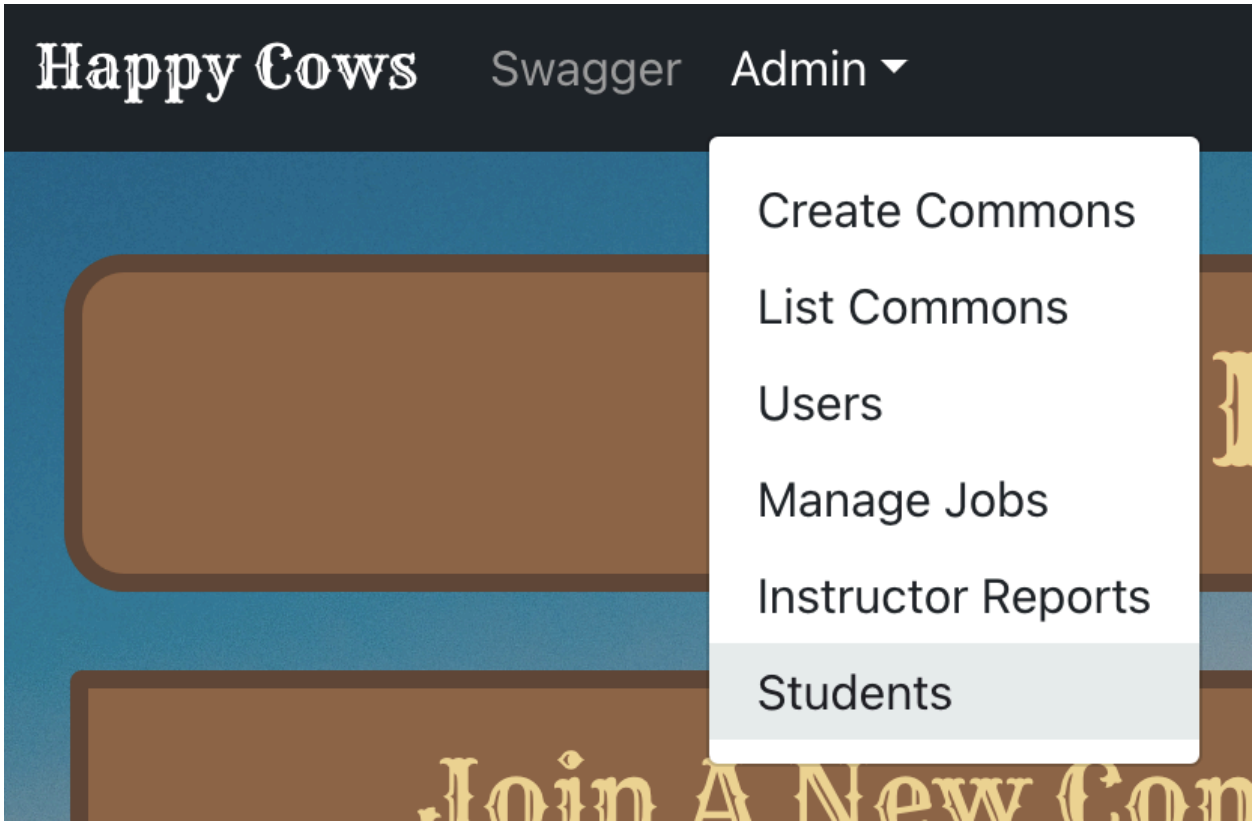
Release Notes

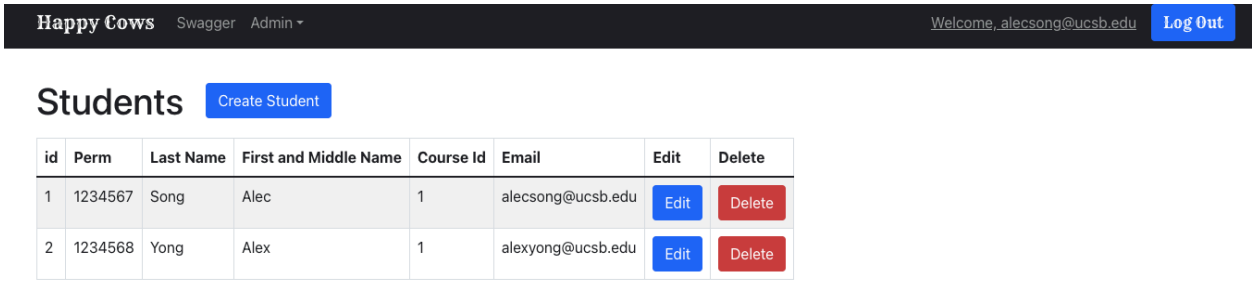
Deployed at https://happycows.dokku-09.cs.ucsb.edu

Students

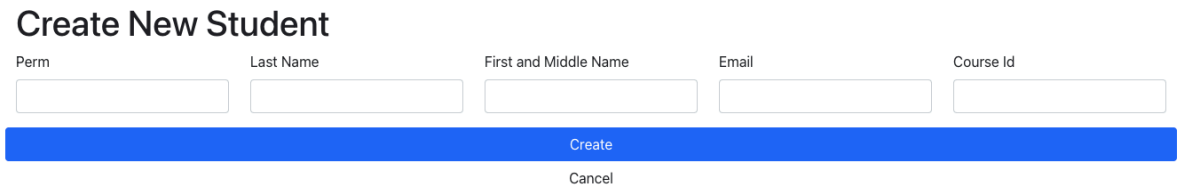
There is now a “Students” option within the admin Menu. To access the list of students, select “Students” from the admin menu, shown below.



Then, you’ll see a page like the one shown below, which shows the roster of Students and their corresponding info.

