

**Capstone Project Final Report**

**Project title: Social Fashion Network (SOFA)**

**Hanoi, May 6th, 2021**

**Table of Contents**

**No table of contents entries found.**

Chapter 1: **Introduction**

1. **Purpose**

This chapter provides an overview of the project including background information, a literature review of the existing system and raising a proposal for ideas of improvement.

1. **Project Information**

* Project name: **SOFA – Fashion Social Network**
* Project code: **SOFA**
* Project group name: **SWP490\_G22**
* Product type: **Mobile Application – Android**

1. **Project Team**

### Supervisor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-mail** | **Title** |
| Supervisor | Phạm Ngọc Hà | 0988623000 | [hapn10](mailto:hapn10)@fpt.edu.vn | Lecturer |

### Team Members

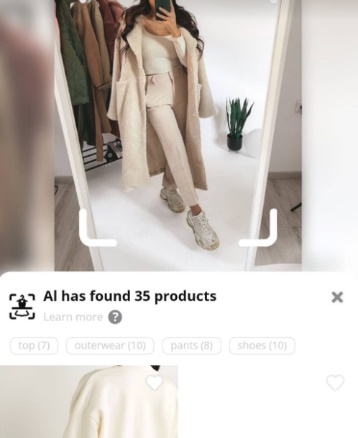
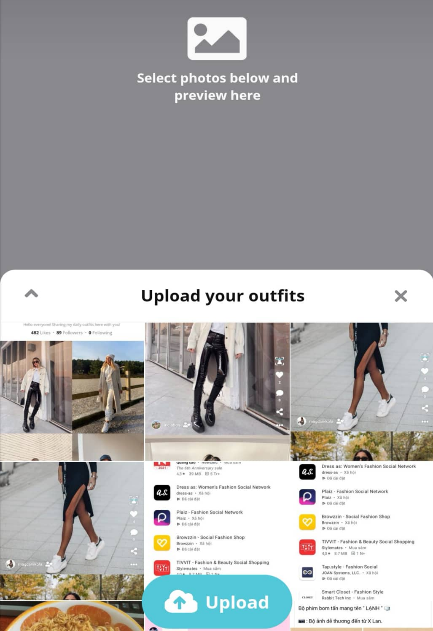
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Full name** | **Student ID** | **Phone** | **E-mail** | **Role** |
| 1 | Lê Thiện Văn | HE130820 | 0375956599 | vanLTHE130820@fpt.edu.vn | Leader |
| 2 | Nguyễn Huy Hoàng | HE130395 | 0355517865 | hoangNHHE130395@fpt.edu.vn | Document Lead |
| 3 | Hà Việt Dũng | HE130952 | 0942561863 | dungHVHE130952@fpt.edu.vn | Member |
| 4 | Mai Văn Viên | SE05783 | 0969614666 | vienMVSE05783@fpt.edu.vn | Test Lead |
| 5 | Nguyễn Thanh Tùng | SE05590 | 0344261997 | tungNTSE05590@fpt.edu.vn | Member |

1. **Background**

Today, in a developing age, living conditions are getting higher and higher. Eat well and dress well instead of eating fully and dressing warmly. Fashion is almost indispensable for every person. Fashion tells almost everything about a person, helps a person more beautiful and makes someone confident. But not all people know how to choose clothes which are suitable for themselves. To resolve this problem, we develop a system that allows users to share their dress styles and suggest them to other users. So, thanks to this system every people could choose suitable clothes and dress style.

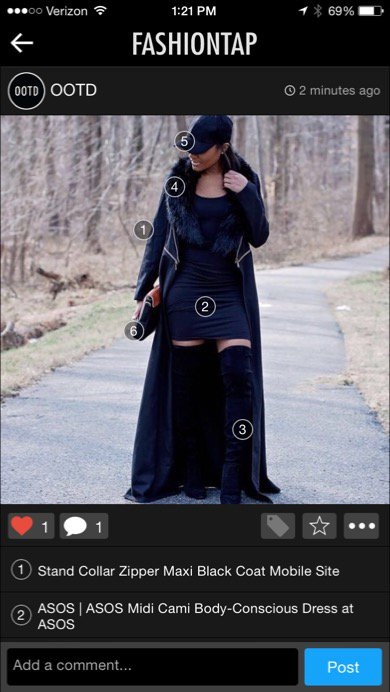
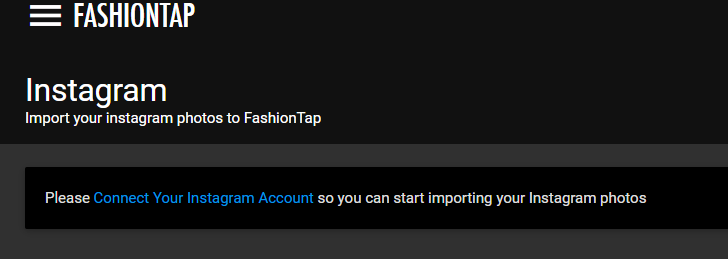
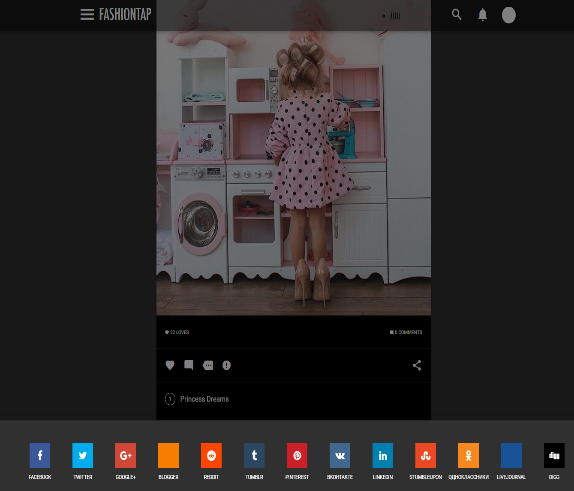
1. **Literature review of existing system**
   1. Dress As

The fashion social network for women. This app can be downloaded on Google Play.



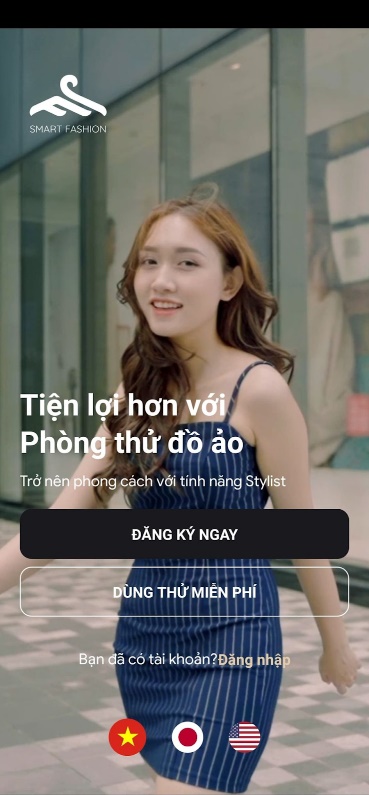
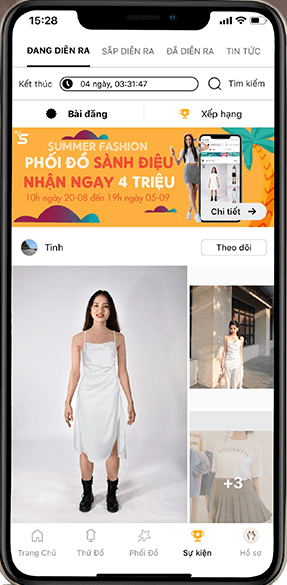
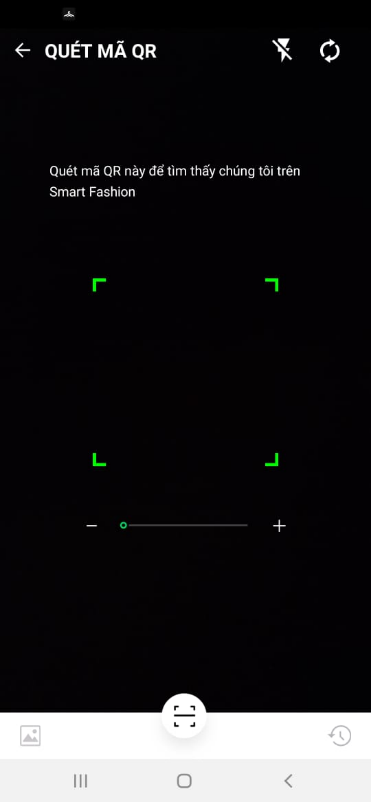
|  |  |  |
| --- | --- | --- |
| **Advantages** | **Disadvantages** | **Compare with SOFA** |
| GUI simple, modern | Just for women | For everybody |
| Easy to share a personal image | Simply an app to share images and search for related products, with little social network direction | A social network, allow interaction between users more than the existed app (Can chat, add friend, comment…) |
| Can search for similar products by image | Don’t allow write the caption for image, don’t allow share post | Allow write the caption, interaction, comment, share |
|  | Don’t have feature fashion trending | Have feature fashion trending |
|  | Don’t have feature suggest style and clothes for user | Based on user's data along with fashion style then suggest style and clothes for post who have similar data |
|  | There isn't feature posts featured based on interactions, posts of KOL | Have feature posts featured based on interactions, posts of KOL |
|  | There isn’t suggest where a user can buy the product. Search product restricted products on asos.com website | Suggest where a user can buy the product. Not restricted search product. |
|  | User are those who want to share pictures of fashion, do not have shop features | Have feature for seller and shop |

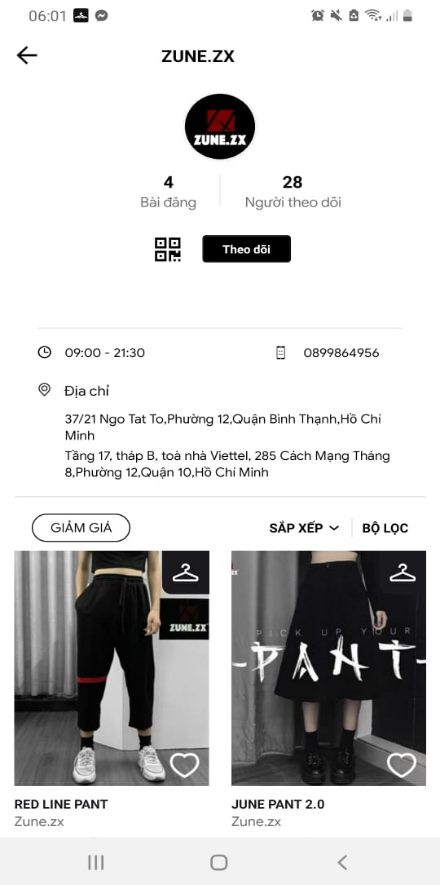
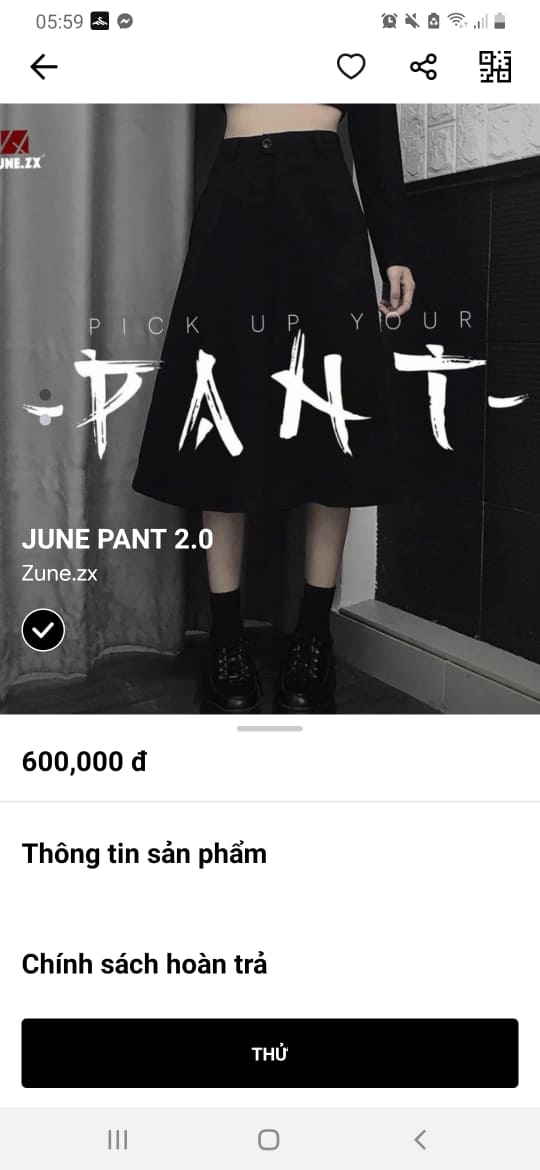
* 1. Fashion Tap



|  |  |  |
| --- | --- | --- |
| **Advantages** | **Disadvantages** | **Compare with SOFA** |
| Allows users to insert links on each item in the photo leading to the place of sale | The system still has many errors. For example, the image cropping feature after upload is still faulty |  |
| Can import photos from Instagram to FashionTap | Simply an app to share images, with little social network direction | The social network allows interaction between users more than existed app (Can chat, add friend, comment…) |
| After uploading photos, you can use the system's tools to edit the photos before posting | Don’t have feature fashion trending | Have feature fashion trending |
| Allows users to share post to other social networking | Don’t have feature suggest style and clothes for user | Based on user's data along with fashion style then suggest style and clothes for other users who have similar data |
| Can report post with bad content | There isn't feature posts featured based on interactions, posts of KOL | Have feature posts featured based on interactions, posts of KOL |
| Can review all product about fashion include clothes, cosmetics, accessories fashion | Don’t have feature rating for the post | User can rate post that show the quality of the post |
| Have feature ask and answer |  |  |

