University Application System
Final Report

Group 32
Joshua Plosz
Morgan Li
Md Azharul Islam Fahim

December 24, 2021

ABSTRACT

When young adults strive to further their education via tertiary institutions they need tools that do not distract nor discourage them from finding the right place for them. Our website, University Application System, helps these individuals narrow their search between thousands of Universities by providing ways to filter the irrelevant institutions via a custom search profile.

Our website also allows institutions to access what information is provided. They are incentivized to add more relevant and updated data to find their institute on more students' search results.

The pre-built technology used in our development includes MongoDB, Mongoose.js, Express.js, React.js, and Node.js. This tech-stack follows the NoSQL methodology for database management.

To provide accessibility to other programs our website contains an API. This report concludes with the API specifications which includes all endpoints as well as examples for clearer descriptions.

INDEX

1.0 PROBLEM DESCRIPTION

1.1 Introduction

Accumulating, evaluating, and ranking hundreds of potential tertiary education institutions around the world, on an individual level, is simply too much to ask of senior high school students or working adults. University ranking websites give you an abstract overview of how the world evaluates these institutions, but do not provide the ability to evaluate them on a personal level.

What we need is a website that allows people to tailor their University searches using personal criteria. Lessening the need to delve into research on institutions that ultimately do not match your expectations or limitations.

1.2 Problem

Every year thousands of students fly to different countries for higher studies. Canada is one of the favorite destinations among international students. According to the government of Canada, in 2017 and 2018, international students spent $18.4B and $22.3B, respectively, in economic activities in Canada [1]. This represents a significant number of students who receive no proper guidelines. Some resources come close to providing an adequate solution, but the problem is not properly solved yet.

When a recent high school graduate tries to decide where they want to go for higher education, there are many factors for them to consider. Country, city, university ranking, tuition, program, along with many more personal preferences or region requirements. Because of the lack of information conveniently available, many prospective students end up reconsidering obtaining a University education or find themselves settling for an unsuitable institution.

Mastersportal.com [2] is a website that attempts to provide students with better search options. Giving incite on the graduate programs for a limited list of Universities. But the information presented is meant for international graduate students, where our solution is meant to help domestic and international undergraduate students. While a Mastersportal.com user can alter their search by selecting relevant parameters, the list of results does not appear sorted. In fact there is no way to prioritize the ordering of the list by any criteria. An appropriate solution should provide some means of ranking the Universities while giving the user the means to sort the results by other criteria.

1.3 Solution

Our solution is to consolidate some of the important requirements into one place. Where an individual can search for Universities based on the merits they already have. In our solution, the resulting search will provide Universities in sorted order, according to University ranking which we think will help the students to choose and know more about the standing of their

desired institutions. Once they have a list of possible institutions they can further explore those Universities' websites. This can limit their search for Universities from hundreds to dozens or less.

The goal is to produce a "first stop" resource for any prospective University student around the world. Our main feature is creating a profile of optional criteria that represents individuals desired qualifications.

Typical criteria:

(a) location of institution

(b) program availability

(c) tuition maximum (domestic and foreign)

(d) primary language requirement (required language proficiency test)

(e) ranking (backed by a scoring system like QS University Ranking [3])

(f) domestic to foreign student ratio

Accumulating information on all Universities world-wide would be an immense and unnecessary task for a group of 3 in such a small time frame. To show our concept on a smaller scale, we have constructed our database of Universities using select institutions that provide various requirements.

2.0 PROJECT DESIGN

2.1 User Functionality

Our application considers three categories of users.

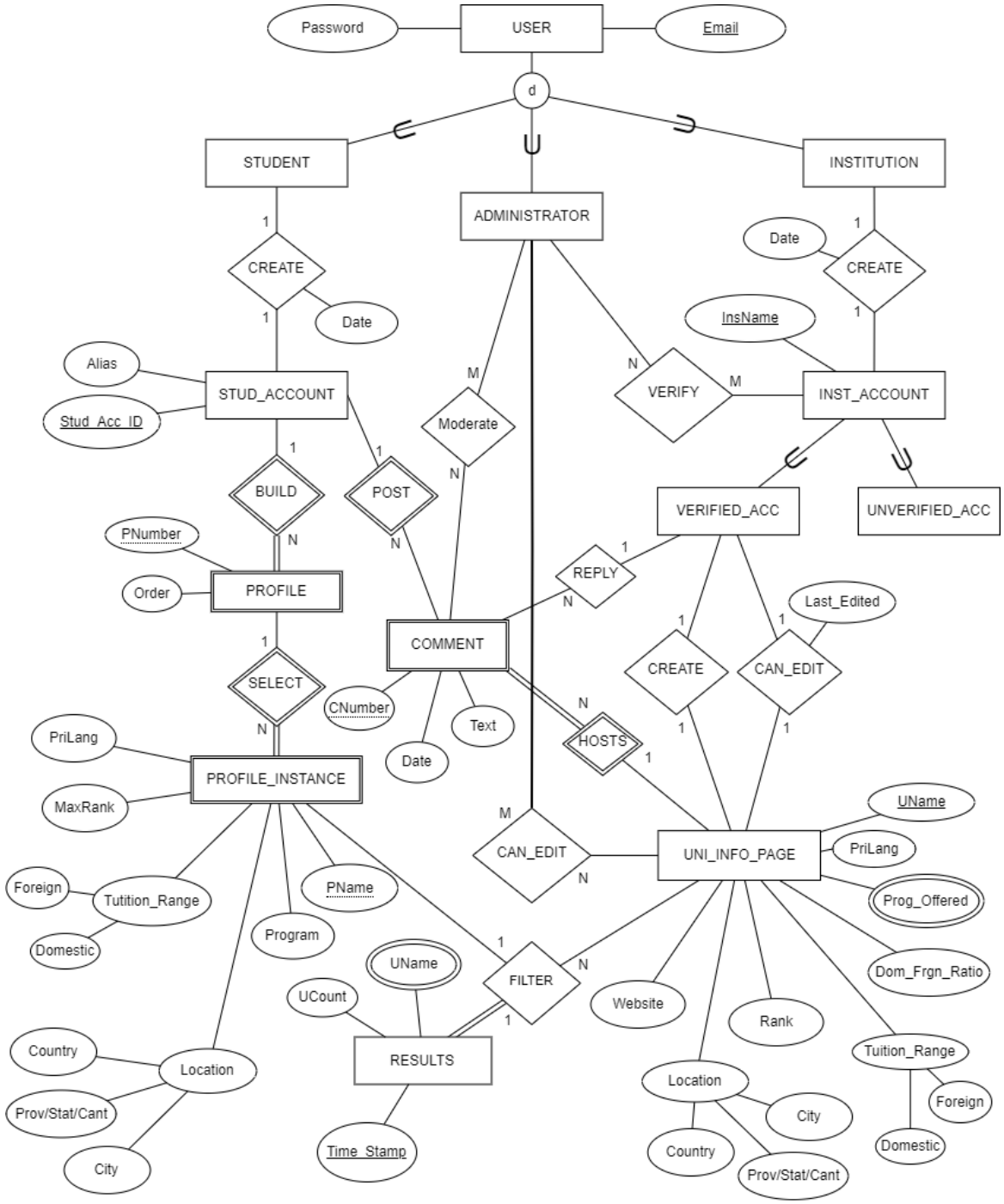
1. Student
2. Institute
3. Administration

Student users have the ability to create a search profile that filters through Universities in the database. To access the profile they must first register an account with a username, email, and password, then login to their account. From their account they can browse a list of all Universities in the database or select a search option to bring up a form of adjustable parameters. By selecting parameters that match the students needs, the application filters through the database and returns a list of Universities matching those parameters. By selecting one of the resulting Universities you are taken to a University Information Page that displays information on the selected University. Additionally, a student can post comments on a University Information Page.

