

UniConnect

Spring 2020 INFO6250 Project

Akshay Phapale, NUID: 001316563

Instructors

Prof. Yusuf Ozbek



Northeastern University
College of Engineering

Date: 04/21/2020

Introduction:

The main objective of this web portal is to provide the digital platform to freshmen to connect with students sharing the same area of interest. Students can search for other student's profiles on various search terms like the stream, job role, university, etc. Students can view the profile without forming a connection with other students. However, to send messages they need to have a connection with the other student. The profile consists of a students' professional experience, semester wise courses completed, internships/co-op, full-time job role, skillset, etc. Freshman can go through such sections from other student profiles, this will help them to develop their skillset and plan for the courses. This application consist of multiple roles and each role is associated with the set of functionalities.

Functionalities:

This application consists of students across all the universities and to manage that we need operations that can be managed by SuperUser. Afterward, each university consists of multiple courses and professors that are associated with the courses and to manage that we need operations that can be managed by UniversityAdmin. These two roles play an essential role in the development of the backend for the application. The User i.e. student has a set of functionalities that we will discuss later.

List of Functionalities:

- CRUD Universities with validations
- CRUD Courses and Professors with validations
- University email-based registration and authentication
- Password encryption using SHA-256
- Data association between multiple entities
- Searching users using various filters i.e. search by university name, aspired role, etc
- Accept, Reject, Remove, Send a connection request to users
- Messaging between the users using Websocket
- Course-professor mapping Request from student to university admin
- Implementation of Filters to implement Xss Validations
- Pagination of the search results

Technology Stack:

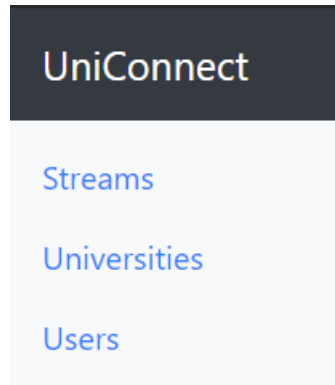
- Technology Stack:
- View/Presentation Layer - JSP
- Business Logic Layer - Spring MVC Java Controller using annotations and Hibernate
- Data Layer – MySQL

User roles and associated tasks:













There are three roles in this web application

1) SuperUser-





Superuser work CRUD activities of the following entities:

































Below is the UI of crud activities for the stream, University, and Users:

			Add Stream
#	Stream Name	Actions	
1	Computer Science	 	
2	Data Analytics	 	
3	Data Science	 	
4	Information Systems	 	
5	Machine Learning	 	
6	Telecommunication	 	
List of Streams			

This will to the validation on the bases of the course name.

				Add University
#	Name	Email Format	Actions	
1	Northeastern University	@northeastern.edu	 	
2	University Of Washington	@uw.edu	 	
List of Universities				

This will validate the university on the university name bases. Also, we take the email format of the university so that we will allow only students to end with university ID to log in.

#	Name	Email	Role	Actions
1	Neu Admin	su1@northeastern.edu	universityadmin	  
2	Su Uwash	su@uw.edu	universityadmin	  
3	Superuser	su@northeastern.edu	superuser	  
4	Superuser	su@uc.org	superuser	  
5	akshay-phapale-1	phapale.a@northeastern.edu	student	 
6	anagha-bhosale-5	bhosale.a@northeastern.edu	student	 
7	divya-girase-17	girase.di@northeastern.edu	student	 
8	divya-taneja-21	taneja.di@northeastern.edu	student	 
9	krishna-khamankar-12	krishak@uw.edu	student	 
10	mandar-bhamare-20	bhamare.m@northeastern.edu	student	 
11	mitali-majrekar-19	majrekar.m12@northeastern.edu	student	 
12	prasad-aroskar-11	aroskar.p@northeastern.edu	student	 
13	ruchit-urunkar-4	Urunkar.r@northeastern.edu	student	 

This is the third activity that can be accessed by the SuperUser. He can create perform CRUD operation on SuperUser and University Admin only. Student's accounts can be made inactive or deleted only.

2) University Admin

University admin work CRUD activities of the following entities:

UniConnect



















Courses

Professors

Student Requests

Below is the UI of crud operations of Courses and Professors

[Add Course](#)

#	Course Name	Course Code	Course Credit	Actions
1	Aed	INFO5100	4	 
2	Data Management And Database Design	INFO6210	4	 
3	Data Science	INFO6100	4	 
4	Machine Learning	INFO7777	4	 
5	Pdp	CS5010	4	 
6	Program Structure And Algorithms	INFO6205	4	 
7	Web Design And User Experience Engineering	INFO6150	4	 
8	Web Development Tools And Methods	INFO6250	4	 
9	Web Devlopement	CS5610	4	 

List of Course

Validation using create operation:

Course Name

Course 123

Course already exists with same name or course code

Course Code

INFO7777



















Course already exists with same name or course code

Course Credit

4

Submit

[Add Professor](#)

#	Professor Name	Professor Email	Actions
1	Amit Shesh	ashesh@ccs.neu.edu	 
2	Amuthan	amuthan@northeastern.edu	 
3	Handan Liu	h.liu@northeastern.edu	 
4	Khaled Bugrara	k.bugrara@northeastern.edu	 
5	Manuel D. Montrond	m.montrond@northeastern.edu	 
6	Nathaniel Tuck	ntuck@ccs.neu.edu	 
7	Robin Hillyard	r.hillyard@northeastern.edu	 
8	Vishal Chawla	chawla.v@northeastern.edu	 
9	Yusuf Ozbek	y.ozbek@northeastern.edu	 

List of Professors

Email field-based validation of professor:

Professor Name

Yusuf Oabek 2

Professor Email

y.ozbek@northeastern.edu

Professor with same email already exists

Select Courses

Aed
Data Science
Pdp
Web Developement

Submit

Mapping request from the student UI

UniConnect				Log Out
Courses	#	Request Message	Request From	Action
Professors	1	Add Course: Web Tools for Professor: ABC	Urunkar.r@northeastern.edu	✓ ✗
Student Requests	List of Requests			


User admin can accept or reject mapping request and the student will receive the **email** that the request has been accepted or rejected


3) Student

- Landing page

UniConnect

Search Message Akshay





Akshay Phapale
Experienced Software Developer | Information Systems Graduate Student
Northeastern University
Aspired Role - Data Science
Email - phapale.a@northeastern.edu
Contact - +1-857-333-8102
[Akshay Phapale](#)

Experience

Software Engineer Intern
Dell RSA
Currently work here . 0 years
Internship

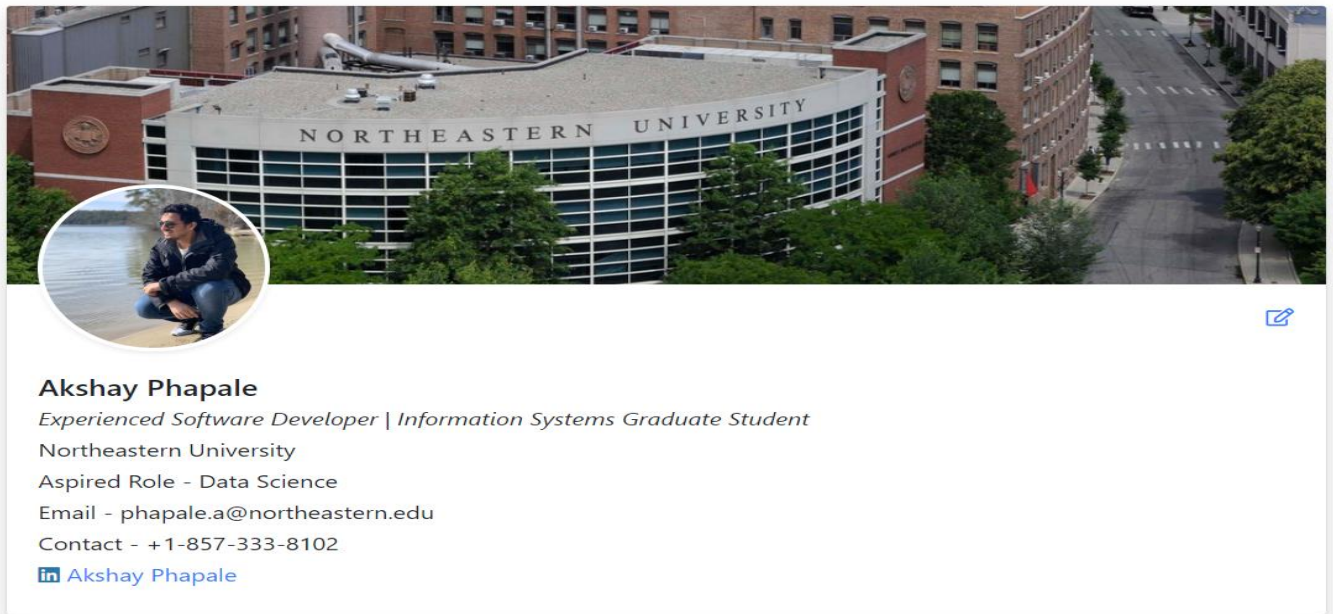
Co-Founder
SAIS Pvt. LTD.

