

# S25-07 Frontiers

**Repo Link:** <https://github.com/ucsb-cs156-s25/proj-frontiers-s25-07>

**Members:** Annika Damstedt, Shelly Zhu, Hannah Zhang, Forrest Zhou, Steven Liu, Hannya Yan

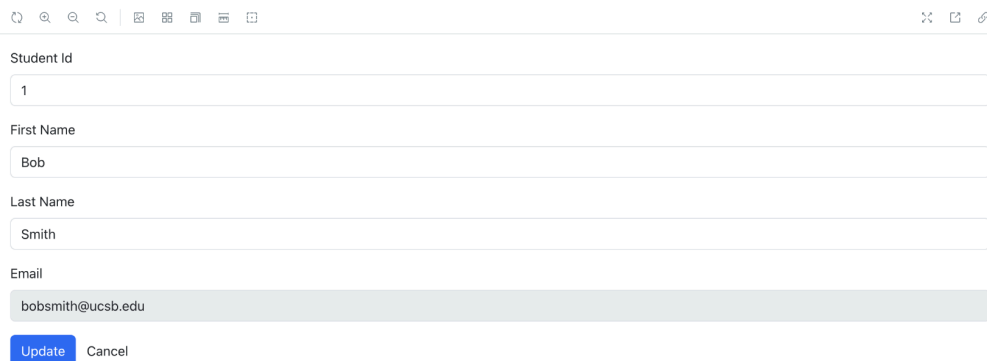
## Frontend Changes

1. RosterStudent form and table have been set up and are deployed on Storybook.

The form component for roster students can be used for Create and Edit functions. It has the following String fields to enter: studentId, firstName, lastName, and email. This form can be used in a future PR to add an index page for roster students.



This screenshot shows the 'Create' form for a new student. It features four text input fields labeled 'Student Id', 'First Name', 'Last Name', and 'Email'. At the bottom left, there is a blue 'Create' button, and at the bottom right, there is a 'Cancel' button. The form is displayed within a Storybook interface, with navigation icons at the top and a toolbar at the bottom.



This screenshot shows the 'Update' form for an existing student. The input fields are pre-filled with the following values: 'Student Id' is '1', 'First Name' is 'Bob', 'Last Name' is 'Smith', and 'Email' is 'bobsmith@ucsb.edu'. At the bottom left, there is a blue 'Update' button, and at the bottom right, there is a 'Cancel' button. The form is displayed within a Storybook interface, with navigation icons at the top and a toolbar at the bottom.

The table component for roster students has buttons for Update and Delete. It can be used in a future PR to add an index page for roster students.

Student Id	First Name	Last Name	Email

Student Id	First Name	Last Name	Email
2	Alice	Brown	alicebrown@ucsb.edu
3	Tom	Hanks	tomhanks@ucsb.edu
4	Emma	Watson	emmawatson@ucsb.edu

Student Id	First Name	Last Name	Email	Edit	Delete
2	Alice	Brown	alicebrown@ucsb.edu	<div>Edit</div>	<div>Delete</div>
3	Tom	Hanks	tomhanks@ucsb.edu	<div>Edit</div>	<div>Delete</div>
4	Emma	Watson	emmawatson@ucsb.edu	<div>Edit</div>	<div>Delete</div>

2. Basic email form and email table are set up and deployed on Storybook, which can be used for instructors and admins. The email form contains a regular expression check to see if the email entered is valid.

Email

Create

Cancel

Email

test

A valid email is required.