Institute users must be verified by an administrator before they have any functionality. Once verified they have editable access to their corresponding University's Information Page. Posted on their page is a stamp showing the last time the page was updated.

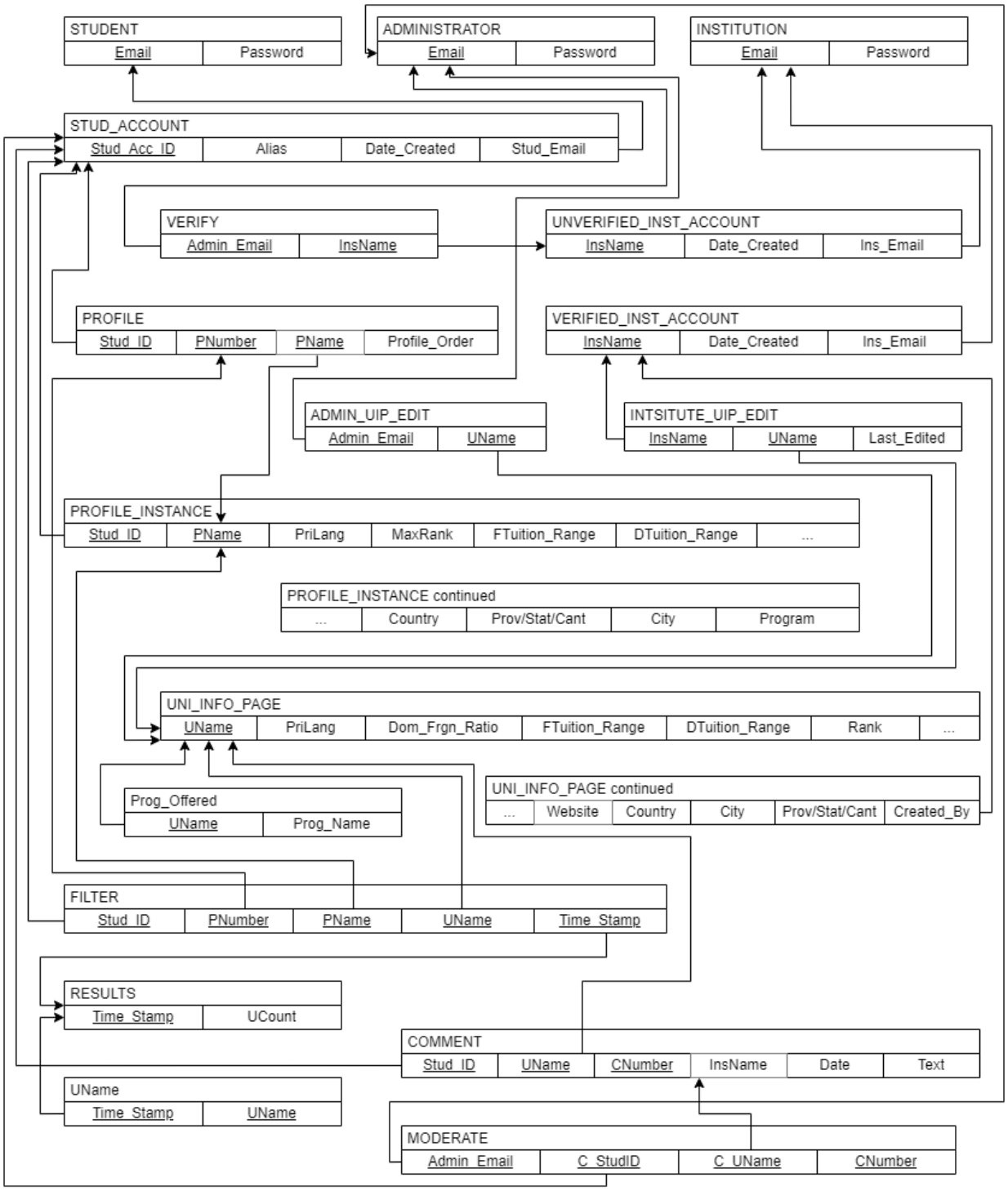
Administration users have editable access to all University Information Pages including the ability to delete Universities entirely. However, their main purpose is to verify an institute account by confirming a University is indeed under their ownership. They also moderate comments by having the ability to delete comments on University Information Pages.

2.2 Enhanced Entity Relationship Diagram



3.0 IMPLEMENTATION

3.1 Relational Model



3.2 Database Management System

Our application uses a NoSQL database management system. Using a document based, key-value pair data management design. This does away with relational tables and makes the Relational Model above more of a guideline for document modeling.

The pre-built technology used in our development includes:

1. MongoDB; a popular program for NoSQL database management.
2. Mongoose.js; an object data modeling library for MongoDB.
3. Express.js; a server framework for web applications and APIs.
4. React.js; a UI development library.
5. Node.js; a Javascript runtime environment

The following is a list of database models used in our application.

3.2.1.1 User

```
const userInfo = new mongoose.Schema(
    {  //CHANGED UNIQUE TO TRUE FOR USERNAME. MIGHT CAUSE PROBLEMS???
        username: { type: String, required: true, unique: true },
        email: { type: String, required: true, unique: true },
        password: { type: String, required: true },
        isAdmin: { type: Boolean, default: false },
        isInstitution: { type: Boolean, default: false },
    },
    { collection: 'UserInfo' }
)
```

3.2.1.2 Uni

```
const uniInfo = new mongoose.Schema(
    {//CHANGED FTUT AND DTUT TO INT FROM STRING
        Rank: { type: Number },
        Uname: { type: String, required: true, unique: true },
        Prog_Offered: [{ type: String, required: true}],
        Dom_Frgn_Ratio: { type: String},
        PriLang: { type: String, required: true },
        Location: { type: String, required: true },
        FTutition_Range: { type: Number},
        DTutition_Range: { type: Number},
        Website: { type: String},
        Type: {type: String },
    },
```

```
    {timestamps: true},
    { collection: 'UniInfo' }
)
```

3.2.1.3 savedUni

```
const savedUni = new mongoose.Schema(
    {//CAN OPTIMIZE BY SWAPPING UNIID TYPE WITH OBJECTID TO SAVE 12 BITS
(24vs12)
        username: { type: String, required: true},
        Rank: { type: Number },
        Uname: { type: String, required: true},
        Location: { type: String, required: true },
        UniID: {type: String, required: true},
    },
    { collection: 'savedUni' }
)
```

3.2.1.4 comment

```
const comment = new mongoose.Schema(
    {
        UniID: { type: String, required: true },
        username: { type: String, required: true },
        desc: { type: String, required: true },
    },
    {timestamps: true},
    { collection: 'Comments' }
)
```

The following is a list of database queries used in our application.

