PES
UNIVERSITY
CELEBRATING 50 YEARS

UE20CS352 Project
Object Oriented Analysis & Design with Java
Mini LinkedIn

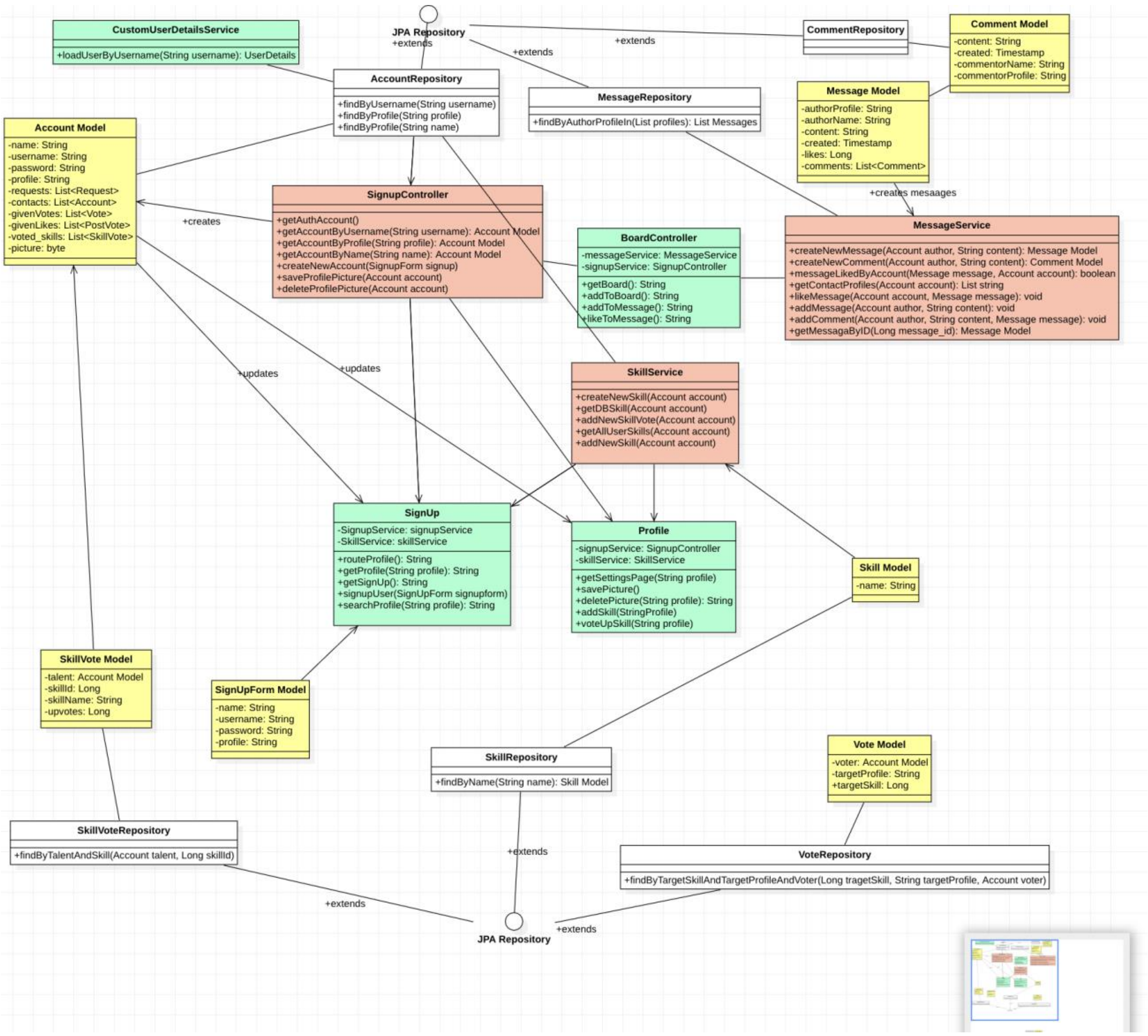
Hita Juneja PES1UG20CS645
Ishita Bharadwaj PES1UG20CS648
Meghana N PES1UG20CS663
Policharla Sai Sailaja PES1UG20CS671

Section K
25th April, 2023.

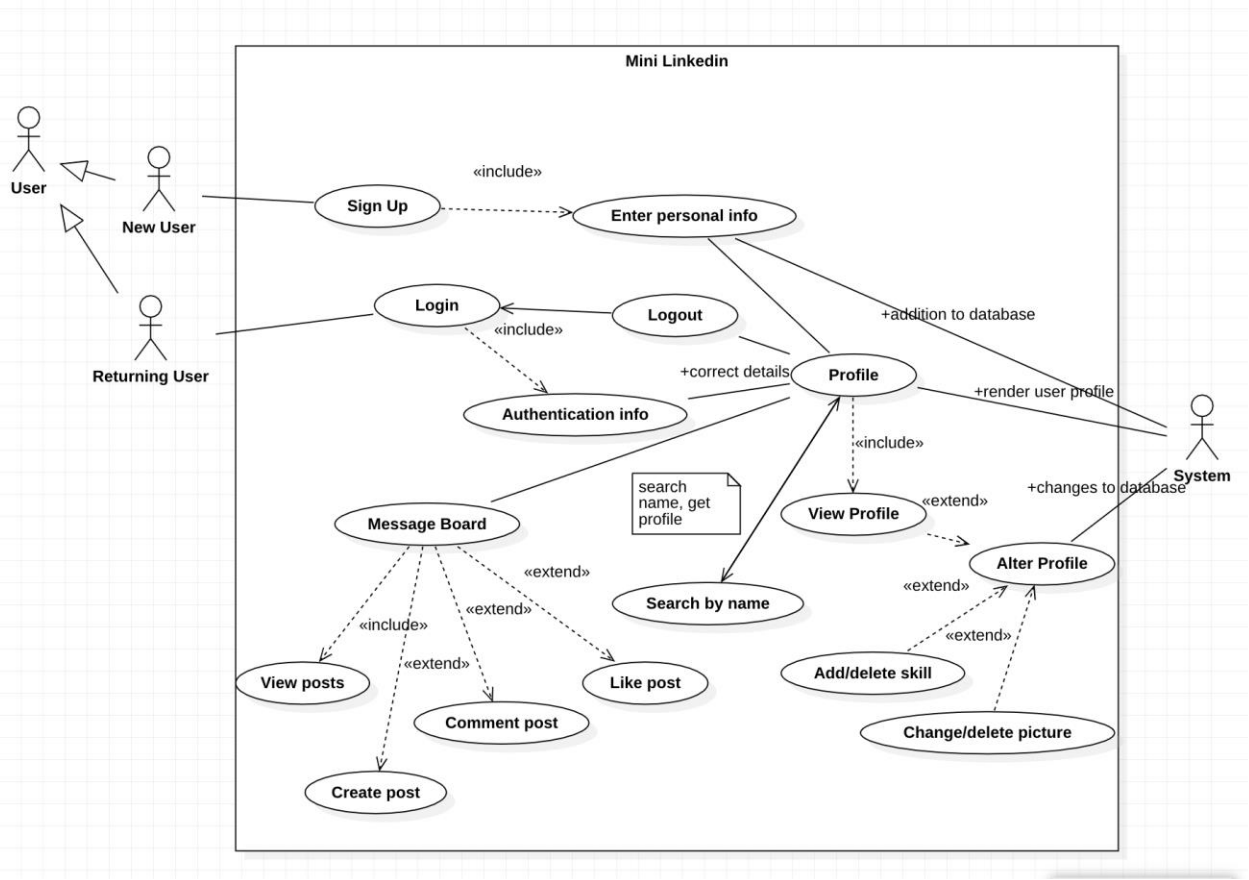
Description

This project is a social media java spring boot application.
It uses H2 in-memory database with a SQL dialect to insert, fetch, update messages, posts, user accounts, comments, skills, skill votes and likes.

