**Team #36:**
**Lawrence Chan, Madeline Gelfand, Joy Qiaoyi Wang, Yuting Summer Yue**

* * *

A modern software infrastructure project isn't done until you understand how it performs, and where the bottlenecks are. Instrument your application to collect timings on various aspects. You should at least be able to determine what the latency in handling each request is, and extra credit will be awarded if you can also see what happens under multiple concurrent requests.

Your final report should include a write-up of:

Introduction and project goals
Basic architecture (not a dump of the classes)
Key features of the project.
Technical challenges and how they were overcome
Description of your complementary data and how you extracted it
Performance evaluation
Potential future extensions

* * *

## I. Introduction

College seniors have a lot on their minds as their University education comes to a close revolving around what jobs to apply for, what salary they should accept, and where they would live. We recognize, however, that these choices are based upon previous ones: high school seniors are under tremendous stress to choose colleges and majors and these are the primary factors that will determine what a college senior's job search looks like. With this in mind, we set out to make a tool for perspective college students. We focus on how salary data relates to college variables.

## II. Project goals

Our goal was to make accessible salary information based on colleges and other demographics. Because many people do not ask outright about salary information and social norms keep them from being vocal about these questions, we wanted to make this information accessible. Providing simple information was not enough, we wanted to include many different factors that went into post-grad salary. We combined many data sets to provide easy access to lots of different information.

## III. Basic architecture (not a dump of the classes)

A. In terms of the site, we built a MEAN stack (NodeJS, AngularJS, Mongo, Express) web page hosted locally. We used Node to create APIs that enable the Angular frontend to get access to our data in both our mySQL database and our Mongo database. We used the d3.js library in the Angular controllers for data visualization and implemented basic HTML and CSS files for the foundation of the page. All pages share the basic structure

in index.html. We used ng-include to insert the footer and the navigation bar. On top of this basic html structure, we insert separate Angular views for different app features. The views include “by salary”, “by major” and “by college”, which can be accessed from the navigation bar and associated with their own controllers.

1. server.js: Node backend that starts the server, connects to both Mongo and mySQL databases and defines the APIs that talk to the databases.
2. public/app/controllers: byCollegeCtrl.js, byMajorCtrl.js, bySalaryCtrl.js and mainCtrl.js are AngularJS controllers used for each page.
3. public/app/services: Angular services and factories used for utility functions shared by all controllers.
4. public/app/routes.js: ngRoute file that directs each URL to its corresponding views and controllers.
5. public/assets: external javascript files and css files.
6. report/: our project report and screenshots used for the writeup.
7. package.json: defines the dependencies, starting scripts and meta information.

B. We used Github for version control and collaboration.

C. Dataset Tables:

1. **Major**: **major_code** varchar2(20), **rank** varchar2(20), **major_name** varchar2(100), **total** varchar2(20), **men** varchar2(20), **women** varchar2(20), **employed** varchar2(20), **full_time** varchar2(20), **part_time** varchar2(20), **unemployed** varchar2(20), **median_salary** varchar2(20), **p_25th_salary** varchar2(20), **p_75th_salary** varchar2(20), **college_jobs** varchar2(20), **non_college_jobs** varchar2(20), **low_wage_jobs** varchar2(20)
2. **Salary_by_degree**: **major_code** varchar2(20), **rank** varchar2(20), **major_name** varchar2(100), **total** varchar2(20), **men** varchar2(20), **women** varchar2(20), **employed** varchar2(20), **full_time** varchar2(20), **part_time** varchar2(20), **unemployed** varchar2(20), **median_salary** varchar2(20), **p_25th_salary** varchar2(20), **p_75th_salary** varchar2(20), **college_jobs** varchar2(20), **non_college_jobs** varchar2(20), **low_wage_jobs** varchar2(20)
3. **Salary_by_school**: **school_name** varchar2(100), **school_type** varchar2(20), **region** varchar2(100), **starting_median_salary** varchar2(20), **mid_career_salary** varchar2(20), **percentage_change_from_starting_to_mid** varchar2(20), **mid_career_10th_percentile** varchar2(20), **mid_career_25th_percentile** varchar2(20), **mid_career_75th_percentile** varchar2(20), **mid_career_90th_percentile** varchar2(20)
4. **Locations**: zipcode, lat, long

## IV. Key features of the project

Key features of this project are outlined below. Please see the project screenshots in our appendix. We tried to pick the most relevant data for students looking at colleges. We assume most people looking at colleges have found a preference for something, whether that be the percentage of the

same gender at your school, the starting salary exiting a major, etc. and we believe we are providing this information in a way that most university pages fail to.