3.2.1 Creating a new user

```
new User({
      username: req.body.username,
      email: req.body.email,
      password: hashedPass,
      isAdmin: req.body.isAdmin,
      isInstitution: req.body.isInstitution
    });
```

3.2.2 Login user

```
User.findOne({ username: req.body.username });
bcrypt.compare(req.body.password, user.password);
```

3.2.3 Search

```
Uni.find(
        {PriLang: queryPriLang,
        Location: {$regex:queryLocation},
        Dom_Frgn_Ratio:{$gte : queryDom_Frgn_Ratio},
        $or: [{ Prog_Offered: queryProgram}, {Prog_Offered: query3}],
        FTutition_Range:{$lte : queryFTutition},
        DTutition_Range: {$lte : queryDTutition},
        }, {Uname: 1, Rank: 1, Location: 1, Dom_Frgn_Ratio: 1,
            Prog_Offered: 1, FTutition_Range:1, DTutition_Range: 1,
            Type: 1
            }).sort({Rank: 1});
```

3.2.4 Create University Page

```
new Uni(req.body);
```

3.2.5 Sorted List of All Universities

```
Uni.find( {}, {Uname: 1, Rank: 1, Location: 1}).sort({Rank: 1});
```

3.2.6 Single University

```
Uni.findById(req.params.id);
```

3.2.7 Update University

```
Uni.findByIdAndUpdate(req.params.id,
      {
        $set: req.body,
      },
      {new: true}
    );
```

3.2.8 Post Comment

```
new Comment({
        UniID: req.body.UniID,
        username: req.body.Username,
        desc: req.body.Description,
    });
```

3.2.9 Universities Comments

```
Comment.find({UniID: queryParam}).sort({createdAt: -1});
```

3.2.10 Delete Comment

```
Comment.deleteMany({UniID: req.params.id});
```

3.2.11 Delete University

```
Uni.findByIdAndDelete(req.params.id);
```

3.2.12 User's Watchlist

```
savedUni.find(
        {username: queryUsername,
        }, {Uname: 1, Rank: 1, Location: 1, UniID: 1 }).sort({Rank: 1});
```

3.2.13 Save to Watchlist

```
new savedUni(req.body);
```

3.2.14 Check Watchlist

```
savedUni.find(
      {username: queryUsername,
          UniID : queryUniID,
      }, {Uname: 1, Rank: 1, Location: 1, UniID: 1 });
```

3.2.15 Delete from Watchlist

```
savedUni.deleteOne(
      {username: queryUsername,
          UniID : queryUniID,});
```

3.2.16 Update Watchlists

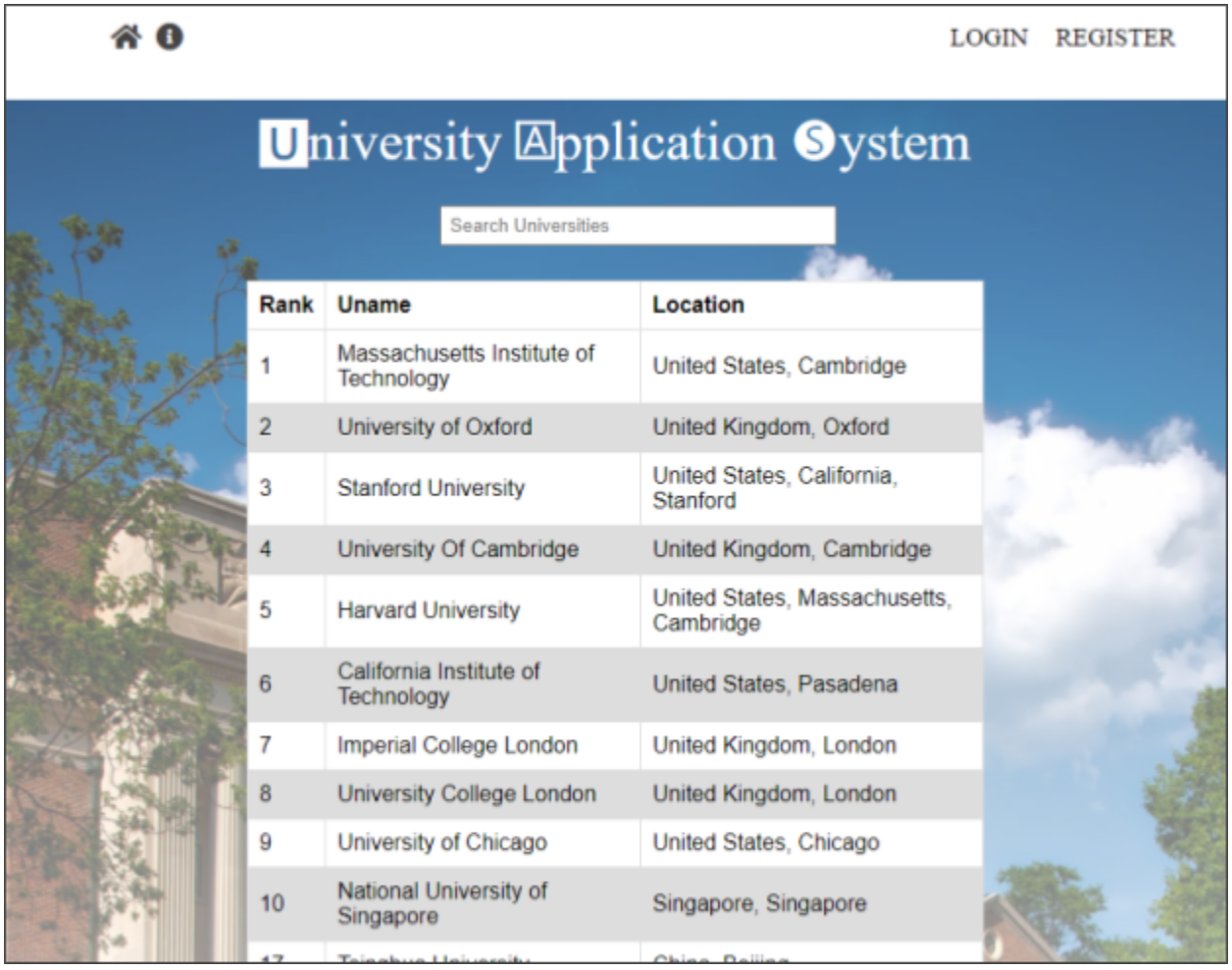
```
savedUni.updateMany({UniID: req.params.id},
      {
        $set: req.body,
      },
      {new: true}
    );
```