## Smart Fashion



****

|  |  |  |
| --- | --- | --- |
| **Advantages** | **Disadvantages** | **Compare with SOFA** |
| User can try clothes on a virtual model (Use deep-learning to simulate clothing and try on a virtual model) | focus on features buy, trial and sell, so it does not attract a large number of users. | User can post, share their image, chat and add friend with other people |
| Can search product by scan QR code | Don't have interaction between users | Social network, allow interaction between users more than existed app (Can chat, add friend, comment…) |
| Can create event (make user interest app) | Don’t have feature fashion trending | Have feature fashion trending |
| Have feature for seller and shop | Don’t have feature suggest style and clothes for user | Based on user's data along with fashion style then suggest style and clothes for other users who have similar data |
| User can save the product that they like | The system still has many errors. |  |

1. **Business Opportunity**

Today, the problem about choose fashion style and clothes is hot, besides that linking between shop and customer is very important. So, it's a good opportunity for fashion social networking and suggests fashion system to develop.

However, User's need and the shop's need are not similar. Shop want to customer know about their product. People want to know how to choose style and clothes for themselves and where they can buy that product. So, SOFA will try to solve the problem by the way that allows users to share personal fashion style, suggest for user the fashion style that suitable for them and hint user shops where they can buy that product.

SOFA will help user choose clothes and find clothes, help customers know about the product of shops.

1. **Software Product Vision**

With the applying recommendation system and deep-learning API to suggest clothes and fashion style for user. We want to we aim to optimize the operation of system. We develop system with backend is a restful API, frontend for the manager is a website and frontend for user is android app. Mobile app help user feel conveniently and easy to use (for example, user can take a picture, upload picture every time, everywhere). Backend restful API will help us develop frontend for user on other platforms like website, android app...

1. **System Functions**

There are 3 types of users in SOFA system

|  |  |
| --- | --- |
| **Type of User** | **Description** |
| Guest | People who use the system without login. They can only use some basic feature but cannot use advanced feature like create post, like, rate,… |
| User | People who have logged into the system. They can use all of feature of the application. |
| Administrator | People who use management system. They have the most powerful rights in the system such as managing accounts, posts, voucher, feedback,… |

SOFA System provides the following functions:

User:

* *Authentication: login, logout, register, forget password*
* *Post: view list post, view post detail, like post, rate post, comment post, markup post, report post, create post*
* *User Interact: view other’s profile, follow user, report user, message to other user*
* *Message: view conversation list, view a conversation, delete a conversation, create new message, delete a message*
* *Balance: view balance, top up balance, view voucher*
* *Hot: view list hot post, all function like post(User)*
* *Recommend: view list recommend posts, all function like post(User)*
* *Support/Feedback: send feedback, view list feedback, view a feedback’s detail, send support request, view support request’s result*
* *Setting: change password, turn on/off notification, change message theme*
* *Notification: view list notification, view detail notification, mark all as read*
* *Markup post: markup a post, view markup post list, view markup post detail*

Administrator:

* *Account management:* view my profile, change password
* *Feedback management:* view feedback list, search feedback, accept feedback

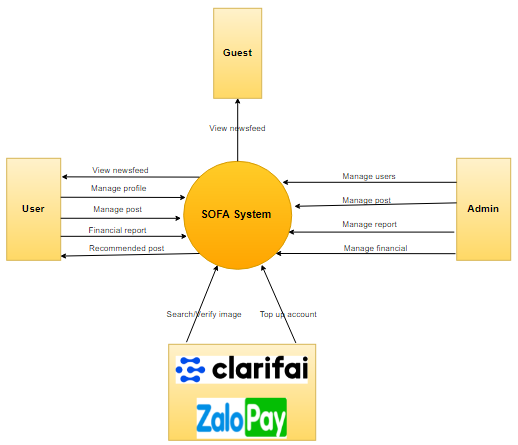
- *Support management:* view support list, search support, approve support, reject support

- *Voucher management:* view voucher list, search voucher, delete voucher

- *Report management:* view report list

- *User management:* view post list, search post, view post detail, delete post

1. **System Overview**

****

1. **Out-of-scope functions**

Due to the limitation of time, we will not implement some of the intended functions, although we are aware that they are also important and useful for users of the system:

* Hide post, unhide post
* Share post
* Block user
* Call voice & call video

1. **Tool and technique**

|  |  |
| --- | --- |
| **Category** | **Tools & techniques** |
| **Programming languages** | C#, Python, HTML, CSS, JavaScript, TypeScript |
| **Framework** | ASP.Net core API, Angular 8, React Native |
| **API** | Clarifai |
| **Third-party** | Zalopay |
| **DBMS** | SQL Server |
| **IDEs/Editors** | Visual Studio Code, Visual Studio 2019 |
| **UML tools** | Lucid chart, draw.io |
| **Version Control** | Github |
| **Deployment server** | IIS Server (Server machine) |
| **Project management tool** | Trello |

Chapter 2: **Project Plan**

1. **Purpose**

This chapter provides an overview of the project plan, including software process model, project overview, project organization. Moreover, the document also details tools will be used, coding conventions, communication management.

1. **Project Overview**

2.1 Channels

* Application: SOFA Application
* Advertisement:
* Facebook: <https://www.facebook.com/Sofateam2021>
* Youtube: <https://www.youtube.com/watch?v=E7ih7YzWL-8>
  1. Project organization
  2. Role and responsibilities

|  |  |
| --- | --- |
| **Team Member** | **Role** |
| Pham Ngoc Ha | Supervisor |
| Le Thien Van |  |
| Nguyen Huy Hoang |  |
| Ha Viet Dung |  |
| Mai Van Vien |  |
| Tran Thanh Tung |  |

* 1. Software process model

1. **Project management Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **WBS Item** | **Complexity** | **Est. Effort (Man-days)** |
| **I** | **SOFA Project** |  | **375** |
| **1** | **Pre-Initiating Phase** |  | **5** |
| 1.1 | Select Project Manager | Simple | 1 |
| 1.2 | Determine the role of team member | Simple | 1 |
| 1.3 | Choose topic | Simple | 2 |
| 1.4 | Register capstone project | Simple | 1 |
| **2** | **Initiating phase** |  | **9** |
| 2.1 | Determine stakeholder | Medium | 1 |
| 2.2 | List up requirement | Simple | 2.5 |
| 2.3 | Define scope of project | Simple | 2.5 |
| 2.4 | Create report 1 - project introduction document | Medium | 2.5 |
| 2.6 | Modify project introduction document | Simple | 0.5 |
| **3** | **Project Plan Phase** |  | **8** |
| 3.1 | Choose working model process | Simple | 2.5 |
| 3.2 | Select tools and techniques | Simple | 0.5 |
| 3.3 | Create project schedule | Simple | 1 |
| 3.4 | Create risk management plan | Medium | 1 |
| 3.5 | Organize project resources | Simple | 2 |
| 3.6 | Create report 2 - software project management plan | Medium | 1 |
| **4** | **Executing Phase** |  | **346** |
| **4.1** | **Design** |  | **59.5** |
| 4.1.1 | List up use case | Simple | 1 |
| 4.1.2 | Create use case description | Medium | 1.5 |
| 4.1.3 | Create report 3 -Software requirement specification | Medium | 5 |
| 4.1.4 | Create Architecture design | Complex | 10 |
| 4.1.4.1 | Data Architecture design | Medium | 2 |
| 4.1.4.2 | Data flow architectures | Medium | 2 |
| 4.1.4.3 | Call and Return architectures | Medium | 2 |
| 4.1.4.4 | Object Oriented architecture | Medium | 2 |
| 4.1.4.5 | Layered architecture | Medium | 2 |
| 4.1.5 | Database design | Medium | 10 |
| 4.1.6 | GUI layout design | Complex | 25 |
| 4.1.7 | Create report 4 - Software design | Simple | 5 |
| 4.1.8 | Redesign use case | Simple | 2 |
| **4.2** | **Implementation** |  | **237** |
| 4.2.1 | Back-end (RESTful API) |  | 92 |
| 4.2.1.1 | Configuration and setup environment | Simple | 2 |
| 4.2.1.2 | Create database | Simple | 15 |
| 4.2.1.3 | Coding common, Middleware, DBContext | Medium | 5 |
| 4.2.1.4 | Coding Model and Data Access Object | Medium | 10 |
| 4.2.1.5 | Coding controller handler request of manager page | Complex | 25 |
| 4.2.1.6 | Coding controller handler request of user app | Complex | 35 |
| 4.2.2 | Front-end |  | 140 |
| 4.2.2.1 | SOFA Manager (Web flatform) | Complex | 40 |
| 4.2.2.2 | SOFA (Android app flatform) | Complex | 100 |
| 4.2.3 | Deploy |  | 5 |
| 4.2.3.1 | Deploy host and domain config | Medium | 5 |
| 4.3 | **Test** |  | **49** |
| 4.3.1 | Test manager website |  | 24 |
| 4.3.1.1 | Create test cases for unit test | Simple | 2 |
| 4.3.1.2 | Unit test | Simple | 2 |
| 4.3.1.3 | Fix bug | Medium | 2 |
| 4.3.1.4 | Create test cases for integration test | Simple | 2 |
| 4.3.1.5 | Integration test | Simple | 2 |
| 4.3.1.6 | Fix bug | Medium | 2 |
| 4.3.1.7 | Create test cases for system test | Simple | 2 |
| 4.3.1.8 | System test | Simple | 2 |
| 4.3.1.9 | Fix bug | Medium | 2 |
| 4.3.1.10 | Create test case for acceptance test | Simple | 2 |
| 4.3.1.11 | Acceptance test | Simple | 2 |
| 4.3.1.12 | Fix bug | Medium | 2 |
| 4.3.2 | Test user application |  | 24 |
| 4.3.2.1 | Create test cases for unit test | Simple | 2 |
| 4.3.2.2 | Unit test | Simple | 2 |
| 4.3.2.3 | Fix bug | Medium | 2 |
| 4.3.2.4 | Create test cases for integration test | Simple | 2 |
| 4.3.2.5 | Integration test | Simple | 2 |
| 4.3.2.6 | Fix bug | Medium | 2 |
| 4.3.2.7 | Create test cases for system test | Simple | 2 |
| 4.3.2.8 | System test | Simple | 2 |
| 4.3.2.9 | Fix bug | Medium | 2 |
| 4.3.2.10 | Create test case for acceptance test | Simple | 2 |
| 4.3.2.11 | Acceptance test | Simple | 2 |
| 4.3.2.12 | Fix bug | Medium | 2 |
| 4.3.3 | Summary and evaluation | Simple | 1 |
| 4.1.9 | Create report 5 - Test case document | Simple | 2.5 |
| 4.1.10 | Create report 6 - Software user guide | Simple | 2.5 |
| **4.4** | **Finish executing phase** |  | **0.5** |
| **5** | **Monitoring and controlling** |  | **2** |
| 5.1 | Progress report | Simple | 2 |
| **6** | **Close** |  | 5 |
| 6.1 | Create final report | Medium | 5 |
| 6.2 | Project complete | Simple | 0 |

