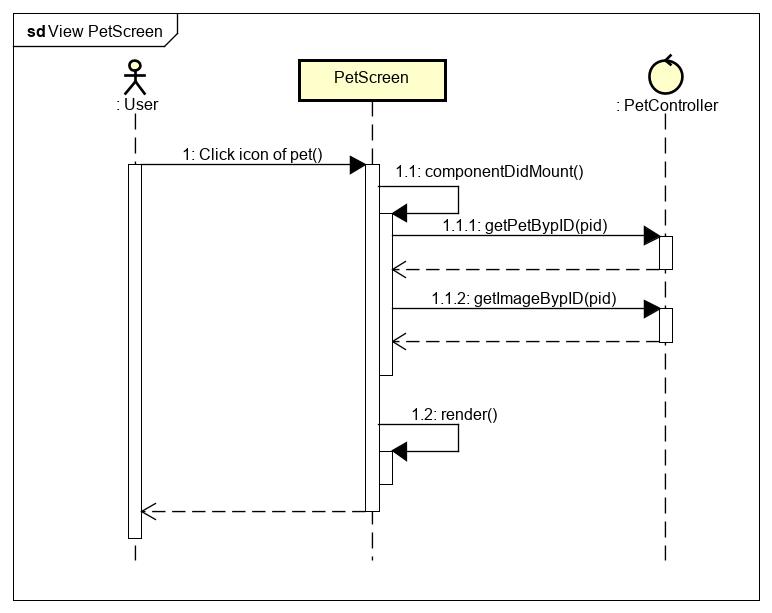
**\*View Pet class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PetController | | | **Type** | Class |
| **Description** | Controller class to control pet. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getPetBypID(pid) | Promise | public | Get information of pet by pet id | | |
| getImageBypID(pid) | Promise | puiblic | Get all featured pictures of pet | | |

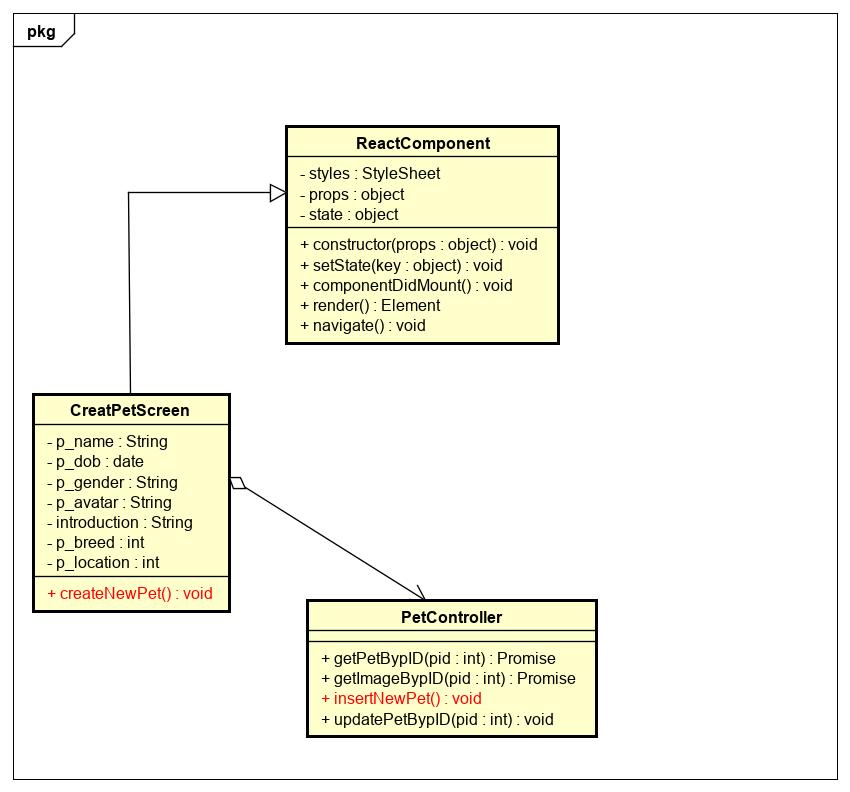
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PetScreen | | | **Type** | Class |
| **Description** | A class that displays pet screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| pid | int | private | Id of pet | | |
| arrImages | Array | private | Array of fetured pictures of pet | | |
| pInfo | Array | private | Information of pet | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |

**\*View Pet sequence diagram**



### Create Pet

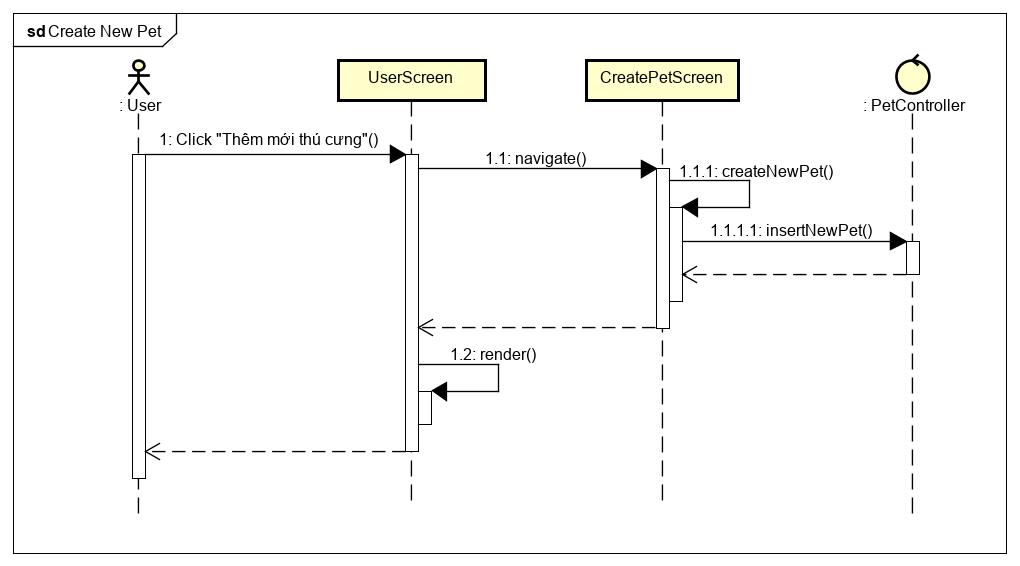
**\*Create Pet class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PetController | | | **Type** | Class |
| **Description** | Controller class to control pet. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| insertNewPet() | void | public | Insert pet to database | | |
|  |  |  |  | | |

