CS-373 IDB Technical Report: Group 1

<u>Motivation</u>

The motivation behind the project is to allow users to find jobs based on geographical location and/or price and simplify the housing search for those looking to move, especially if they have a budget or preferred location.

<u>User stories</u>

Phase 1 (Received by our group)

User Story: Search for cities by job

"One feature that would be very useful would be the ability to filter/search cities by a specific job title(s). This way, a user could find all the cities they could work in and pick one to live in based on information on each city. Right now, you can search for jobs which contain information on what city they are located in, but it's extremely likely that multiple jobs will be located in one city, so it isn't very useful for finding what cities you could potentially work/live in."

This is a useful idea! It should be doable because we can get location data for each job posting and we can search for job titles in the API. We will implement this in Phase 3.

User Story: View City Job Demographics

"I'm a recent graduate who is looking to move somewhere new, but don't know what kind of job I want. Currently I can search for individual jobs in a city, but ideally I want to be able to view the top jobs in each city. Having the ability to view the top X jobs/industries in a city by quantity/pay/rating would allow me to get a better sense of where I want to live."

This idea seems to be challenging but worthwhile. We should be able to get information on the number of jobs within a certain industry in a city, as well as information about pay within each job and industry, which we can then either display on the city page or on the jobs page. We will have this implemented by Phase 3, once we have implemented sorting and searching.

User Story: City budget search

"Hi, I already have a remote job and I need to know what cities fit my budget. Could there be some sort of price search which will return the closest cities and apartments as well as similar ones? Maybe a range or average cost?"

This idea seems redundant with the ability to sort cities by budget score, then sort apartments by price within that city, which will already be implemented in Phase 3.

User Story: Implement job search by apartment

"*I'm a very materialistic person, so I've decided to search for my dream apartment first. I'll need to come up with the money somehow, so I'll need to get a nearby job. Implement the reverse of your proposal: given an apartment, show me all the possible jobs in the city that I'd have to get in order to financially support my dream of living in this apartment."*

This idea results in the list of jobs being sent back to the user being the same as that of the list of highest paying jobs, up until the point at which the income is not enough for the apartment. However, we will try to implement a reverse search in Phase 3.

User Story: Filter jobs by several categories

"I'm a recent college graduate looking for a job and I would like to be able to quickly find the best jobs that I am suited for. Specifically, being able to filter by experience requirements to find something entry level would be a huge save on time and on top of that filtering by location and salary would be great as well."

This is a great idea! Using multiple filters would absolutely be useful when searching for jobs that fit a specific skill set and situation. This will be implemented in Phase 3, when sorting and searching are implemented.

Phase 1 (Given by our group)

User Story - Search by Product

I eat a lot of fruits at home, and I'm getting pretty tired of the fruits I have near me. It's nice to be able to get the names of the markets near me and a list of the products they have, but I would like to only see the markets with the food I want. It would be nice to be able to search for a food product and be given the closest markets that sell it. Est - one hour.

"This is absolutely a feature we will implement! Although it's outside the scope of the current phase, we plan to provide such an option when we implement filtering, searching, and sorting of the model pages in Phase 3."

User Story - Find Markets by City

I live in Dallas, Texas, where farmers' markets are relatively far and few between. I like that when I use this service, I can see markets by the state I'm in, but I'm not sure how helpful it is for me to see a market from El Paso when it would be a 14-hour drive. I hope to soon have the ability to search for the markets by city, so I can find what's actually near me. Est. 1-2 hours.

"Thanks for the feedback! We haven't implemented it in this phase, but we plan to store location information of our markets with Google maps so that users will be able to sort markets by what is closest to them."

User Story - See Markets by Geographical Distance

I am a customer who is okay with seeing markets up to a 30 mile radius of where I am. It's nice to be able to see the markets by the state that they are in, and even potentially by their city, but I'd like to know what's nearby as well without being flooded with every market in my state. Filtering markets by their geographical distance to me would be really helpful, especially if I happen to live closer to an out-of-state market than one in my state. Est - 30 minutes.

"Thanks for the insight! Filtering by geographical distance should be possible when we use information about the location of our markets on Google Maps in future phases."

