CS 373 IDB Project Technical Report: FuturFindr

***Group 17**: Seth Parsons, Frank Sosa, Josiah Valdez, Ritesh Thakur, Saiprathik Chalamkuri*

Motivation

The motivation for this project is allowing users who want to plan for their future the ability to see the three most important things - career, housing, and education - all at the same time. Since these three things are inherently intertwined and broadly applicable to most people's futures, it should be pretty useful - having to consider all three separately is hard enough, so we want to save people time (and chrome tabs) by showing them how their decision in one area could affect their options in the other and vice-versa.

User Stories

Phase 1

Received from our Customer (and 5 we created for ourselves):

1. *"I am a college student and want to find a part-time job nearby. I want to be able to filter by location so that I can find a job nearby and make sure it's part-time so that I can also focus on college work as well. Additionally, a slider for the pay range would be a bit annoying... maybe keep a min text box?*

   These are all filter options. Should be pretty straightforward to implement in the appropriate phase - phase 3.

2. *"I am entering my senior year of high school and I want to look for colleges near my city, preferably in my state in the US. I want to be able to filter colleges (or other higher education institutions) by location and maybe tuition (combine the filters) so that I can choose the best institution for me based on these factors. Additionally, a slider for the tuition range would be annoying; maybe keep a min/max text box?*

   These are more filter options that will be implemented in phase 3. A maximum and minimum box is an interesting idea that has now been requested twice - for things with numbers, maybe we should have boxes and/or sliders for all of them?

3. *“I am moving to a new city and need to find housing nearby as soon as possible. I want to be able to filter by location, rent/lease/price, and neighborhood rating. Also, I want to make sure it's near my job, so maybe filter by the location of that job too.*

   This request wants us to have our filtering system work between models - for instance, being able to choose a job and have housing filtered such that it would be applicable to that job. This could be challenging, depending on how complex they really want this to be. This will be implemented in phase 3.

4. *“I am in my 30's have no idea what I want to do. I don't know whether I should continue my education, find a new job, move locations, or whatever. I want this site to display all the data for these 3 areas of my life where I have no clue how to further so I can plan, or find, my future. Maybe if you could show pictures of the city, that would incentivize me to move there?”*

   This is a multimedia function that may not be covered by our current APIs. We’ll implement this in phase 2 and possibly consider adding an additional API if our current ones don’t include pictures of the city.

5. *“I am all about the MONEY and want to find the best 'software engineering' job in my location as well as filter by salary. Also, the company that I would work for needs to be common enough to show up on all job sites so that I can figure out which is the best place to apply from. Additionally, a company rating wouldn't be bad.”*

   The filtering by salary should be already covered by our current development plan, but being able to pick out companies specifically is a good idea that may need us to add more options than we thought. This will be implemented in phase 3.

6. *“I am a freshman and I want to transfer to a better college. However, I don't want to pay more than what I am paying right now. I want to be able to filter colleges by ranking and tuition so I can choose a better college with similar tuition. Maybe I want to see which majors have the highest ranking in that college, so I can check whether I can transfer to my current major.”*

A ranking system will be filterable, but not sure about ranking by major because of the uniqueness and how difficult it would be to access an API for that. Still, we will try our best.

7. *“I lived in a dormitory last year, but I want to live off-campus for the upcoming semester. Unfortunately, I don't have a license, so I need to take a bus or walk to school. So, I want to check whether there is transportation from an apartment to my college. Or, I just want a filter based on the distance from my college, so I can check whether the apartment is walkable from my college.”*

   Yes, that's the great part about our website: you will get to see which colleges are literally in the city you live in! Addresses will be given for each too, so maybe we could work on some sort of way to show the distance from the currently viewing apartment to close colleges.

8. *“I visited my friend in Chicago last month and it was a wonderful place, so I decided to move to Chicago. Before I move to Chicago, I want to get a job and find an affordable apartment based on a new salary. So, I want a filter that can check a salary of a job and a price of an apartment at the same time. Also, it would be better if I can pin jobs and apartments so I won't have to find my favorite jobs and apartment.”*

   We will do our best to implement a filter by location so that the job and housing are in the same place and so that the salary is desirable.

