UE20CS352 Project
Object Oriented Analysis & Design with Java
Mini LinkedIn

Hita Juneja PES1UG20CS645
Ishita Bharadwaj PES1UG20CS648
Meghana N PES1UG20CS663
Policharla Sai Sailaja PES1UG20CS671

Section K
25th April, 2023.

Description

This is a Java Spring project based on designing a LinkedIn application using Object Oriented Approach and implemented it using JAVA that supports MVC Architectural Pattern. Project entails a full-stack web application where the user may register and log in to the service where they can create themselves a professional portfolio. User can invite other users to their contact network and praise their skills.

Project is built with Java Spring framework, Thymeleaf, HTML and Bootstrap CSS.

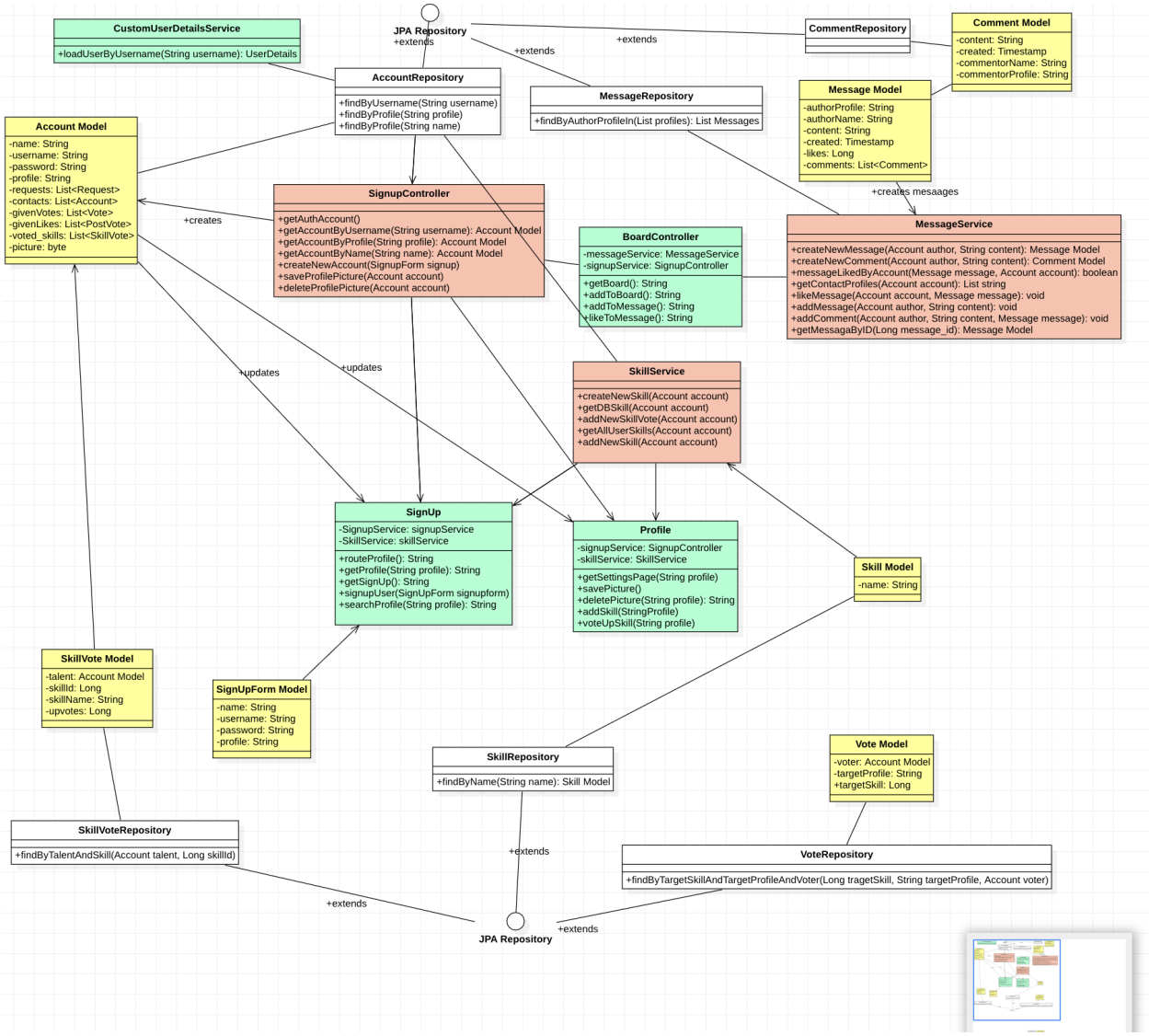
It uses H2 in-memory database with a SQL dialect to insert, fetch, update messages, posts, user accounts, comments, skills, skill votes and likes.

Service functionalities

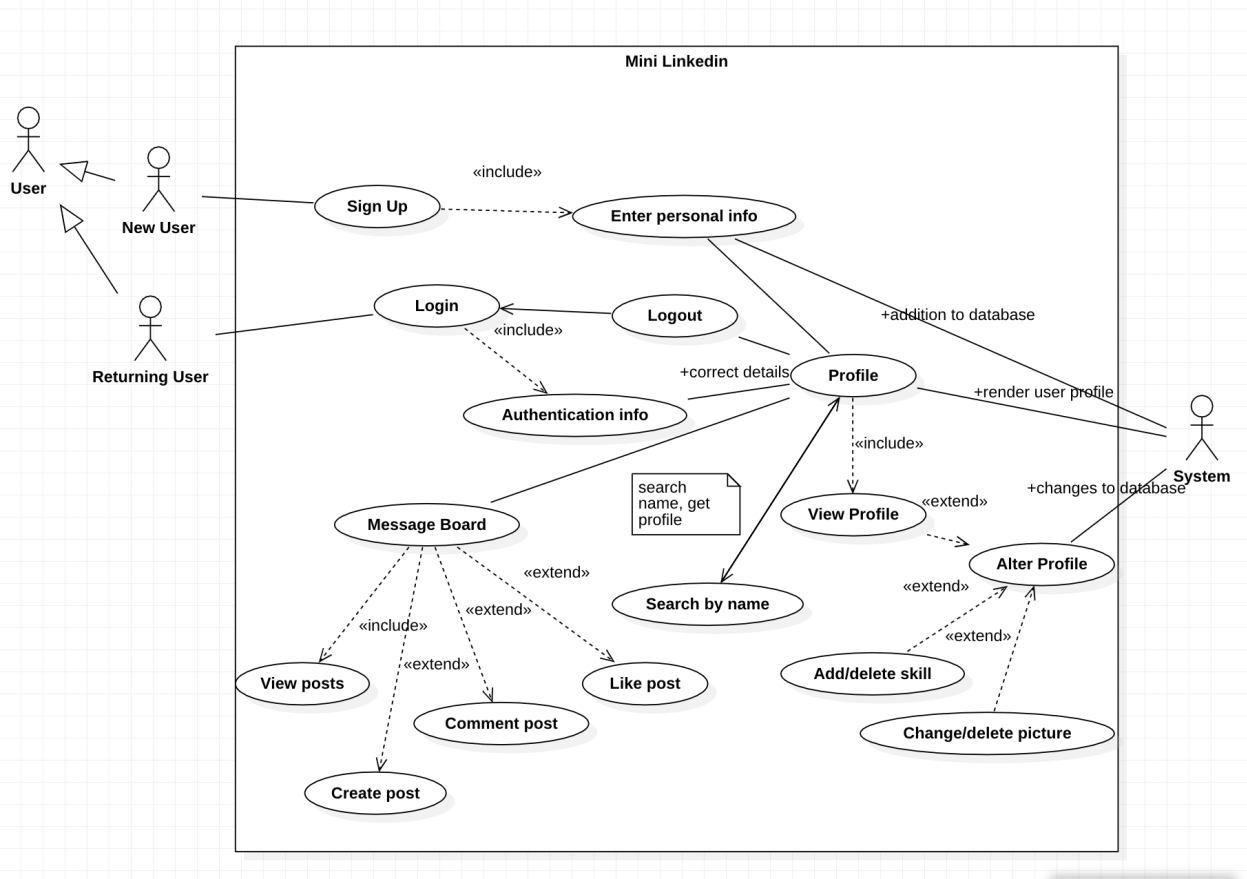
- Users may register to the service and log in and out to the service
- User has their own profile where they may add a profile picture and list skills for other users to see - like a small resume
- Logged in user can vote any one other user's skill. User's skill can be voted only once by the same user.
- Logged in user may ask another user to become their contact from this user's profile page. A request is sent and when this request is accepted these two users become contacts.
- Unlogged user may only view profiles and search them by users full name.

