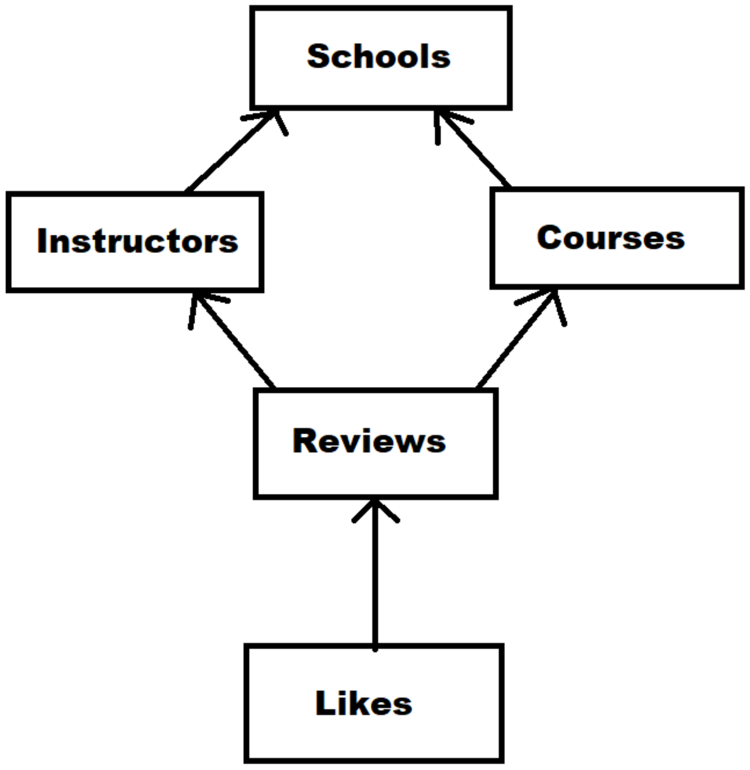
Here are the implementation details of the website:

Database:

The database has the following tables: schools, instructors, courses, reviews, and likes. The following picture shows how references work between tables. An arrow from table A to B means table A has a reference to table B.



The schools database holds all the schools a course or instructor can belong to. Only UCSC is in this table. We originally planned to have several schools but it was not in our time budget to allow this. We did leave the table in, however, so it would be easy to implement in the future. Each table contains the following fields

schools:

- name

instructors:

- first_name
- last_name
- reference to schools table
- department

courses:

- name
- reference to schools table
- description

reviews:

- reference to courses table
- reference to instructors table
- body
- rating
- user

likes:

- is_like (boolean true for like, false for dislike)
- user_email
- reference to reviews table

Front Page:

This is mostly server generated, but there is some Vue code to allow for more courses and instructors to be added to the search bars. The instructors and courses are retrieved through an API endpoint supplied by the server and populated into the search bar. The search bars are implemented using datalist HTML elements.

Add Instructor/Course Page:

These are completely server generated using py4web forms with Bulma styling.

Course/Instructor Page:

These pages heavily use Vue code. Depending on which page is selected, all courses or instructors for the same school as the page are loaded in through an API endpoint, along with review and like data. The course/instructor data is used to populate the datalist element which is used when creating a new review, similar to the home page.

JavaScript code handles ordering reviews. They are in reverse chronological order, with the exception of the “most helpful review”. This is the review with the most likes on the page. JavaScript code figures out if it exists and which review it is, and moves it to the top of the reviews regardless of posting order.

The code for editing a new review uses separate Vue data from adding a new review, so these can be done simultaneously. However, no more than one edit can be done at a time, which greatly simplifies the review editing code. Similarly to the review adding code, for the review editing code a flag is set when a review is being edited which tells the HTML to display the editing fields, allowing the review to be edited. API endpoints are used to update data in the database before editing the Vue data.

Ratings are also calculated dynamically whenever a review is added or edited. If no reviews exist for a page, then the page will indicate this instead of displaying a rating. Otherwise, a simple average of each review rating on the page is used to calculate the page rating.

Review deletion is handled using an API endpoint to update the database before removing the Vue data.

Thumbs up and down work in a similar way, using endpoints to update values before displaying them to the user. After a user likes or dislikes a review, reviews must be re-enumerated. The enumeration function has extra functionality which checks for the existence of a most-liked review for displaying it as the "most helpful review". This is done by finding the review with maximum likes and checking if the amount of likes is above 0. A review must have at least 1 like to be the most helpful. Once a most liked review is determined, it is copied to the top of the list, and deleted from its old position. This does the reordering for a most helpful review, but also means that each time the enumerate function is called, it must re-sort the reviews chronologically in case the most liked review has changed, so that the old one can be put back in its place chronologically.

Profile Page:

The page is split into two columns. The left column is narrow, and contains the user's name and email, as well as a default profile picture. The profile picture is purely aesthetic and cannot be changed. The default profile picture is a public domain picture. In the right column are all of the reviews the person whose profile is being displayed has made. They are displayed in reverse chronological order. They do not display with most helpful review if they qualify for the label on any instructor page or course page. We felt that including this could be confusing, as you would not know what page it was most helpful on without navigating to possibly two pages first.

Extra code had to also be added to every endpoint that leads to a displayed page. It would add a link to the user's profile if they are logged in to the HTML code. This is used in Layout.html to display a link to the user's profile page on every page.

Reviews on this page can be liked or disliked by anyone, just as on the instructor and course page. If the profile page you are viewing is your own, you can also edit and delete reviews, just as on the course or instructor page.

Logging In and Out:

Logging in is handled by Google OAuth. No information except email and name are collected for this website. Logged out users can navigate to any page of the site, except for the Add Instructor and Add Course page. However, logged out users cannot like/dislike any reviews and cannot write reviews. Users cannot change any data relating to their profile (name, email) once it has been created.

URL Signing:

No URLs to pages are signed, since it is not required to be secure. However, all API endpoints are signed which includes loading data and liking reviews, as well as more sensitive tasks like editing and deleting reviews

Hosting and other Extras:

The website and its database are hosted on Google cloud. The website was also configured to use a custom domain: reportcardapp.com. We also made logos for the site to make it look more professional, which are statically stored along with the code of the website.

Report CArd