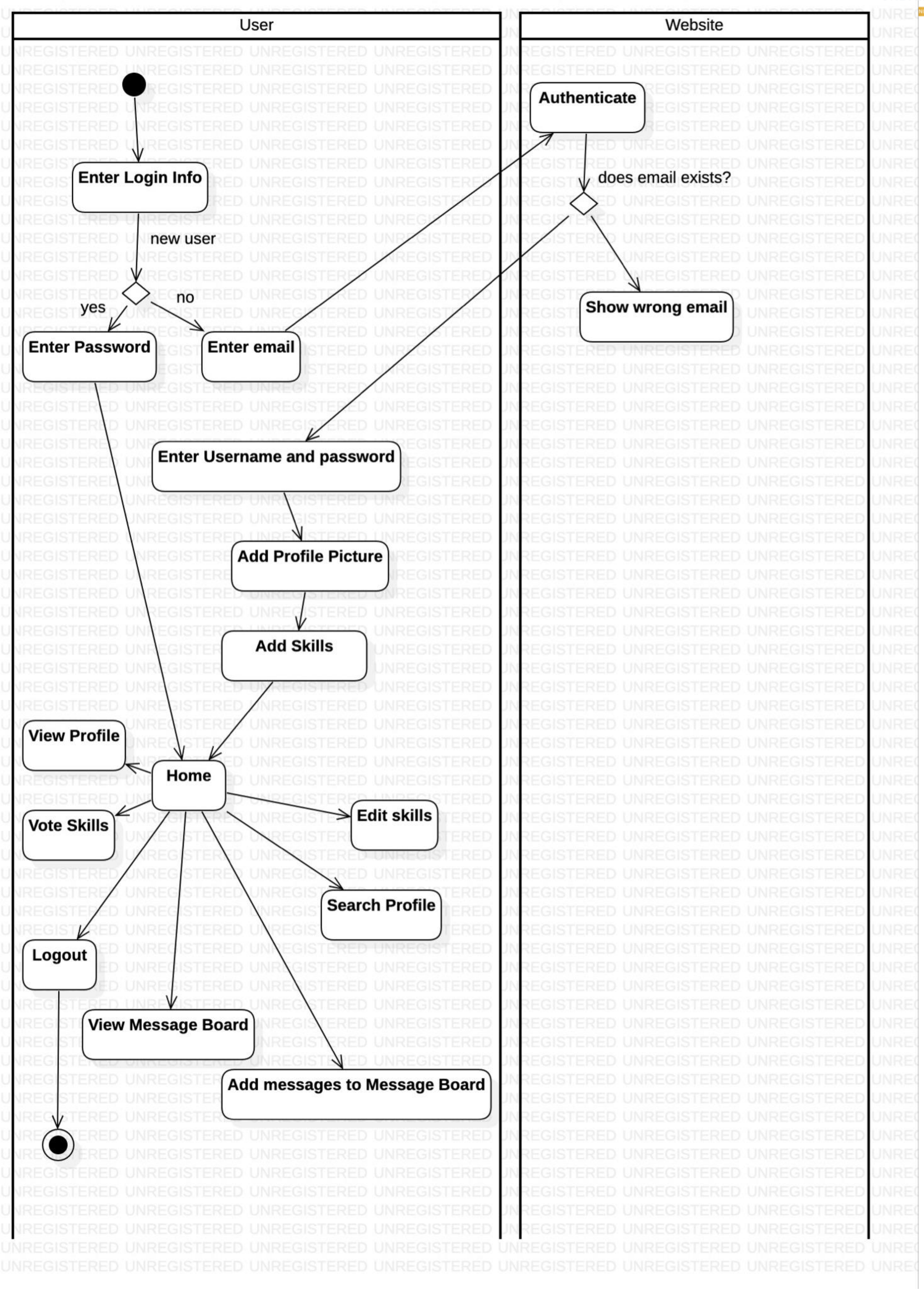
Class Diagram


CustomUserDetailsService
+loadUserByUsername(String username): UserDetails
JPA Repository
+extends
+extends
+extends
CommentRepository
Comment Model
-content: String
-created: Timestamp
-commentorName: String
-commentorProfile: String
AccountRepository
+findByUsername(String username)
+findByProfile(String profile)
+findByProfile(String name)
MessageRepository
+findByAuthorProfileIn(List profiles): List Messages
Message Model
-authorProfile: String
-authorName: String
-content: String
-created: Timestamp
-likes: Long
-comments: List<Comment>
Account Model
-name: String
-username: String
-password: String
-profile: String
-requests: List<Request>
-contacts: List<Account>
-givenVotes: List<Vote>
-givenLikes: List<PostVote>
-voted_skills: List<SkillVote>
-picture: byte
+creates
SignupController
+getAuthAccount()
+getAccountByUsername(String username): Account Model
+getAccountByProfile(String profile): Account Model
+getAccountByName(String name): Account Model
+createNewAccount(SignupForm signup)
+saveProfilePicture(Account account)
+deleteProfilePicture(Account account)
+creates mesaages
MessageService
+createNewMessage(Account author, String content): Message Model
+createNewComment(Account author, String content): Comment Model
+messageLikedByAccount(Message message, Account account): boolean
+getContactProfiles(Account account): List string
+likeMessage(Account account, Message message): void
+addMessage(Account author, String content): void
+addComment(Account author, String content, Message message): void
+getMessagaByID(Long message_id): Message Model
BoardController
-messageService: MessageService
-signupService: SignupController
+getBoard(): String
+addToBoard(): String
+addToMessage(): String
+likeToMessage(): String
+updates
+updates
SkillService
+createNewSkill(Account account)
+getDBSkill(Account account)
+addNewSkillVote(Account account)
+getAllUserSkills(Account account)
+addNewSkill(Account account)
SignUp
-SignupService: signupService
-SkillService: skillService
+routeProfile(): String
+getProfile(String profile): String
+getSignUp(): String
+signupUser(SignUpForm signupform)
+searchProfile(String profile): String
Profile
-signupService: SignupController
-skillService: SkillService
+getSettingsPage(String profile)
+savePicture()
+deletePicture(String profile): String
+addSkill(StringProfile)
+voteUpSkill(String profile)
Skill Model
-name: String
SkillVote Model
-talent: Account Model
-skillId: Long
-skillName: String
-upvotes: Long
SignUpForm Model
-name: String
-username: String
-password: String
-profile: String
SkillRepository
+findByName(String name): Skill Model
Vote Model
-voter: Account Model
-targetProfile: String
+targetSkill: Long
SkillVoteRepository
+findByTalentAndSkill(Account talent, Long skillId)
+extends
VoteRepository
+findByTargetSkillAndTargetProfileAndVoter(Long tragetSkill, String targetProfile, Account voter)
+extends
+extends
JPA Repository