- Logged in user can post content in the message board. Content will be show to all contacts the user has.
- Contacts may comment and like each others' posts.

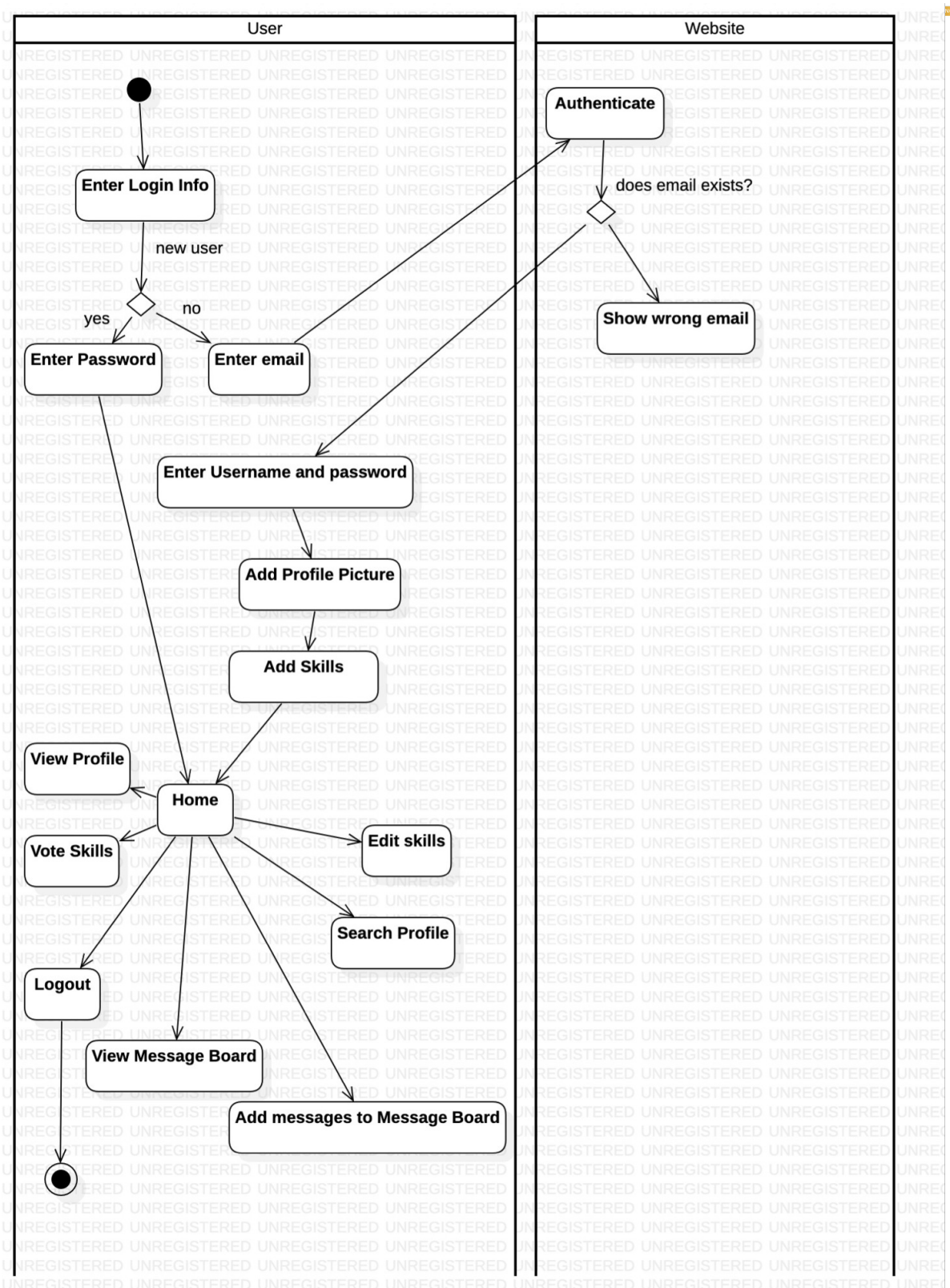
Class Diagram



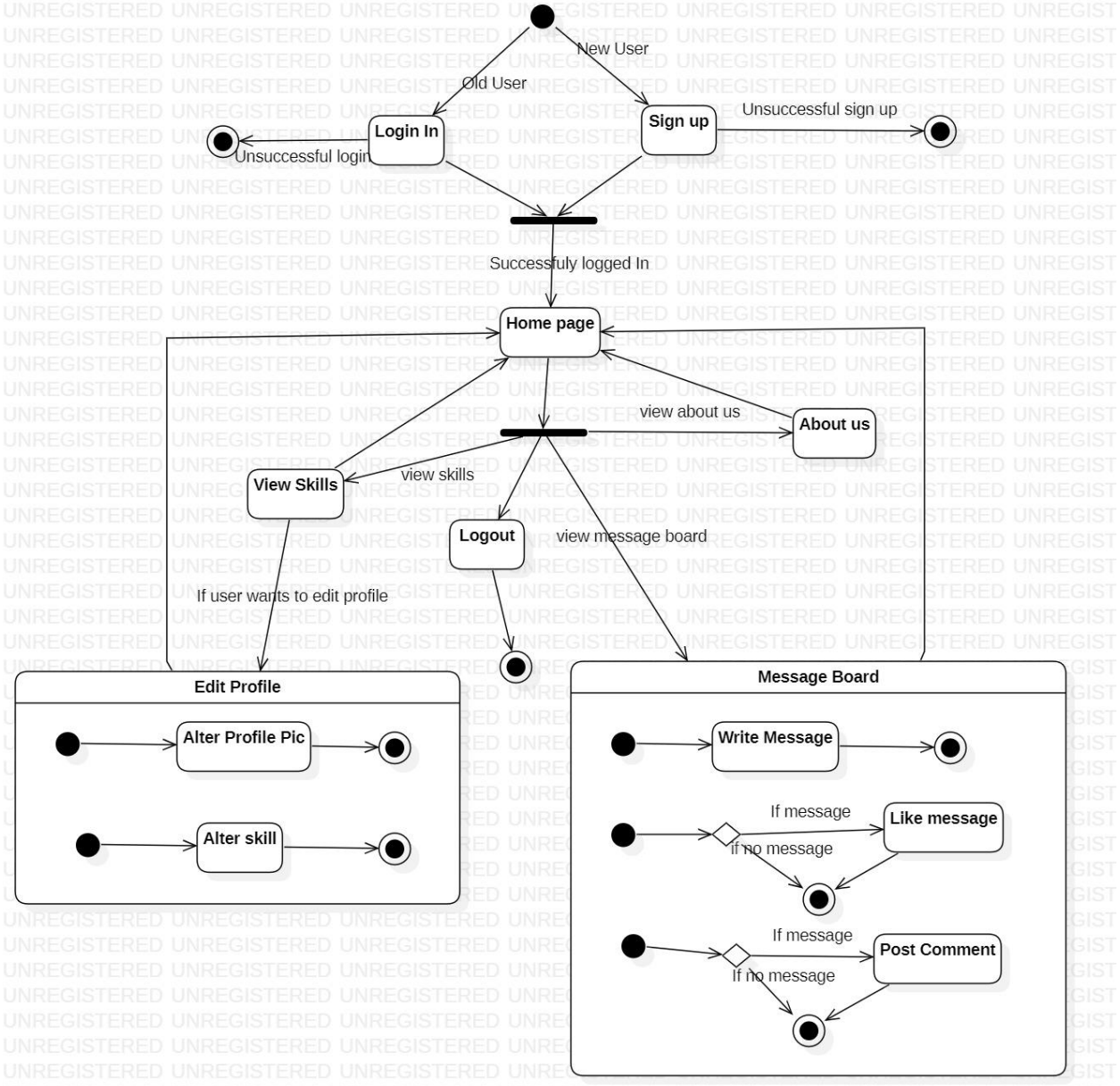
Use Case


Mini Linkedin
User
New User
Returning User
Sign Up
«include»
Enter personal info
Login
Logout
«include»
+addition to database
+correct details
Profile
+render user profile
Authentication info
«include»
System
search name, get profile
+changes to database
Message Board
View Profile
«extend»
Alter Profile
«extend»
«extend»
Search by name
«include»
«extend»
«extend»
Add/delete skill
View posts
Like post
Comment post
Change/delete picture
Create post

Activity Diagram



State Diagram



Design Pattern

Repository Pattern

In this project, the JPA repository class contains all persistence-related code but no business logic. It provides methods to per-sist, update, and remove an entity or methods that instantiate and execute specific queries. The goal of this pattern is to separate persistence-related code(Repository classes) from business code(Service and Controller classes) and to improve the reusability of persistence-related code. It also makes business code easier to read and write. It enables us to generate repositories easily for the various entities. Repository interface defines a set of methods for standard write operations, such as save, delete, and read operations. Skill Repository, SkillVote Repository, Account Repository,
Message Repository and Comment Repository all extend the JPA repository.

Design Principles

Interface Segregation Principle

Interface segregation principle (ISP) states that no code should be forced to depend on methods it does not use. ISP splits interfaces that are very large into smaller and more specific ones so that clients will only have to know about the methods that are of interest. This is why we have separate interfaces that implement on the classes that are related to them. These include Repository, SkillVote Repository, Account Repository, Message Repository and Comment Repository.

Single Responsibility Principle

The single-responsibility principle (SRP) states that "A module should be responsible to one, and only one, actor."
As shown in class diagram, since we have used MVC architecture pattern, there is only 1 model entity class, 1 controller class, 1 view/UI class for a particular use case. There aren't 2 or more classes that execute the same functionality. Every class is assigned a single responsibility.

Indirection

The Indirection pattern is another way to reduce coupling by creating an intermediary class (or any kind of component) between two classes. Indirection introduces a layer between components in order to reduce coupling. This is shown by our service classes, MessageService, UserDetailsService, SkillService that act as an intermediary layer between their respective model and controller classes and reduce coupling.

High Cohesion

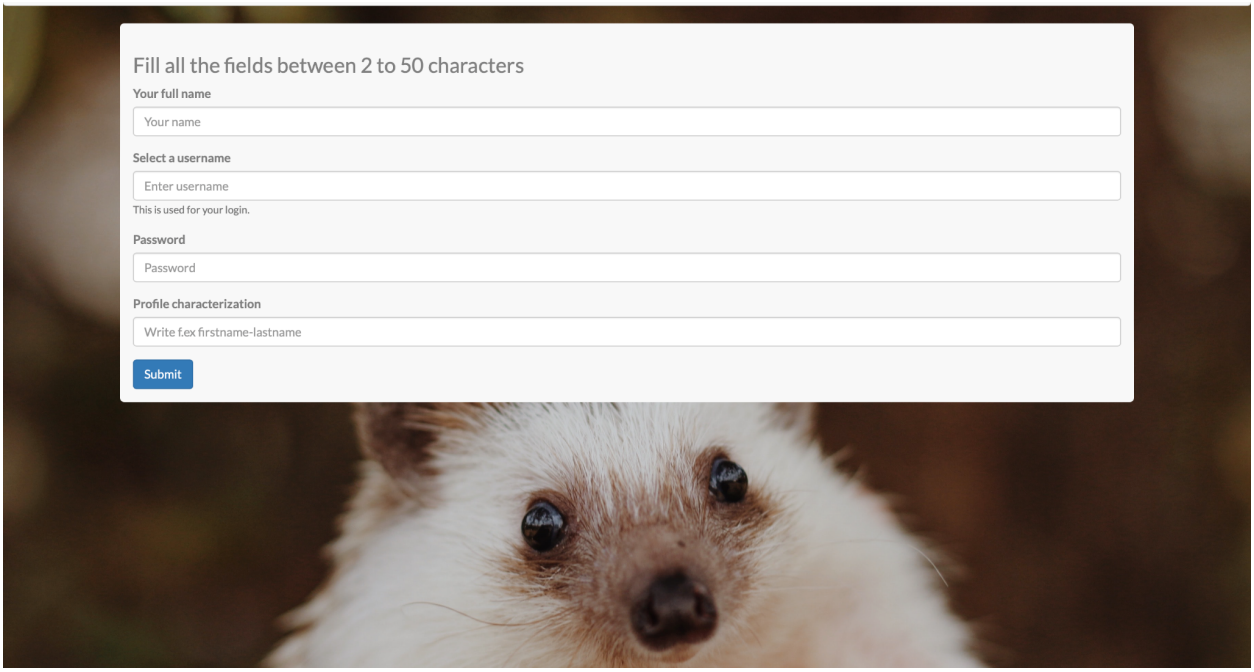
Related responsibilities are in to one manageable unit. High cohesion clearly defines the purpose of the element. It provides benefits of code reuse and low coupling. All operations in class represent a central purpose to implement all operations related to the message board.(read/retrieve message, post message, comment on message, delete message, like message).

Code (Github link)

