Team Information

Github Repo Link: https://github.com/UT-SWLab/TeamE3

Team Members:

- Harrison Berrier
  - EID: hlb962
  - Email: harrison.berrier@utexas.edu
  - Github username: harrisonberrier
  - Estimated completion time for each member: 9
  - Actual completion time for each member: 11
- Mohit Gupta
  - EID: mg58629
  - Email: mohit.gupta@utexas.edu
  - Github username: mohitg17
  - Estimated completion time for each member: 10
  - Actual completion time for each member: 12
- Nikhil Jalla
  - EID: nj5473
  - Email: nikhiljalla17@utexas.edu
  - Github username: nikhiljalla17
  - Estimated completion time for each member: 10
  - Actual completion time for each member: 11
- Silas Strawn
  - EID:
  - Email: strawnsc@gmail.com
  - Github username: StrawnSC
  - Estimated completion time: 10
  - Actual completion time: 11

Canvas Group: E3

Project Name: US Colleges & Universities Internet Database

Motivation and Users

We envision our site being used by prospective students to assess which college or university is right for them, as well as which major or field of study they should pursue. Our database lets students view universities in the US filtered by their city, or by the majors they offer. For example, if a student knows they are only interested in schools that offer a particular program, they can go to the major page for that field of study, and browse the colleges that offer that major. On the other hand, if the student is interested in colleges only in a particular location, they can pull up all the colleges for a particular city. Once they're on a university or college's

page, the prospective student will be able to see crucial information for making their choice. For instance, they will see facts on the cost of attendance, the acceptance rate, average SAT scores, the size of the school, etc. If they want to learn more, they can also follow a link to the institution's web page.

Stories:

- As a high school student, I want to be able to look through a list of universities in the United States so that I can decide where I want to go to college.
    - Estimate: 2.5
    - Actual: 2.5
- As a high school student that wants to move away from home, I want to be able to look at university education statistics for different cities so that I can decide where I want to move.
    - Estimate: 2
    - Actual: 3
- As a high school student that hasn't decided what I want to major in, I want to be able to look through a list of different majors and their associated statistics so that I can decide what I want to major in.
    - Estimate: 2.5
    - Actual: 3
- As a user, I want to read about the website that I am using so that I know where my information is coming from and what the creator's motivation is.
    - Estimate: 2
    - Actual: 3
- As a user that isn't very good with technology, I want the website's interface to be simple and navigation to be easy so that I can access the information that I want quickly and easily.
    - Estimate: 2.5
    - Actual: 3

Design

Pages

Every page inherits the base.html file, which includes the navbar. Each of the model base pages uses the model.html. Each instance uses the [model]_instance.html file.

The landing page (index.html) is a central page where users can navigate to any of the model's base page. It contains a carousel and pictures that link to the models.

The about page (about.html) contains information about the site and the developers.

Flask

Data is hardcoded in dictionaries in main.py and passed to the instance pages. The routes follow the format [model]/[instance_name].

Models

University Model
Each university instance contains relevant statistics about the university along with its location and the majors that it offers. The page has an overview at the top that includes the school size, acceptance rate, and in-state tuition. Below that, there are sections for school info, majors, admissions, demographics, and tuition and aid. The left panel has links that navigate to each section. The right panel includes extra data about degrees awarded, completion rate, average earnings for graduates, and overall retention rate. All of the data is pulled from the CollegeScorecard API.

We would like to present some of the data graphically to make the page more visually appealing. Demographics data can be easily represented in a pie chart. More data about admissions and completion rates can also be displayed in intuitive ways. We can also make the pages more interactive for the user by adding more responsive elements such as collapsibles and buttons. We also need to link the university instances to the city and major instances.

City Model
Each city instance is described by statistics that would be relevant to students who would like to attend a school in the respective city. Currently, the page lists the area of the city, population, population density, community type, median gross rent, and of course the schools that are in the city. This data was acquired from this website - http://www.city-data.com.

The next step for this page would be to find an api with more relevant information and a better way to present the information. The core point of this page should be the information about the city, not the schools that can be found in the city.

Major Model
Each major instance page contains a list of universities that offer that major, as well as some aggregated statistics on the number of programs in the US for that field of study. We currently list the number of programs that offer a bachelor's degree, an associate's degree, and certificate (after less than one year of study) in the given major. All of this data was acquired from the Department of Education's College Scorecard API. Finally, we list the median starting salary and median mid-career salary for graduates of that major. Since the Department of Education didn't provide this information, we had to acquire it from an article posted on "Visual Capitalist."