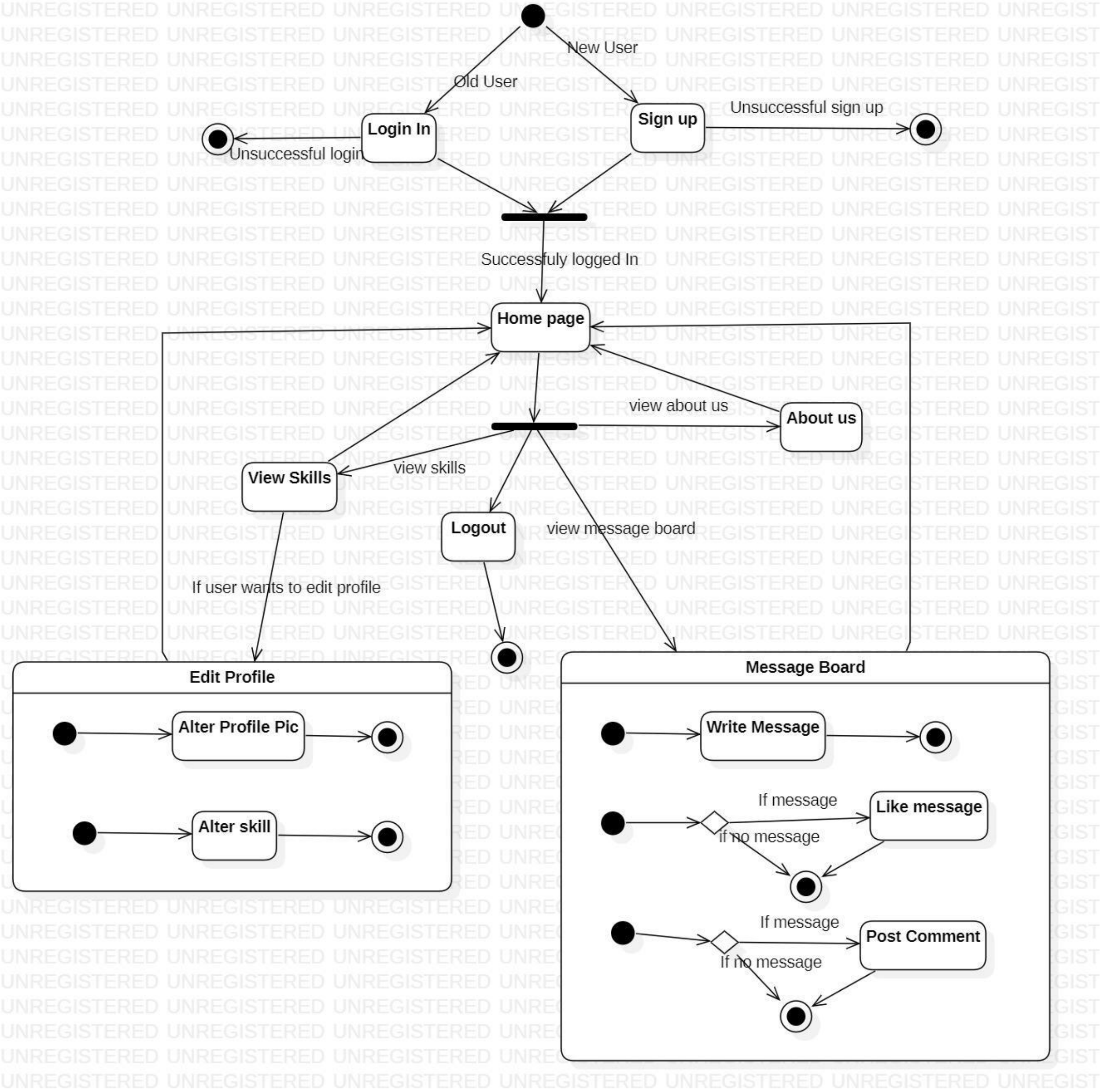
Use Case


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

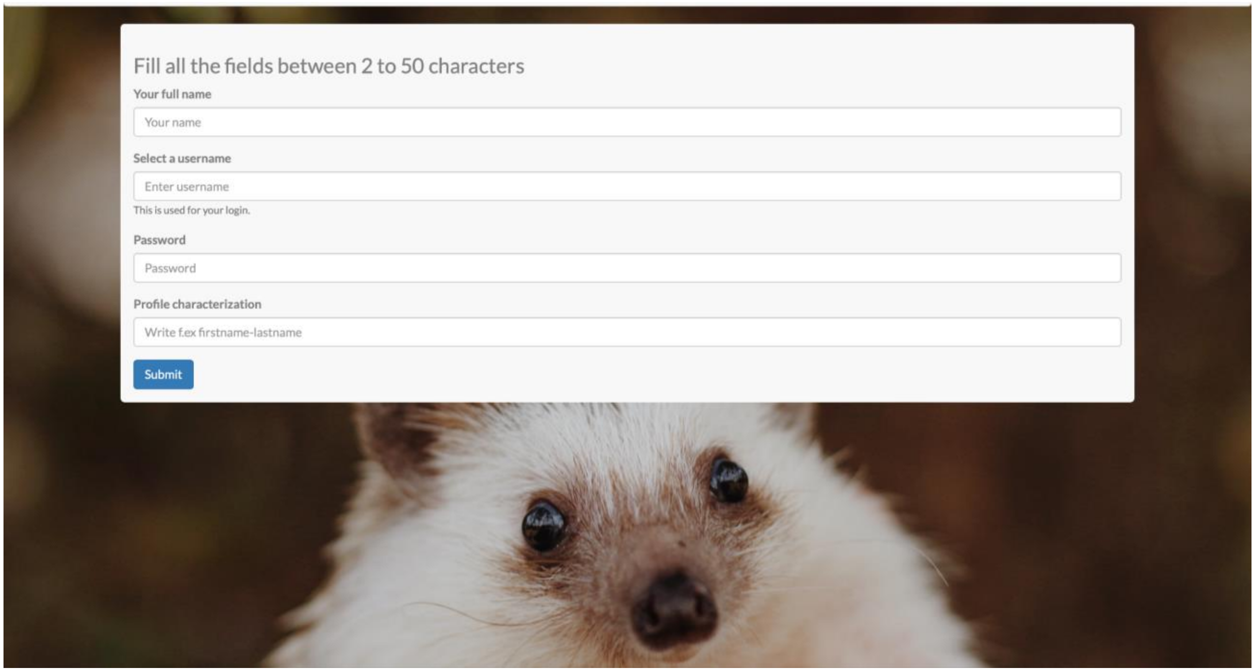
High Cohesion

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 (Githu link)
https://github.com/IshitaBharadwaj/OOAD_LinkedIn

OUTPUT

Signup page



Fill all the fields between 2 to 50 characters

Your full name

Your name

Select a username

Enter username

This is used for your login.