3.2.17 Remove from All Watchlists

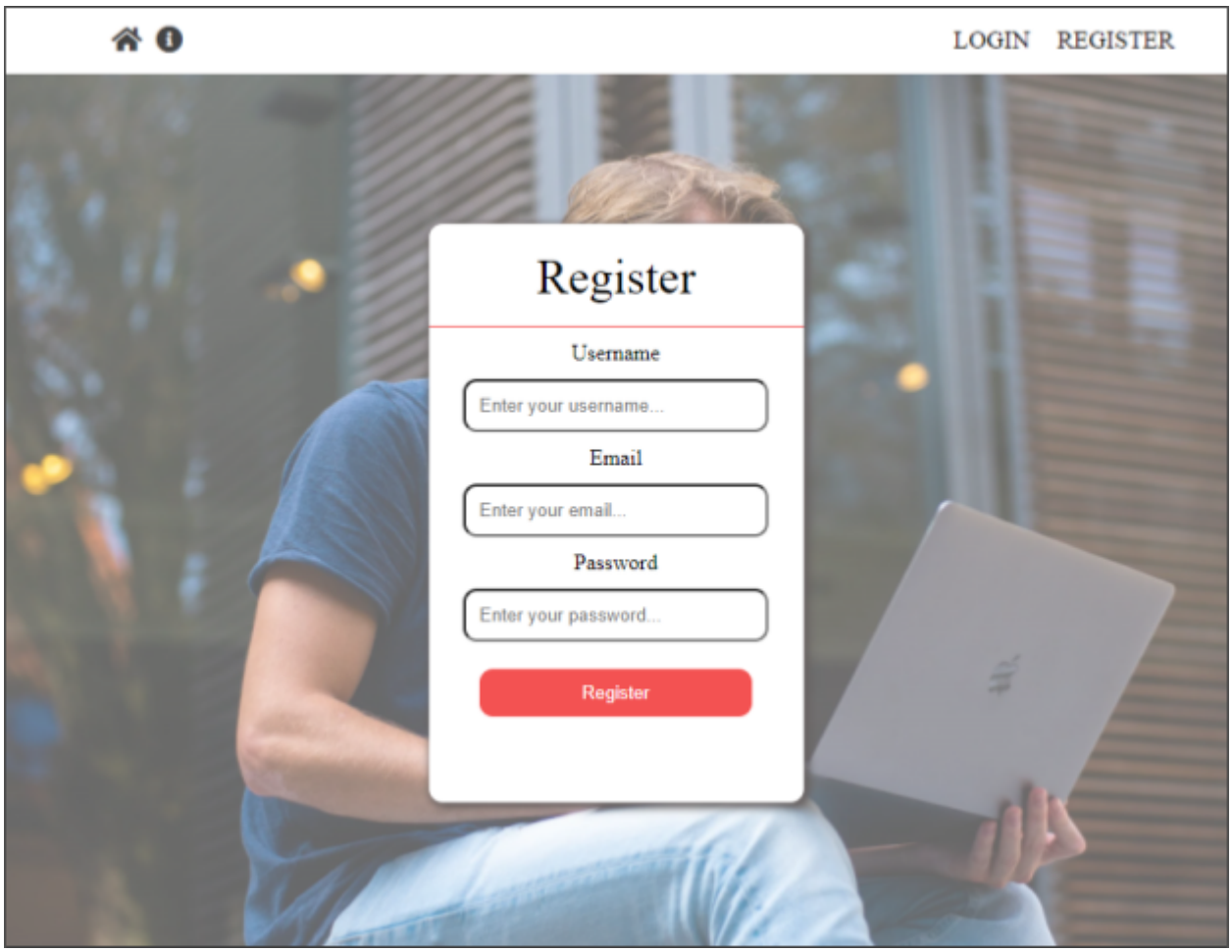
```
savedUni.deleteMany({UniID: req.params.id});
```

4.0 WEBSITE USER GUIDE

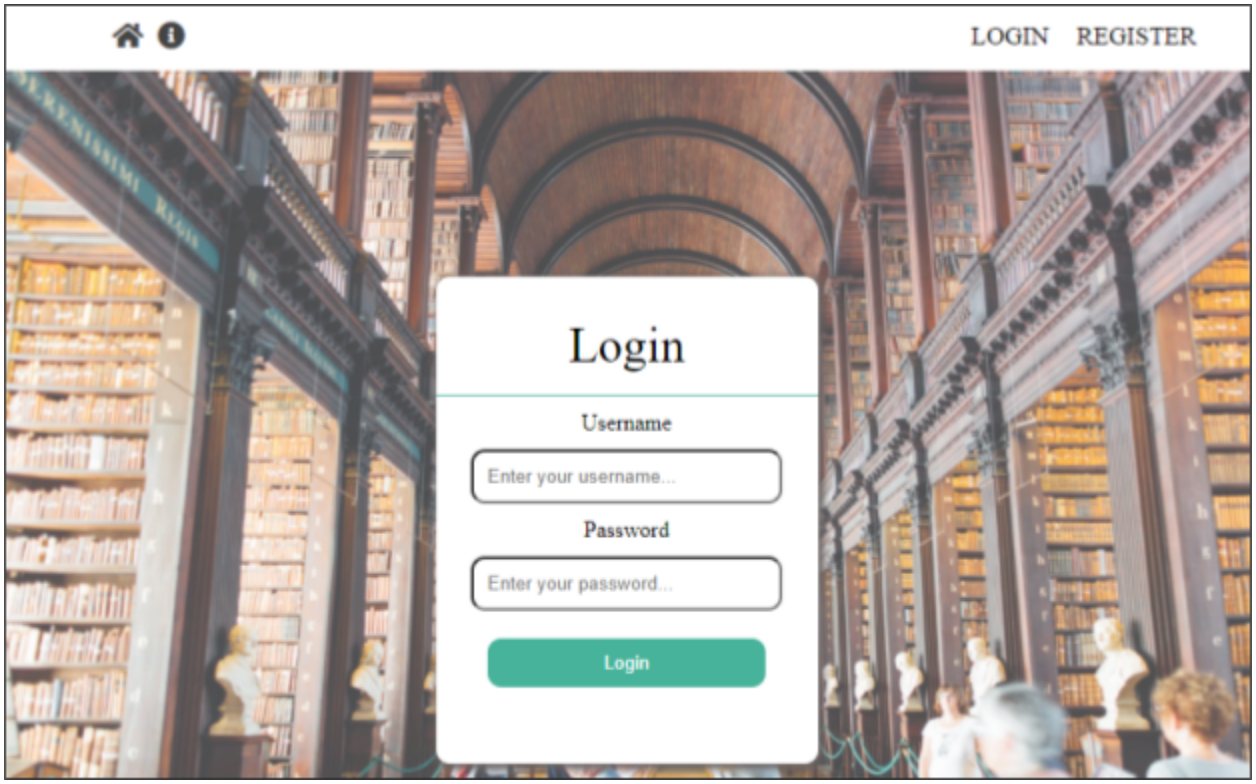
Greeted at the home page (http://localhost:3000) is a list of all Universities stored in our database, sorted in order by world rank. Above the list is a search bar that filters through University names. Located on the top banner is a home link to return to this page, a link to direct you to an “About” page, a link for users to login, and a link for newcomers to register an account.