Connection Requests

Anagha Bhosale ✓ ✗

The landing page has 4 sections let's see all 4 sections in detail:

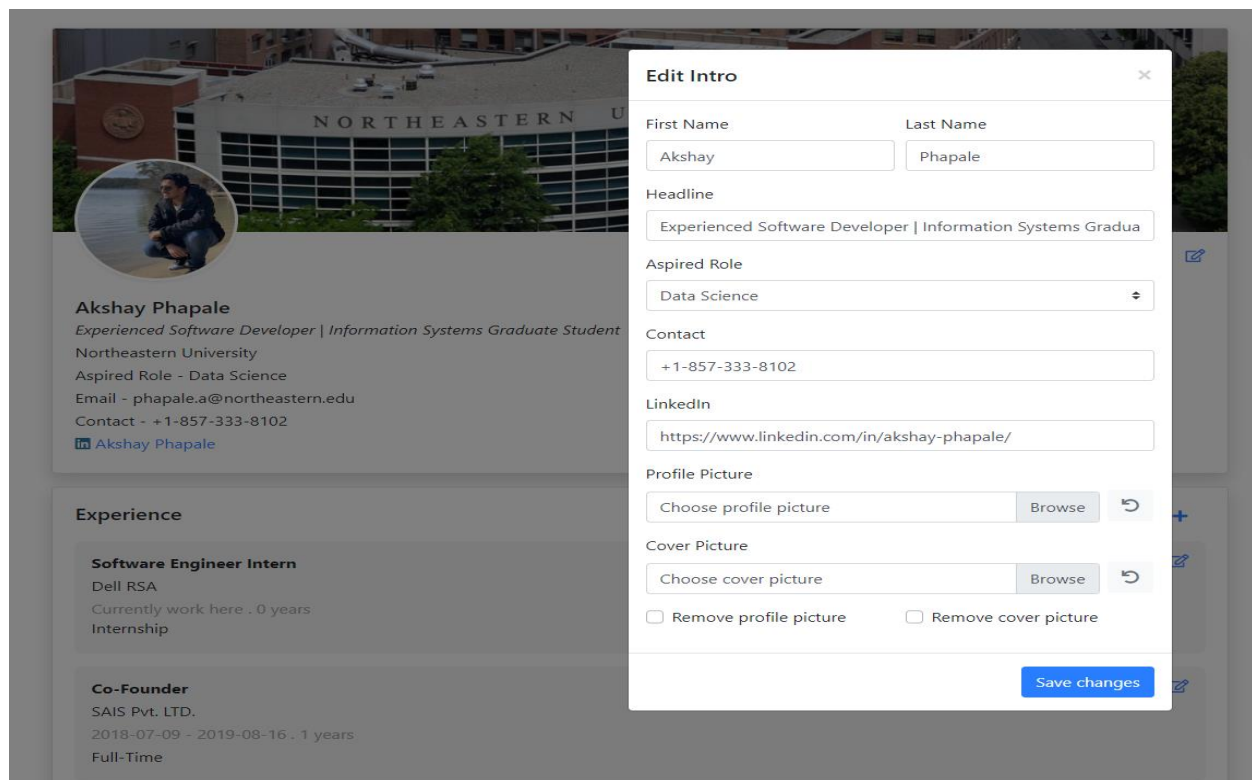
Intro Section



The profile card features a circular profile picture of a man crouching by a lake. The background is a large image of Northeastern University. The text on the card includes the user's name, title, university, and contact information. An edit icon is in the top right corner.

Akshay Phapale
Experienced Software Developer | Information Systems Graduate Student
Northeastern University
Aspired Role - Data Science
Email - phapale.a@northeastern.edu
Contact - +1-857-333-8102
[Akshay Phapale](#)

This part has basic information about the user. Users can edit the basic information using the edit button provided in the corner. The edit button displays the below form.



The 'Edit Intro' form is a modal window that allows users to update their profile information. It contains fields for First Name, Last Name, Headline, Aspired Role, Contact, and LinkedIn. It also includes options to upload or remove profile and cover pictures. A 'Save changes' button is at the bottom right.

Edit Intro

First Name: Akshay Last Name: Phapale

Headline: Experienced Software Developer | Information Systems Gradua

Aspired Role: Data Science

Contact: +1-857-333-8102

LinkedIn: https://www.linkedin.com/in/akshay-phapale/

Profile Picture: Choose profile picture Browse

Cover Picture: Choose cover picture Browse

☐ Remove profile picture ☐ Remove cover picture

[Save changes](#)

This form makes an ajax request to the controller method and information is updated without page refresh.

Experience Section

Experience

Software Engineer Intern
Dell RSA
Currently work here . 0 years
Internship

Co-Founder
SAIS Pvt. LTD.
2018-07-09 - 2019-08-16 . 1 years
Full-Time

Software Engineer
Ingram Micro SSC Pvt Ltd
2016-07-04 - 2018-07-06 . 2 years
Full-Time

Users can add experience or edit current experience. On click, the form will be shown in as pop up and This form makes an ajax request to the controller method and information is updated without page refresh

Experience

Software Engineer Intern
Dell RSA
Currently work here . 0 years
Internship

Co-Founder
SAIS Pvt. LTD.
2018-07-09 - 2019-08-16 . 1 years
Full-Time

Software Engineer
Ingram Micro SSC Pvt Ltd
2016-07-04 - 2018-07-06 . 2 years
Full-Time

Edit experience

Organization Name

Dell RSA

Designation

Software Engineer Intern

Experience Type

Internship

Start Date

04/14/2020

☒ Currently work here?

End Date


DELETE


Save changes

Enrollment


Aed
Course Code: INFO5100
Professor: Khaled Bugrara (k.bugrara@northeastern.edu)

Enrollment Section


Enrollment 

Aed 

Course Code: INFO5100
Professor: *Khaled Bugrara (k.bugrara@northeastern.edu)*
Grade: 5

Data Science 

Course Code: INFO6100
Professor: *Handan Liu (h.liu@northeastern.edu)*
Grade: 4

Web Development Tools And Methods 

Course Code: INFO6250
Professor: *Yusuf Ozbek (y.ozbek@northeastern.edu)*
Grade: 4

Users can add multiple enrollments also can edit enrollments. The form will work again in an async manner. Here, mapping of course and professor is fetched on a load of the document. Dynamically professor dropdown is created based on the selected course. Also, students can request the University admin for mapping of particular courses and professors.


Software Engineer
Ingram Micro SSC Pvt Ltd
2016-07-04 - 2018-07-06 . 2 years
Full-Time


Enrollment


Aed
Course Code: INFO5100
Professor: *Khaled Bugrara (k.bugrara@northeastern.edu)*
Grade: 5

Data Science
Course Code: INFO6100
Professor: *Handan Liu (h.liu@northeastern.edu)*
Grade: 4

Web Development Tools And Methods
Course Code: INFO6250
Professor: *Yusuf Ozbek (y.ozbek@northeastern.edu)*
Grade: 4

Edit enrollment 

Select Course
Aed 

Select Professor
Khaled Bugrara 

Grade
5

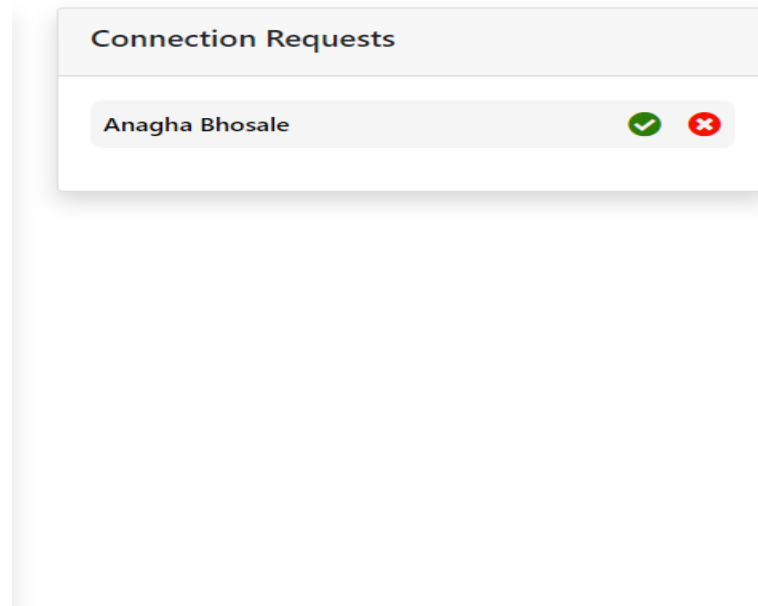
DELETE

Request new mapping

Save changes

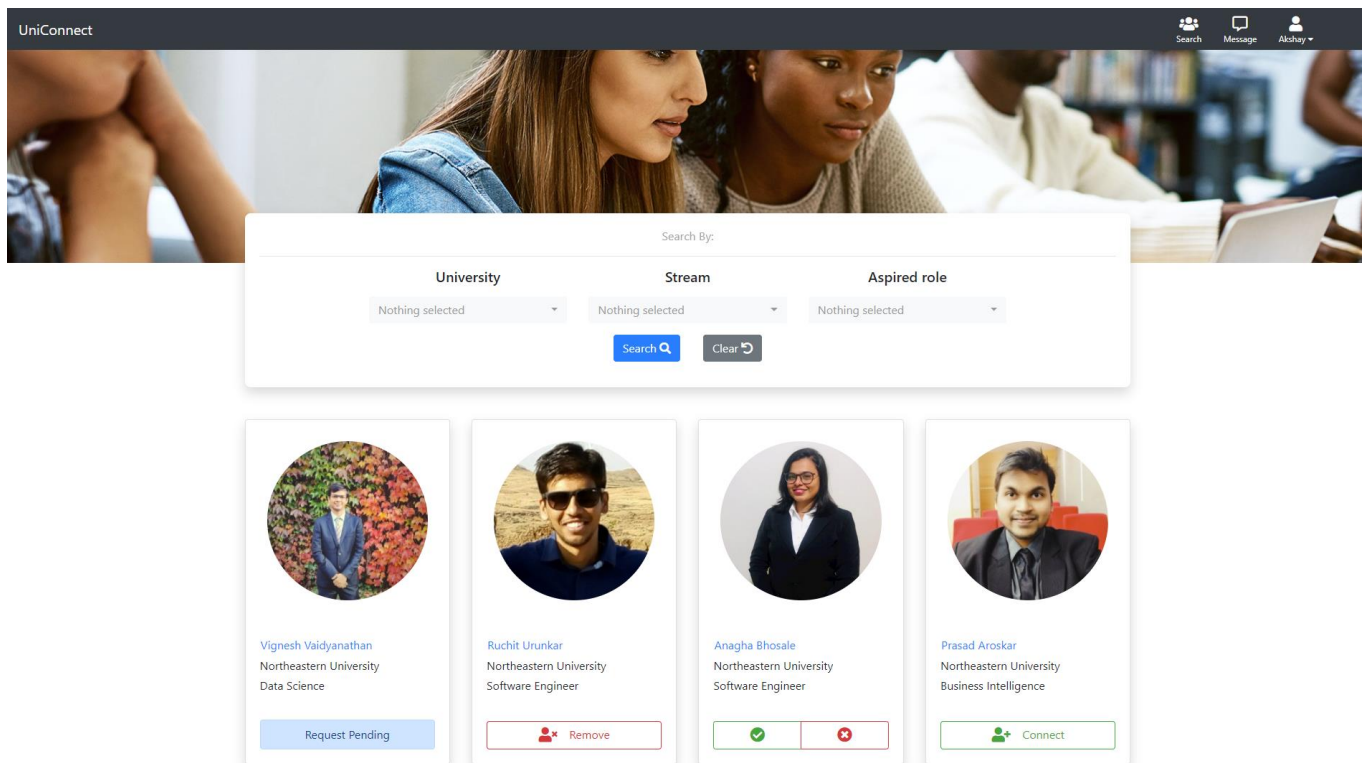
Connection Requests

This is the right section of the website where the user can see the pending requests. They can accept or reject the request.