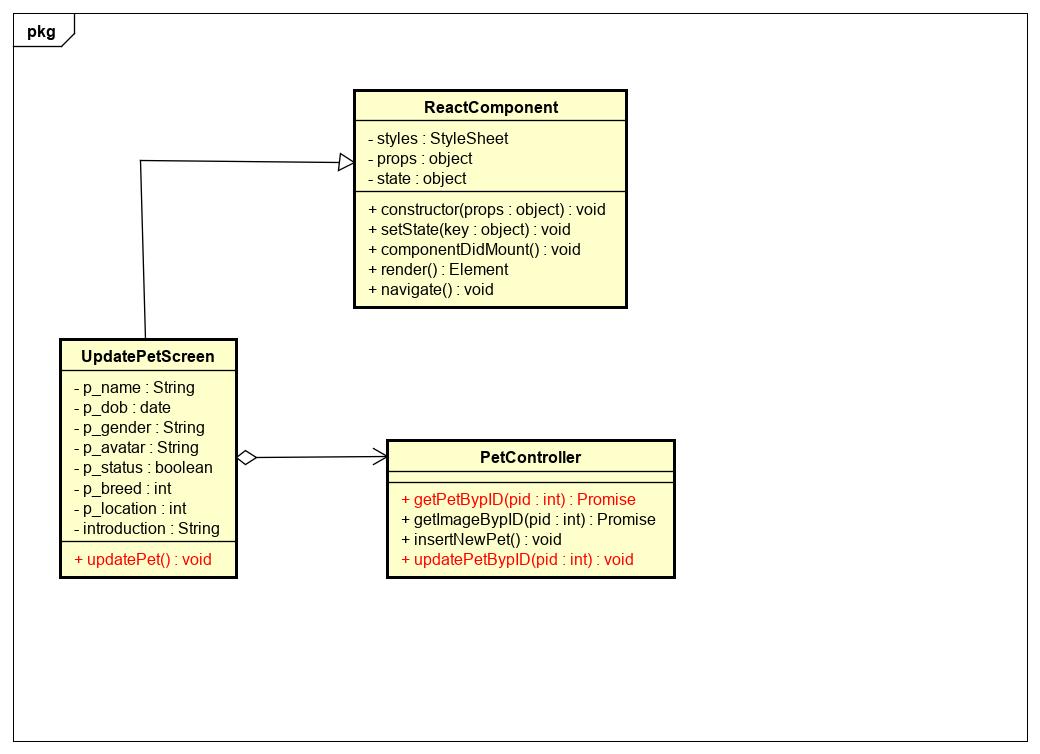
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | CreatePetScreen | | | **Type** | Class |
| **Description** | A class that displays create pet screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| p\_name | String | private | Name of pet | | |
| p\_dob | date | private | Birthdate of pet | | |
| p\_gender | String | private | Gender of pet | | |
| p\_avatar | String | private | Avatar of pet | | |
| p\_breed | int | private | Breed of pet | | |
| p\_location | int | private | Location of pet | | |
| introduction | String | private | Introduction about pet | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| createNewPet() | void | public | Handle create pet | | |

**\*Create Pet sequence diagram**



### Update Pet

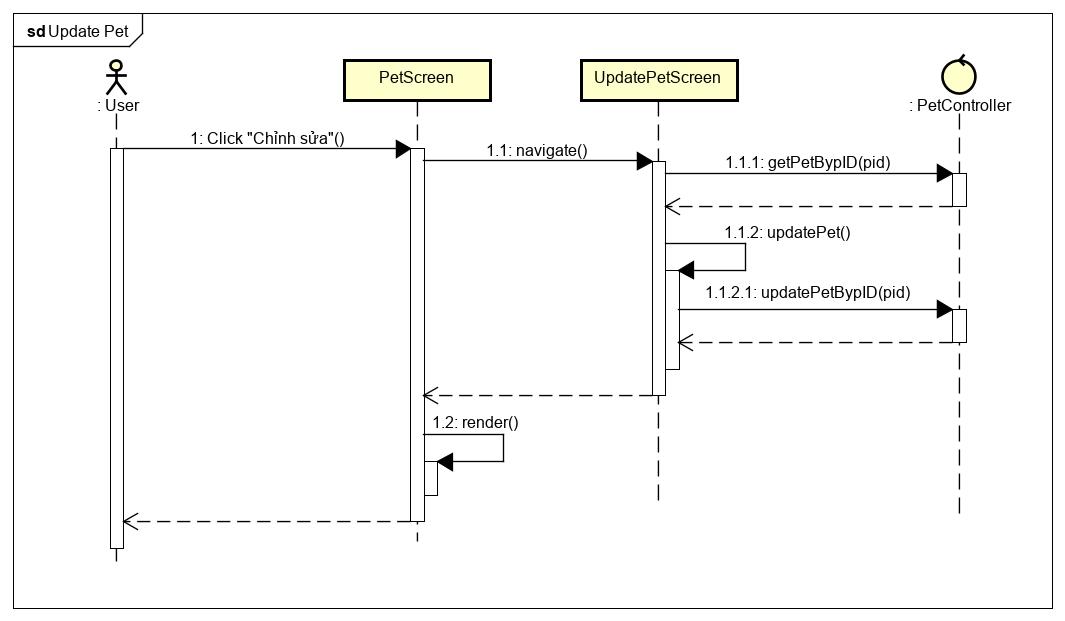
**\*Update Pet class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PetController | | | **Type** | Class |
| **Description** | Controller class to control pet. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| updatePetBypID(pid) | void | public | Update pet in database | | |
| getPetBypID(pid) | Promise | public | Get information of pet | | |

