Project Report : All About Austin
Phase 2 - Team Amber

Team Members

Name, E-mail, Github User
Justin DuPont, justindpnt@gmail.com, justindpnt
Canyon Evenson, canyon@utexas.edu, cpe342
John Koelling, john.k.koelling@gmail.com, lucundus
Yixing Wang, yixing.wang@utexas.edu, AlienEdith
Grayson Watkins, graysonwatkins@gmail.com, Graysless
Zach Wempe, zdwempe@gmail.com, zachwempe

Project Leader, Phase 2

John Koelling

URL to Website, Git Repository, Google Docs

Deployed Website:
http://www.allaboutaustin.info/
GitHub:
https://github.com/Iucundus/AustinData
Google Docs:
https://docs.google.com/document/d/1R-suefPzFPDNJkwbQ2wc0wpQ5aZYFyTxBlKfVDbHA-4

Tasks Completed:

Tasks for this phase from Project Proposal:

- At least 5 additional user stories on issue boards (10 Total)
- Begin daily data collection in MongoDB system
- Refinement of data collection APIs
- Dynamic website with user-weighted ranking pages hosted on GCP
- Report refinement
- Create the static web pages for each zip code

Completed Tasks:

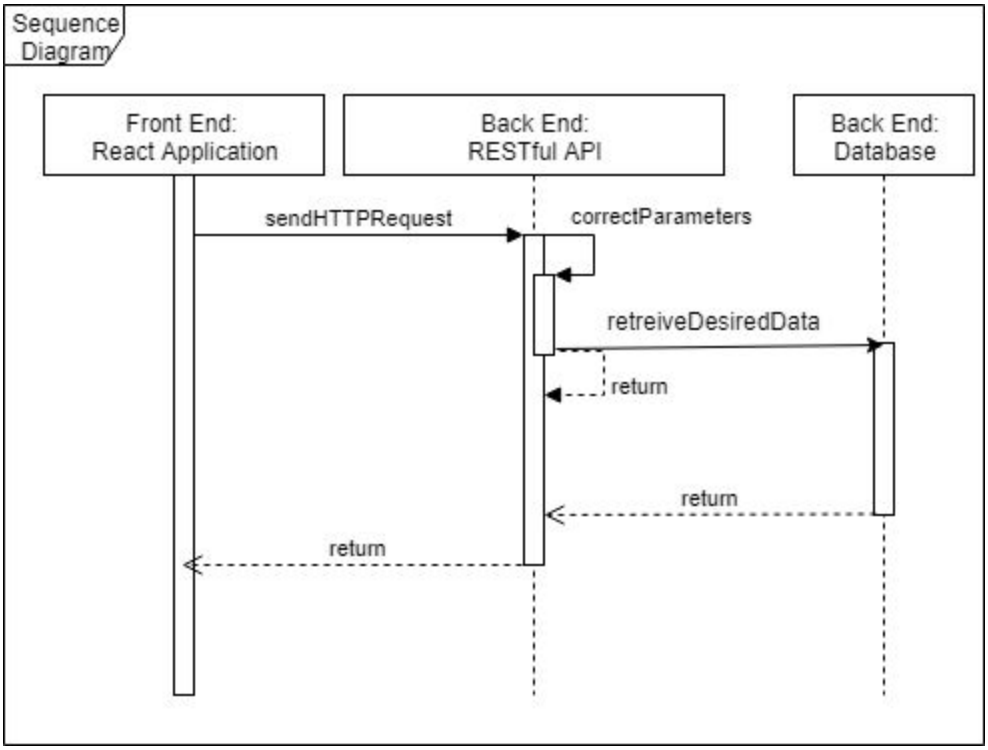
- RESTful API implemented for backend data operations using Spring Boot framework, hosted on https://allaboutaustinapi.appspot.com/api/.
- MongoDB cluster set up for data storage
- Data collection modules written to retrieve data from external sources, process it, and store to MongoDB