1. **Meeting minutes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Communication Item** | **Who/ Target** | **Purpose** | **When, Frequency** | **Type, Tool, Method(s)** |
| Weekly meeting | Supervisor and team members | Review members’ work achievements during the week and report the project’s progress and status. | Every Tuesday | Face to face |
| Daily meeting | Team members | Report the progress that members achieved each day. | 21h,  every day | Google meet |
| Unscheduled Meeting | Team members | Occurred when there's a critical problem that need to be resolved immediately |  | Face to face, google meet |

1. **Risk management Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Risk** | **Description** | **Impact** | **Possibility** | **Response Plans** |
| 1 | Inaccurate Estimations | Schedule is inaccurate | Medium | Medium | 1) Create schedule carefully 2) Add backup time (or backup schedule) 3) Focus on close targets |
| 2 | Poor Quality Code | The code may be difficult to read, meaning it is difficult for other developers to review or make changes | High | Low | 1) Code reviews 2) Clear coding standards and guides 3) Testing of all code |
| 3 | Inadequate Human Resources | Don't have enough member to complete project be on time | Medium | Low | 1) Register overtime for team member |
| 4 | Technical Difficulties | Have issues during coding | Medium | Medium | 1) Choose design pattern appropriate 2) Create architecture design more detail and exactly 3) Have a professional in team |
| 5 | Security | The system is unsafe for user, hacker can attack and take user information… | High | High | 1) Using token for improving backend security 2) Add more algorithm for hash private information 3) Take preventive measures hacking |
| 6 | Serious error in synchronization between web and mobile platform | display on browser and mobile is more difference. Maybe, the component just can use on web but not mobile | Medium | High | 1) Choose another component to fix 2) Make components difference but similar feature |
| 7 | lack of requirement | Developer create code was not enough requirement | High | Medium | 1) Code reviews 2) Clear coding standards and guides 3) Testing of all code |

1. **Document management Plan**

We use Github as our tool for sharing, editing and version control of our documents, along with Status Reports with-in each document. It allows us to see what is changed in the documents and who is responsible for that change, reverse and simultaneously compare between versions of the document.

1. **Source code management Plan**

For version control of our source code, we use Git hub. It tracks the changes team members make to files, so we have a record of what has been done, and we can revert to specific versions should we ever need to. Github also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

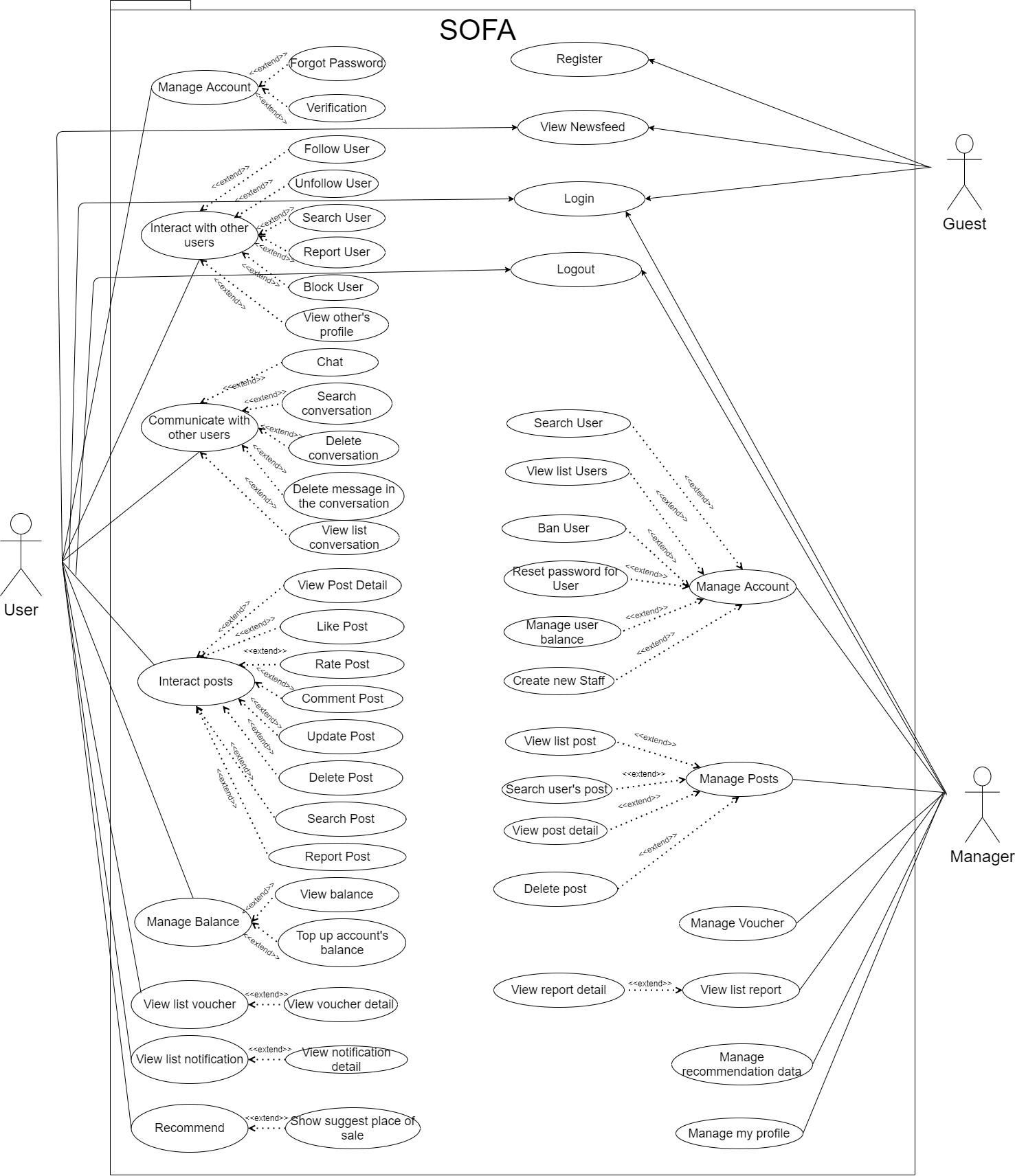
Github also features branches that provide an isolated environment for every change to our codebase. When a team member wants to start working on something - no matter how big or small - he creates a new branch. This ensures that the master branch always contains production - quality code.

**Chapter 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.

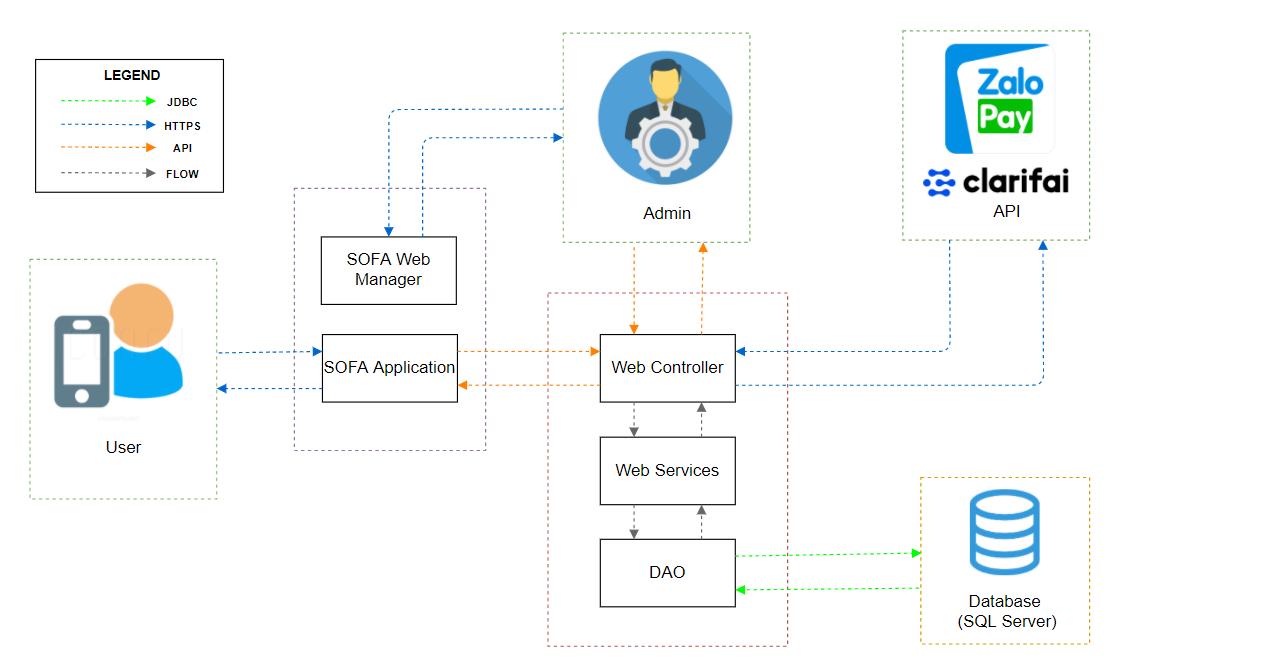
1. **User requirements**
   1. Overview
2. Use case diagram



# Detailed Description of Components

Chapter 4: **Detailed Description of Components**

## *System Architecture Design*



*Figure 4.1 – System Architecture Design*

## *Software Architecture Design*

## 

*Figure 4.2 – Software Architecture Design*

## *3. Description of System Interface*

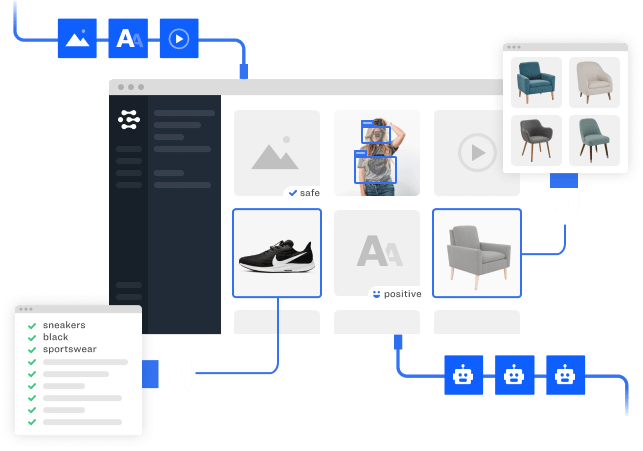
***3. 1. Database management system***

## 

*Database management system*

Database management system is a software which is used to manage the database. For example: SQL Server, MySQL, Oracle, etc. are a very popular commercial database which is used in different applications. DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more. It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

***3.2. Clarifai***

******

**Clarifai** is an Image Recognition Software solution highly appreciated by both users and experts in the field of Deep Learning Software.