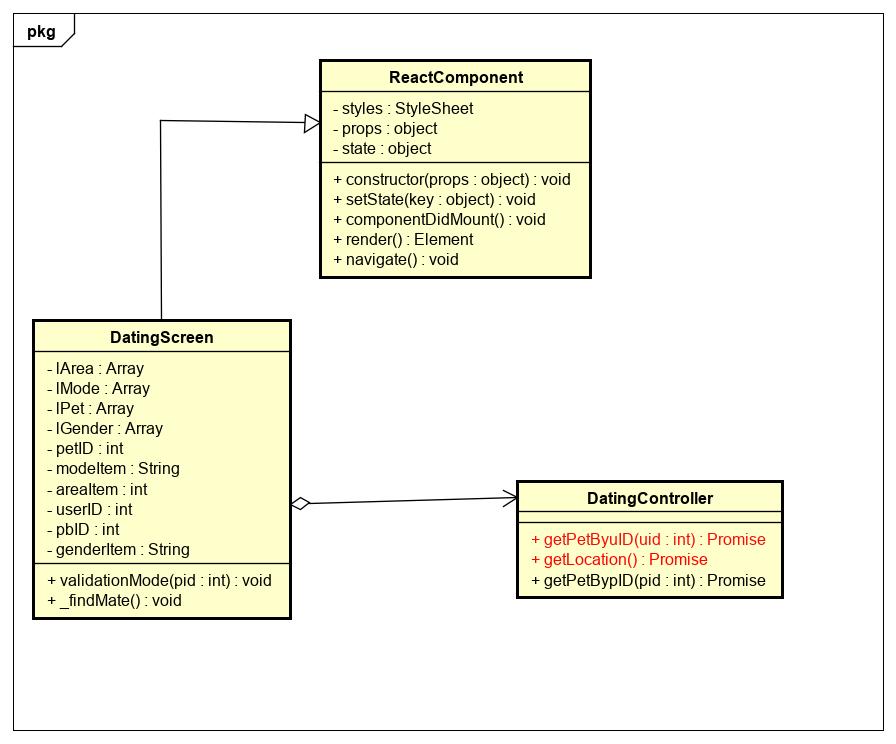
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UpdatePetScreen | | | **Type** | Class |
| **Description** | A class that displays update pet screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| p\_name | String | private | Name of pet | | |
| p\_dob | date | private | Birthdate of pet | | |
| p\_gender | String | private | Gender of pet | | |
| p\_avatar | String | private | Avatar of pet | | |
| p\_status | boolean | private | Status of pet | | |
| p\_breed | int | private | Breed of pet | | |
| p\_location | int | private | Location of pet | | |
| introduction | String | private | Introduction about pet | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| updatePet() | void | public | Handle update pet | | |
|  |  |  |  | | |

**\*Update Pet sequence diagram**



### View DatingScreen

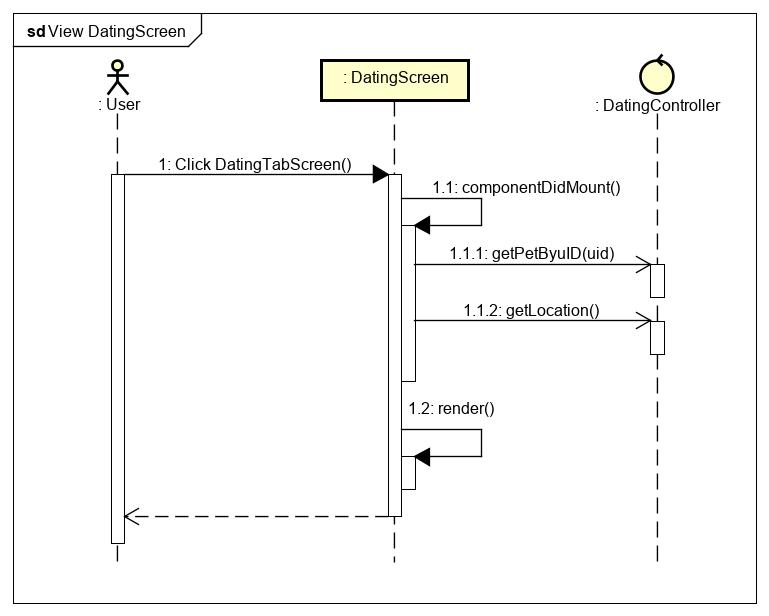
**\*View DatingScreen class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | DatingController | | | **Type** | Class |
| **Description** | Controller class to control dating. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getPetByuID(uid) | Promise | public | Get information of pets that owned by user | | |
| getLocation() | Promise | public | Get locations from database | | |

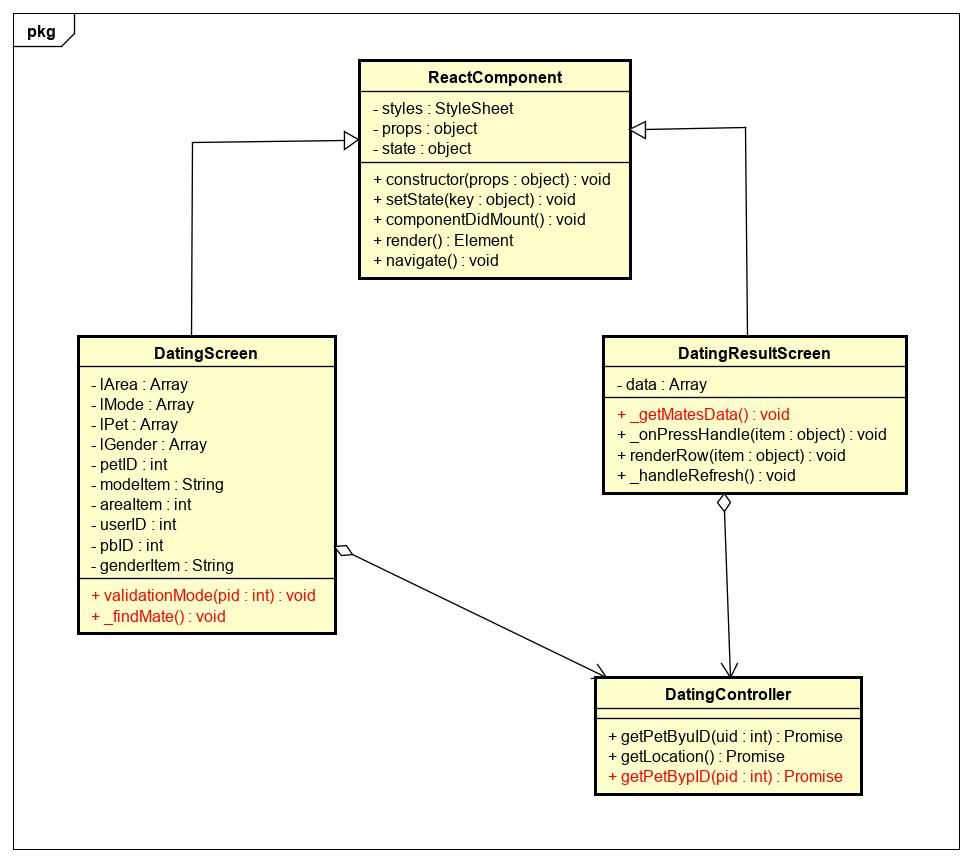
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | DatingScreen | | | **Type** | Class |
| **Description** | A class that displays dating screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| lArea | Array | private | Array of locations | | |
| lMode | Array | private | Array of mode | | |
| lPet | Array | private | Array of pets | | |
| lGender | Array | private | Array of gender | | |
| petID | int | private | Id of pet that user picked | | |
| modeItem | String | private | Name of mode that user picked | | |
| areaItem | int | private | Name of location that user picked | | |
| userID | int | private | Id of user | | |
| genderItem | String | private | Gender of pet that user picked | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
|  |  |  |  | | |