- Category-specific webpages implemented
    - Food: gives top 10 Austin zip codes list based on the quality of available food options as determined by Zomato data, user can also sort the list according to other categories.
    - Education: gives top 10 Austin zip codes based on the quality of education in the area as determined by government data, user can also sort the list according to other categories.
    - Traffic: gives top 10 Austin zip codes based on intensity of traffic as determined by City of Austin data, user can also sort the list according to other categories.
- Static zip code-specific webpages implemented: gives detailed information about a specific zip code on its own webpage.
- 5 new user stories on issue board
- Survey and Recommendation Page
    - Weights determined by sliders are now used to rank the various Austin zip codes in terms of desirability according to the user's input.
    - The recommendation page then shows the top zip codes based on user's input of their priorities.
- Backend regular data scraping: Created Google Cloud application that could be deployed to regularly pull data from Austin and Zomato APIs for storage in the MongoDB backend database.
- Capability for user to search zip codes and be redirected via search function in "Zip Codes", "Food", "Education", and "Traffic" pages
    - Click on magnifying glass to initiate onClick trigger
    - Ex. : Type in 78703 and click magnifying glass

Design:

- **Overall Design**

  The core part of our website is our self-implemented backend RESTful API, which connects our front end view part and back end database together. When the front end page is loaded or takes user inputs, it sends an HTTP request to our RESTful API with corresponding requirements/parameters. When the RESTful API receives the request, it will ensure the request is valid, fetch the data from our MongoDB database, then send it to the front end part for display.



- **RESTful API Design:**
  Thus far, our API supports the basic interfaces necessary for our website, and will be expanded later for further usage.
  - https://allaboutaustinapi.appspot.com/api/zipcodes: return all zip codes information based on the descending order of self-calculated average score by default.
    Optional Parameters:
    - sortBy=food/traffic/education: sort the list based the individual category score
    - order=asc/desc: sort the list according to ascending or descending order.
  - https://allaboutaustinapi.appspot.com/api/zipcodes/{zipcode}: return the information of one specific zip code, specified by {zipcode}.
    Parameters: None
  - https://allaboutaustinapi.appspot.com/api/zipcodes/top10: return the information of top 10 zip code based on category (required parameter).
    Required Parameters:
    - category=food/traffic/education

    Optional Parameters:
    - sortBy=food/traffic/education: sort the list based the individual category score
    - order=asc/desc: sort the list according to ascending or descending order.

  - https://allaboutaustinapi.appspot.com/api/zipcodes/ranking?food=&traffic=&education=: return the information of top 10 zip codes which is determined by user input weight for each category.
    Required Parameters:

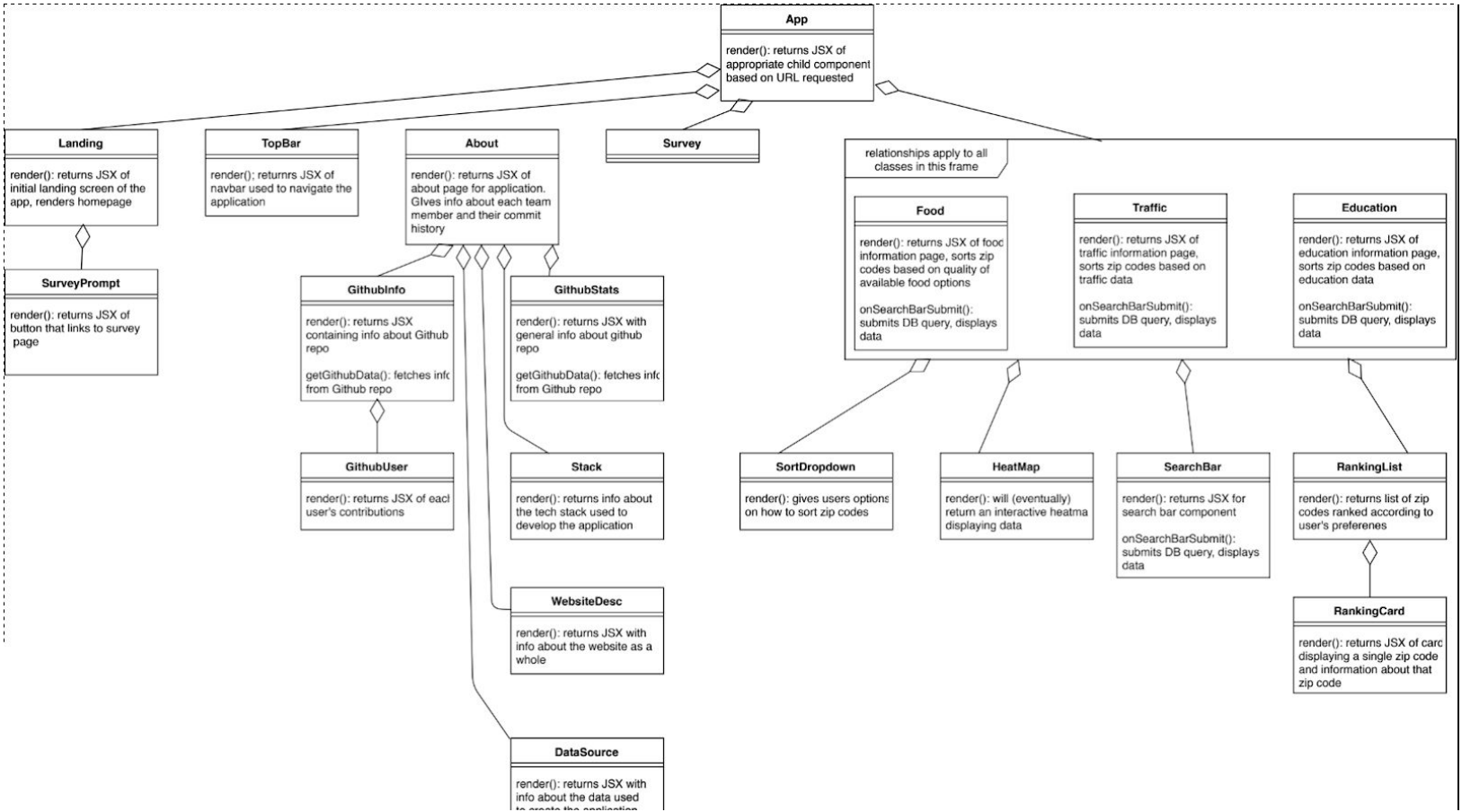
- food={weight} User preference weight (Integer 0-100)
- traffic={weight} User preference weight (Integer 0-100)
- education={weight} User preference weight (Integer 0-100)

- **Database Design:**
  The MongoDB cluster stores data in several forms: Food data, traffic data, education data, and zipcode data. Each of the food, traffic, and education data groups are original data provided by the APIs. We calculate a score for each data point in the range of 0-10, and store it associated with the zip code which that data point is from. Then a second pass on the data is made where we iterate through all zipcodes and fetch the entire set of data for that zipcode, and store it back into MongoDB as zipcode data with all information attached.

- **Data Collection Design:**
  There are three types of data collectors in this part of the program, each for a different data source and its API. The Austin Government collector (SodaCollector) is unique because it collects two databases instead of one: one database for traffic sensor information, and another database for information about those sensors (used to locate them and identify the correct zipcodes.) GoogleZipFinder is a differently structured class that is used to convert latitude / longitude coordinates into zip codes, using the Google Maps Geocoding API.

A DataSet class is used to hold data, and is written generically so that it can hold any kind of data for the zipcodes. FoodRawData, TrafficRawData, and SchoolRawData are the classes used to store this data into MongoDB.