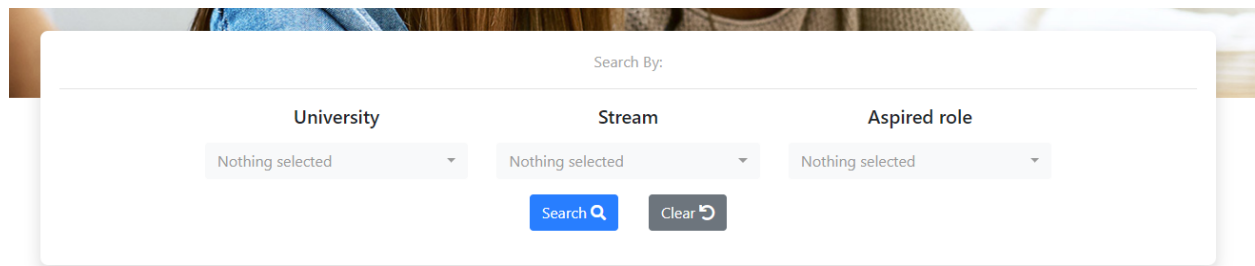


- Search

UI

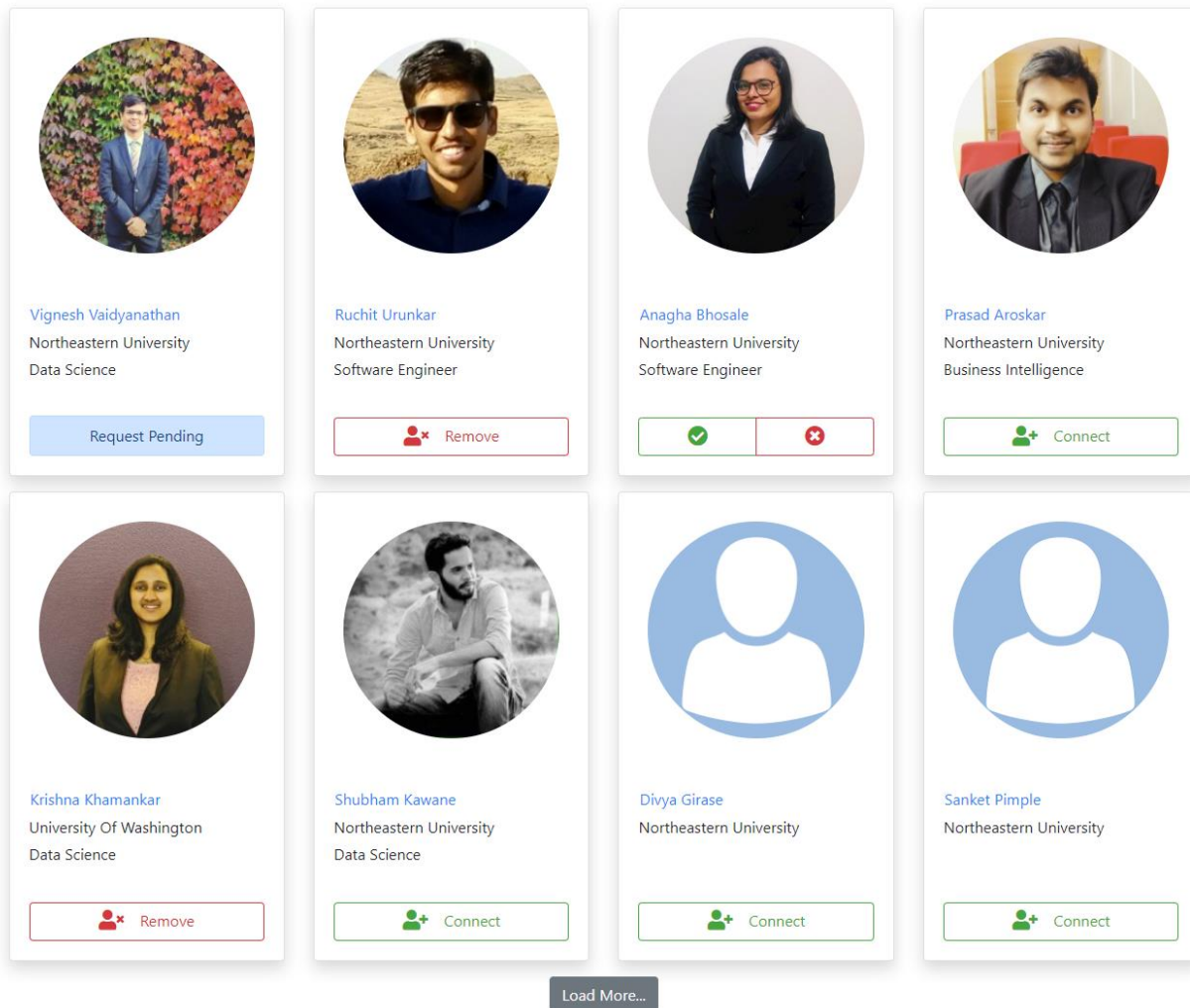


Here you can see, Users can do search operations on three filters.



A search filter interface with three dropdown menus labeled 'University', 'Stream', and 'Aspired role'. Each dropdown currently shows 'Nothing selected'. Below the dropdowns are two buttons: a blue 'Search' button with a magnifying glass icon and a grey 'Clear' button with a circular arrow icon.

Results are shown in below way



A grid of eight user profile cards arranged in two rows of four. Each card displays a circular profile picture, the user's name, their university, and their role. Below each card is an action button. The first card (Vignesh Vaidyanathan) has a 'Request Pending' button. The second card (Ruchit Urunkar) has a 'Remove' button. The third card (Anagha Bhosale) has 'Accept' and 'Reject' buttons. The fourth card (Prasad Aroskar) has a 'Connect' button. The fifth card (Krishna Khamankar) has a 'Remove' button. The sixth card (Shubham Kawane) has a 'Connect' button. The seventh card (Divya Girase) has a 'Connect' button. The eighth card (Sanket Pimple) has a 'Connect' button. A 'Load More...' button is located at the bottom center of the grid.

Name	University	Role	Action
Vignesh Vaidyanathan	Northeastern University	Data Science	Request Pending
Ruchit Urunkar	Northeastern University	Software Engineer	Remove
Anagha Bhosale	Northeastern University	Software Engineer	Accept, Reject
Prasad Aroskar	Northeastern University	Business Intelligence	Connect
Krishna Khamankar	University Of Washington	Data Science	Remove
Shubham Kawane	Northeastern University	Data Science	Connect
Divya Girase	Northeastern University		Connect
Sanket Pimple	Northeastern University		Connect

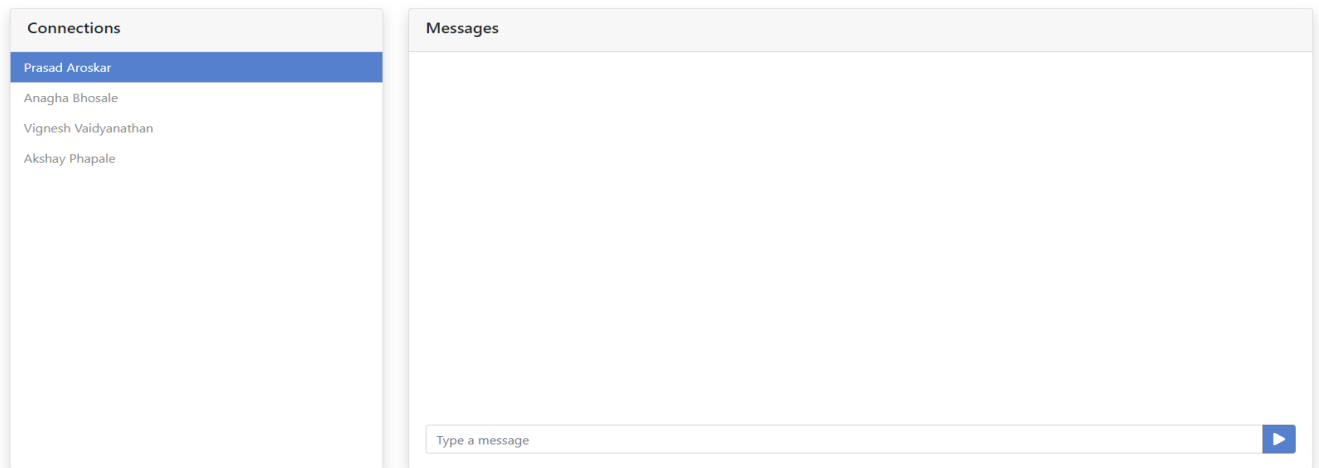
The load more button is working for serving the pagination. The pagination is loaded using an async call.

You can see buttons for **Accept, Reject, Remove, Send a connection request to users**

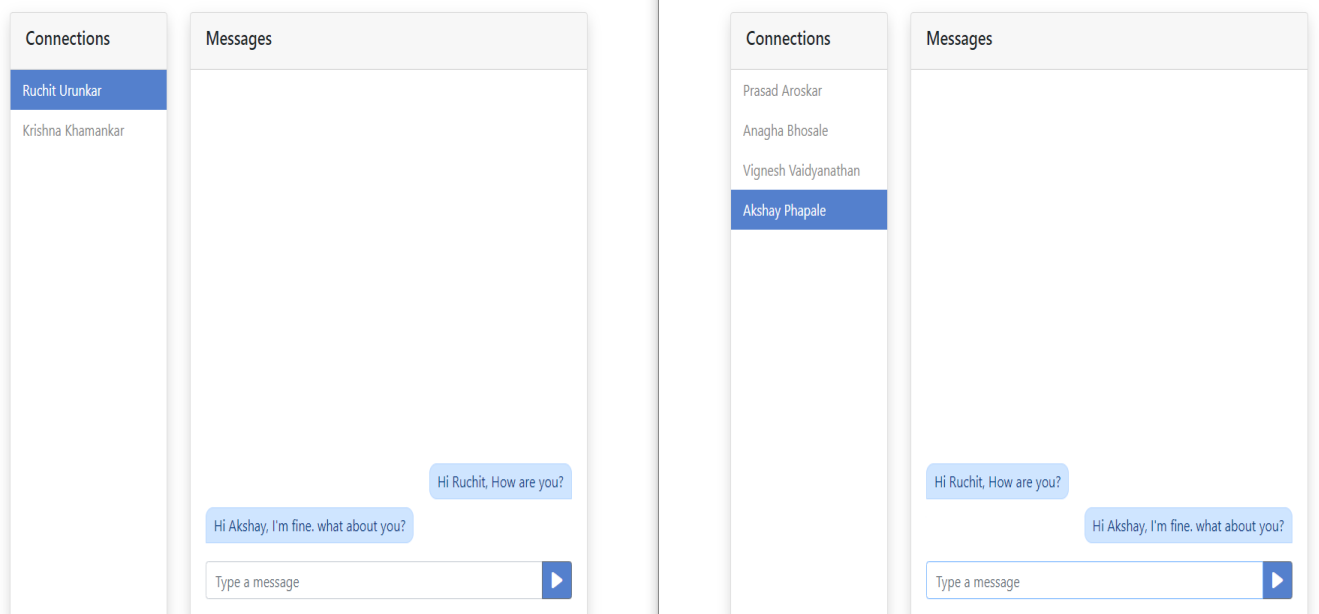
- Messaging

This is one of the challenging tasks to implement. I used a WebSocket to connect two sessions and transfer messages.

