State Finder Oral Presentation 2

Alyiah Proctor, Emily Crabtree, Anne Nguyen, Darlyn Mendez

Design Perspective Subsystems

- Database Subsystem – designed to break down livable wages by Metropolitan area and other factors, and average occupation salaries by job level
- User Interface Subsystem – HTML files that create aesthetically pleasing and user friendly interface
- Python Subsystem – Python files used to compare user data with data from database

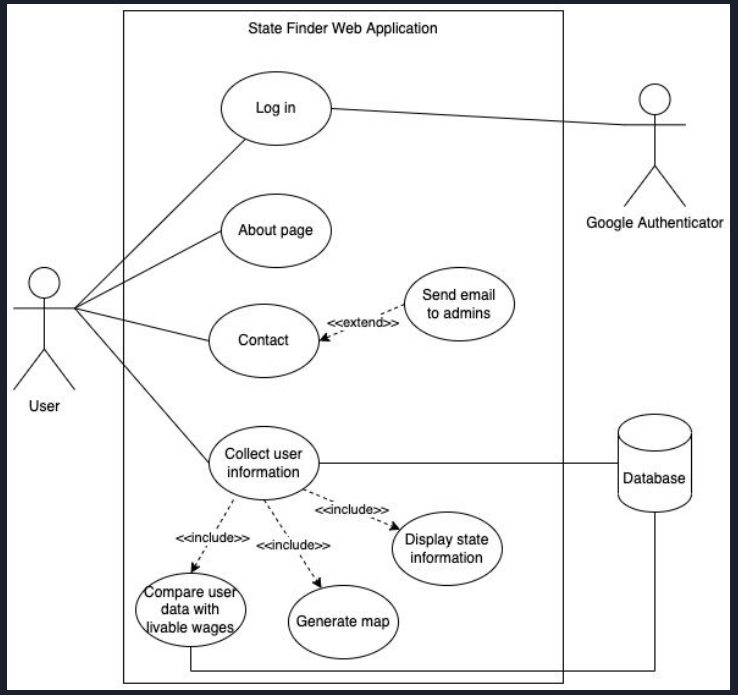
Design Choices

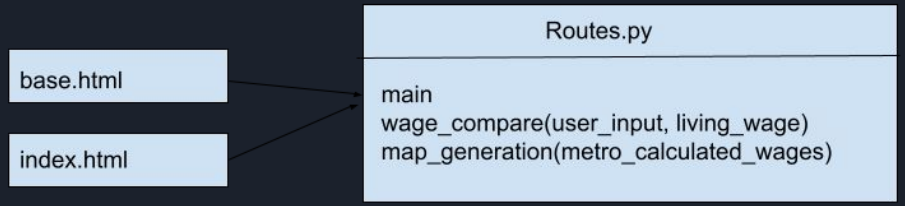
- Drop down menus
  - User friendly
  - Lessens chance of error
- Heat/choropleth map
  - General representation
  - Color coded
- Google authenticator for sign in

Subsystem Communication

- Python subsystem -> Database subsystem
  - Pull data from SQL database into Python file
- UI subsystem -> Python subsystem
  - Send user data to Python file
- Python subsystem -> UI subsystem
  - Send compared data back
  - Populate map

