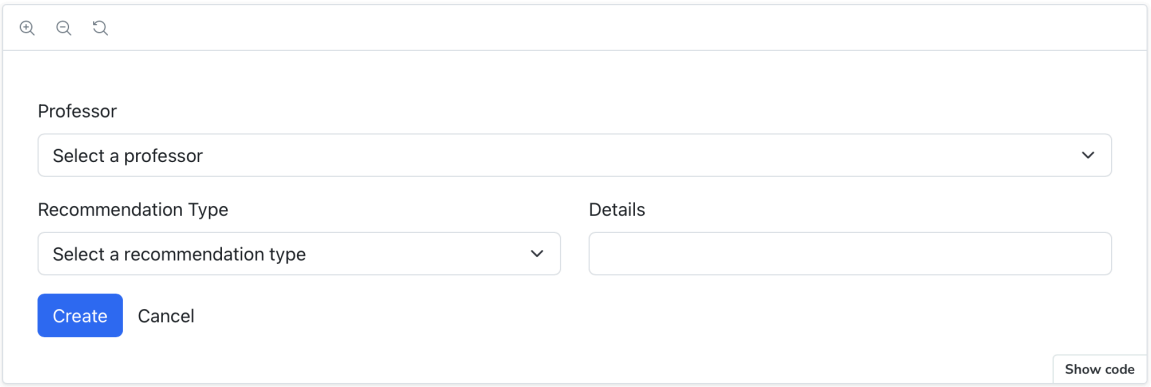
Features Already Implemented

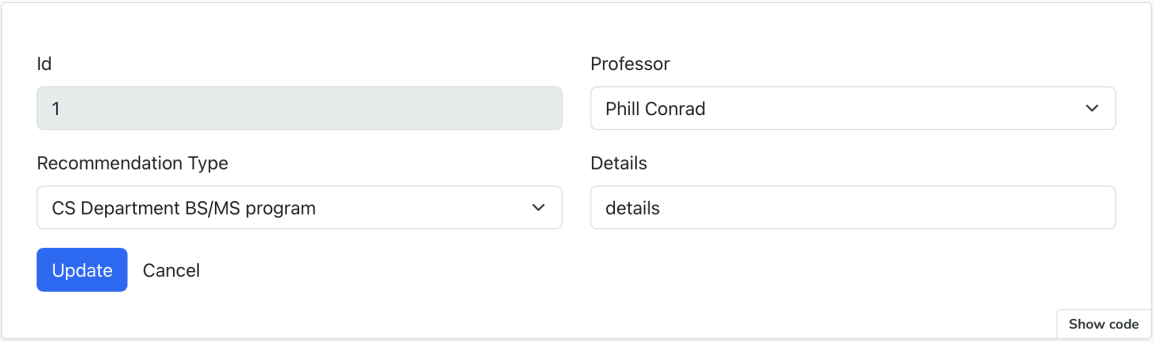
Frontend

Form Component:

- In both form views, the professor and recommendation type dropdowns are populated with the results of api queries (/api/admin/users/professors and /api/requesttypes/all) using a useEffect hook. Users must select a professor and recommendation type or else errors will be thrown on the frontend - if a user would like to specify a recommendation type that is not in the RequestType database, they can select the Other option and specify their new recommendation type in the details section (not required). This form is meant for people who are making requests, not professors who need to change the status of the request.
- Create:



-
- Edit:



-

Table Component:

User:

id	Professor Name	Professor Email	Requester Name	Requester Email	Recommendation Type	Details	Status	Submission Date	Last Modified Date	Completion Date	Due Date	Delete	Edit
2	Chaewon Bang	chaewonbang@ucsb.edu	Mia Scott	mscott@ucsb.edu	CS Department BS/MS program	lots of details	COMPLETED	2022-01-02T12:00	2022-06-02T12:00	2022-06-02T12:00	2022-09-02T12:00	Delete	Edit
3	Lithika Anabarasan	lithika@ucsb.edu	Shrena Punj	shrenapunj@ucsb.edu	CS Department BS/MS program	test details	PENDING	2022-02-02T12:00	2022-02-02T12:00		2022-10-02T12:00	Delete	Edit
4	Riya Seghal	riya@ucsb.edu	Krish Seghal	krishseghal@ucsb.edu	CS Department BS/MS program	test details	COMPLETED	2022-02-02T12:00	2022-02-02T12:00	2022-02-03T12:00	2022-04-02T12:00	Delete	Edit