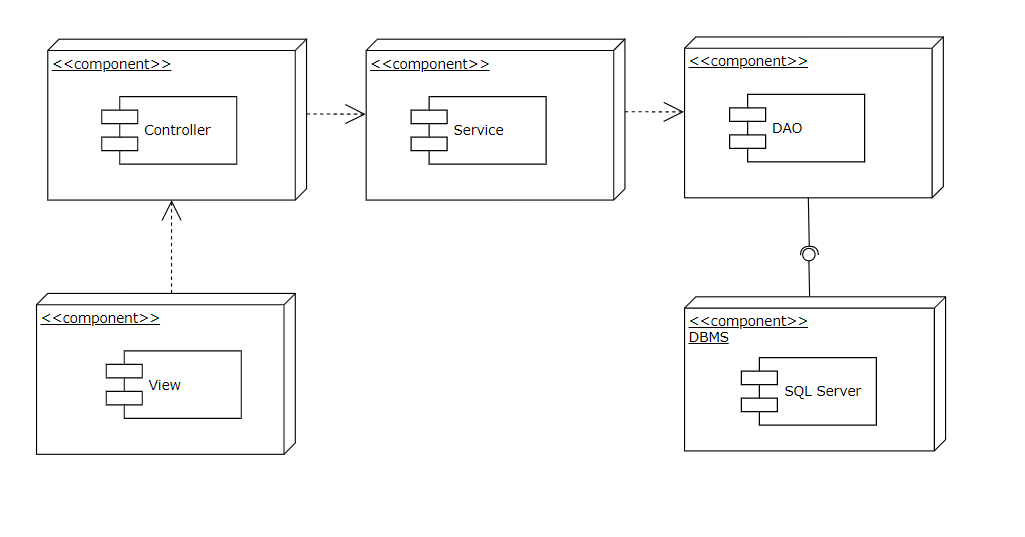
The platform includes the ability to moderate content, perform visual search, visual similarity, and organize media collections. It has pre-built recognition models that can identify a specific set of concepts like food or travel, NSFW, and its general model which can identify a range of concepts including objects, ideas, and emotion. It also has the ability to create custom models which can identify other arbitrary objects such as cars or breeds of dogs.

***3.3. Zalopay***

******

ZaloPay is an e-wallet platform and online payment system developed by VNG to deploy in-app payments and scan QR codes for payment on mobile devices, allowing users to do so. transactions using phones and tablets running Android and iOS. For many years (from 2018 to 2020), many channels have been introduced as the top 5 recommended e-wallets in Vietnam