A. “By Major” Page: once a user selects a few majors that she or he is considering (shown as a multi-select input box on the page), a bar graph of the median salaries comes up, comparing each major visually.
B. Once a user selects a few majors, for each major selected whose gender information is contained in our database, a pie chart shows up indicating the percentage of women and men for these majors.
C. Once a user selects a few majors, a bar graph of unemployment rate comes up, comparing each major visually.
D. “By Salary” Page: given a range of median salaries, a resulting list of colleges and majors will pop up where the median salary is in the range. The list is in descending order by median salary.
E. “By College” Page: information on one specific college. User will write in college and receive the following information:
    1. Admission rate
    2. Degree types available
    3. Location
    4. School size (number of undergraduate students)
    5. Average cost
    6. Average net price

## V. Technical challenges and how they were overcome

A. Using Angular to set up the scope of the controller and pass data from database to controller and to view.
B. While importing our dataset to the mySQL database on Amazon, the system was not smart enough to treat Excel numbers as number types. Therefore, some fields are shown as all 0s due to this type mismatch. We had to manually change the data types to re-import to solve the problem.
C. Using routeProvider in AngularJS to insert views into the main page was challenging. There were times when a blank page showed up with no error logs. This made debugging difficult. At one point, we created separate html pages for each feature. However, we decided against it because it was bad engineering practice which made the codebase hard to maintain. So we spent more time learning the routing functions in AngularJS and did the ng-views properly.
D. D3.js was very challenging. The labels on the x-axis would overlap due to the long major names. We had to truncate it to make it show up nicely.
E. AngularJS factories were new to us. We did not want to repeat the general utility code that all AngularJS controllers shared. Therefore, we created a factory called “utilService” and passed it into controllers that needed these general string processing functions.
F. Data processing for certain query results were necessary. For example, the major database had everything in all capital letters, which was difficult to read. We needed to capitalize it property (but without capitalizing words such as “and”). And when it is time

to query results from these majors, we had to turn it back into all caps so the database search returns our expected results.

## VI. Description of your complementary data and how you extracted it

A. Data *(Note: Specific data information and data tables can be found in Appendix A on the last page of this report. Outlined here are the general descriptions of each dataset)*

1. Our first dataset comes from FiveThirtyEight, this gave us college majors with gender distributions, employment rate, and median income. (Relational)
2. The next dataset comes from Kaggle consists of 3 datasets. One of the salary distributions by undergraduate major, one on the salary distribution on school name and school type, the last dataset is on salary distribution by region. The salary distribution is represented by Starting Median Salary, Starting Median Salary, Mid-Career Median Salary, Mid-Career 10th Percentile Salary, Mid-Career 25th Percentile Salary,Mid-Career 75th Percentile Salary and Mid-Career 90th Percentile in all three datasets. (Relational)
3. The College Scorecard project is designed to increase transparency, putting the power in the hands of students and families to compare how well individual postsecondary institutions are preparing their students to be successful. This project provides data to help students and families compare college costs and outcomes as they weigh the tradeoffs of different colleges, accounting for their own needs and educational goals. (NoSQL JSON with API)
4. The Gist dataset mapped latitude and longitude lines to zipcodes. One of our datasets has locations stored as latitude and longitude coordinates, and this helps translate coordinates into zip codes. (Relational)

B. Extraction

1. We downloaded the csv files and processed them locally. Once they existed in ideal format, we uploaded the files to Amazon RDS. We used DB Snapshot and mysqlimport.

## VII. Performance evaluation

The performance for the website is very good overall. There is no significant delays in rendering the pages. Since we inserted the views into the index page, we do not need to re-render same components such as the footer and the navigation bar. The The bottleneck falls on when we call the Node APIs that send queries to the databases hosted online. For all our features, we have to wait until the data to come back before starting the visualization. Some future solutions for this bottleneck:

- Cache the most commonly searched queries locally.
- Host databases on machines that are closer to our server.