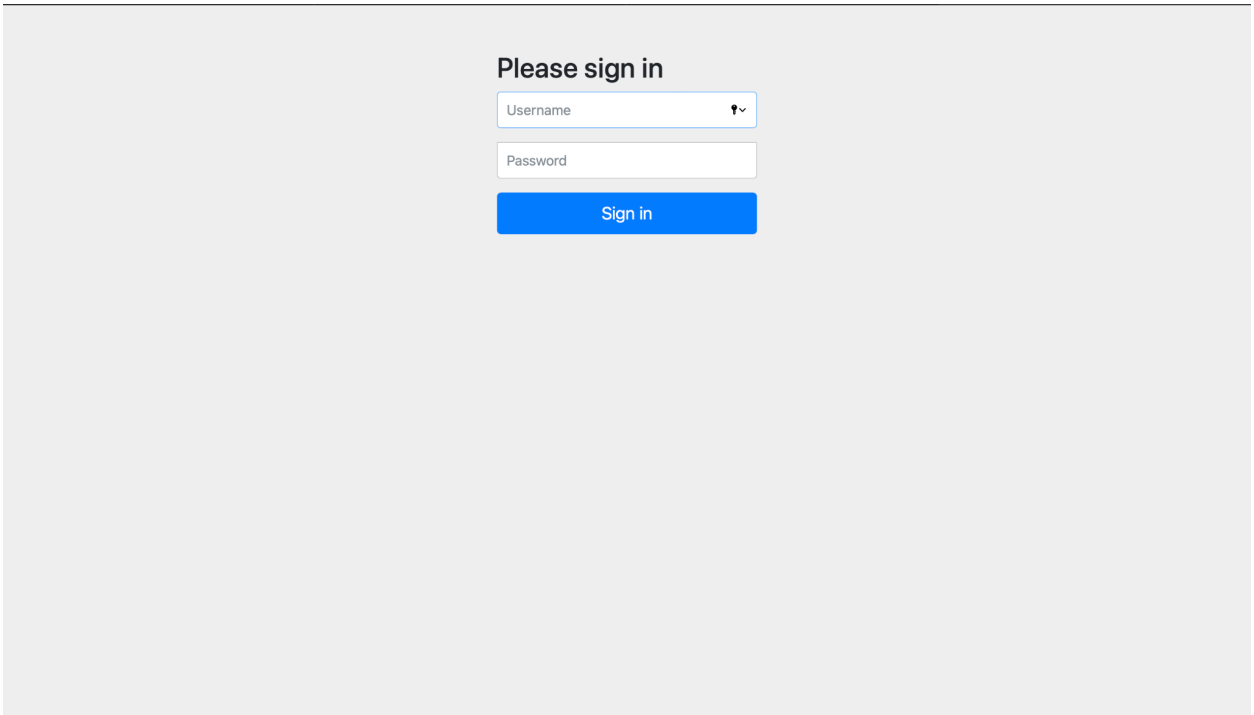
https://github.com/IshitaBharadwaj/OOAD_LinkedIn