9. *“I am very busy and I don't want to spend much time searching for an apartment. I chose a place to live, so I want a website that recommends an apartment based on rating and price. Also, after I choose a certain apartment, I want to know apartments that have similar prices and ratings from that location, so I can save time.”*

   Ratings are not currently part of our dataset but will be soon. We will add sorting sliders and text boxes to enter your desired ranges for these data to filter out housing locations.

10. *“I am a student majoring in computer science, and I am looking for a software engineer internship for this summer. I want to see whether there is an open position near my apartment or college. So, it would be good for me if your website provides an open position for each company. Also, I want to see ratings and reviews of that companies, so I can choose the right one.”*

    Yes, the page for each intern position will show the location as well. However, a rating is not part of our data set, so we will provide links to well-known sources that show people's ratings on their experience at this position (internship).

Given to our Developer:

1. *“I want to go to a game with my friends but we don't want to spend too much on transportation, so we want to look for an event near our city. Also, once we find the sporting event, we want to learn about the players so we can know what's going on in the game.”*
2. *“There's this famous football player that apparently made some controversial tweets. I would like to see them through this website after filtering for this player and clicking on their card to see (or get a link to) their social media.”*
3. *“My favorite team will play this week against their rivals! I want to be able to see a history of games against their rival so that I can predict whether they will win this time or not. Also, a win-loss ratio for the current season won't hurt!”*
4. *“I've found an event nearby and I want to go there to the live sports event. I want to have a link straight to the page where I can book seats for that game. Also, I should be able to share this link with my friends.”*
5. *“For an individual athlete, I want to see their stats for their team and their past teams so that, when they get traded, I can see how they did on previous teams and I can predict how they will do on this team. Also, if it's possible, have instances to links to similar players in the same sports realm.”*

Phase 2

Received from our Customer:

1. *“I am a student who hates socializing and needs time to stay alone, so I am looking for a 1x1 apartment, so I do not have to share my apartment with other people. Can you*

create a filter for your website, so I can search only 1x1 apartments?"

We were given lots of data from our API's but hardly any data on apartments. For those that did, they gave bedroom and bathroom counts per listing, so yes, you will be (in the next phase) able to filter by them.

2. *"I will move to Austin, and I am looking for a new apartment. I love your website, as I can see the nearest apartment to my job. However, can you create a map, so I can see the exact location of the apartment?"*

   We are working to implement the Google Maps API to show the exact location of the apartment. Hopefully, we can cross this off our To-Do list soon!

3. *"I am so lazy, so I don't want to drive for more than 10 minutes. So, I want to see facilities near the apartments so I can reduce my driving time. I want an apartment that I can drive to any market in 10 minutes. Also, more restaurants will be a better option for me. Can you provide that information so I can see which facilities are near my apartment?"*

   While we don't have access to store location APIs, we are working to figure out a solution that lets a user sort through housing options based on location, and some locations are more congested than others, so they are bound to have more stores and restaurants nearby.

4. *"I am a gym rat. I live in a gym, and I don't want to waste my time going to a gym. So, I want an apartment with a gym. So, can you provide the information on whether an apartment has a gym or not?"*

   The link provided to the housing posting will give you full details about that housing option. The website the link redirects you to should also have gym information.

5. *"I am a rising senior, and I want to move to another apartment. My previous apartment does not have an indoor washer and an indoor dryer, so I need to take my laundry to my friend's house. My friend graduated last semester, so I need an apartment with a washer*

and dryer. Can you provide me with whether an apartment has a washer and dryer?"

The link provided to the housing posting will give you full details about that housing option. The website the link redirects you to should also have washer and dryer information as well.