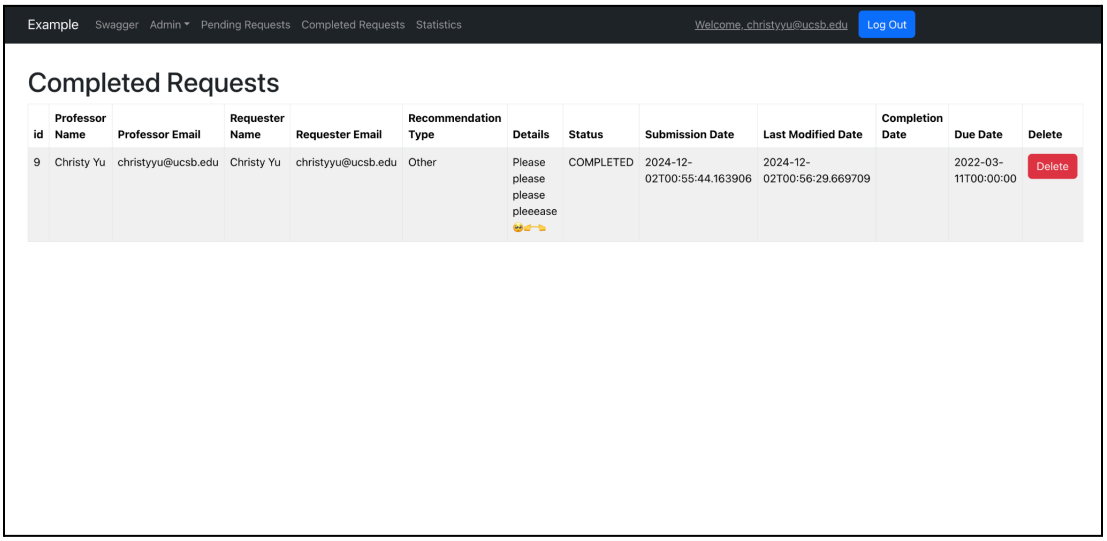
Admin:

id	Professor Name	Professor Email	Requester Name	Requester Email	Recommendation Type	Details	Status	Submission Date	Last Modified Date	Completion Date	Due Date	Delete
2	Chaewon Bang	chaewonbang@ucsb.edu	Mia Scott	mscott@ucsb.edu	CS Department BS/MS program	lots of details	COMPLETED	2022-01-02T12:00	2022-06-02T12:00	2022-06-02T12:00	2022-09-02T12:00	Delete
3	Lithika Anabarasan	lithika@ucsb.edu	Shrena Punj	shrenapunj@ucsb.edu	CS Department BS/MS program	test details	PENDING	2022-02-02T12:00	2022-02-02T12:00		2022-10-02T12:00	Delete
4	Riya Seghal	riya@ucsb.edu	Krish Seghal	krishseghal@ucsb.edu	CS Department BS/MS program	test details	COMPLETED	2022-02-02T12:00	2022-02-02T12:00	2022-02-03T12:00	2022-04-02T12:00	Delete

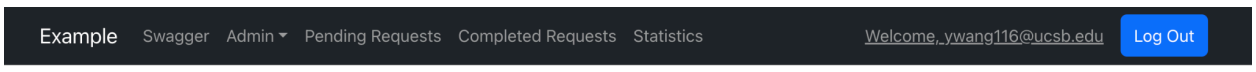
- The table component has been implemented for admins and students. The table for admins shows all the recommendation requests (and their data) that have been made. Based on the implementation for the CRUD operations, the delete button is accessible for admins. The table for students shows all the recommendation requests (and their data) that the student logged in has made. Based on the implementation for the CRUD operations, the delete button and edit button are accessible for students. If the edit page is implemented, the edit button should lead to the edit page where students can only update the details of their request.