OUTPUT

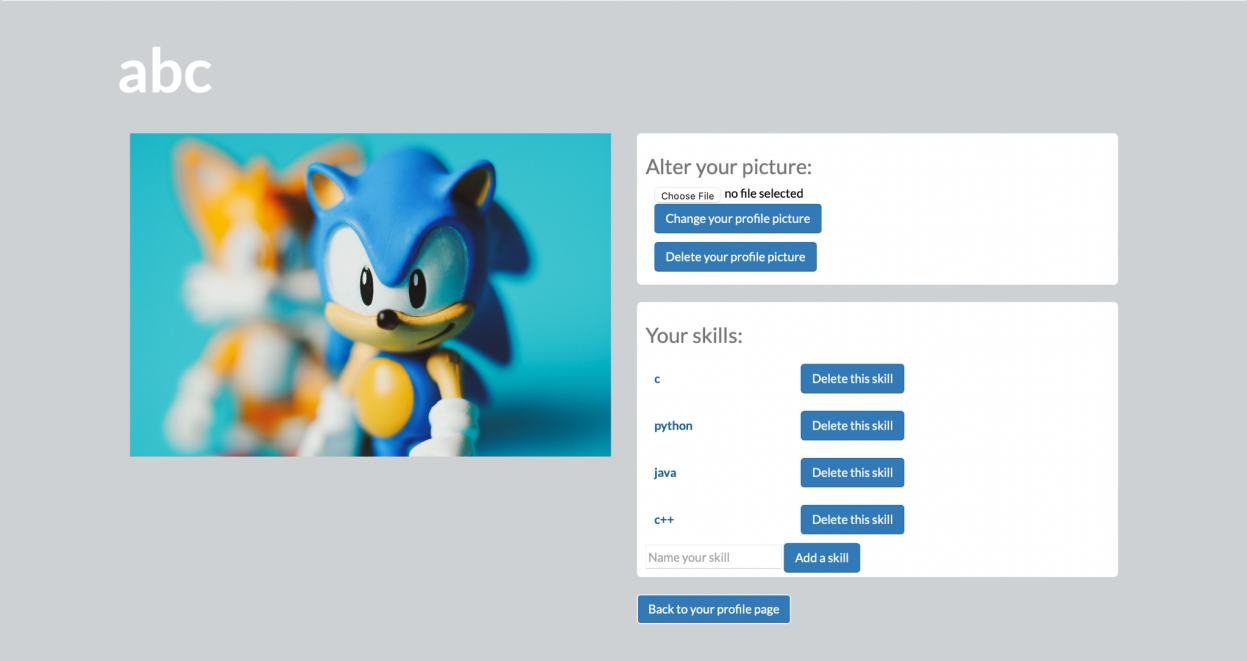
Signup page



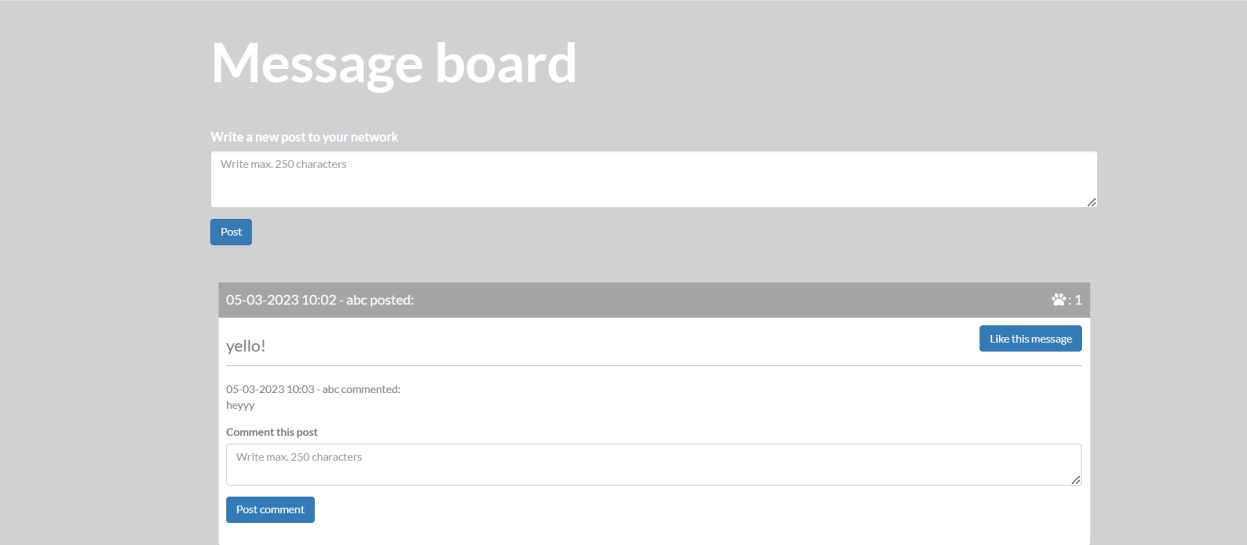
Login Page



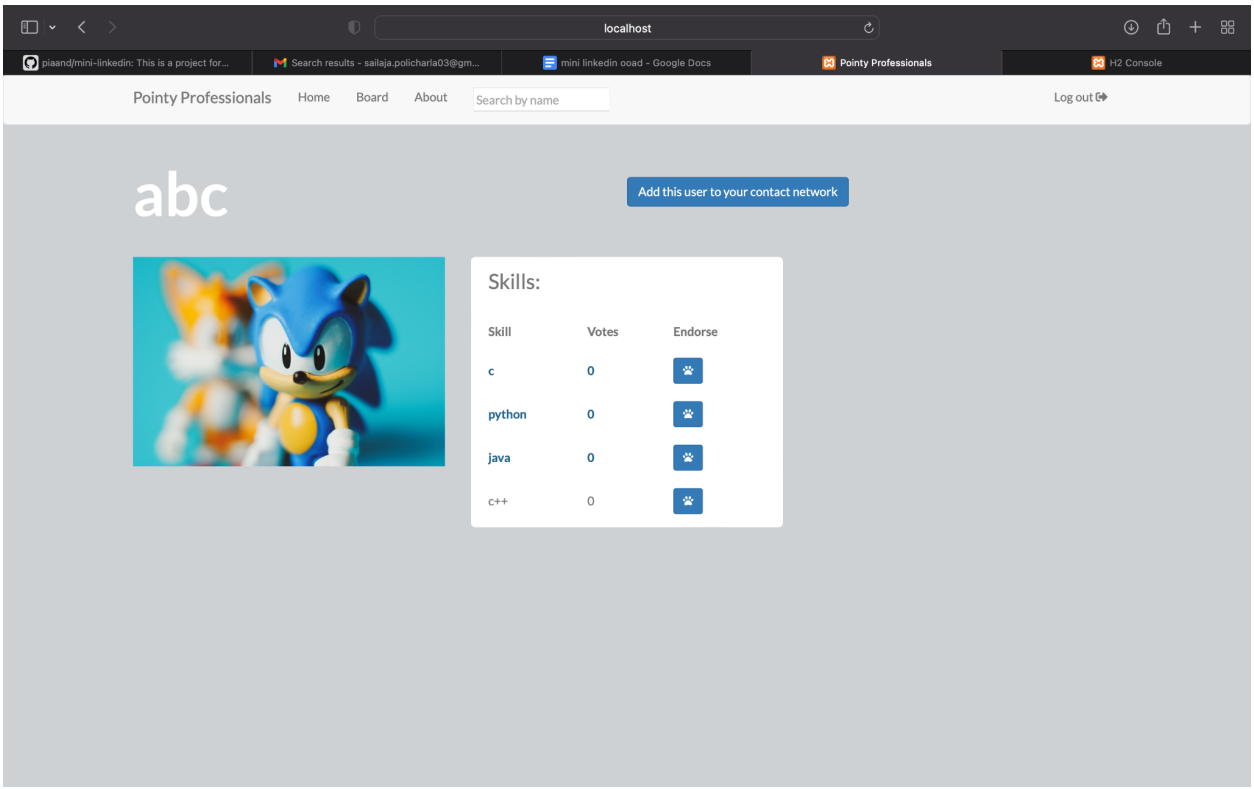