UI



Live chatting screenshot below



More Screenshots:

Encrypted Password in DB

```
1 • SELECT * FROM uniconnect.user;
```

ID	USERNAME	ENABLE	LOGIN_CODE	NAME	PASSWORD
1	su@northeastern.edu	1	NULL	Superuser	WkdjskWhWwFhZ9z+6w6nJWyy\$HnQPds3Zxd46pphCGxUWsk9qdFMx4dQCCQYaOEZMg200
2	su1@northeastern.edu	1	NULL	Neu Admin	DdkjuUhsQz1+WX5DCet7Rhvu\$nbJq7oV39HjLujrAGQyoS/1d4BGWCFtLGK3pE6jLo4M=
3	phapale.a@northeastern.edu	1	NULL	akshay-phapale-1	pe5y2MX+5/2JvzIe7bu0ON28\$XZaW0Y3EWLPkHd3A/sTUtjSpI83zFYbPDMhUtdX+KA8=
12	aroskar.p@northeastern.edu	1	NULL	prasad-aroskar-11	amVqED3Qct99Qt4AF6/V7ZDf\$+KVLpKDI3sW80kUCvaWbdOKbkzf+MyFfsGA6Dtmvloxk=
5	vaidyanathan.v@northeastern.edu	1	NULL	vignesh-vaidyanat...	tmbWHvcSYEiWHIezJGuFSR.\$Jb5VPxHGMkFCh7yXGEXia2LswRMOYpi+6BTan9EQtdM=
6	Urunkar.r@northeastern.edu	1	NULL	ruchit-urunkar-4	EddgCOFjNt16AzHMEKKe2d7\$PwDGmfHwbg4J9pMUnOxyco5Gr6L06OtWsvw5QTjZTSY=
7	bhosale.a@northeastern.edu	1	NULL	anagha-bhosale-5	QhaXEC7KVtyOk9P1E7wB5zmC\$++f2gfrTbERwbYA9zhRrLjJ5Zl6+TOeh+cIBX8pbGA=

University Email based Registration:

localhost:8080 says
University Email Address required: @northeastern.edu

Student Regi

OK

First Name
Akshay

Last Name
Phapale

University
Northeastern University

Stream
Information Systems

Email
phpale.a@gmail.com

Create

Controllers: All controllers follows REST request mapping

1) StreamController:

```
package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.IStreamService;
import com.uniconnect.model.Stream;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

@Controller
public class StreamController {
    @Autowired
    private IStreamService streamService;

    //fetch all
    @GetMapping("/admin/streams")
    public String get(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        List<Stream> list = streamService.get();
        Collections.sort(list, Comparator.comparing(Stream::getStreamName));
        model.addAttribute("list", list);
        return "admin/stream/index";
    }

    //create new
    @GetMapping("/admin/streams/new")
    public String add(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("stream", new Stream());
        return "admin/stream/new";
    }

    //create operation
    @PostMapping("/admin/streams")
    public String add(@Valid Stream stream,
        BindingResult result, Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
```

```

        return "redirect:/error";
    }
    model.addAttribute("duplicate","");
    if (result.hasErrors()) {
        return "admin/stream/new";
    }
    stream.setStreamName(StringFormatter.capitalizeWord(stream.getStreamName().trim()));

    if (invalid(stream.getStreamName())) {
        model.addAttribute("duplicate","Name already exists");
        return "admin/stream/new";
    }

    if (streamService.add(stream) == 1) {
        return "redirect:/admin/streams";
    } else {
        return "redirect:/error";
    }
}

//edit
@GetMapping("/admin/streams/{id}/edit")
public String edit(@PathVariable("id") int id, Model model, HttpServletRequest req) {
    User user = ActiveUser.IsActiveSuperuser(req);
    if(user==null){
        return "redirect:/error";
    }
    Stream stream = streamService.get(id);
    model.addAttribute("stream", stream);
    return "admin/stream/new";
}

//update
@PostMapping("/admin/streams/{id}")
public String edit(@PathVariable("id") int id, @Valid Stream stream, BindingResult result, Model model,
HttpServletRequest req) {
    User user = ActiveUser.IsActiveSuperuser(req);
    if(user==null){
        return "redirect:/error";
    }
    model.addAttribute("duplicate","");
    if (result.hasErrors()) {
        return "admin/stream/new";
    }

    stream.setStreamName(StringFormatter.capitalizeWord(stream.getStreamName().trim()));
    if(invalid(stream.getStreamName(),id)){
        model.addAttribute("duplicate","Name already exists");
        return "admin/stream/new";
    }
    if (streamService.update(id, stream) == 1) {
        return "redirect:/admin/streams";
    } else {
        return "redirect:/error";
    }
}

//Delete
@GetMapping("/admin/streams/{id}/delete")
public String delete(@PathVariable("id") int id, HttpServletRequest req) {

```



```

        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        if (streamService.delete(id) == 1) {
            return "redirect:/admin/streams";
        } else {
            return "redirect:/error";
        }
    }
}

//Error TODO: move to general mapping if needed
@GetMapping("/error")
public String error() {
    return "admin/stream/error";
}

private boolean invalid(String streamName) {
    return streamService.get().stream().anyMatch(x -> x.getStreamName().equals(streamName));
}

private boolean invalid(String streamName, int id) {
    return streamService.get().stream().anyMatch(x -> x.getStreamId() != id &&
x.getStreamName().equals(streamName));
}
}

```

2) UniversityController:

```

package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.IStreamService;
import com.uniconnect.iservice.IUniversityService;
import com.uniconnect.model.Stream;
import com.uniconnect.model.University;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.*;

@Controller
public class UniversityController {
    @Autowired
    private IUniversityService universityService;
    @Autowired
    private IStreamService streamService;

    //fetch all
    @GetMapping("/admin/universities")
    public String get(Model model, HttpServletRequest req) {

```

```

        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        List<University> lstUniversity = universityService.get();
        Collections.sort(lstUniversity, Comparator.comparing(University::getUniversityName));
        model.addAttribute("lstUniversity", lstUniversity);
        return "admin/university/index";
    }

    //create new
    @GetMapping("/admin/universities/new")
    public String add(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("streams", streamService.get());
        model.addAttribute("university", new University());
        return "admin/university/new";
    }

    @PostMapping("/admin/universities")
    public String add(@Valid University university, BindingResult result, HttpServletRequest req, Model
model) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("duplicate", "");
        if(result.hasErrors()){
            model.addAttribute("streams",streamService.get());
            return "admin/university/new";
        }
        university.setUniversityName(StringFormatter.capitalizeWord(university.getUniversityName()));
        if(invalid(university.getUniversityName())){
            model.addAttribute("streams",streamService.get());
            model.addAttribute("duplicate", "Name already exists");
            return "admin/university/new";
        }
        String[] checkedIds = req.getParameterValues("streams");
        Set<Stream> streams = new HashSet<>();
        for (String id : checkedIds) {
            streams.add(streamService.get(Integer.parseInt(id)));
        }
        university.setUniversityStreams(streams);
        if (universityService.add(university) == 1) {
            return "redirect:/admin/universities";
        } else {
            return "redirect:/error";
        }
    }

    //edit
    @GetMapping("/admin/universities/{id}/edit")
    public String edit(@PathVariable("id") int id, Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
    }

```

```

        University university = universityService.get(id);
        model.addAttribute("university", university);
        Set<Integer> streamId = new HashSet<>();
        university.getUniversityStreams().forEach(x->streamId.add(x.getStreamId()));
        model.addAttribute("streamId", streamId);
        model.addAttribute("streams", streamService.get());
        return "admin/university/new";
    }

    //Update
    @PostMapping("/admin/universities/{id}")
    public String edit(@PathVariable("id") int id, @Valid University university, BindingResult result,
        HttpServletRequest req, Model model){
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("duplicate", "");
        if(result.hasErrors()){
            model.addAttribute("university", university);
            model.addAttribute("streams", streamService.get());
            return "admin/university/new";
        }
        university.setUniversityName(StringFormatter.capitalizeWord(university.getUniversityName()));
        if(invalid(university.getUniversityName(), id)){
            model.addAttribute("university", university);
            model.addAttribute("streams", streamService.get());
            model.addAttribute("duplicate", "Name already exists");
            return "admin/university/new";
        }
        String[] checkedIds = req.getParameterValues("streams");
        Set<Stream> streams = new HashSet<>();
        for (String streamId : checkedIds) {
            streams.add(streamService.get(Integer.parseInt(streamId)));
        }
        university.setUniversityStreams(streams);
        if (universityService.update(id, university) == 1) {
            return "redirect:/admin/universities";
        } else {
            return "redirect:/error";
        }
    }

    //Delete
    @GetMapping("/admin/universities/{id}/delete")
    public String delete(@PathVariable("id") int id, HttpServletRequest req){
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        if (universityService.delete(id) == 1) {
            return "redirect:/admin/universities";
        } else {
            return "redirect:/error";
        }
    }

    private boolean invalid(String universityName) {
        return universityService.get().stream().anyMatch(x -> x.getUniversityName().equals(universityName));
    }
}

```

```

    private boolean invalid(String universityName, int id) {
        return universityService.get().stream().anyMatch(x -> x.getUniversityId() != id &&
x.getUniversityName().equals(universityName));
    }
}