## VIII. Potential future extensions

Given more time, we would have added more features to this project. While the following features were conceived of for this CIS550 group project, the site in general could have a lot of future extensions. We could have included much more data on each college, more aggregate data, and effectively turned this into a college stats page with all kinds of information on colleges and visualizations for comparisons.

A. A US map visualization with different regions and states. When the user clicks on certain parts of the map indicating a particular region,
    1. a line chart indicating mid career median salary appears on the side. (Including 10th, 25th, 75th and 90th percentiles)
    2. a pie chart appears indicating the school type distributions in this region (Salaries by college JOINS region by college).
B. Graphics, improved design
C. Links to colleges, scholarships, and other sites for specific information
D. Select on a range of tuition / average cost, and have the option to choose to see results by school, satisfying the salary range
E. Shows a list of colleges and their tuition / average

**Appendix A - Specific Information on Datasets**

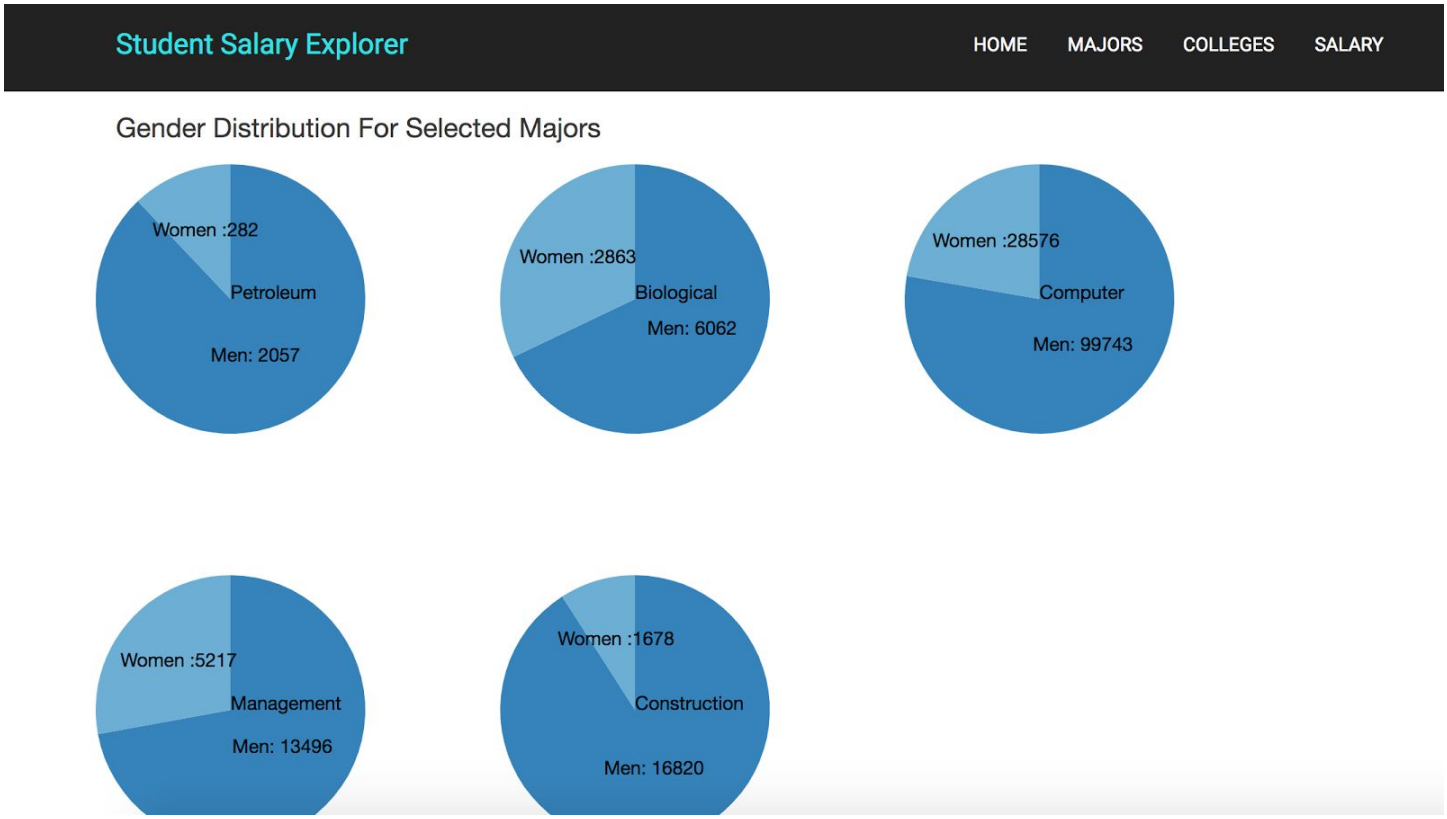
FiveThirtyEight
**Type**: Relational
**Description**: The FiveThirtyEight dataset on salary data for different majors and demographics
**Link**: https://github.com/fivethirtyeight/data/blob/master/college-majors/recent-grads.csv
**File**: recent-grads.csv
**Fields**: Rank, Major_code, Major, Total, Men, Women, Major_category, ShareWomen, Sample_size, Employed, Full_time, Part_time, Full_time_year_round, Unemployed, Unemployment_rate, Median, P25th, P75th, College_jobs, Non_college_jobs, Low_wage_jobs