**\*View DatingScreen sequence diagram**



### Dating

**\*Dating class diagram**

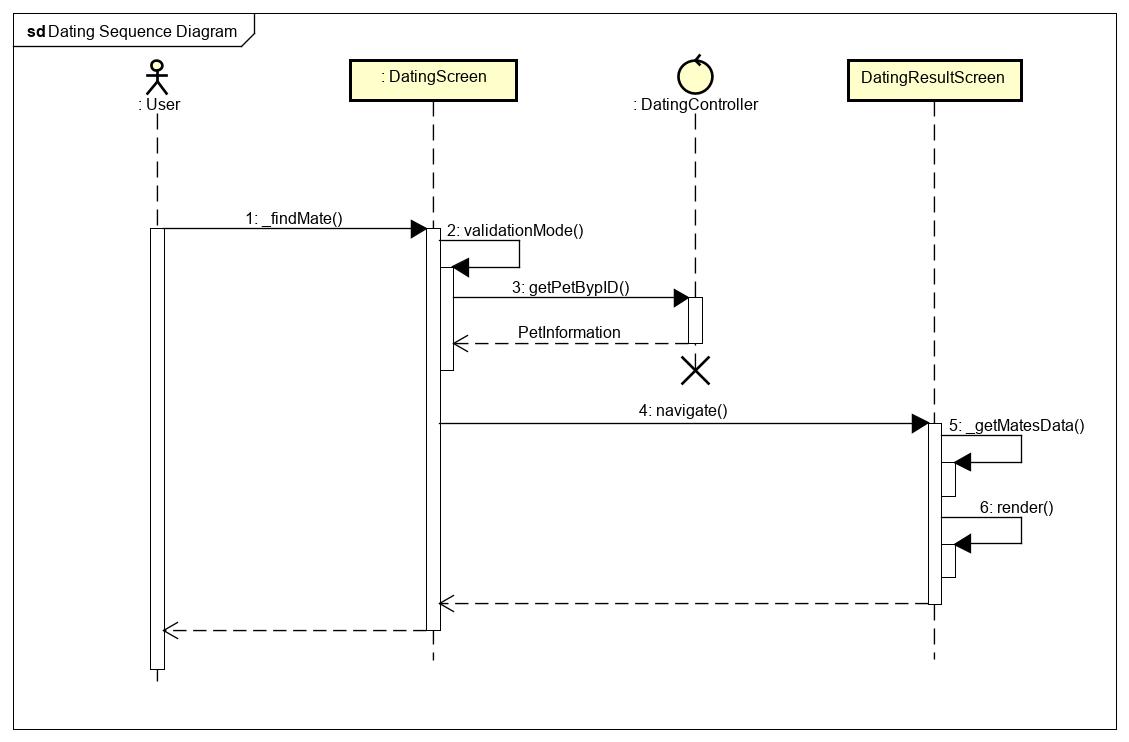


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | DatingController | | | **Type** | Class |
| **Description** | Controller class to control dating. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getPetBypID(pid) | Promise | public | Get information of pet by pet id | | |
|  |  |  |  | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | DatingScreen | | | **Type** | Class |
| **Description** | A class that displays dating screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| validationMode(pid) | void | public | Validate mode of dating | | |
| \_findMate | void | public | Handle mate for pet | | |

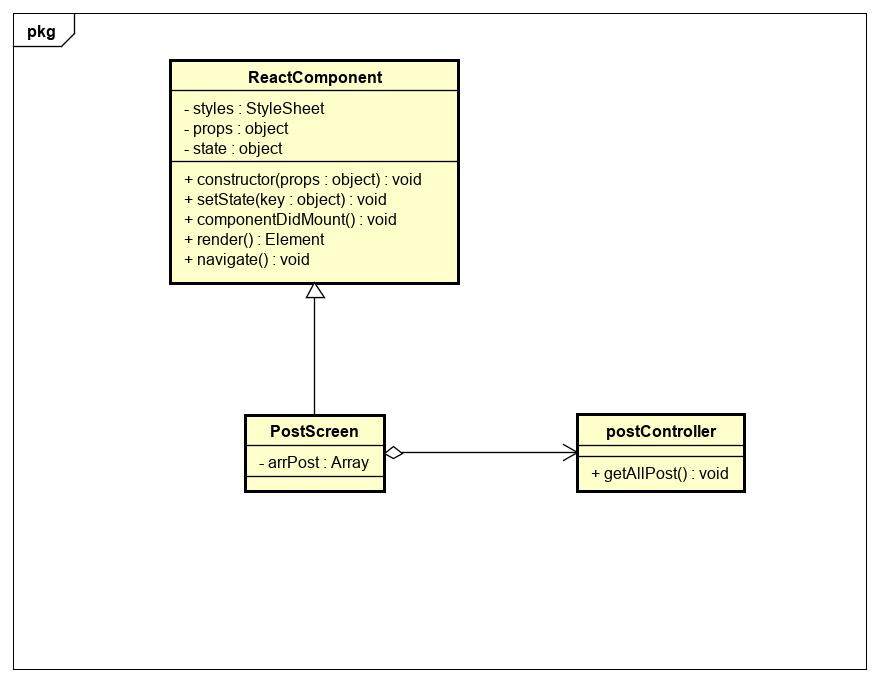
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | DatingResultScreen | | | **Type** | Class |
| **Description** | A class that displays datingresult screen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| data | Array | private | Array of pets that valid information picked | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| \_getMatesData() | void | public | Get mates for pet from last information | | |
|  |  |  |  | | |