Given to our Developer:

1. *"I enjoy watching basketball and keeping up with all the statistics that relate to the NBA. I was wondering if there is a way to compare and filter players and their statistics based on whether they are active in the NBA versus all time. This would make looking at data holistically easier for me."*
2. *"I want to be able to see my favorite teams in the western and easter conferences based on different stats like most steals or most points made rather than just wins and losses. This would make for a more interesting analysis of different teams."*
3. *"I want to be able to plan for my events in the best way possible which includes trying to find parking at the venues. I want to be able to learn more about the venue prior to reaching. This will help me enjoy my events the best way possible."*
4. *"I love my Dallas Mavericks and want to be able to get the most out of my clicks on Sports Now. I want to be able to see more detailed information like recent trades and overall championships and statistics related to that. I want to be able to consume all of the information that I want that is available."*
5. *"I want to be able to see players related to a specific team when looking at a team or vice versa. The same would be nice to see if we could see the venue related to the teams and players playing. Having all this information consolidated in one area would benefit the user looking to find all the information."*

Phase 3

Received by our Customer:

1. <INSERT USER STORY FROM CUSTOMER>
2. <INSERT USER STORY FROM CUSTOMER>
3. <INSERT USER STORY FROM CUSTOMER>
4. <INSERT USER STORY FROM CUSTOMER>
5. <INSERT USER STORY FROM CUSTOMER>

Given to our Developer:

1. *“User Story #11 Looking for a specific Player: I am an avid sports enthusiast. I have interest in keeping up with a few players in the NBA. I want to be able to find lebron and his stats quickly through searching.”*
2. *“User Story #12 Looking for specific Team: I am a huge Dallas Mavericks fan. I want to be able to quickly find it when searching. This should be as easy as typing up the words "Dallas Mavericks" and I should see the results.”*
3. *“User Story #13 Searching for Specific Events: I am an avid sports watcher. I want to find out more about an upcoming mavericks game. Instead of looking though all of the events available, I want to be able to search for that specific one.”*
4. *“User Story #14 Filtering Events: I am a busy guy and only have limited time to do stuff I like. I am busy Monday through Friday and only have time to do stuff I like in the weekends. I want to be able to filter through events that are only available during the weekends.”*
5. *“User Story #15 Filtering Players: I am a basketball fan. Although I appreciate other sports, I only want to see basketball players when browsing Sports Now. I want to be able to filter out players who are not basketball related.”*

Restful API

https://documenter.getpostman.com/view/25807396/2s93CExcaC

- **GET all colleges** returns all of our college data. It has (optional) **page** and **per_page** parameters for pagination.
    - **page** and **per_page** are integers
- **GET all housing** returns all of our housing data. It has (optional) **page** and **per_page** parameters for pagination.
    - **page** and **per_page** are integers
- **GET all jobs** returns all of our jobs data. It has (optional) **page** and **per_page** parameters for pagination.
    - **page** and **per_page** are integers
- **GET college by ID** returns a detailed JSON object of the college with that **ID**.
    - **ID** is an integer
- **GET housing by ID** returns a detailed JSON object of the housing with that **ID**.
    - **ID** is an integer

- **GET job by ID** returns a detailed JSON object of the job with that **ID**.
    - **ID** is a string
- **GET search all** returns college, job, and housing records based off of search parameters in the query
    - **Query** is a string
- **GET search model** returns college, job, or housing records based off of search parameters in the query
    - **Query** is a string

Three Models

Colleges - the model which provides all information regarding colleges or universities someone would want to attend, including location, tuition, and acceptance rate. Relation to jobs: one may want to have a part-time job near their institution to make money and learn, or one may want their kid to go to a nearby institution as the job they are looking for. Relation to housing: one may want to have housing nearby their college.

Jobs - the model which provides all information regarding employment, including company, salary, location, description, and link to the application. Relation to colleges: one may want to grab an MBA nearby their job already so distance wouldn't be a factor. Relation to housing: one may have gotten a promotion and may want to look for housing nearby their new location.

Housing - the model provides information regarding housing, including location, property type, images, cost, and beds/baths. Relation to colleges: one may want to live near their college. Relation to jobs: one may want to live near where they work (or will work).

