State Finder

Alyiah Proctor, Anne Nguyen, Darlyn Mendez, and Emily Crabtree

Overview

Our website aims to help others in their location and career decisions by providing users with initial information regarding livable areas based on a specific income. We ask users for their potential career, the seniority level, and household demographic data. With that we populate a map with livable areas based on the residual income they'd have in the location. Users are also provided demographic data on each metro area to learn a bit more about the places they could choose.

Initial Information

Languages used

- Python, HTML, CSS, Javascript

Databases used

- Azure SQL Server

Deployment method

- Heroku

Livable areas subsystem

- Data pulled from first html page
- User income is calculated by pulling the occupation data
- Living costs calculated by pulling the corresponding living wage data.
- Residual income is calculated by : user income - living costs.

- Map population
  - Geojson file used to provide shape boundaries for metro areas
  - Plotly used for map creation

Map Infographics Subsystem

- Composed of US Infographic page and User Selected State Infographic page
  - Information shown in hover box includes:
    - Total population of state/metropolitan area
    - Total housing units of state/metropolitan area
    - Breakdown of occupied and vacant housing units
    - Popular attractions in each state

- Plotly used for choropleth map generation (heat value based on population)
- Data and geojson comes from US Census Bureau

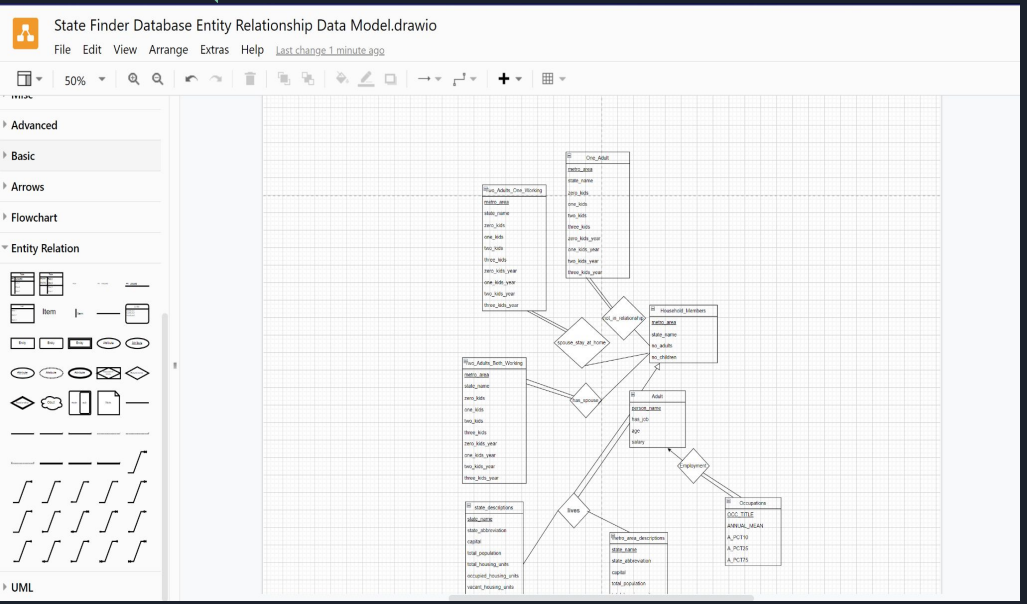
HTML Subsystem

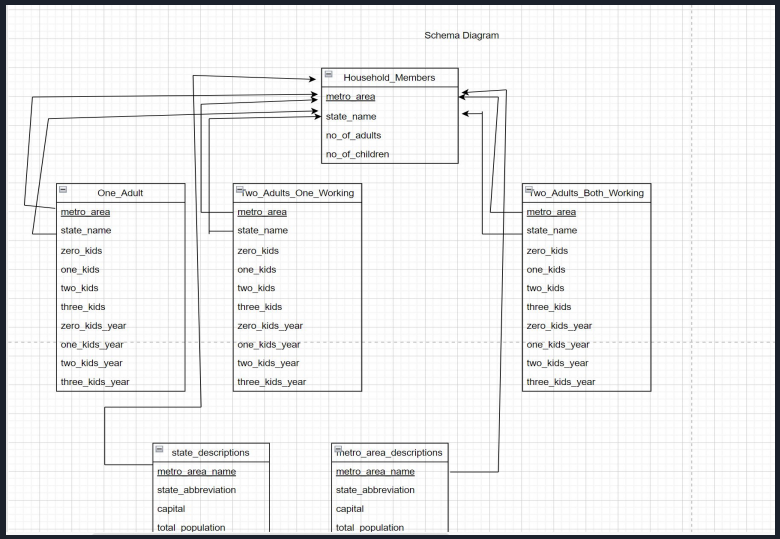
- Written in HTML, CSS, and Javascript
  - Originally included Bootstrap
- Index page
  - Collect user information
    - Occupation
    - Job level
    - Working adults
    - Children (0-3)
- About page
  - Combined with resource page
- Contact page
  - Send admins an email
  - Auto email response
- Login page
- Design choices:
  - Dropdown menus
  - Error messages on contact page

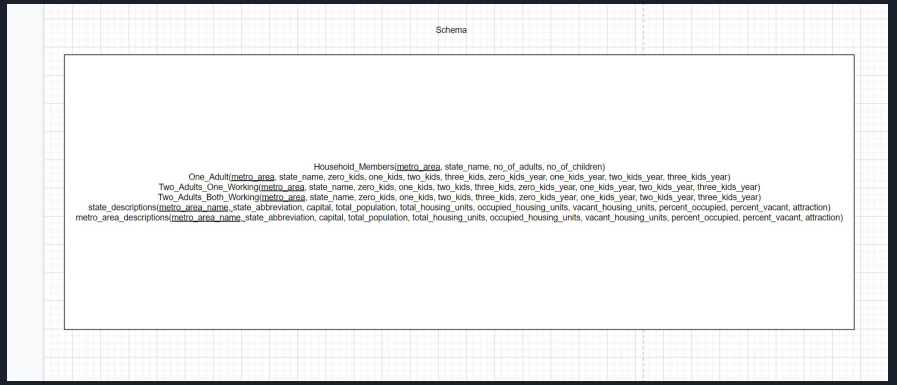
Database Subsystem

- The Initial Design and Model of the Database includes an ER Diagram and a Schema Diagram that would help us restructure the csv files that we have into tables so that we can use it later on.
- The design choices that we made is to have the “metro_area_name” be the foreign key constraint that connects all of the entities. Some of the metro area names have the same name such as Jacksonville so we added a suffix such as “_one”.
- We used diagrams.net and its UML shapes to create the ER diagram and the schema diagram.
- Data dictionary includes attributes from these entities and their data types.
- The changes from the initial model include the addition of the state_descriptions entity and the metro_area_descriptions table.
- We used an Azure database and made changes in Azure Data Studio.
- Language - SQL Server
- Database has been tested by providing the azure database credentials through an ODBC driver in the Flask App.
- During deployment the ODBC driver had to be changed to pymssql - db api due to some issues.

Database Subsystem







Login Subsystem

- Used firebase docs as a reference for implementation.
- We were initially focused on google authentication, wanted to let users have other options.
- Faced issues with Facebook and Twitter Authentication when initializing the firebase app multiple times in order to get the login popup.
- Coding - Used Javascript to implement the login subsystem.
- Firebase provided some javascript sdks when we created the project.
- Had to make sure the firebase app link to the developer sites in order to connect it to firebase.

Login Subsystem