Object Oriented Analysis and Design (INT3110)

Course Project

Tinder.com

Project team

Hoang Son Tung - 17021351

Nguyen Van Manh - 16020046

Do Thi Hong Ngat - 17020061

Revision History

Name	Date	Reason For Changes	Version
Tung	17/10/2019	Initialize problem, specifications	1.0
Tung, Ngat, Manh	1/11/2019	Finished Use-case Model	1.1
Tung, Ngat, Manh	10/11/2019	Finished Use-case Analysis	1.2
Tung, Ngat, Manh	20/11/2019	Finished Use-case Design & Finializing	2.0

Table of Contents

1. Requirements

1.1. Problem Statement

1.1.1 The Problem

In the modern world, people are becoming increasingly oriented towards being exceptional at one domain or professional to optimize the quality of their work. As a consequence, a lot of people do not have sufficient social skills to develop a relationship for themselves. Moreover, it is usually hard for them to find the time to develop these important skills and relationships which constitute a large part of their fulfilling life

1.1.2 The Solution

1.2. Glossary

1.3. Supplementary specifications

1.3.1 Objectives

The purpose of this document is to define the requirements of the Tinder.com system. This Supplementary Specification lists the requirements that are not rapidly captured in the use case of the use-case model. The Supplementary Specification and the use-case model together capture a complete set of requirements on the system.

1.3.2 Scope

- This Supplementary Specification applies to the Tinder.com system, which is an online dating website worldwide.
- This specification defines the non-functional requirements of the system: such as reliability, usability, performance, and supportability, as well as functional requirements that are common across a number of use cases. (The functional requirements are defined in the Use Case Specification.)

1.3.3 References

- None.

1.3.4 Functionality

- Multiple users must be able to perform their work concurrently.

1.3.5 Usability

- The software must be easy to use so that a new user can learn how to use the system within a few minutes.
- The user interface has to be friendly and intuitive.

1.3.6 Reliability

The system must be available 24 hours a day, 7 days a week. The system must also have less than 1% downtime.

1.3.7 Performance

- The system shall support up to ~ 100 millions simultaneously users against a central database, and up to ~10 simultaneous users against the local servers at any given time.
- The system shall provide access to the database with no more than 5ms seconds latency.
- The system must be able to complete 99,9% of all transactions within 1 seconds.

1.3.8 Supportability

- None.

1.3.9 Security

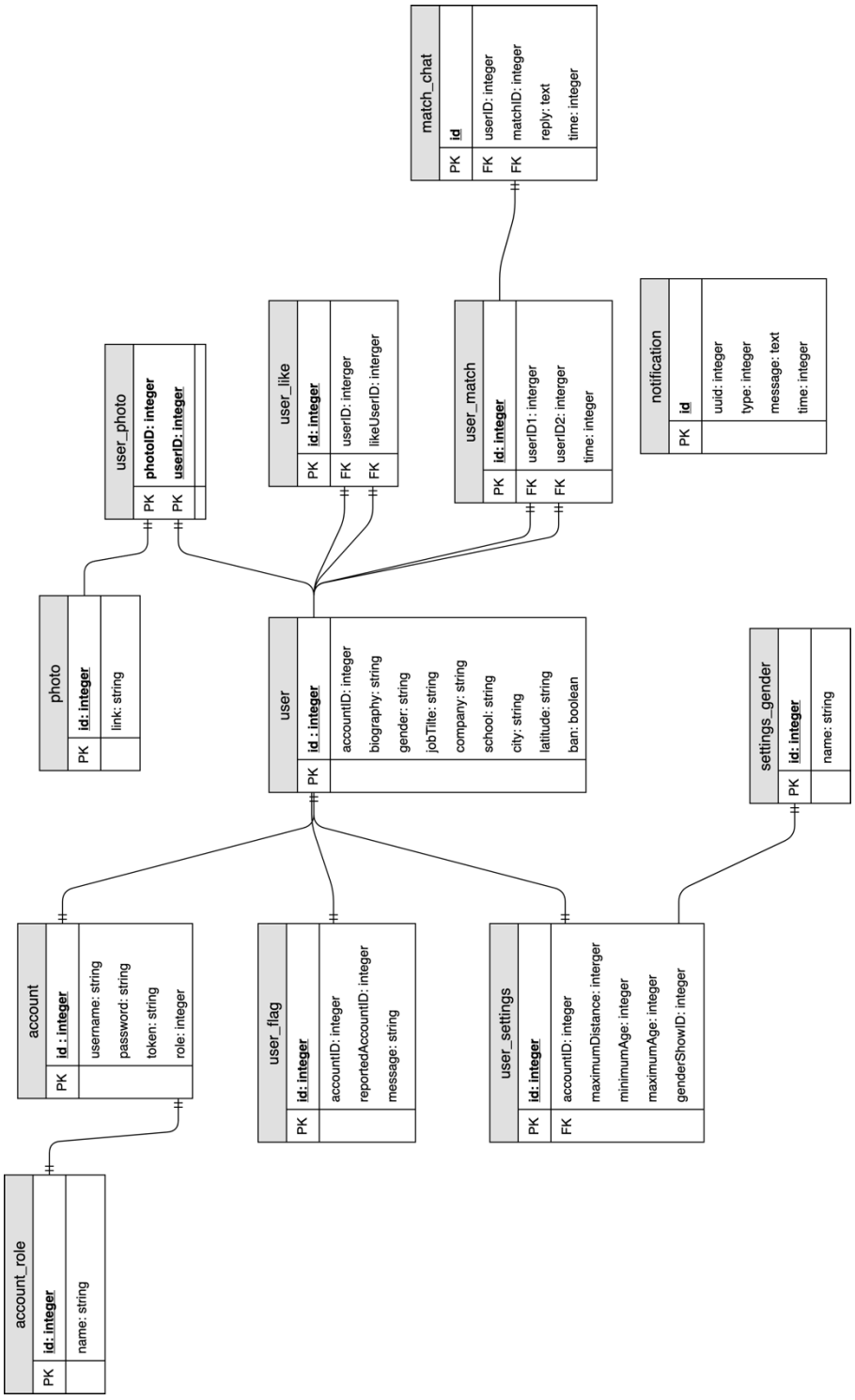
- The system must prevent users from logging in if they do not have a password.
- Only Administrators can delete User accounts.
- Profile can only be edited by its owner and deleted by its owner or approved by a System Administrator.

1.3.10 Design Constraints

- The system must provide a responsive web-based interface usable on computers and mobile devices & mobile apps for iOS and Android.

1.4. Use-Case Model

1.4.1 Diagrams



1.4.2 Login

1.4.2.1 Brief Description

This use case describes how a user login Tinder

1.4.2.2 Flow of Events

1.4.2.2.1 Basic flows

This use case starts when the Visitor requests to create an account on the website

- The system displays a form that asks the Visitor to enter either his/her Phone Number or Email Address
- The system validates the entered username and password and logs the actor into the system

Invalid username/password

If the Visitor enters an invalid username and/or password, the system displays an error message. the Visitor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.

Signup new account

If the Visitor hasn't had the account to log into the system, he/she can choose to sign up by filling the form

Forgot password

If the Visitor hasn't had the account to log into the system, he/she can choose to sign up by filling the form

1.4.2.2.2 Alternative flows

Invalid Information

If any of the above fields (except for self-introduction) are not filled in, the system displays an error message. The Visitor can continue making changes to the registration form or cancel the registration, at which point the use case ends.

User Already Exists

If the specified user Email or Phone Number already exists, the system displays an error message. The Visitor is redirect to Login use case

1.4.2.3 Special Requirements

None.

1.4.2.4 Pre-Conditions

None.

1.4.2.5 Post-Conditions

If the use case was successful, a new user is added to the system. Otherwise, the system state remains unchanged.

1.4.2.6 Extension Points

None.

1.4.3 Sign Out

1.4.3.1 Brief Description

This use case describes how a User or an Administrator sign out of the Application.

1.4.3.2 Flow of Events

1.4.3.2.1 Basic flows

This use case starts when the actors tap on the sign out button

1. The system prompt the user 2 options:
    - Forget me on this device
    - Remember me on this device
2. The system sign out the current actor on the device

1.4.3.2.2 Alternative flows

Operation cancelled

User cancel the sign out in the verification prompt, the user will not be sign out of the Application

1.4.3.2.3 Special Requirements

None.

1.4.3.2.4 Pre-Conditions

The system is in the login state and has the logged-in screen displayed.

1.4.3.2.5 Post-Conditions

If the use case is successful, the actor is now logged in to the system. Otherwise, the system state is unchanged.

1.4.3.2.6 Extension Points

None.

1.4.4 Update Account

1.4.4.1 Brief Description

This use case describes how a User or an Administrator update their account.

1.4.4.2 Flow of Events

1.4.4.2.1 Basic flows

This use case starts when User/Administrator tap on Update Account. The system prompt the User/Administrator 2 options:

- Change password
- Remove Account

Change Password

If the Visitor want to change his password, he/she can fill the form and tap Done

Remove Account

If the Visitor want to remove his account, he/she can tap Remove Account and make a confirmation

1.4.4.2.2 Alternative flows

None.

1.4.4.2.3 Special Requirements

None.

1.4.4.2.4 Pre-Conditions

The system is in the login state and has the logged-in screen displayed.

1.4.4.2.5 Post-Conditions

If the use case is successful, the actor is now log out the system. Otherwise, the system state is unchanged.

1.4.4.2.6 Extension Points

None.

1.4.5 User Update Profile

1.4.5.1 Brief Description

This use case describes how a user update his/her profile

1.4.5.2 Flow of Events

1.4.5.2.1 Basic flows

This use case starts when user tap on his/her profile and choose Update Profile or Settings

Edit Information

- The system show the editing page for his/her profile
- The user edit his profile field by field (Name, Gender, Age, Interest, Self-description, etc..)
- User tap "Update Profile", the edited profile will be received by the system

Update Settings

- The system show the settings page for his/her profile
- The user can update settings by field (Distance, Show Gender, Age Range, Notifications, etc..)
- User tap "Done", the settings will be received by the system

1.4.5.2.2 Alternative flows

Operation cancelled

Actor cancel the update in the verification prompt, the profile will not be saved to the system

1.4.5.3 Special Requirements

None.

1.4.5.4 Pre-Conditions

The user must be logged in to the system

1.4.5.5 Post-Conditions

If the use case is successful, the edited profile is written to the system. Otherwise, the profile is unchanged

1.4.5.6 Extension Points

None.

1.4.6 Like/Dislike Profile

1.4.6.1 Brief Description

This use case describes the flow when a user show Interest/Disinterest in another user

1.4.6.2 Flow of Events

1.4.6.2.1 Basic flows

This use case starts when (while browsing other users) a user swipe left/right on a profile. Swipe Left means disinterest someone and Swipe Right means interest someone

- The system updates the interest information
- The system shows another random other user's profile
- The user keep browsing other profiles

1.4.6.2.2 Alternative flows

None.

1.4.6.3 Special Requirements

None.

1.4.6.4 Pre-Conditions

The user must be logged in to the system

1.4.6.5 Post-Conditions

If the use case is successful, the "Interested list" the being-interested user will be updated to the system. The current user will be shown another profile to keep browsing.

1.4.6.6 Extension Points

None.

1.4.7 Match Profile

1.4.7.1 Brief Description

This use case describes how two people match each other's choices

1.4.7.2 Flow of Events

1.4.7.2.1 Basic flows

This use case starts when two people both interested in one another (swipe right/like)

- The system notifies both users about the other matching person:
- Users can either choose to send messages to the other person or keep swiping
- The matching person will be stored to users message box

1.4.7.2.2 Alternative flows

None

1.4.7.3 Special Requirements

None.

1.4.7.4 Pre-Conditions

- User has logged in to the app
- 2 User has liked each other

1.4.7.5 Post-Conditions

The matching users will be stored in each other's inbox

1.4.7.6 Extension Points

None.

1.4.8 Chat

1.4.8.1 Brief Description

This use case describes when one user want to chat with their matches

1.4.8.2 Flow of Events

1.4.8.2.1 Basic flows

This use case starts when a user chooses a match from his/her inbox and start to chat. User start type message on typing box and send message as usual.

1.4.8.2.2 Alternative flows

None.

1.4.8.3 Special Requirements

None.

1.4.8.4 Pre-Conditions

2 people must have matched before

1.4.8.5 Post-Conditions

If use case execute successfully, message will be sent to server. Otherwise nothing will happen.

1.4.8.6 Extension Points

None.

1.4.9 Unmatched Profile

1.4.9.1 Brief Description

This use case describes when one user unmatched his/her matched others

1.4.9.2 Flow of Events

1.4.9.2.1 Basic flows

This use case starts when a user chooses a match from his/her inbox and set it to unmatched

1. The app prompts the unmatched-requesting user a questionnaire about his/her decision to unmatched the other, 5 options:
    - Inappropriate messages
    - Inappropriate photos
    - Feels like spam
    - Other
    - No Reason
2. Once the requesting user has filled in the reason, the requested user will be removed from the requesting user's inbox

1.4.9.2.2 Alternative flows

User choose "Other"

Application prompt another form for user to fill in their reason

1.4.9.3 Special Requirements

None.

1.4.9.4 Pre-Conditions

The unmatched-requesting user must have the unmatched-requested user in his/her inbox. 2 User must have matched before.

1.4.9.5 Post-Conditions

The reason for unmatched will be submitted to the server, and the requested user will be unmatched (remove) from the requesting user's inbox

1.4.9.6 Extension Points

None.

1.4.10 Review Inappropriate Account

1.4.10.1 Brief Description

This use case describes how an Administrator views the list of reported users and administrator decide to ban or remove a user

1.4.10.2 Flow of Events

1.4.10.2.1 Basic flows

This use case starts when an administrator, after review, has decided that the user is toxic, thus ban or remove that user

1. The system displays user's profile and the associated reports, it also displays the "Ban" button

2. If the Administrator selects "Ban", the system prompts for the number of days the user will be banned. The Administrator fill in the number, system will mark that users as disabled for that corresponding number of days

1.4.10.2.2 Alternative flows

Operation Cancelled

Administrator selects "Cancel" at any step, the profile is neither Banned

1.4.10.3 Special Requirements

None.

1.4.10.4 Pre-Conditions

The Administrator must be logged into the system.

1.4.10.5 Post-Conditions

If the use case is successful, the profile is published or rejected. Otherwise, it is unchanged.

1.4.10.6 Extension Points

None.

1.4.11 Get Insights

1.4.11.1 Brief Description

This use case describes how an Administrators starts his management session by going the Administration Dashboard

1.4.11.2 Flow of Events

1.4.11.2.1 Basic flows

This use case starts when Administrators logged in system

- The System shows show basic info (Number Accounts, System Load, Storage Load, Database Monitor,…)
- Administrators can choose User Photos and remove any toxic photos any time
- Administrators can get reports about User Using Data

1.4.11.2.2 Alternative flows

None.

1.4.11.3 Special Requirements

None.

1.4.11.4 Pre-Conditions

The Administrator must be logged into the system.

1.4.11.5 Post-Conditions

None.

1.4.11.6 Extension Points

None.

2. Use-case Analysis

2.1. Architectural Analysis

2.1.1 High-level organisation of the model

The bisides figure describes the high-level organisation of the software system. The system consists of three layers:

- The Application layer contains the design elements that are specific to each use case of the system.
- The Business Services layer encapsulates some key abstractions and services common to all use cases. It is accessible from the Application layer.
- The Middleware layer offers services to enable data communication and management on distributed system

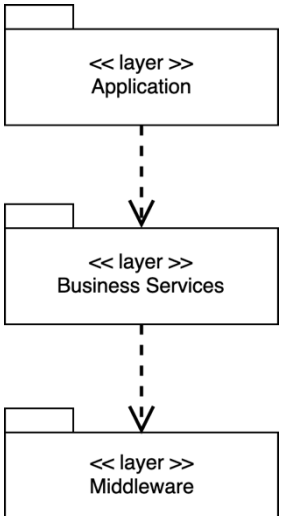


Figure 2.1.1 - Layering Approach

2.1.2 Key Abstractions

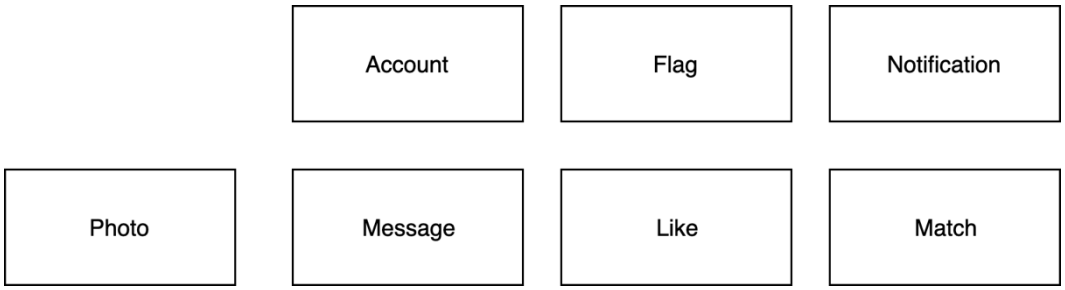


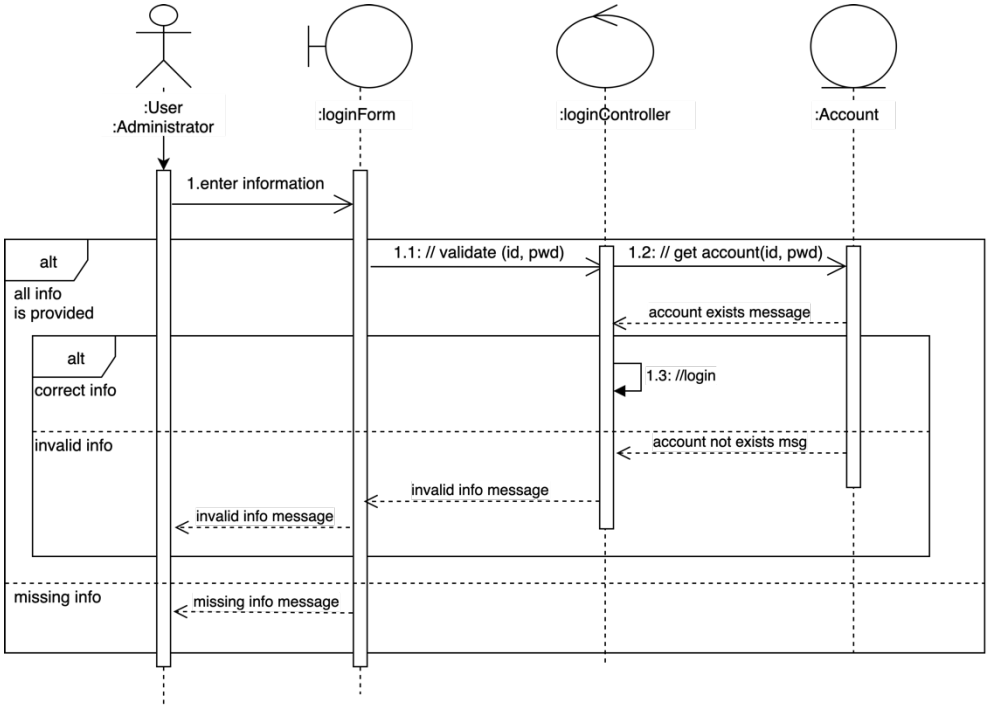
Figure 2.1.2 - Key Abstractions

- Account: A record about a user/administrator. Each account has a unique user ID and a password, which is used to identify the user/administrator and grant them access to secure parts of the system.
- Notification: An notification sent from system which was sent to device's push notifications service
- Flag: A notification sent from a registered user to the administrators about an account's violation of the website's policy.
- Photo: Records about user's photos. Only user can upload photos to system
- Message: Records about user's message to each other
- Like: Records about an user's liked profile list
- Match: Records about an user's matched profile list