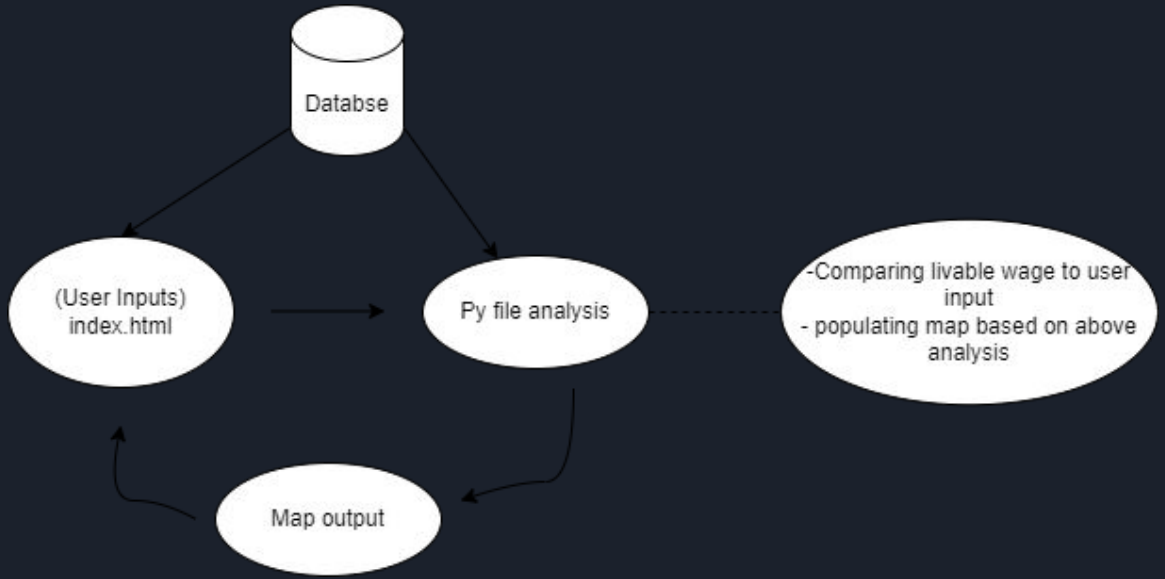
Diagrams


State Finder Web Application
Log in
About page
Contact
Send email to admins
<<extend>>
Collect user information
<<include>>
<<include>>
<<include>>
Display state information
Compare user data with livable wages
Generate map
User
Google Authenticator
Database


base.html
index.html
Routes.py
main
wage_compare(user_input, living_wage)
map_generation(metro_calculated_wages)