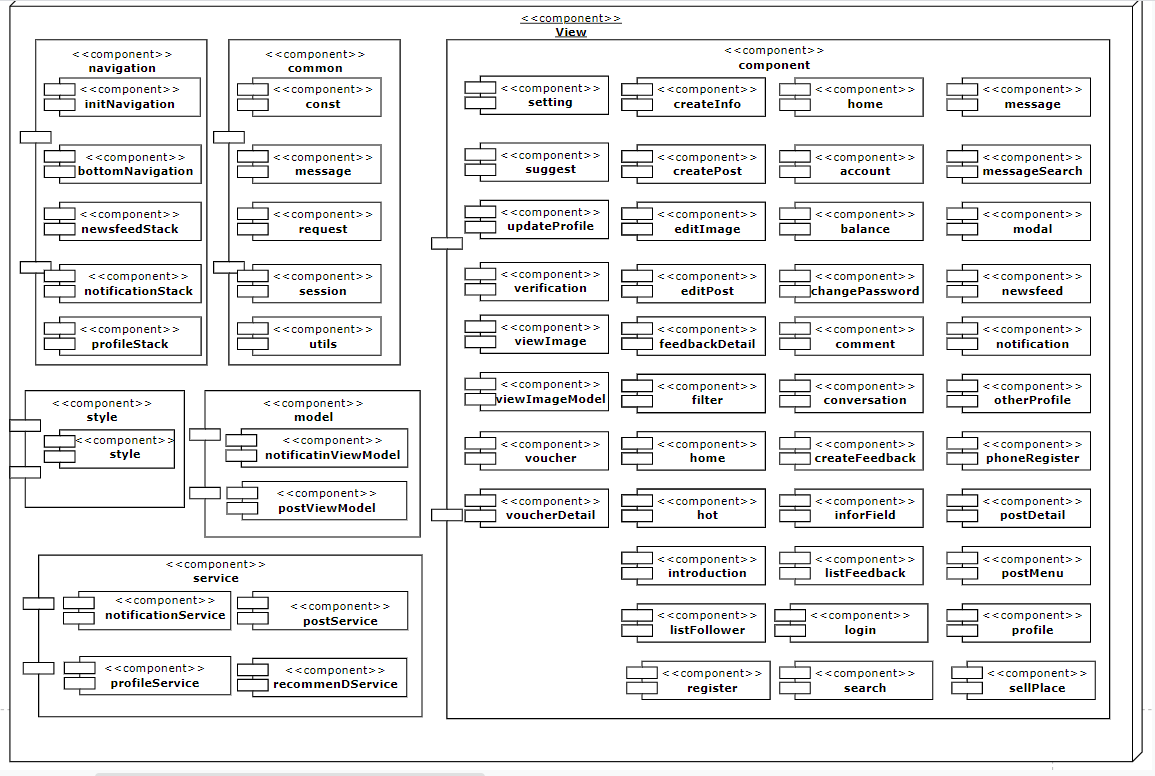
**4. Component Diagram**

****

## 

## 5. Detailed Description of Component

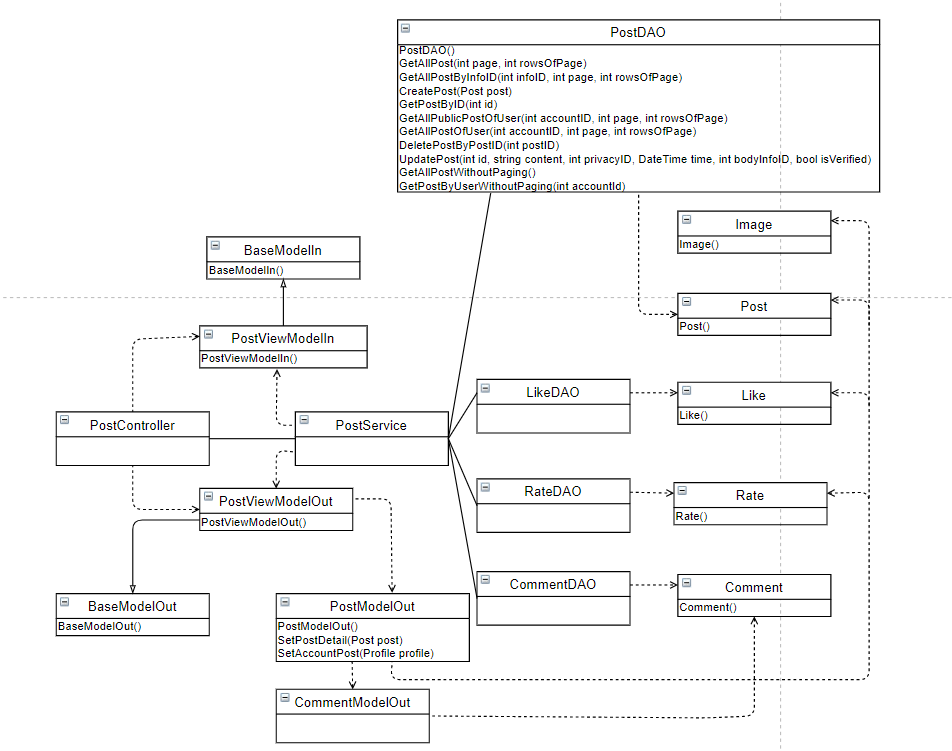
***5.1. View***



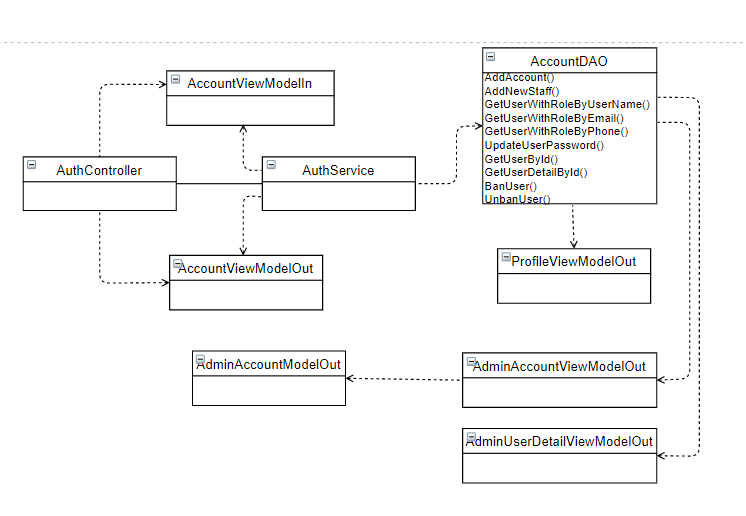
## Class Diagram

***6.1. Class diagram***

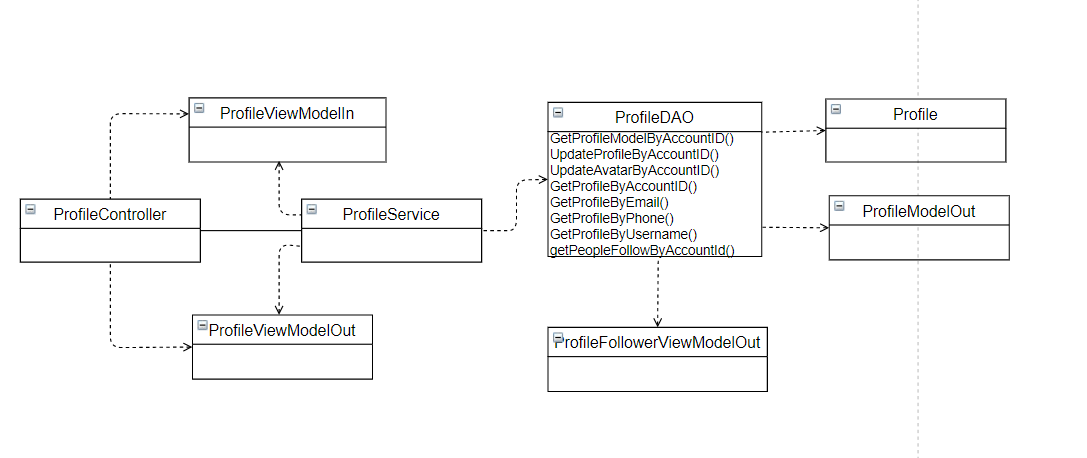
***6.1.1. Post Class Diagram***

******

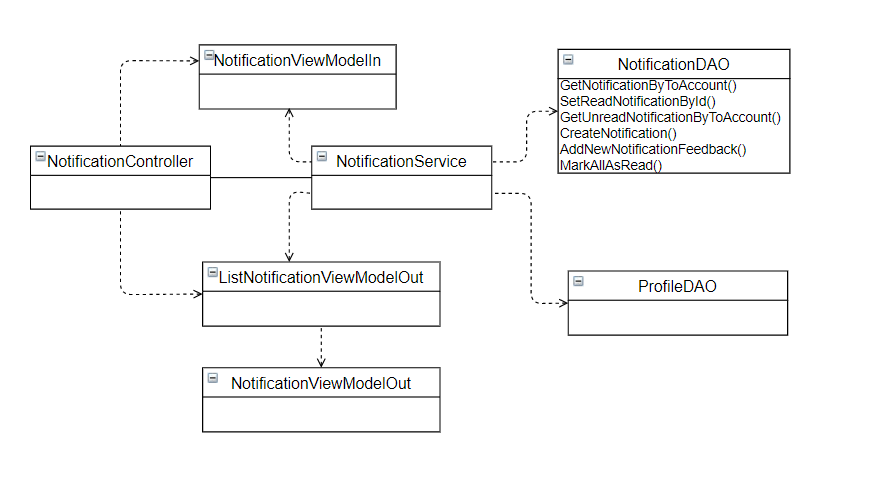
***6.1.2. Account Class Diagram***

******

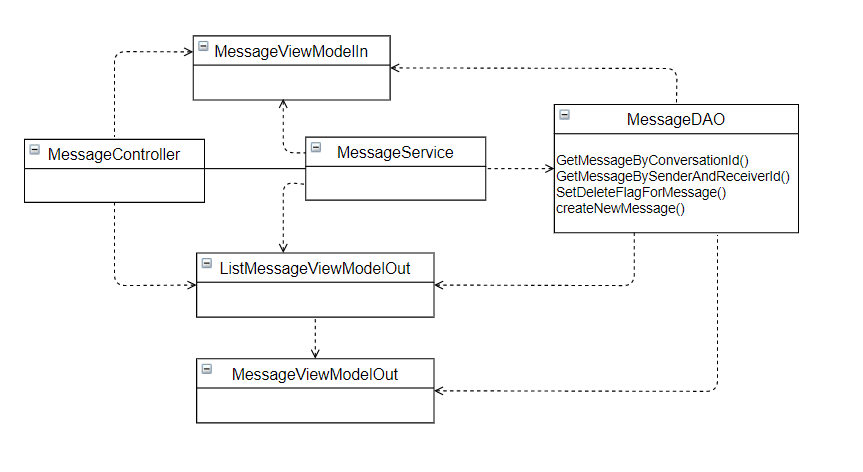
***6.1.3. Profile Class Diagram***

******

***6.1.4. Notification Class Diagram***

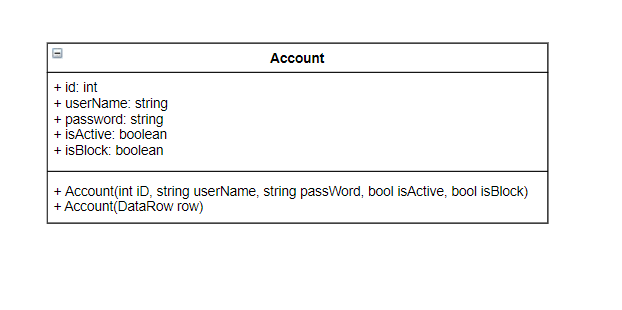
******

***6.1.5. Message Class Diagram***

******

***6.2. Models***

***6.2.1 Account***

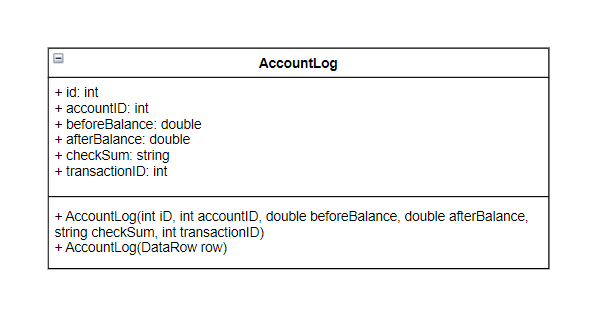
****

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | ID of account |
| 02 | username | string | User Name of account |
| 03 | password | string | Password of account |
| 04 | isActive | boolean | Status active of account |
| 05 | isBlock | boolean | Status block of account |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Account(int iD, string userName, string password, bool isActive, bool isBlock) | Default constructor |
| 02 | Account(DataRow row) | Another constructor |

***6.2.2 AccountLog***

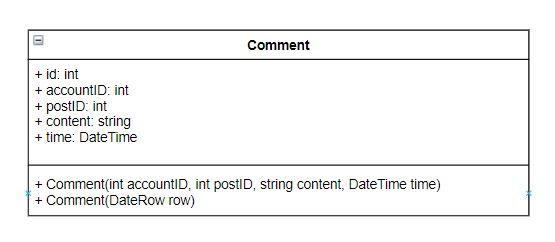
****

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | ID of log |
| 02 | accountID | int | ID of account log |
| 03 | beforeBalance | double | Balance of account before transaction |
| 04 | afterBalance | double | Balance of account after transaction |
| 05 | transactionID | int | ID of transacation |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | AccountLog(int iD, int accountID, double beforeBalance, double afterBalance, int transactionID) | Default constructor |
| 02 | AccountLog(DataRow row) | Another constructor |

***6.2.3 Comment***

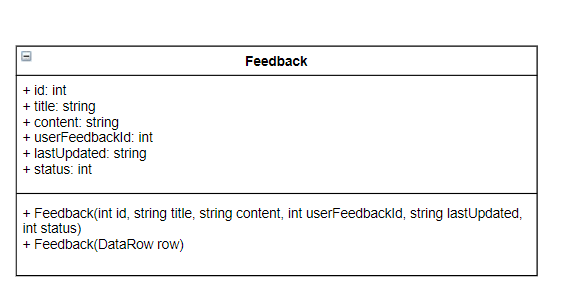
******

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | Id of comment |
| 02 | accountID | int | Id of account comment |
| 03 | postID | int | Id of the post |
| 04 | content | string | Content of the comment |
| 05 | time | DateTime | Time comment is created |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Comment(int id, int accountID, int postID, string content, Datetime time) | Default constructor |
| 02 | Comment(DataRow row) | Another constructor |

***6.2.4 Feedback***

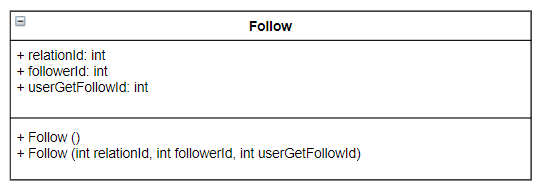
******

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | Id of the feedback |
| 02 | title | string | Title of the feedback |
| 03 | content | string | Content of the feedback |
| 04 | userFeedbackId | id | Id of the user have send feedback |
| 05 | lastUpdated | string | The final date update the feedback |
| 06 | status | int | Status of Feedback:   * Status = 1: Resolved * Status = 2: Unresolved * Status = 3: Rejected |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Feedback(int id,string title, string content, int userFeedbackId, string lastUpdated, int status) | Default constructor |
| 02 | Feedback(DataRow row) | Another constructor |

***6.2.5 Follow***

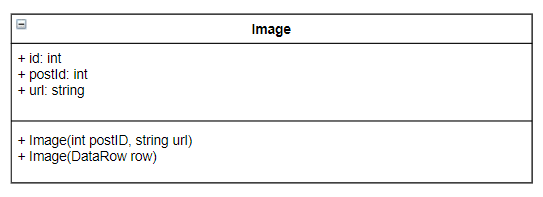
******

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | relationId | int | Relation of the Following :   * Relation = 1 : Follow * Relation = 2 : Block |
| 02 | followerId | int | Id of the user follow |
| 03 | userGetFollowId | int | Id of the user is followed |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Follow() | Default constructor |
| 02 | Follow(int relationId, int followerId, int userGetFollowId) | Another constructor |

***6.2.6 Image***

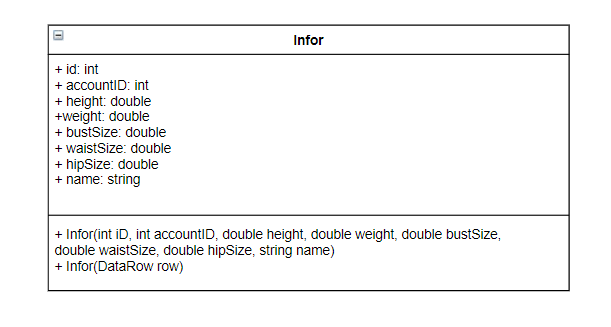
******

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | Id of the image |
| 02 | postId | int | Id of the post contain the image |
| 03 | url | string | The path to the image file |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Image(int postID, string url) | Default constructor |
| 02 | Image(DataRow row) | Another constructor |

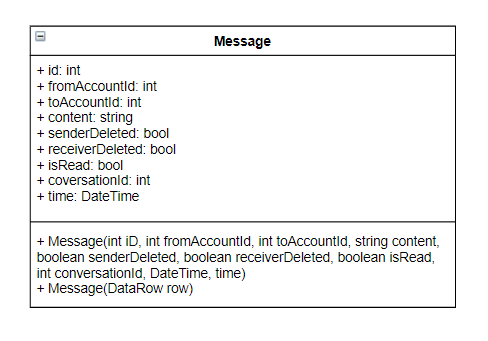
***6.2.7 Infor***

******

**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | Id of the Infor |
| 02 | accountID | int | Id of the account have the infor |
| 03 | height | double | Height of the person |
| 04 | weight | double | Weight of the person |
| 05 | bustSize | double | The size of the butt |
| 06 | waistSize | double | The size of the waist |
| 07 | hipSize | double | Size of the hip |
| 08 | name | string | Name of the Infor |

***6.2.8 Message***

******

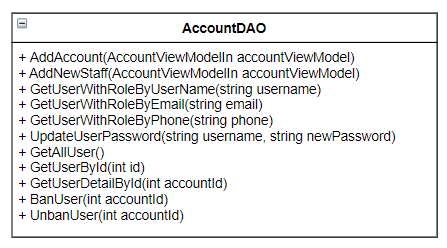
**Class specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **Attribute** | **Type** | **Description** |
| 01 | id | int | Id of the message |
| 02 | fromAccountId | int | Id of sender |
| 03 | toAccountId | int | Id of the receiver |
| 04 | content | string | Content of the message |
| 05 | senderDeleted | boolean | Status of the message on sender :   * Status = 0 : UnDeleted * Status = 1 : Deleted |
| 06 | receiverDeleted | boolean | Status of the message on receiver:   * Status = 0 : Undeleted * Status = 1 : Deleted |
| 07 | isRead | boolean | Status of the message on receiver is read:   * Relation = 0 : Unread * Relation = 1 : Read |
| 08 | conversationId | int | Id of the conversation |
| 09 | time | DateTime | Time of the message |

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Message(int iD, int fromAccountId, int toAccountId, string content,boolean senderDeleted, boolean receiverDeleted, boolean isRead,int conversationId, DateTime, time) | Default constructor |
| 02 | Message(DataRow row) | Another constructor |

**6.3. DAO**

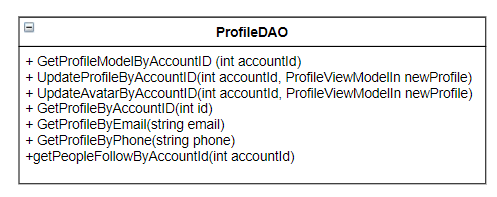
***63.1 AccountDAO***

****

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | AddAccount(AccountViewModelIn accountViewModel) | Create an account for users |
| 02 | AddNewStaff(AccountViewModelIn) | Create a new account for staff manager |
| 03 | GetUserWithRoleByUserName(string username) | Get information of users with their username |
| 04 | GetUserWithRoleByEmail(string username) | Get information of users with their Email |
| 05 | GetUserWithRoleByPhone(string username) | Get information of users with their Phone |
| 06 | UpdateUserPassword(string username, string newPassword) | Change user password |
| 07 | GetAllUser() | Get all user in system |
| 08 | GetUserById(int id) | Get an user’s account and profile by his/her ID |
| 09 | GetUserDetailById(int id) | Get an user’s profile and statistics |
| 10 | BanUser(int accountId) | Ban an user |
| 11 | UnbanUser(int accountId) | Unban an user |

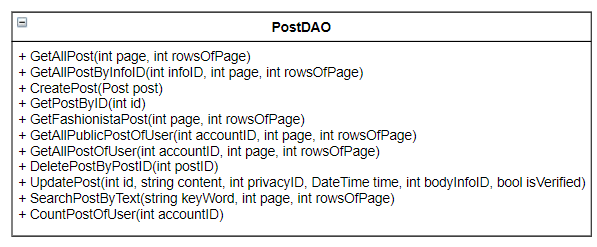
***6.3.2. ProfileDAO***

****

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetProfileModelByAccountID (int accountId) | Get an custom profile by account id |
| 02 | UpdateProfileByAccountID(int accountId, ProfileViewModelIn newProfile) | Update an profile by account id |
| 03 | UpdateAvatarByAccountID(int accountId, ProfileViewModelIn newProfile) | Update an user’s profile avatar by account id |
| 04 | GetProfileByAccountID(int id) | Get user’s profile by account id |
| 05 | GetProfileByEmail(string email) | Get user’s profile by user’s email |
| 06 | GetProfileByPhone(string phone) | Get user’s profile by user’s phone |
| 07 | getPeopleFollowByAccountId(int accountId) | Get people follow an account by account id |