**\*Dating sequence diagram**



### View PostScreen

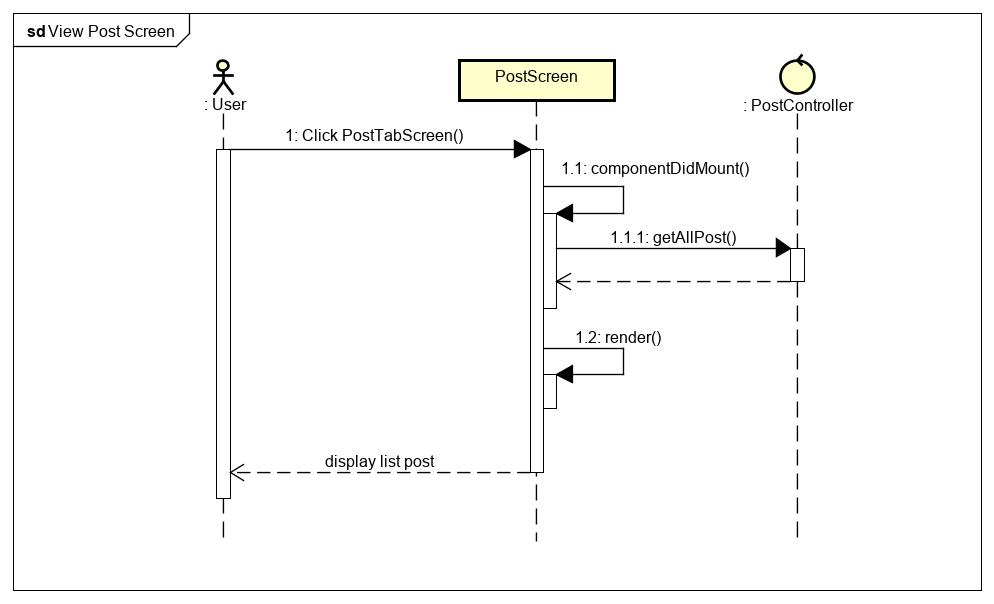
**\*View PostScreen class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PostScreen | | | **Type** | Class |
| **Description** | A class that displays list post screen | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| arrPost | Array | private | Array of post that contains all posts. | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |

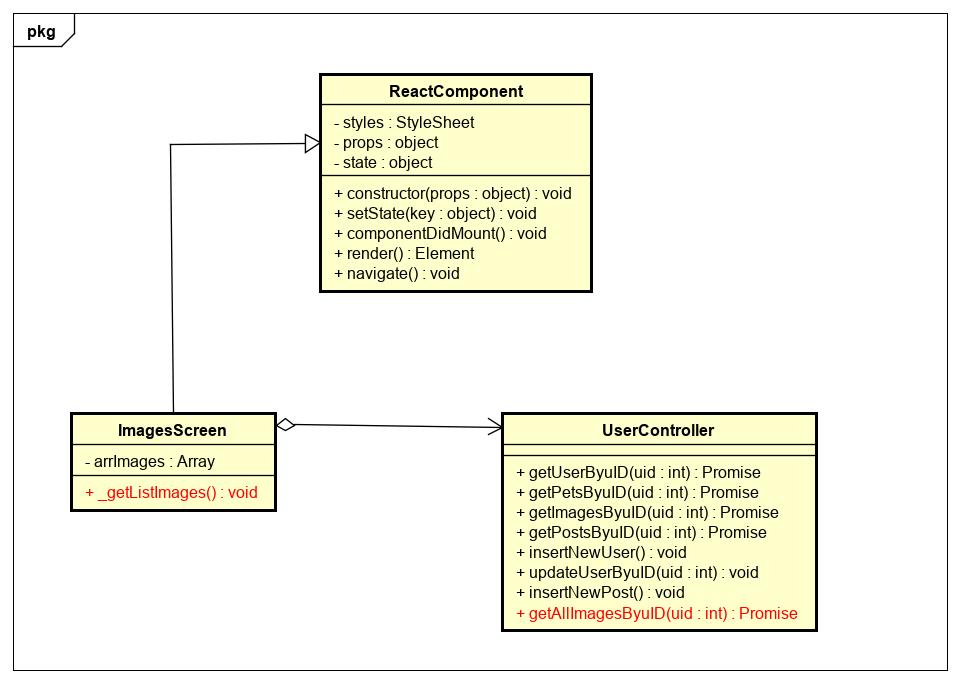
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | PostController | | | **Type** | Class |
| **Description** | A class that control PostScreen | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getAllPost() | void | public | Get all posts | | |

**\*View PostScreen sequence diagram**



### View ListImages

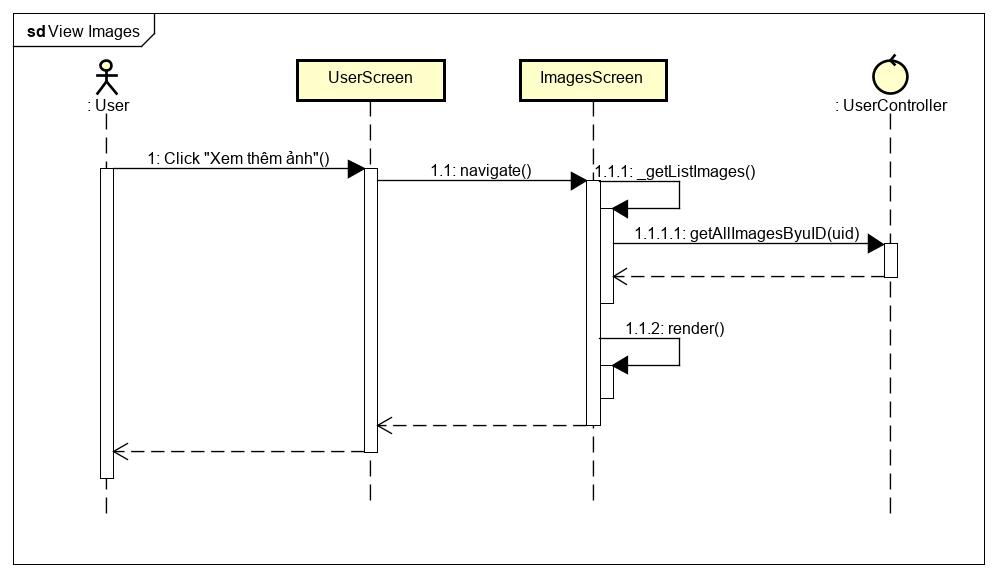
**\*View ListImages class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserController | | | **Type** | Class |
| **Description** | Controller class to control user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| getAllImagesByuID(uid) | Promise | public | Get all images that uploaded by user | | |
|  |  |  |  | | |

