ECT584 Final Project Report

Dachi Xu

1. **Project overview**

The purpose of this project is to build a recommendation system application for the DePaul CDM courses, the assumed users will be DePaul CDM students, and the application is supposed to recommend the most relevant courses based on the user profile and interests.

**Project demo URL:** [**https://depaulmazon.herokuapp.com/**](https://depaulmazon.herokuapp.com/)

1. **System architecture**

The application is designed to be a web application, and I designed and developed both front and back end of the application, the technique stack of the application is listed as follow:

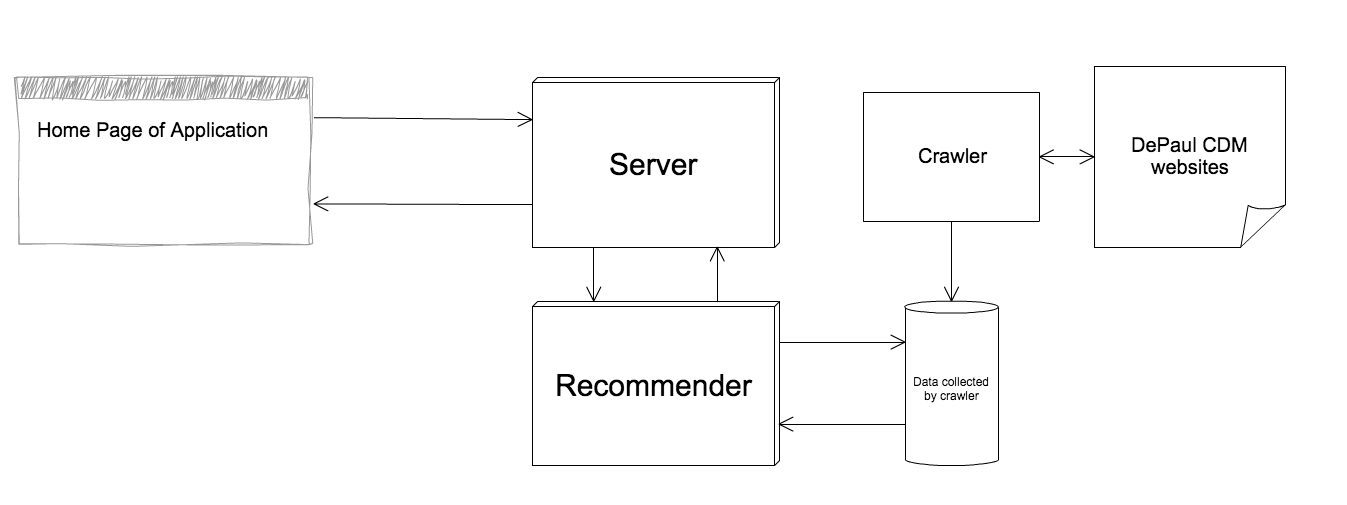
Material CSS library (A front end CSS & JS library)

Flask (A Python back-end framework)

Numpy, Scipy, Sklearn (Python computation and machine learning frameworks)

The architecture of the application is divided into three major part: crawler, recommender and server, the crawler is used to collect course and faculty information, recommend is built to process and extract recommendation information from the data collected by the scrapper, and server part is used to handle the user request and return recommendation back to front end.

The illustration of architecture is in the graph below



The complete workflow of the system is organized as below:

1. **System implementation**
   1. Scrapper

This part does not require extra algorithm implementations, I am using a third party python library called “BeautifulSoup”for scrapping the courses and instructors data from DePaul websites. I collected several attributes of courses and teachers listed as below:

* + - * Course Id: e.g. ECT584
      * Course title: e.g. Web Data Mining
      * Course description: e.g “An in-depth study…”
      * Course previous Instructors: e.g Jonanthan Gemmell, Bamshad Mobasher
      * Course URL: <http://www.cdm.depaul.edu/academics/pages/courseinfo.aspx?Subject=ECT&CatalogNbr=584>
      * Instructor name: Jonanthan Gemmell
      * Instructor url: <http://www.cdm.depaul.edu/about/pages/people/facultyinfo.aspx?fid=494>

The scrapper collects those data and store them to local json files for later tasks.

* 1. Server

The main server program is the “index.py”, this scripts servers as the server and handles all requests from the front-end, it will subtract the user profile data from request and pass them to the recommender functions, and the server will send the recommendations computed by the recommender back to the front-end.

* 1. Recommender

This is the core part in calculating the recommendations. There are also several parts in the recommender

* + - Loading data

This step will load the courses, teachers and some other pre-computed data for recommendation calculation

* + - Compute course x term frequency

I used sklearn countvectorizer to tokenize and compute the term frequency of all the course description information, I collected 118 courses whose major includes CSC, ECT, IS, IT and HCI. Sklearn generated 1490 terms, so I got a 118\*1490 matrix after this step.

* + - Compute course x term td\*idf

I used sklearn tfidftransformer to compute the td\*idf matrix of the course \* term matrix from previous step.

* + - Compute course x course cosine similarity

After I got the td\*idf matrix, I calculated the cosine similarities using the td\*idf matrix, this similarity is useful for recommending courses later on.