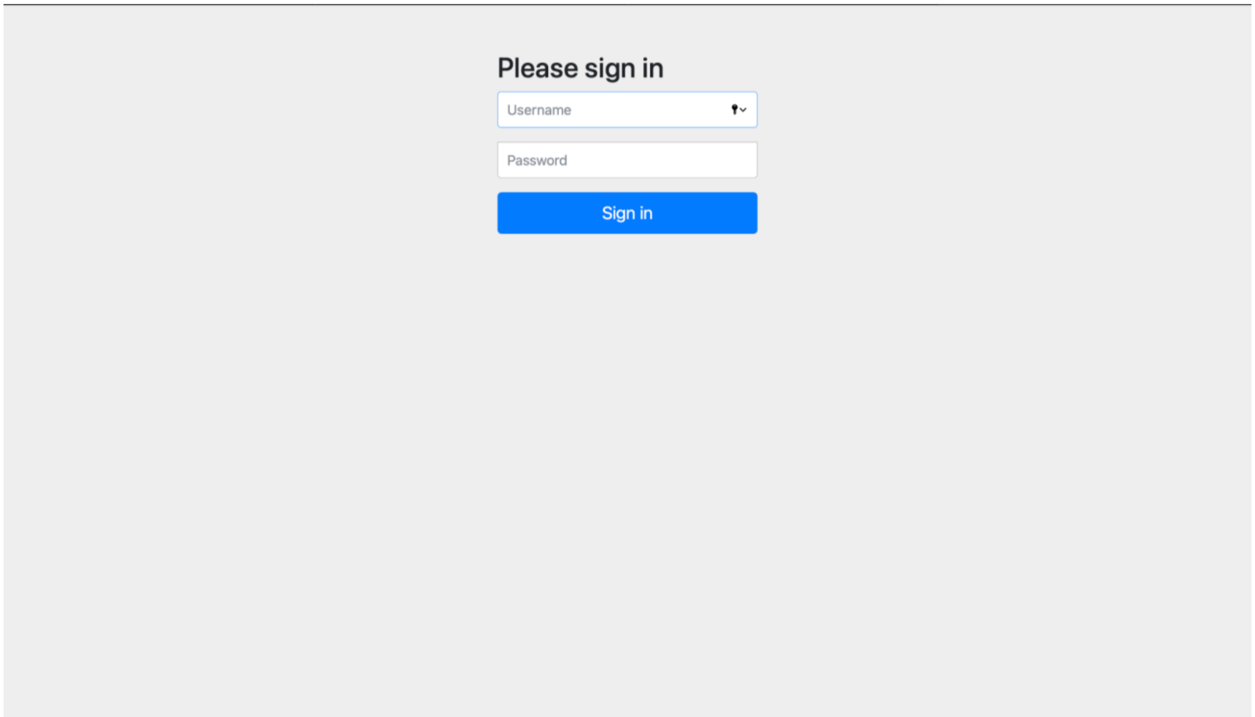
Password

Password

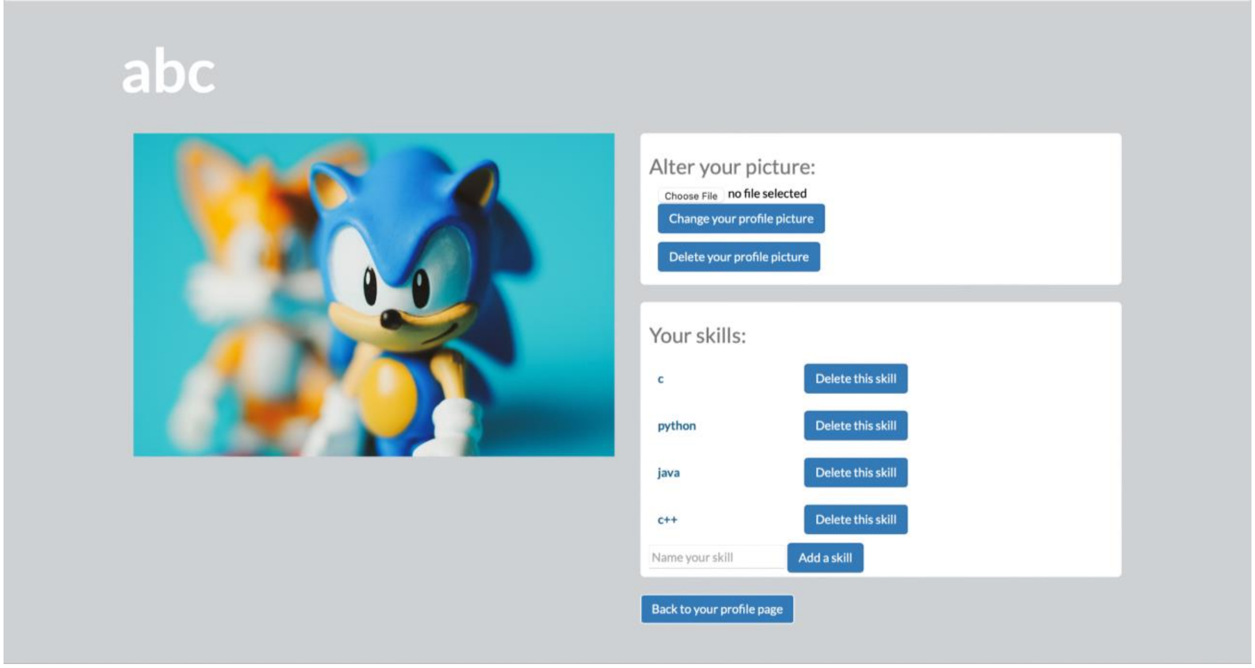
Profile characterization

Write f.ex firstname-lastname

Login Page

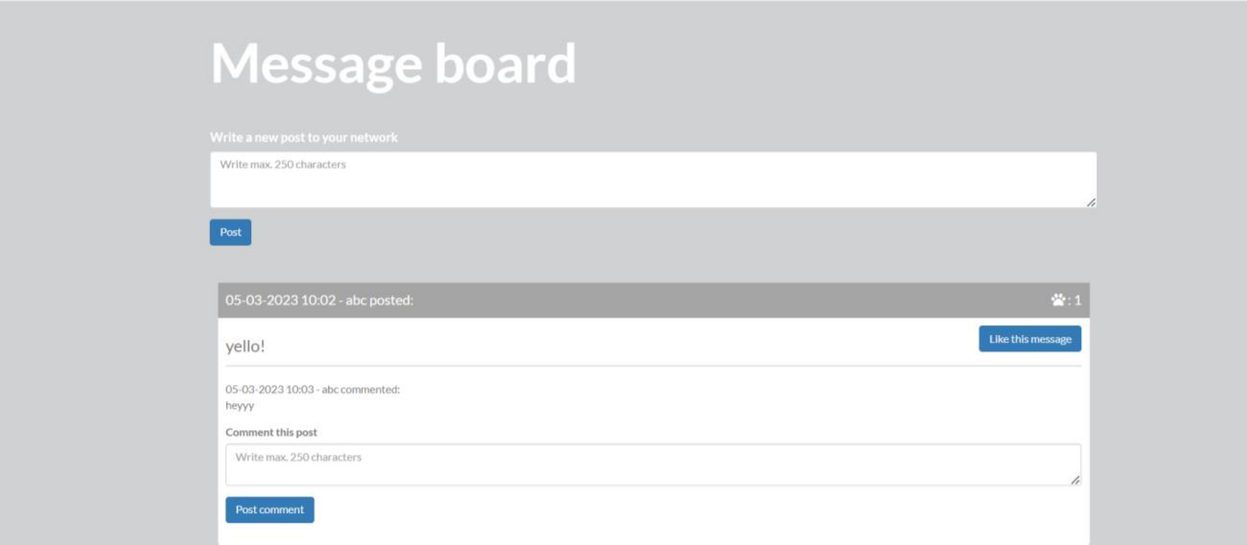


Profile



c
python
java
c++

Message Board



Message board

Write a new post to your network

Write max. 250 characters

Post

05-03-2023 10:02 - abc posted: : 1

yello!

Like this message

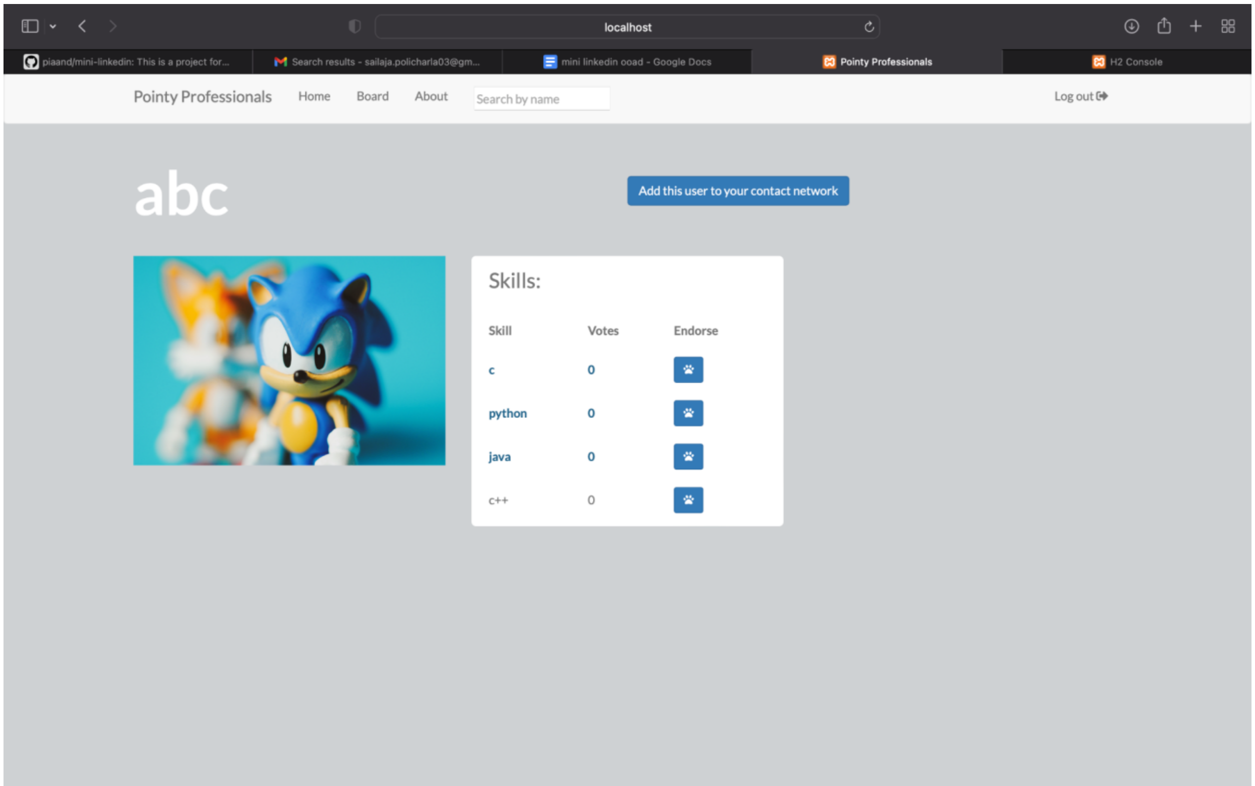
05-03-2023 10:03 - abc commented:
heyyy

Comment this post

Write max. 250 characters

Post comment

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



localhost
piaand/mini-linkedin: This is a project for...
Search results - sailaja.policharla03@gm...
mini linkedin ooad - Google Docs
Pointy Professionals
H2 Console
Pointy Professionals
Home
Board
About
Search by name
Log out