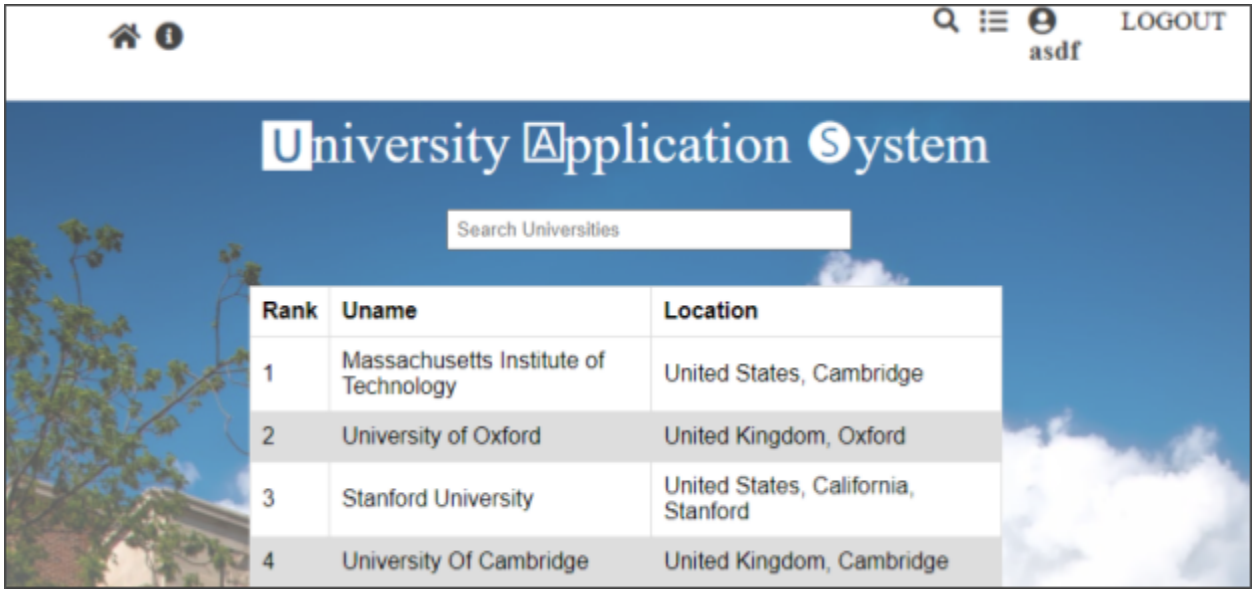
Let us first take a look at registering an account by clicking the “REGISTER” link on the top right. This takes you to the registration page (http://localhost:3000/register) where you are provided with a form. Filling in a preferred username, a valid email address, and a private password is all that is needed to create an account. Pressing the ‘Register’ button submits the form and takes you to the login page (http://localhost:3000/login).



Here you can input your chosen username and password (which is hidden) and hit the ‘Login’ button to login to your account and be directed back to the home page (http://localhost:3000/login).

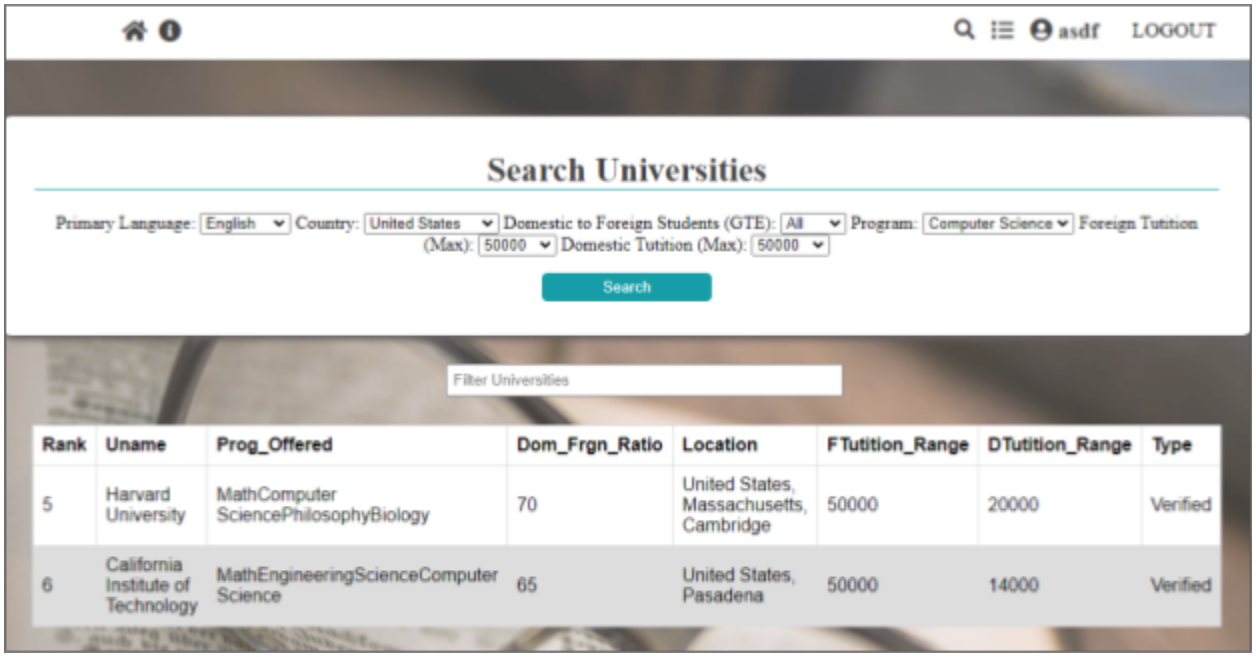


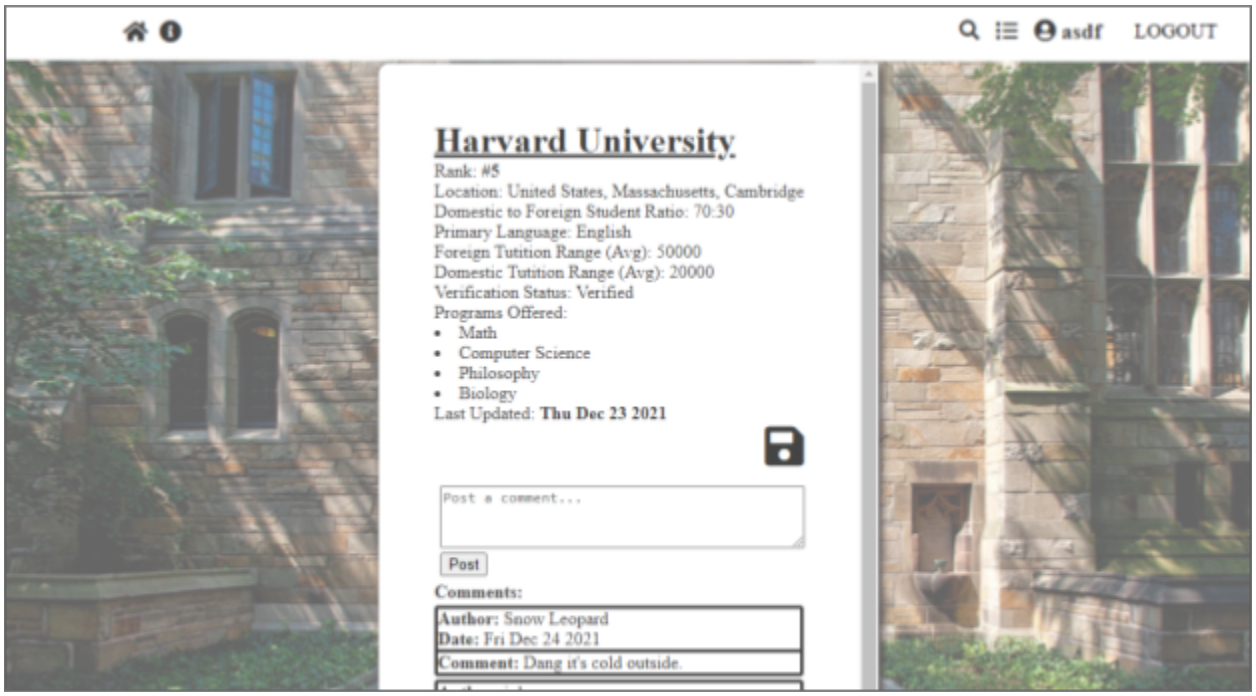
Notice now that the top banner now contains different icons on the top right. Namely, the search, watchlist, and logout icon links starting from the left. Click on the search icon to be directed to the search profile.