Profile



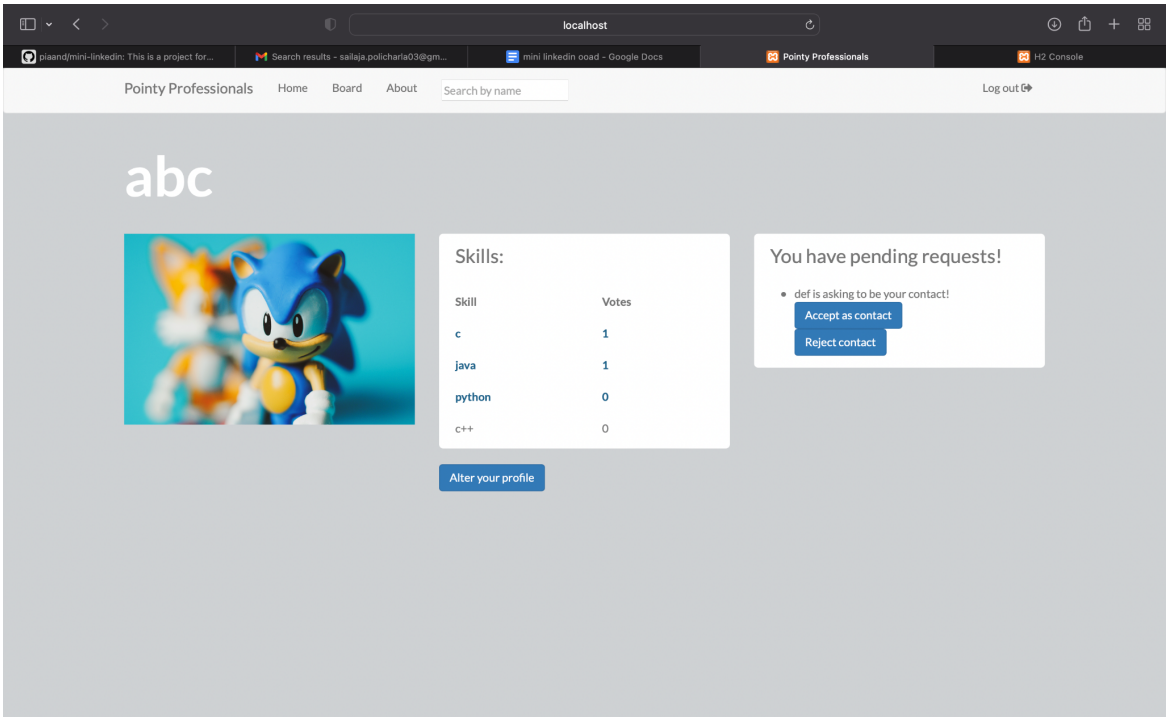
Message Board



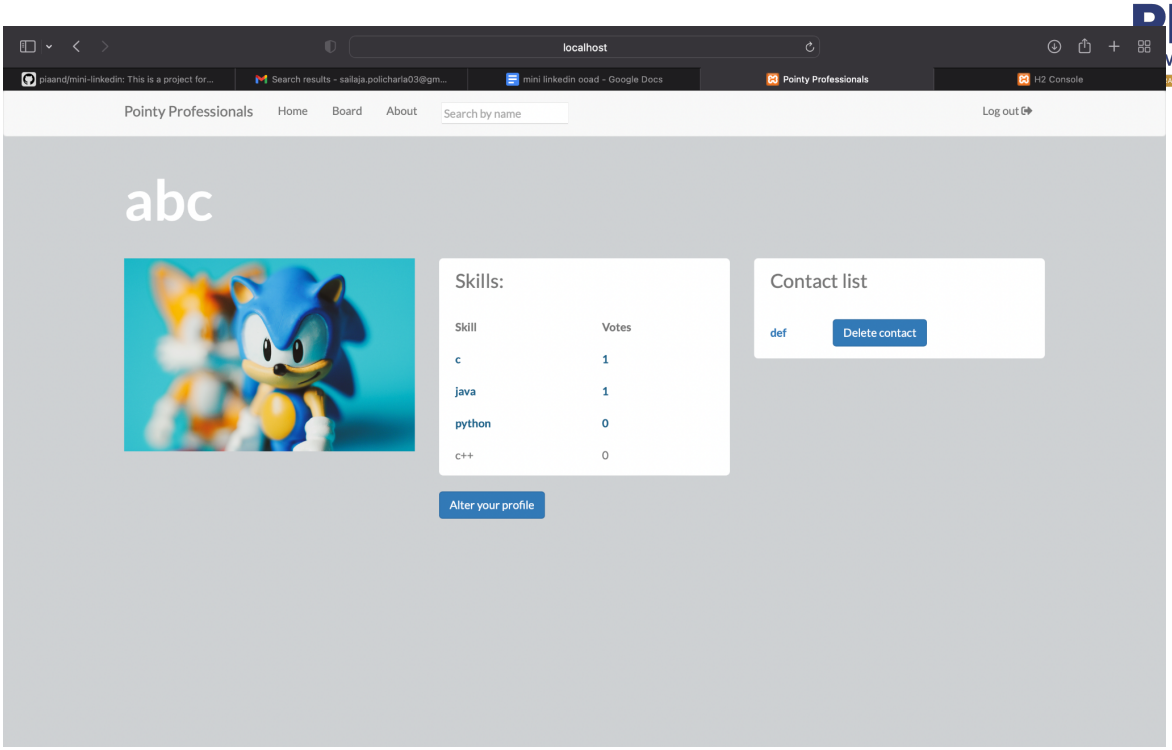
Signed in as user “def” and searched for user “abc”



