StudySpots.me

Authors: Bianca Alvarado, Joanne Chen, Vincent Chen, Ami Iyer, Jake Medina

Motivation

As college students, it can be hard to balance the responsibilities of studying for classes with the desire to explore the city and enjoy the time at college. On top of this, there are other circumstances and new environments and experiences to adapt to that make it more difficult to balance everything in life.
Our goal was to help college students balance these responsibilities. We aim to provide a variety of different areas that can fit the circumstances that a college student might need at any given time to work at (i.e. a quiet place, a place that opens for long hours, a place that has cheap food, etc.) and that is around different locations to encourage students to explore more places around where they are.

User Stories

User Story: Filter Study Spots by Closing Time

> *I am a busy university student who often studies late in the evening. I would like to be able to quickly find study spots near me that open until late. Implement a feature that allows the user to filter coffee shops and/or libraries by closing time.*

The hours that the place was open was something we wanted to include because we know it's a useful attribute to consider when choosing a location to work at. Adding a filter to sort the place by their hours is a good idea for convenience and just to make sure the place is open! We will implement this as something that can be sorted by in phase 2.

User Story: Recommend Coffee Shops Closest to My Location

> *I am a busy university student who likes to study at coffee shops. I would like to quickly retrieve a list of coffee shops closest to my locations. Implement a feature that allows the user to obtain a list of the 3 closest coffee shops along with their description (hours, rating).*

This is a really good idea and one of the initial reasons that led to us developing this idea. We'll have to look more into the feasibility (legally and technically) of asking a user for their location, but if everything checks out, we'll add this into the sorting/filter logic in phase 2. If we run into issues, we'll definitely still look into using the location to sort but in a different manner.

User Story: Custom User Locations

> *As campuses are quite large, it would be useful to be able to input your own location and search from there.*

This is a really good idea and one of the initial problems that we faced that led us into thinking about study spots as a possible idea for the project. We'll have to look more into the feasibility (legally and technically) of asking a user for their location, but if everything checks out, we'll add this into the sorting/filtering logic in phase 2. If we run into issues, we'll definitely still look into using the location of the places themselves to sort, but in a different manner.

User Story: Estimated Travel Time to Location

It would be nice to have an estimate for how long it takes to get to various locations like in google maps; i.e. walking, biking, by car, etc. Just having walking should be sufficient though.

This is a really cool attribute that we didn't think about including. We can include this extra information about transportation between locations in the instance page once we begin adding the connections between the different models. We'll look into adding these connections and extra information in phase 2.

User Story: Study Spots That Do Not Require Purchases

I am a university student who needs a place to study regularly. I would like to be able to find study spots that do not require purchases. Please implement a feature that allows the user to find study spots that do not require purchases.

This is definitely a sentiment that a lot of college students share. We will have to look more into whether or not there is any way to identify which coffee shops require purchases since that knowledge is more likely to be informal knowledge based on word-of-mouth of which places are more relaxed in their purchase requirements, so it might be out of scope. Nonetheless, we'll look further into this in phase 2 when we're starting to add more sorting and filtering logic.

RESTful API

Link to our API documentation: https://documenter.getpostman.com/view/23653833/2s83tGoBu1

Models

Universities

There will be roughly 200-300 expected instances for the Universities model.
Each instance will have the following sortable and searchable attributes: state, city, enrollment, names (alphabetical), size of the school, average tuition cost, year founded, type of college (public, private, community).
In addition to these, this instance will also have the following additional searchable attributes: phone, website, distance to nearest city, distance from me, zip code.
The different media types that will be on each instance page are the location of the university on a map, the link to the website, images of the university, and any videos that college might have.

Libraries

There will be roughly 500-600 expected instances for the Libraries model.
Each instance will have the following sortable and searchable attributes: ratings, location, business of the library, name (alphabetical), location, number of reviews, and hours.

In addition to these, this instance will also have the following additional searchable attributes: phone, website, amenities, distance from me, and size.
The different media types that will be on each instance page are the location of the library on a map, pictures of the library, library description, website link.