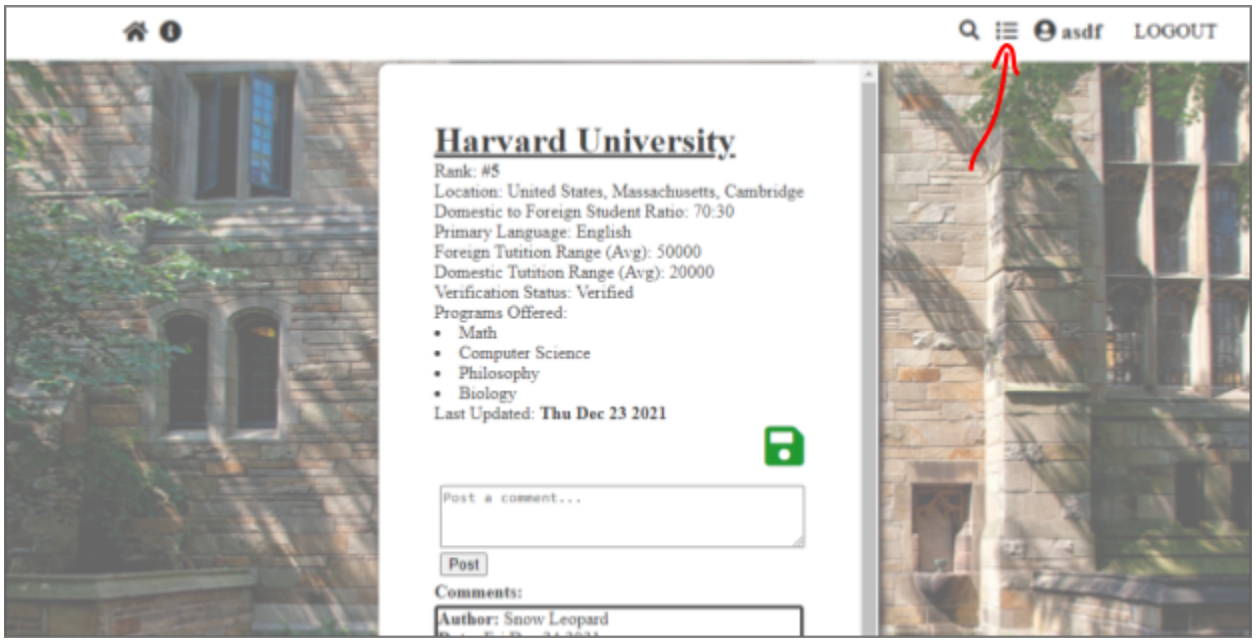
Here you can find another form with dropdown panels to guide your search query. By selecting an option from any number of fields and clicking the ‘Search’ button you will be given a list of Universities from the database that match these options. Selecting one of these Universities will take you to their respective UniPage.

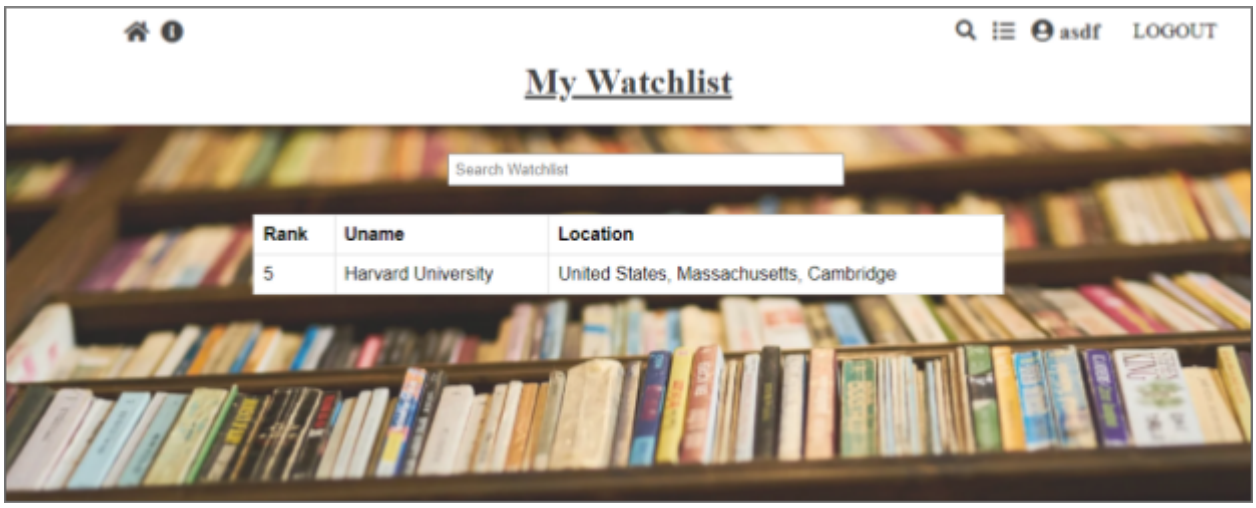
(ex. http://http://localhost:3000/uniPage/61c4426691561101f1a334e9)





Each University has its own University Information page that depicts details about them. You can view those details, click on the University name header to be directed to their website, post a comment to be left for all to read, or save this University page to your watchlist by clicking on the floppy disk icon just above the comment text box. A saved University page is indicated by the floppy disk changing to green. Let's save this page to our watchlist and then click the 'watchlist' icon on the top banner to take us to our watchlist page (http://localhost:3000/watchlist).