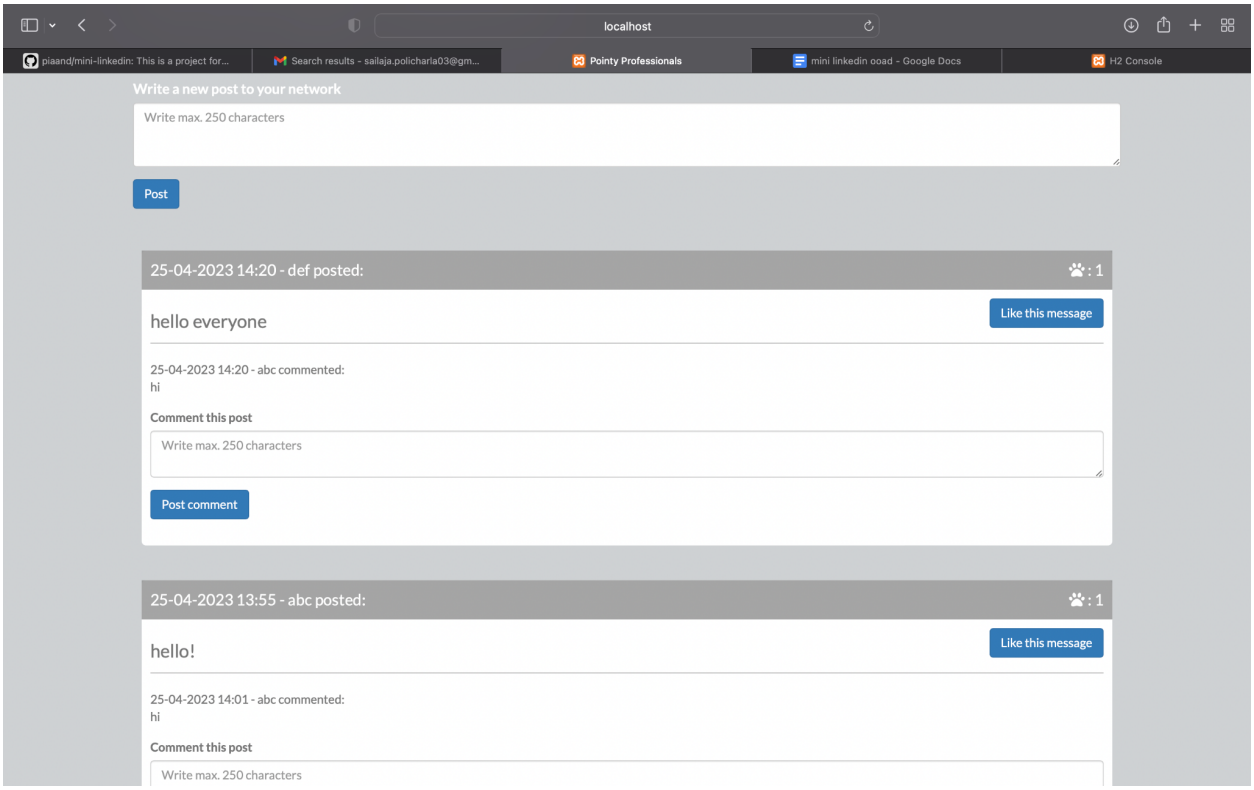
Pending Requests showing on user profile



After accepting connection request by user “abc” from user “def”, “def” is added to the contact list of user abc



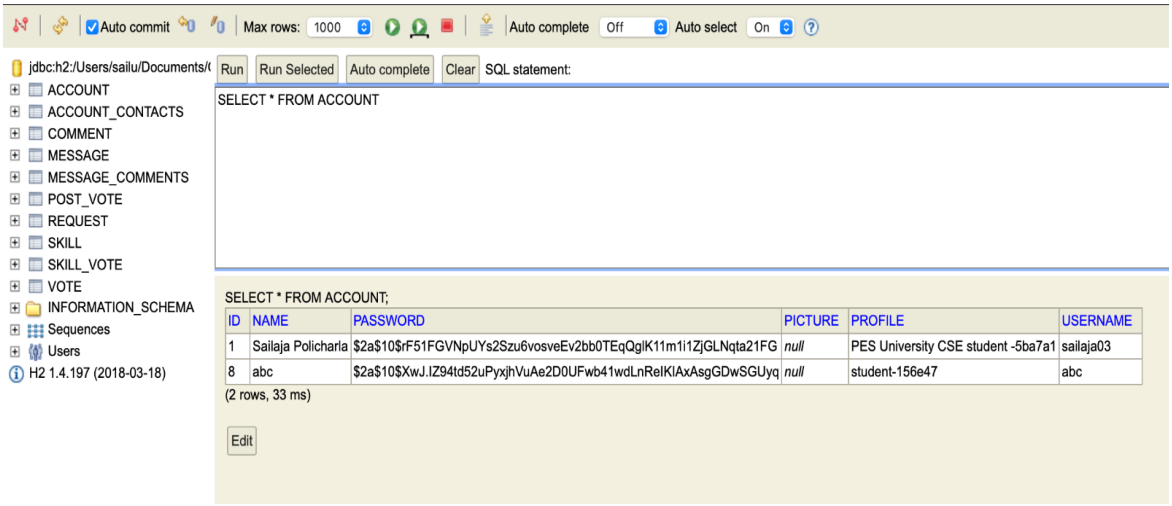
Message posted by user “def” was commented by user “abc”



DATABASE

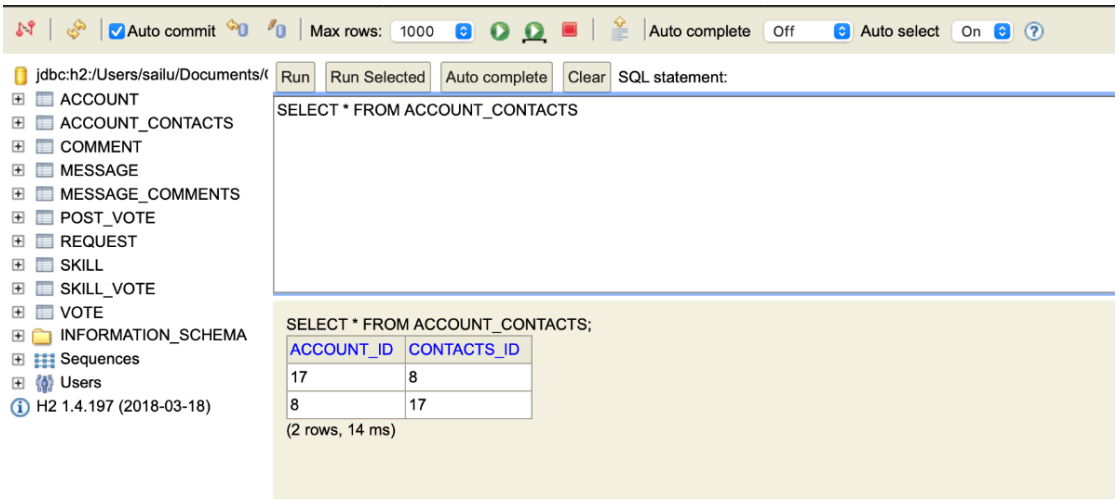
ACCOUNT

Account table provides information about the user account like name, their password, picture, profile and their username that they use to login.