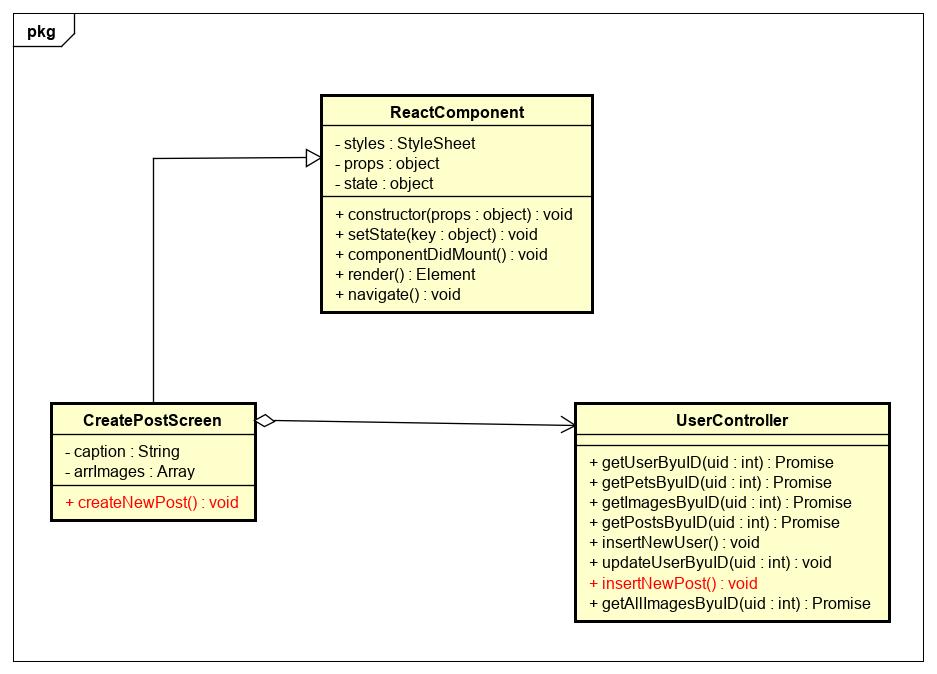
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | ImagesScreen | | | **Type** | Class |
| **Description** | A class displays list images that uploaded by user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| arrImages | Array | private | Array of images that uploaded by user | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| \_getListImages() | void | public | Handle get images that uploaded by user | | |
|  |  |  |  | | |

**\*View ListImages sequence diagram**



### Create NewPost

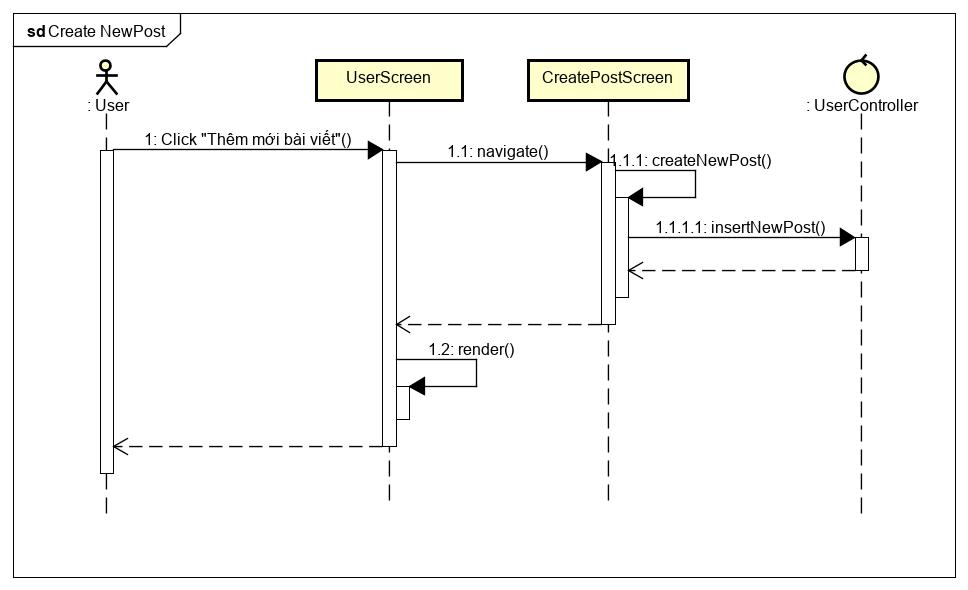
**\*Create NewPost class diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | UserController | | | **Type** | Class |
| **Description** | Controller class to control user. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
|  |  |  |  | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| insertNewPost() | void | public | Insert new post to database | | |
|  |  |  |  | | |

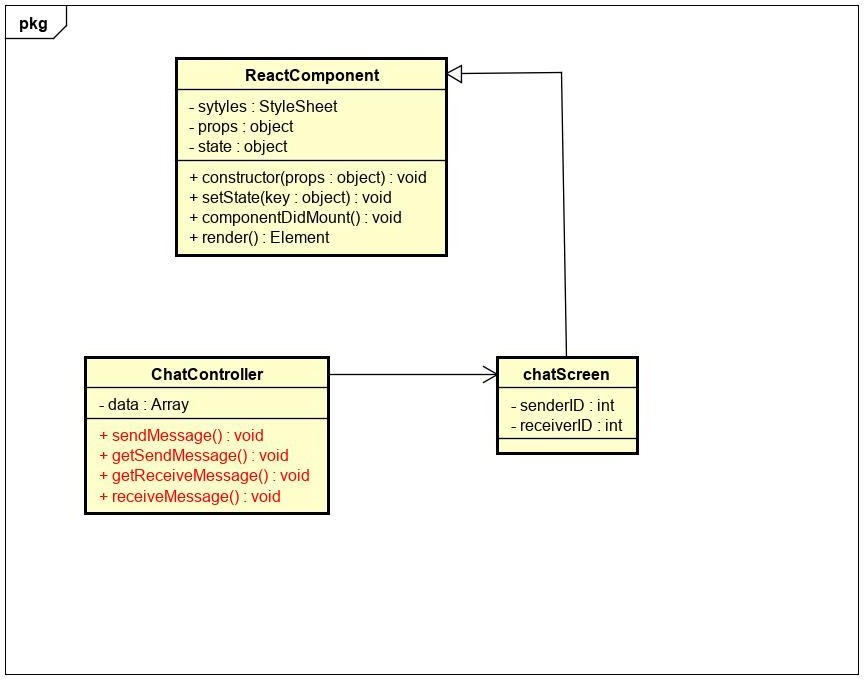
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | CreatePostScreen | | | **Type** | Class |
| **Description** | A class display createPostScreen. | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| arrImages | Array | private | Array of images that uploaded in thist post | | |
| caption | String | private | Caption of this post | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| createNewPost() | void | public | Handle create new post | | |
|  |  |  |  | | |

