CS411

Project Proposal

Title: UIUC Review

Project Summary

This is a centralized review app for courses, professors, and departments at UIUC. This project aims to allow students to provide reviews for classes they've taken which will include a written review as well as other descriptors including a rating out of 5 stars, the professor they took the course with, the grade they got, and details like the average and maximum hours the student spent on the class each week. Students can provide reviews for professors in a similar fashion and this information can be aggregated to create an overall rating and summary for different departments.

This will allow students to better understand the experience of a particular class, professor, or department to help them make a more educated decision on which classes to take in the future. The website could even provide recommendations for students based on the courses they've reviewed and their corresponding ratings.

Description

This application will be web-based and will provide a central source of information that is easily accessible for students and guest viewers who may be considering attending UIUC. We will provide users with the ability to search for a particular department, course, or professor and view all associated reviews that have been posted by other students. Courses, professors, or departments can be compared based on aggregate information from the reviews. This will help students make more informed decisions about which courses they choose to take.

Usefulness

The purpose of this application is to create a centralized database for students to provide input and review courses based on certain predetermined criteria to allow for a more holistic view of a course. Though such resources related to UIUC do exist in the form of the Grade Disparity/GPA distributions and the UIUC Reddit page, there is no centralized application where more personal experience can be summarized in a systematic way.

There are websites which provide some functionality similar to this. These include RateMyCourses and RateMyProfessors which allow people to make reviews for courses and professors respectively for different institutions. However, in most cases there are a very limited number of reviews and the apps do not provide the level of information that we wish to provide. Furthermore, no existing app provides information on both courses and professors. Having access to information on both is an important consideration.

The most similar website that we are aware of is Brown University's Critical Review. This provides students with aggregate information about courses in each department and the professor that taught the course. Though this provides more interesting metrics about each class like the average time students spent per week and ratings of the workload and usefulness of the course, it does not allow for written reviews.

Our app aims to combine many of these features for UIUC so that students have a reliable source of information.

Realness

Information about courses and faculty can easily be acquired online via the UIUC website to determine which courses, departments, and professors we should add to our database

Since our app uses specific information from UIUC students for the reviews, we can collect student information through forms spread via school emails or course announcements with the help of professors and other students. We could potentially ask for access to previously completed course forms to aggregate previously collected data. Since this could be sensitive data, it may make sense to keep this anonymous. Alternatively, we could initially generate random reviews as a dummy placeholder for authentic data.

Functionality

Initially, our users will only be students at UIUC who will be able to log in with some credentials (ideally these would eventually use proper UIUC credentials) or guest users who are just browsing the site. The main page will provide a search bar where the user can search for a particular course, department or professor. These search results can then be filtered based on certain criteria such as average GPA, rating, time spent per week, course level, etc. Once a particular course or professor is selected, the user sees information about the course including a quick summary, the professors who have taught it recently, aggregate information like average rating, GPA and time spent, as well as a list of the existing reviews from other students. Again, the user can filter the reviews based on criteria such as the date of the review, the professor who taught the class when the reviewer took it, the rating of the review, etc. Users can also select multiple courses, departments, or professors for comparison based on aggregate information.

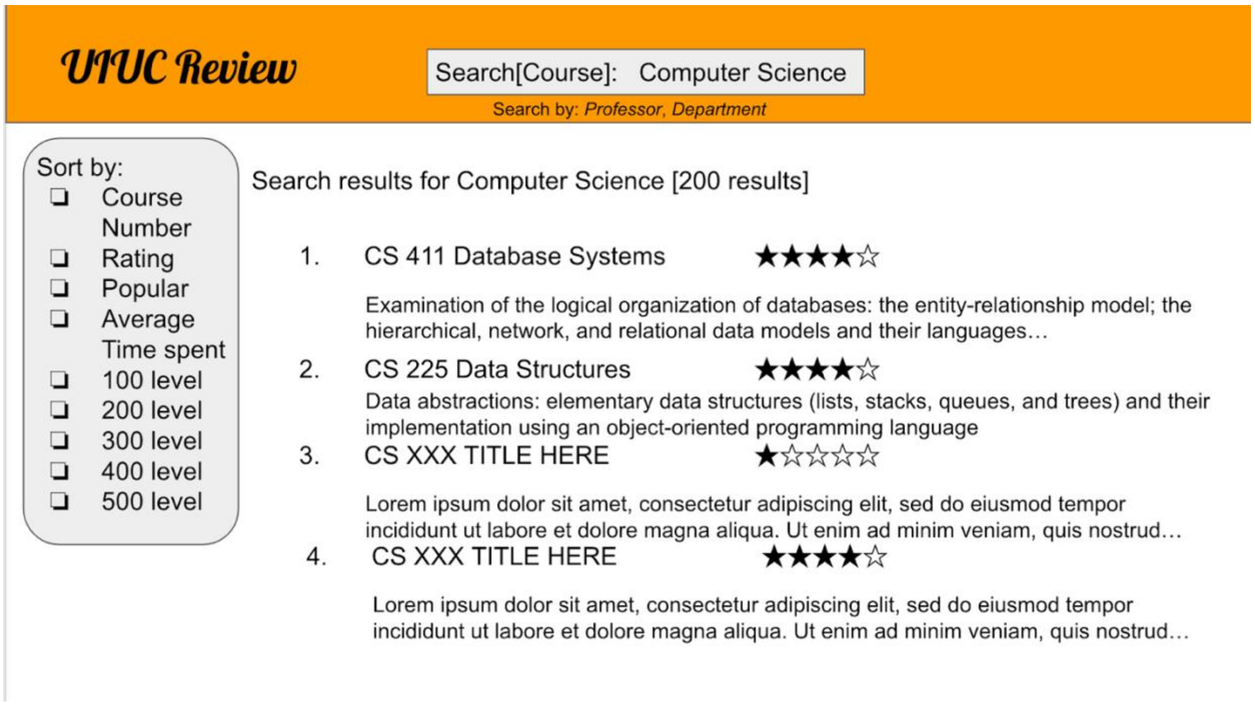
For students only, there will be an account page which shows their personal information as well as existing reviews that they have posted. Here they will have a chance to edit or delete any reviews. Similarly, when visiting the page for a course, department, or professor they can update, delete, or add reviews if they have taken the course or been taught by the professor.