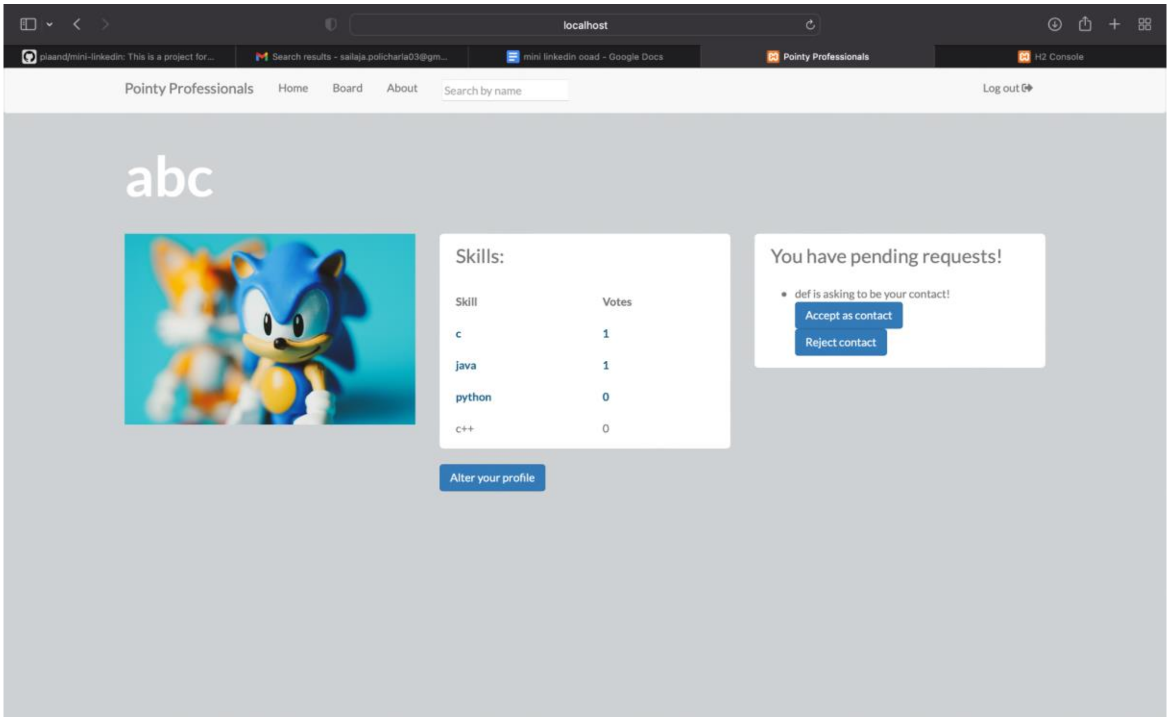
abc

Add this user to your contact network

Skills:

Skill	Votes	Endorse
c	0	
python	0	
java	0	
c++	0	

Pending Requests showing on user profile



abc

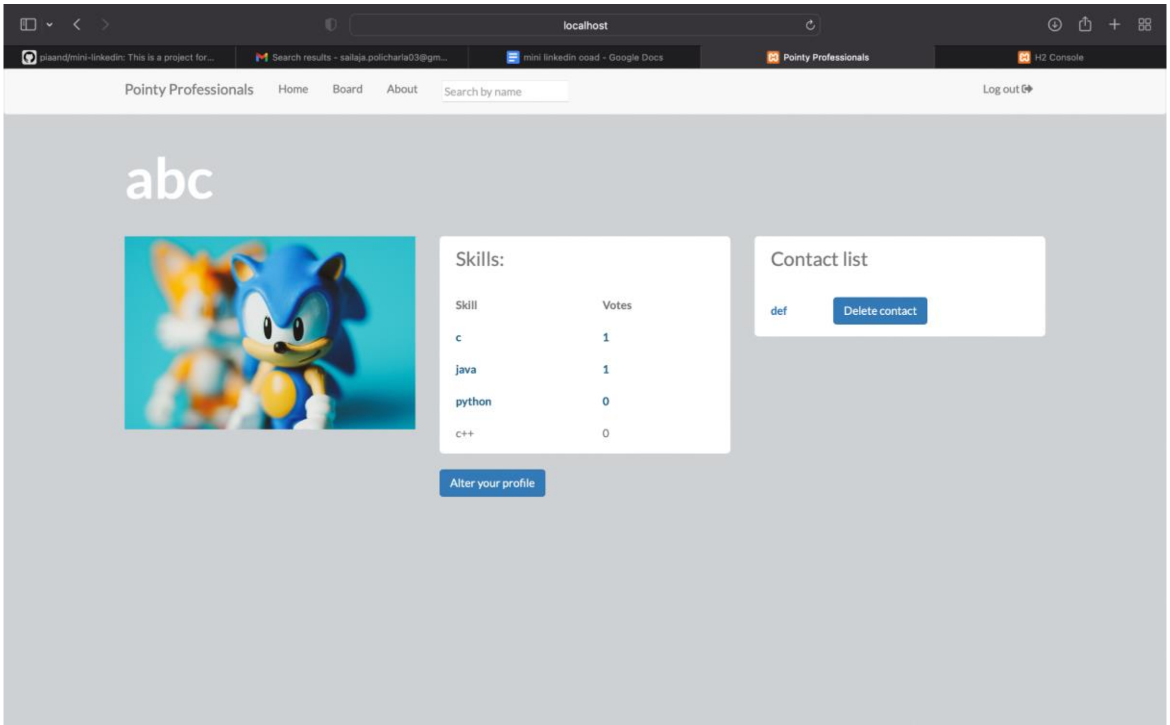
Skills:

Skill	Votes
c	1
java	1
python	0
c++	0

You have pending requests!

- def is asking to be your contact!

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



abc

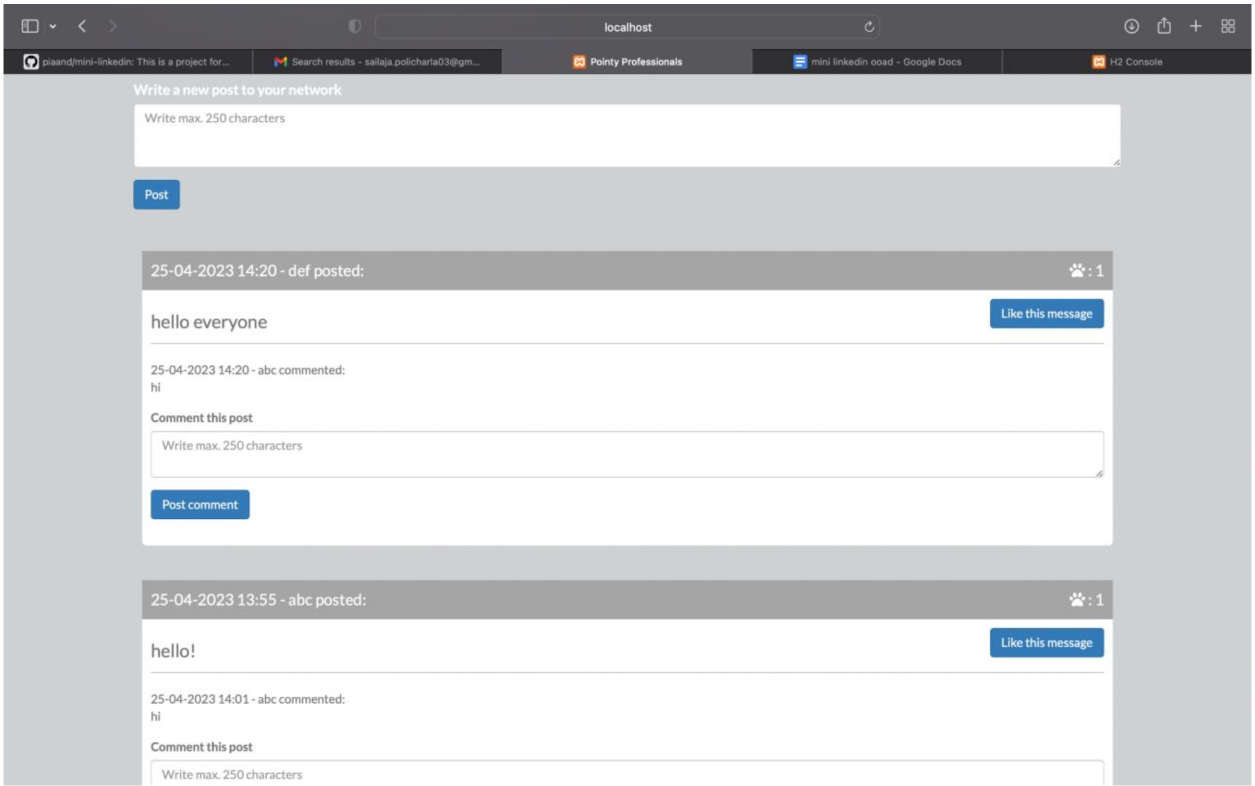
Skills:

Skill	Votes
c	1
java	1
python	0
c++	0

Contact list

def

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


localhost
piaand/mini-linkedin: This is a project for...
Search results - sailaja.policharla03@gm...
Pointy Professionals
mini linkedin ooad - Google Docs
H2 Console
Post
Like this message
Post comment
Like this message

Write a new post to your network

Write max. 250 characters

25-04-2023 14:20 - def posted: 🐾: 1

hello everyone

25-04-2023 14:20 - abc commented:
hi

Comment this post

Write max. 250 characters

25-04-2023 13:55 - abc posted: 🐾: 1

hello!