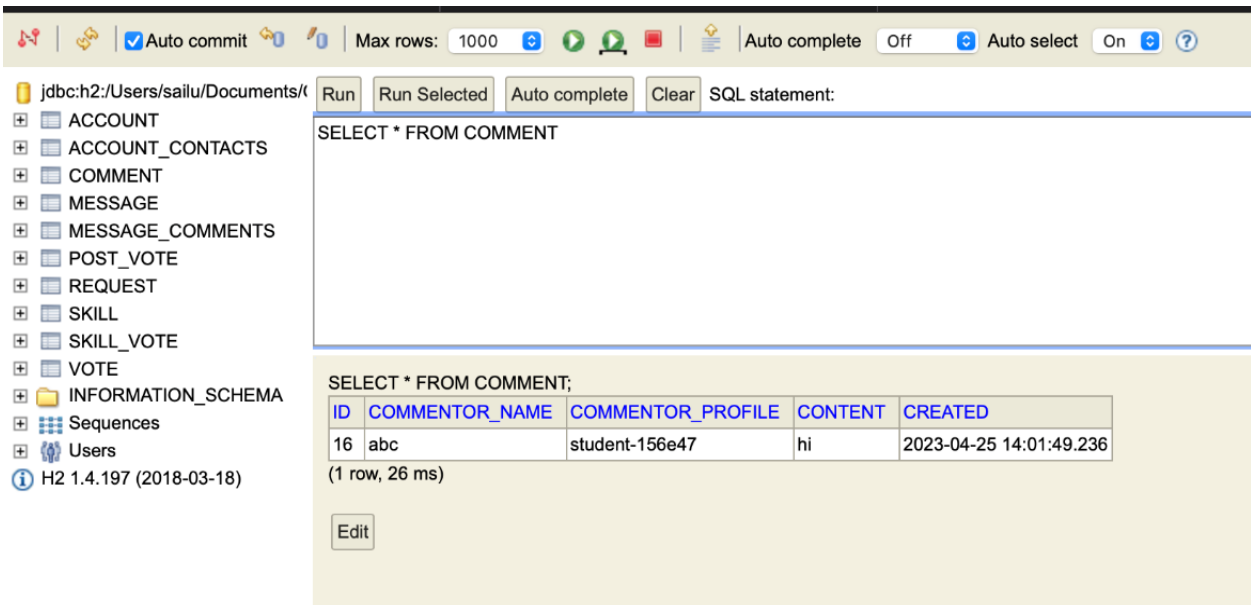
ACCOUNT CONTACTS

This shows the connections that one account has by showing the account ID and the account ID it has as part of it's connections



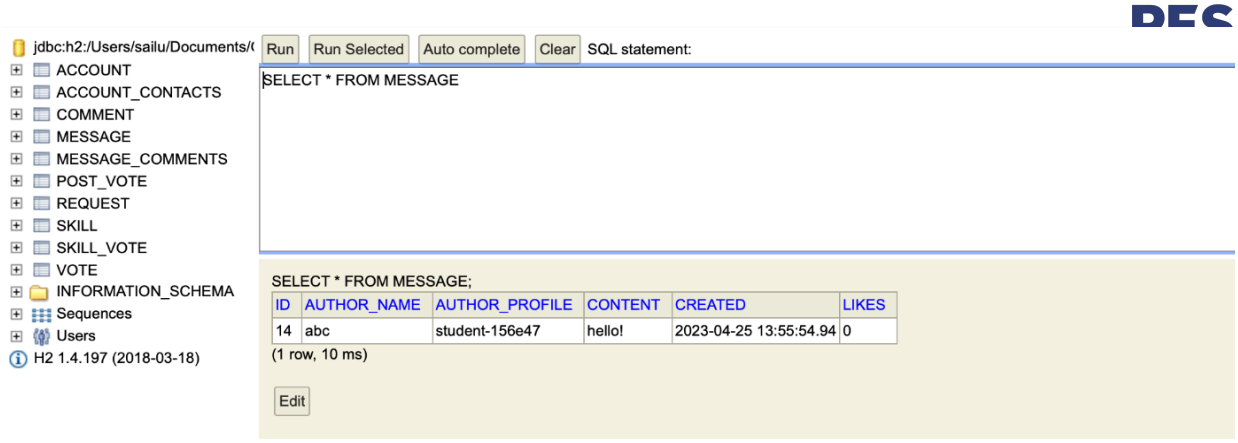
COMMENT

This table shows which account has commented what to a message posted by the owner

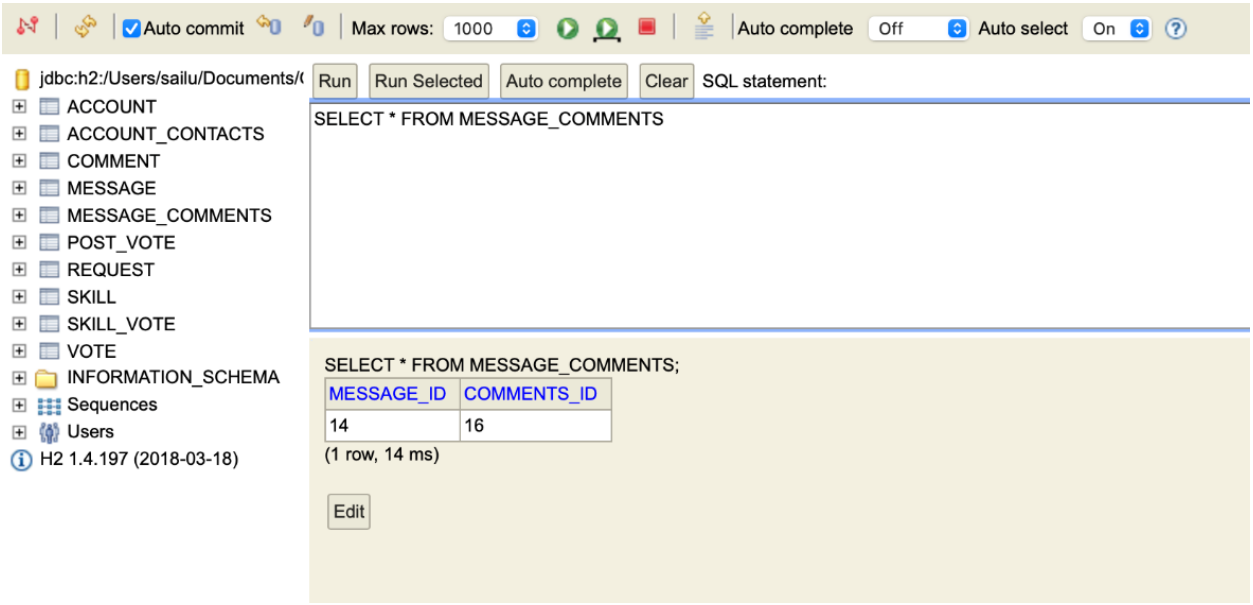


MESSAGE

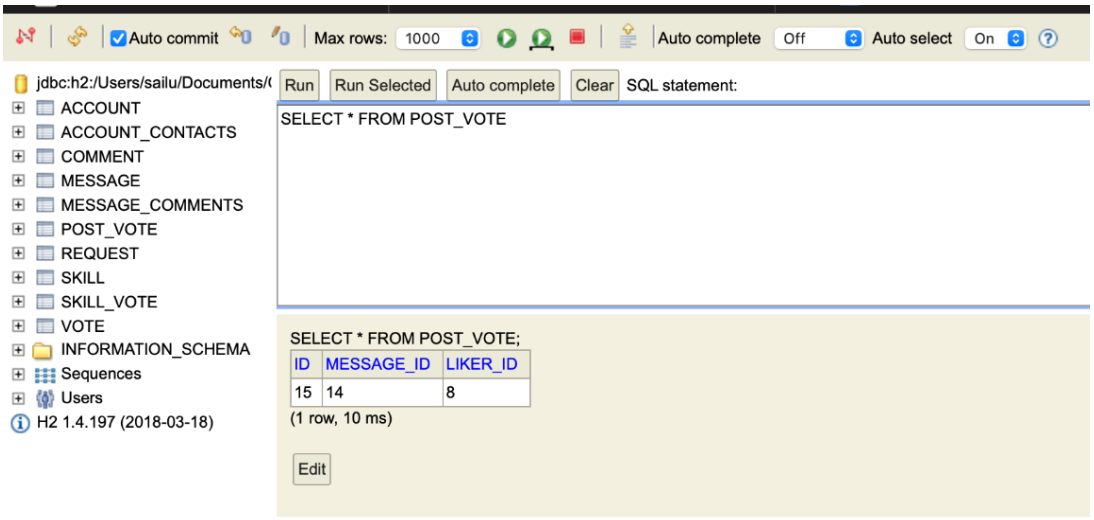
This database shows the message that is posted on the respective user's message board