```

3) UserController:

```

package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.Encryption;
import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.IStreamService;
import com.uniconnect.iservice.IUniversityService;
import com.uniconnect.iservice.IUserService;
import com.uniconnect.model.Stream;
import com.uniconnect.model.University;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

@Controller
public class UserController {
    @Autowired
    private IUserService userService;
    @Autowired
    private IUniversityService universityService;

    //fetch all
    @GetMapping("/admin/users")
    public String get(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
        List<User> list = userService.get();
        Collections.sort(list, Comparator.comparing(User::getName));
        model.addAttribute("list",list);
        return "admin/user/index";
    }

    //create new
    @GetMapping("/admin/users/new")
    public String add(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveSuperuser(req);
        if(user==null){
            return "redirect:/error";
        }
    }
}

```

```

    }
    model.addAttribute("user", new User());
    model.addAttribute("type", "");
    List<University> universities = universityService.get();
    model.addAttribute("universities", universities);
    return "admin/user/new";
}

//create operation
@PostMapping("/admin/users")
public String add(@Valid User user,
                 BindingResult result, Model model, HttpServletRequest req) {
    User activeUser = ActiveUser.IsActiveSuperuser(req);
    if(activeUser==null){
        return "redirect:/error";
    }
    model.addAttribute("duplicate", "");
    model.addAttribute("type", "");
    if (result.hasErrors()) {
        List<University> universities = universityService.get();
        model.addAttribute("universities", universities);
        return "admin/user/new";
    }
    user.setName(StringFormatter.capitalizeWord(user.getName().trim()));

    if (invalid(user.getEmail())) {
        List<University> universities = universityService.get();
        model.addAttribute("universities", universities);
        model.addAttribute("duplicate", "Email already exists");
        return "admin/user/new";
    }
    String role = req.getParameter("role");
    user.setRole(role);
    user.setPassword(Encryption.encryptPassphrase(user.getPassword()));
    if(role.equalsIgnoreCase("universityadmin")){
        Integer universityId = Integer.parseInt(req.getParameter("university"));
        user.setUniversityId(universityId);
    }
    user.setActive(true);
    if (userService.add(user) > 0) {
        return "redirect:/admin/users";
    } else {
        return "redirect:/error";
    }
}

//edit
@GetMapping("/admin/users/{id}/edit")
public String edit(@PathVariable("id") int id, Model model, HttpServletRequest req) {
    User activeUser = ActiveUser.IsActiveSuperuser(req);
    if(activeUser==null){
        return "redirect:/error";
    }
    User user = userService.get(id);
    model.addAttribute("user", user);
    model.addAttribute("type", "edit");
    List<University> universities = universityService.get();
    model.addAttribute("universities", universities);
    return "admin/user/new";
}

```

```

//update
@PostMapping("/admin/users/{id}")
public String edit(@PathVariable("id") int id, @Valid User user, BindingResult result, Model model,
HttpServletRequest req) {
    User activeUser = ActiveUser.IsActiveSuperuser(req);
    if(activeUser==null){
        return "redirect:/error";
    }
    model.addAttribute("duplicate","");
    model.addAttribute("type","edit");
    if (result.hasErrors()) {
        model.addAttribute("universities", universityService.get());
        return "admin/user/new";
    }

    user.setName(StringFormatter.capitalizeWord(user.getName().trim()));
    if(invalid(user.getEmail(),id)){
        List<University> universities = universityService.get();
        model.addAttribute("universities", universities);
        model.addAttribute("duplicate","Email already exists");
        return "admin/user/new";
    }
    String role = req.getParameter("role");
    user.setRole(role);
    if(role.equalsIgnoreCase("universityadmin")){
        Integer universityId = Integer.parseInt(req.getParameter("university"));
        user.setUniversityId(universityId);
    }
    if (userService.update(id, user) == 1) {
        return "redirect:/admin/users";
    } else {
        return "redirect:/error";
    }
}

//Delete
@GetMapping("/admin/users/{id}/delete")
public String delete(@PathVariable("id") int id, HttpServletRequest req) {
    User user = ActiveUser.IsActiveSuperuser(req);
    if(user==null){
        return "redirect:/error";
    }
    if (userService.delete(id) == 1) {
        return "redirect:/admin/users";
    } else {
        return "redirect:/error";
    }
}

//active inactive
@GetMapping("/admin/users/{id}/active")
public String active(@PathVariable("id") int id, HttpServletRequest req) {
    User user = ActiveUser.IsActiveSuperuser(req);
    if(user==null){
        return "redirect:/error";
    }

    if (userService.active(id) == 1) {
        return "redirect:/admin/users";
    } else {
        return "redirect:/error";
    }
}

```

```

    }
}

private boolean invalid(String email) {
    return userService.get().stream().anyMatch(x -> x.getEmail().equalsIgnoreCase(email));
}

private boolean invalid(String email, int id) {
    return userService.get().stream().anyMatch(x -> x.getUserId() != id &&
x.getEmail().equalsIgnoreCase(email));
}
}

```

4) StudentController

```

package com.uniconnect.controller;

import com.sun.org.apache.xpath.internal.operations.Mod;
import com.uniconnect.common.*;
import com.uniconnect.iservice.*;
import com.uniconnect.model.*;
import com.uniconnect.model.RequestMapping;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.LocalDate;
import java.time.YearMonth;
import java.time.temporal.ChronoUnit;
import java.util.*;

@Controller
public class StudentController {
    @Autowired
    IUniversityService universityService;

    @Autowired
    IUserService userService;

    @Autowired
    IStudentService studentService;

    @Autowired
    IStreamService streamService;
}

```

```

@Autowired
ICourseService courseService;

@Autowired
IProfessorService professorService;

@GetMapping("/uc/{id}")
public String homePage(@PathVariable("id") int id, ModelMap model, HttpServletRequest req) {
    User user = ActiveUser.isActiveStudent(req);
    if (user == null) {
        return "redirect:/login";
    }
    Student student = studentService.getId();
    TreeSet<Enrollment> setOfEnrollment = new TreeSet<>(student.getEnrollments());
    TreeSet<Experience> setOfExperience = new TreeSet<>(student.getExperiences());
    model.addAttribute("student", student);
    model.addAttribute("setOfEnrollment", setOfEnrollment);
    model.addAttribute("setOfExperience", setOfExperience);
    model.addAttribute("isEditable", id == user.getStudentId());
    List<Student> lstStudent = new ArrayList<>();
    lstStudent.add(student);
    List<Friend> friends = studentService.getFriendsAmongList(user.getStudentId(), lstStudent);
    Map<Integer, Integer> friendsMap = new HashMap<>();
    for (Student s : lstStudent) {
        Friend fs = friends.stream().filter(x -> x.getSenderId().equals(s.getStudentId()) &&
x.getReceiverId().equals(user.getStudentId())).findFirst().orElse(null);
        Friend fr = friends.stream().filter(x -> x.getReceiverId().equals(s.getStudentId()) &&
x.getSenderId().equals(user.getStudentId())).findFirst().orElse(null);
        if (fs != null) {
            if (fs.getIsApproved() == 0) {
                friendsMap.put(s.getStudentId(), 1); // pending req from our side
            }
            if (fs.getIsApproved() == 2) {
                friendsMap.put(s.getStudentId(), 2); // already friends
            }
        }
        if (fr != null) {
            if (fr.getIsApproved() == 0) {
                friendsMap.put(s.getStudentId(), 3); // pending req from Their side
            }
            if (fr.getIsApproved() == 2) {
                friendsMap.put(s.getStudentId(), 2); // already friends
            }
        }
    }
    model.addAttribute("friendsMap", friendsMap);
    List<Friend> pendingReq = studentService.getPendingReq(user.getStudentId());
    List<Student> requesters = new ArrayList<>();
    if (pendingReq.size() > 0) {
        for (Friend f : pendingReq) {
            requesters.add(studentService.get(f.getSenderId()));
        }
    }
    model.addAttribute("requesters", requesters);
    return "student/index";
}

@PostMapping("/uc/{id}/updateIntro")
public @ResponseBody
boolean updateIntro(@ModelAttribute IntroForm introForm, HttpServletRequest req) throws IOException
{

```



```

    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return false;
    }

    Student student = studentService.get(user.getStudentId());
    student.setFirstName(introForm.getFirstName());
    student.setLastName(introForm.getLastName());
    student.setHeadline(introForm.getHeadline());
    student.setRole(introForm.getAspiredRole());
    student.setContact(introForm.getContact());
    student.setLinkedIn(introForm.getLinkedIn());
    student.setOtherRole(introForm.getOtherRole());

    if (introForm.isRemoveProfilePic()) {
        student.setImageURL(null);
    } else if (introForm.getProfilePic() != null && introForm.getProfilePic().getSize() > 0) {
        String[] fileNameData = introForm.getProfilePic().getOriginalFilename().split("\\.");
        String filename = student.getId() + "_profile." + fileNameData[fileNameData.length - 1];
        String imageUrl =
"C:\\Users\\phapa\\Desktop\\Newfolder\\UniConnect\\Project\\src\\main\\ImageData\\" + filename;
        byte[] bytes = introForm.getProfilePic().getBytes();
        Files.write(Paths.get(imageUrl), bytes);
        student.setImageURL(filename);
    }

    if (introForm.isRemoveCoverPic()) {
        student.setCoverImageUrl(null);
    } else if (introForm.getCoverPic() != null && introForm.getCoverPic().getSize() > 0) {
        String[] fileNameData = introForm.getCoverPic().getOriginalFilename().split("\\.");
        String filename = student.getId() + "_cover." + fileNameData[fileNameData.length - 1];
        String imageUrl =
"C:\\Users\\phapa\\Desktop\\Newfolder\\UniConnect\\Project\\src\\main\\ImageData\\" + filename;
        byte[] bytes1 = introForm.getCoverPic().getBytes();
        Files.write(Paths.get(imageUrl), bytes1);
        student.setCoverImageUrl(filename);
    }

    if (studentService.update(student.getId(), student) == 1) {
        user.setName(student.getFirstName().toLowerCase() + "-" + student.getLastName().toLowerCase() + "-"
+ student.getId());
        userService.update(user.getUserId(), user);
        return true;
    }
    return false;
}

@ResponseBody
@PostMapping("/uc/{id}/updateExperience")
public boolean updateExp(@ModelAttribute ExperienceForm experience, HttpServletRequest req) {
    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return false;
    }
    if (experience != null) {
        Student student = studentService.get(user.getStudentId());
        if (experience.getExperienceId() == null) { //add exp
            Experience newExperience = new Experience();
            newExperience.setOrganizationName(experience.getOrganizationName());
            newExperience.setDesignation(experience.getDesignation());