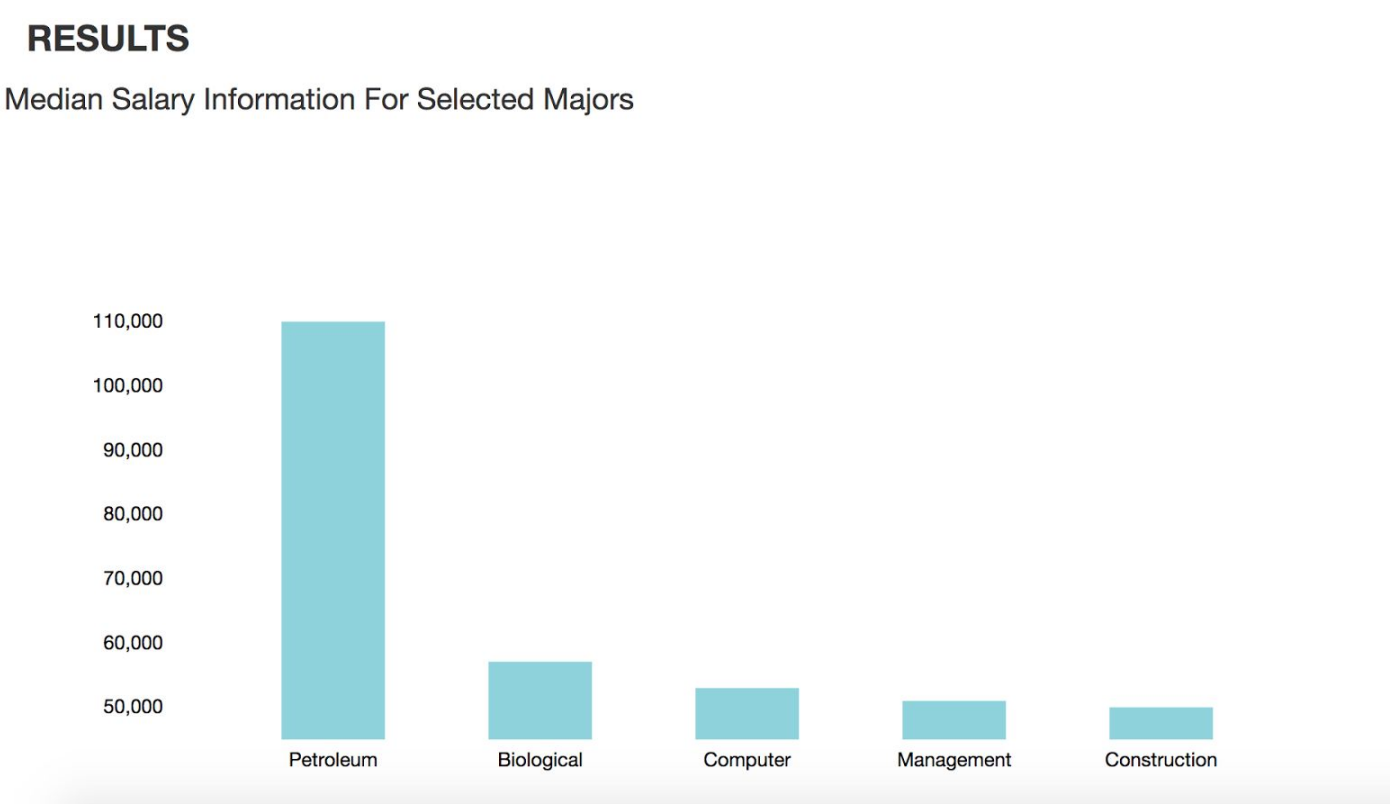
Kaggle
**Type**: Relational
**Description**: The Kaggle dataset named "Where it pays to attend college" on the relationship between salary data and colleges
**Link**: https://www.kaggle.com/wsj/college-salaries/data
**File**: degrees-that-pay-back.csv
**Fields**: Undergraduate Major, Starting Median Salary, Mid-Career Median Salary, Percent change from Starting to Mid-Career Salary, Mid-Career 10th Percentile Salary, Mid-Career 25th Percentile Salary, Mid-Career 75th Percentile Salary, Mid-Career 90th Percentile
**File**: salary-by-college-type.csv
**Fields**: School Name, School Type, Starting Median Salary, Mid-Career Median Salary, Mid-Career 10th Percentile Salary, Mid-Career 25th Percentile Salary, Mid-Career 75th Percentile Salary, Mid-Career 90th Percentile
**File**: salaries-by-region.csv
**Fields**: School Name, Region, Starting Median Salary, Mid-Career Median Salary, Mid-Career 10th Percentile Salary, Mid-Career 25th Percentile Salary, Mid-Career 75th Percentile Salary, Mid-Career 90th Percentile