Create

Cancel

Email

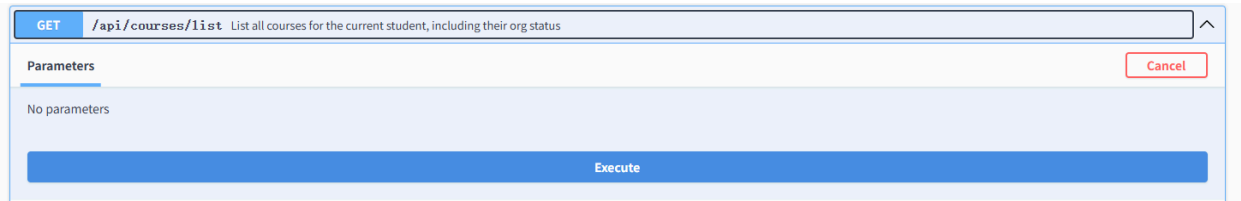
hannahzhang@ucsb.edu

CreateCancel

Email	Delete
admin@example.com	Delete
user@example.org	Delete

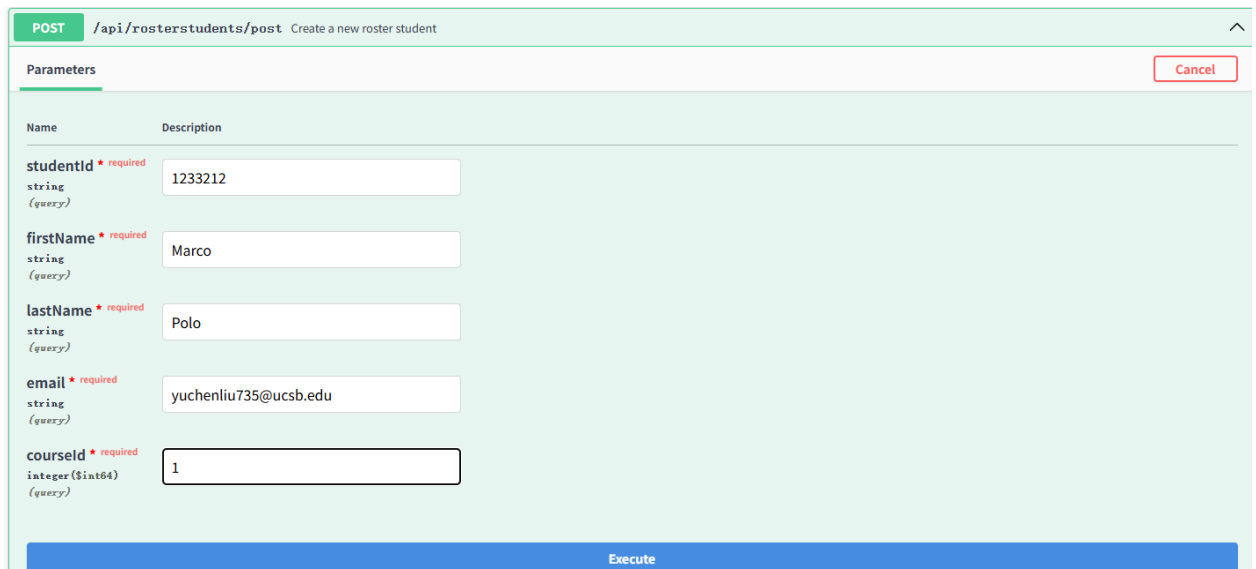
## Backend Changes

1. Created a GET endpoint for Course to list all the courses that the current user is enrolled in.



The screenshot shows a REST client interface for a GET endpoint. The top bar displays the method 'GET' and the path '/api/courses/list' with a description 'List all courses for the current student, including their org status'. Below this, a 'Parameters' section is shown with the text 'No parameters'. A blue 'Execute' button is at the bottom, and a red 'Cancel' button is in the top right corner.

- If you create a roster student enrolled in a course that exists using your own email that is used to log in the frontiers website:



The screenshot shows a REST client interface for a POST endpoint. The top bar displays the method 'POST' and the path '/api/rosterstudents/post' with a description 'Create a new roster student'. Below this, a 'Parameters' section contains a table of fields to be sent in the request body. The fields are: 'studentid' (string, required, value: 1233212), 'firstName' (string, required, value: Marco), 'lastName' (string, required, value: Polo), 'email' (string, required, value: yuchenliu735@ucsb.edu), and 'courseId' (integer (\$int64), required, value: 1). A blue 'Execute' button is at the bottom, and a red 'Cancel' button is in the top right corner.

Name	Description
<b>studentid</b> * required string (query)	1233212
<b>firstName</b> * required string (query)	Marco
<b>lastName</b> * required string (query)	Polo
<b>email</b> * required string (query)	yuchenliu735@ucsb.edu
<b>courseId</b> * required integer (\$int64) (query)	1

- By executing this list endpoint, you will be able to see a list of courses enrolled by the current user:

**GET** `/api/courses/list` List all courses for the current student, including their org status

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://frontiers.dokku-07.cs.ucsb.edu/api/courses/list' \
  -H 'accept: */*' \
  -H 'X-XSRF-TOKEN: ca2f32b0-f493-471f-b7a2-9191a169e1a3'
```

Request URL

```
https://frontiers.dokku-07.cs.ucsb.edu/api/courses/list
```

Server response

Code	Details
200	<p>Response body</p> <pre>[   {     "id": 1,     "installationId": "67002695",     "orgName": "hannahz0-testorg01",     "courseName": "PSTAI 3280",     "term": "S25",     "school": "UCSB",     "studentStatus": "NONE"   } ]</pre> <p><span>Download</span></p>

Response headers

2. Created the Instructors table (backend) and its corresponding POST, GET, DELETE operations.

Instructors			^
POST	/api/admin/instructors/post	Create a new Instructor	▼
GET	/api/admin/instructors/get	List all Instructors	▼
DELETE	/api/admin/instructors/delete	Delete an Instructor by email	▼

- There is a POST request to create a new instructor (with single field email)

**POST** /api/admin/instructors/post Create a new Instructor

Parameters Cancel

Name	Description
email * required string (query)	<input type="text" value="instructor@email.com"/>

Execute

**Responses**

Curl