25-04-2023 14:01 - abc commented:
hi

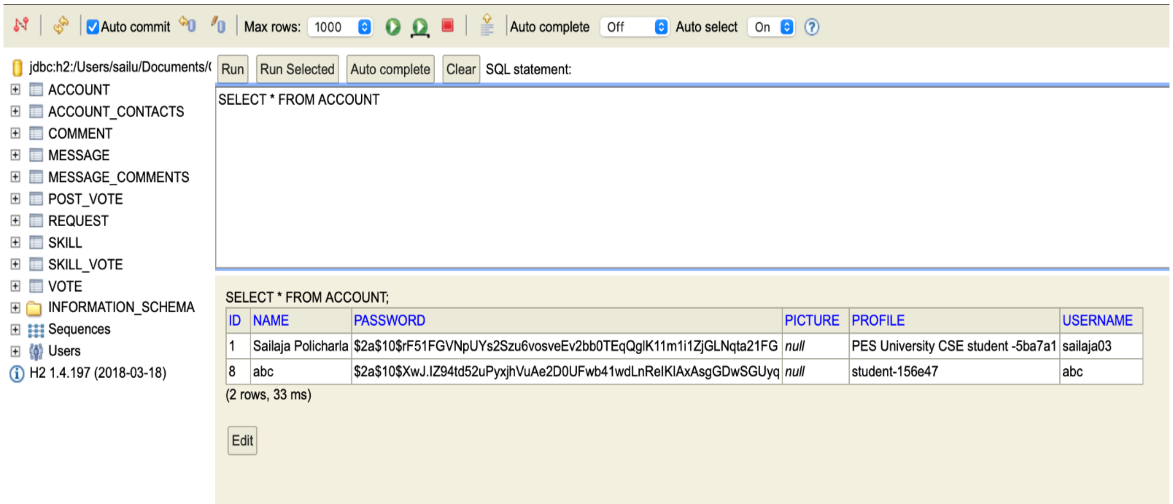
Comment this post

Write max. 250 characters

DATABASE

ACCOUNT

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



```
SELECT * FROM ACCOUNT
```

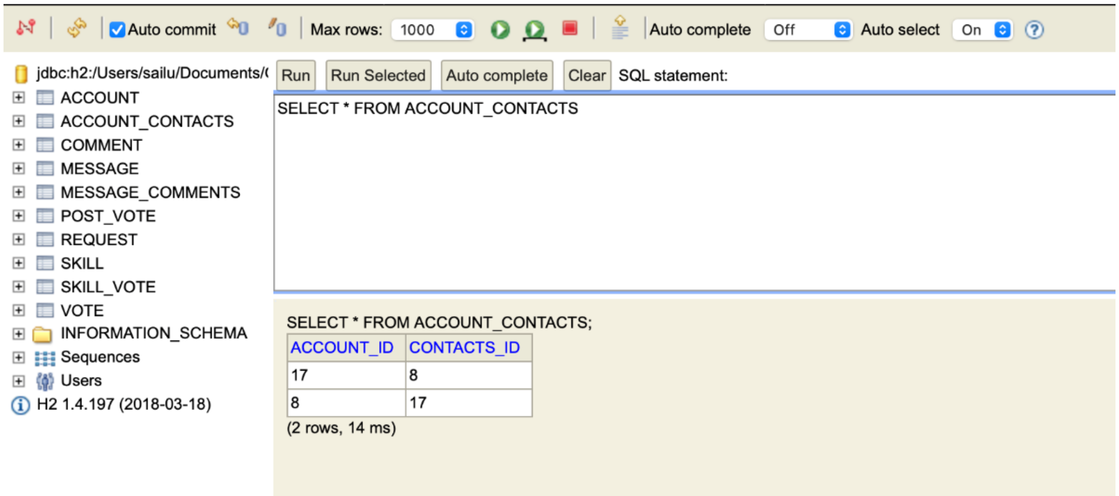
SELECT * FROM ACCOUNT;

ID	NAME	PASSWORD	PICTURE	PROFILE	USERNAME
1	Sailaja Policharla	$2a$10$rF51FGVNpUYs2Szu6vosveEv2bb0TEqQglK11m1i1ZjGLNqta21FG	*null*	PES University CSE student -5ba7a1	sailaja03
8	abc	$2a$10$XwJ.IZ94td52uPyxjhVuAe2D0UFwb41wdLnRelKIAxAsgGDwSGUyq	*null*	student-156e47	abc

(2 rows, 33 ms)

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



```
SELECT * FROM ACCOUNT_CONTACTS
```

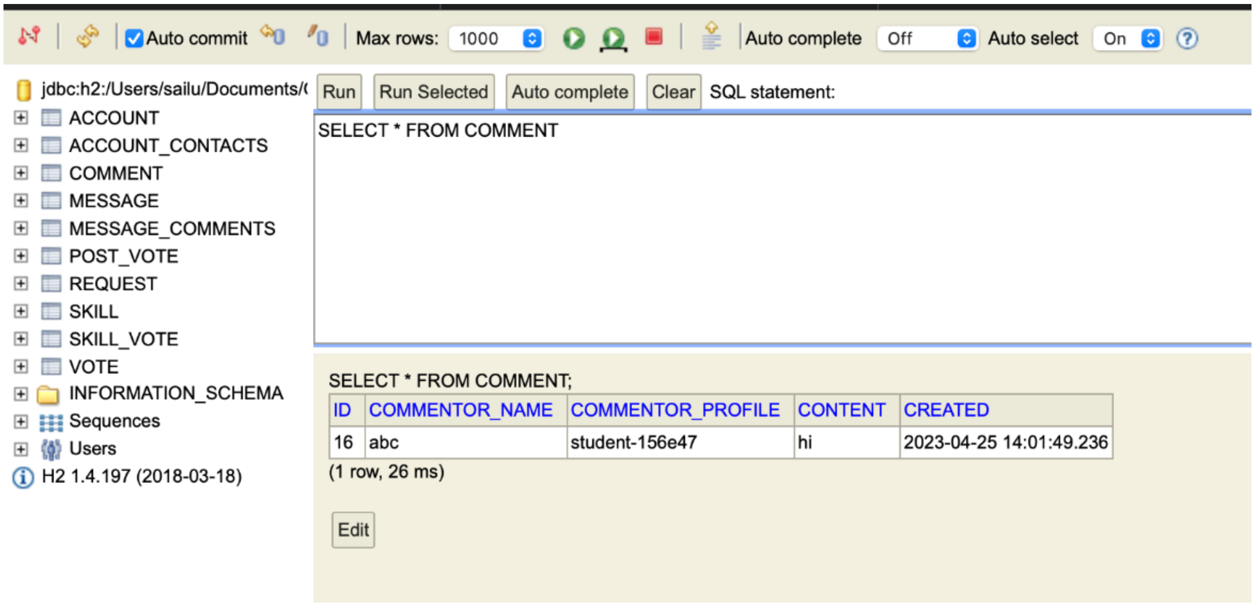
SELECT * FROM ACCOUNT_CONTACTS;

ACCOUNT_ID	CONTACTS_ID
17	8
8	17

(2 rows, 14 ms)

COMMENT

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



```
SELECT * FROM COMMENT
```