MESSAGE COMMENTS

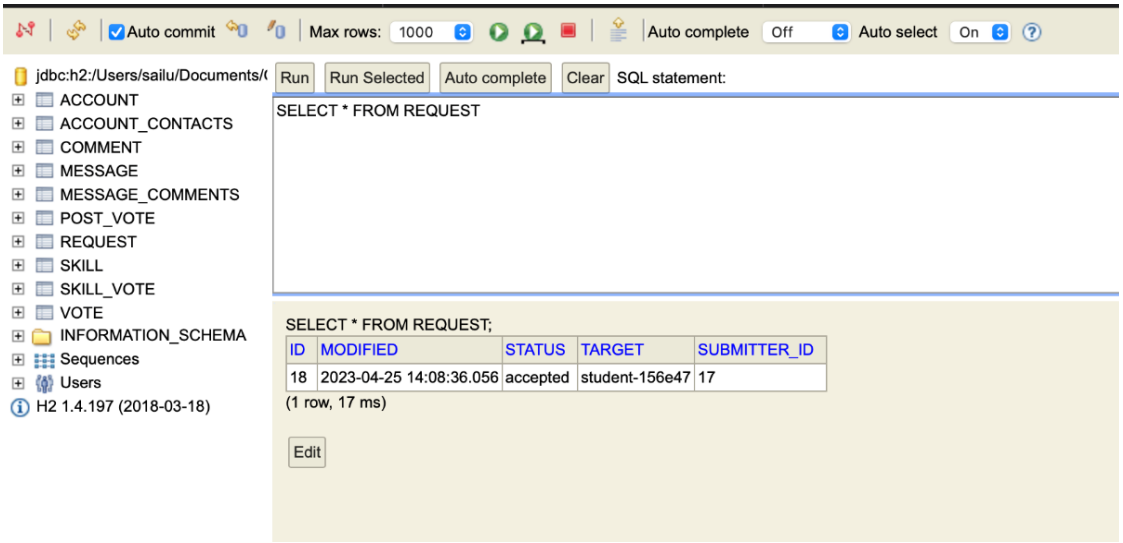


POST_VOTE



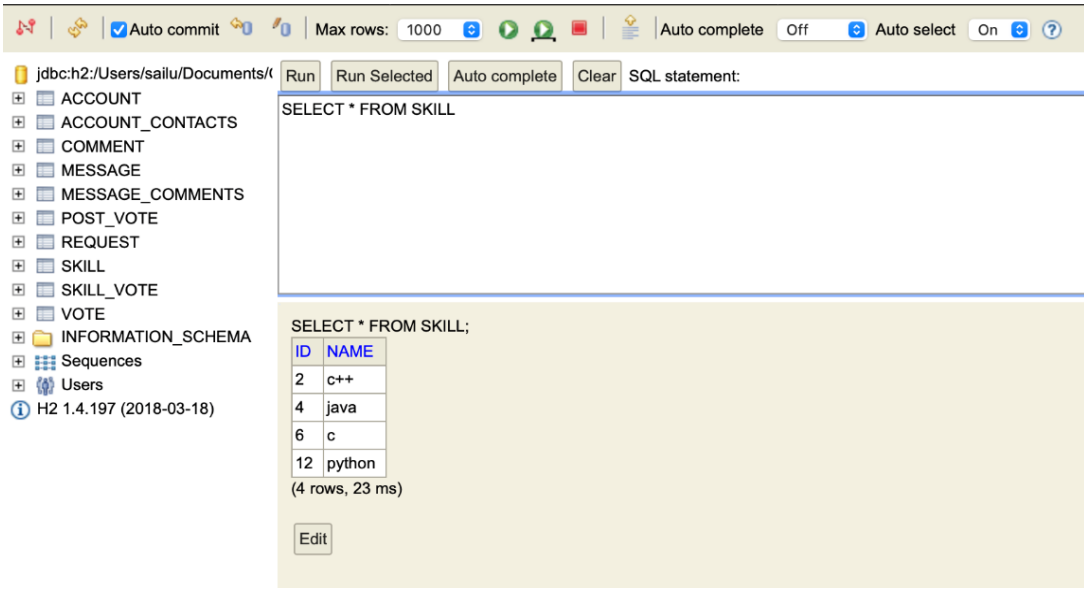
REQUEST

This table shows the details of the account that has sent a connection request to another account and the status of the connection



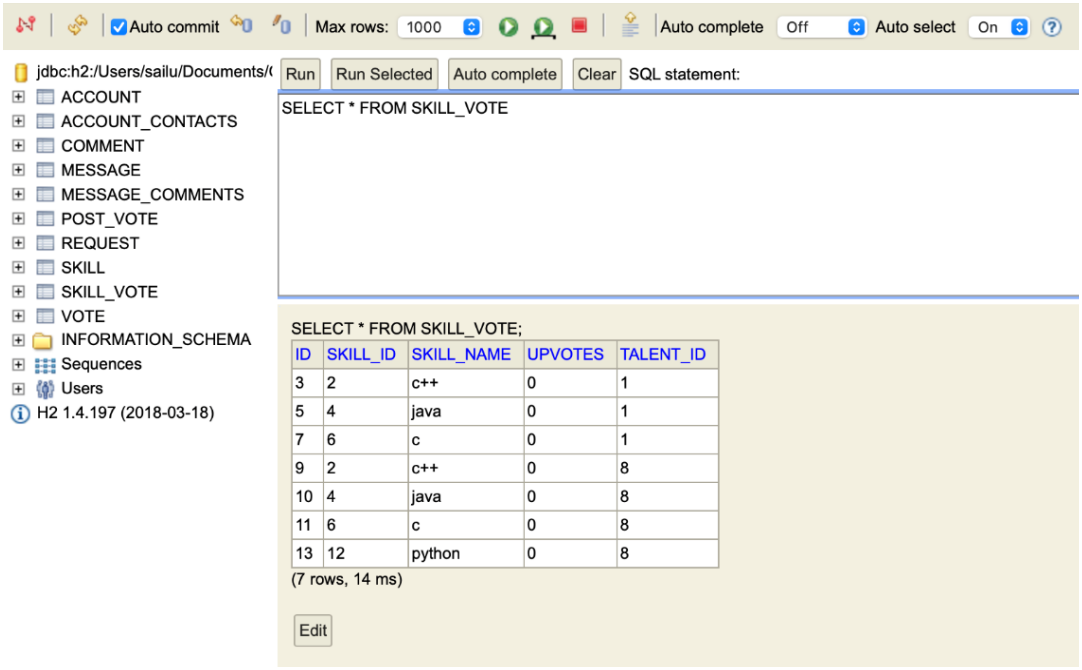
SKILL

This table shows the particular skills that are added by the user to their profile



SKILL VOTE

This table shows the skills of a particular account and the number of accounts that have endorsed these skills



VOTE

This shows the target profile and the skills that were voted