Tools

General

- **Microsoft Teams** - application for team communication
- **GitLab** - repository location for our project that allows for source version control and CI/CD
- **Visual Studio Code** - text editor that the entire team used to work on the project

- **Docker** - a platform that uses OS-level virtualization to deliver software in containers that allow teams to work in a standardized environment with the necessary tools, used for both frontend and backend

Frontend

- **React** - main javascript library used for our front-end
- **React-Bootstrap** - a javascript framework works with react to power front-end
- **NameCheap** - where we bought our domain name
- **AWS Amplif**y - the service that hosts our React application
- **Axios** - allowed us to contact both GitLab API and our own API from within react app
- **Jest** - frontend tests in general
- **Selenium** - create GUI tests that use click functionality for url redirects

Backend

- **AWS Certificate Manager** - issued our SSL certificate for using HTTPS
- **AWS Relational Database** - hosted our PostgreSQL database
- **AWS EC2** - hosted the linux virtual machine that runs our backend, also runs a load balancer that allows connecting through HTTPS
- **Flask** - web framework that provides tools and features for making web applications in Python
- **SQLAlchemy** - a Flask extension and toolkit for efficient database access
- **Unittest** - library to test source code
- **Selenium** - testing tool to validate functionality of web applications
- **Marshmallow** - flask extension for additional API tools
- **PlantUML** - open source tool used to create UML models for databases
- **PostgreSQL** - relational database and management system
- **pgAdmin4/psql** - administration and development platform for PostgreSQL databases (psql is the CLI for PostgreSQL)
- **Postman** - used for testing/documenting APIs
- **JavaScript (axios)** - very helpful for gathering all the API data
- **Black** - use for formatting python code

Data Sources

*(Subject to change; more APIs may be added in the future)**

- **RealtyMole.com** - housing info and images
- **Adzuna.com** - job/employment info
- **CollegeScorecard.ed.gov** - college info
- **Google Maps API** - turning addresses into latitude and longitude
- **Google Custom Search API** - images for colleges and jobs
- **Gitlab.com** - on top of version control, also where we get contributor, commit, and issue info

Frontend Hosting

Once we had decided on futurfindr, we attempted to get the futurfindr.me domain off domaincheap for free using their ‘Free Domains for Students' program. Though a couple of us tried to grab it, we would all get stuck at the email verification step, so Seth just bought it (for a whopping $8.90). After we had registered the domain, we transferred it over to AWS, where we used Amplify to host our React application. At that point, we followed a YouTube tutorial to have one of our repository branches deploy automatically to AWS when pushed to. Not too bad to figure out, at least once we had secured the domain. During phase 2 we also deployed a second development branch so that we could make sure the changes we were making had no issues being deployed while not affecting the main website URL.

Backend Hosting

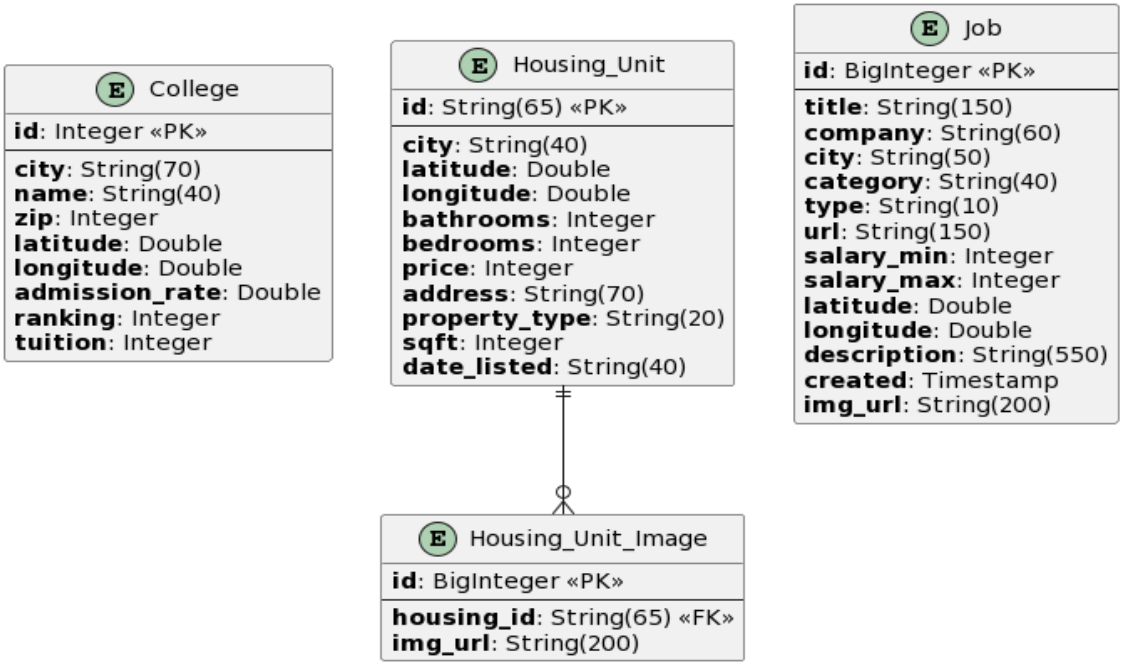
The first thing we did during phase 2 is create a PostgreSQL relational database and host it on AWS using their RDS service. After that, we scraped data from our APIs and wrote the dp.py script to insert that data into the database - only then did we start to play around with writing our API in Flask. Once the API was fleshed out enough and ran locally, we used Amazon’s EC2 service to host our backend for production use. Once running, we realized our API could only be accessed through HTTP and not HTTPS, which was a requirement - after unsuccessfully trying methods to have our Flask application accept HTTPS natively, we instead looked at AWS’s load balancers for EC2 which could be used to convert HTTPS traffic to HTTP for our Flask application. We went through the process to have an SSL certificate issued by AWS for api.futurfindr.me, and after