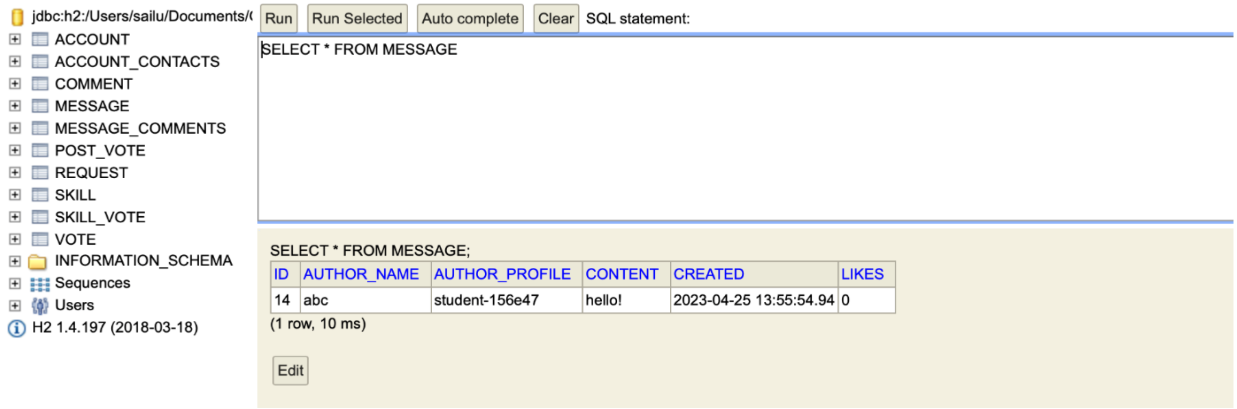
SELECT * FROM COMMENT;

ID	COMMENTOR_NAME	COMMENTOR_PROFILE	CONTENT	CREATED
16	abc	student-156e47	hi	2023-04-25 14:01:49.236

(1 row, 26 ms)

MESSAGE

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



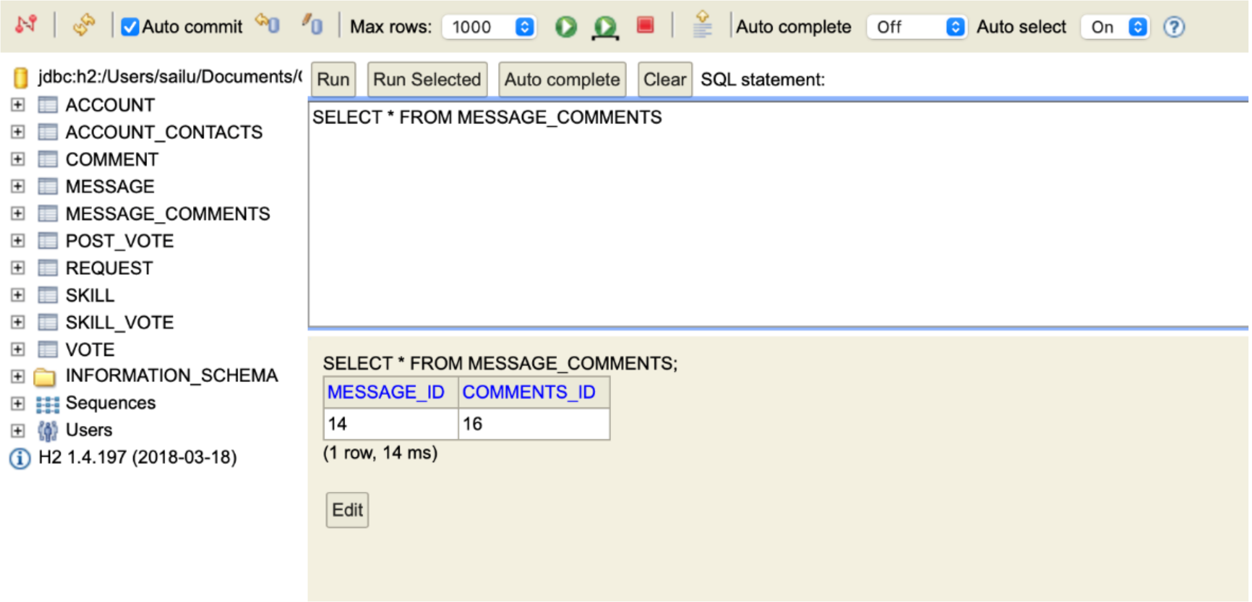
```
SELECT * FROM MESSAGE
```

SELECT * FROM MESSAGE;

ID	AUTHOR_NAME	AUTHOR_PROFILE	CONTENT	CREATED	LIKES
14	abc	student-156e47	hello!	2023-04-25 13:55:54.94	0

(1 row, 10 ms)

MESSAGE COMMENTS



☑ Auto commit

Max rows: 1000

Auto complete: Off

Auto select: On

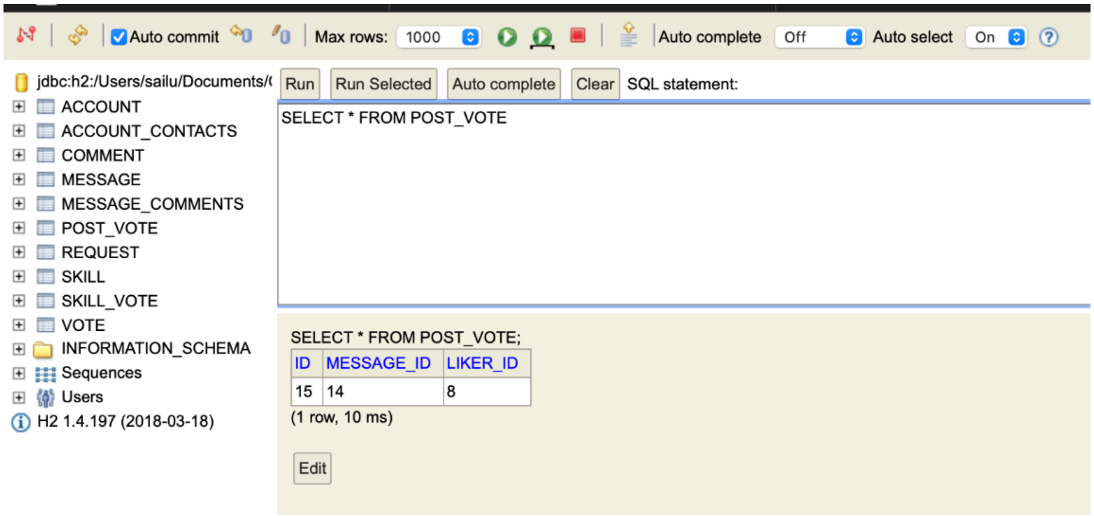
```
SELECT * FROM MESSAGE_COMMENTS
```

SELECT * FROM MESSAGE_COMMENTS;

MESSAGE_ID	COMMENTS_ID
14	16

(1 row, 14 ms)

POST_VOTE



☑ Auto commit

Max rows: 1000

Auto complete: Off

Auto select: On

```
SELECT * FROM POST_VOTE
```

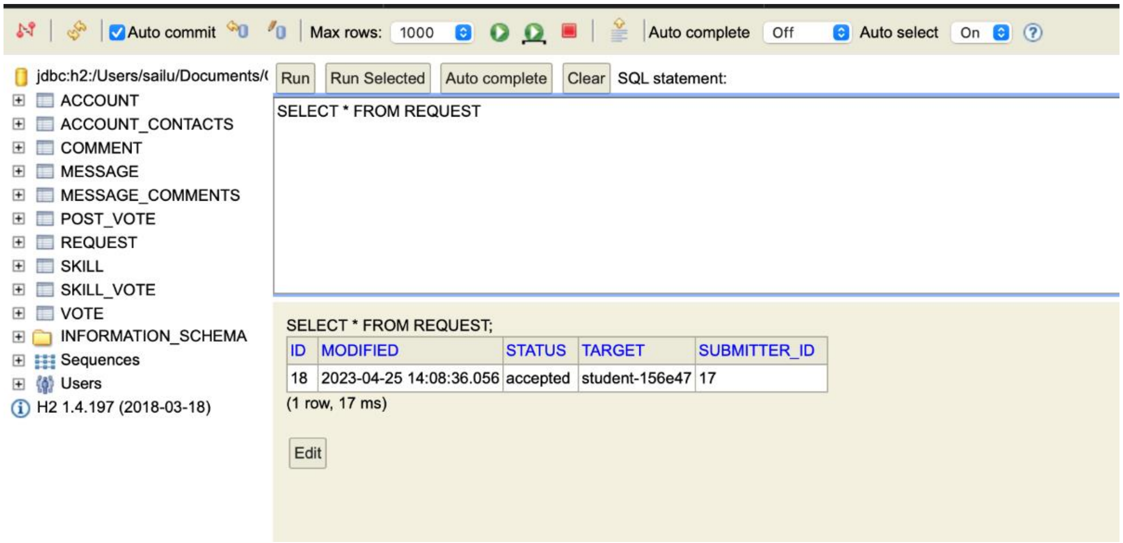
SELECT * FROM POST_VOTE;