User Story - See Product Prices for the Markets

I am a potential user who is a little low on cash at the moment. I see that I can see the commodity prices when I look at the state's main agricultural commodities, but it might be easier for me to see it by the market I'm looking at. Being able to find commodity prices by looking at what the market offers and either getting the farmers' market's prices for it or the state's given cost for that commodity could help me budget. Est - 30 minutes if it's part of the dataset.

"Great point! I don't think our current APIs have that price information, but we could incorporate historical data to get accurate estimates of the farmer's market prices for any given commodity!"

User Story - List of Vendors

I'm a potential customer who really wants to support my cousin's beet business in a farmers' market. It's cool that I can see the number of vendors as well as the products that are available, but I'd like to know if the beets I'm buying are his special beets. I'd really like if there was a way for me to associate products at a farmers market to a vendor, or possibly get a list of vendors with their associated products at the farmer's market. Est - 1 hour if part of dataset.

"Thanks for the comment! Yes, while it's not available in the current phase, in a future phase we will obtain data from our APIs and that will give us a list of vendors at each farmer's market."

Phase 2 (Received by our group)

User Story: Given a job and city instance, describe my commute

"I have used your website to first find a job and then an apartment within the city. I would like to know what my daily commute would look like. Could you give some sort of indication of what I should expect when traveling to my job? It could be something like directions, duration of travel, recommended travel method (like walking, transit, car), or distance."

I think a lot of this is dependent on the data we're collecting, so we'll look into that once we establish how we're filtering things in Phase 3. The most likely is looking into distance.

User Story: Explore Nearby Cities

"I am someone who is living in a state where major cities are very close and I can theoretically travel to any. It would be beneficial to sort by distance from my current city, and get both apartment and job listings within a certain range. If this is too generalized being able to see the listings in the closest city or two would be helpful."

This kind of goes along with the filtering we want to get done in Phase 3, so we'll try it out then. It also goes along with the earlier issue from Brian where he wanted to see similar cities. We'll look into implementing this!

User Story: Display walk scores on the instance page

"Is there any chance you could put the walk, transit, and bike scores of a city on the instance page itself? I think people would have a better user experience if they didn't have to leave the site to check those ratings. You can still keep the link to walkscore.com, since it provides some additional information, but I just want to quickly be able to find a city's walk score by opening its instance page."

We'll look into collecting the walk score data and displaying it within this next phase.

User Story: Similar Cities

"I'm not sure if this is something you are filtering on already, but it would be cool if on the city instance pages, there were suggestions for similar cities with overlapping things that they're known for. This would be really nice if there was some reason you couldn't live in a certain city. then you could find alternatives quickly."

This kind of goes along with the sorting/filtering we want to do in Phase 3, so we'll work on this then.

User Story: Buttons for changing between models

“Right now on the model pages, there is blue, hyperlinked text to switch between various models. I think it would look a lot better if these hyperlinked text bullets were instead changed into some sort of GUI button that users could click. Right now, it looks fairly ugly, and changing it would help make the webpage look better.”

We just fixed this! Thanks for your input!

Phase 2 (Given by our group)

User Story - Clickable Instance Cards

I am trying to find a market in Texas to buy some food from. The current design of the website has the splash page pretty much only have a navigation bar that takes me to each page, but I'll like to understand the pages before I start navigating to them. For this, clickable instance cards for the pages might be great to have on the splash page so I can read about each page before choosing one to go to. Est - 1 hour.

“That is a great idea! We are hoping to get this implemented in Phase 3 as we switch from using a table to cards to display our information.”

User Story - Build navigation bar component

I'm a very visual user of websites and I like to have a good-looking and clean user interface when I work. It'd be pretty cool if I could see a navigation bar component at the top of the page with a good design. If it ran across the top of my screen, I'd find it easier to read and navigate through the website. Est - 1 hour.

“We have a navigation bar! We worked on making it look better in this phase.”

User Story - Images on Model Pages

I'm trying to find a market nearby and unfortunately, I can't read! It'd be great to not just have to click the state link and then click on Texas – I'd like the option to potentially have an image for the state I'm looking at so that I can recognize Texas! There's only 50 states, so why not organize by state on that page and have images? Est - 30 minutes.