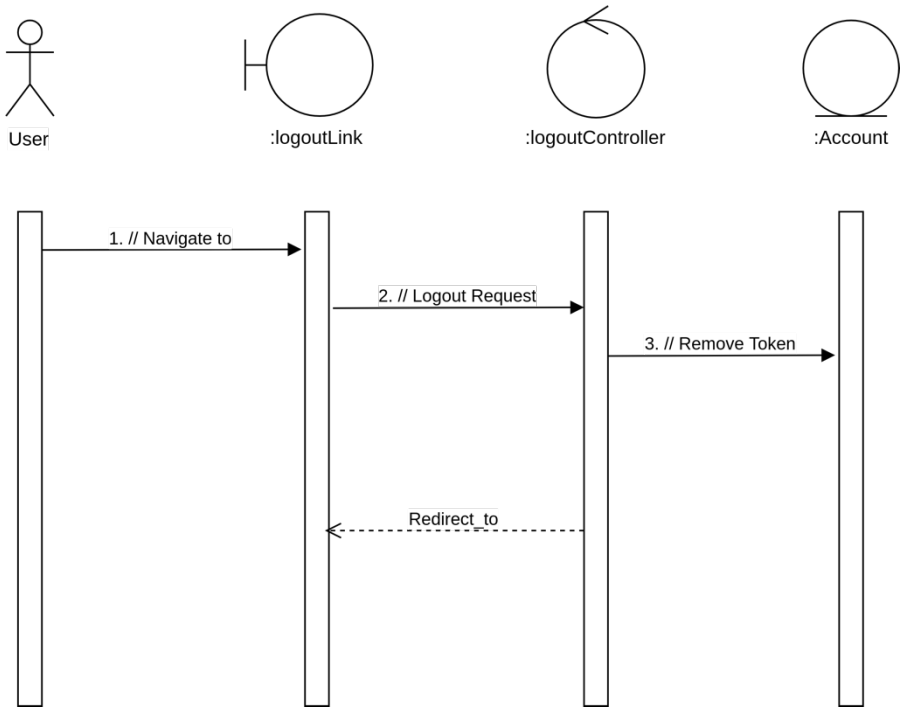
2.2. Use-case realizations

2.2.1 Use-case realizations: Sequence diagrams

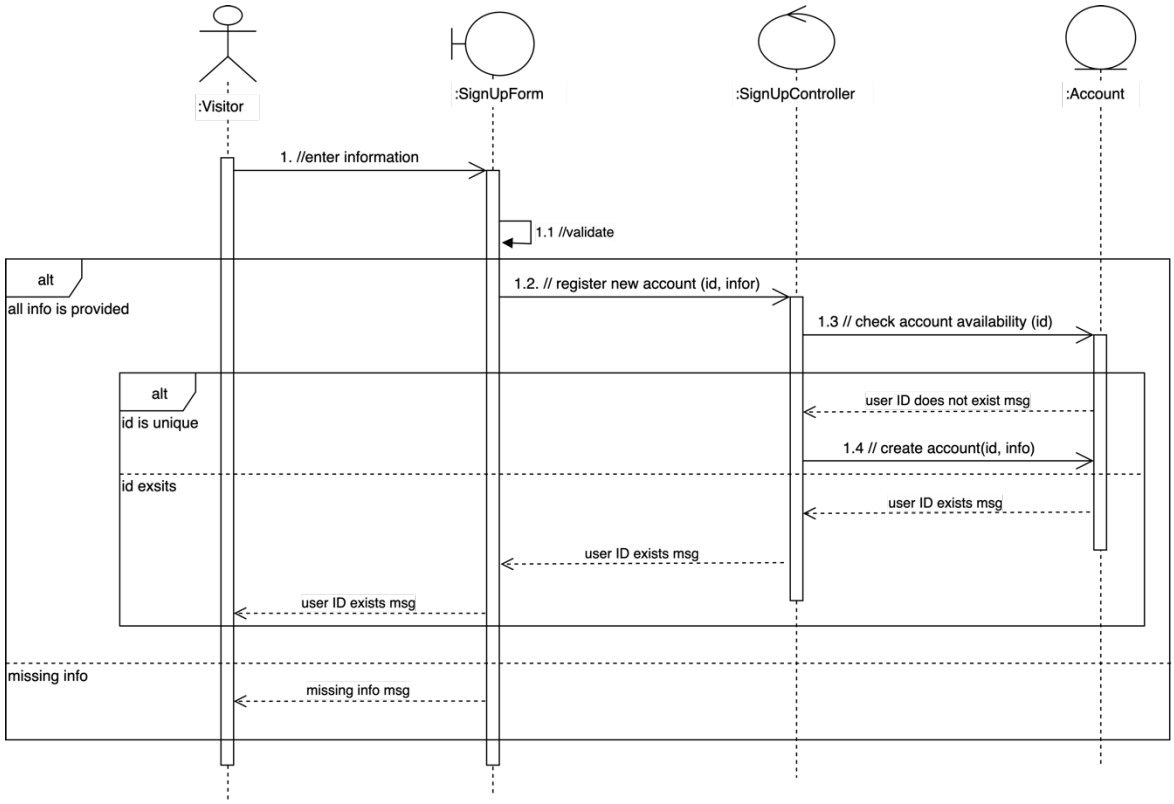
2.2.1.1 Sequence diagram for the Sign In Use Case



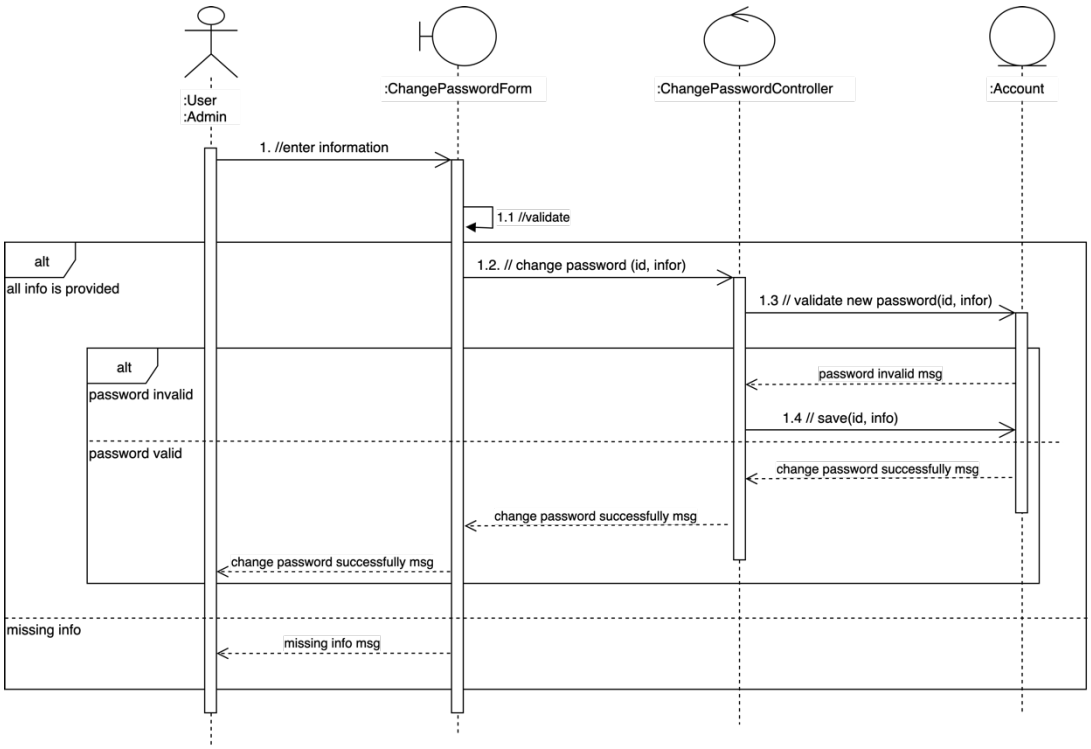
2.2.1.2 Sequence diagram for the Sign Out Use Case



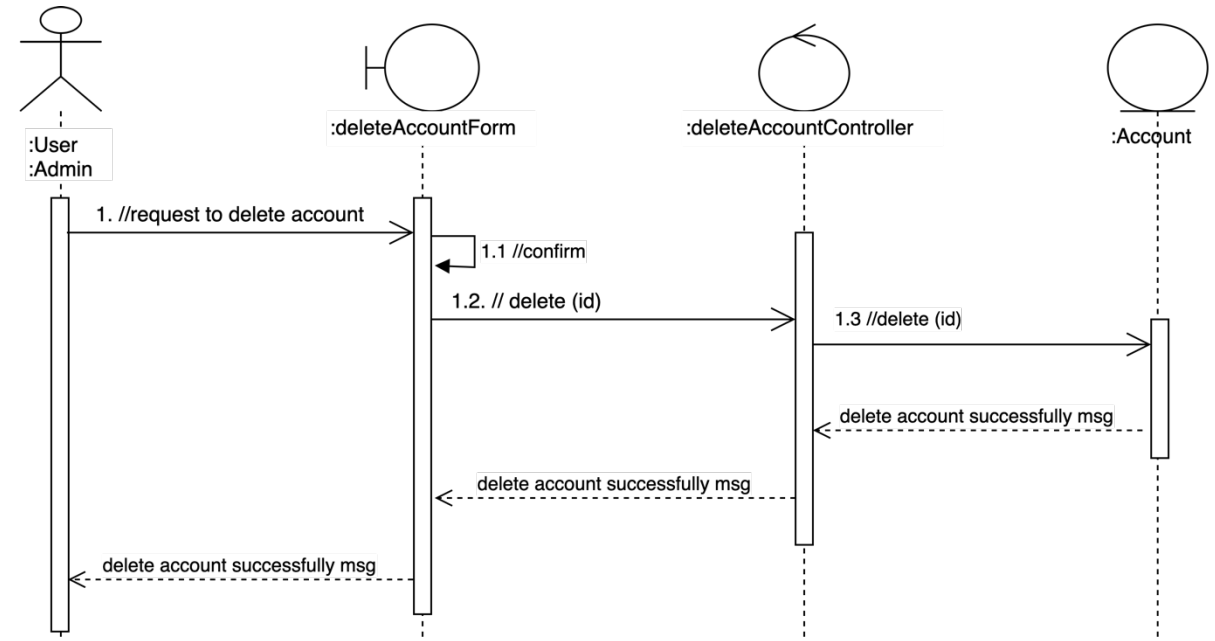
2.2.1.3 Sequence diagram for the Sign Up Use Case



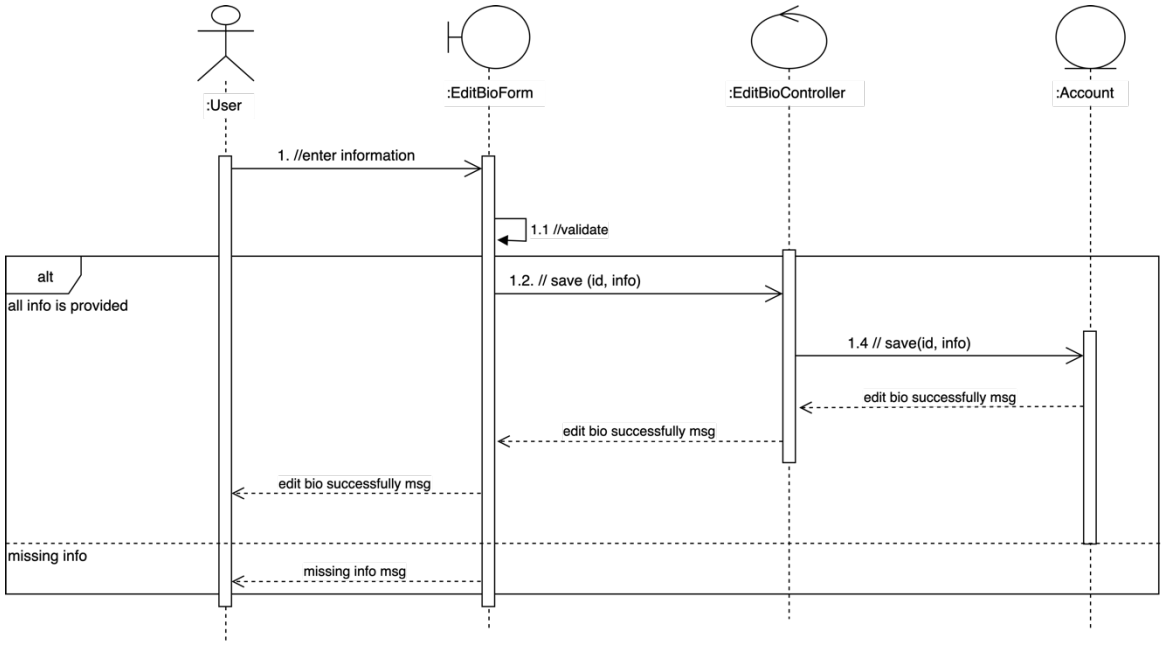
2.2.1.4 Sequence diagram for the Change Password Use Case



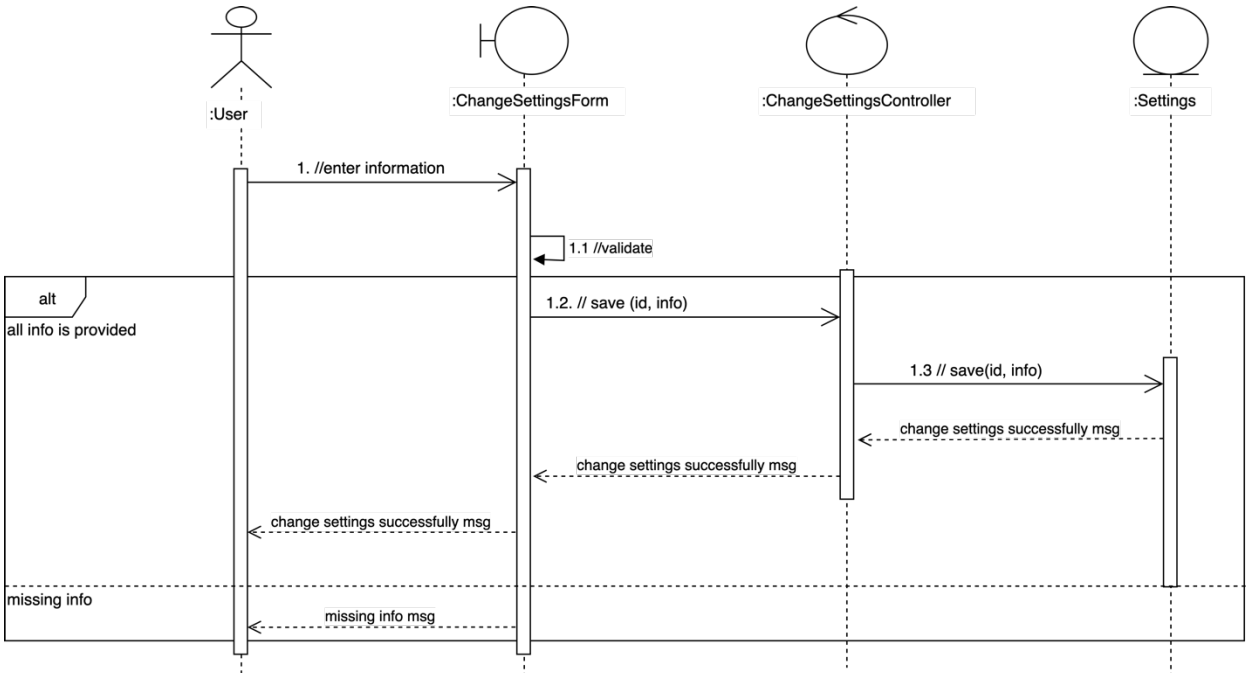
2.2.1.5 Sequence diagram for the Remove Account Use Case



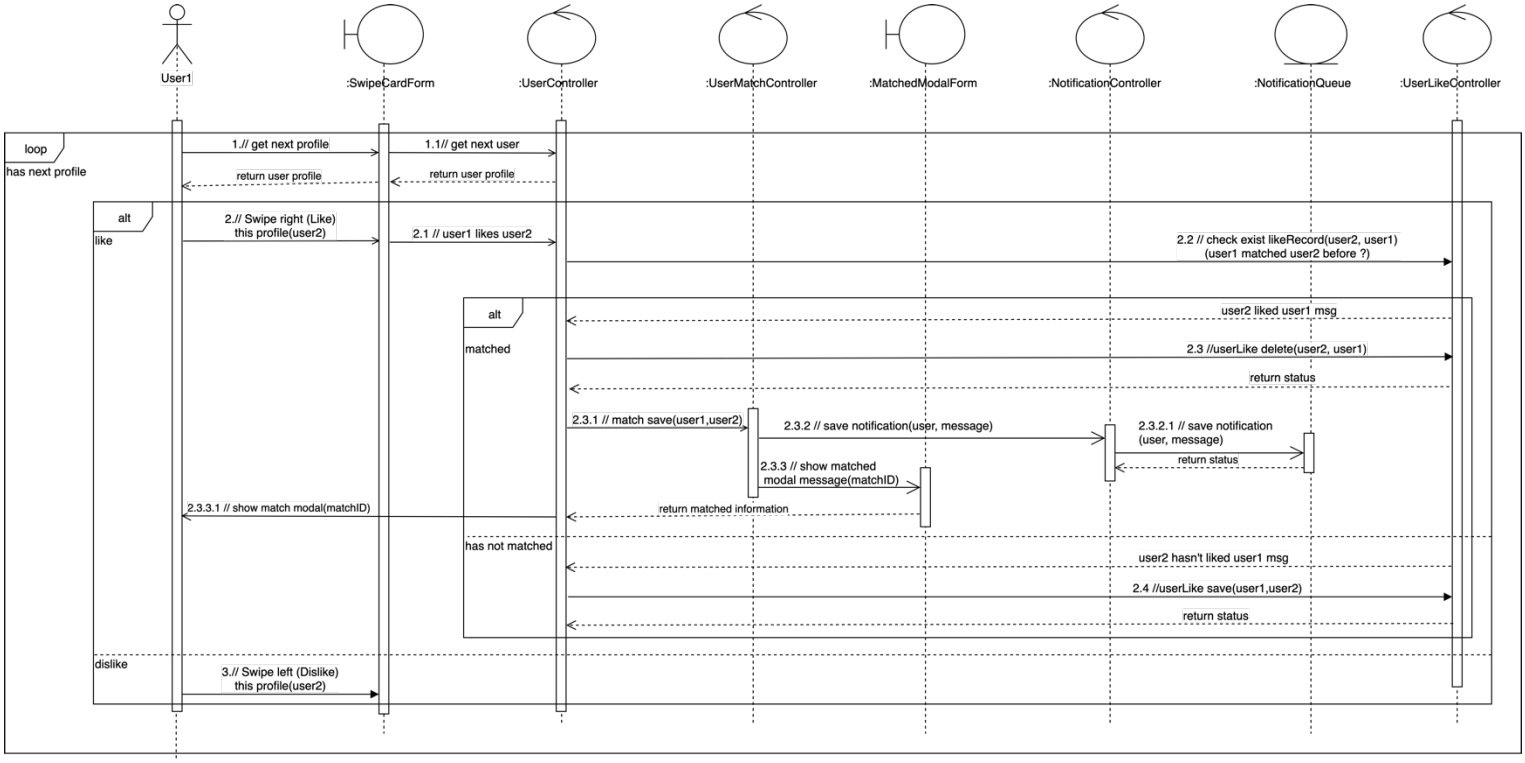
2.2.1.6 Sequence diagram for the Edit Information Use Case



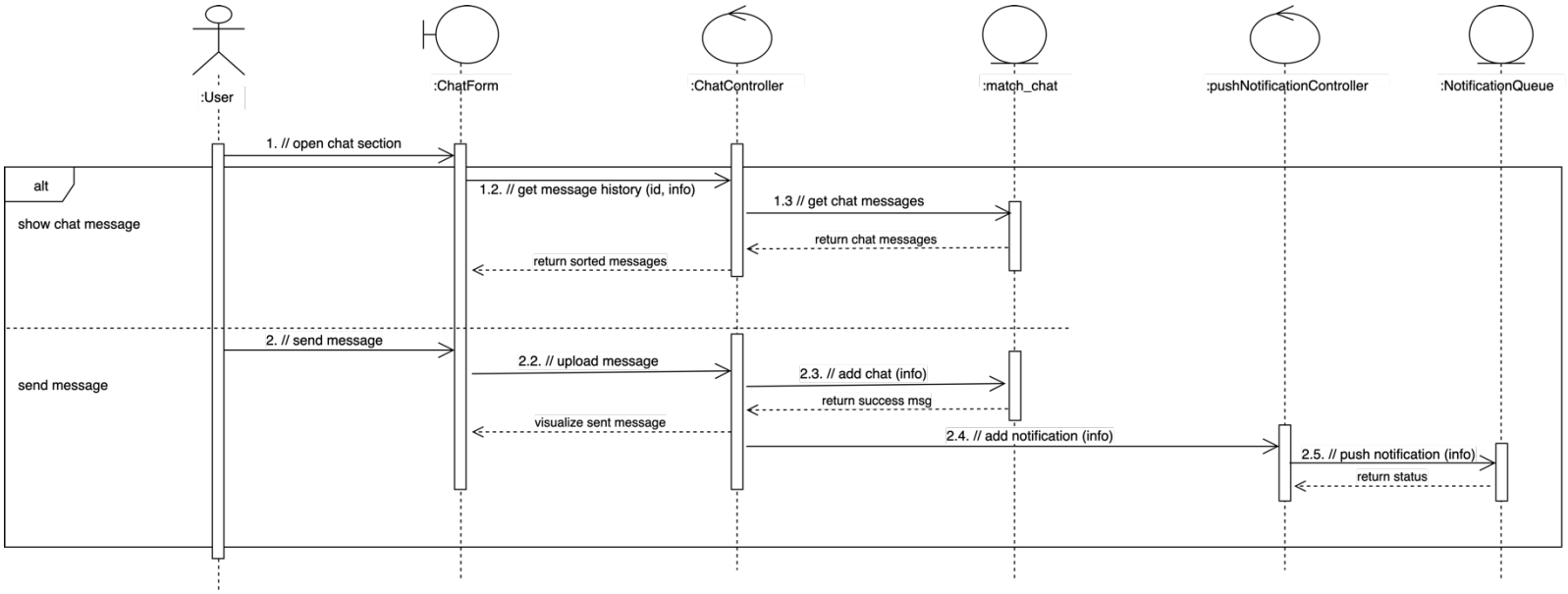
2.2.1.7 Sequence diagram for the Update Settings Use Case