* + - Select courses

To recommend course we need both the user profile and course \* term td\*idf, user will choose different profile options (including major, level and concentrations), concentrations is actually the terms from course \* term matrix, I mapped terms to different majors, for example, I mapped “algorithm”and “application” to CS major. Once a user choose CS major in UI, the webpage will display “algorithm” and “application, the recommender will pick the courses with the highest values in the td\*idf matrix on that specific column of algorithm, in addition to that, the recommender also picks the most similar courses with the courses selected on the previous step.

* + - Select teachers

The teacher is selected based on the course selections, the course data stores the previous instructors, so the recommender will collect information for those teachers.

* + - Return results

The recommender does not return all the courses, it will choose some random items from the courses and instructors from the last two steps, so the user will not be overwhelmed by too much courses and teachers at once.

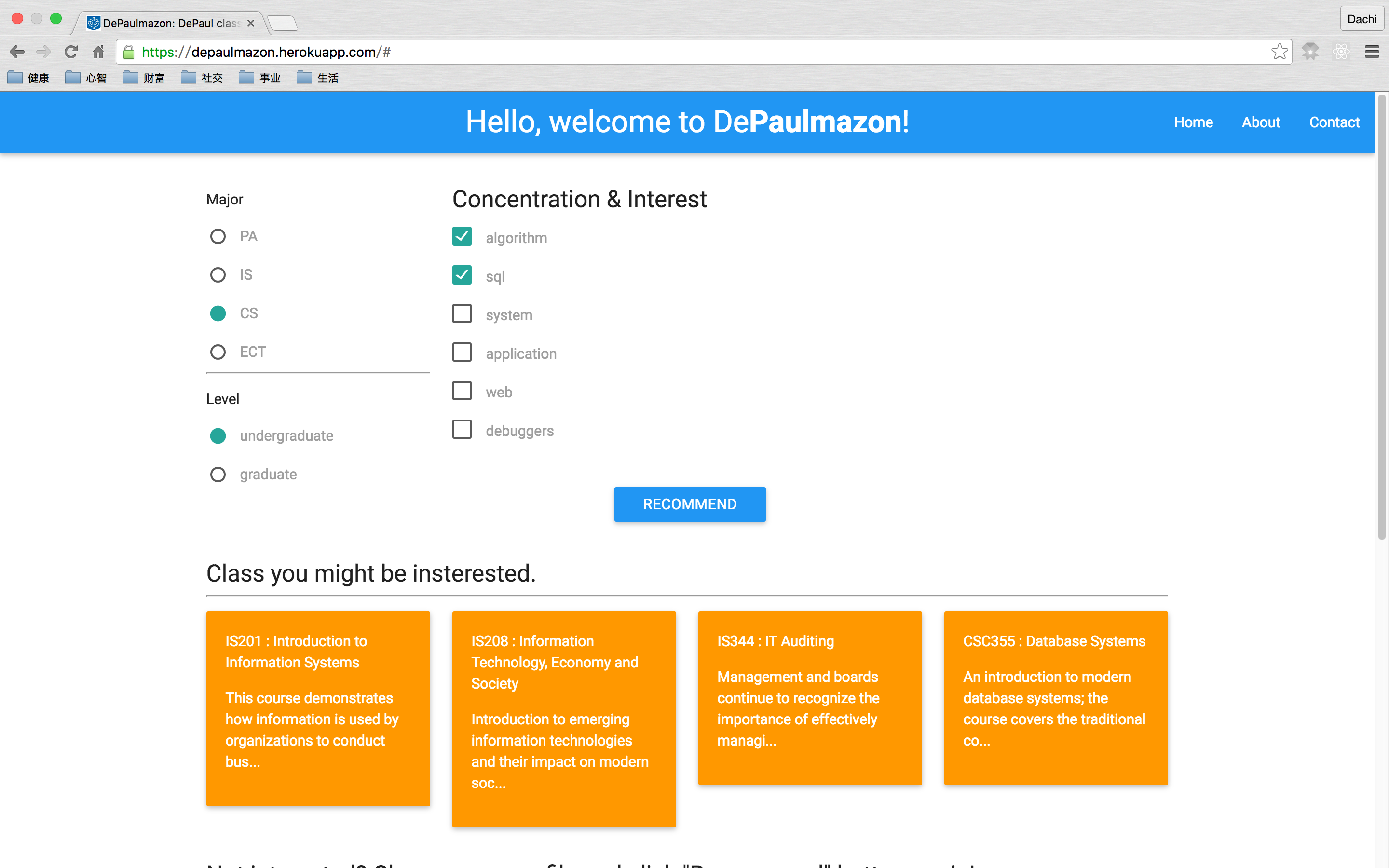
1. **System evaluation**

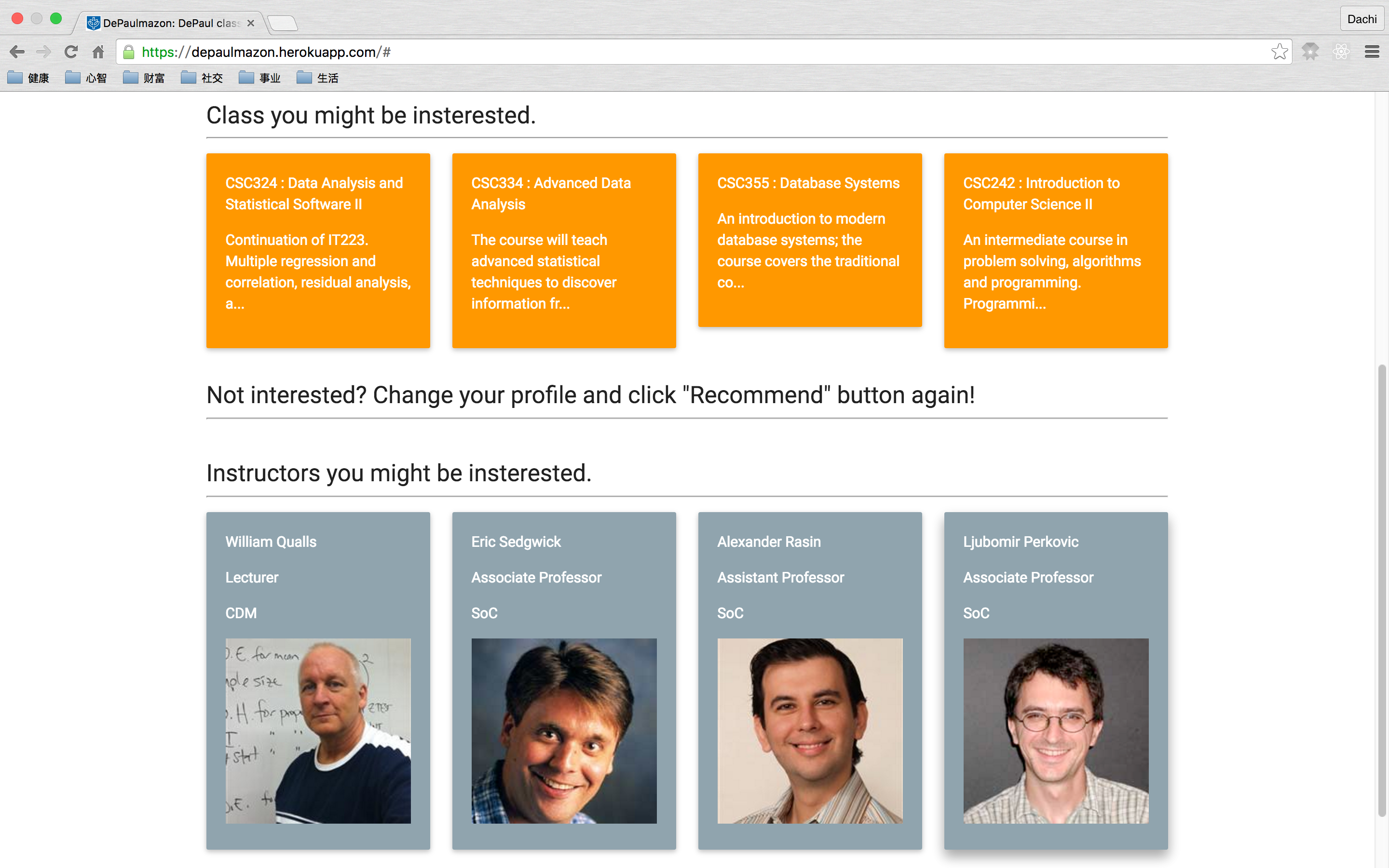
**Since this is a brand new application, it is hard to collect enough data to estimate the accuracy and efficiency of the application, currently the tests and evaluations are all based the experiments by myself and some of my classmates.