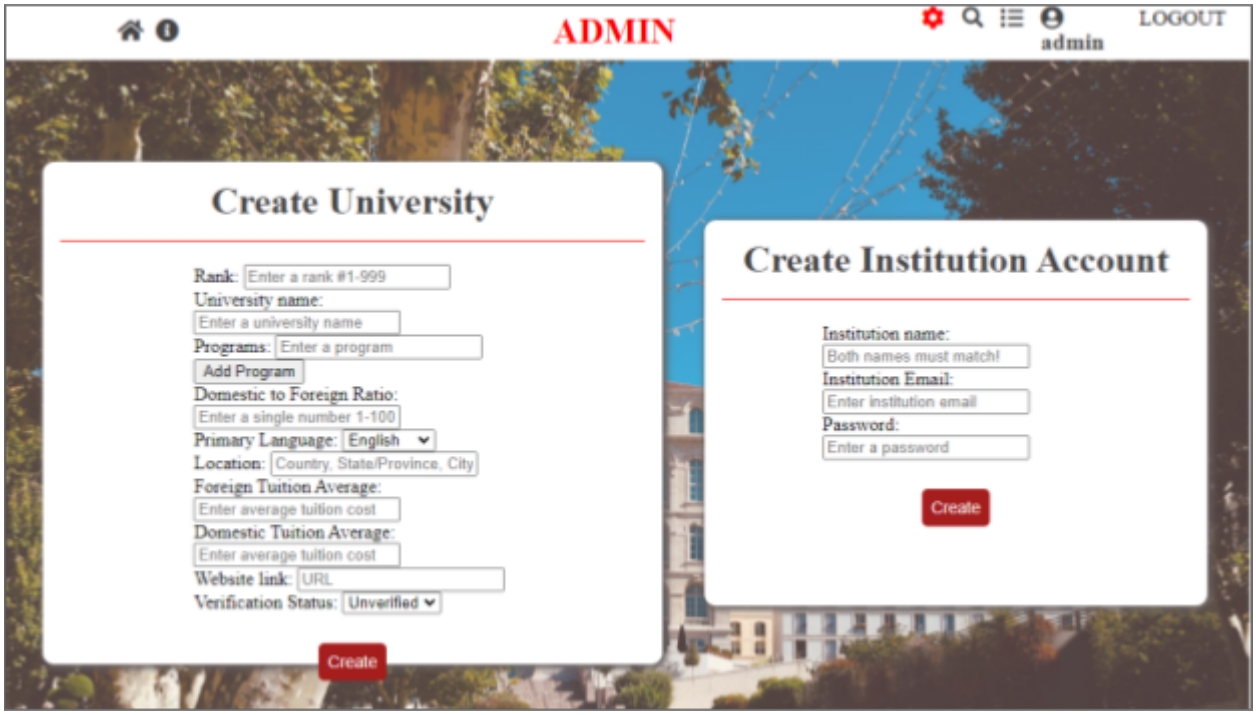
Here we can browse the Universities we are interested in and have saved.

From here we'll logout with the link on the top right of the banner and log back in as an institute (username: University of Calgary; password: cat). Not much looks different so let's search for our institute using the search bar and visit our University page.

You'll notice a new icon at the bottom right of the University Information page. This is the edit icon accessible by Institute accounts. Click the edit icon to edit the page.

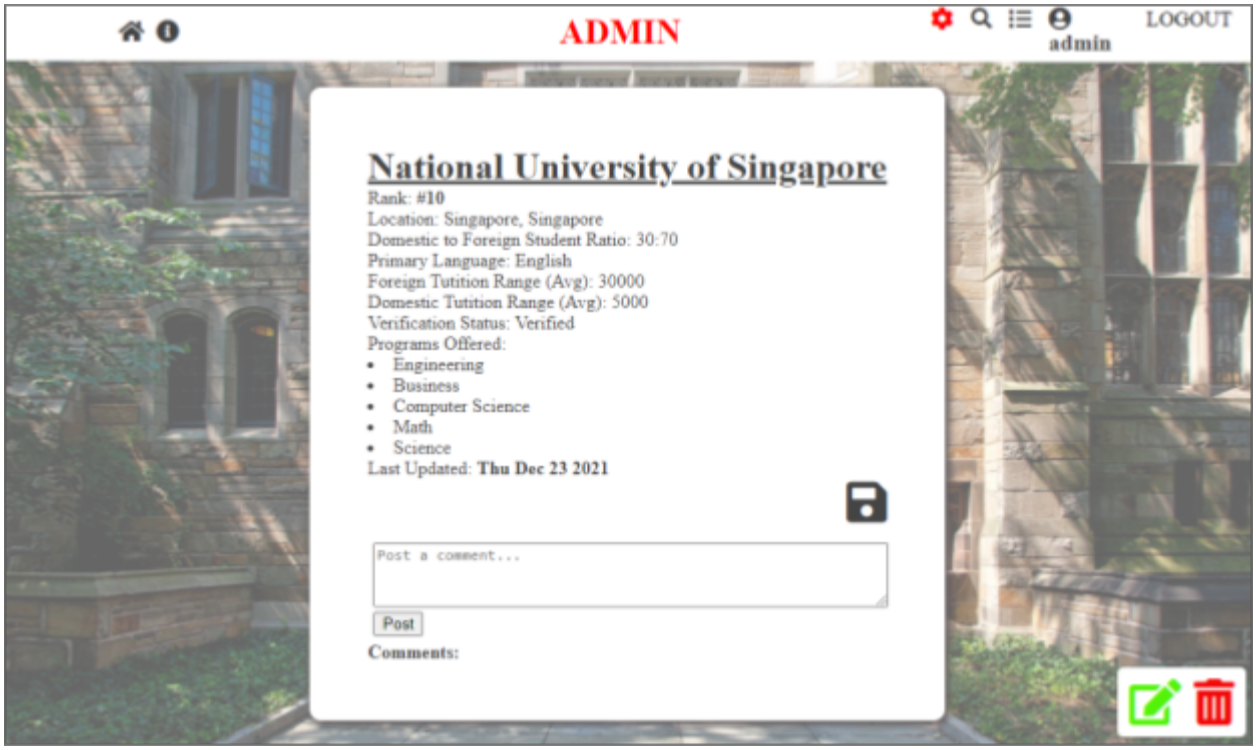
Here you can provide updated information about your institute which will automatically update the timestamp that shows users when the page was last updated.

Logout one more time and log back in as an administrator this time (username: admin; password: admin). You'll notice on the top banner that you have another new cogwheel icon. Click the new icon to be directed to the Admin page (http://localhost:3000/admin).



This is where an administrator can create new University entries for our database as well as accompanying Institution accounts.

Lastly, let us return to the home page and select any University to go to their University Information page.



Similar to the Institute account the admin user can edit previously created University details. Additionally, you can now see a bright-red garbage can icon on the bottom right. This link will delete this current University from the database along with any comments posted on it's page. Be wary not to click it unless you are sure you want to delete it.