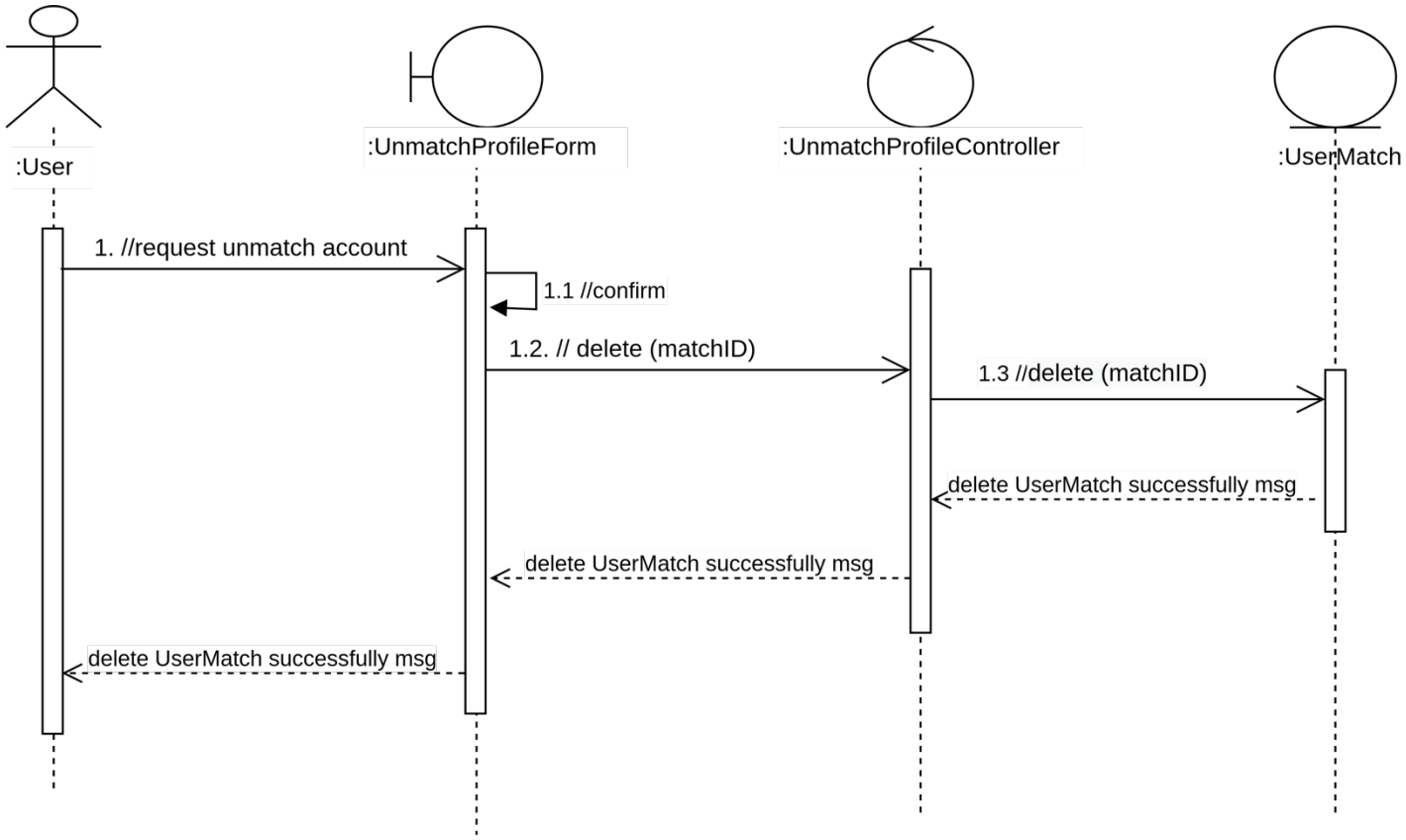
2.2.1.8 Sequence diagram for the Like/Dislike & Match Profile Use Case



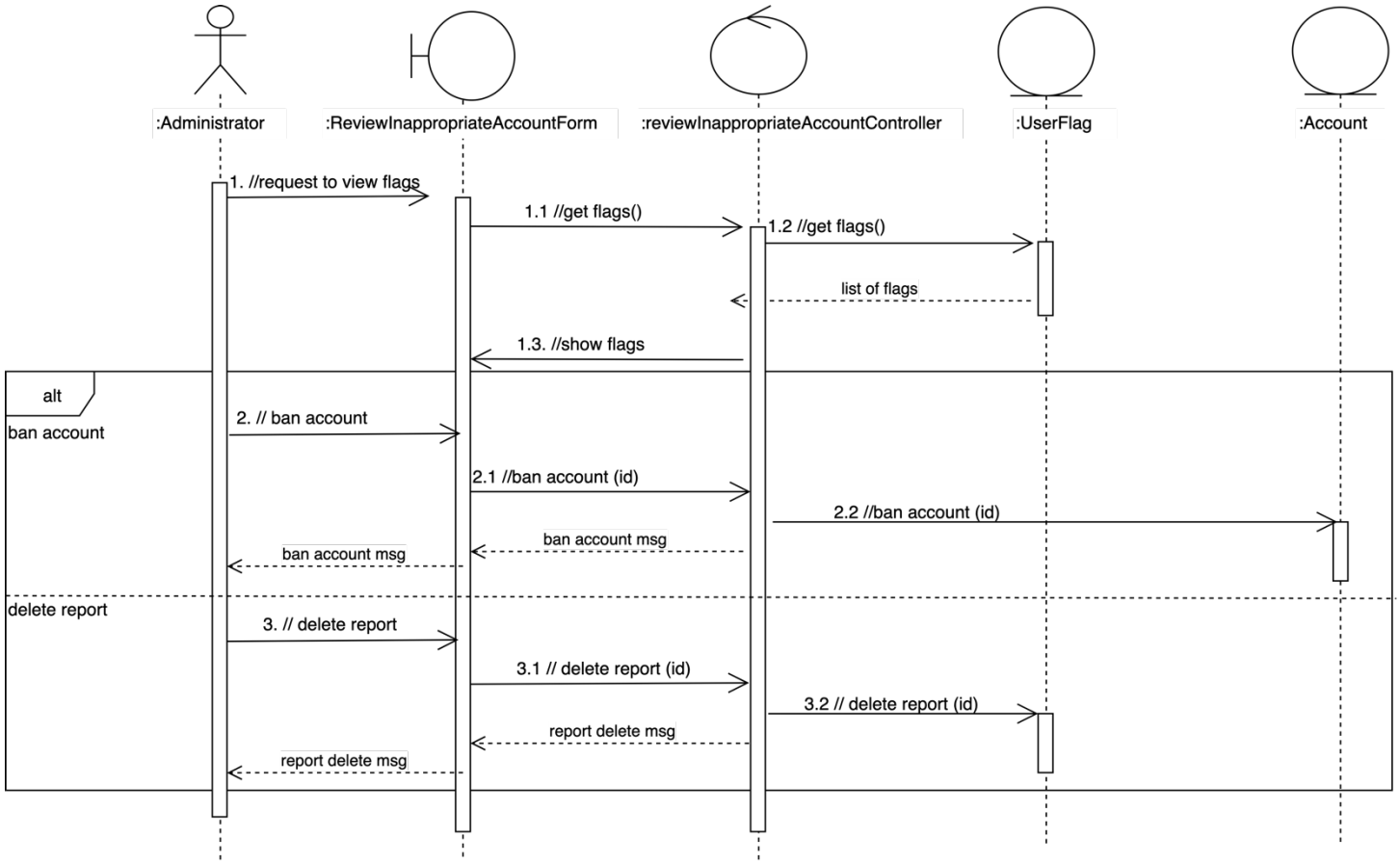
2.2.1.9 Sequence diagram for Chat Use Case



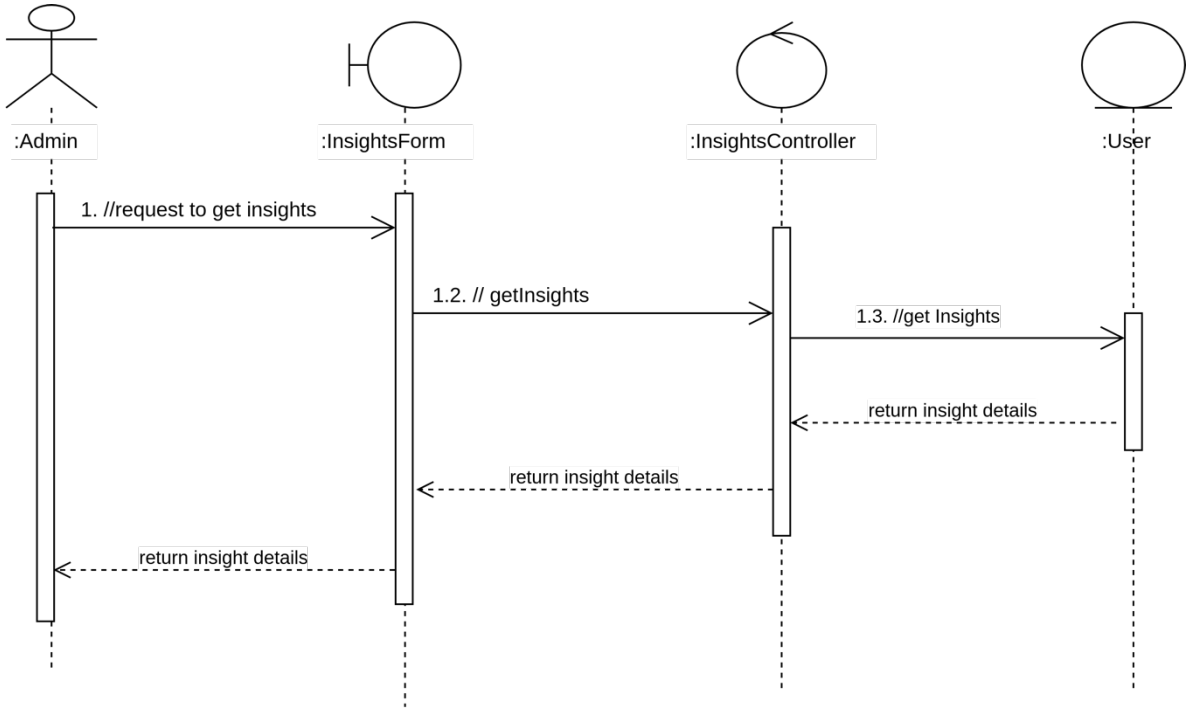
2.2.1.10 Sequence diagram for the Unmatched Use Case



2.2.1.11 Sequence diagram for Review Inappropriate Account Use Case

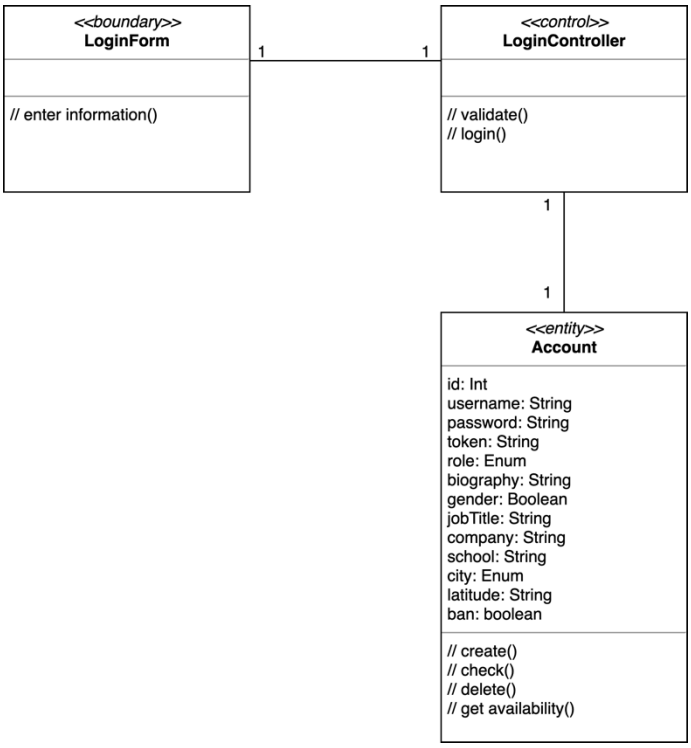


2.2.1.12 Sequence diagram for the Get Insights Use Case

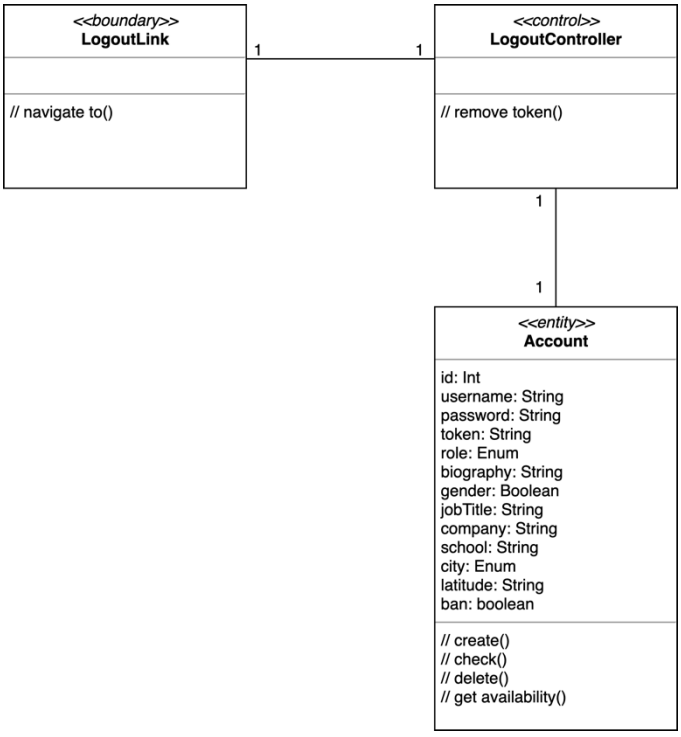


2.2.2 Use-case realizations: views of participating classes

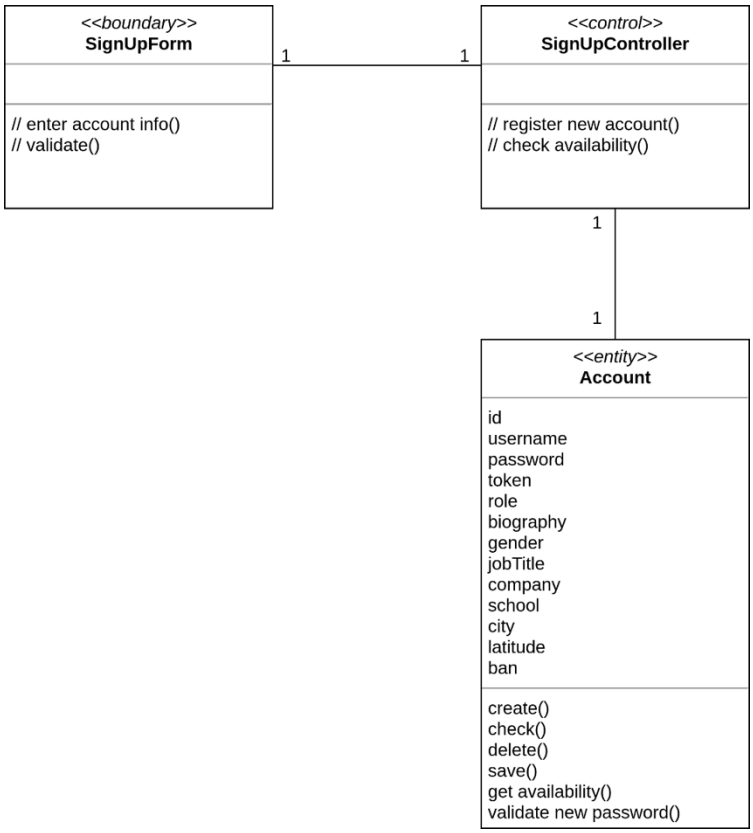
2.2.2.1 VOPC for the Sign In Use Case



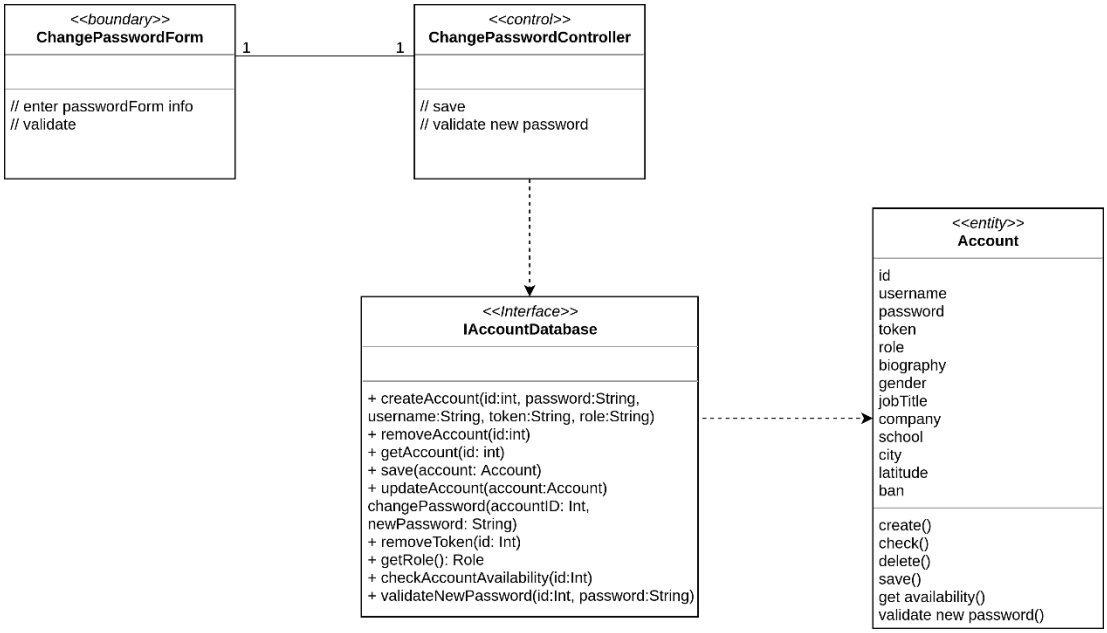
2.2.2.2 VOPC for the Sign Out Use Case



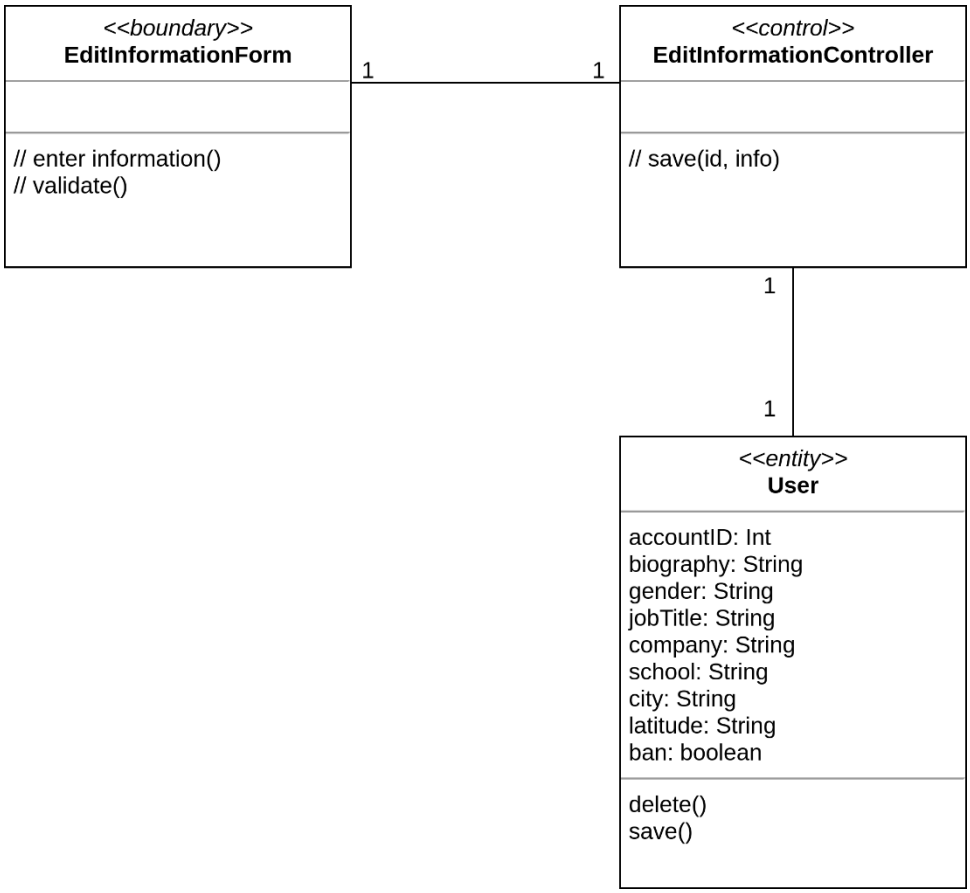
2.2.2.3 VOPC for the Sign Up Use Case



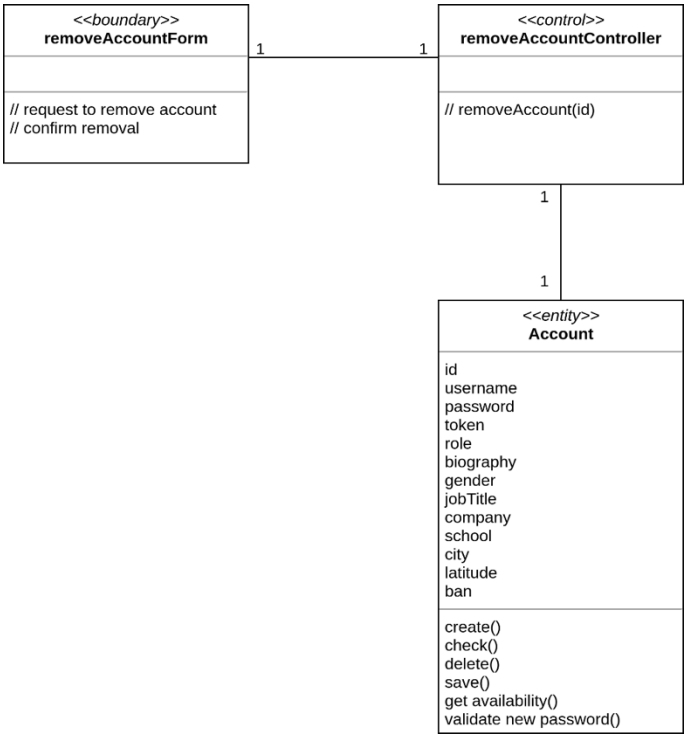
2.2.2.4 VOPC for the Change Password Use Case