Coffee Shops

There will be roughly 500-600 expected instances for the Universities model.
Each instance will have the following sortable and searchable attributes: ratings, location, business of the place, name (alphabetical), number of reviews, prices, hours, local/chain.
In addition to these, this instance will also have the following additional searchable attributes: phone, website, wifi, amenities, and distance from me.
The different media types that will be on each instance page are the location of the coffee shop on a map, ratings, pictures of the library, website links, and library descriptions.

Connections

All of these models will mainly be connected to one another based on proximity in location.

Tools

Software

We used the following software to develop our website:

- **React** is an open-source frontend JavaScript library for development of UI components.
  - We used this to create the overall website UI design and structure.
- **React Router** is a library for routing in React, enabling certain behaviors like navigating among components and changing the browser URL.
  - We used this to create connections and allow navigation between the different pages.
- **React Bootstrap** is a library that has some preconfigured components developed by React.
  - We used this to add some components that it already had to our website.
- **AWS Amplify** handles the entire solution for frontend developers to build, ship, and deploy applications on AWS.
  - We used this to host our website.
- **Docker** packages all of the software needed for development (specified libraries, tools, etc.) into containers that can quickly be run and deployed.
  - We used this to create and synchronize the environment for development by having it include all the necessary installations.
- **Gitlab** is an open source code repository that helps facilitate continuous development and collaboration with other developers.
  - We used this for source control, collaboration with each other, and to keep track of issues that we and our customers come across.

APIs

We used the following APIs to develop our website:

- **GitLab API**: https://docs.gitlab.com/ee/api/
    - Information from this API was pulled to find out the number of commits and issues for each member in order to dynamically populate the About page.
- **Google Maps API:** https://developers.google.com/maps
    - Information from this API was pulled to find out more information about libraries. Generally, this API provides further information on all of our models.
- **Yelp API:** https://www.yelp.com/developers
    - Information from this API was pulled to find out more information about coffee shops.
- **College Scorecard API:** https://collegescorecard.ed.gov/data/
    - Information from this API was pulled to find out more information about different universities.

Hosting

We registered the domain name *studyspots.me* on NameCheap and hosted the React website using AWS Amplify. AWS Amplify is connected to our GitLab develop and main branch. It build sthe React website in the frontend folder and hosts the web application at that domain name. AWS Amplify connects to the domain name through the SSL/TLS certificate, resulting in the overall website being https://studyspots.me. The develop branch is associated with develop.studyspots.me and the main branch is associated with the domain name itself.

Design

Overview of Implementation

For phase 1 of the project, the website is structured into the different pages: About, Splash, Universities, Libraries, and Coffee Shops.
The Splash page and About page were mostly done separately since they had distinct layouts compared to the model pages. Thus, both of these pages were manually created by just structuring the different components on the page and providing any links to other pages if necessary.
The About page also had the added complexity of having to call the GitLab API. Thus, it first tries to call the GitLab API to receive all the information it needs (number of commits, number of issues, etc.) and then after that fetching is finished, the page is rendered.
The Universities, Libraries, and Coffee Shops pages were all similar in structure but would just contain different information. Thus, for these pages, a common component was created to make a model page template that can then be reused for each of the different pages, just filling it in with different information. Similarly, on each model page, there are also instance pages on each model page that will also be similar in structure. Thus, a generic instance page template was created for the individual instances to then be able to fill out the specific information but with the generalized template.

Challenges

One of the main challenges that we faced was learning frontend. Most of us didn't have that much experience with React, so there was a bit of a learning curve from trying to understand how React structures their design around the idea of components along with the same learning curve from just trying to understand HTML/CSS and knowing what properties to change and how to surround certain tags with containers in order to get the exact formatting desired. Most of this challenge was just overcomed by having to manually test out how assigning different properties or moving different components would generate changes in the website, so it was nice that React updated every time there was a code change since it made it very easy to just check what changes were made.

Another challenge that we faced was understanding how to best structure the entire project in order to be able to properly link together the different components. For instance, we had to decide how to best organize the model and instance pages, identifying that there is redundancy in structure amongst these but that they also have specific information tailored for each different model/instance.