“This is a great idea, we added images to the instance pages in this phase.”

User Story - Splash page with components

I'm a new customer and I'm a big stickler about the design of products I use. I'd like to not just see a navigation bar, but also components on your splash page that tell me more about the mission of the website, or at least give me a good introduction. I, too, want to support the farmers, but I need some react-bootstrap components to really push the message to me! Maybe even try revolving images? Est - 30 minutes.

"This is a good idea that we hadn't thought about yet. We will work on getting that on the splash page asap."

User Story - Cards for each developer

I'm a nosy customer and I want to easily know how much work everyone is doing. It's cool that you already list out all the issues you have, but I'd like to see a count of how many are solved by each developer and have that be under each developer's description. The same goes for their commits. Having a developer card could help you organize that in a cleaner way to improve the user experience and help me find that information. Est - 30 minutes.

"We worked on consolidating this information during this phase, so it should be on the updated about page."

Phase 3 (Received by our group)

User Story - Years of Experience

"As a job seeker, something that would really save me a lot of time is to know how many years of experience a job requires. I don't have much business applying to a job I'm not qualified for and it's definitely one of the first things I filter on when searching for jobs. I currently don't see this on the model or instance pages, so that would be a helpful addition."

Thanks for the suggestion Brian, this is something we will look into for the next phase. Maybe even instead of years of experience, we could also try categorizing between full-time, part-time, and internships.

User Story - Looking for more information on scores

"As a web user who is unfamiliar with your API sources, I do not always and easily remember what certain scores mean. Currently for example a city has multiple scores that I have to remind myself what they really represent, it would be useful to have a tooltip somewhere (on hover, at the bottom) that explains what each means and maybe links to further reading. It is also is hard to gauge a city at a quick glance, coloring each score to distinguish them (red = bad, green = good) would make it easy to view this information at a glance."

Thank you for your input, we think this is a good idea. It could potentially be enough to explain scores somewhere at the top of the page, but the idea to color the text of the score can also be a great indication of what it means. We'll try to implement some sort of indicator of what scores mean in this next phase.

Three more user stories have not been provided by the Customer group yet.

Phase 3 (Given by our group)

User Story - Cards for Data Sources

I'm a customer trying to use your website to find data sources about markets, states, and produce. I'd like to know more about your data, and I went to your about page to see more. I think y'all do a good job of telling me what the data sources are, but it would be visually more appealing to see it in a clickable card and maybe even have an explanation on how the data is used, now that we're doing filtering and searching. Est - 20 minutes.

No response yet

User Story - Cards & Links for Tools Used

I am a customer who wants to build a website of my own and I'd love to use your About page as a reference. However, I have no idea how to find the tools you used because, although described, they are not linked on your page. I'd like to see links to some sort of resource regarding your tools, whether that's documentation or something else. It would look even better with cards! Est - 30 minutes.

No response yet

User Story - Loading Visual

I am a customer with really bad internet at home. When pages freeze, it often makes me wonder if my wifi is slow or if the website is just unresponsive. I see a little "Loading..." text on only the State page, but it would be nice to have visual indicators of when my page is loading. Because your Markets and Produce pages are not up right now, it took me a little bit to realize that it was just down, and not that it wasn't loading on my laptop. Est - 10 minutes.

No response yet

User Story - Search Component on Home Page

I am a customer who isn't sure what to start with, and I like to be provided with my options up front. I already wrote a story about having page cards on the Home page, but it could also be helpful for there to be a drop-down input group search bar on the home page. It could allow me to search by State, Market, or Produce right off the home page. Est - 1-2 hours.

No response yet

User Story - Website Title and Logo

I am a customer who also has an IDB project and I have 30 tabs open on my screen that all say "React App" with the same default React logo. It is quite difficult for me to find your website's tab quickly because it looks like all the others. A good change would be to create even a simple logo to replace the image, as well as change your site's tab name to something like "Find My Market" instead of the default "React App". Est - 10 minutes.

No response yet

RESTful API

https://documenter.getpostman.com/view/23607622/2s83tFHBCK

Models

City - This model provides all around information about living in a given city, including: population, rating, budget score, safety score, ability to get around on foot, bike, or public transit, images, and various tags on what the city is known for.