we hooked that up to the load balancer, the backend was more or less ready to go.

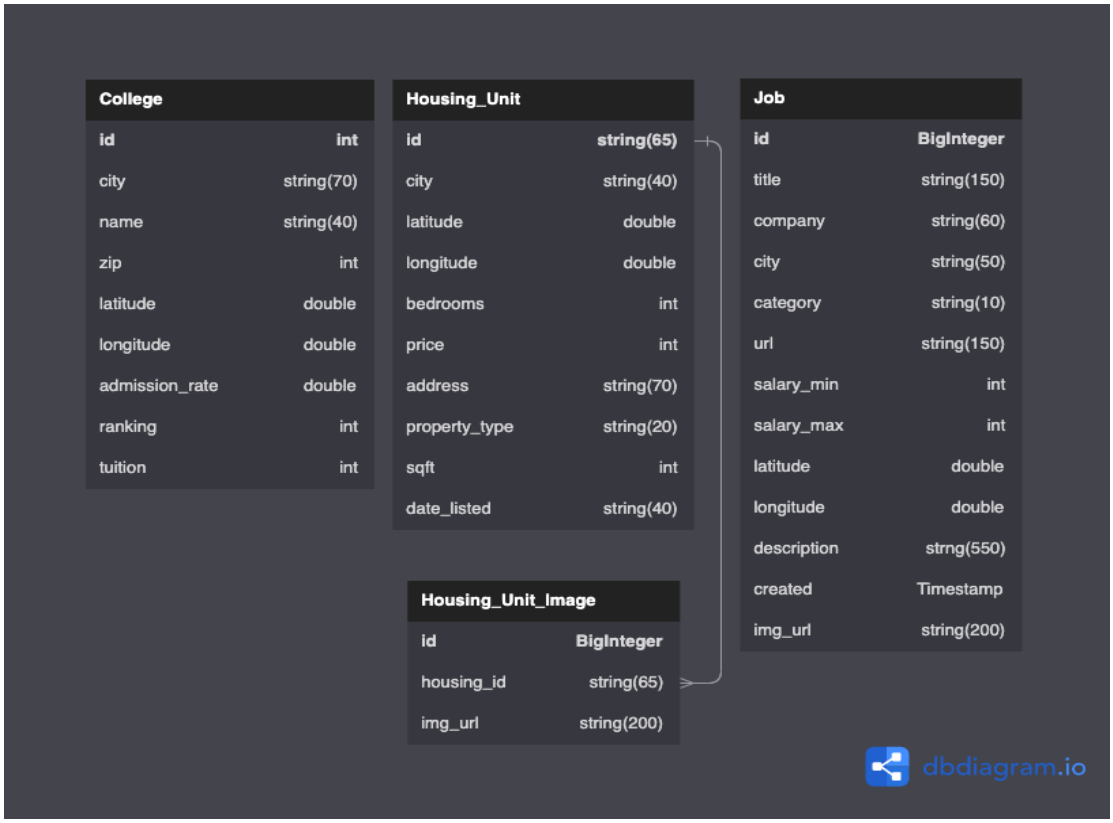
Database Implementation

We used the models that we formulated for the project as the types of data that we would be storing/fetching from the database. We requested data from the selected APIs as JSON files, which we then parsed and populated the respective tables using Flask-SQLAlchemy.

Database UML Model



Database Diagram



Pagination

Pagination on the backend was pretty simple - we created a helper method in app.py that takes a query, a page number, and a number of items per page and uses SQLAlchemy's query.paginate() function to return the right items. If an API call includes a page number or the number of items per page, the method gets called.

Pagination on the front end was relatively easy to figure out after getting familiar with how React keeps track of state for components. After the backend was implemented to support pagination as well it was just about making the right GET request depending on the pagination item selected. The React bootstrap pagination component was also easy to work with and the only challenge was in

making it so that only relative page numbers were displayed so that the pagination component did not overflow the screen.

Sorting

We focused on sorting on things like tuition, and admission rate in the colleges model and having the option to view it ascending or descending. Similarly, in the jobs page, we sorted by the salaries, maximum and minimum as well as in ascending and descending order for those respective things. On the back end, this just meant adding a couple lines of code to order any query results by the “sort” and “asc” api parameters. In the front end, we made drop down menus for the respective sorting features, which allows the user to choose how they want to see the respective results.

Filtering

Filtering on the back end was simple, thanks to built-in SQLalchemy functions that more or less just meant taking in a parameter from the API call and appending it to a database query.
Similar to the other two, on the front end, we had a few filtering options, whether it be by city, or by a rangeslider for tuition for colleges or salary for jobs. We created the respective user interface to handle these requests, and had to store lists for things like the cities in each model to be able to have a dropdown menu show all the available options to filter by. And we send out the appropriate api call to the backend and display the results accordingly.