```

```

        newExperience.setExpType(experience.getExpType());
        newExperience.setStartDate(experience.getStartDate());
        if(!experience.isPresentExp()){
            newExperience.setPresentExp(false);
            newExperience.setEndDate(experience.getEndDate());
        }
        newExperience.setYears(Year.getDiffYears(newExperience.getStartDate(),newExperience.getEndDate()));
    }
    else{
        newExperience.setPresentExp(true);
        newExperience.setEndDate(null);
        newExperience.setYears(Year.getDiffYears(newExperience.getStartDate(),new Date()));
    }
    newExperience.setStudent(student);
    if (studentService.addExperience(newExperience) != 0) {
        student.getExperiences().add(newExperience);
        return true;
    }
} else {
    Experience exp = student.getExperiences().stream().filter(x -> x.getExperienceId() ==
experience.getExperienceId()).findFirst().orElse(null);
    if (exp != null) {
        exp.setOrganizationName(experience.getOrganizationName());
        exp.setDesignation(experience.getDesignation());
        exp.setExpType(experience.getExpType());
        exp.setStartDate(experience.getStartDate());
        if(!experience.isPresentExp()){
            exp.setPresentExp(false);
            exp.setEndDate(experience.getEndDate());
            exp.setYears(Year.getDiffYears(exp.getStartDate(),exp.getEndDate()));
        }
        else {
            exp.setPresentExp(true);
            exp.setEndDate(null);
            exp.setYears(Year.getDiffYears(exp.getStartDate(),new Date()));
        }
        if (studentService.updateExperience(exp) != 0) {
            return true;
        }
    }
}
}
return false;
}

@ResponseBody
@PostMapping("/uc/{id}/deleteExperience")
public boolean deleteExp(@RequestParam String expId, HttpServletRequest req) {
    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return false;
    }
    if (expId != null) {
        Student student = studentService.get(user.getStudentId());
        Experience exp = student.getExperiences().stream().filter(x -> x.getExperienceId() ==
Integer.parseInt(expId)).findFirst().orElse(null);
        if (exp != null) {
            if (studentService.deleteExperience(exp) != 0) {
                return true;
            }
        }
    }
}

```

```

    }
    return false;
}

@ResponseBody
@PostMapping("/uc/{id}/updateEnrollment")
public boolean updateEnrollment(@ModelAttribute EnrollmentForm enrollment, HttpServletRequest req)
{
    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return false;
    }
    if (enrollment != null) {
        Student student = studentService.get(user.getStudentId());
        int universityId = student.getUniversity().getUniversityId();
        if (enrollment.getEnrollmentId() == null) {
            Enrollment newEnrollment = new Enrollment();
            newEnrollment.setCourse(courseService.get(universityId, enrollment.getSelectedCourse()));
            newEnrollment.setProfessor(professorService.get(universityId, enrollment.getSelectedProfessor()));
            newEnrollment.setGrade(enrollment.getGrade());
            newEnrollment.setStudent(student);
            if (studentService.addEnrollment(newEnrollment) != 0) {
                return true;
            }
        } else {
            Enrollment enr = student.getEnrollments().stream().filter(x -> x.getEnrollmentId() ==
enrollment.getEnrollmentId()).findFirst().orElse(null);
            if (enr != null) {
                enr.setCourse(courseService.get(universityId, enrollment.getSelectedCourse()));
                enr.setProfessor(professorService.get(universityId, enrollment.getSelectedProfessor()));
                enr.setGrade(enrollment.getGrade());
                if (studentService.updateEnrollment(enr) != 0) {
                    return true;
                }
            }
        }
    }
    return false;
}

@ResponseBody
@PostMapping("/uc/{id}/deleteEnrollment")
public boolean deleteEnr(@RequestParam String enrId, HttpServletRequest req) {
    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return false;
    }
    if (enrId != null) {
        Student student = studentService.get(user.getStudentId());
        Enrollment enr = student.getEnrollments().stream().filter(x -> x.getEnrollmentId() ==
Integer.parseInt(enrId)).findFirst().orElse(null);
        if (enr != null) {
            if (studentService.deleteEnrollment(enr) != 0) {
                return true;
            }
        }
    }
    return false;
}

@ResponseBody

```

```

    @GetMapping("uc/{id}/getCourseProfessorMapping")
    public Map<Map<Integer, String>, List<Map<Integer, String>>>
getCourseProfessorMapping(HttpServletRequest req) {
    Map<Map<Integer, String>, List<Map<Integer, String>>> data = new HashMap<>();
    User user = ActiveUser.IsActiveStudent(req);
    Student student = studentService.get(user.getId());
    University university = student.getUniversity();
    for (Course course : university.getUniversityCourses()) {
        if (course.getProfessors().size() > 0) {
            Map<Integer, String> courseMap = new HashMap<>();
            courseMap.put(course.getId(), course.getName());
            List<Map<Integer, String>> lstProfessor = new ArrayList<>();

            for (Professor prof : course.getProfessors()) {
                Map<Integer, String> professorMap = new HashMap<>();
                professorMap.put(prof.getId(), prof.getName());
                lstProfessor.add(professorMap);
            }
            data.put(courseMap, lstProfessor);
        }
    }
    return data;
}

@ResponseBody
@PostMapping("uc/{id}/requestMapping")
public boolean getRequestMapping(@RequestParam String course, @RequestParam String professor,
HttpServletRequest req) {
    User user = ActiveUser.IsActiveStudent(req);
    Student student = studentService.get(user.getId());
    User universityAdmin = userService.getByUniversityId(student.getUniversity().getId());
    if (universityAdmin != null) {
        RequestMapping rm = new RequestMapping();
        rm.setActive(true);
        rm.setMessage("Add Course: " + course + " for Professor: " + professor);
        rm.setUser(universityAdmin);
        rm.setStudentEmail(student.getEmail());
        universityAdmin.getRequests().add(rm);
        if (studentService.addRequestMapping(rm) != 0) {
            return true;
        }
    }
    return false;
}

//-----MESSAGING-----

//-----REGISTRATION-----
@GetMapping("/register")
public String registerStudent(HttpServletRequest req, Model model) {

    List<University> lstUniversity = universityService.get();
    HttpSession session = req.getSession();
    session.setAttribute("lstUnivesity", "HI");
    model.addAttribute("lstUniversity", lstUniversity);
    return "student/registration";
}

```

```

@PostMapping("/register")
public String createRegister(Model model, HttpServletRequest req) {

    String firstName = StringFormatter.capitalizeWord(req.getParameter("firstName")).trim();
    String lastName = StringFormatter.capitalizeWord(req.getParameter("lastName")).trim();
    String email = req.getParameter("email").trim();
    int univerisityId = Integer.parseInt(req.getParameter("univerisity"));
    int streamId = Integer.parseInt(req.getParameter("stream"));

    Student student = new Student();
    student.setFirstName(firstName);
    student.setLastName(lastName);
    student.setEmail(email);
    student.setUniversity(universityService.get(univerisityId));
    student.setStream(streamService.get(streamId));

    Integer studentId = studentService.add(student);

    String codeGenerator = CodeGenerator.getSaltString();

    User user = new User();
    user.setName(firstName.toLowerCase() + "-" + lastName.toLowerCase() + "-" + studentId);
    user.setEmail(email);
    user.setActive(false);
    user.setStudentId(studentId);
    user.setRole("student");
    user.setLoginCode(Encryption.encryptPassphrase(codeGenerator));
    int userId = userService.add(user);

    if (!Email.send("system.uniconnect@gmail.com", "Ingram@123", email, "Activation Code",
codeGenerator)) {
        return "redirect:/error";
    }
    HttpSession session = req.getSession(true);
    session.setAttribute("user", userService.get(userId));
    return "student/validation";
}

@PostMapping("/validation")
public String verify(Model model, HttpServletRequest req) {

    HttpSession session = req.getSession(true);
    model.addAttribute("error", "");
    User user = (User) session.getAttribute("user");
    if (user == null && user.getPassword() != null) {
        return "redirect:/error";
    }
    String code = req.getParameter("code");
    try {
        boolean result = Encryption.verifyPassphrase(code, user.getLoginCode());
        if (result == false) {
            model.addAttribute("error", "Code does not match");
            return "student/validation";
        }
    }
    user.setLoginCode(null);
    userService.registerStudent(user);
    session.setAttribute("user", user);
} catch (Exception ex) {
    model.addAttribute("error", "Code does not match");
    return "student/validation";
}

```

```

    }
    return "student/createPassword";
}

@PostMapping("/createStudent")
public String createPassword(Model model, HttpServletRequest req) {
    HttpSession session = req.getSession(true);
    User user = (User) session.getAttribute("user");
    if (user == null && user.getPassword() != null && user.getLoginCode() != null) {
        return "redirect:/error";
    }
    String password = req.getParameter("password");
    user.setPassword(Encryption.encryptPassphrase(password));
    user.setActive(true);
    Student s = studentService.getForRegister(user.getStudentId());
    s.setActive(true);
    studentService.update(user.getStudentId(), s);
    if (userService.registerStudent(user) != 1) {
        return "redirect:/error";
    }
    session.removeAttribute("user");
    return "redirect:/login";
}

@GetMapping("/register/universities")
public @ResponseBody
Map<Object, String> getUniversityStreams(@RequestParam Integer id) {
    Map<Object, String> map = new HashMap<>();
    University university = universityService.get(id);
    for (Stream s : university.getUniversityStreams()) {
        map.put(s.getStreamId(), s.getStreamName());
    }
    map.put("email", university.getUniversityEmailFormat());
    return map;
}

@GetMapping("/register/validateEmail")
public @ResponseBody
boolean isValidEmail(@RequestParam String email) {
    for (User u : userService.get()) {
        if (u.getEmail().equalsIgnoreCase(email) && u.getActive() == true) {
            return false;
        }
    }
    return true;
}
}

```

5) SearchController

```

package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.iservice.IStreamService;
import com.uniconnect.iservice.IStudentService;
import com.uniconnect.iservice.IUniversityService;
import com.uniconnect.model.Friend;
import com.uniconnect.model.Student;
import com.uniconnect.model.StudentSearchForm;