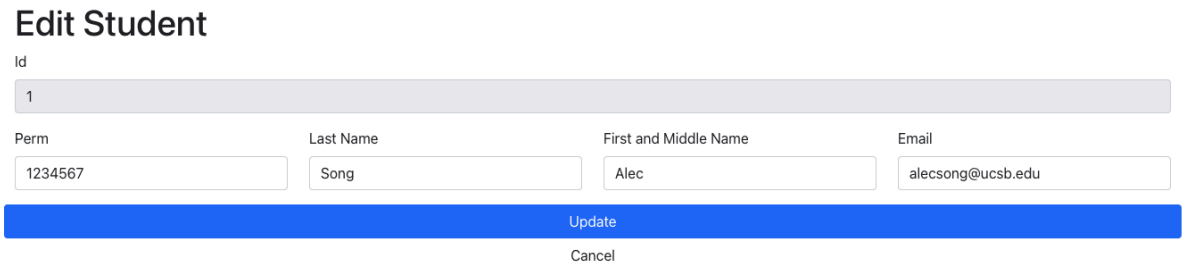


To create a new student, click the “Create Student” button and it should take you to the following page.



Input the data for your student and hit “Create”, and it should show up on the previous page.

You can also edit a student’s data by clicking the “Edit” button, which should take you to the following page.

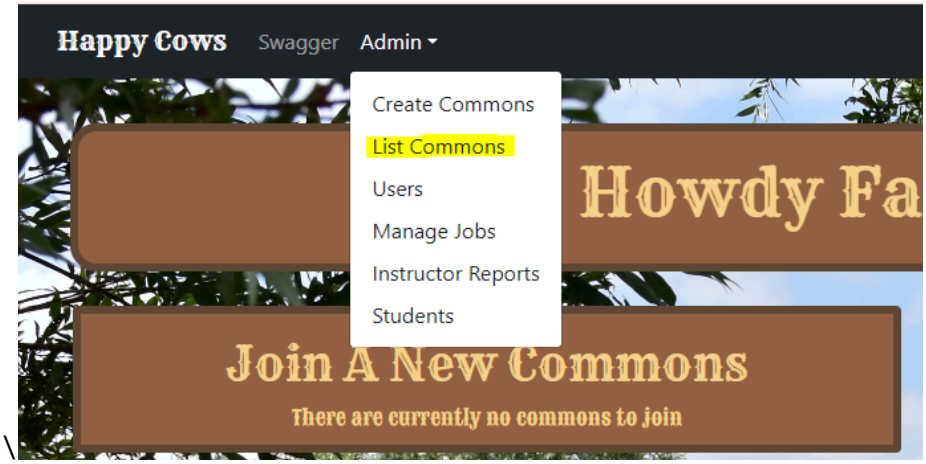


Note that you cannot edit the Course ID for a student, as rosters are class-specific; if you would like to edit this field, delete the student and re-create it with the proper Course ID. Click “Update” to confirm the edits to the students’ data.

You can also delete a student by simply clicking the “Delete” button, and the student should disappear from the roster.

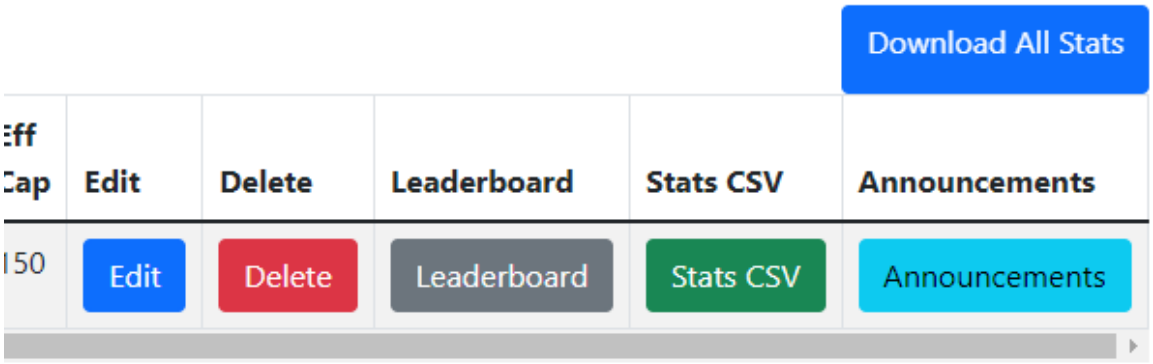
Announcements

1. There is now an “Announcements” page for each commons. To find this page, be sure to be logged in as an admin user, then click the “Admin” dropdown at the top and select “**List Commons**”. *It is also important to **have at least one commons** created already