Searching

While searching on the back end took a lot of lines of code, it was conceptually easy to understand and implement. It just meant taking in data from the api call, converting it to whatever data types we needed, and creating a search query that looked for any instance of the search data in any row of the given model. For the general search, we just did this for every model, and returned any and all results the SQL query(s) returned.
On the front end, we had the option of having a global search which would give the user results for all three models and individual search bars in each respective model page. It would work by taking in the user input and making the search to the api call to the back end and displaying the results accordingly.

Challenges

- Inexperience! Basically, none of us had used React or Flask before so there was a lot to learn about. Without YouTube tutorials and the many references and examples we were given, it would've been impossible to put this together in time.
- Like we said in the hosting section, getting the domain was annoying and the free methods didn't work. This forced us to buy the domain.
- APIs, APIs, APIs! We had also never dealt with Rest APIs, so it took a minute to learn what postman does, on top of actually finding good data sources that weren't paid.
- Finding a source of images for our colleges or jobs was a challenge, as they didn't come with our other APIs. We had to look into a lot of choices - mainly Bing and Google's search APIs that could return images - which were either paid or deprecated somehow. After a lot of time, we learned about Google's Custom Search API, which can return images and comes with a free trial, so that's where we ultimately got our images for colleges and jobs.
    - Additionally even after finding a college api, some of the links associated did not have https at the start, making it so that they were treated as relative paths. To fix this we had to append the "https://" on the frontend whenever the link did not start with it.
- Coordinating work was a particular challenge during phase II - everyone's availability just didn't line up very well and other classes got in the way of putting in as much work this time as we would've liked.
- It was pretty annoying to have to figure out how to cast/convert data from backend API calls to be used in searches, as updates to SQLAlchemy have changed how previous groups did it (specifically changing strings into numerics that SQL can actually compare with floats, ints, etc).