**

**From the experiments we found sometime that the recommended course is not in line with the concentration selected by the user, few reasons could contribute to that, the first reason is that not many course are having positive values on the column of that concentration in td\*idf matrix, so the recommender is returning some courses with 0 values in the matrix, the second reason is that the recommender also pick course similar to the courses with high td\*idf values, the courses might be similar for some different column(term) values, thus making the recommended course less relevant to the specified terms.**

1. **System run example**

**Below is example run on the recommender system.**

**The user select the major(“CS” in this example) and level(“undergraduate”), the server then displayed some concentrations and interests for user to choose, I choose algorithm and sql and clicked “RECOMMEND” button, then the server processed the request and returned some potential interesting courses and instructors who taught those course before to me.**

****

**And if you click those courses and teachers, a new tab that leads to the official school pages of those course/teachers will be open.**

1. **References:**

1. Wikipedia on recommender system <https://en.wikipedia.org/wiki/Recommender_system>

2. Michael J.Pazzani et al, Content-based Recommendation Systems: <http://www.fxpal.com/publications/FXPAL-PR-06-383.pdf>

3. Pasquale Lops, Marco de Gemmis and Giovanni Semeraro, Content-based Recommender Systems: State of

the Art and Trends: <http://facweb.cs.depaul.edu/mobasher/classes/ect584/Papers/ContentBasedRS.pdf>