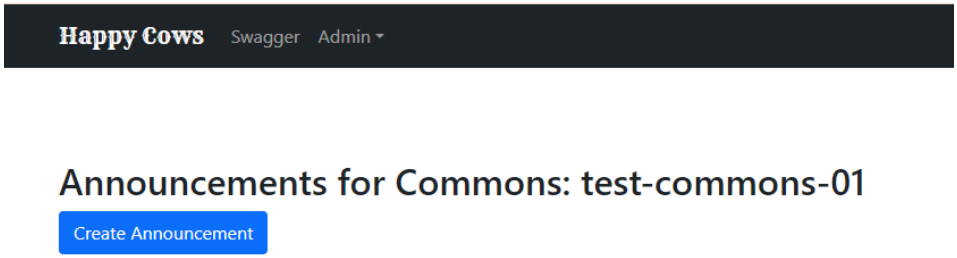


\

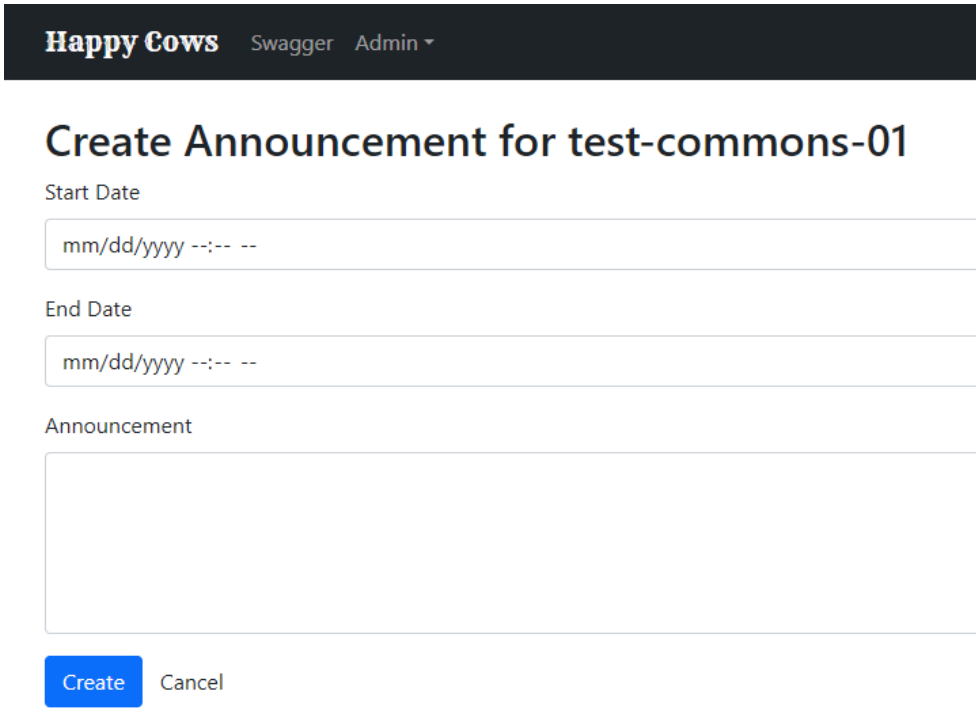
Then on the **very right** of the table, select the light blue “Announcements” button for the desired commons.



The announcements page will look like this, with a button to **create** a new announcement. **Coming soon** in the next release should be buttons to view, edit, and delete announcements.



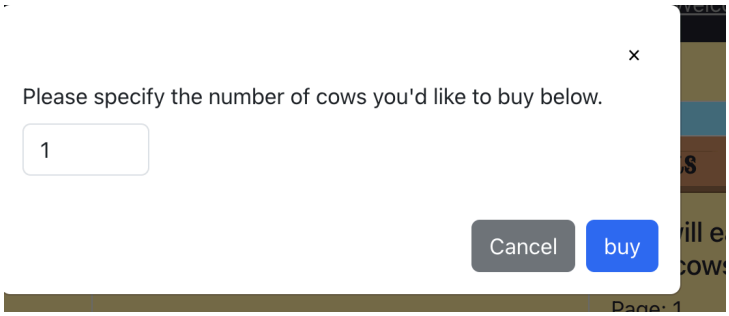
2. There is now a form to create a new announcement. This form allows for the user to select a start date, end date, and text for the actual announcement. To navigate to this page, click the “Create Announcement” button from the previous page (Announcements Page).

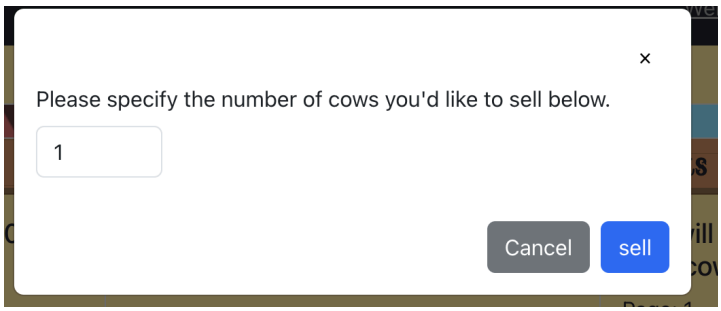


a) Start date is a required field indicating when the announcement should begin to be displayed.
b) End date is an optional field indicating when the announcement should stop being displayed. If left blank, the announcement will be displayed indefinitely. It is also a requirement that the end date is some date which is after the start date.
c) Announcement is a required field and should be filled in by the user/instructor with the actual text that should be displayed in the announcement.
d) Create is the button to press once you have filled out the form and are ready to submit. Upon pressing the button, you will be sent back to the home page and receive a confirmation notification at the top right corner.
e) Cancel is the button that you can press if you decide to not create an announcement. This will result in the navigation of the user back to the “Announcements” page for that specific commons.

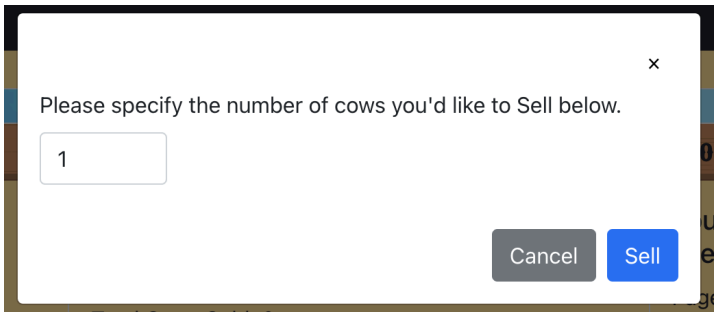
Bug Fixes

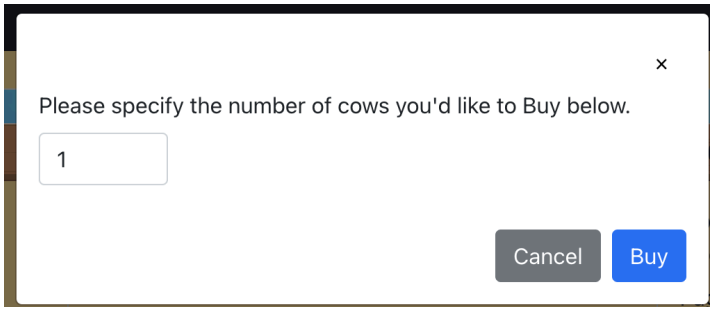
1. The inconsistent capitalization for ‘Buy’ and ‘sell’ is fixed.
   Before:





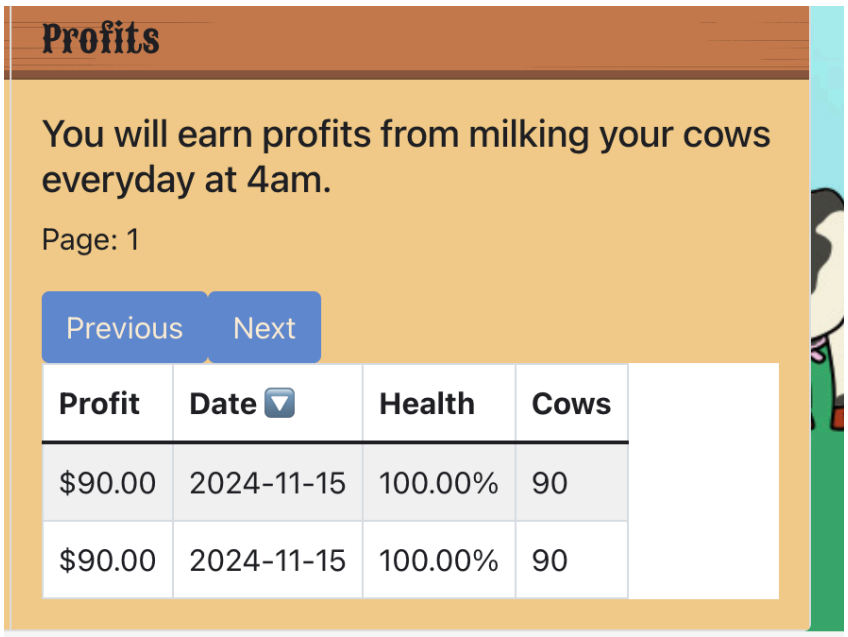
   After:



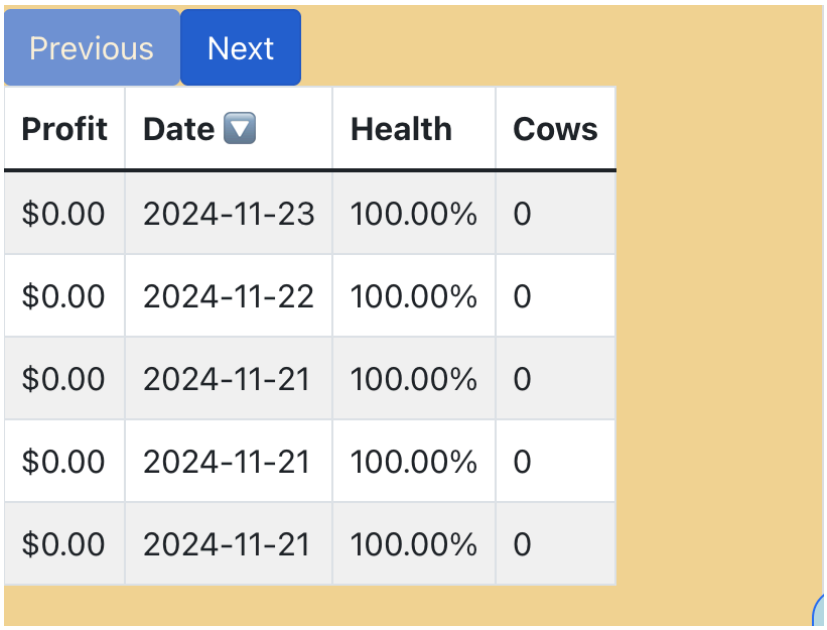


   Navigate to visit a common and click the visit button. There will be two buttons “Buy cows” and “Sell cows”. Click these two buttons to check the capitalization is consistent.

2. After navigating to the commons, the issue with the profits table containing white space instead of taking the whole frame is fixed.
   Before:



   After:



   Navigate to visit a common and click the visit button. There will be a Profits table on the right side of the page. Now, the table should take up the whole frame with no white space left.

3. After navigating to the Leaderboard page, the leaderboard table was incorrectly sorting the users by their total wealth.
   Before:

Farmer	Total Wealth ⬆️	Cows Owned	Cow Health	Cows Bought	Cows Sold	Cow Deaths
Alec Song	$1,100.00	89	100	100	11	0
Alec Song	$10,000.00	0	100	0	0	0
Alec Song	$100.00	99	100	99	0	0