Network Diagram

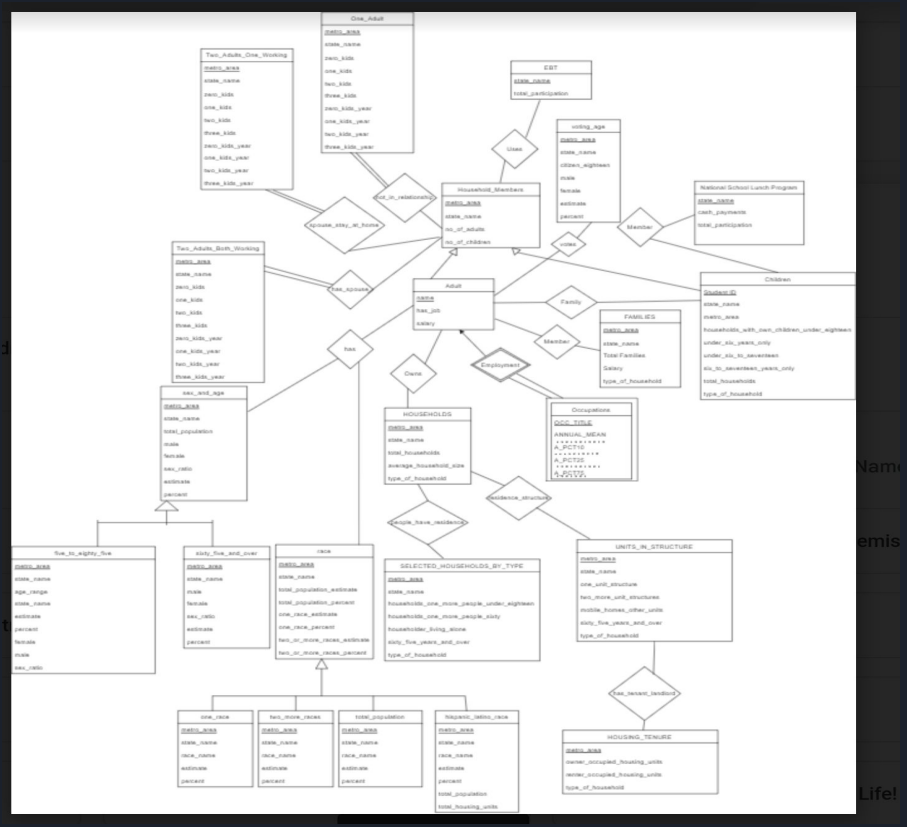
Analysis Perspective Subsystems



Analysis Perspective Continued

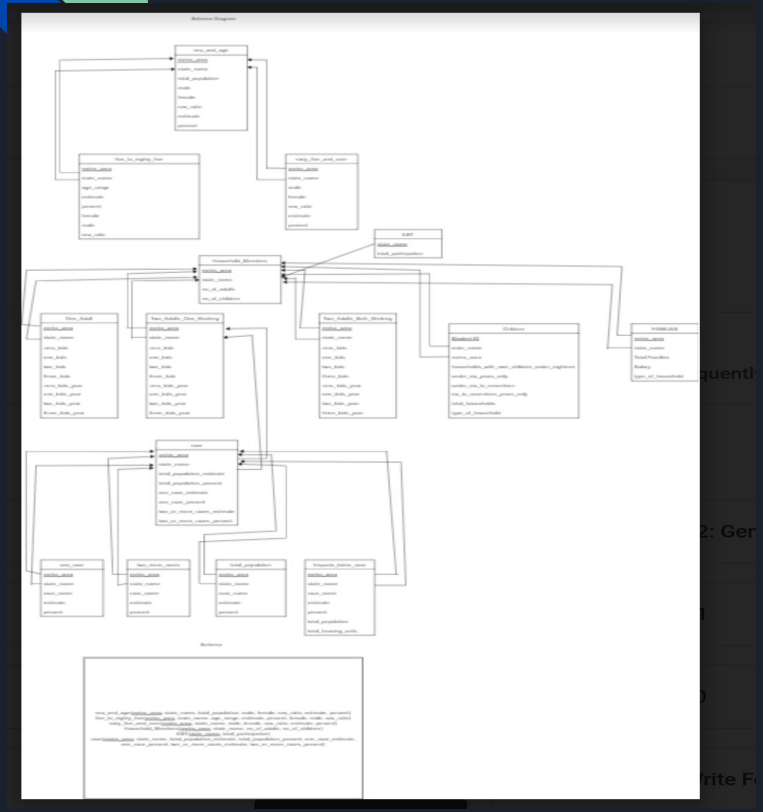
- Each subsystem a Python function
  - Database called only once
  - Data stored as tables in pandas to use from thereon out
  - This adds flexibility as it will allow the site to effectively and efficiently run if the user decides to continuously change their selected inputs.
- Runtime
  - Big O of n.
  - program should only have to run through all methods once to receive the required results.
  - If the user calls n times it will run n times.

ER/Database layout



- Most of the tables have a foreign key constraint on “metro_area” and “state_name”.
- The state_info_population data had to be separated into several tables.
- To reduce confusion some new variables were added such as “race_name” and “type_of_housing”.
- Most of the tables have a relation to the “Adult” table.

Schema and Schema Diagram



- Does not show any of the relations between the tables.
- Most of the tables don’t have any foreign key constraints other than its location.
- Some of the variables have the same name, but they are reflecting different data, so no foreign key constraint there.
- Schema showing a lower level overview of the Schema Diagram.

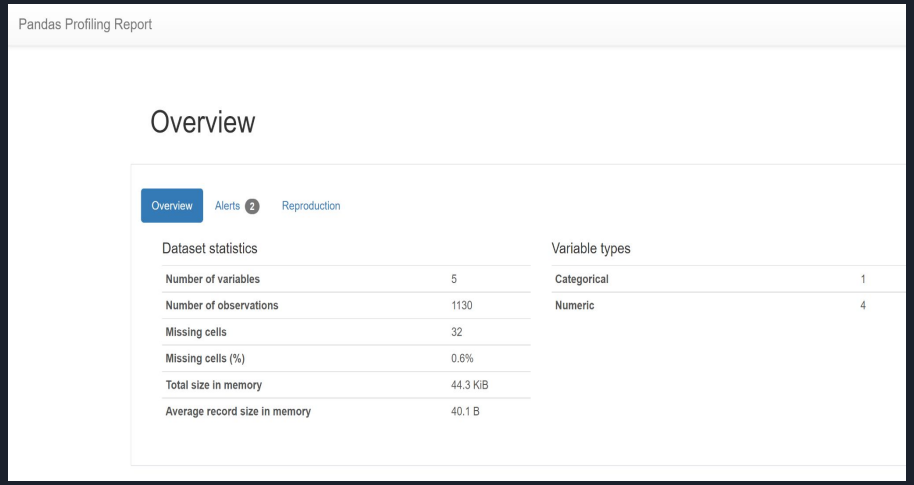
Data Dictionaries

- Our database consists of four tables
  1. Livable_Wages: Contains the livable wages of each metropolitan area in the US (depending on the number of children and working adults per household)
  2. Occupations: Contains the average salaries of various occupations
  3. State_info_population: Contains the demographics of each metropolitan area in the US
  4. State_info_housing: Contains information about the types of housing and housing occupancy of each metropolitan area in the US

Initial Data Findings/Cleaning

- Quick Exploratory Data Analysis using Pandas Profiling.
- Helps give us an idea about where to focus our data cleanup efforts on such as outliers and repeated values.
- Exported the profile report of the data to an HTML file since there was some trouble loading it.

Occupation Data EDA Report



One Adult EDA Report