```
curl -X 'POST' \
  'https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/post?email=instructor%40email.com' \
  -H 'accept: */*' \
  -H 'X-XSRF-TOKEN: 38d2ed2f-6fea-4aad-a9c7-e75274cb42b3' \
  -d ''
```

Request URL

```
https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/post?email=instructor%40email.com
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "email": "instructor@email.com" }</pre> <span>Download</span>

- There is a GET request to get all instructors

**Responses**

Curl

```
curl -X 'GET' \
  'https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/get' \
  -H 'accept: */*' \
  -H 'X-XSRF-TOKEN: 38d2ed2f-6fea-4aad-a9c7-e75274cb42b3'
```

Request URL

```
https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/get
```

Server response

Code	Details
200	<p>Response body</p> <pre>[   {     "email": "instructor@email.com"   } ]</pre> <span>Download</span>

- There is a DELETE request where you can specify an email and it will delete that

**DELETE**
/api/admin/instructors/delete
Delete an Instructor by email

Parameters
Cancel

Name	Description
<b>email</b> * required string (query)	<input type="text" value="instructor@email.com"/>

Execute
Clear

Responses

Curl

```
curl -X 'DELETE' \
  'https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/delete?email=instructor%40email.com' \
  -H 'accept: */*' \
  -H 'X-SRP-TOKEN: 38d2ed2f-6fea-4aad-a9c7-e75274cb42b9'
```

Request URL

```
https://frontiers.dokku-07.cs.ucsb.edu/api/admin/instructors/delete?email=instructor%40email.com
```

Server response

Code	Details
200	Response body <pre>Instructor with email instructor@email.com deleted.</pre> Download

- Each of the methods above is available only to Admins

3. Created a webhook listener for GitHub organization membership events that allows the application to track when students are invited to or join a course's GitHub organization.

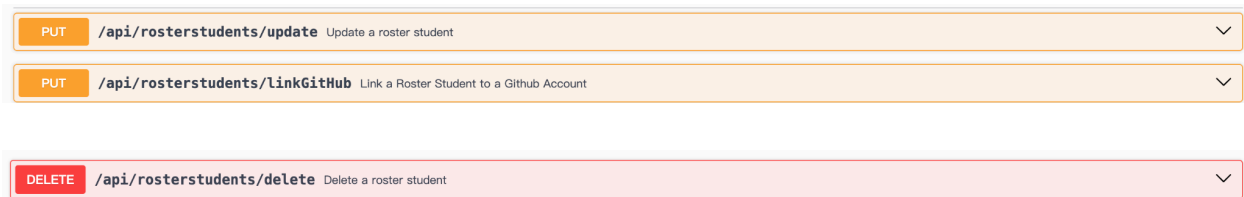
When a student is invited to an installed Github organization, their orgStatus will be changed to INVITED;

When a student successfully joins the Github organization, their orgStatus will be changed to MEMBER.

4. Created a PUT endpoint for roster students. PUT enables admins to update a student's first name, last name and student ID.

Created a PUT endpoint for roster students that links a rosterStudentId to their Github account.

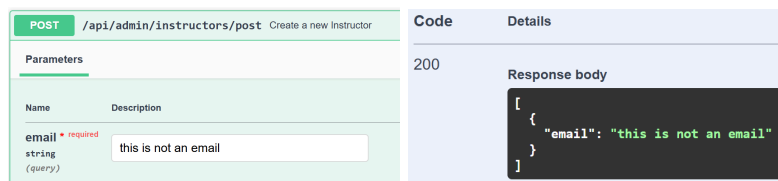
Created a DELETE endpoint to delete a student from the student roster. DELETE will also remove the student from the courses they're enrolled in.



Be aware that updating a roster student's email through PUT is yet to be implemented.

To update the email field of a roster student, so far we can only delete an existing student and add this student again with the updated email.

No email format checking extensions have been implemented yet. So if you try to POST something like "this is not an email", then it will be accepted by the system.



5. Admin backend has been set up (PUSH, GET all, and DELETE). This includes loading any emails in the ADMIN\_EMAILS variable into the admins table on start up.

DELETE only works for non ADMIN\_EMAILS emails.