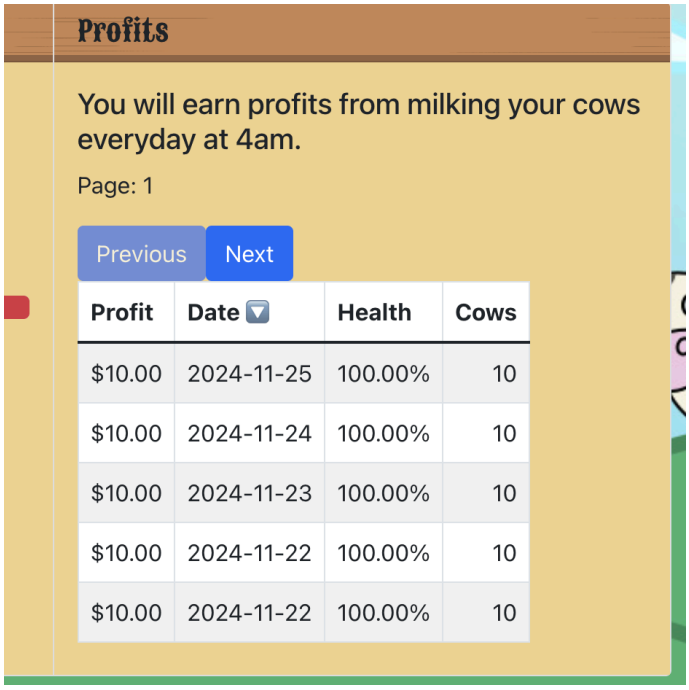
Farmer	Total Wealth ⬇️	Cows Owned	Cow Health	Cows Bought	Cows Sold	Cow Deaths
Alec Song	$100.00	99	100	99	0	0
Alec Song	$10,000.00	0	100	0	0	0
Alec Song	$1,100.00	89	100	100	11	0

   After:

Farmer	Total Wealth ⬇️	Cows Owned	Cow Health	Cows Bought	Cows Sold	Cow Deaths
Alec Song	$10,000.00	0	100	0	0	0
Alec Song	$1,100.00	89	100	100	11	0
Alec Song	$100.00	99	100	99	0	0

Farmer	Total Wealth ⬆️	Cows Owned	Cow Health	Cows Bought	Cows Sold	Cow Deaths
Alec Song	$100.00	99	100	99	0	0
Alec Song	$1,100.00	89	100	100	11	0
Alec Song	$10,000.00	0	100	0	0	0

4. The profit table in the commons page now has all of the numeric values to be right aligned rather than left-aligned to make it easier to read.



5. Fixed a bug when joining a commons so that it automatically populates to the right side underneath “Visit a Commons” without having to refresh

proj-happycows-f24-09

For Developers:

Bug Fixes:

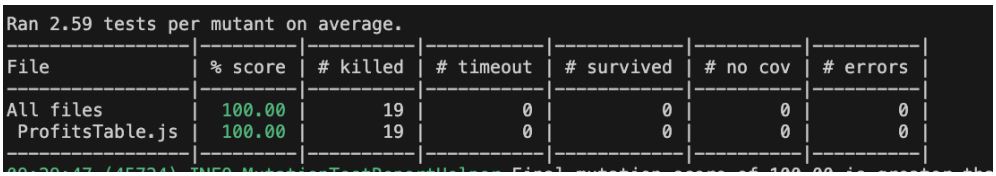
Footer link: Corrected footer link from the outdated link: https://devries.chem.ucsb.edu/mattanjah-de-vries to the proper https://devries.chem.ucsb.edu/about-5.

Manage cows notes: formatted the notes section of the “manage cows” with elements.
Before:

After:

ProfitsTable.js - killed existing mutations with new tests.

```
[Survived] ArrowFunction
src/main/components/Commons/ProfitsTable.js:11:27
-                   accessor: (row) => `$${row.amount.toFixed(2)}`,
+                   accessor: () => undefined,

[Survived] StringLiteral
src/main/components/Commons/ProfitsTable.js:11:36
-                   accessor: (row) => `$${row.amount.toFixed(2)}`,
+                   accessor: (row) => ``,

[Survived] StringLiteral
src/main/components/Commons/ProfitsTable.js:15:27
-                   accessor: "date",
+                   accessor: "",

[Survived] StringLiteral
src/main/components/Commons/ProfitsTable.js:19:36
-                   accessor: (row) => `${row.avgCowHealth + '%'}`
+                   accessor: (row) => ``

[Survived] StringLiteral
src/main/components/Commons/ProfitsTable.js:19:58
-                   accessor: (row) => `${row.avgCowHealth + '%'}`
+                   accessor: (row) => `${row.avgCowHealth + ""}`

[Survived] ArrowFunction
src/main/components/Commons/ProfitsTable.js:19:27
-                   accessor: (row) => `${row.avgCowHealth + '%'}`
+                   accessor: () => undefined

[Survived] StringLiteral
src/main/components/Commons/ProfitsTable.js:23:27
-                   accessor: "numCows",
+                   accessor: "",
```



```
Ran 2.59 tests per mutant on average.
-----------------|---------|----------|-----------|------------|---------|----------|
File             | % score | # killed | # timeout | # survived | # no cov | # errors |
-----------------|---------|----------|-----------|------------|---------|----------|
All files        |  100.00 |       19 |         0 |          0 |        0 |        0 |
 ProfitsTable.js |  100.00 |       19 |         0 |          0 |        0 |        0 |
-----------------|---------|----------|-----------|------------|---------|----------|
```

Students

1. Students.java - entity for the Students class
2. StudentsRepository.java - repository for Students class
3. StudentsController.java - handles URL mapping to corresponding features. Contains the functionality for GET (list all Students), POST (create a new Student), PUT (Edit an existing student), and DELETE (Delete an existing student).
4. StudentsControllerTests.java - ensures all methods within the Students controller are fully functional and have full test coverage.
5. StudentsForm.js - form for Students frontend implementation. Used within the frontend interface for creating a student, as it allows a Students' fields to be inputted.
6. studentsFixtures.js - example students for usage in testing.
7. StudentsTable.js - table displaying all students and their corresponding information.
8. StudentsIndexPage.js - frontend "landing" page for Students that displays the Students Table. Can edit, delete, and create a new student from this page.
9. StudentsEditPage.js - frontend component for editing a student. Utilizes StudentsEditForm.js instead of StudentsForm.js to disallow editing of the Course ID field.
10. StudentsCreatePage.js - frontend component for creating a new student. Utilizes StudentsForm.js.