Ideally, we would like to sort the universities in each major instance's list by the ranking for that particular major, but we have not found open datasets with such information. On the other hand, we will be able to order the Universities/Colleges by percentage of degrees awarded for a given major. However, since the site content is currently static, and we don't have many University pages up yet, this was not implemented in Phase I. In future phases, we may also consider

augmenting the major data with other datasets because the Department of Education data is occasionally overly-broad. For example, all engineering majors are grouped together as one field of study.

Tools, Software, and Frameworks

Flask
Our backend framework is flask, a microservices framework for python. Flask is useful in that it is lightweight: it was trivially simple to set up the server and start running it locally. When we want to add new pages, we simply add a route for the new page. We also took advantage of Flask's template mechanics. Using flask templates, we dynamically generate instance pages for our major, university, and city models. This allows us to reuse the same HTML, instead of adding hundreds of nearly identical HTML files for each university.

Bootstrap
Our frontend framework is bootstrap, which we use alongside HTML and CSS to design our web pages. Bootstrap has been crucial for allowing our pages to be reactive. The pages in our site dynamically resize based on the size of the viewport, allowing our format to remain robust and look clean, despite the window size limitations of the user.

Google Cloud Platform
Our site is deployed to Google Cloud Platform. We used GCP because it is simple to set up and use with flask applications. All we have to do to update the deployment of our application is run the shell command "gcloud app deploy" in the /idb_app/ directory, using the gcloud CLI.

MongoDB and Mongoengine
Since Phase I allows the website content to be static, we are not yet pulling data from a database. However, knowing that we intend to have our data deployed to mongoDB, we preliminarily started writing some basic models with mongoengine, an open-source Python Object-Document Mapper. Mongoengine will allow us to define "schemas" (mongoDB does not technically have schemas since it's NoSQL) with python objects, so we can interact with our database purely in terms of University, Major, and City python objects that we design.

Sources

Flask documentation: https://flask.palletsprojects.com/en/1.1.x/quickstart/
Bootstrap reference guide: https://www.w3schools.com/bootstrap4/default.asp
Footer guide: https://www.w3schools.com/howto/howto_css_fixed_footer.asp
CSS variables: https://www.w3schools.com/css/css3_variables.asp
Major salary data: https://www.visualcapitalist.com/visualizing-salaries-college-degrees/

Reflection

What we learned:

We've learned how to work together and divide work evenly among us. We were able to distribute tasks early on and each person completed their parts when convenient for them. We also learned how to build a website from ground up using html, css, bootstrap, and flask. Through designing the site, we've learned more about bootstrap and how to style components. We've also learned how to use the APIs selected to acquire the data that we need.

Five things that went well:

1. We quickly moved to the use of templates so that multiple pages can be changed quickly.
2. We have been able to make decisions quickly and then adapt as problems arise, given our heavy usage of slack to coordinate and communicate.
3. Our API's are well documented, making them easier to interact with. In particular, the Department of Education's College Scorecard API has extensive higher education data, and provides a spreadsheet explaining all the possible fields one can request from the API.
4. Since flask is a lightweight backend, we were able to get started putting pages together and making routes for them easily.
5. Flask also allows us to pass in parameters to the html, and we used this to dynamically generate different instance pages off of the same base html. Although the data is currently hard-coded in the python flask app, we will be able to easily migrate this over to database queries without having to change the frontend code.

Five things that went poorly:

1. We haven't been able to add a stylesheet other than bootstrap so we are stuck writing styling into each file at the moment.
2. Formatting pages to fit requirements has been very difficult versus formatting a page without specifications because of the way that html elements and they're styling interact with each other.
3. At first, The college scorecard API was overwhelming because of the sheer amount of data and the complexity of crafting queries to get the needed information.
4. Often, the styling of elements with HTML, CSS, and Bootstrap does not work as we would expect. For example, when we were making the about page, we had to put in elements with our photos and bios together. For some reason, adding more text to the paragraph element next to an image would increase the image's size. We ended up using different bootstrap elements to get around this.
5. When we were first setting up the deployment on Google Cloud Platform, the deploy would seem to work in the CLI, but when we tried to visit the site, we would get a 500

error. Apparently, if you don’t have the exact right directory structure and naming of certain modules, GCP doesn’t know how to deploy your flask app.