**\*Create NewPost sequence diagram**



### Chatting

**\*Chatting Class Diagram**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | ChatController | | | **Type** | Class |
| **Description** | The screen controls chatting action | | |  |  |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| data | Array | private | Contain data of fetch URL | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| sendMessage() | function | public | Send message from client to server | | |
| getSendMessage () | function | public | Get send\_message from client | | |
| receiveMessage(i) | function | public | Receive message from server | | |
| getReceiveMessage () | function | public | Get receive\_message from server | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | chatScreen | | | **Type** | Class |
| **Description** | The screen displays content of chattin | | | g |  |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| senderID | int | private | Id of sender | | |
| receiverID | int | private | Id of receiver | | |
| **Operations** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |

**\*Chatting Sequence Diagram**



**CHAP 5: Software Testing Document**

5.1. Introduction

5.1.1. Purpose

The primary purpose of this chapter is to detect and prevent defects which may be created by developers while developing the software and this may lead to software failures. On the other hand, another objective of this chapter is to provide information about the level of quality and to make sure that the end result meets the business and user requirements. It contains the following sections:

* Scope of Testing
* Testing Tool and Environment
* Resources and responsibilities
* Test strategy: Test approach, test stage
* Test schedule
* Feature to be tested
* Feature not to be tested
* Defect Log
* Test report

5.1.2. Scope of testing

* Stages of testing:

There are 4 phases in Testing Process: Unit testing, Integration testing, System testing and Acceptance testing.

|  |  |  |
| --- | --- | --- |
| ID | Test Stages | Description |
| 1 | Unit testing | The main aim of this endeavor is to determine whether the application functions as designed. In this phase, a unit can refer to a function, individual program or even a procedure, and a White-box Testing method is usually used to get the job done. |
| 2 | Integration testing | Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to find interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. |
| 3 | System testing | System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. |
| 4 | Acceptance testing | Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During this phase, the tester and some alpha test user will test the system to find out whether the application meets their business' needs. |

* Type of testing

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

* + GUI test
  + Function test
  + Regression test
  + Acceptance test include alpha test
  + Unit test

5.1.3. Range of testing

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

5.2. Test plan

5.2.1 Testing Tools and Environment

**5.2.1.1. Testing tools**

* **Front-end and Project Testing**
  + Chrome Developer Tool: To view logs.
  + Git: Manage bug.
  + Microsoft Excel: Manage testcase
* **API testing**
  + PostMan: to manage the list of all APIs and manually test APIs’ result
* **Unit test**
  + Jest: Unit test for JavaScripts
* **UI testing**
  + Expo client: View application when development

**5.2.1.2. Testing environment**

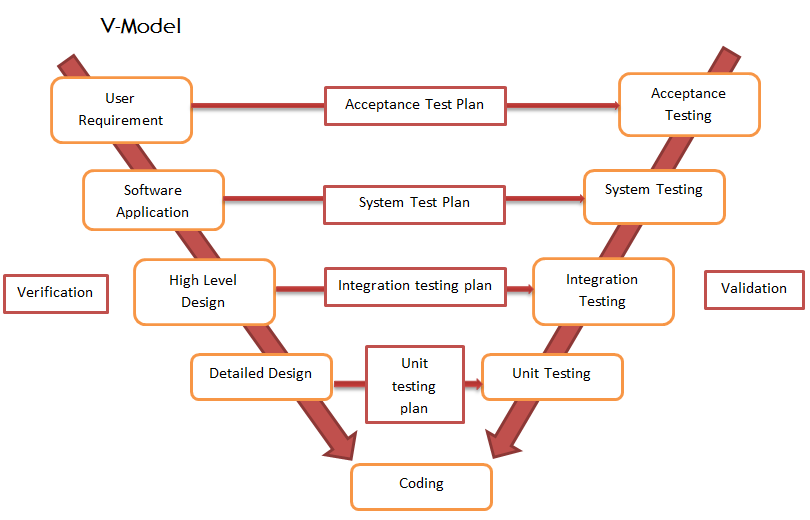
* **Device 1: Laptop Dell** 
  + CPU: i5 7300hq
  + Ram: 8gb
  + Gpu: gtx1050ti
* **Device 2: Laptop MSI**
  + CPU: i7 7700hq
  + Ram: 16gb
  + Gpu: gtx1050
* **Other device: Mobile phone**
  + HTC one M8
  + Xiaomi Redmi 4
  + Samsung a50
  + Nokia 8.1

5.2.2. Resources and Responsibilities

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

5.2.3. Test Strategy

**5.2.3.1. Test Model**



V-model

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

PetDating API has 2 levels of test:

* Unit testing: Automation tests that cover logic of Models and Libraries
* API testing: Automation tests that involve testing APIs directly (in isolation) to determine whether APIs return the correct response (in the expected format) for a broad range of feasible requests, react properly to edge cases such as failures and unexpected/extreme inputs.
* PetDating Front-end works mostly with GUI instead of logic and it depends on PetDating API, so that PetDating Front-end apply system testing which coverage of whole PetDating system.

**5.2.3.2. Test types**

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

* + - Unit test:
* Testing all individual implemented methods, functions of model class or library class.
* Unit test also includes database testing to verify constraint, transaction, default value, data types, data format, and check null and junk characters which are mentioned in database design and software requirement.
* Test case will have to cover all logic branch that function or method could execute with difference data input. Another alternative logic branch should be covered if not, that logic branch should be detected at API testing level.
* Implemented function's error message and database error message will be included in this test.
  + - API test:
* Involves testing APIs directly to determine if they meet expectations for functionality, reliability, performance, and security. API testing will test all of individual implemented API of PetDating API.
* Test case will verify constraint of data which be mention in Business rule
* Basically, almost all API test cases are executed as automation test. After that all API with standard sample datasets will be saved and confirmation tests will be executed by using Postman with developer's local database.
  + - UI test:
* User Interface testing verifies a user's interaction with the software. The goal of GUI testing is to ensure that the GUI provides the user with an appropriate access and navigation through the functions of the target-of-test. In addition, GUI testing ensures that the objects within the GUI function as expected and conform to requirement
* GUI test will be performed fully on all screens.
* This test targets to cover the verification of the overall look and feel of the OCFU system including initial position, font, text size, color, focus, initial button, tab order, label, screen sizes and sentences width.
* Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, it must be able to provide inputs to the input fields.
* Check if Error Messages are displayed correctly.
* Check if Font used in application is readable.
* Check if the alignment of the text is proper.
* Check if the Color of the font and warning messages is aesthetically pleasing.
* Check if images have good clarity.
* Check if images are properly aligned.
* Check the positioning of GUI elements for different screen resolution.
  + - Regression testing
      * The testing is to confirm that the bug was removed including the extent of the impact, when developer fix bug, developer and tester will confirm with each other what is the impact of fix bug modification, after that all impact unit or function will be retested by developer then tester after that
    - Acceptance testing
      * This test type will be executed by tester with designed test cases, acceptance test is a test type conducted to determine if the requirements of a specification or contract are met
      * It also includes alpha testing; alpha testing takes place at close relation user’s site and are free test to detect bug and strange behavior. By that, development team will improve UX and UI of system