Leaderboard

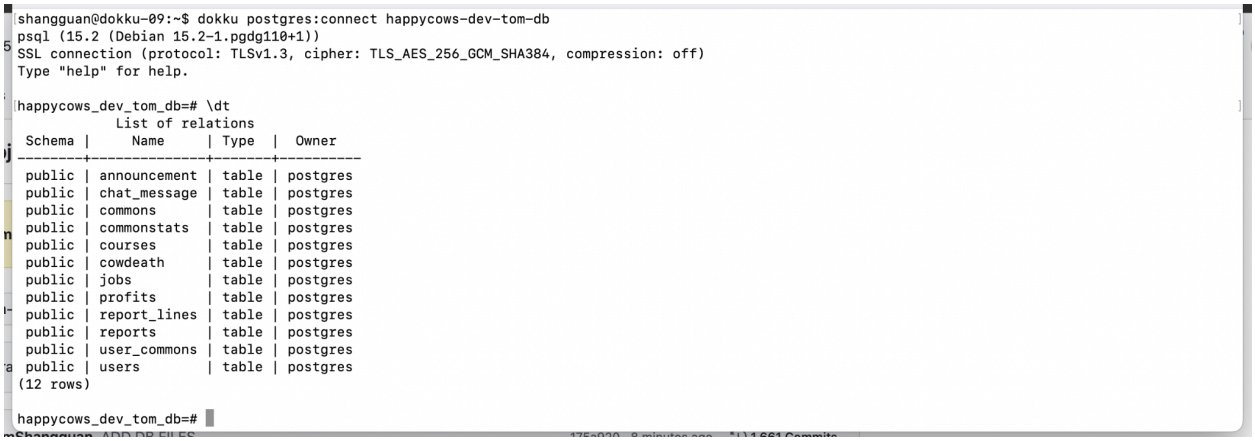
1. LeaderboardTable.js - corrected the code to sort by total wealth by formatting the value as USD (US Dollar). Added an accessor for “total wealth” so it can be sorted
2. LeaderboardTable.test.js - Added test to verify the unsorted, ascending, and descending options. 100% test coverage on LeaderboardTable.js

Announcements

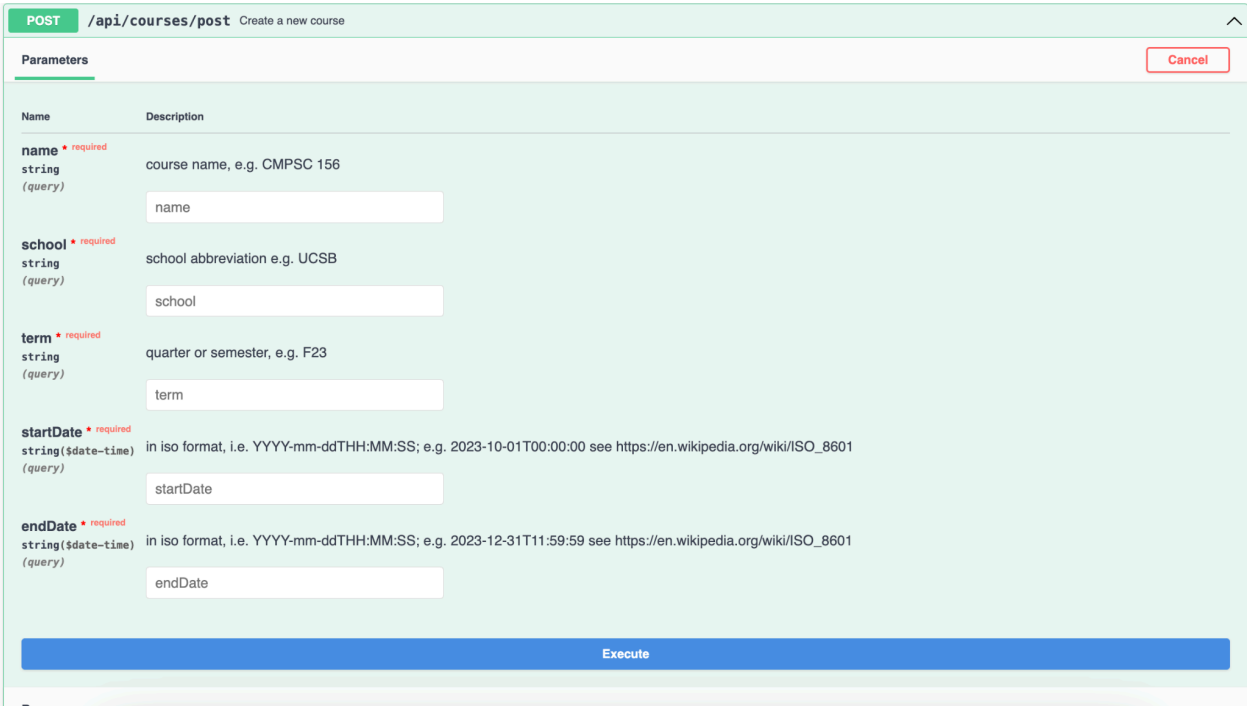
1. AnnouncementControllerForm.js - fixed page mapping to navigate to the Announcements page per commons (include commons id). Additionally, fixed the parameter mapping on the create announcement response entity. Lastly, added code to change the time zone to PST (when selecting the current date for announcement start date). Added new tests and fixed existing tests to match the new file.
2. AdminAnnouncementPage.js - Additional tests added to reach 100% line coverage for the file. Button added to the page to navigate to the AdminCreateAnnouncementsPage.
3. AdminCreateAnnouncementsPage.js and AnnouncementForm.js - Implemented the announcement form into the create announcement page. Fixed the communication between the frontend and the backend. Added navigation to the page to send the user to the home page once the form was submitted. Added full test coverage for both files. The form requires the end date inputted to be after the start date.