Pages:

- *Completed Requests Page*
  - This allows both professors and students to see the completed recommendation requests in a concise, easy-to-parse table
  - For professors, this shows them the recommendation requests that they have either completed or denied
  - For students, this shows them the recommendation requests that they have sent, in which the receiving professors either completed or denied them
  -



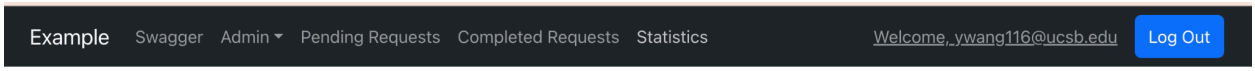
- *Pending Page*



Pending Requests page not yet implemented

  - This is a placeholder for the pending requests page. After logging in, all admin users should see this “Pending Requests” menu item in the navigation bar.

  - This is an intermediate step toward showing all pending or accepted requests associated with the user's account.
- *Statistics Page*



Statistics page not yet implemented

  - This is a placeholder for the statistics page. After logging in, all admin users should see this "Statistics" menu item in the navigation bar.
  - This is an intermediate step toward showing the number of pending and completed requests associated with the user's account.

Admin Users Page

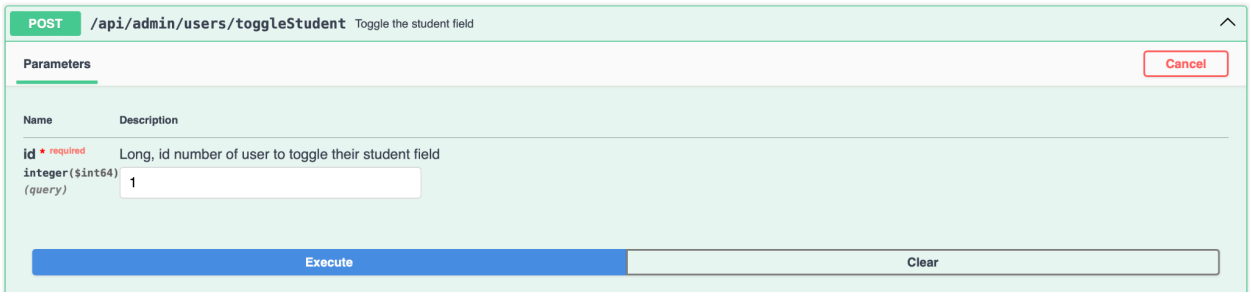
Users

id	First Name	Last Name	Email	Admin	Professor	Student	Toggle Admin	Toggle Professor	Toggle Student
6	[redacted]			false	false	false	Toggle Admin	Toggle Professor	Toggle Student
4	[redacted]			true	true	false	Toggle Admin	Toggle Professor	Toggle Student
2	[redacted]			true	true	false	Toggle Admin	Toggle Professor	Toggle Student
5	[redacted]			true	false	true	Toggle Admin	Toggle Professor	Toggle Student
1	[redacted]			true	false	false	Toggle Admin	Toggle Professor	Toggle Student
3	[redacted]			false	false	true	Toggle Admin	Toggle Professor	Toggle Student