5.0 API SPECIFICATION

5.1 User Authentication

5.1.1 Register a User

Description:

Register a new student user, institute, or administration. To specify user type set “isAdmin” to **true** for administration, “isInstitution” to **true** for institution, and set both to **false** or leave the field blank for student

URL: http://localhost:5000/api/auth/register
Method: **Post**
Body:

```
{
        username: string #required
        email: string #required
        password: string #required
        isAdmin: boolean
        isInstitution: boolean
}
```

Example:

```
POST /api/auth/register
Content-Type: application/json
{
        “username”: “paul”,
        “email”: “paul@test.com”,
        “password”: “psswrd",
        “isAdmin”: false,
        “isInstitution”: false
}
```

5.1.2 Login a User

Description:
For any type of user provide the “username” and “password” in the body of the request.
URL: http://localhost:5000/api/auth/login
Method: **Post**
Body:
{
username: string **#required**
password: string **#required**
}

Example:
POST /api/auth/login
Content-Type: application/json
{
“username”: “test”,
“password”: “test"
}

5.2 Filter Institutions

5.2.1 Search

Description:
To find a list of institutions that cohere to a search profile provide, as a query, the parameters that match the profile. All parameters are required.
URL: http://localhost:5000/api/search/
Method: **Get**
Query:
{
PriLang: string **#required** (primary language spoken)
Location: string **#required**
Dom_Frgn_Ratio: string **#required** (min domestic to foreign ratio)
Program: string **#required**
FTutition: string **#required** (max foreign tuition)
DTutition: string **#required** (max domestic tuition)
}

Example:
GET /api/search?PriLang=English&Location=United+States&
Dom_Frgn_Ratio=40&Program=Math&
FTutition=45000&DTutition=20000

5.3 University Pages

5.3.1 Create University Page

Description:
Provide the parameters of a University in the body of a request to create a new page.
URL: http://localhost:5000/api/uniPages/
Method: **Post**
Body:
{
Rank: string
Uname: string **#required #unique** (University Name)
Prog_Offered: [string **#required**] (List of programs offered)
Dom_Frgn_Ratio: string (min domestic to foreign ratio)
PriLang: string **#required** (primary language spoken)
Location: string **#required**
FTutition_Range: Number (max foreign tuition)
DTutition_Range: Number (max domestic tuition)
Website: string (Institute URL)
Type: string ([Verified | Unverified] verified if institute runs the page)
}

Example:
POST /api/uniPages
Content-Type: application/json
{
"Uname": "Borkley University",
"Prog_Offered": [
"Math",
"Science",
"Business"
],
"PriLang": "English",
"Location": "Canada, Manitoba, Winnipeg",
"Website": "https://www.borku.ca/"
"Type": "Unverified"

}

5.3.2 Sorted List of All Universities

Description:
Retrieve the “Uname”, “Rank”, and “Location” of all Universities in the database sorted by Rank in ascending order.
URL: http://localhost:5000/api/uniPages/
Method: **Get**

Example:
GET /api/uniPages/

5.3.3 Single University

Description:
Retrieve a single University by the unique “id” that is attached to the end of the url when navigated to a specific University’s page.
URL: http://localhost:5000/api/uniPages/:id
Method: **Get**

Example:
GET /api/uniPages/61c451c40bab6fbf037a7b97

5.3.4 Update University

Description:
Update a University’s page with the University “id” that is attached to the end of the url when navigated to a specific University’s page and the new parameter values provided in the body of the request. All parameters are optional
URL: http://localhost:5000/api/uniPages/:id
Method: **Put**
Body:
{

Rank: string	
Uname: string **#unique**	(University Name)
Prog_Offered: [string]	(List of programs offered)
Dom_Frgn_Ratio: string	(min domestic to foreign ratio)
PriLang: string	(primary language spoken)
Location: string	
FTuition_Range: Number	(max foreign tuition)

DTuition_Range: Number (max domestic tuition)
Website: string (Institute URL)
Type: string ([Verified | Unverified] verified if institute runs the page)
}

Example:
PUT /api/uniPages/61c451c40bab6fbf037a7b97
Content-Type: application/json
{
"Rank": 944,
"FTuition_Range": 6000,
"Type": "Verified"
}

5.3.5 Post Comment

Description:
Publishing a comment on a University's page from a specific user.
URL: http://localhost:5000/api/uniPages/comment
Method: **Post**
Body:
{
UniID: string **#required**
Username: string **#required**
Description: string **#required**
}

Example:
POST /api/uniPages/comment
Content-Type: application/json
{
"UniID": "61c4426691561101f1a334e9",
"Username": "Lisa",
"Description": "Here is a comment by Lisa."
}

5.3.6 Universities Comments

Description:
Retrieve all comments from a specific University using the University “id” that is attached to the end of the url when navigated to a specific University’s page.
URL: http://localhost:5000/api/uniPages/:id/comment
Method: **Get**

Example:
GET /api/uniPages/61c4424f91561101f1a334e7/comment

5.3.7 Delete Comment

Description:
Delete all comments from a specific University using the University “id” that is attached to the end of the url when navigated to a specific University’s page.
URL: http://localhost:5000/api/uniPages/:id/comment
Method: **Delete**

Example:
DELETE /api/uniPages/61c4424f91561101f1a334e7/comment

5.3.7 Delete University

Description:
Delete a specific University using the University “id” that is attached to the end of the url when navigated to a specific University’s page.
URL: http://localhost:5000/api/uniPages/:id
Method: **Delete**

Example:
DELETE /api/uniPages/61c4424f91561101f1a334e7

5.4 Watchlist

5.4.1 User's Watchlist

Description:
Collect the name, rank, location, and ID of all Universities, on a user's watchlist, in sorted order by rank.
URL: http://localhost:5000/api/watchlist
Method: **Get**
Query:
{
username: string **#required**
}

Example:
GET /api/watchlist?username=test

5.4.2 Save to Watchlist

Description:
Saving a University to a user's watchlist.
URL: http://localhost:5000/api/watchlist
Method: **Post**
Body:
{
username: string **#required**
Rank: Number
Uname: string **#required**
Location: string **#required**
UniID: string **#required**
}

Example:
POST /api/watchlist
Content-Type: application/json
{
"username": "test"
"Uname": "University of Calgary",
"Location": "Canada",
"UniID": "61c4421591561101f1a334e4"
}

5.4.3 Check Watchlist

Description:
Check if a University is already in a user's watchlist.
URL: http://localhost:5000/api/watchlist/alreadySaved
Method: **Get**
Query:
{
username: string **#required**
UniID: string **#required**
}

Example:
GET /api/watchlist?username=test&UniID=61c4e1c176d5a24e6f60657d

5.4.4 Delete from Watchlist

Description:
Delete a University from a user's watchlist.
URL: http://localhost:5000/api/watchlist/alreadySaved
Method: **Delete**
Query:
{
username: string **#required**
UniID: string **#required**
}

Example:
DELETE /api/watchlist?username=test&UniID=61c4e1c176d5a24e6f60657d

5.4.5 Update Watchlists

Description:
Update a University on all user's watchlists.
URL: http://localhost:5000/api/watchlist/:id
Method: **Put**

Example:
PUT /api/watchlist/61c4426691561101f1a334e9

5.4.6 Remove from All Watchlists

Description:
Deleting a University from all user's watchlists. Used for cleanup when deleting a University.
URL: http://localhost:5000/api/watchlist/:id
Method: **Put**

Example:
PUT /api/watchlist/61c4426691561101f1a334e9

References

[1] Government of Canada (2020), *Economic impact of international education in Canada - 2020 update.* Retrieved from Government of Canada website https://www.international.gc.ca/education/report-rapport/impact 2018/index.aspx?lang=eng

[2] Studyportals University search engine website https://www.mastersportal.com/

[3] QS World University Rankings 2021 https://www.topuniversities.com/university-rankings/world-university rankings/2021