```

```

import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Controller
public class SearchController {

    @Autowired
    IUniversityService universityService;
    @Autowired
    IStreamService streamService;
    @Autowired
    IStudentService studentService;

    @GetMapping("/search")
    public String getStudents(Model model, HttpServletRequest req){
        User user = ActiveUser.IsActiveStudent(req);
        if (user == null) {
            return "redirect:/error";
        }
        model.addAttribute("lstUniversity", universityService.get());
        model.addAttribute("lstStreams", streamService.get());
        List<Student> lstStudent = studentService.filterStudent(0,8,null,null,null,user.getStudentId());
        List<Friend> friends = studentService.getFriendsAmongList(user.getStudentId(),lstStudent);
        Map<Integer, Integer> friendsMap = new HashMap<>();
        for(Student s : lstStudent){
            Friend fs = friends.stream().filter(x->x.getSenderId().equals(s.getStudentId()) &&
x.getReceiverId().equals(user.getStudentId())).findFirst().orElse(null);
            Friend fr = friends.stream().filter(x->x.getReceiverId().equals(s.getStudentId()) &&
x.getSenderId().equals(user.getStudentId())).findFirst().orElse(null);
            if(fs!=null){
                if(fs.getIsApproved()==0){
                    friendsMap.put(s.getStudentId(),1);// pending req from our side
                }
                if(fs.getIsApproved()==2){
                    friendsMap.put(s.getStudentId(),2);// already friends
                }
            }
            if(fr!=null){
                if(fr.getIsApproved()==0){
                    friendsMap.put(s.getStudentId(),3);// pending req from Their side
                }
                if(fr.getIsApproved()==2){
                    friendsMap.put(s.getStudentId(),2);// already friends
                }
            }
        }
        model.addAttribute("friendsMap",friendsMap);
        model.addAttribute("lstStudents",lstStudent);
        return "student/search";
    }
}

```

```

@PostMapping("/search")
public String searchStudents(HttpServletRequest req, Model model){
    User user = ActiveUser.IsActiveStudent(req);
    if (user == null) {
        return "redirect:/error";
    }
    Integer universityId =
req.getParameter("universityId")==null?null:Integer.parseInt(req.getParameter("universityId"));
    Integer streamId = req.getParameter("streamId")==null?null:
Integer.parseInt(req.getParameter("streamId"));
    Integer start = req.getParameter("start")==null?0: Integer.parseInt(req.getParameter("start"));
    Integer pageSize = req.getParameter("pageSize")==null?0:
Integer.parseInt(req.getParameter("pageSize"));

    String role = req.getParameter("aspiredRole");

    model.addAttribute("lstUniversity", universityService.get());
    model.addAttribute("lstStreams",streamService.get());
    List<Student>
lstStudent=studentService.filterStudent(start,pageSize,universityId,streamId,role,user.getStudentId());
    model.addAttribute("lstStudents",lstStudent);
    List<Friend> friends = studentService.getFriendsAmongList(user.getStudentId(),lstStudent);
    Map<Integer, Integer> friendsMap = new HashMap<>();
    for(Student s : lstStudent){
        Friend fs = friends.stream().filter(x->x.getSenderId().equals(s.getStudentId()) &&
x.getReceiverId().equals(user.getStudentId())).findFirst().orElse(null);
        Friend fr = friends.stream().filter(x->x.getReceiverId()==s.getStudentId() &&
x.getSenderId().equals(user.getStudentId())).findFirst().orElse(null);
        if(fs!=null){
            if(fs.getIsApproved()==0){
                friendsMap.put(s.getStudentId(),1);// pending req from our side
            }
            if(fs.getIsApproved()==2){
                friendsMap.put(s.getStudentId(),2);// already friends
            }
        }
        if(fr!=null){
            if(fr.getIsApproved()==0){
                friendsMap.put(s.getStudentId(),3);// pending req from Their side
            }
            if(fr.getIsApproved()==2){
                friendsMap.put(s.getStudentId(),2);// already friends
            }
        }
    }
    model.addAttribute("friendsMap",friendsMap);
    model.addAttribute("req",req);
    return "student/search";
}

```

6) RequestController

```

package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.Email;
import com.uniconnect.iservice.IStudentService;
import com.uniconnect.iservice.IUserService;

```



```

import com.uniconnect.model.RequestMapping;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

import javax.servlet.http.HttpServletRequest;

@Controller
public class RequestController {

    @Autowired
    IStudentService studentService;

    @Autowired
    IUserService userService;

    @GetMapping("/admin/requests")
    public String getRequests(HttpServletRequest req, Model model) {

        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if (user == null) {
            return "redirect:/error";
        }
        User user2= userService.get(user.getUserId());
        model.addAttribute("requests", user2.getRequests());
        return "admin/request/index";
    }

    @GetMapping("/admin/requests/{id}/{type}")
    public String processReq(@PathVariable("id") int id, @PathVariable("type") String type,
HttpServletRequest req) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if (user == null) {
            return "redirect:/error";
        }
        User user2= userService.get(user.getUserId());
        RequestMapping rm = user2.getRequests().stream().filter(x -> x.getRequestMappingID() ==
id).findFirst().orElse(null);
        if (rm == null) {
            return "redirect:/error";
        }
        if(type.equalsIgnoreCase("approve")){
            Email.send("system.uniconnect@gmail.com", "Ingram@123", rm.getStudentEmail(), "Request Mapping
Status: "+type, "Your request for course professor mapping has been approved. Please visit the uniconnect
website");
        }
        else {
            Email.send("system.uniconnect@gmail.com", "Ingram@123", rm.getStudentEmail(), "Request Mapping
Status: "+type, "Your request for course professor mapping has been rejected. Please visit the uniconnect
website");
        }
        rm.setActive(false);
        studentService.updateRequestMapping(rm);
        return "redirect:/admin/requests";
    }
}

```

7) ProfessorController:

```
package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.ICourseService;
import com.uniconnect.iservice.IProfessorService;
import com.uniconnect.iservice.IUniversityService;
import com.uniconnect.model.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.*;

@Controller
public class ProfessorController {
    @Autowired
    private IProfessorService professorService;

    @Autowired
    IUniversityService universityService;

    @Autowired
    ICourseService courseService;

    //fetch all
    @GetMapping("/admin/professors")
    public String get(HttpServletRequest req, Model model) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if(user==null){
            return "redirect:/error";
        }
        List<Professor> list = professorService.get(user.getUniversityId());
        Collections.sort(list, Comparator.comparing(Professor::getProfessorName));
        model.addAttribute("list", list);
        model.addAttribute("universityAdmin", user);
        return "admin/professor/index";
    }

    //create new
    @GetMapping("/admin/professors/new")
    public String add(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("universityAdmin", user);
        model.addAttribute("professor", new Professor());
        model.addAttribute("courses", courseService.get(user.getUniversityId()));
        return "admin/professor/new";
    }
}
```

```

//create operation
@PostMapping("/admin/professors")
public String add(@Valid Professor professor,
    BindingResult result, Model model, HttpServletRequest req) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){
        model.addAttribute("courses", courseService.get(user.getUniversityId()));
        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    model.addAttribute("duplicate", "");
    if (result.hasErrors()) {
        model.addAttribute("courses", courseService.get(user.getUniversityId()));
        return "admin/professor/new";
    }
    professor.setProfessorName(StringFormatter.capitalizeWord(professor.getProfessorName().trim()));

    if (invalid(professor.getProfessorEmail(), universityService.get(user.getUniversityId()))) {
        model.addAttribute("duplicate", "Professor with same email already exists");
        model.addAttribute("courses", courseService.get(user.getUniversityId()));
        return "admin/professor/new";
    }
    String[] checkedIds = req.getParameterValues("professorCourses");
    Set<Course> courses = new HashSet<>();
    for (String courseId : checkedIds) {
        courses.add(courseService.get(user.getUniversityId(), Integer.parseInt(courseId)));
    }
    professor.setCourses(courses);
    if (professorService.add(user.getUniversityId(), professor) == 1) {
        return "redirect:/admin/professors";
    } else {
        return "redirect:/error";
    }
}

//edit
@GetMapping("/admin/professors/{id}/edit")
public String edit(@PathVariable("id") int id, Model model, HttpServletRequest req) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){
        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    Professor professor = professorService.get(user.getUniversityId(), id);
    model.addAttribute("professor", professor);
    model.addAttribute("courses", courseService.get(user.getUniversityId()));
    Set<Integer> courseId = new HashSet<>();
    professor.getCourses().forEach(x->courseId.add(x.getCourseId()));
    model.addAttribute("courseId", courseId);
    return "admin/professor/new";
}

//update
@PostMapping("/admin/professors/{id}")
public String edit(@PathVariable("id") int id, @Valid Professor professor, BindingResult result, Model
model, HttpServletRequest req) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){
        model.addAttribute("courses", courseService.get(user.getUniversityId()));
        return "redirect:/error";
    }
}

```

```

model.addAttribute("universityAdmin", user);
model.addAttribute("duplicate", "");
if (result.hasErrors()) {
    model.addAttribute("courses", courseService.get(user.getUniversityId()));
    return "admin/professor/new";
}

professor.setProfessorName(StringFormatter.capitalizeWord(professor.getProfessorName().trim()));
if (invalid(professor.getProfessorEmail(), universityService.get(user.getUniversityId(), id)) {
    model.addAttribute("duplicate", "Professor with same email already exists");
    model.addAttribute("courses", courseService.get(user.getUniversityId()));
    return "admin/professor/new";
}

String[] checkedIds = req.getParameterValues("professorCourses");
Set<Course> courses = new HashSet<>();
for (String courseId : checkedIds) {
    courses.add(courseService.get(user.getUniversityId(), Integer.parseInt(courseId)));
}
professor.setCourses(courses);

if (professorService.update(user.getUniversityId(), id, professor) == 1) {
    return "redirect:/admin/professors";
} else {
    return "redirect:/error";
}
}

//Delete
@GetMapping("/admin/professors/{id}/delete")
public String delete(@PathVariable("id") int id, HttpServletRequest req, Model model) {
    User user = ActiveUser.isActiveUniversityAdmin(req);
    if (user == null) {
        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    model.addAttribute("courses", courseService.get(user.getUniversityId()));
    if (professorService.delete(user.getUniversityId(), id) == 1) {
        return "redirect:/admin/professors";
    } else {
        return "redirect:/error";
    }
}

private boolean invalid(String professorEmail, University university) {
    for (Professor p : university.getUniversityProfessor()) {
        if (p.getProfessorEmail().equalsIgnoreCase(professorEmail)) {
            return true;
        }
    }
    return false;
}

private boolean invalid(String professorEmail, University university, int id) {
    for (Professor p : university.getUniversityProfessor()) {
        if (p.getProfessorEmail().equalsIgnoreCase(professorEmail) && p.getProfessorId() != id) {
            return true;
        }
    }
    return false;
}