Apartment - This model provides information about apartments in a given city, including: location, rating, number of reviews, phone number, and images.

Job - Provides information about any job given a job title or industry, including: company, expected salary range, location, job description.

Tools

General

Discord: online communication platform
GitLab: git software that allows for managing version control and continuous integration/ continuous deployment
Visual Studio Code: code editor
Zoom: online communication platform

Frontend

AWS Amplify: platform for deploying and hosting web applications, supports continuous
Axios: HTTP client library, used to access APIs from React application
Docker: Used to create environment to be able to use tools, Flask and SQLAlchemy
Material UI: provides React components that implement Google's Material Design standard
NameCheap: domain name registrar, allows purchasing and managing domain names integration and continuous deployment
Node Package Manager (NPM): manages installation of node packages
React: JavaScript library for building user interfaces, powers the front-end
React-Bootstrap: CSS framework built on React, simplifies integration of the frameworks
React-Highlight-Regex: package for highlighting text in components based on RegEx matching
React Router: library for routing in React, enable navigation between pages
Semantic-UI: Provides React components (Item and Divider) to better display information for pages.

Backend

AWS Certificate Manager: Used to provision, manage, and deploy public and private SSL/TLS certificates for use with AWS service
AWS Elastic Beanstalk: An orchestration service offered by Amazon Web Services for deploying applications which orchestrates various AWS services, including EC2
AWS Relational Database Service: Supports an array of database engines to store and organize data and helps with relational database management tasks, such as data migration, backup, recovery and patching
Flask: Web framework used for implementing the API
Jest: Testing framework for JavaScript codebase, used for frontend testing
MySQL: Tool used to manage databases and organizing data within them
Marshmallow: Python library used for converting complex data types to python types
PlantUML: Open source tool for creating diagrams from plaintext
Postman: API platform for documenting, building, and using APIs
Selenium: Testing tool used to validate functionality of web applications
SQLAlchemy: Toolkit that allows Python programs to interact with databases
Unittest: Library for testing backend Python code and used in conjunction with selenium to test the front end
Waitress: A Web Server Gateway Interface used as calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language

Data Sources

RoadGoat: get information about cities
Google Maps Embed: get maps showing where apartments and jobs are located

Realty Mole: get information about apartments
Adzuna: get information about jobs
GitLab: version control and issue tracking, gets information about commits and issues

Frontend Hosting

First, we registered the domain *geojobs.me* on NameCheap. For hosting, we used a tool called AWS Amplify, which is part of Amazon Web Services. We had to set up SSL/TLS with NameCheap and Amplify. This allowed us to connect different branches of our GitLab repository to different deployments, which we could then associate with different CNAME records on NameCheap. We set dev.geojobs.me to point to the development branch and geojobs.me (and www.geojobs.me) to point to the main branch.