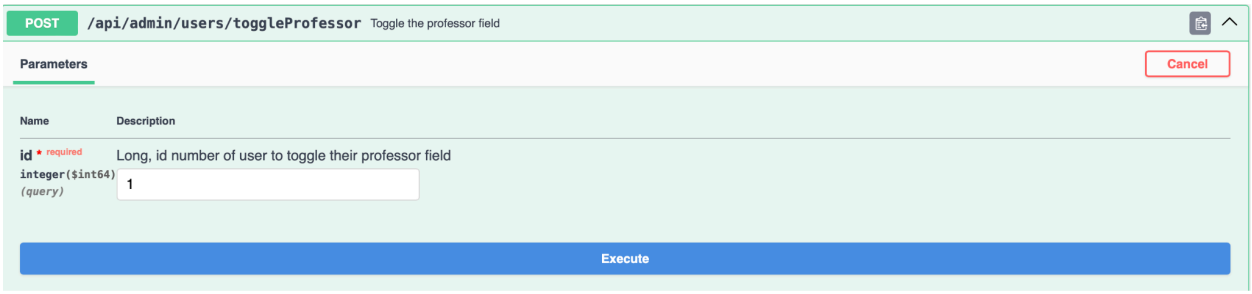
  - If a user is logged in as an administrator, they can navigate to the Admin > Users page to see a list of all the application users.
  - From here, they can see the User ID, First Name, Last Name, and Email. Additionally, they can see if each user is an Admin, Professor, or Student; which are each represented as Boolean values.
  - Additionally, Admins can click the "Toggle" buttons to switch the Boolean values and change the roles, and thus the permissions, of the Users based on if they are an Admin, Student, or Professor.

Backend

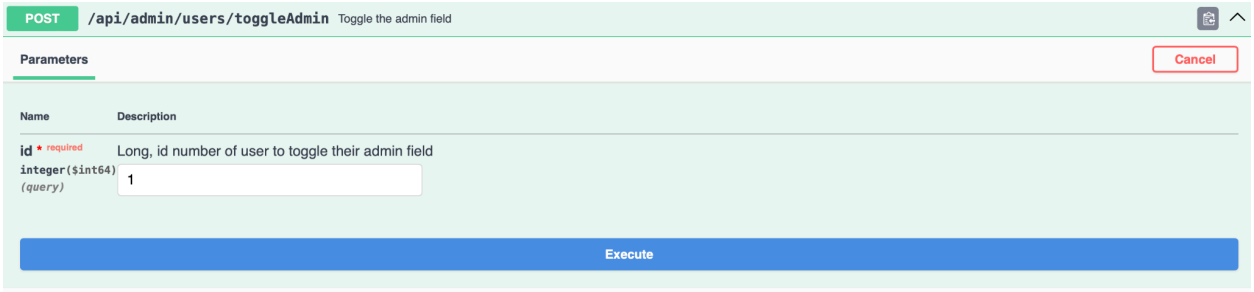
User Information (The following can only be executed by those with Admin Status except for the /professors route):
POST:



- Takes in a User ID. Toggles the Student role between “True” and “False” for the user the ID corresponds to.

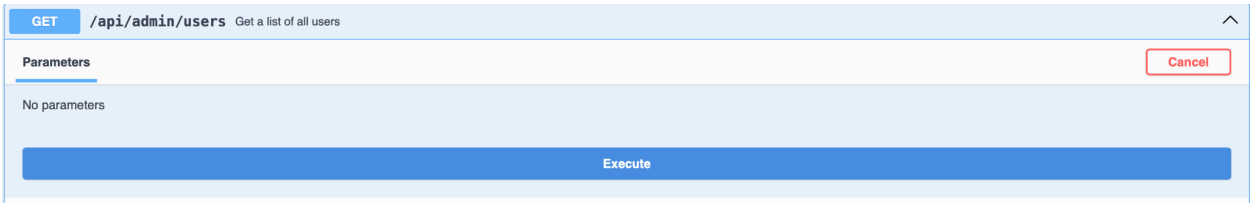


- Takes in a User ID. Toggles the Professor role between “True” and “False” for the user the ID corresponds to.