**5.2.3.3. Test stage**

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

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

**5.2.3.4. Test schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | | Start date | End date |
| Iterator 1 | Create test plan | 15/10/2019 | 17/10/2019 |
| Create and execute unit test | 18/10/2019 | 1/11/2019 |
| Create and execute integration test | 18/10/2019 | 1/11/2019 |
| Create and execute system test | 25/10/2019 | 1/11/2019 |
| Create and execute acceptance test | 28/10/2019 | 2/11/2019 |
| Iterator 2 | Create test plan | 5/11/2019 | 11/11/2019 |
| Create and execute unit test | 11/11/2019 | 2/12/2019 |
| Create and execute integration test | 15/11/2019 | 1/12/2019 |
| Create and execute system test | 28/11/2019 | 1/12/2019 |
| Create and execute acceptance test | 30/11/2019 | 2/12/2019 |

**5.2.3.5. Deliverables**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Deliverables | Responsibilities | Delivered date |
| 1 | Test plan | Test Leader |  |
| 2 | Unit test | Developer |  |
| 3 | Integration test | Tester + Developer |  |
| 4 | System test | Tester + Developer + PM |  |
| 5 | Acceptance test | All member |  |
| 6 | Defect logs | All member |  |
| 7 | Final test summary report | Test Leader |  |

5.2.4. Features to be tested

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

5.3. Test case

5.3.1. UnitTest

5.3.2. System testing

Detail test cases will be described in PetDating\_Final\_TestCase.xlsx file.

5.3.3. Acceptance test

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

The content of the Checklist is shown in the table below

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Check lists | Yes | No |
| General | | | |
| T-01 | Text on all screens and pages for spelling and grammatical errors |  |  |
| T-02 | Functionality of buttons available on all screens and pages |  |  |
| T-03 | Validation error messages are displayed properly below the field |  |  |
| T-04 | All error messages are displayed in red color |  |  |
| T-05 | All inactive items are gray out |  |  |
| T-06 | Delete, cancel functionality for any record on screen are asked for confirmation |  |  |
| T-07 | All numeric values are formatted properly |  |  |
| GUI and Usability | | | |
| T-08 | Screens are designed follow project standards |  |  |
| T-09 | The screen well organized and easy to use |  |  |
| T-10 | All fields on screen and page (e.g. text box, radio options, dropdown lists) should be aligned properly |  |  |
| T-11 | Information is arranged symmetrically with adequate spacing between components |  |  |
| T-12 | The most important fields are located where they are easy to see |  |  |
| T-13 | Information is presented in the order that the user needs it |  |  |
| T-14 | The screen designed to fit the requirements for international use |  |  |
| T-15 | The text easy to translate. Don't use slang, acronyms, and abbreviations |  |  |
| T-16 | Icons and images are designed impression and copyright |  |  |
| T-17 | Font size, style and color for headline, description test, labels, infield data, and grid information standard as specified in SRS |  |  |
| T-18 | The static text is clear, concise, and meaningful |  |  |
| T-19 | Buttons follow the project standards for size and position (e.g., Negative button is correct size and to the right of the Positive button) |  |  |
| T-20 | Graphical objects are used appropriately and according to the guidelines specified in Graphical User Interface Design |  |  |
| T-21 | A list view is used to allow a collection of items that are on a single hierarchical level |  |  |
| T-22 | A tree view is used to allow a collection of items to be displayed and manipulated within varying hierarchical levels |  |  |
| T-23 | Pop-up menus are provided for the user to access information about an object's properties or perform specific tasks on the object |  |  |
| T-24 | Command button are used to trigger application processes |  |  |
| T-25 | Switch buttons are used to change status on/off. |  |  |
| T-26 | System display notification message when meet trouble, error |  |  |
| Database | | | |
| T-27 | Correct data is getting saved in database upon successful backend submit |  |  |
| T-28 | Values columns are not accepting null values |  |  |
| T-29 | Data should be stored in single multiple tables based on design |  |  |
| T-30 | Input data is not truncated. Field length shown to user on application, web and in database schema should be same |  |  |
| T-31 | Input numeric fields with minimum, maximum, and float values |  |  |
| T-32 | Input numeric fields with negative values (for both acceptance and non-acceptance) |  |  |
| T-33 | Drop down list are saved correctly in database |  |  |
| T-34 | Database fields are designed with correct data type and data length |  |  |
| T-35 | All collections constraints like Primary key, etc. are implemented correctly |  |  |
| T-36 | Input field leading and trailing spaces are truncated before committing data to database |  |  |
| Performance | | | |
| T-37 | Real time connection |  |  |
| T-38 | Mobile app and web page load time is within acceptable range |  |  |
| T-39 | Mobile app and web page load on slow connections |  |  |
| T-40 | Response time for any action under light, normal, moderate and heavy load conditions |  |  |
| T-41 | Database query execution time |  |  |
| T-42 | Stress testing of application |  |  |
| Security | | | |
| T-43 | Test cookie allowed (disabled or allowed to be edited) |  |  |
| T-44 | Test the form giving away security information if the source is viewed |  |  |
| T-45 | Test password security and password policy enforcement |  |  |
| T-46 | Check application log out functionality |  |  |
| T-47 | Cookie information should be stored in encrypted format only |  |  |

5.3.4. Defect Log

Excel and github are used to manage bugs of FCWB Processes:

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

The defect is logged in “PetDating\_Defect-Log.xlsx” file.

5.4. Test Report