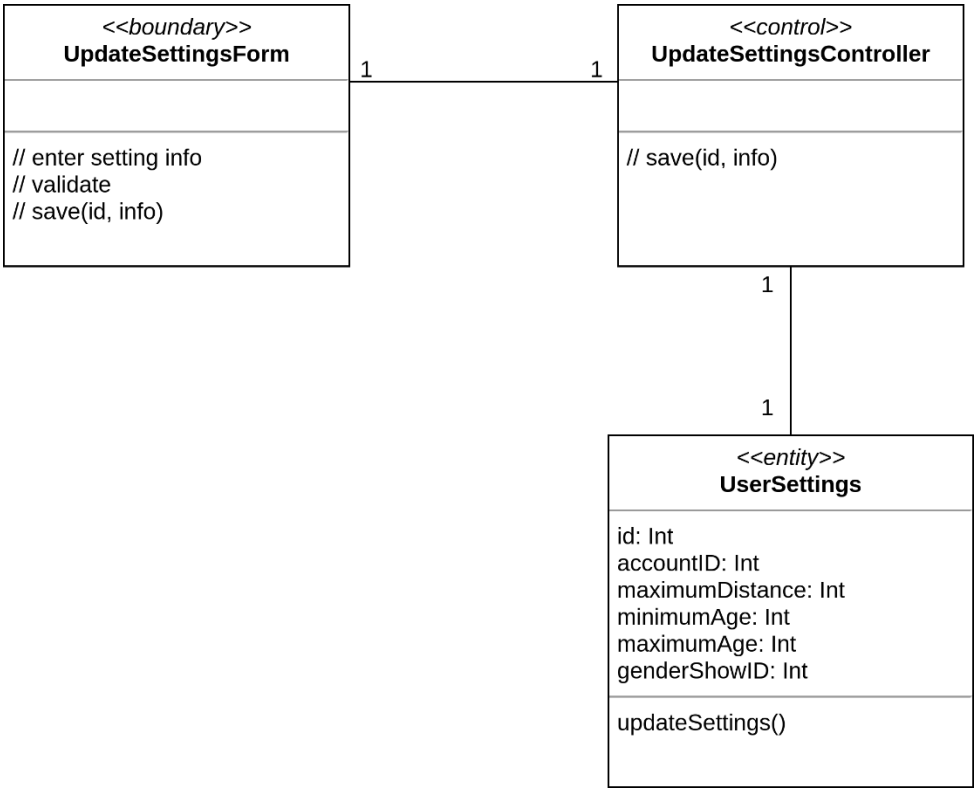
2.2.2.5 VOPC for the Remove Account Use Case



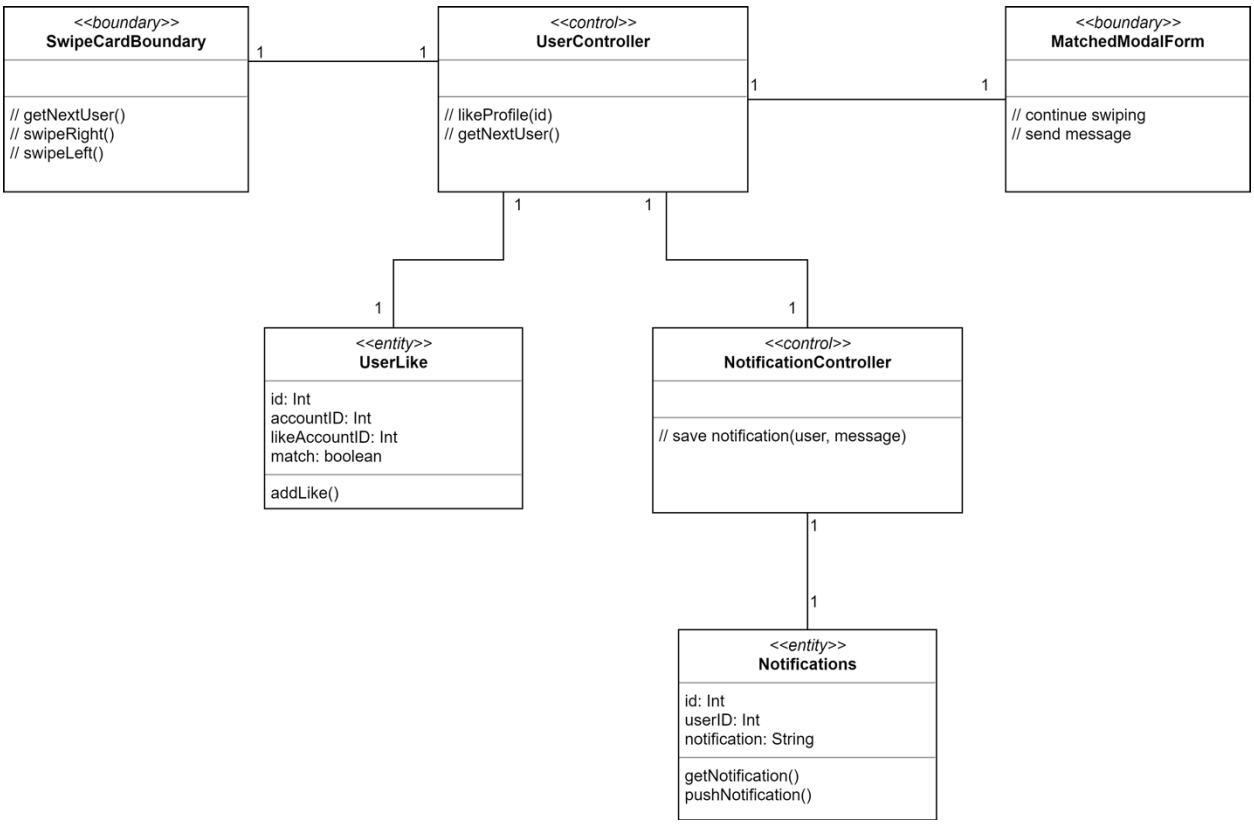
2.2.2.6 VOPC for the Edit Information Use Case



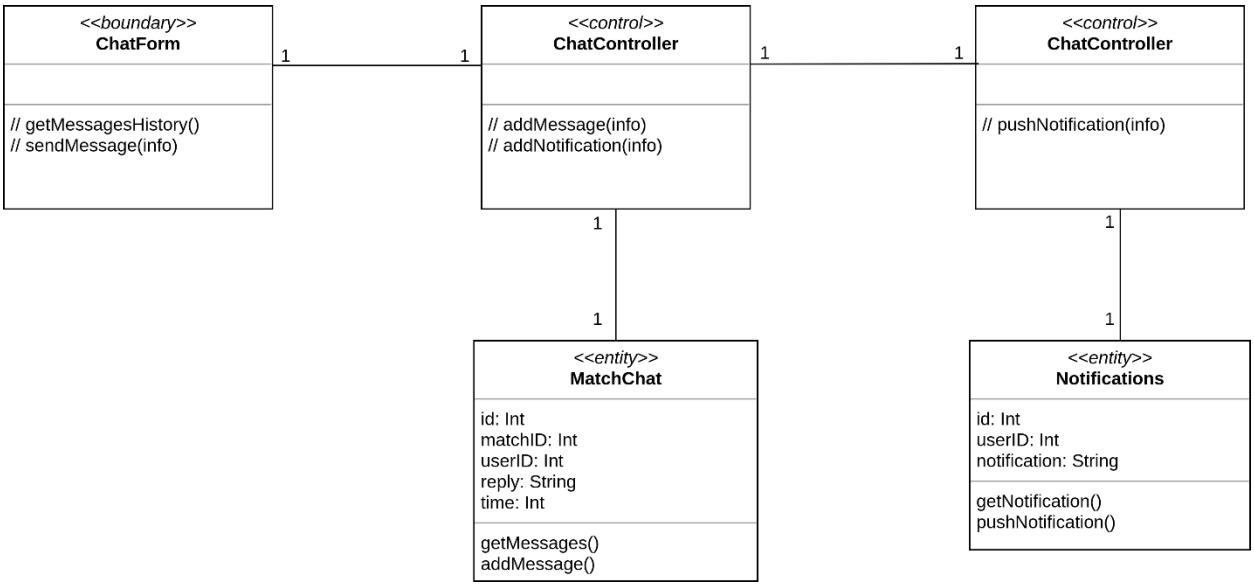
2.2.2.7 VOPC for the Update Settings Use Case



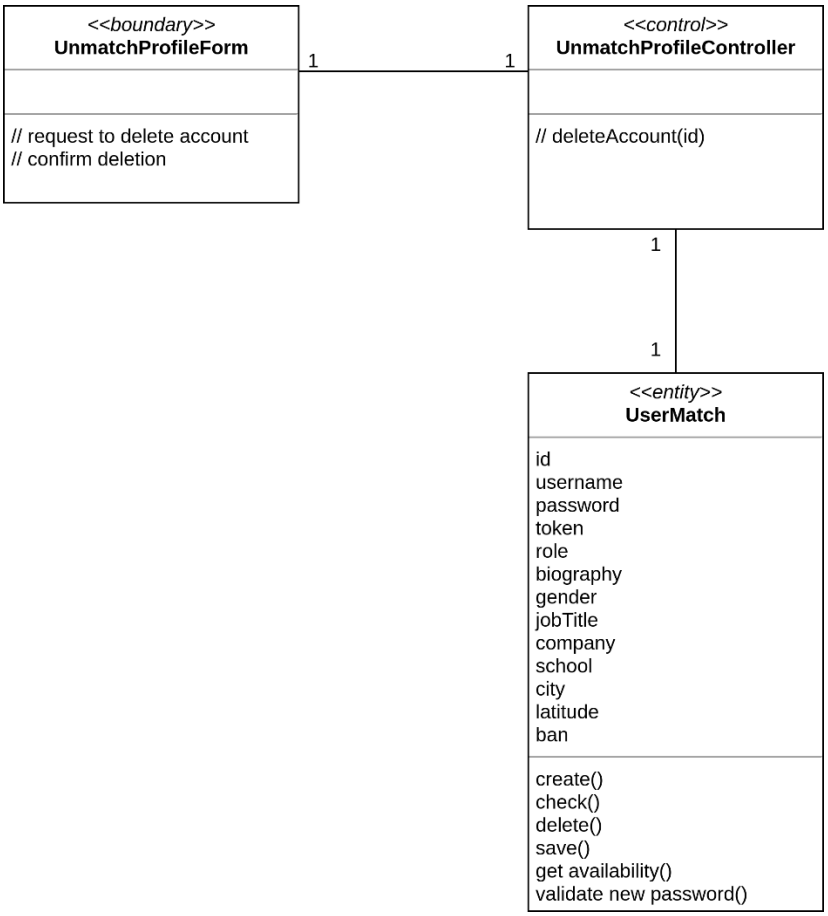
2.2.2.8 VOPC for the Like/Dislike & Match Profile Use Case



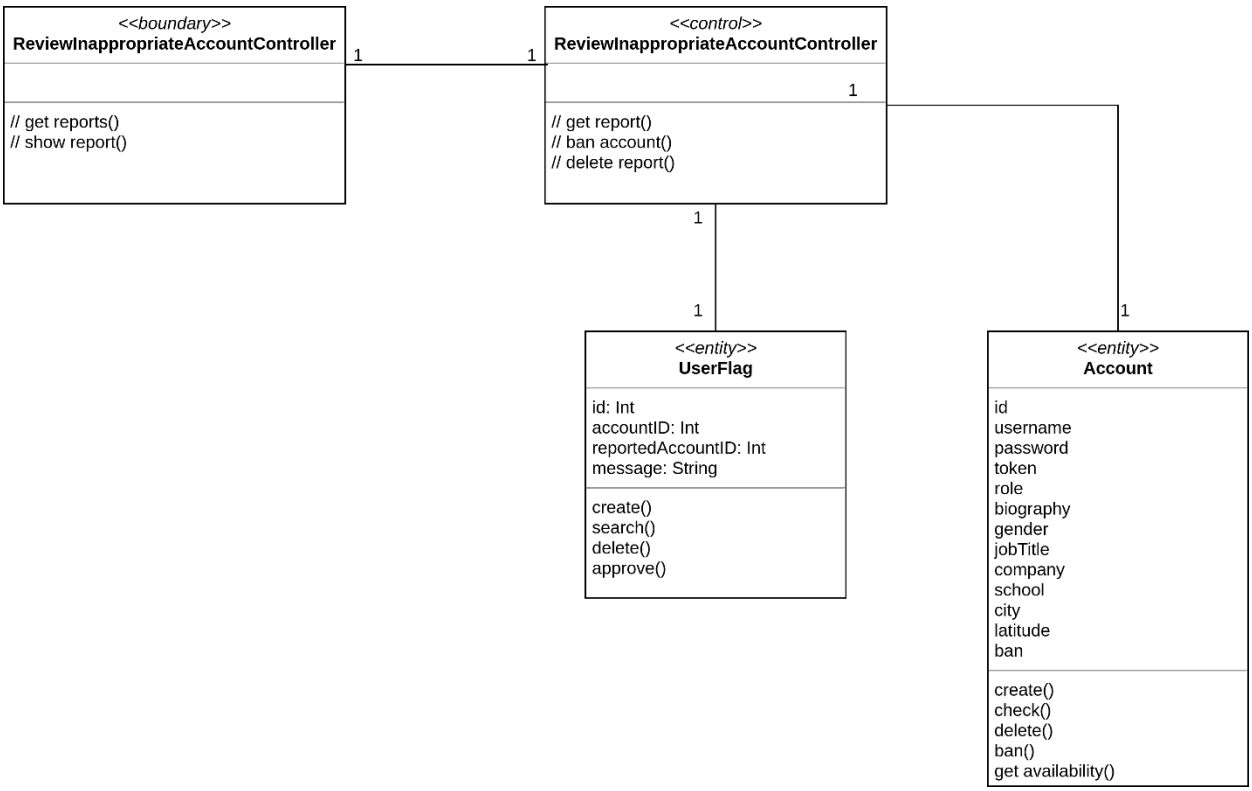
2.2.2.9 VOPC for Chat Use Case



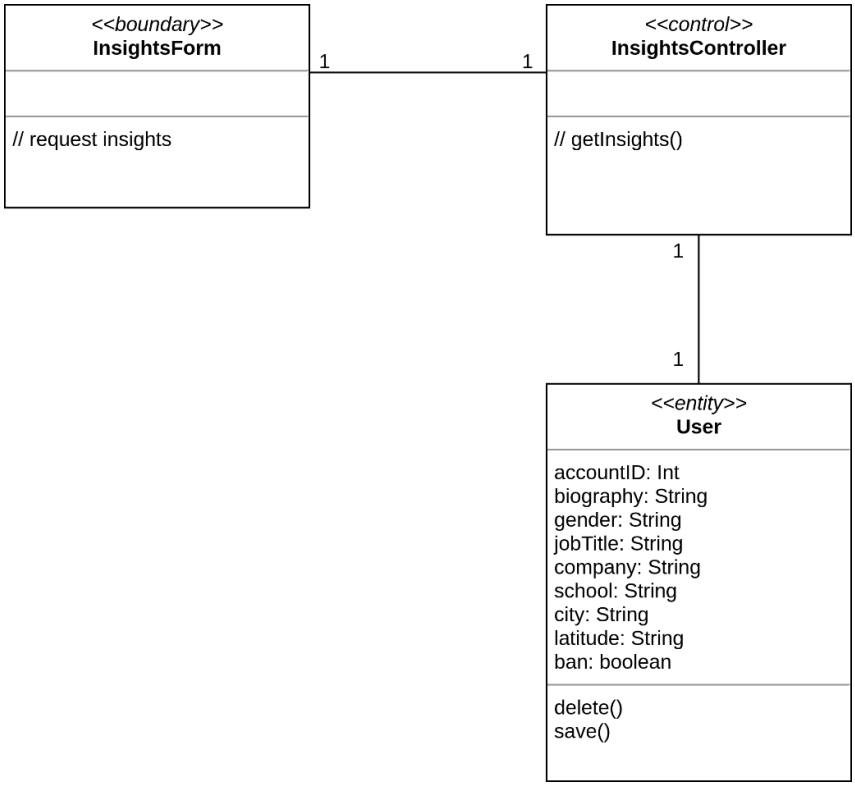
2.2.2.10 VOPC for the Unmatched Use Case



2.2.2.11 VOPC for the Review Inappropriate Account Use Case



2.2.2.12 VOPC for the Get Insights Use Case



2.2.3 Describe Analysis Mechanism

Analysis Class	Analysis Mechanism
Account	Persistency, Security
Flag	
Notification	
Photo	
Message	
Like	
Match	
UpdateSettingsController	Distribution
EditInformationController	
ChangePasswordController	
RemoveAccountController	
UserController	
ChatController	
UnmatchProfileController	
ReviewInappropriateAccountController	
InsightsController	
UserMatchController	
SignUpController	
SignInController	

2.2.3.1 Analysis Mechanism Characteristics

2.2.3.1.1 Security

- Data granularity: attribute level
- User granularity: three roles – unregistered users, registered users and administrators
- Security rules:
    - Only registered users/administrators may log into the system.
    - Only logged in users may view and edit their own account profile.
    - Only logged in users can like/dislike, flag, message, match other profile.
    - Only logged in users can receive their own notifications.
    - Only Administrator can review flag account, view insights

2.2.3.1.2 Persistency

Class	**Account**	**Flag**	**Notification**	**Photo**	**Message**	**Like**	**Match**
Granularity	100 KB per product	1Kb per product	1KB per product	1MB per product	1-4KB per product	1KB per product	1KB per product
Volume	Up to 1,000,000	Up to 1,000,000	Up to 1,000,000,000	Up to 10,000,000	Up to 1,000,000,000	Up to 9,000,000,000	Up to 3,000,000,000

Access frequency	Create: 10000 per day Update: 5000 per day Delete: 100 per day	Create: 100000000 per day Update: 500000000 per day Delete: 100000 per day	Create: 1000000 per day Update: 5000000 per day Delete: 100000 per day	Create: 10000000 per day Update: 5000000 per day Delete: 10000000 per day	Create: 1000000 per day Update: 50000000 per day Delete: 1000000 per day	Create: 10000000 per day Update: 5000000 per day Delete: 1000000 per day	Create: 100000000 per day Update: 5000000 per day Delete: 1000000 per day

3. Use-case Design

3.1. Architectural Refinement

3.1.1 Identify Design Elements

3.1.1.1 Identify Classes

Analysis Class	Design element
Account	Account, Database subsystem
Flag	Flag, Database subsystem
Notification	Notification, Database subsystem
Photo	Photo, Database subsystem
Message	Message, Database subsystem
Like	Like, Database subsystem
Match	Match, Database subsystem
UpdateSettingsController	Map directly to design classes
EditInformationController	
ChangePasswordController	
RemoveAccountController	
UserController	
ChatController	
UnmatchProfileController	
ReviewInappropriateAccountController	
InsightsController	
UserMatchController	
SignUpController	
SignInController	
MatchModalForm	
SwipeCardForm	
InsightForm	
RemoveAccountForm	

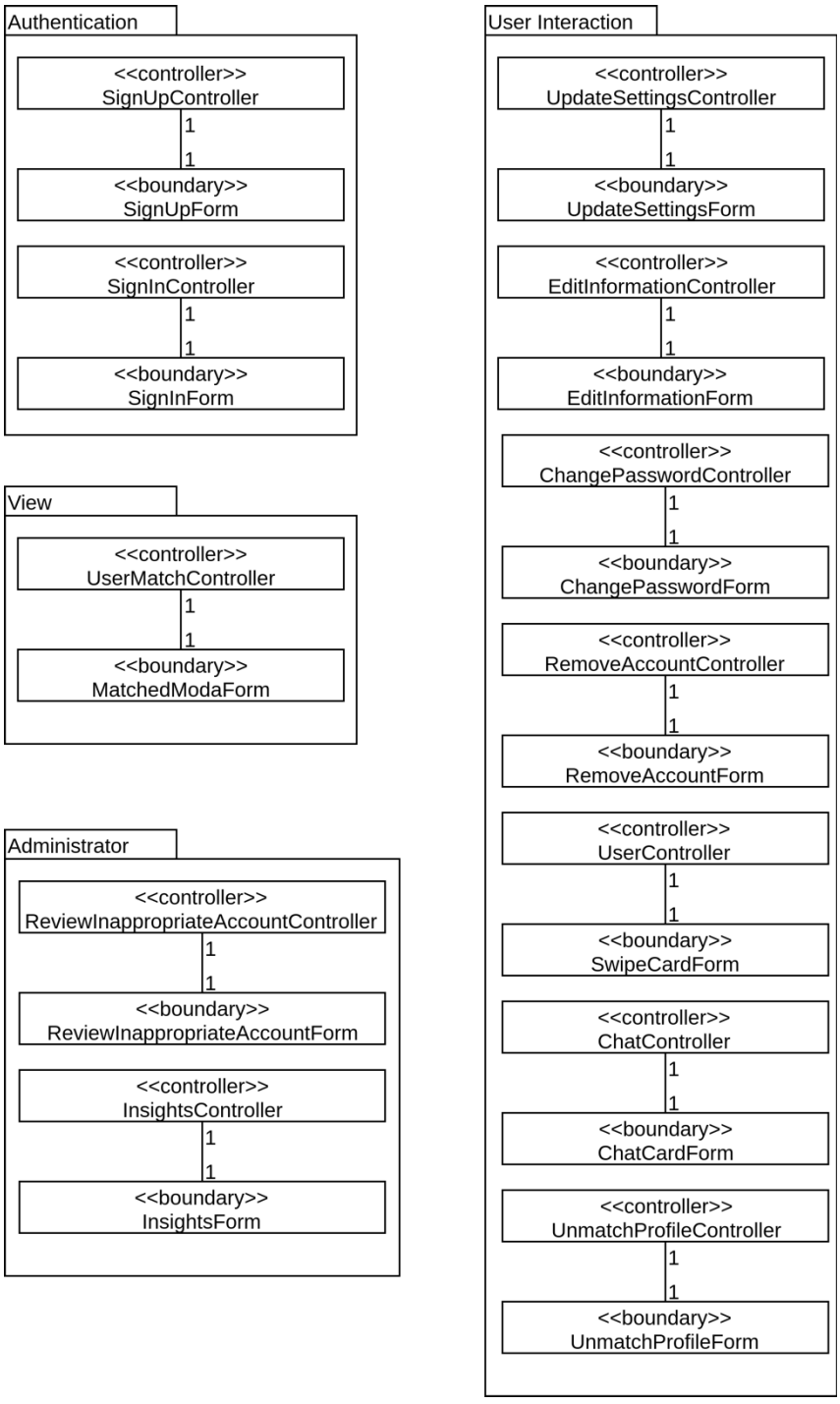
3.1.1.2 Identify subsystems and interfaces

The Database subsystem provides support for relational databases written in the SQL language. The subsystem is designed as follows:

3.1.1.3 Identify Packages

Each layer in the analysis corresponds to a high-level package in the system.