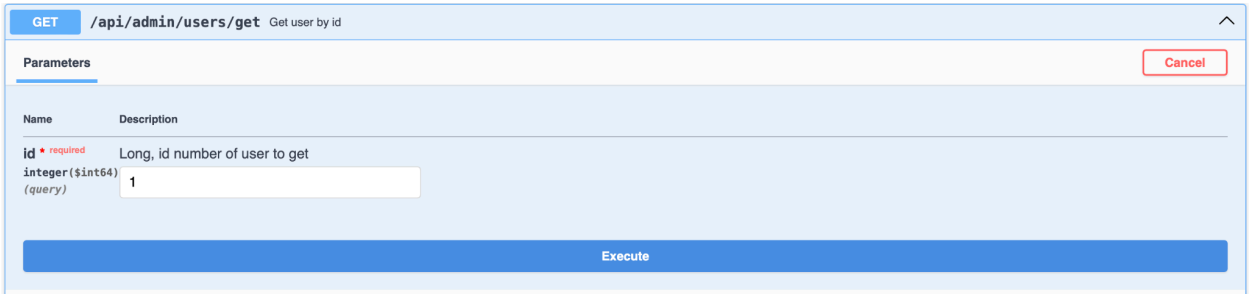


- Takes in a User ID. Toggles the Admin role between “True” and “False” for the user the ID corresponds to.

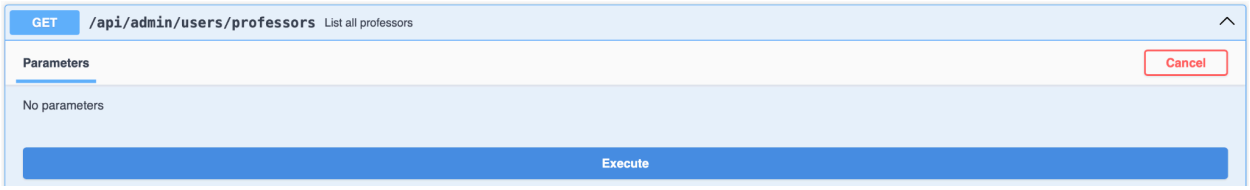
GET:



- Takes in no parameters. Returns a list of all the users to the admin.

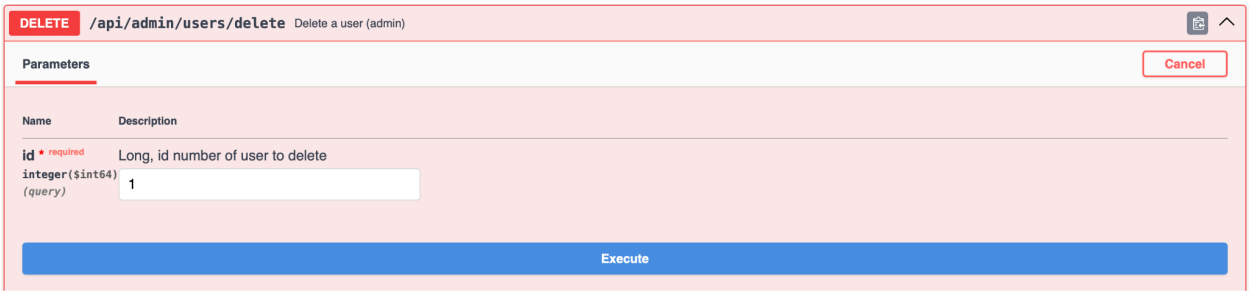


- Takes in a User ID. Returns the user information of the User that the ID corresponds to.



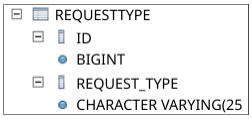
- Takes no parameters, returns simplified json versions (full names and ids, no other user details) of all Users with professor == true. This is the only Users route that a User does not need to be an admin to use; it is used within the form dropdown to present the list of available professors.

DELETE:

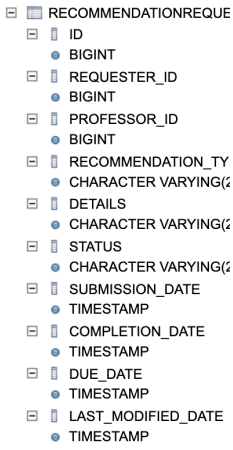


- Take in a User ID. Deletes the User that the ID corresponds to.

