Roam. Milestone 4
03-15-2019

1. Section - 105-2

2. Team Name - Intravenous Caffeine

3. Team Member Names
    a. Alex Tsalyuk
    b. Andre Dugas
    c. Ari Schermer
    d. Logan Howerter
    e. Richard Diaz Bustos

4. The project plan can be found here: https://github.com/users/andre495/projects/1 on the Roam. project page.

Revised List of Features:

- Feature 1: Value-sliders
    - Primary site function that allows users to denote on simple javascript sliders (0-10) which values/things they value most in a vacation. This is our main source of user-data that will be used to recommend vacations.
- **Functional:** User should be able to adjust certain sliders to fit criteria on the vacation that they would like to experience.
- **Non-Functional:** This feature should be organized and easily understood. The user's inputs should accurately affect the suggestions.

- Feature 2: Select by country/region
    - Allows users to narrow down the total possible area from which a vacation can be pulled and recommended.
- **Functional:** A user should be able to select a specific destination location if they know ahead of time where they want to be travelling.
- **Non-Functional:** This should be presented as an optional feature. The interface should be simple to understand/navigate (ie. dropdown menu) but also aesthetically pleasing.

- Feature 3: Estimated hotel/flight/restaurant quality/type/prices database
    - After accounting for preferred region/country/style-of-vacation using the value-slider data, this feature spits out possible flights/hotels/attractions/locations.

- **Functional:** After the user selects their values through the sliders and check-boxes, our javascript functions will interface with the database to determine the best vacation to recommend.
- **Non-Functional:** We will possibly scrape data from various websites that contain pricing for hotels, flights, and attractions. Include pictures and summary for each hotel/activity so that the user can get an idea of each option without clicking. If a user clicks an option, a more detailed summary will be pulled up and they will have the option to buy/ make a reservation.

- Feature 4: Login and registration page
    - Simple account page that allows users to maintain an account that tracks their prior searches and saved vacation-plans
- **Functional:** User should be able to login with their previously created account name and password.
- **Non-Functional:** This should accurately match the password with the inputted user name in the database. Along with this when inputting the password they should be shown as dots and not as characters to keep the user's privacy.

- Feature 5: Account history tracking
    - Function that maintains users' histories to better serve them with recommendations
- **Functional:** A user should be able to view their past searches and saved vacation objects.
- **Non-Functional:** This should be presented in an aesthetically pleasing and easy to understand way, and should be interfaceable such that the user can change their history or saved searches as they change their opinions.

- Feature 6: Vacation cost summarizer
    - Adds together all of the costs from the recommended travel/hotel/attraction list.
- **Functional:** Take all of the accepted vacation objects (flights, hotels, restaurants, etc.) and sum together their estimated prices to present the user with an approximate budget.
- **Non-Functional:** Present data in a way that explains where the sum comes from, and allows the user to immediately strike items from their list if they are too expensive, or scale up or down the entire vacation based on final cost.

Architecture Diagram:

Javascript/HTML/Bootstrap (hosted by GoDaddy) <- python integration layer<- interfaces with postgreSQL database <- run by NodeJS server

Front End Design: Our front-end design will have three main components. First, we have the log in / register page. Users can can sign into their account on this page. Next, we have the

home page. Here, users will see their past search history and well as options to start a new search. Our search page will be where the preference sliders are located. A user will select their preferences for a travel experience here. Finally, we have the recommendation page. This page will appear after a search, and will show the user where we recommend they should travel.

For a visualization of our front-end architecture, please visit the following link (figma prototype) and press the play button on the top right of the web page.

https://www.figma.com/file/NTao9H3FPJqtbomyUScWAVsM/Roam?node-id=12%3A3

Web Service Design:

We'll be using an HTTP Web Service through Asynchronous JavaScript And XML, User data and inputs will be passed to the API, and Vacation values and recommendation keys will be received from the API.

Back End Design:

PostgreSQL database with many columns where each row is keyed by a country. This database contains cost averages and general facts about countries (GDP/capita, flora/fauna type, etc.) This data will be pulled and manipulated by a python integration layer. The columns that we are using in the database to define locations are:
In_country - which is defining if the location is based inside the country or not
Popage - which is defined on a scale of 1-5 the average age of residents at these travel destinations
Expense - which is defined on a scale of 1-5 the expense that it would cost to travel to these locations
Faraway - which is defined on a scale of 1-5 how far the travel location is from denver
Luxury - which is defined on a scale of 1-5 how luxurious the locations are for travelers
Scenic - which is defined on a scale of 1-5 how scenic the location is.
Touristy - which is defined on a scale of 1-5 based on the average visitors in tourism the location has.
Lodging - which is defined on a scale of 1-5 based on the quality of lodging that the location offers to travelers.

This deliverable provides a summary design of your application's database. The design document should identify each type of data being stored in your database. This may be documented in terms of a schema definition, showing data entities ("files") and attributes ("fields"). This may be documented via an Entity Relationship Diagram showing database tables and columns. The document should identify the specific DBMS technology being used to store your application data (PostgreSQL, MySQL, Firebase, etc.)