College Scorecard
**Type**: NoSQL JSON with API
**Description**: Comprehensive information on different metrics for colleges and their graduates' salary information
**Link**: https://collegescorecard.ed.gov/data/

Gist
**Type**: Relational
**Description**: Zipcodes mapped to latitude and longitude values
**Link**: https://gist.github.com/erichurst/7882666
**Fields**: ZIP, LAT, LNG

## Appendix B - Feature Screenshots (By Major)



Student Salary Explorer

HOME MAJORS COLLEGES SALARY

## BY MAJOR

Let's explore infomration about selected majors, including starting and mid career salary, gender distribution and unemployment rate.

Please select the majors you are interested in:

Civil Engineering
Construction Services
Engineering Technologies
Environmental Engineering

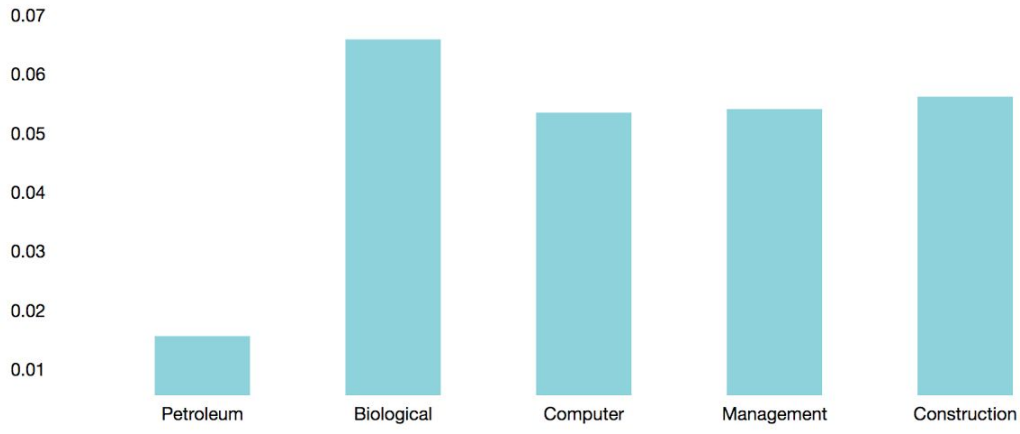
Explore

## RESULTS

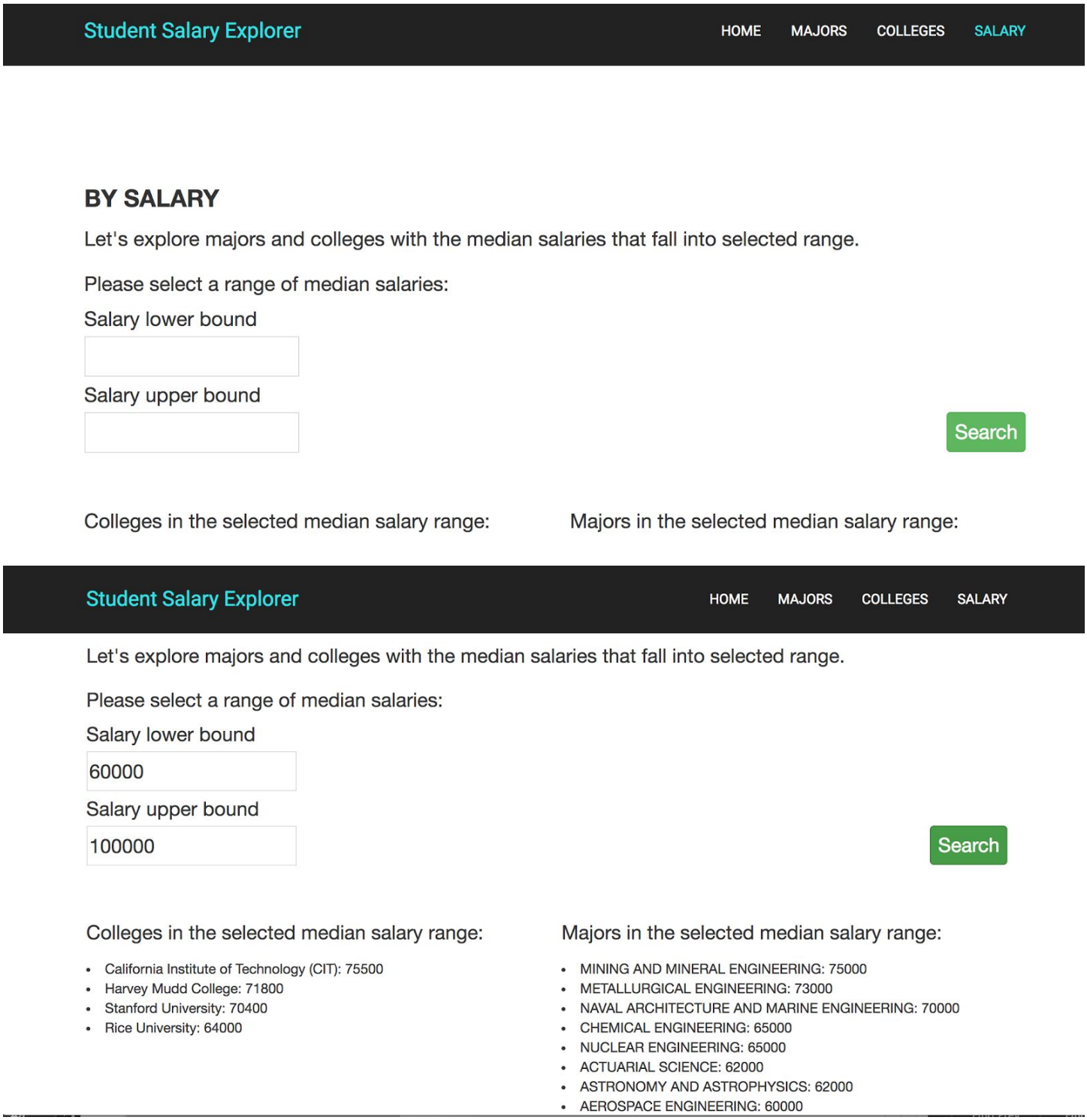
Student Salary Explorer

HOME MAJORS COLLEGES SALARY

Unemployment Rate For Selected Majors



## Appendix C - Feature Screenshots (By Salary)



## Appendix D - Feature Screenshots (By College)