Courses

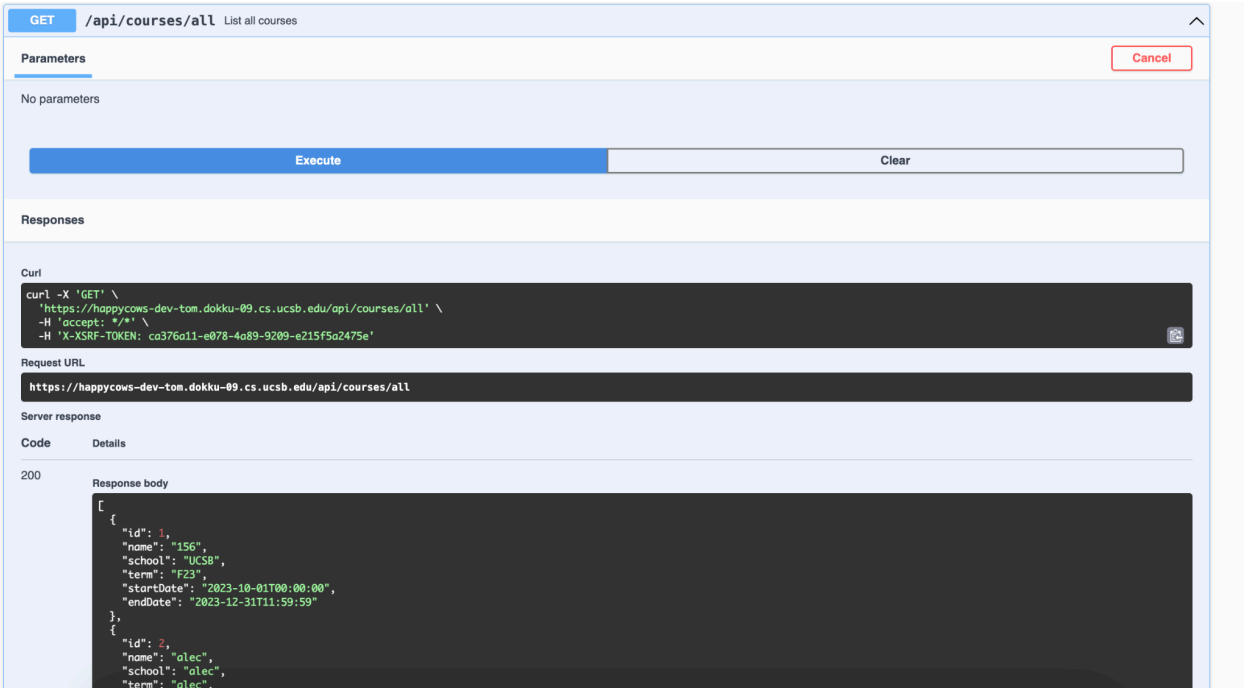
1. Courses.json - Database table for Courses in the backend. For the Courses table, added attributes include: “ID”, “Name”, “School”, “Term”, “Start Date”, and “End Date”. The table also synced in the dokku.



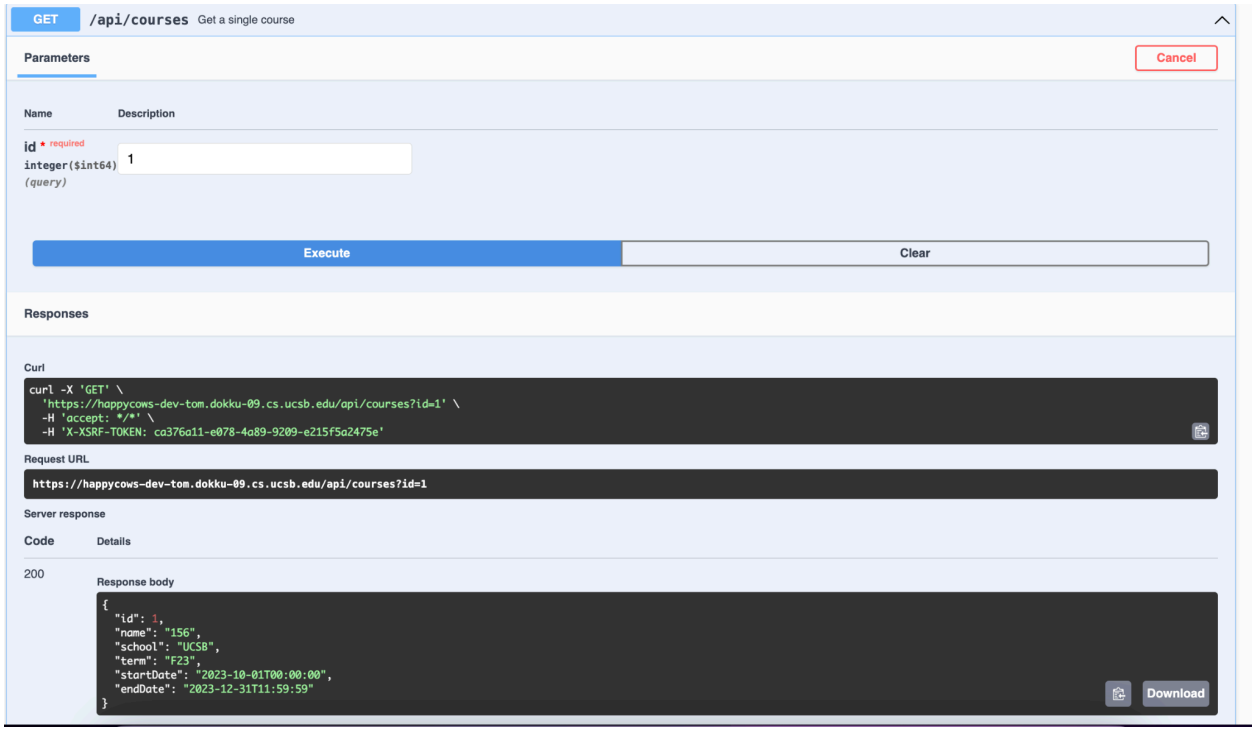
2. Courses.java - Backend APIs for couses
    a. Add a new course: Post method, which includes attributes name, school, term, start date, end date. All these attributes are required to add a new course in the backend. After all these attributes are filled, click “execute” to add it to the database.