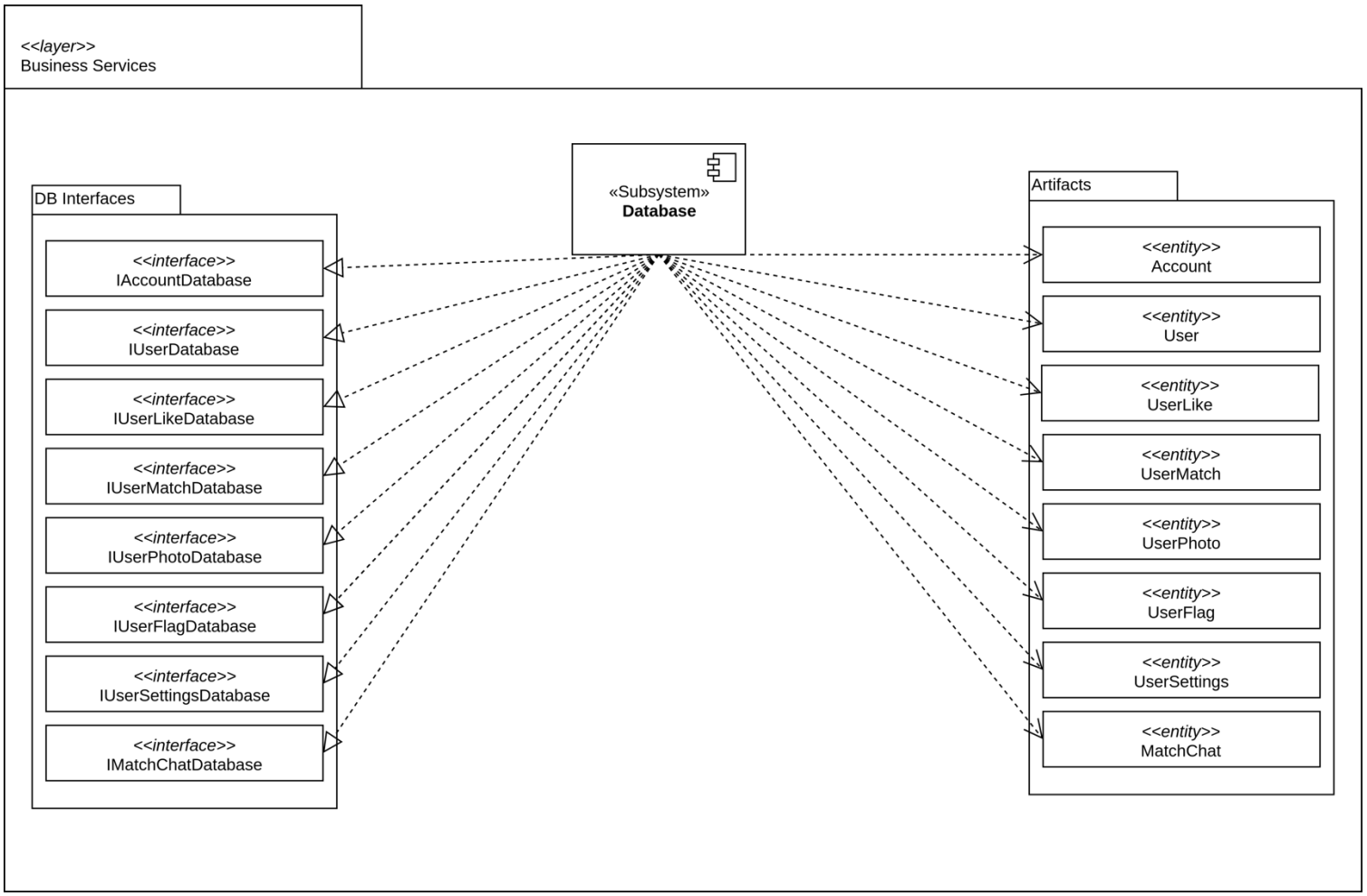
3.1.1.3.1 The Application Package



The *Application* package contains the boundary and control classes, which are present in the client application. It is further divided into four sub-packages, each responsible for a different part of the application:

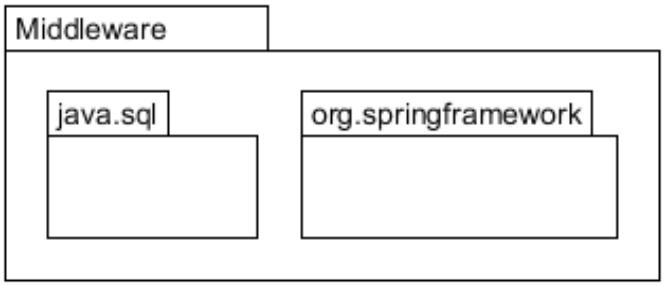
- The Authentication sub-class handles account creation and login.
- The View sub-class is the realization of two use cases related to like/dislike profile, message with matches, flag account. This part of the application can be freely accessed by anyone.
- The User Interaction sub-class contains classes involving actions which require the user to be logged in
- The Administration sub-class contains utilities that help administrators maintain accounts, review toxic accounts, get system insights. Only Administrators have access to these tools.

3.1.1.3.2 The Business Services package



The Business Services package contains the Database subsystem and its interfaces, as well as the entity classes. These elements are common to all use cases.

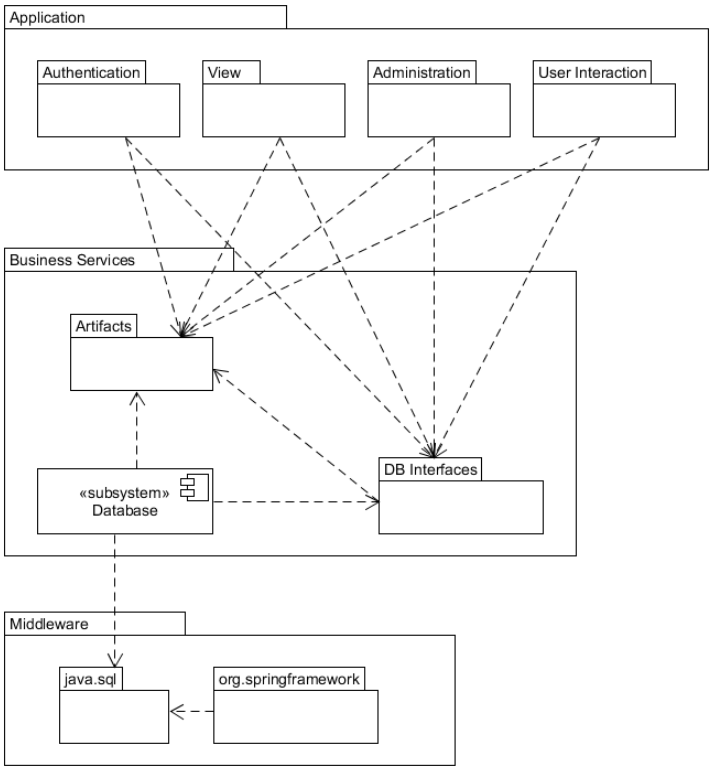
3.1.1.3.3 The Middleware package



The Middleware package includes Java's SQL package, which provides access to databases and the Java Spring framework, which provides network services.

3.1.1.3.4 Packages and their dependencies

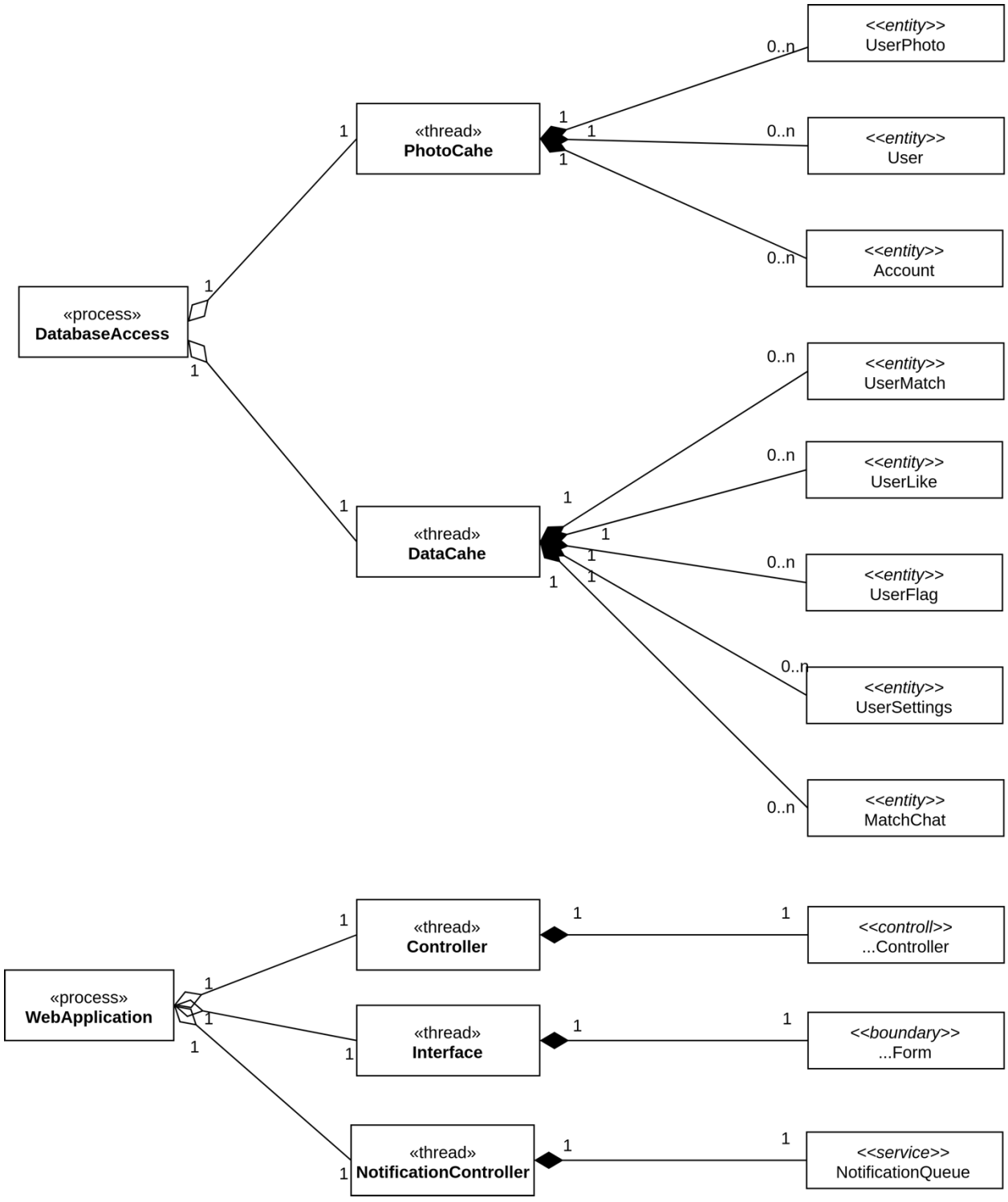
As already stated, the *Application* package depends on the *Business Services* package, which in turn depends on the Middleware package.



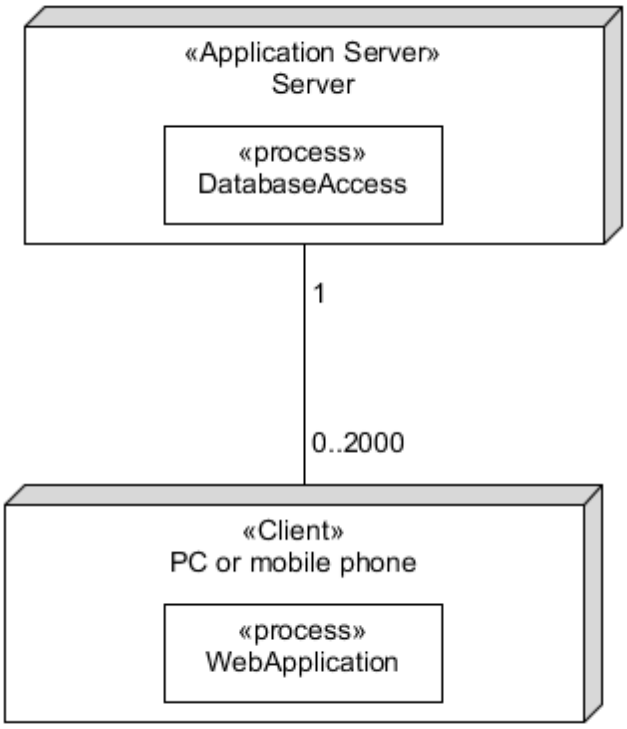
3.1.2 Identify design mechanisms

Analysis Mechanism	Design Mechanism	Implement Mechanism
Persistency	RDBMS	JDBC
Security	Web Tokens	Java Spring Framework
Distribution	REST API	Java Spring Framework

3.2. Describe the run-time architecture



3.3. Describe distribution

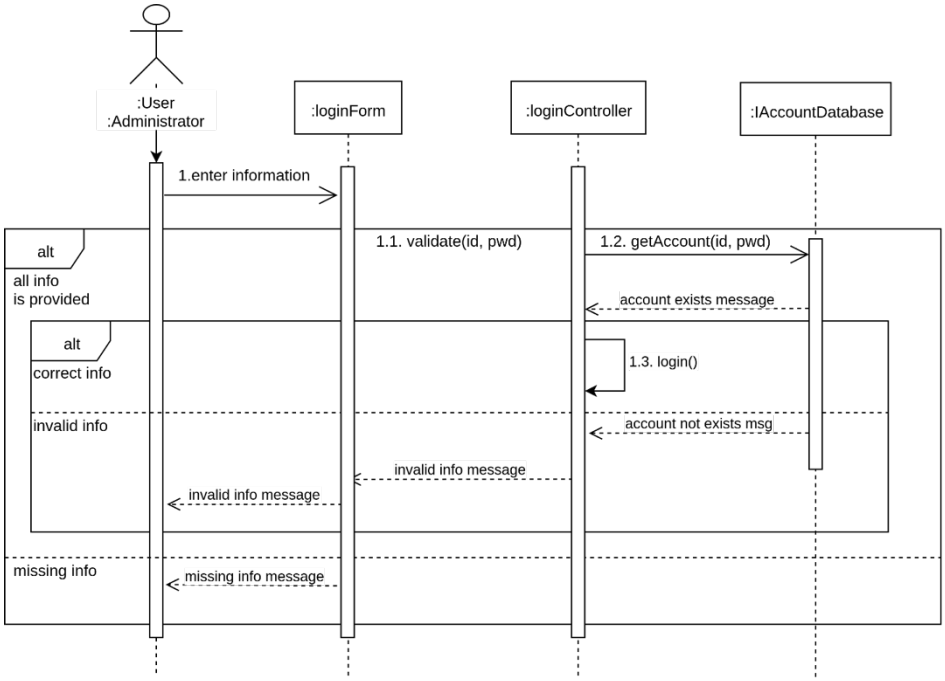


3.4. Use-case design

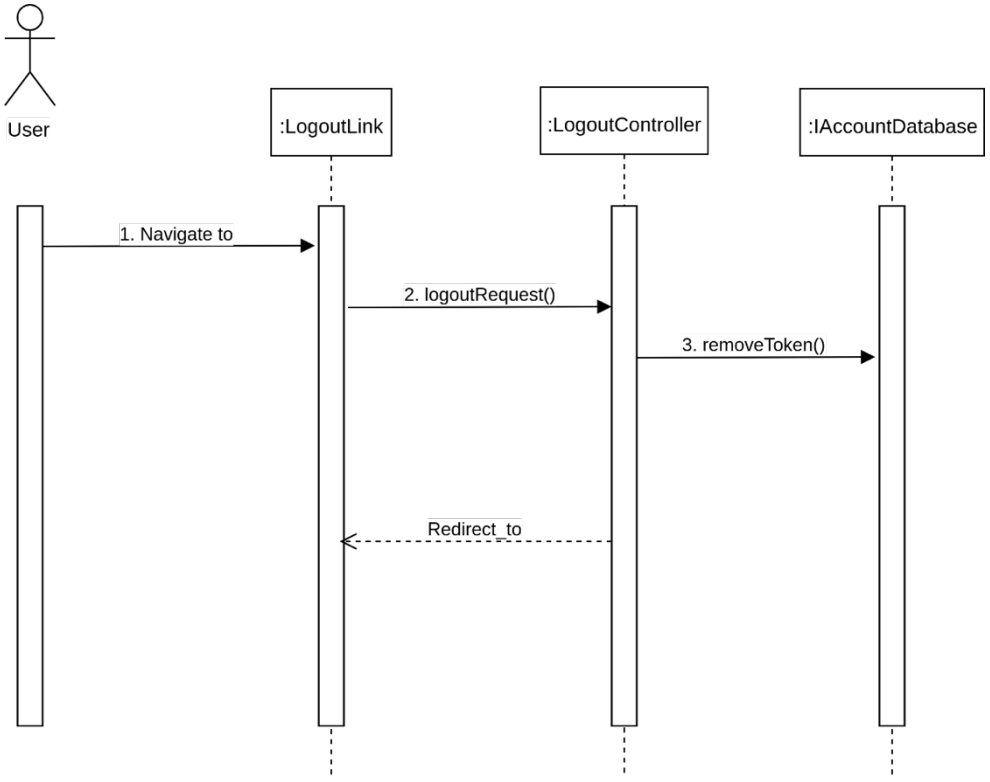
3.4.1 Design sequence diagrams

After incorporating the Database subsystem, the model's sequence diagrams are updated as follows. Some method parameters are elided for conciseness and legibility – they are shown in full in the Class Design section.

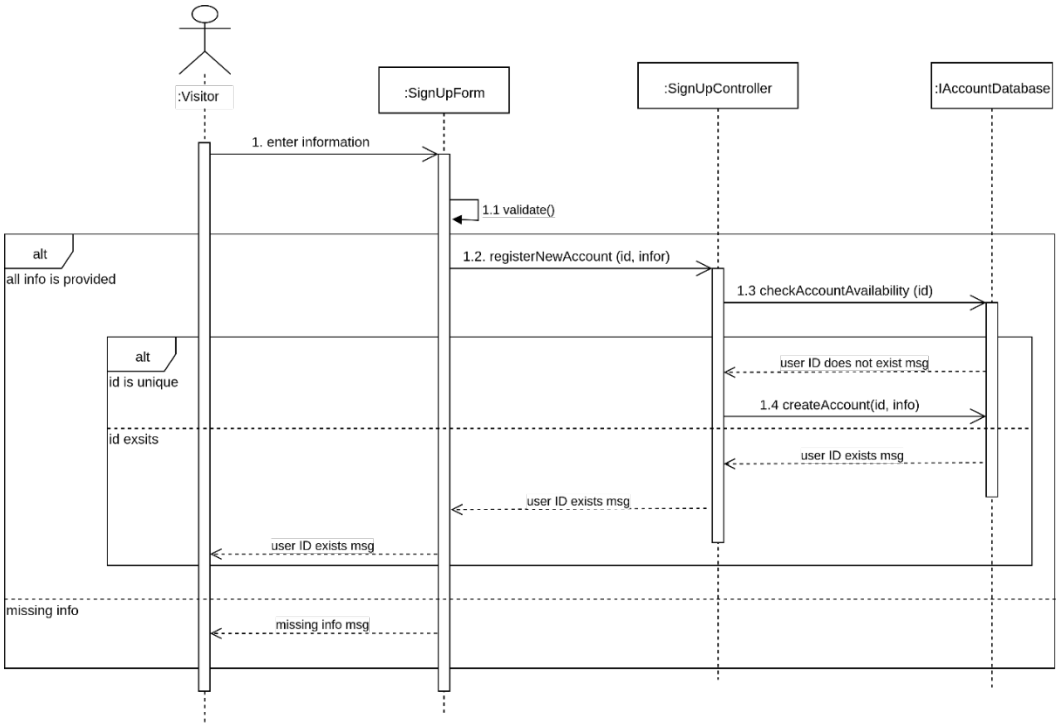
3.4.1.1 Design sequence diagram for the Sign In Use Case