ID	MESSAGE_ID	LIKER_ID
15	14	8

(1 row, 10 ms)

REQUEST

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



```
SELECT * FROM REQUEST
```

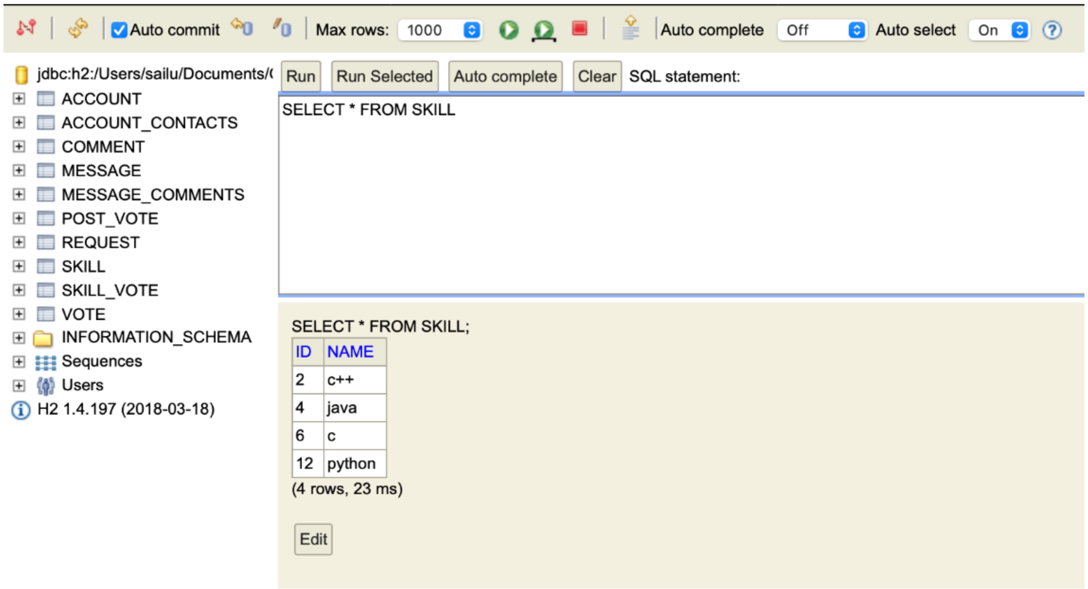
SELECT * FROM REQUEST;

ID	MODIFIED	STATUS	TARGET	SUBMITTER_ID
18	2023-04-25 14:08:36.056	accepted	student-156e47	17

(1 row, 17 ms)

SKILL

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



```
SELECT * FROM SKILL
```

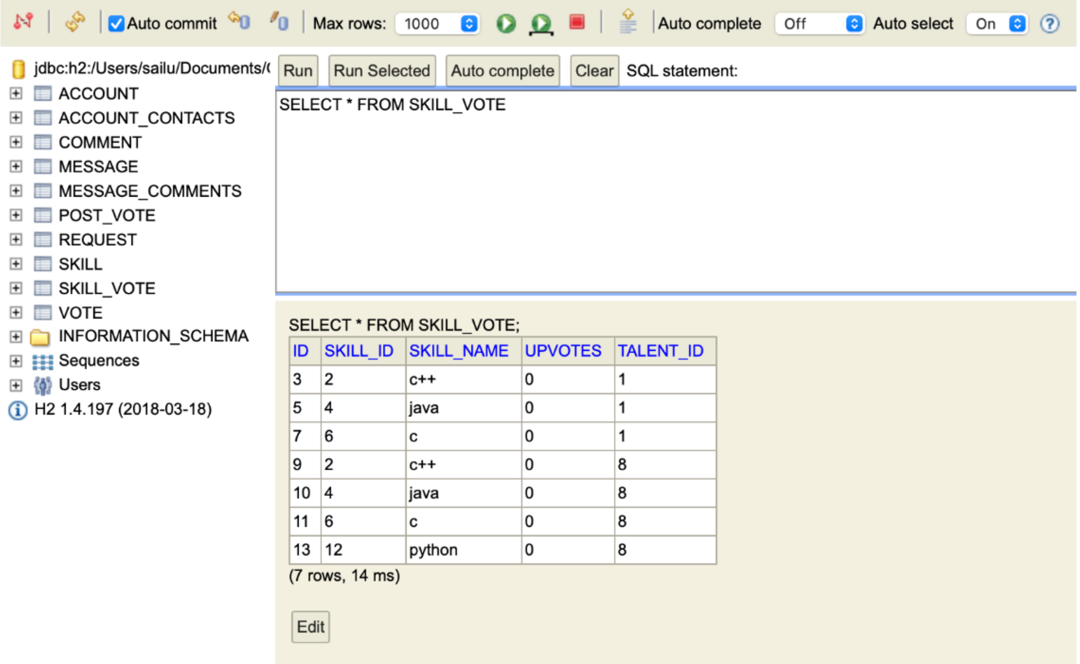
SELECT * FROM SKILL;

ID	NAME
2	c++
4	java
6	c
12	python

(4 rows, 23 ms)

SKILL VOTE

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



```
SELECT * FROM SKILL_VOTE
```

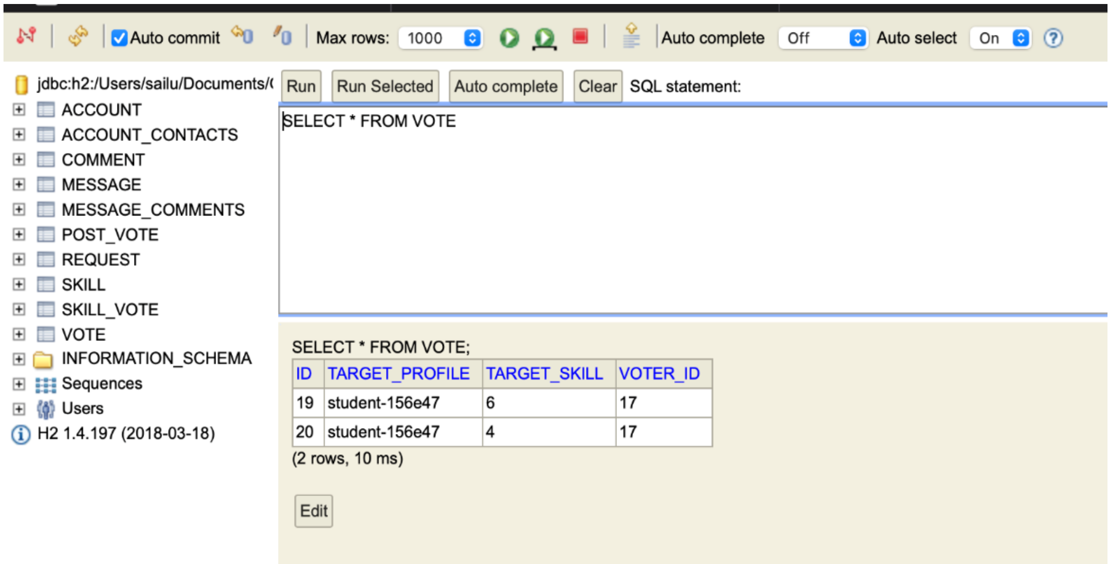
SELECT * FROM SKILL_VOTE;

ID	SKILL_ID	SKILL_NAME	UPVOTES	TALENT_ID
3	2	c++	0	1
5	4	java	0	1
7	6	c	0	1
9	2	c++	0	8
10	4	java	0	8
11	6	c	0	8
13	12	python	0	8

(7 rows, 14 ms)

VOTE

This shows the target profile and the skills that were voted



```
SELECT * FROM VOTE
```

SELECT * FROM VOTE;

ID	TARGET_PROFILE	TARGET_SKILL	VOTER_ID
19	student-156e47	6	17
20	student-156e47	4	17

(2 rows, 10 ms)