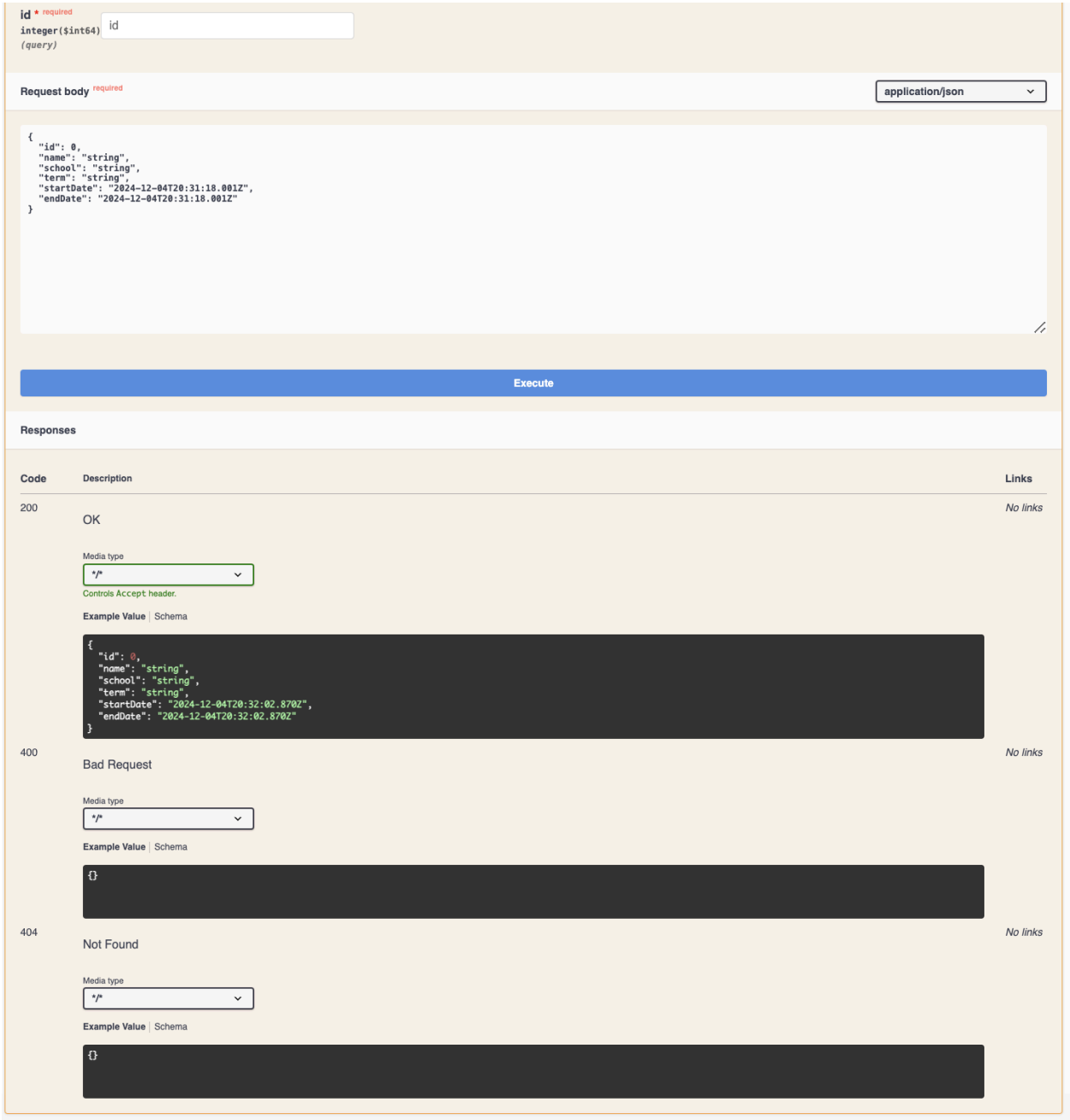
    b. Get all courses: Get method, which lists all the rows (courses) in the Course Database. If the database is empty, an empty value will be returned.



c. Get a single course: Get method, which requires the course ID to query the corresponding course in the database. An empty value will be returned if the corresponding course is not in the database.



d. Put method, which requires the course ID to query the corresponding course in the database. Empty input will return 400 and if the id can not find will be returned 404.



3. CoursesRepository.java - repository for courses table
4. CoursesControllerTest.java - tests for the API of courses controller, which has 100% test coverage and mutation coverage.

Commons:

1. We also added integration testing for the commons controller to test the feature of adding to make sure