To achieve this, we need multiple different relations. These include a relation for student information (netID, department, major, gpa, year/status), professor information (name, department), course information (course name, course ID, department), course reviews (review body, overall rating, avg hours spent per week, professor, courseID, year, etc), professor reviews (review body, overall rating, various other ratings such as their level of feedback, how effectively they presented class material, etc), course enrollment (studentID, courseID, grade, professor, year), and perhaps department information. Note that this is not a complete list of relations or attributes since there will be more complex relationships that would require additional tables. There would also be an additional relation to store login information. How this data could be collected is mentioned above.

Some potential additional and complex features could include an autocomplete feature for the search bar, a recommendation system for students which suggests courses or professors based on the courses they have taken and the reviews they have made. The app could also include some more social features where departments or courses can make announcements on the app to notify students of new information. These could be achieved using NLP toolkits, additional statistical libraries for data analysis, or additional accounts and authentication that could allow for professors or departments to be separate users that can post information.

UI Mockup

Below is a simple version of what the main search page would look like following a search of the CS department



UIUC Review

Search[Course]: Computer Science

Search by: *Professor, Department*

Sort by:
- ❑ Course Number
- ❑ Rating
- ❑ Popular
- ❑ Average Time spent
- ❑ 100 level
- ❑ 200 level
- ❑ 300 level
- ❑ 400 level
- ❑ 500 level

Search results for Computer Science [200 results]

1. CS 411 Database Systems ★★★★☆

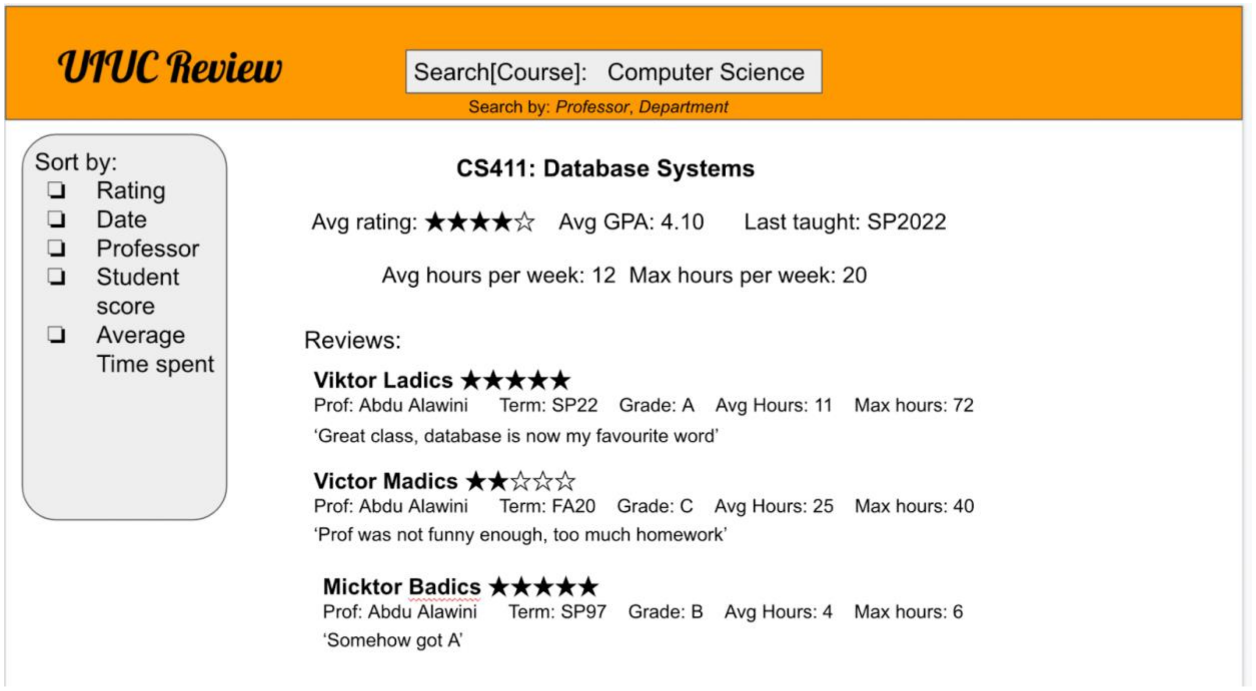
   Examination of the logical organization of databases: the entity-relationship model; the hierarchical, network, and relational data models and their languages…
2. CS 225 Data Structures ★★★★☆