```

```
}  
}
```

8) MessageController

```
package com.uniconnect.controller;  
  
import com.uniconnect.common.ActiveUser;  
import com.uniconnect.iservice.IStudentService;  
import com.uniconnect.model.Message;  
import com.uniconnect.model.OutputMessage;  
import com.uniconnect.model.Student;  
import com.uniconnect.model.User;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.messaging.handler.annotation.Header;  
import org.springframework.messaging.handler.annotation.MessageMapping;  
import org.springframework.messaging.handler.annotation.Payload;  
import org.springframework.messaging.simp.SimpMessagingTemplate;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
import javax.servlet.http.HttpServletRequest;  
import java.security.Principal;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
@Controller  
public class MessageController {  
    @Autowired  
    IStudentService studentService;  
  
    @Autowired  
    private SimpMessagingTemplate simpMessagingTemplate;  
  
    @GetMapping("/message")  
    public String getConnectionMessages(HttpServletRequest req, Model model){  
        User user = ActiveUser.IsActiveStudent(req);  
        if (user == null) {  
            return "redirect:/login";  
        }  
        List<Student> lstConnection = new ArrayList<>();  
  
        List<Integer> friends= studentService.getConnections(user.getStudentId());  
        friends.forEach(integer -> lstConnection.add(studentService.get(integer)));  
        model.addAttribute("lstConnection", lstConnection);  
        model.addAttribute("fromUser",user.getStudentId());  
        return "student/message";  
    }  
  
    @MessageMapping("/room")  
    public void sendSpecific(  
        @Payload Message msg,  
        Principal user,  
        @Header("simpSessionId") String sessionId) throws Exception {  
        OutputMessage out = new OutputMessage(  

```

```

        msg.getFrom(),
        msg.getTo(),
        msg.getText(),
        new SimpleDateFormat("HH:mm").format(new Date()));
    simpMessagingTemplate.convertAndSendToUser(
        msg.getTo(), "/user/queue/specific-user", out);
    }
}

```

9) LoginController

```

package com.uniconnect.controller;

import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.IUserService;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

@Controller
public class LoginController {

    @Autowired
    IUserService userService;

    @PostMapping("/login")
    public String login(HttpServletRequest req, ModelMap model) {
        String email = req.getParameter("email");
        String password = req.getParameter("password");
        model.addAttribute("login", "");
        User user = userService.login(email, password);
        if (user == null || !user.getActive()) {
            model.addAttribute("login", "Email or Password incorrect");
            if (user != null && !user.getActive()) {
                model.addAttribute("login", "User is inactive");
            }
            return "index";
        }
        HttpSession session = req.getSession(true);
        session.setAttribute("activeUser", user);
        if (user.getRole().equalsIgnoreCase("superuser")) {
            return "redirect:/admin/streams";
        }
        if (user.getRole().equalsIgnoreCase("universityadmin")) {
            return "redirect:/admin/courses";
        }
        model.clear();
        session.setAttribute("studentId", user.getStudentId());
        String name = user.getName().split("\\-")[0];
        name = StringFormatter.capitalizeWord(name);
        session.setAttribute("studentName", name);
        return "redirect:/uc/" + user.getStudentId();
    }
}

```

```

}

@GetMapping("/{"/,"/login"})
public String homepage(){
    return "index";
}

@GetMapping("/logout")
public String logout(HttpServletRequest req){
    HttpSession session = req.getSession(true);
    if(session.getAttribute("activeUser")!=null){
        session.removeAttribute("activeUser");
        session.removeAttribute("studentId");
    }
    return "index";
}
}
}

```

10) ImageController

```

package com.uniconnect.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

@Controller
public class ImageController {
    @RequestMapping(value = "/getImage/{imageId:.+}")
    @ResponseBody
    public byte[] getImage(@PathVariable String imageId, HttpServletRequest request) throws IOException {
        Path path =
Paths.get("C:\\Users\\phapa\\Desktop\\Newfolder\\UniConnect\\Project\\src\\main\\ImageData\\"+imageId);
        byte[] data = Files.readAllBytes(path);
        return data;
    }
}

```

11) CourseController

```

package com.uniconnect.controller;

import com.uniconnect.common.ActiveUser;
import com.uniconnect.common.StringFormatter;
import com.uniconnect.iservice.ICourseService;
import com.uniconnect.iservice.IUniversityService;
import com.uniconnect.model.Course;
import com.uniconnect.model.University;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;

```

```

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

@Controller
public class CourseController {
    @Autowired
    private ICourseService courseService;

    @Autowired
    IUniversityService universityService;

    //fetch all
    @GetMapping("/admin/courses")
    public String get(HttpServletRequest req, Model model) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if(user==null){
            return "redirect:/error";
        }
        List<Course> list = courseService.get(user.getUniversityId());
        Collections.sort(list, Comparator.comparing(Course::getCourseName));
        model.addAttribute("list", list);
        model.addAttribute("universityAdmin", user);
        return "admin/course/index";
    }

    //create new
    @GetMapping("/admin/courses/new")
    public String add(Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("universityAdmin", user);
        model.addAttribute("course", new Course());
        return "admin/course/new";
    }

    //create operation
    @PostMapping("/admin/courses")
    public String add(@Valid Course course,
        BindingResult result, Model model, HttpServletRequest req) {
        User user = ActiveUser.IsActiveUniversityAdmin(req);
        if(user==null){
            return "redirect:/error";
        }
        model.addAttribute("universityAdmin", user);
        model.addAttribute("duplicate", "");
        if (result.hasErrors()) {
            return "admin/course/new";
        }
        course.setCourseName(StringFormatter.capitalizeWord(course.getCourseName().trim()));
    }
}

```



```

        if
(invalid(course.getCourseName(),course.getCourseCode(),universityService.get(user.getUniversityId())) {
    model.addAttribute("duplicate","Course already exists with same name or course code");
    return "admin/course/new";
}

    if (courseService.add(user.getUniversityId(),course) == 1) {
        return "redirect:/admin/courses";
    } else {
        return "redirect:/error";
    }
}

//edit
@GetMapping("/admin/courses/{id}/edit")
public String edit(@PathVariable("id") int id, Model model, HttpServletRequest req) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){
        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    Course course = courseService.get(user.getUniversityId(),id);
    model.addAttribute("course", course);
    return "admin/course/new";
}

//update
@PostMapping("/admin/courses/{id}")
public String edit(@PathVariable("id") int id, @Valid Course course, BindingResult result, Model model,
HttpServletRequest req) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){
        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    model.addAttribute("duplicate","");
    if (result.hasErrors()) {
        return "admin/course/new";
    }

    course.setCourseName(StringFormatter.capitalizeWord(course.getCourseName().trim()));

if(invalid(course.getCourseName(),course.getCourseCode(),universityService.get(user.getUniversityId()),id))
{
    model.addAttribute("duplicate","Course already exists with same course name or course code");
    return "admin/course/new";
}
    if (courseService.update(user.getUniversityId(),id, course) == 1) {
        return "redirect:/admin/courses";
    } else {
        return "redirect:/error";
    }
}

//Delete
@GetMapping("/admin/courses/{id}/delete")
public String delete(@PathVariable("id") int id, HttpServletRequest req, Model model) {
    User user = ActiveUser.IsActiveUniversityAdmin(req);
    if(user==null){

```

```

        return "redirect:/error";
    }
    model.addAttribute("universityAdmin", user);
    if (courseService.delete(user.getUniversityId(), id) == 1) {
        return "redirect:/admin/courses";
    } else {
        return "redirect:/error";
    }
}

private boolean invalid(String courseName, String courseCode, University university) {
    for (Course c : university.getUniversityCourses()) {
        if (c.getCourseCode().equalsIgnoreCase(courseCode) ||
c.getCourseName().equalsIgnoreCase(courseName)) {
            return true;
        }
    }
    return false;
}

private boolean invalid(String courseName, String courseCode, University university, int id) {
    for (Course c : university.getUniversityCourses()) {
        if ((c.getCourseCode().equalsIgnoreCase(courseCode) ||
c.getCourseName().equalsIgnoreCase(courseName)) && c.getCourseId() != id) {
            return true;
        }
    }
    return false;
}
}

```

12) ConnectionController

```

package com.uniconnect.controller;

import com.uniconnect.iservice.IStudentService;
import com.uniconnect.model.Connection;
import com.uniconnect.model.Friend;
import com.uniconnect.model.Student;
import com.uniconnect.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

@Controller
public class ConnectionController {

    @Autowired
    IStudentService studentService;

    @GetMapping("/Connect/{studentId}")
    public String connect(@PathVariable("studentId") String studentId, HttpServletRequest req) {
        HttpSession session = req.getSession(true);
        User user = (User) session.getAttribute("activeUser");
    }
}

```

```

        if (user == null && user.getPassword() != null && user.getLoginCode() != null) {
            return "redirect:/error";
        }
        Student stdFrom = studentService.get(user.getStudentId());
        Student stdTo = studentService.get(Integer.parseInt(studentId));
        Friend friend = new Friend();
        friend.setSenderId(stdFrom.getStudentId());
        friend.setReceiverId(stdTo.getStudentId());
        friend.setIsApproved(0);
        studentService.addConnection(friend);
        return "redirect:/uc/"+stdTo.getStudentId();
    }

    @GetMapping("/Approve/{studentId}")
    public String Approve(@PathVariable("studentId") String studentId, HttpServletRequest req){
        HttpSession session = req.getSession(true);
        User user = (User) session.getAttribute("activeUser");
        if (user == null && user.getPassword() != null && user.getLoginCode() != null) {
            return "redirect:/error";
        }
        studentService.approve(user.getStudentId(),Integer.parseInt(studentId));
        return "redirect:/uc/"+Integer.parseInt(studentId);
    }

    @GetMapping("/Reject/{studentId}")
    public String Reject(@PathVariable("studentId") String studentId, HttpServletRequest req){
        HttpSession session = req.getSession(true);
        User user = (User) session.getAttribute("activeUser");
        if (user == null && user.getPassword() != null && user.getLoginCode() != null) {
            return "redirect:/error";
        }
        studentService.reject(user.getStudentId(),Integer.parseInt(studentId));
        return "redirect:/uc/"+Integer.parseInt(studentId);
    }

    @GetMapping("/UnFriend/{studentId}")
    public String UnFriend(@PathVariable("studentId") String studentId, HttpServletRequest req){
        HttpSession session = req.getSession(true);
        User user = (User) session.getAttribute("activeUser");
        if (user == null && user.getPassword() != null && user.getLoginCode() != null) {
            return "redirect:/error";
        }
        studentService.unFriend(user.getStudentId(),Integer.parseInt(studentId));
        return "redirect:/uc/"+Integer.parseInt(studentId);
    }
}

```