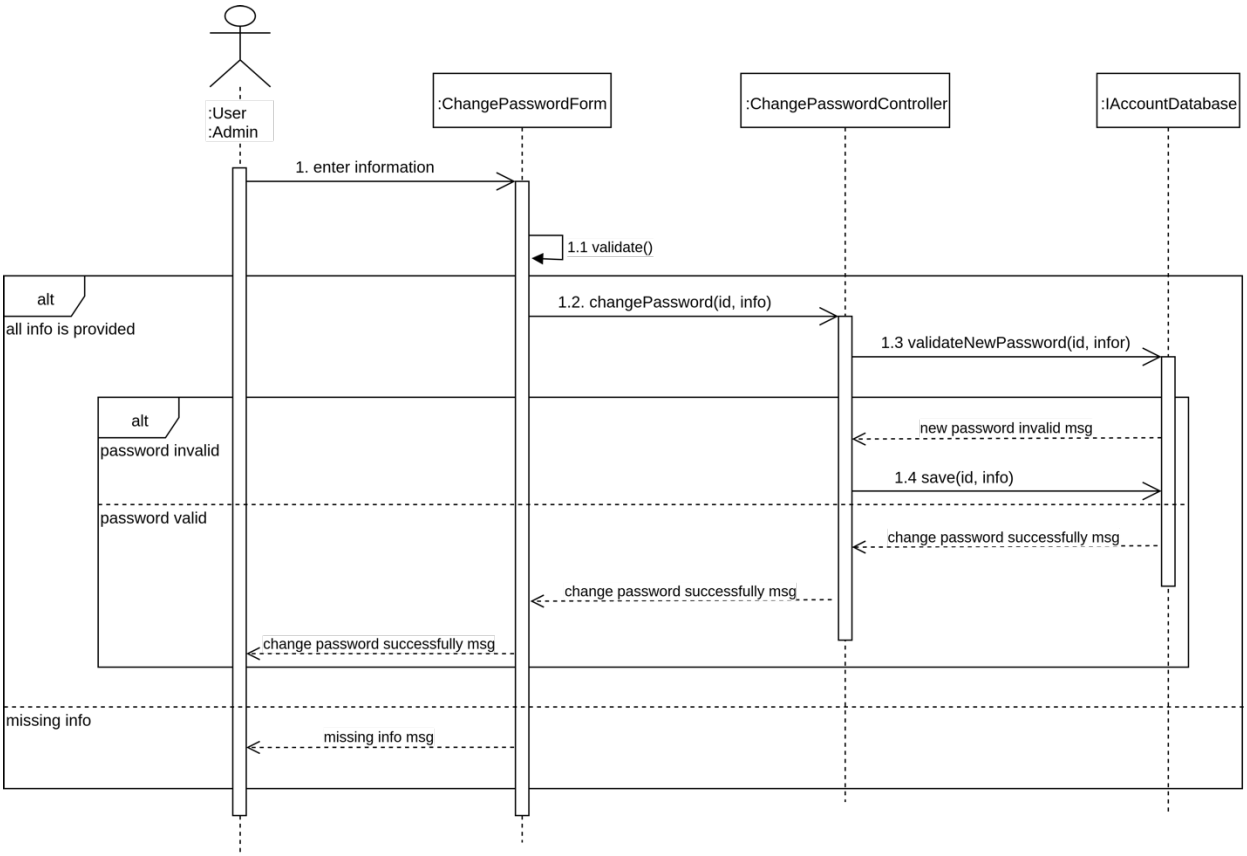
3.4.1.2 Design sequence diagram for the Sign Out Use Case



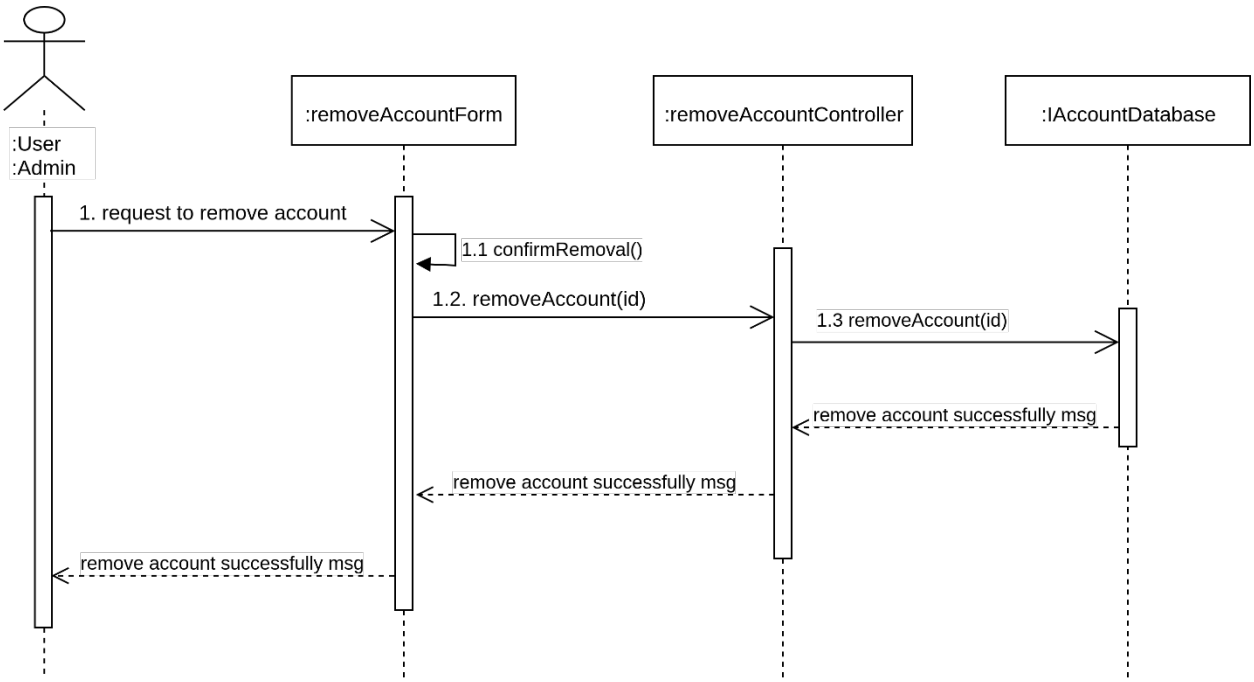
3.4.1.3 Design sequence diagram for the Sign Up Use Case



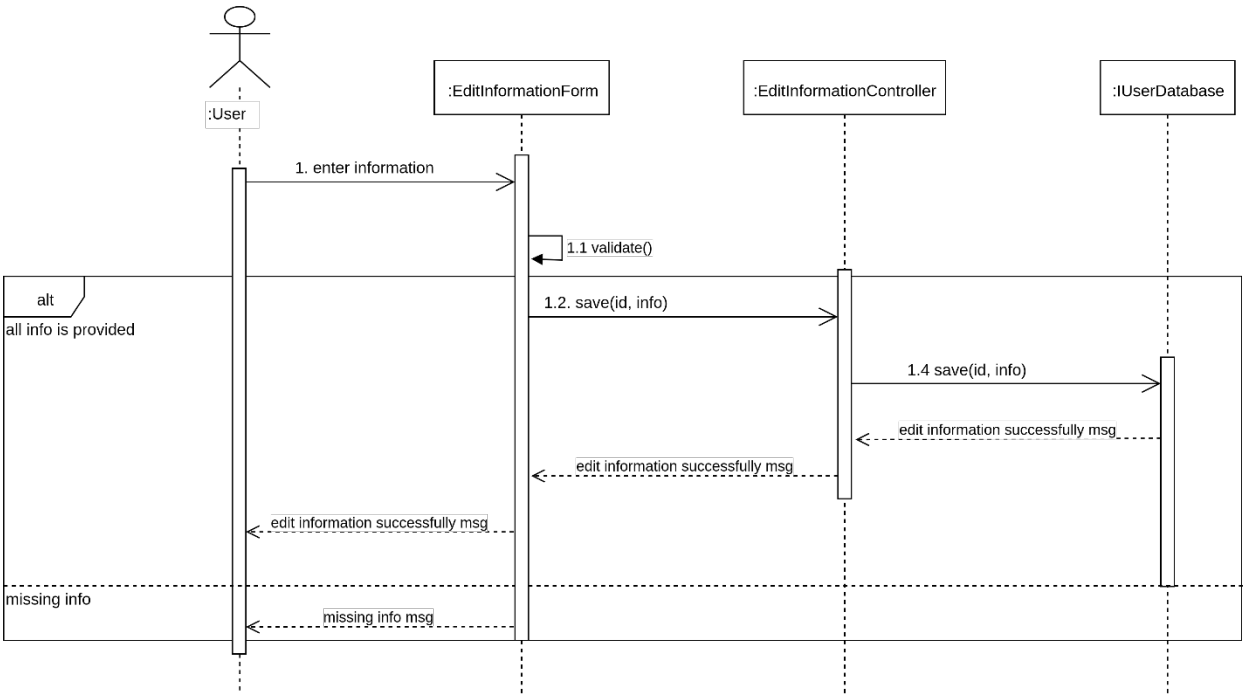
3.4.1.4 Design sequence diagram for the Change Password Use Case



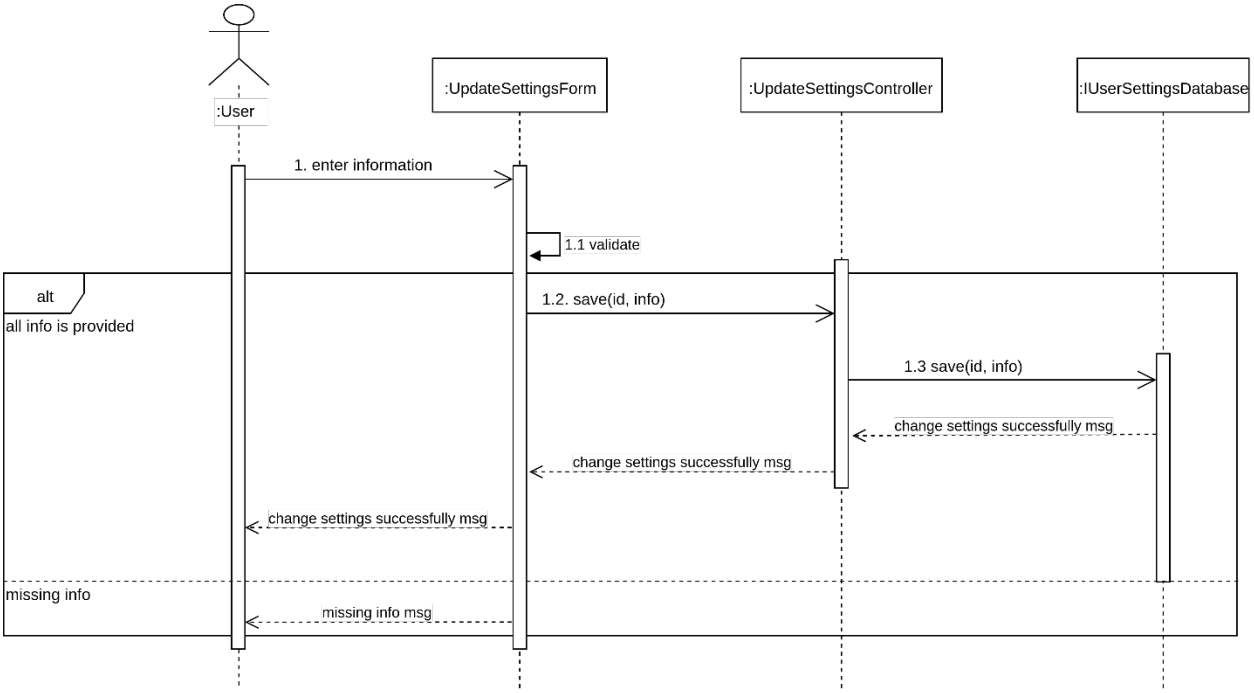
3.4.1.5 Design sequence diagram for the Remove Account Use Case



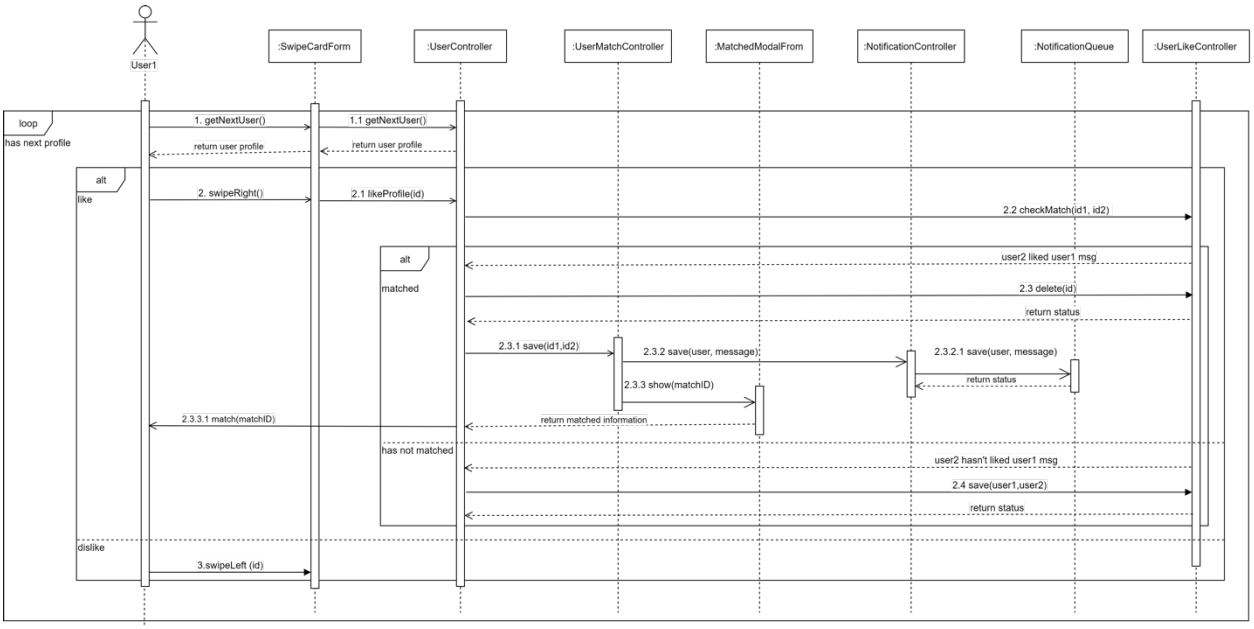
3.4.1.6 Design sequence diagram for the Edit Information Use Case



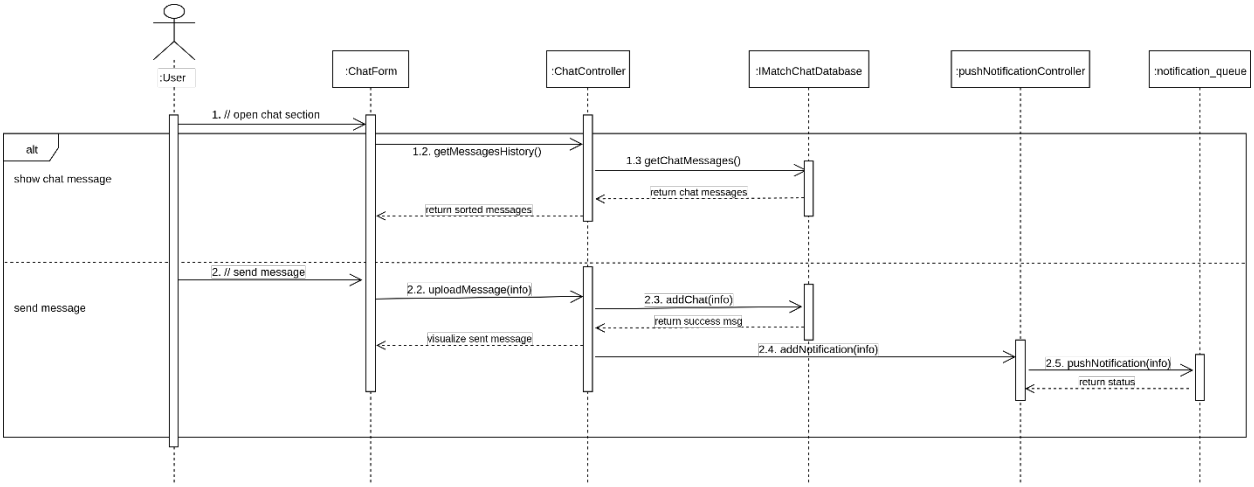
3.4.1.7 Design sequence diagram for the Update Settings Use Case



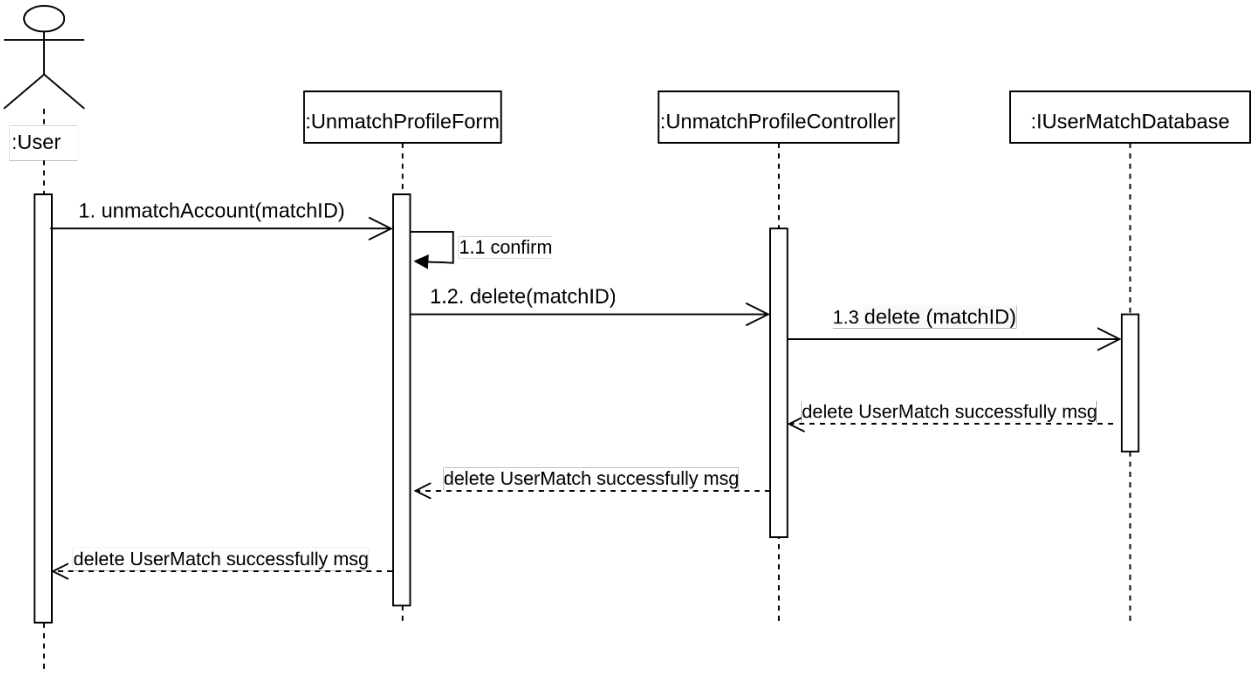
3.4.1.8 Design sequence diagram for the Like/Dislike & Match Profile Use Case



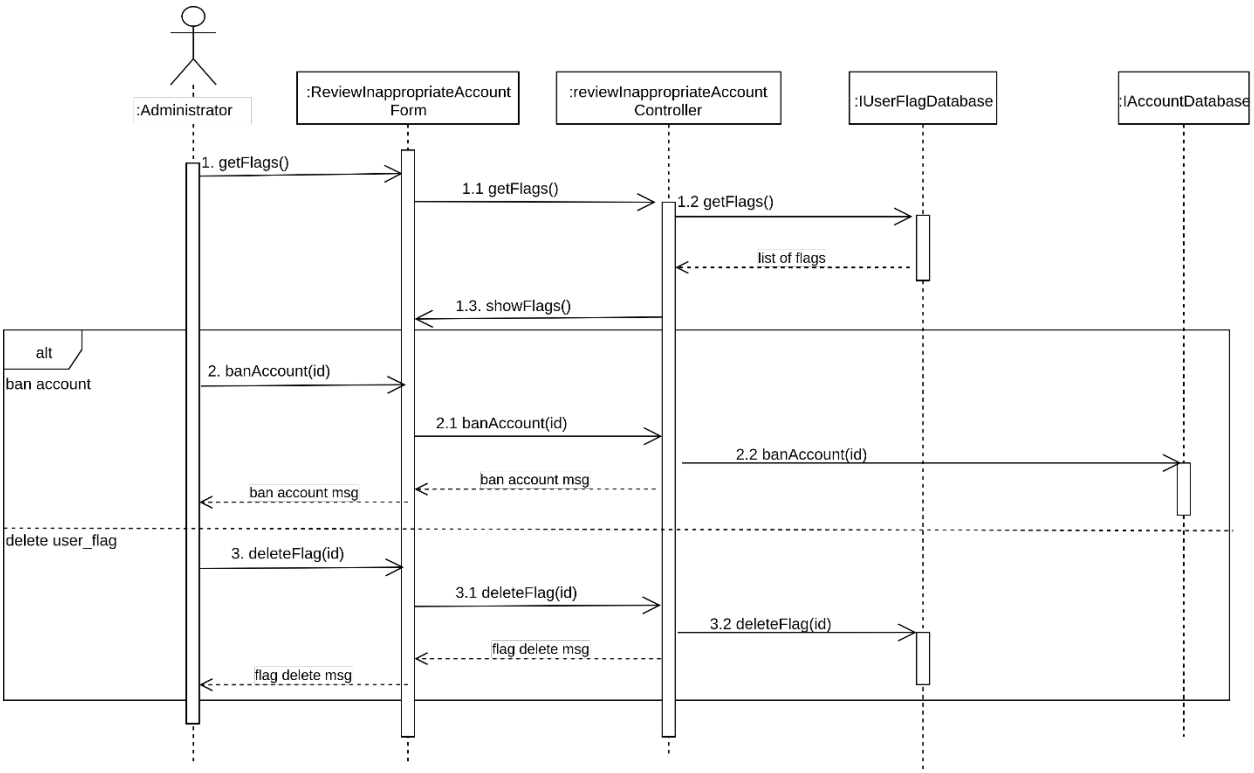
3.4.1.9 Design sequence diagram for Chat Use Case