   Data abstractions: elementary data structures (lists, stacks, queues, and trees) and their implementation using an object-oriented programming language
3. CS XXX TITLE HERE ★☆☆☆☆

   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud…
4. CS XXX TITLE HERE ★★★★☆

   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud…

Below is a simple version of what the page for a particular class or professor could look like with aggregate information about the course near the top and reviews listed below.



CS411: Database Systems

Avg rating: ★★★★☆ Avg GPA: 4.10 Last taught: SP2022

Avg hours per week: 12 Max hours per week: 20

Reviews:

Viktor Ladics ★★★★★
Prof: Abdu Alawini Term: SP22 Grade: A Avg Hours: 11 Max hours: 72
'Great class, database is now my favourite word'

Victor Madics ★★☆☆☆
Prof: Abdu Alawini Term: FA20 Grade: C Avg Hours: 25 Max hours: 40
'Prof was not funny enough, too much homework'

Micktor Badics ★★★★★
Prof: Abdu Alawini Term: SP97 Grade: B Avg Hours: 4 Max hours: 6
'Somehow got A'

Project Work Distribution

Work is distributed based on the requirements for each stage as well as the general backend functionality as shown below.

Stage 2

- Agree on our 5+ core entities
  - **Everyone**
- Make ER/UML diagrams
  - **2 people: Jainil, Michael**
- Convert ER/UML diagrams to logical design
  - **2 people : Sushma, Viktor**

Stage 3

- Implement database in MySQL on GCP
  - Generate data (with a Python script or something)/collect data (online source or survey)
    - **2 people: Michael, Viktor**
  - Create relations, write up the commands we used to set up our database in a PDF, insert data into tables
    - **2 people: Sushma, Jainil**

Develop two advanced SQL queries and run each on our data to show top 15 results

- Each need to involve at least two of the following: joins, set operations, aggregation, subqueries
- **2 people per query:**
  - **Michael, Jainil**
  - **Viktor, Sushma**

CS411

Indexing

- Use EXPLAIN ANALYZE to measure your advanced query performance before adding indexes
- Explore adding different indices to different attributes on the advanced query and see how performance changes with EXPLAIN ANALYZE
- Report on the index design
    - **Everyone**

Stage 4: midterm demo

- Create simple UI that performs CRUD
    - **2 people for actual UI: Viktor, Michael**
    - **2 people for CRUD : Sushma, Jainil**
    - **People can switch as we go**
- Implement each operation at least once (5+ for CRUD and search)
    - **Everyone**
- Integrate both advanced SQL queries from stage 3 into our app
    - **Sushma, Michael**
- Demo app to our TA based on the slot we sign up for
    - Demo is timed, cannot go over slot time
    - Whole team doesn't need to be there
    - **1 person up to everyone: Jainil, Viktor**

Stage 5: final demo

- Complete the application with basic CRUD operations
    - Improve UI
    - Add more operations and features
    - **Michael, Viktor**
- Implement an advanced database program related to the app
    - Integrate advanced database program with the app
    - Trigger the advanced database program by some front-end interaction
        - Should include transaction + trigger OR a stored procedure + trigger
    - **Sushma, Jainil**

Demo

- Demo itself
    - **1 person: Viktor**
- Discuss system design choices
    - **2 people : Sushma, Michael**
- Discuss future improvements or extensions
    - **1 person: Jainil**

Extra credit: develop a creative component (function) that complements your project

- Can use existing libraries and/or APIs
- Can be simple (e.g interactive visualization tool) or more complex (e.g. ML recommendation system)
- **1 person up to everyone**

Stage 6

- Final reflection report
    - **Everyone**
- Demo video
    - **Everyone**

CS411

Functionality

- Search on Courses/Departments
    - **Everyone**
- Filtering based on the search. Sorting on the filtered results
    - **Sushma**
- Statistic calculation like AvgGPA, AvgCourseLoad, MaxGrade etc.,
    - **Michael**
- Adding, editing and deleting a review if you are a student
    - **Viktor**
- Comparison of different courses/departments based on selected metrics
    - **Jainil**