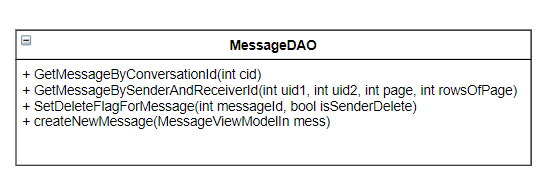
***6.3.3 PostDAO***

******

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetAllPost(int page, int rowsOfPage) | Get all public post in the system |
| 02 | GetAllPostByInfoID(int infoID, int page, int rowsOfPage) | Get all public post have body measurement equal infor id |
| 03 | CreatePost(Post post) | Create an post |
| 04 | GetPostByID(int id) | Get an post by its id |
| 05 | GetFashionistaPost(int page, int rowsOfPage) | Get fashionista’s post |
| 06 | GetAllPublicPostOfUser(int accountID, int page, int rowsOfPage) | Get all public post of user in the system |
| 07 | DeletePostByPostID(int postID) | Delete a post by its id |
| 08 | UpdatePost(int id, string content, int privacyID, DateTime time, int bodyInfoID, bool isVerified) | Update a post |
| 09 | SearchPostByText(string keyWord, int page, int rowsOfPage) | Search a post by keyword |
| 10 | CountPostOfUser(int accountID) | Get number of an user’s post |

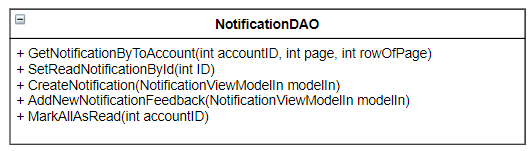
***6.3.4 MessageDAO***

****

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetMessageByConversationId(int cid) | Get messages by conversation id |
| 02 | GetMessageBySenderAndReceiverId(int uid1, int uid2, int page, int rowsOfPage) | Get messages by sender and receiver |
| 03 | SetDeleteFlagForMessage(int messageId, bool isSenderDelete) | Delete an message in an conversation |
| 04 | createNewMessage(MessageViewModelIn mess) | Create a new message then create an conversation if it not belong into an existed conversation |

***6.3.5 NotificationDAO***

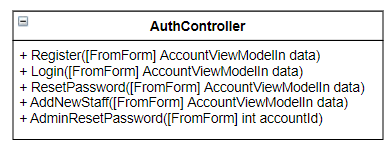
****

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetNotificationByToAccount(int accountID, int page, int rowOfPage) | Get messages by conversation id |
| 02 | SetReadNotificationById(int ID) | Get messages by sender and receiver |
| 03 | CreateNotification(NotificationViewModelIn modelIn) | Delete an message in an conversation |
| 04 | AddNewNotificationFeedback(NotificationViewModelIn modelIn) | Create a new message then create an conversation if it not belong into an existed conversation |
| 05 | MarkAllAsRead(int accountID) |  |

**6.4 Controller**

***6.4.1 AuthController***

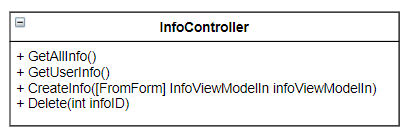
******

**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | Register([FromForm] AccountViewModelIn data) | Register for an account |
| 02 | Login([FromForm] AccountViewModelIn data) | Login for an user to the system |
| 03 | ResetPassword([FromForm] AccountViewModelIn data) | Reset password for an user |
| 04 | AddNewStaff([FromForm] AccountViewModelIn data) | Add new staff for admin web server |
| 05 | AdminResetPassword([FromForm] int accountId) | Reset password for admin |

## 

## *6.4.2 InfoController*

****

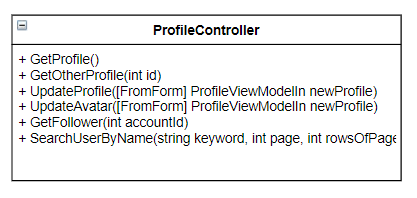
**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetAllInfo() | Get all body measurement of all users |
| 02 | GetUserInfo() | Get body measurement of an user |
| 03 | CreateInfo([FromForm] InfoViewModelIn infoViewModelIn) | Create an body measurement for an user |
| 4 | Delete(int infoID) | Delete an body measurement of an user |

## 

## 

## *6.4.3 ProfileController*

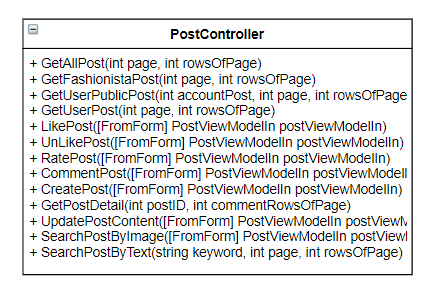


**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetProfile() | Get user’s profile |
| 02 | GetOtherProfile(int id) | Get another user’s profile |
| 03 | UpdateProfile([FromForm] ProfileViewModelIn newProfile) | Update user’s profile |
| 04 | UpdateAvatar([FromForm] ProfileViewModelIn newProfile) | Update user’s avatar |
| 05 | GetFollower(int accountId) | Get follower of user |
| 06 | SearchUserByName(string keyword, int page, int rowsOfPage) | Search user by name |

## 

## *6.4.4 PostController*

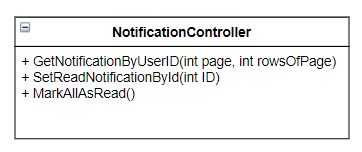


**Class specification**

|  |  |  |
| --- | --- | --- |
| NO | Method | Description |
| 01 | GetAllPost(int page, int rowsOfPage) | Get all users’s posts |
| 02 | GetFashionistaPost(int page, int rowsOfPage) | Get fashionistas’s posts |
| 03 | GetUserPublicPost(int accountPost, int page, int rowsOfPage) | Get public user’s posts |
| 05 | LikePost([FromForm] PostViewModelIn postViewModelIn) | Like a post |
| 06 | UnLikePost([FromForm] PostViewModelIn postViewModelIn) | Unlike a post |
| 07 | RatePost([FromForm] PostViewModelIn postViewModelIn) | Rate a post |
| 08 | CommentPost([FromForm] PostViewModelIn postViewModelIn) | Comment a post |
| 09 | CreatePost([FromForm] PostViewModelIn postViewModelIn) | Create a new post |
| 10 | GetPostDetail(int postID, int commentRowsOfPage) | Get detail of a post |
| 11 | UpdatePostContent([FromForm] PostViewModelIn postViewModelIn) | Update a post’s content |
| 12 | SearchPostByImage([FromForm] PostViewModelIn postViewModelIn) | Search a post by Image |
| 13 | SearchPostByText(string keyword, int page, int rowsOfPage) | Search a post by keyword |

## 

## *6.4.5 NotificationController*

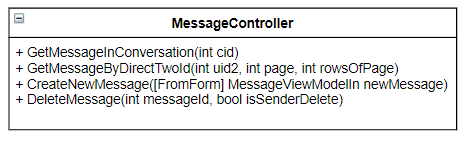


**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetNotificationByUserID(int page, int rowsOfPage) | Get notification of user |
| 02 | SetReadNotificationById(int ID) | Set a notification is read |
| 03 | MarkAllAsRead() | Set all notifications are read |

## 

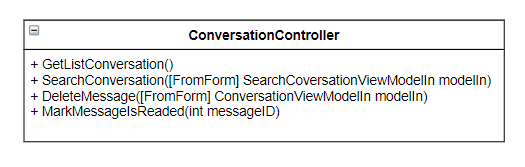
## *6.4.6 MessageController*



**Class specification**

|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetMessageInConversation(int cid) | Get messages in a conversation |
| 02 | GetMessageByDirectTwoId(int uid2, int page, int rowsOfPage) | Get messages by user |
| 03 | CreateNewMessage([FromForm] MessageViewModelIn newMessage) | Send a message to another user |
| 04 | DeleteMessage(int messageId, bool isSenderDelete) | Delete a message |

## *6.4.7 ConversationController*

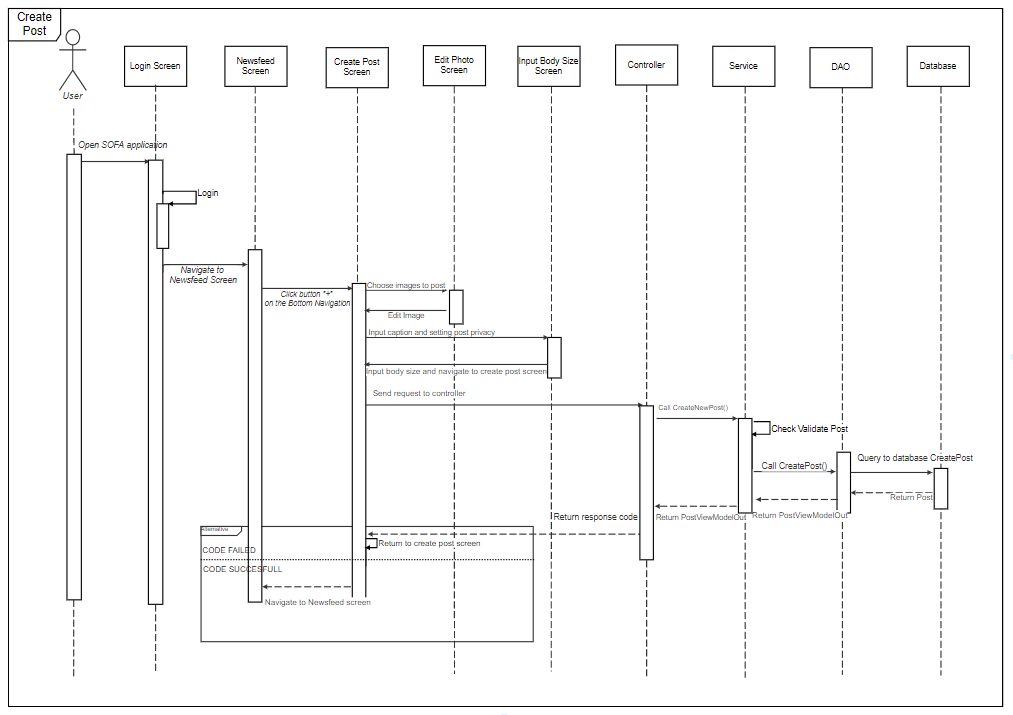


**Class specification**

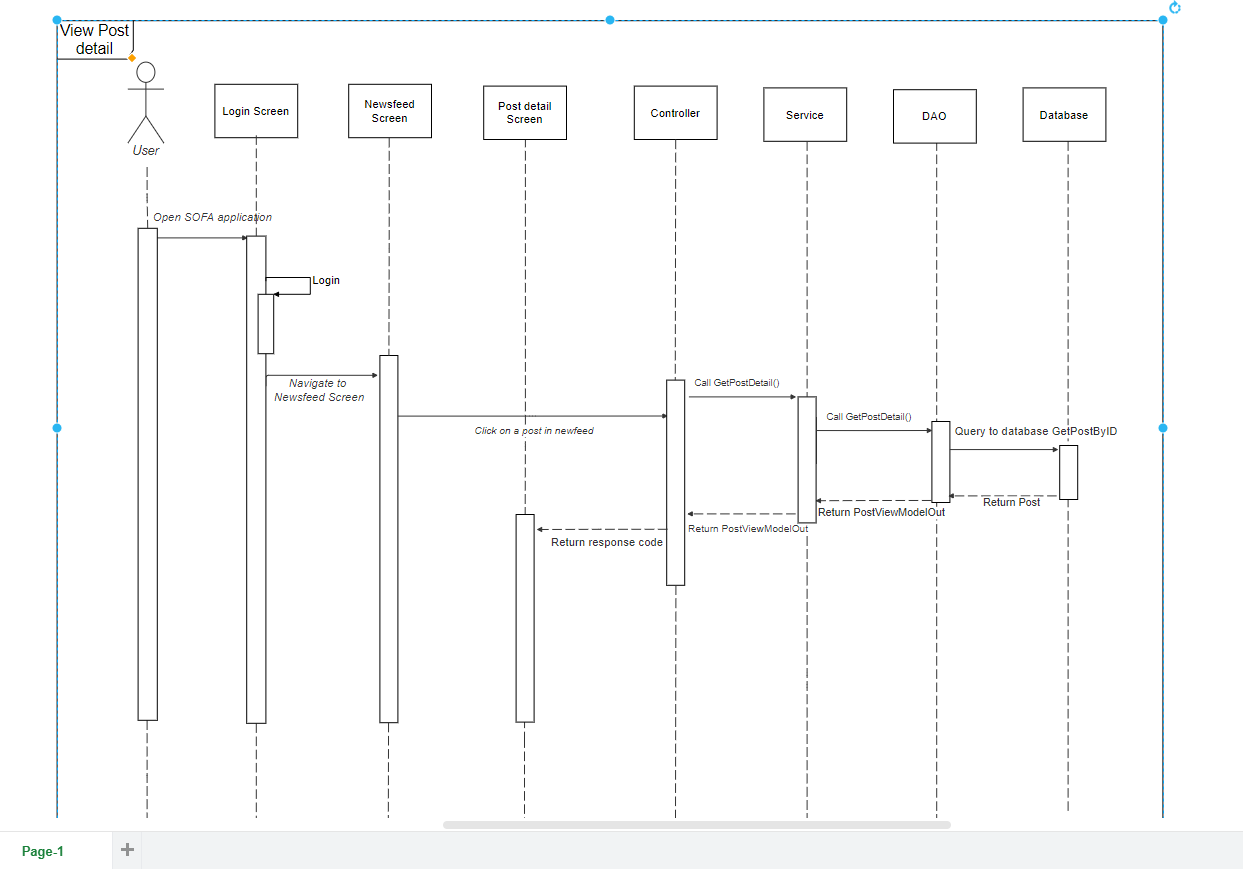
|  |  |  |
| --- | --- | --- |
| **NO** | **Method** | **Description** |
| 01 | GetListConversation() | Get list user’s conversation |
| 02 | SearchConversation([FromForm] SearchCoversationViewModelIn modelIn) | Search conversation by username |
|  | DeleteMessage([FromForm] ConversationViewModelIn modelIn) | Delete a conversation |
|  | MarkMessageIsReaded(int messageID) | Mark conversation is read |