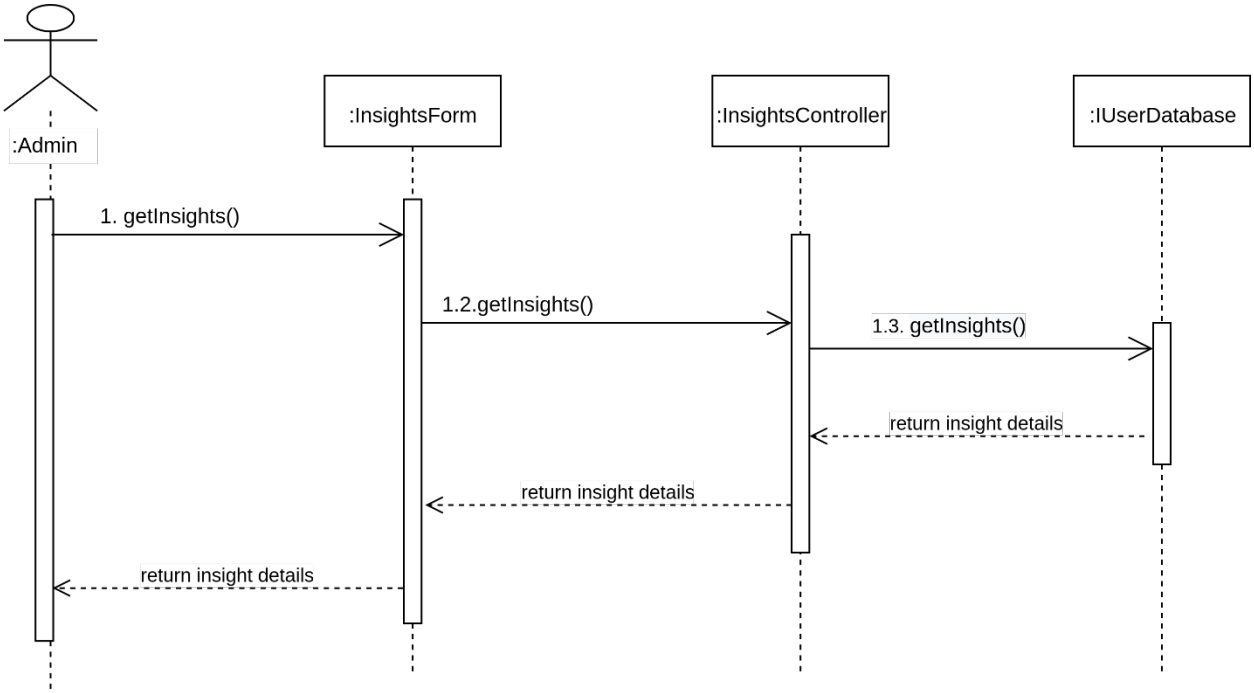
3.4.1.10 Design sequence diagram for the Unmatched Use Case



3.4.1.11 Design sequence diagram for the Review Inappropriate Account Use Case



3.4.1.12 Design sequence diagram for the Get Insights Use Case

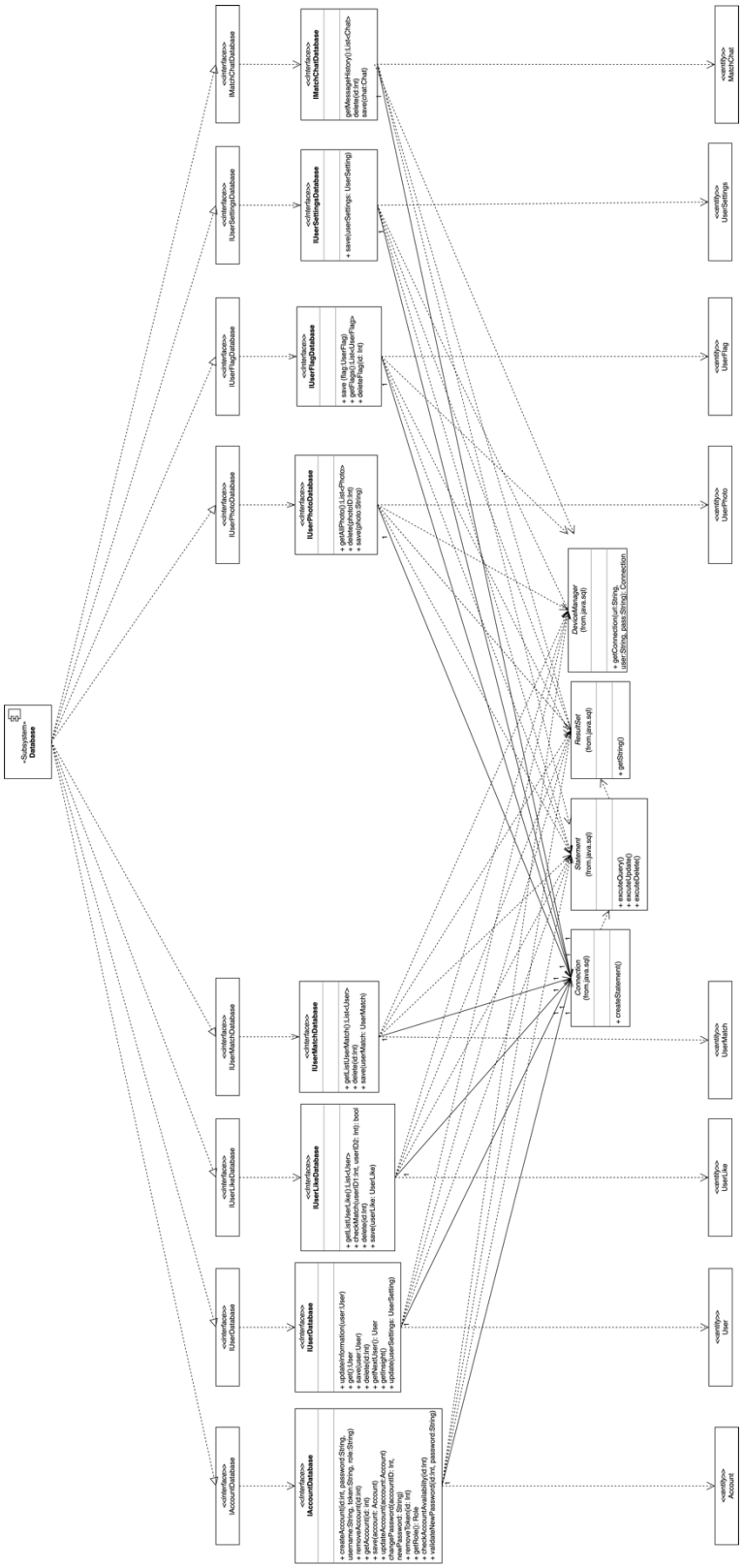


3.4.2 Design views of participating classes

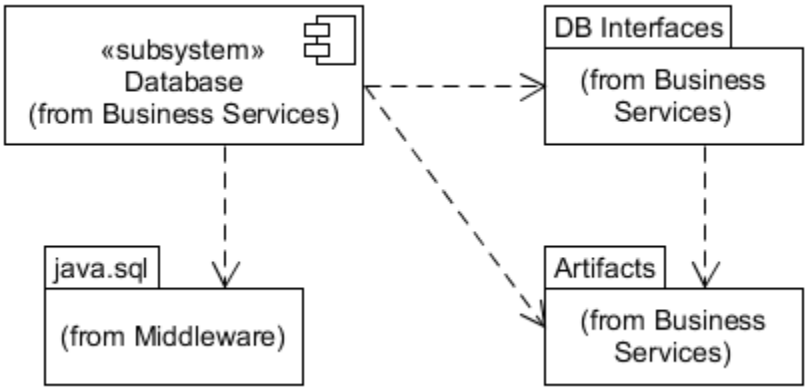
The updated views of participating classes for each use case are described in the Class Design section.

3.5. Sub System Design

3.5.1 Database subsystem elements diagram

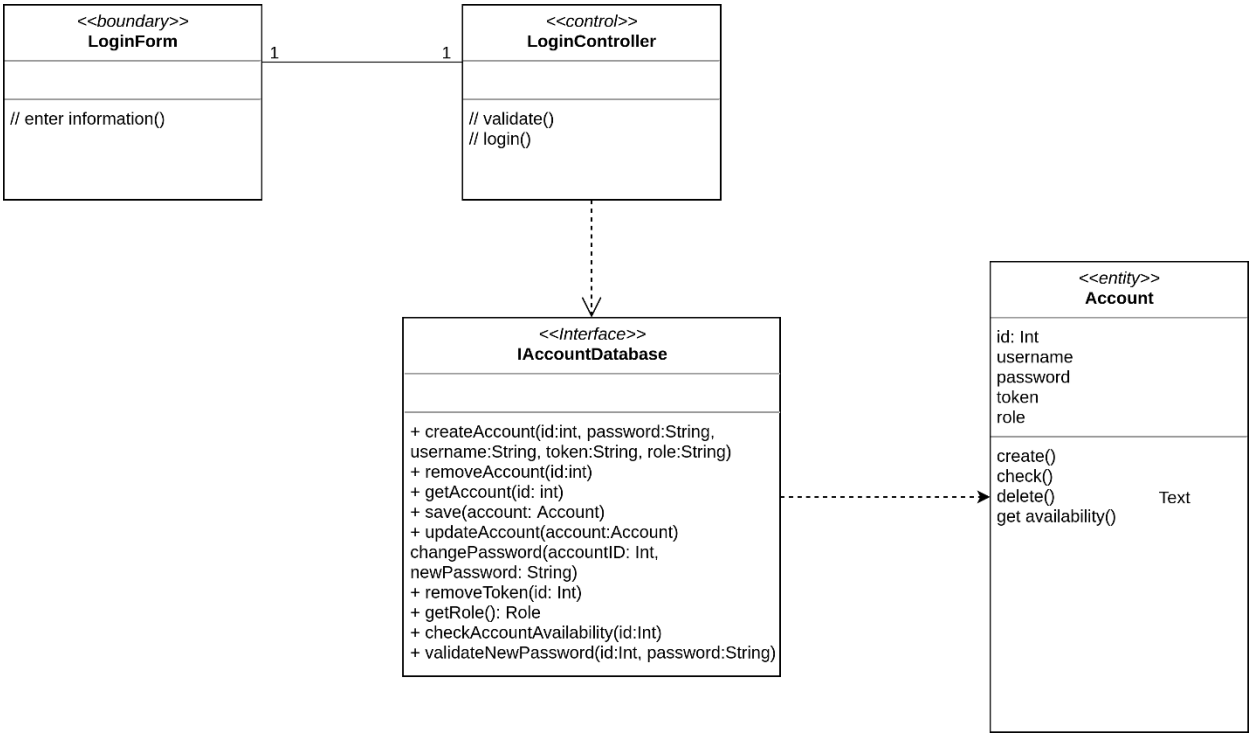


3.5.2 Subsystem dependencies class diagram

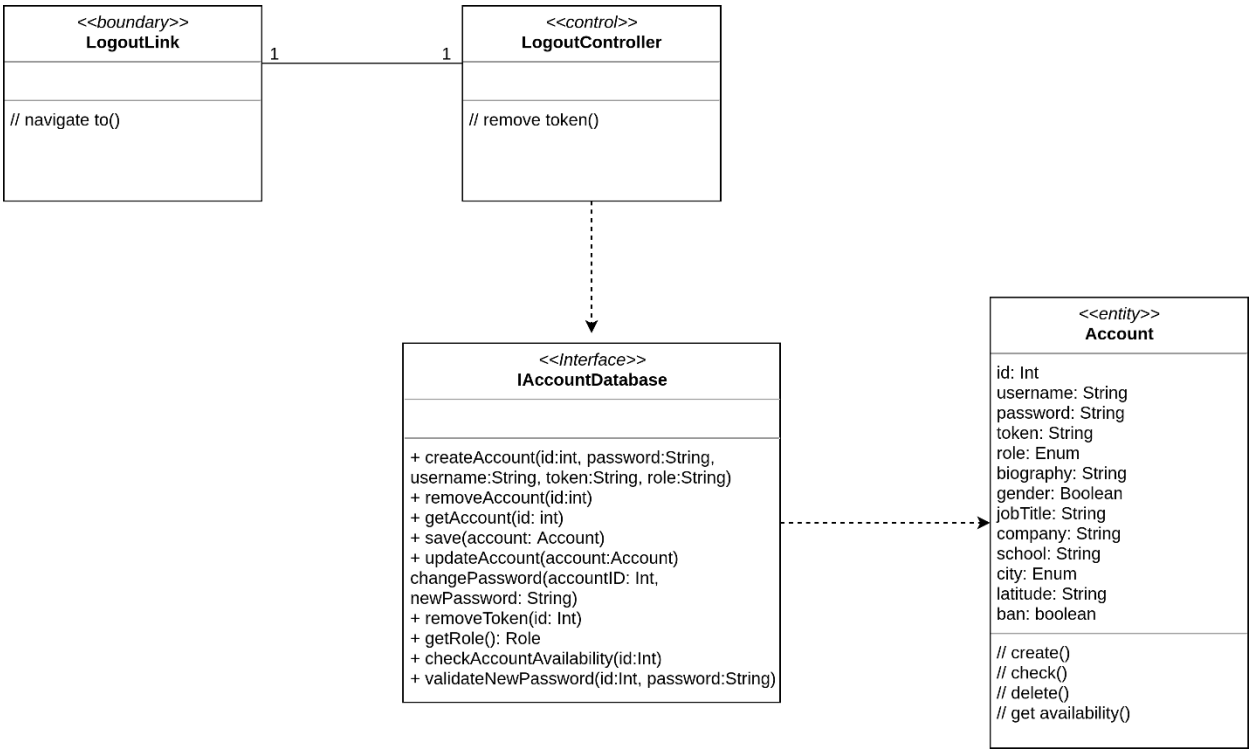


3.6. Class design

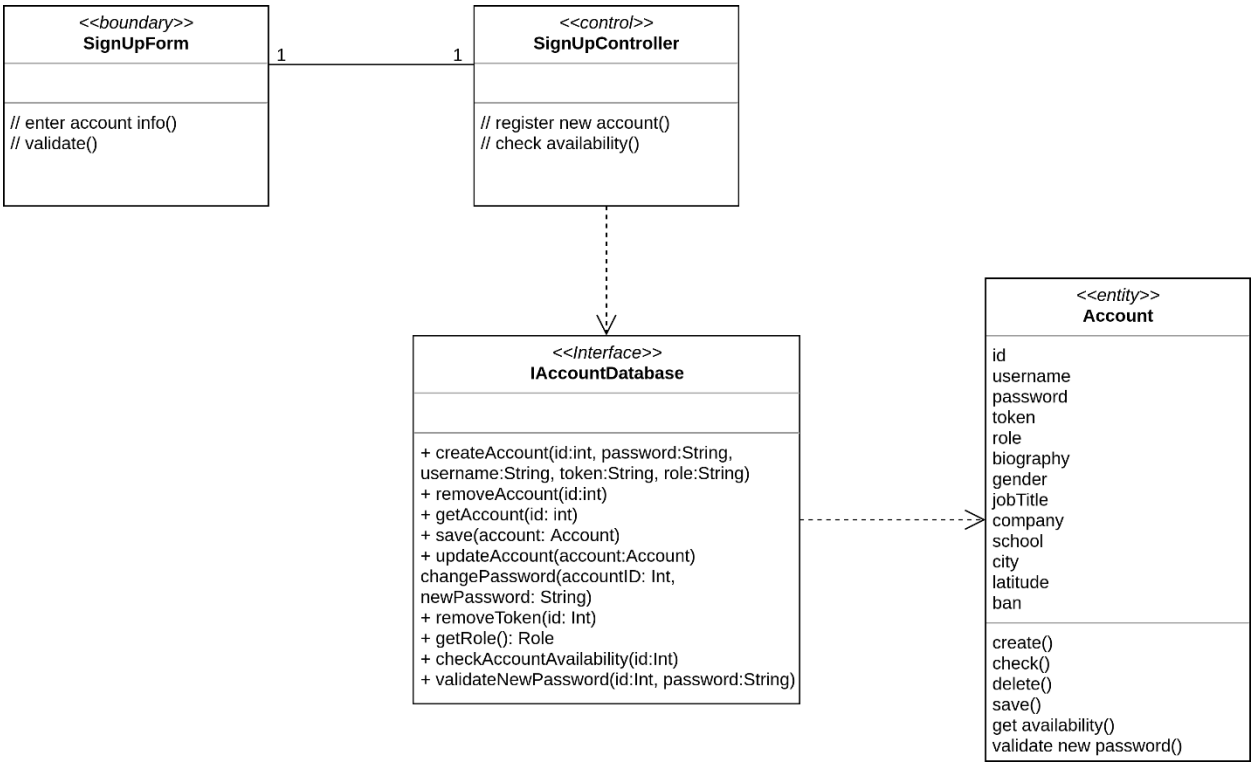
3.6.1 *Design VOPC* diagram for the Sign In Use Case



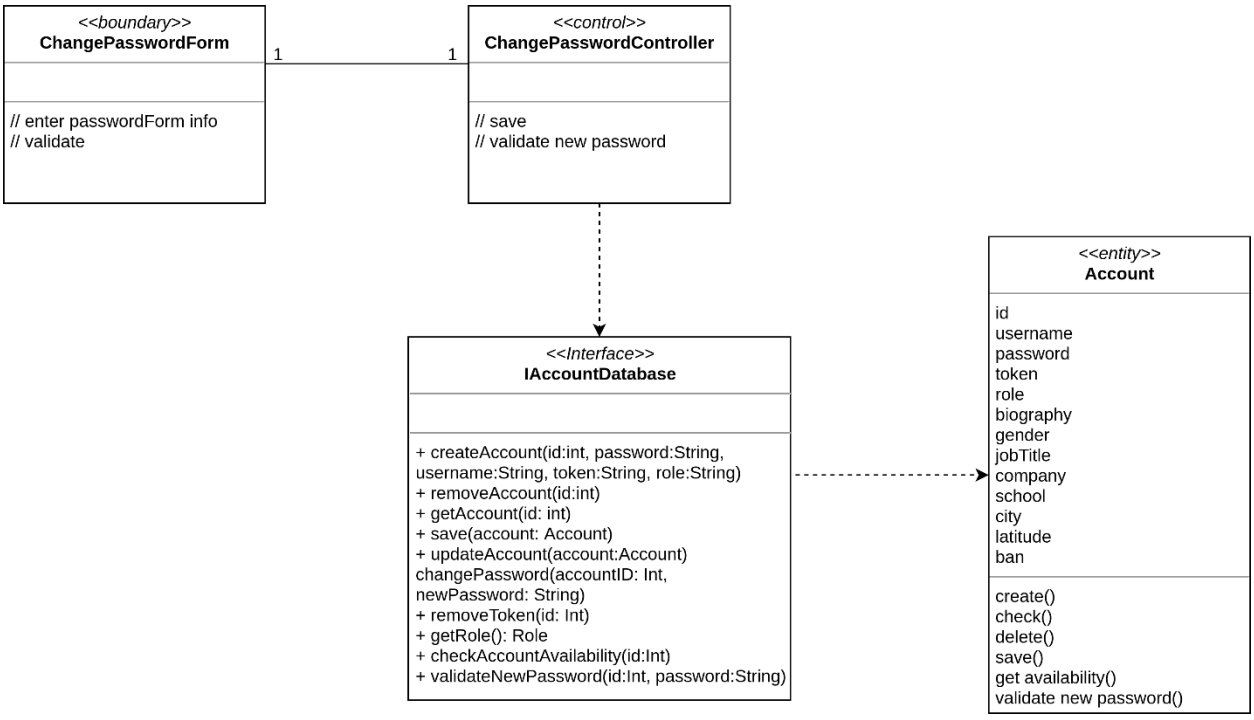
3.6.2 Design VOPC diagram for the Sign Out Use Case



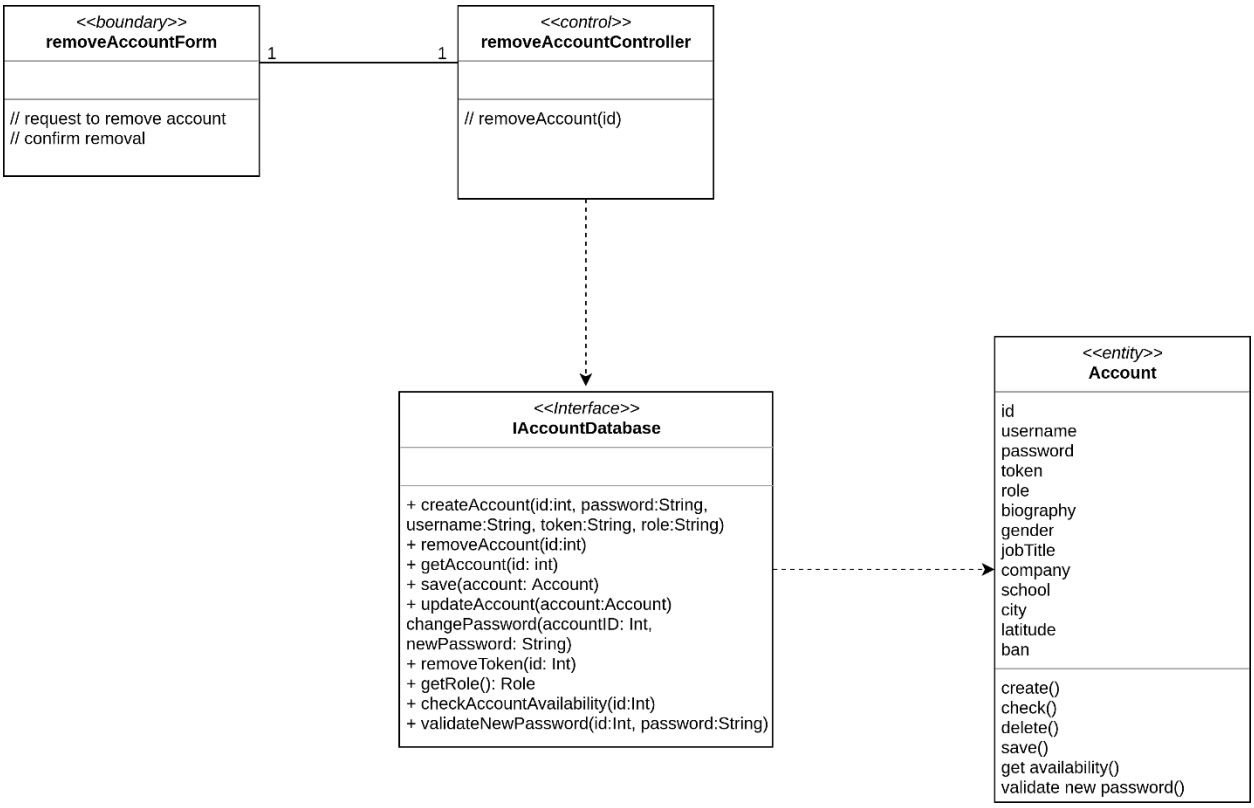
3.6.3 Design VOPC diagram for the Sign Up Use Case