Tables:

- RequestType
  - 

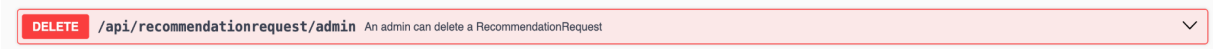
  - ID is automatically generated on creation of a new RequestType
  - REQUEST_TYPE should describe a way in which to categorize new RecommendationRequests
- RecommendationRequest

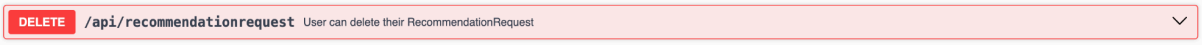


- requester_id and professor_id are foreign keys connected to the Users table, both with Many-to-One relationships
- submission_date and last_modified_date are automatically managed in the backend via entity listeners and JPA auditing with a DateTime provider defined in the ExampleApplication.java.

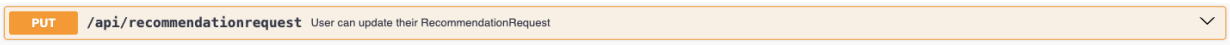
Recommendation Request CRUD:

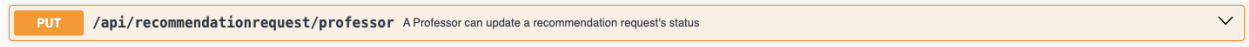
- DELETE





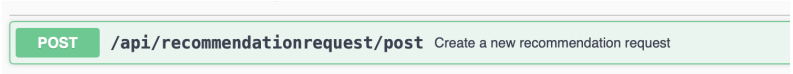
- The DELETE operation was implemented for admins and students (students are users who only have the role ROLE_USER). The operation takes in the ID for the recommendation request that needs to be deleted. To ensure that students can only delete their own requests, the operation checks to see if the ID and the requester matches the original request. If the original requester and the requester attempting to delete the request do not match, the operation throws the exception, EntityNotFoundException. This adds an extra layer of security to the delete operation since students can already only see all their requests through their GET operations. Admins can delete any existing recommendation request with the ID of the recommendation request.
- PUT

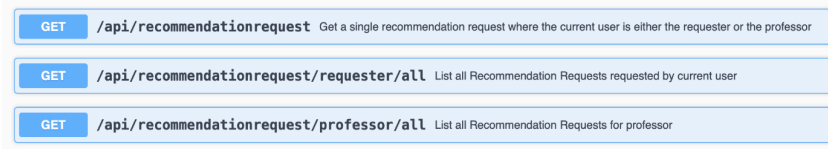


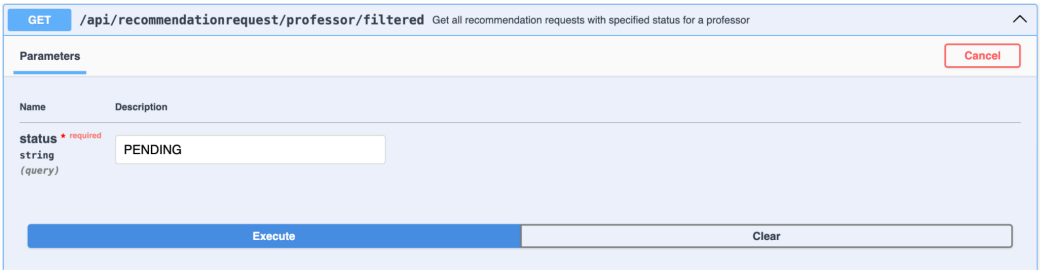


- The PUT operation was implemented for professors and students. Professors can only update the status of a recommendation request while students can only update the details of their recommendation request. The PUT operation takes in the ID for the recommendation request that needs to be updated. Similar to the DELETE operation, to ensure that students can only update their own request, the operation checks to see if the ID and the requester matches the original request. If the original requester and the requester attempting to update the request do not match, the operation throws the exception, EntityNotFoundException. A professor is able to update the status of an existing

recommendation request with the ID. Professors are only able to view (and therefore update) recommendation requests addressed to them through the GET operation for professors.