## 7. Sequence Diagram

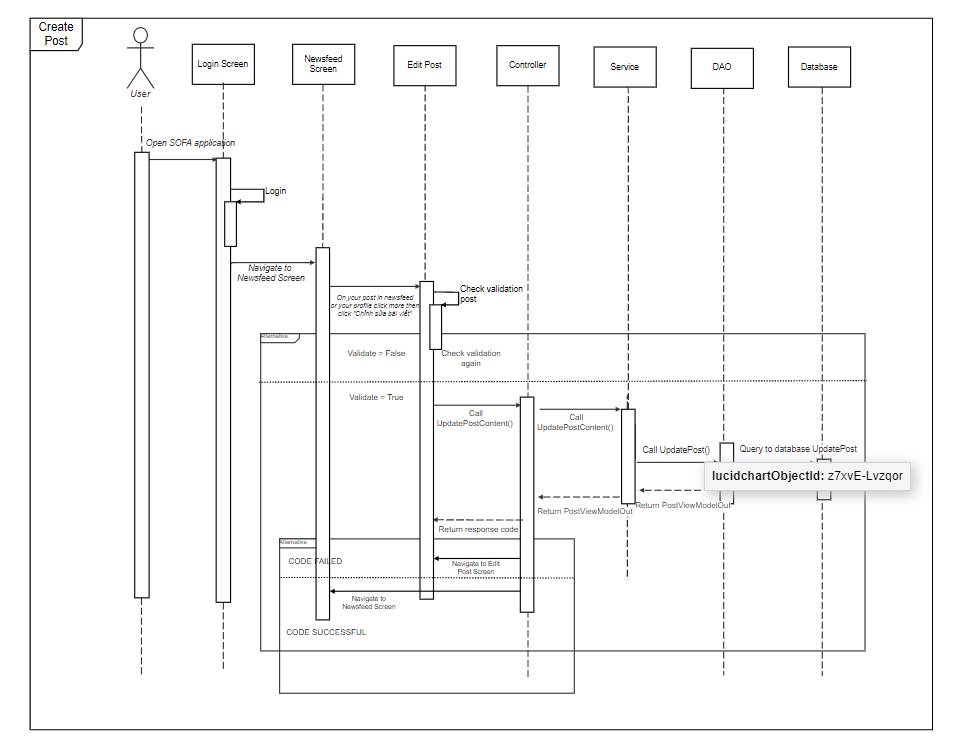
***7.1. Create New Post***

******

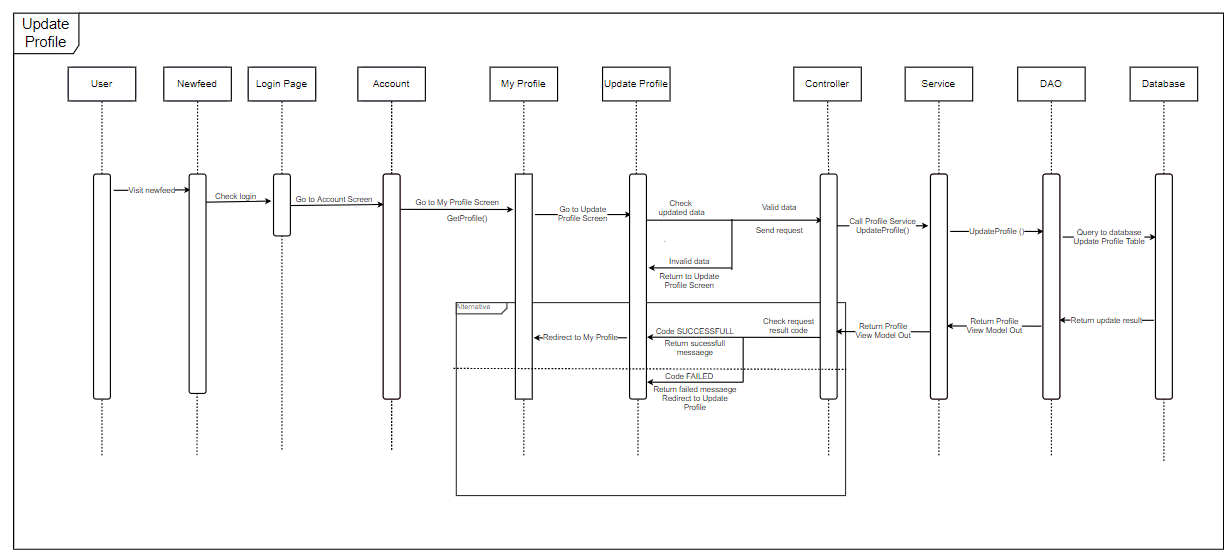
***7.2. View post detail***

******

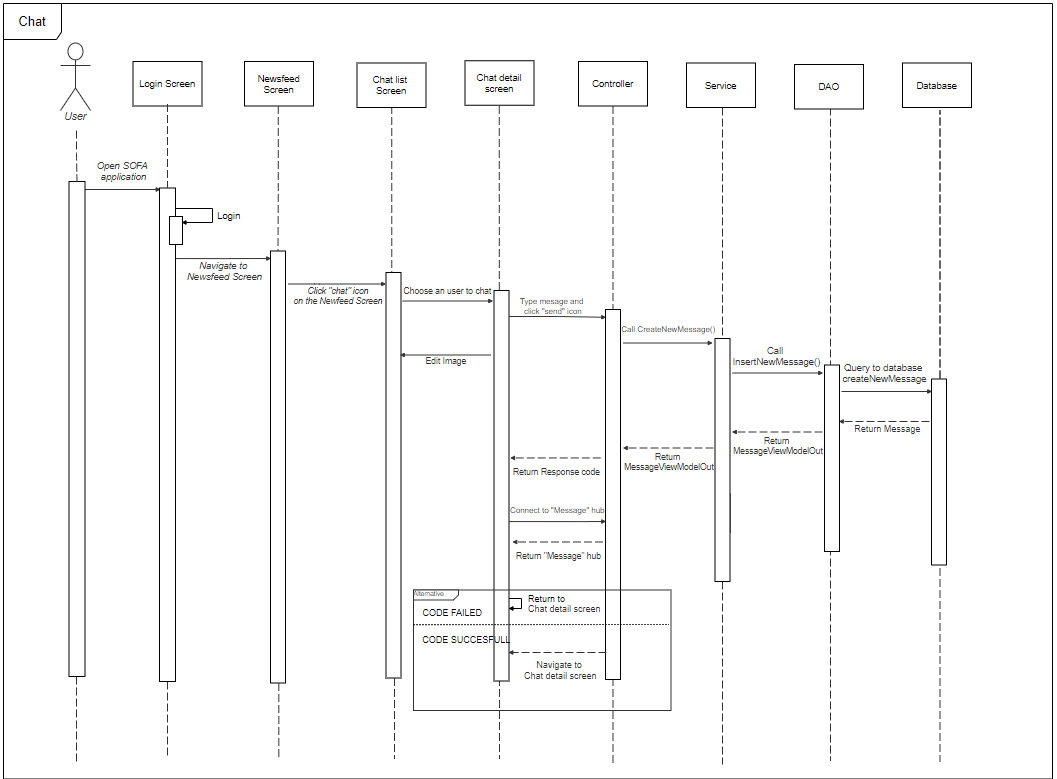
***7.3. View edit post***

******

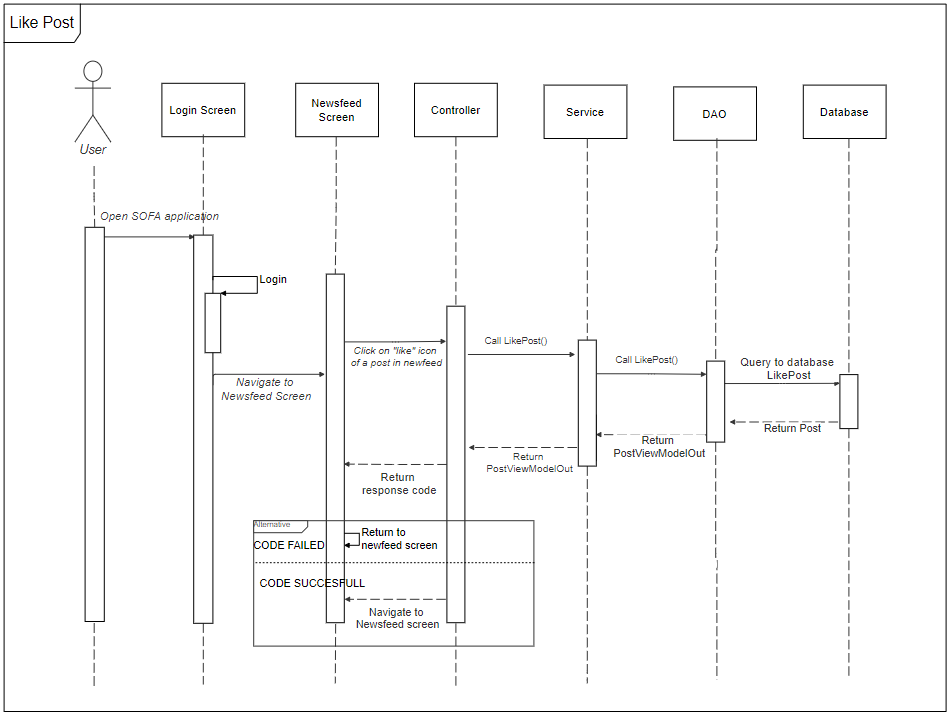
***7.4. Update Profile***

******

***7.5. Chat***

******

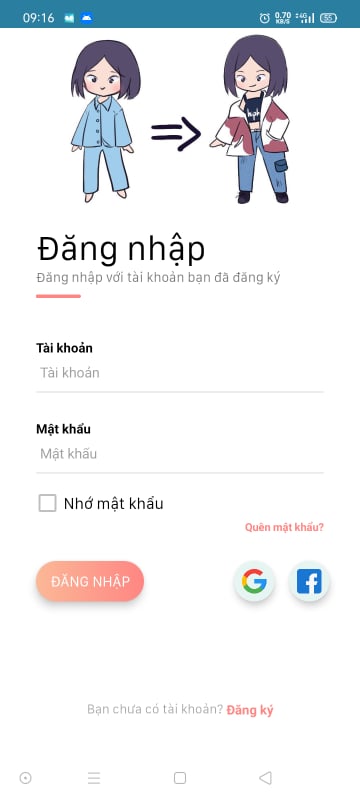
***7.6. Like Post***

******

## User Interface Design

## *8.1. Screen Image*

8.1.1 Login



8.1.2. Register



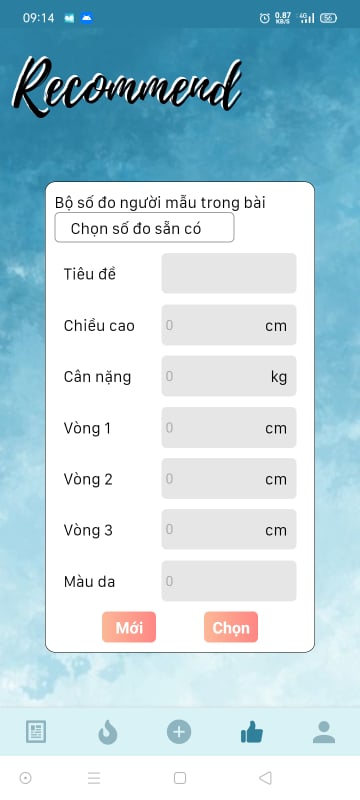
8.1.3. Newsfeed



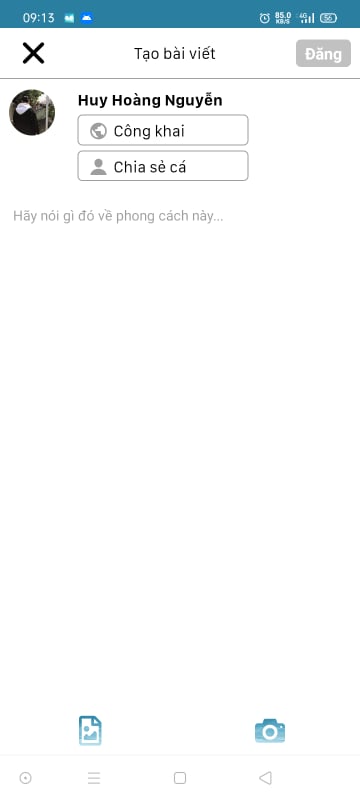