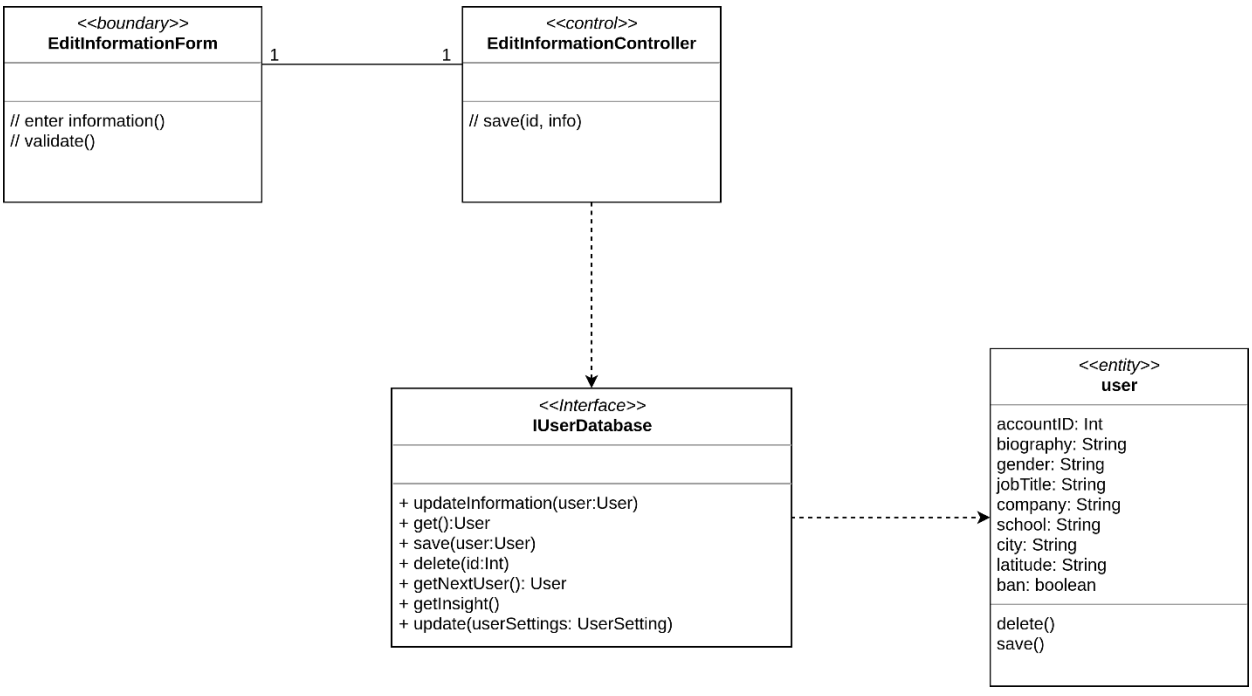
3.6.4 Design VOPC diagram for the Change Password Use Case



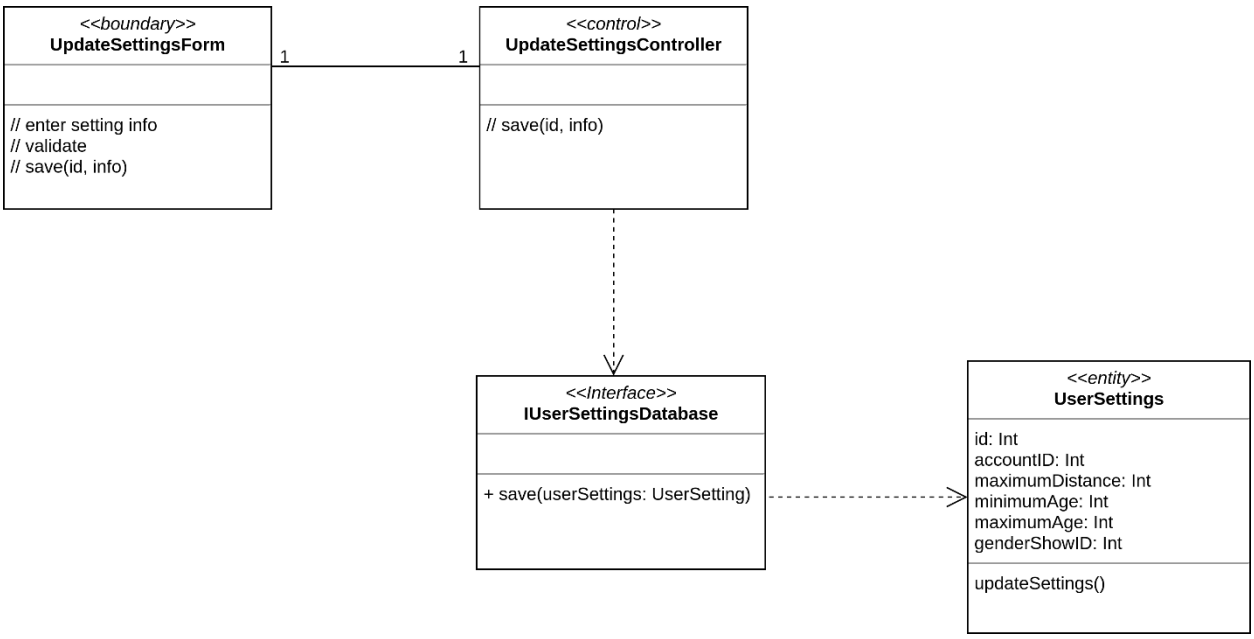
3.6.5 Design VOPC diagram for the Remove Account Use Case



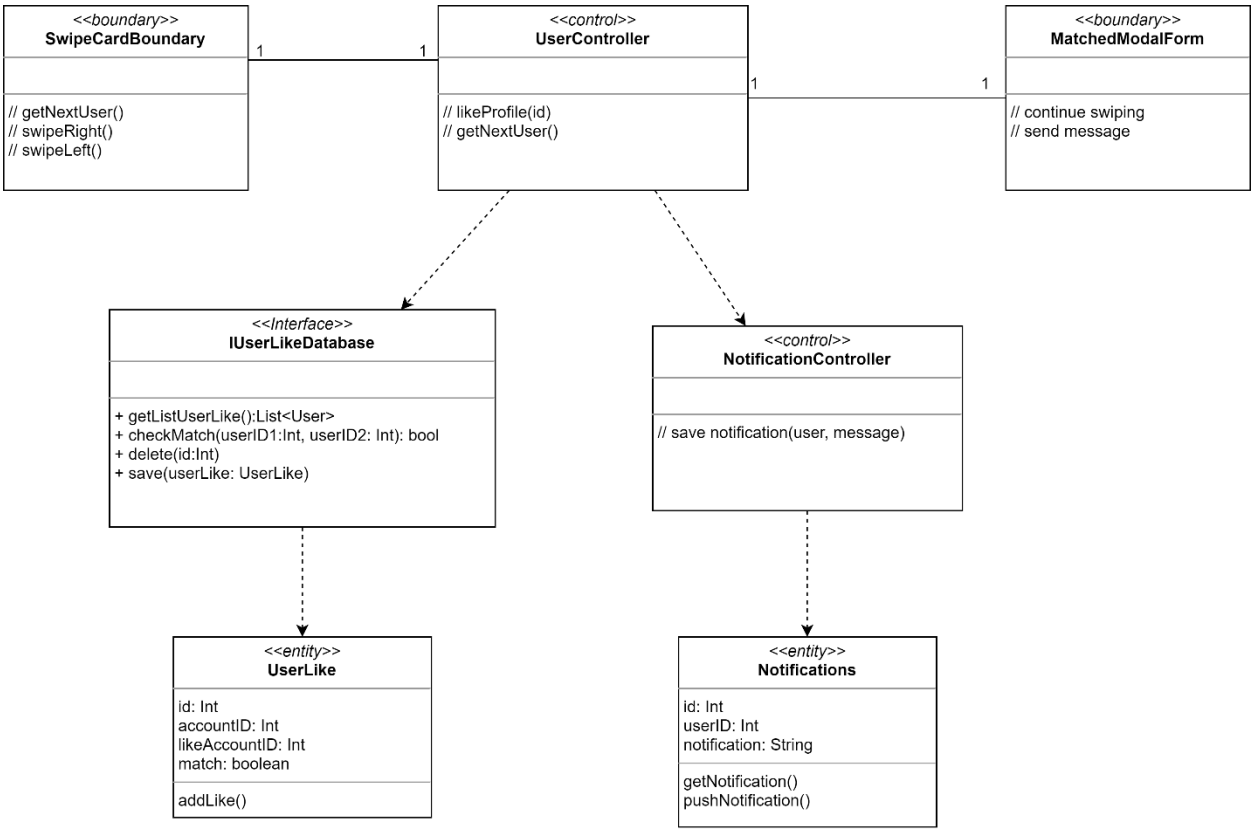
3.6.6 Design VOPC diagram for the Edit Information Use Case



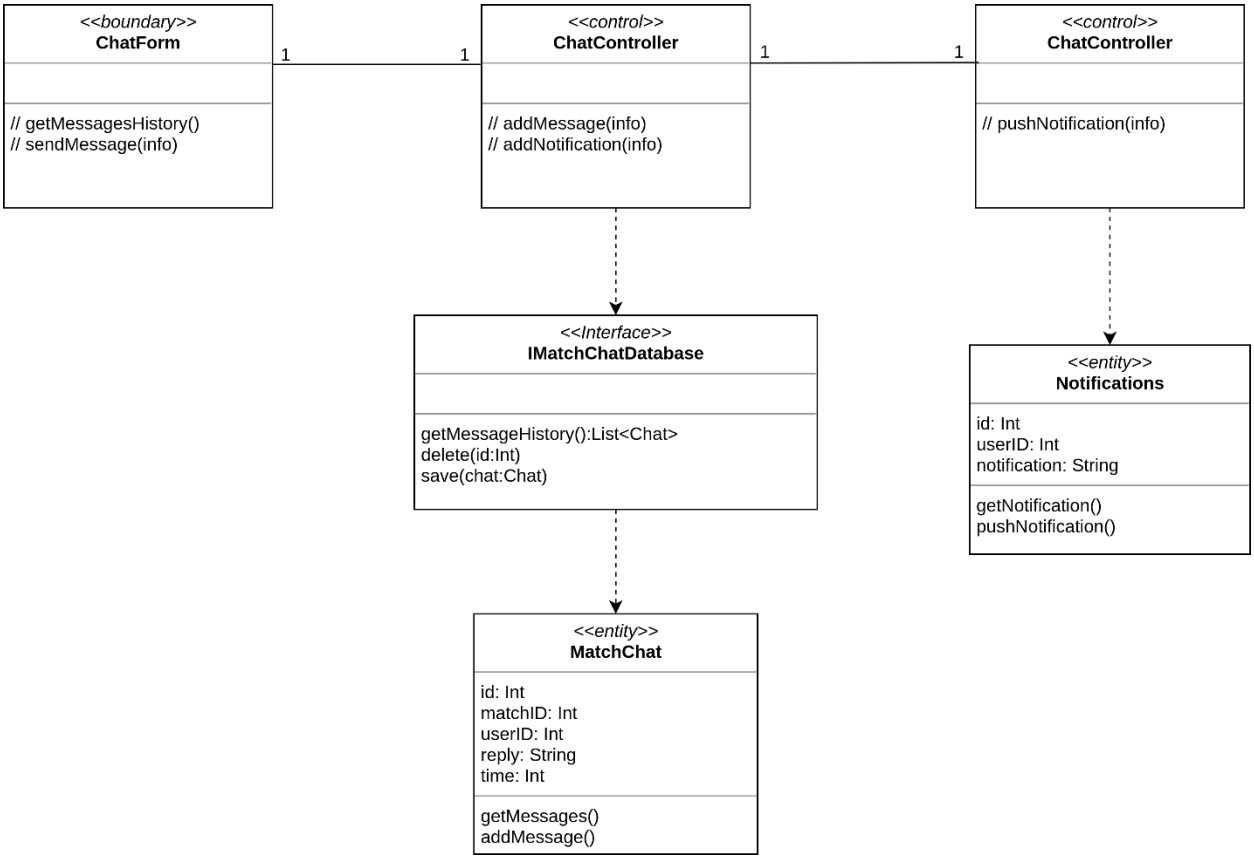
3.6.7 Design VOPC diagram for the Update Settings Use Case



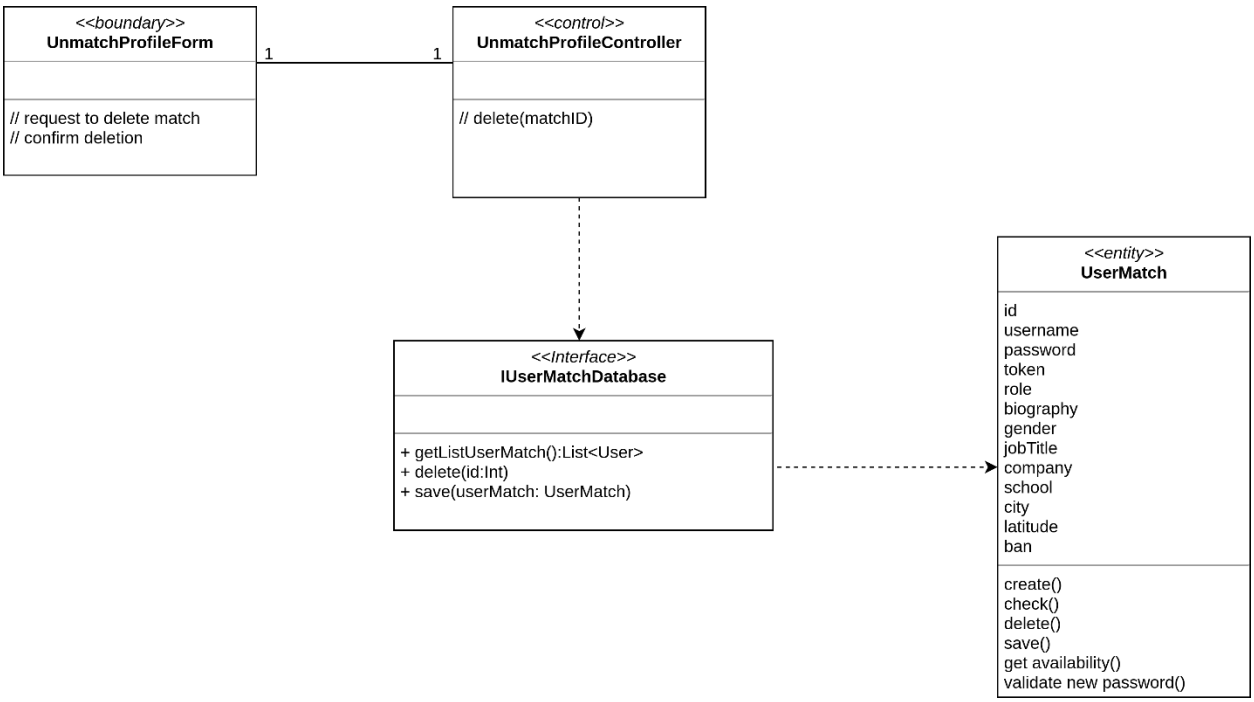
3.6.8 Design VOPC diagram for the Like/Dislike & Match Profile Use Case



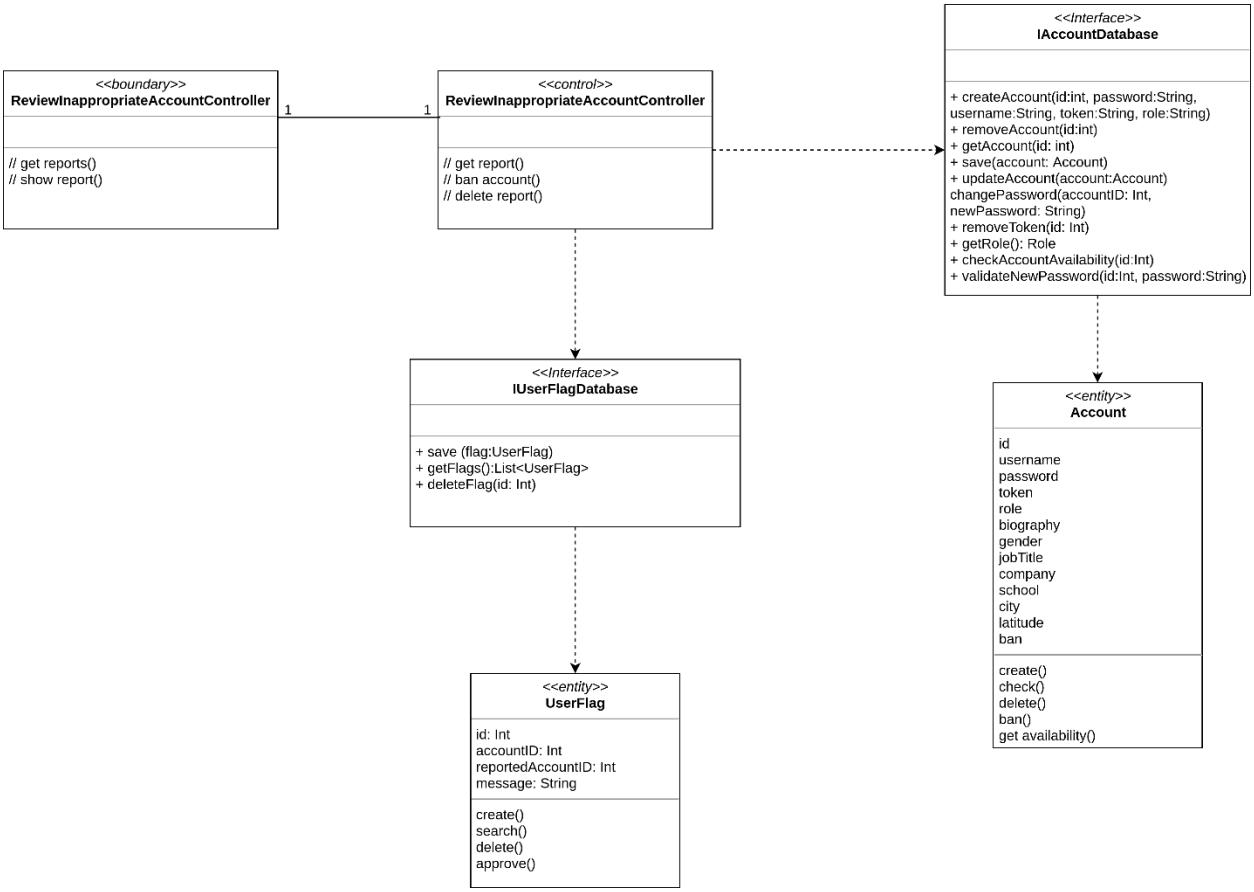
3.6.9 Design VOPC diagram for Chat Use Case



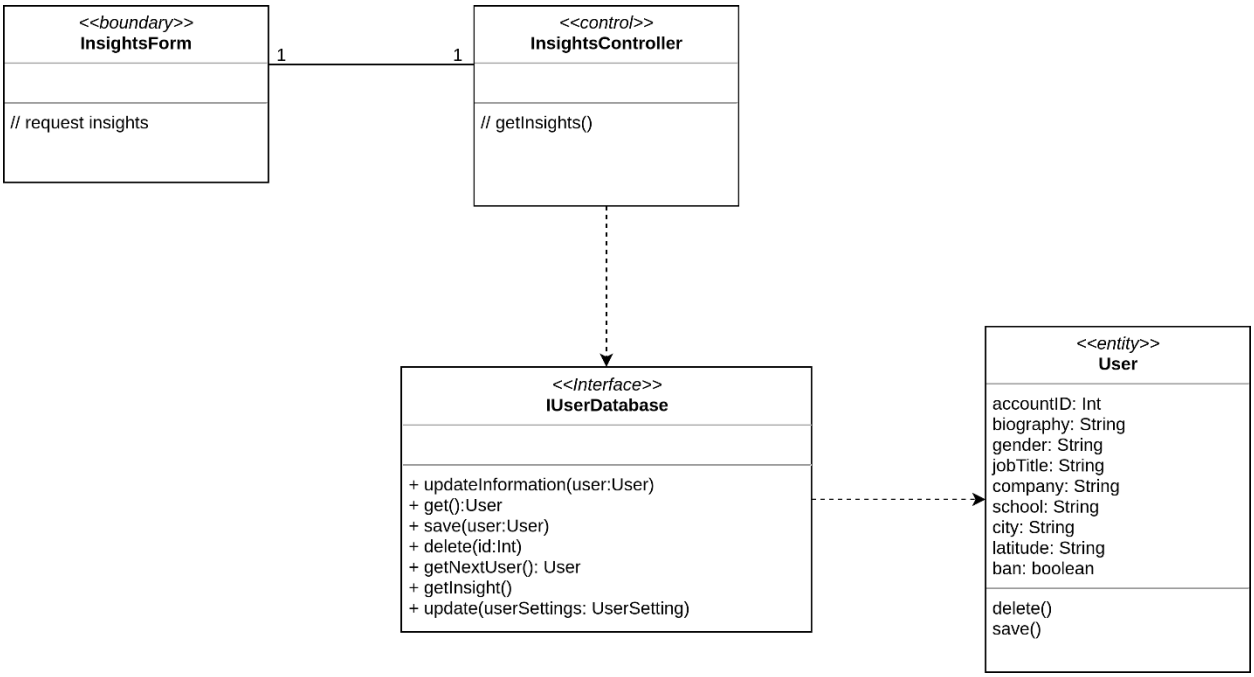
3.6.10 Design VOPC diagram for the Unmatched Use Case



3.6.11 Design VOPC diagram for the Review Inappropriate Account Use Case



3.6.12 Design VOPC diagram for the Get Insights Use Case



3.7. Database design