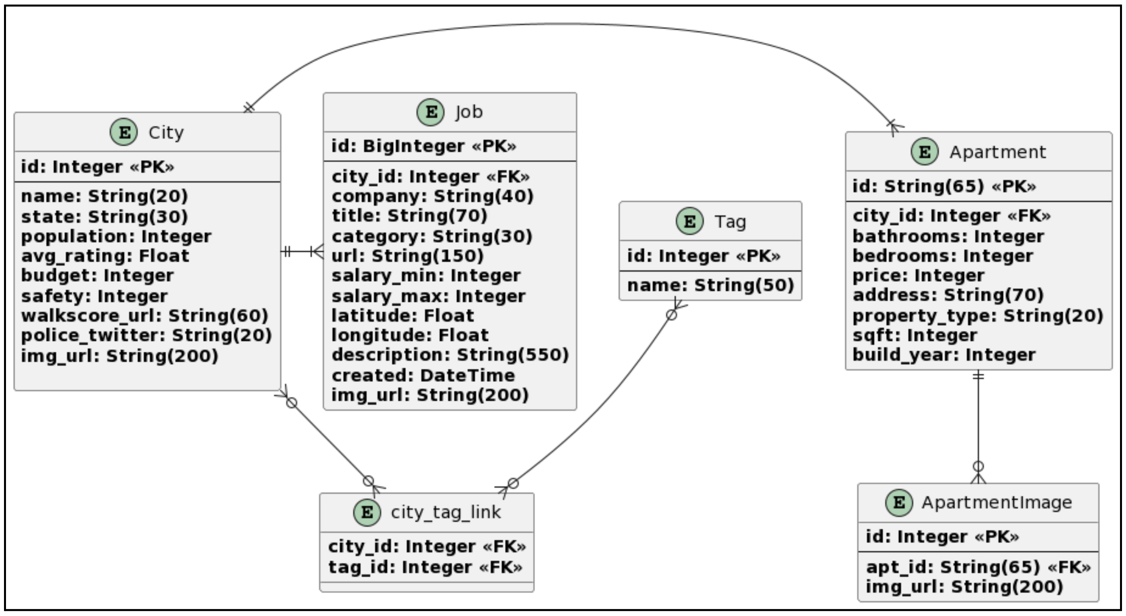
Backend Hosting

We created a MySQL database and used AWS Relational Database Service to host it. We used Waitress as a production-grade WSGI server for our Flask application that runs the API, and then hosted that on AWS Elastic Beanstalk using Docker on Amazon Linux. To allow HTTPS connections, we obtained an SSL certificate from AWS Certificate Manager, and then set up a listener for the application load balancer on port 443 that used the previously acquired SSL certificate and forwarded HTTPS traffic to the Flask application running on the Waitress server. After configuring the security groups for RDS and Elastic Beanstalk to accept incoming traffic on their respective ports, we were able to access these services programmatically.

Database Implementation

For our database implementation, we first created models for the different types of data we would be storing in our database and set up relation tables in Flask-SQLAlchemy. We then requested data from various APIs and stored them in JSON files. For additional information that we could not obtain from the APIs, we used Selenium to scrape the web and store this information in supplementary JSON files. To populate the database, we fed these files into a Flask-SQLAlchemy application that created the tables and added the instances in the JSON files for each model. In total, we had 6 tables in the database (as shown below): 5 for the models and 1 to hold the many-to-many relation between cities and tags.

UML Diagram for Database



Pagination

Pagination was implemented in the backend, where the front end would pass a page number and a limit per page as arguments in the request to the backend, then the backend would only return a number of instances up to the limit and return a set of instances that corresponds to the page number.
On the front-end, react-bootstrap pagination components were added to the top of the Jobs and Apartments pages.

Sorting

Sorting was implemented in the backend. The frontend would pass an attribute of the model they wanted to sort in the ‘sort’ argument in the call to the backend. The backend would then use .order_by() to sort the items in the query. By default, it was sorted in descending order, but ‘asc’ could be passed as an argument to change the sort order to ascending.

Filtering

Filtering was implemented in the backend, where the frontend would pass the object they wanted to filter with and then the corresponding value they would like to filter by. The back end then returns the instances that correspond to the value that the front end requests to filter by.

Searching

Searching was implemented in the backend at first, where new API calls were added for searching individual models or searching throughout the entire database. The user query was split into individual subqueries, and each query was run with an “or” operator to find any instance that matched any of the terms in the query. The results were then sorted by numbers of times they were matched against, so the instances that were most relevant to the query were ranked the highest.

The frontend was then updated to use the search query by creating the appropriate URL to make a request from the API based on the user’s input, then building a regular expression to match the user’s input as well. This regex was used to highlight the user query terms on the resulting models’ cards.

Visualizations

The visualizations were created using recharts. The data was collected and formatted into a dictionary using a python script that called our and our provider’s API. The data was then hard coded into the visualizations page to improve the performance of the page.

Challenges

It was hard for us to get started with React and Bootstrap, so there was a learning curve before we could really do much on the front-end of the site. We had difficulty finding apartment APIs that gave us the information we wanted: for example, we tried Zillow, but it wasn’t quite working for what we wanted to do. At some point, the AWS Amplify project was not redeploying when our main branch was updated, so we had to manually redeploy instead of it happening automatically; identifying this issue took a lot of time though, as we were not familiar with the intricacies of AWS Amplify.