8.1.4 Hot

## 

8.1.5 Recommend



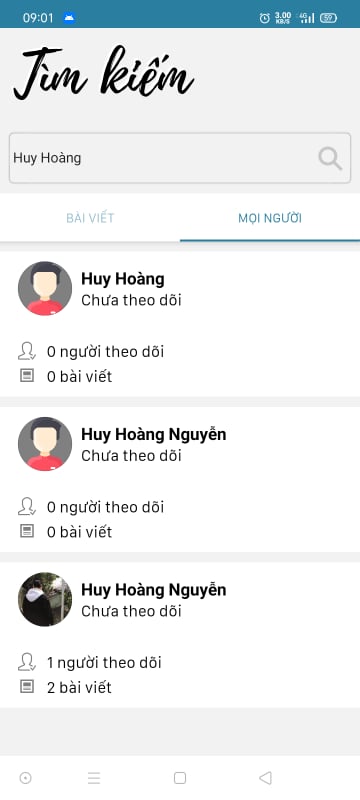
8.1.6 Create Post

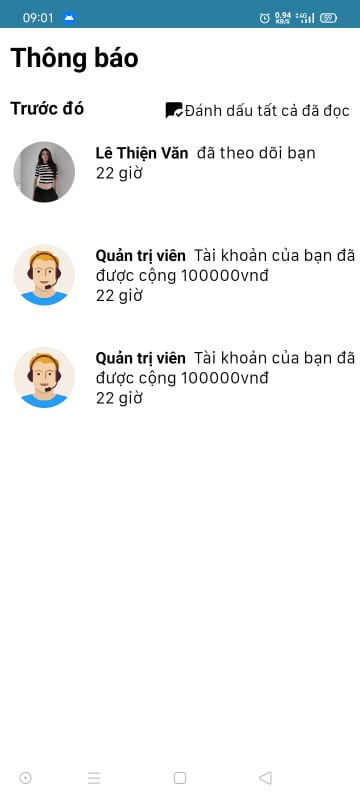


8.1.7. Post Detail

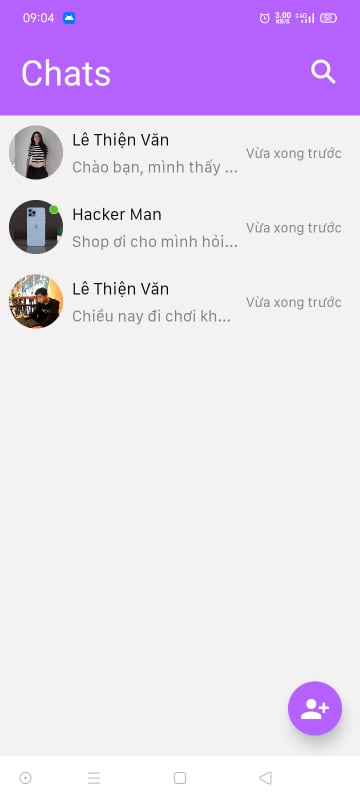


8.1.8 Search

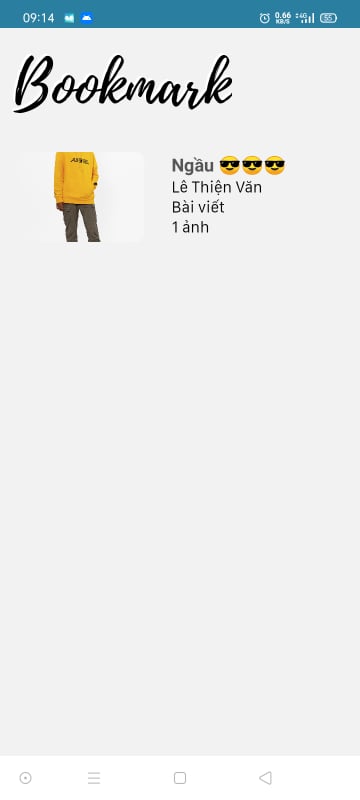


8.1.9 Notification

8.1.10 Chat



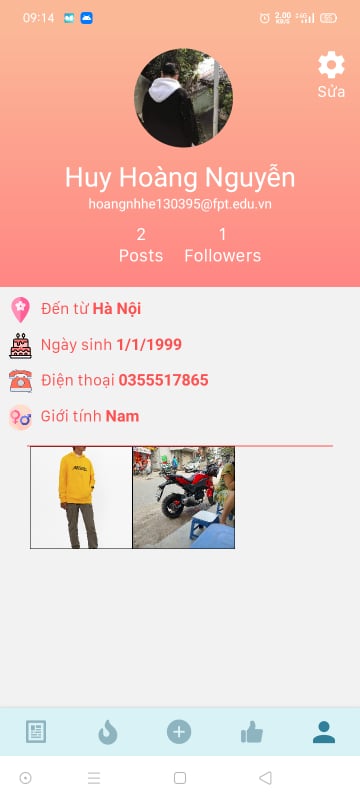
8.1.11 Bookmark



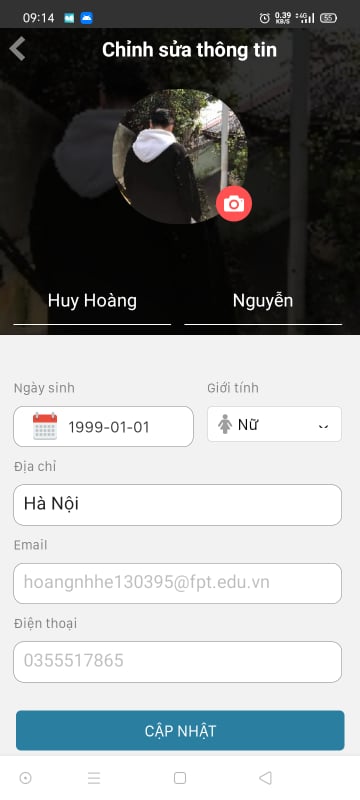
8.1.12 Account



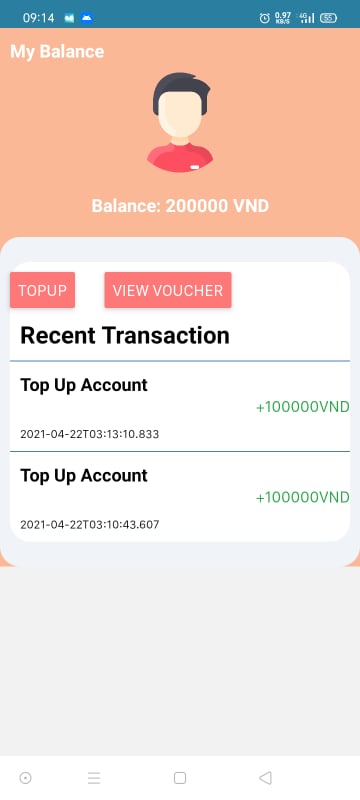
8.1.13 Profile



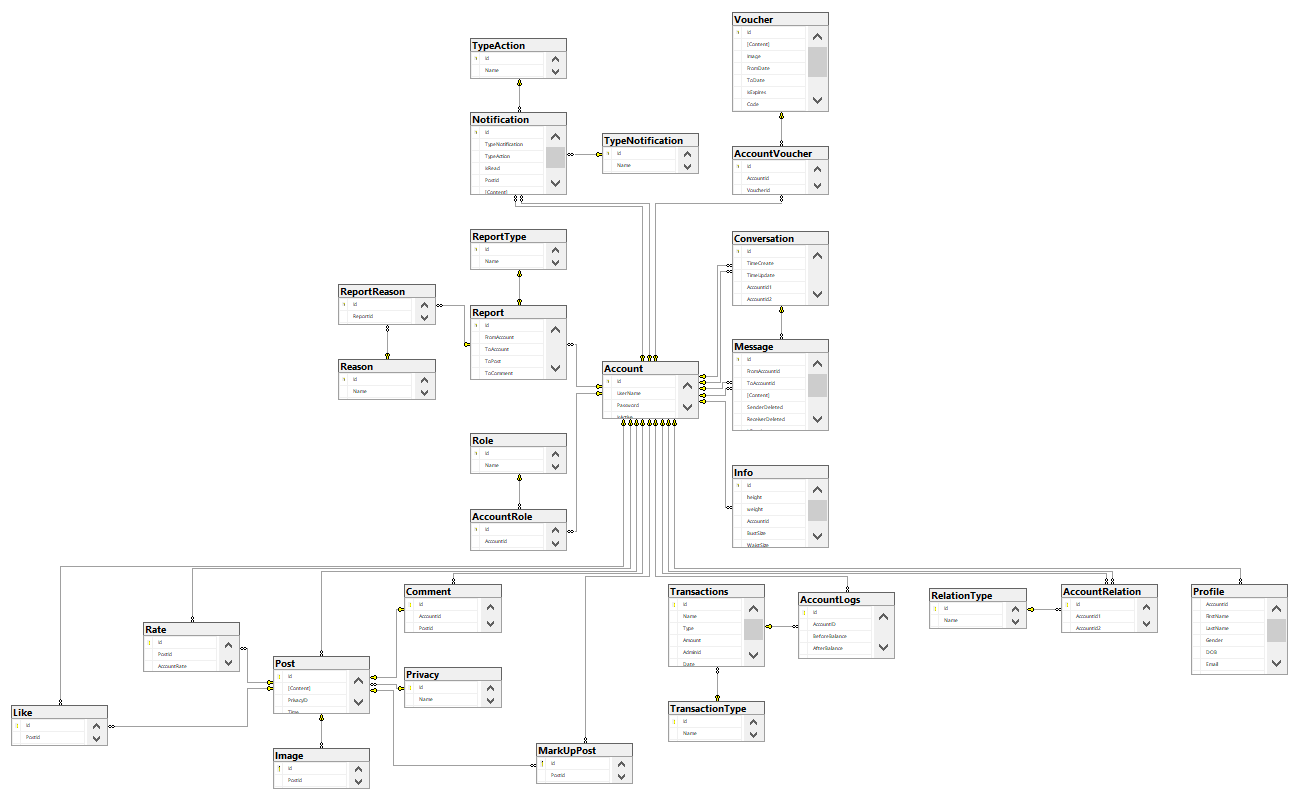
8.1.14 Edit Profile



8.1.15 Balance



## Database Design

**