- POST
  - 

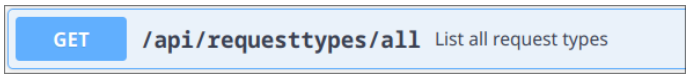
  - The POST endpoint takes in a professorId, recommendationType, details, and an ISO dueDate. Both the professorId and recommendationType are checked against the User and RequestType database to make sure they are valid selections (unless recommendationType is “Other”, in which case users enter into details a special type of recommendation request); they each throw an EntityNotFoundException if not found. If both are valid, they are set in a new RecommendationRequest Entity along with dueDate and details; additionally, the backend automatically sets the status to “PENDING”, sets lastModifiedDate and submissionDate, and saves the new object to the repository.
- GET
  - 

  - /api/recommendationrequest: this route gets a single request with a given id - however, the recommendation request will only be returned to the user if the user is either the professor or requester. If either the id does not exist in the RecommendationRequest table or the current user is neither the professor or requester, an EntityNotFoundException is thrown. All users can access this route.
  - /api/recommendationrequest/requester/all: this route retrieves all recommendation requests where the current user is the requester. All users can access this route.
  - /api/recommendationrequest/professor/all: this route retrieves all recommendation requests where the current user is the professor listed on the request. Only users with the role of professor can access this route.
  - 

    - /api/recommendationrequest/professor/filtered: this route takes in status as the parameter and returns all recommendation requests of the specified status where the current user is the professor listed on the request. Only users with the role of professor can access this route.

Request Types CRUD:
The RequestType entity allows users to categorize each recommendation request (e.g. Scholarship, PhD Program, etc.). Instructors and admins can add/edit/delete request types as necessary. Each RequestType entity has 2 fields: An autogenerated ID, and a requestType value stored as a string.

- **GET**



  - /api/requesttypes/: This route retrieves a single RequestType with a given ID. If either the id does not exist in the RequestType table, an EntityNotFoundException is thrown. All users can access this route.
  - /api/requesttypes/all: This route retrieves all RequestTypes from the table. All users can access this route.

- **POST**

  - /api/requesttypes/post: The POST operation creates a new RequestType that will then be populated in the fronted dropdown menu. It will takes single input field: requestType. This field should be a string that clearly describes how to categorize Recommendation Requests. If the RequestType is a duplicate of an already existing type, an IllegalArgumentException will be thrown. Only users with the role admin can access this route.

- **PUT**

  - /api/requesttypes: The PUT operation takes in the ID of an existing RequestType and allows the requestType field to be edited. In other words, change the “name” of a RequestType. If the ID does not exist, an EntityNotFoundException will be thrown. If the new RequestType is a duplicate of an already existing type, an IllegalArgumentException will be thrown. Only users with the role admin can access this route.

- **DELETE**

  - /api/requesttypes: The DELETE operation allows a single RequestType to be removed by taking the ID as an input. If the ID is valid, then the associated RequestType will be deleted from the table. If the ID does not exist, an EntityNotFoundException will be thrown. Only users with the role admin can access this route.

Features Not Yet Implemented/Needs To Be Implemented

Frontend

Request Types CRUD

- Admin or instructor can manage new request types for the request dropdown, page where new requests are created should have a dropdown menu with a variety of request types which comes from request type table

Pending Requests Page

- Placeholder created, needs actual implementation

Statistics Page

- Placeholder created, needs actual implementation

Index Page for Recommendation Request

- A PR was created for the index page, but due to some changes made to the recommendation request controller the index page was not ready to be merged. The index page does not currently differentiate roles (professor, admin, and student) and returns the proper index page based on that role (i.e the create button exists for all users regardless of their role).

Create Recommendation Request Page:

- PR was submitted but major updates on the controller side forced it to be thrown to the wayside - could be reworked to fit the codebase and would be usable pretty easily

Edit Recommendation Request Page:

- PR was submitted but major updates on the controller side forced it to be thrown to the wayside - could be reworked to fit the codebase and would be usable pretty easily

Backend

Table Changes:

- recommendationType inside the RecommendationRequest table should optimally be a foreign key to the RequestType table, and Other should be an entry in the RequestType table