- **Front-end Design:**

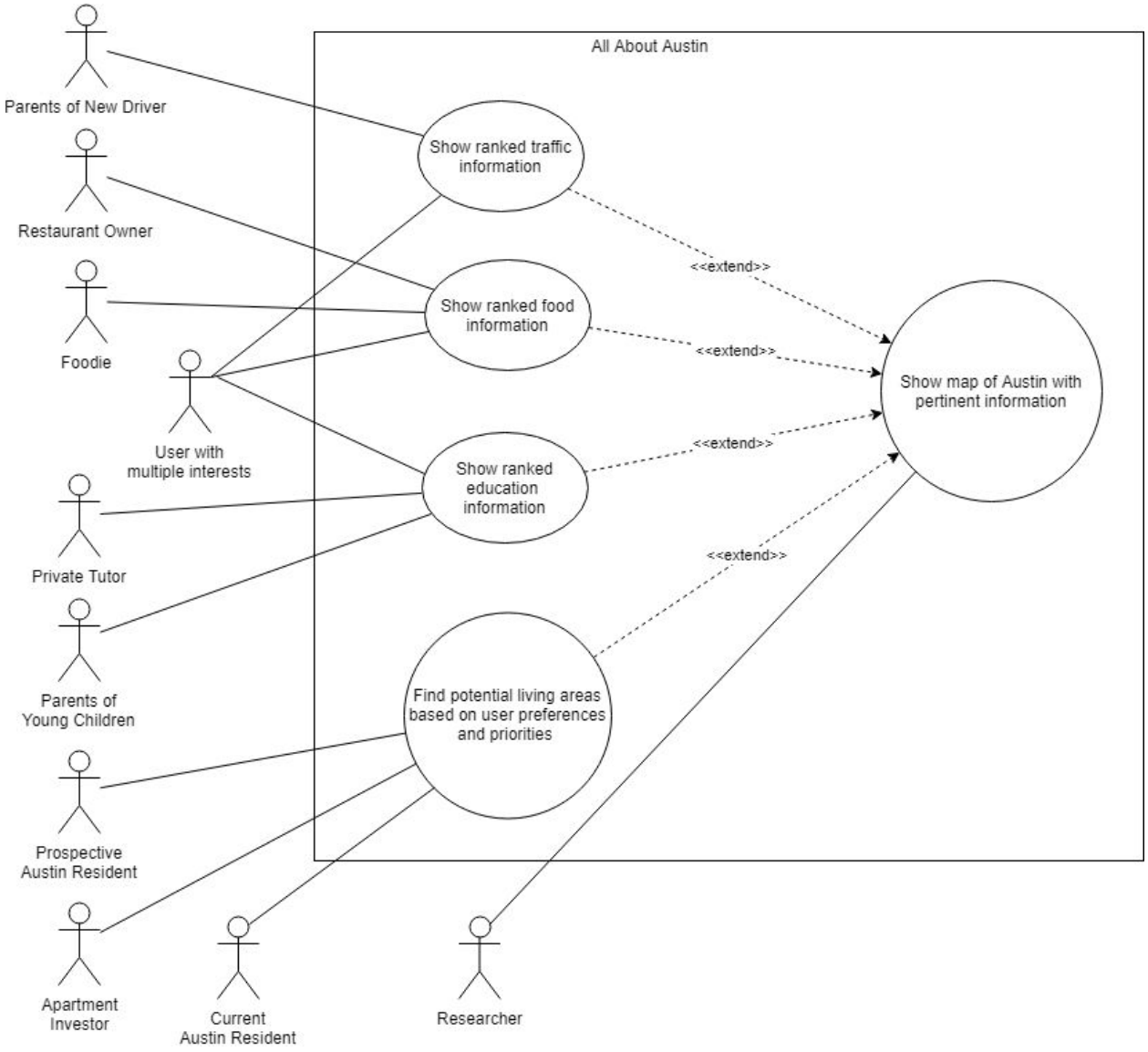


User Requirements:

User Stories for Phase II:

- User Story 6
    - Current Austin resident wants to see the rankings for their current zip code in terms of food, education, and traffic.
    - User Story: As an Austin resident, I want to see the ratings for education, food, and traffic so I can brag about it to my friends.
- User Story 7
    - Disgruntled tenant of an apartment complex in Austin is seeking a place to live with a better food scene and will compare their current zip code to others to make an informed upgrade to their living situation
    - User Story: As a current Austin resident, I want to be able to search my current and surrounding zip codes so I can make sure my new location will be an upgrade in terms of food.
- User Story 8
    - A family who is moving to Austin wants to choose a new zip code to live in, but since they always eat at home the family wants to see the zipcodes in Austin ranked strictly by traffic and education, but does not want the food ranking of restaurants in each zip code to factor into their zip code ranking list.
    - User Story: As a person moving into Austin, I want to be able to choose what factors determine the order that the zip codes in Austin are ranked, so that the list of zip codes is personalized to my preferences.
- User Story 9
    - An Austin resident who wants to move to another zip code wants to consider where they move based on food, traffic, and education but has a list of preferences that vary in importance. Those preferences are: They are no longer a student so they aren't interested in education, they are very particular about their food and will only eat the best food available, and while traffic is a factor in considering where they want to move, it is not nearly as important to them as close by food locations.
    - User Story: As an AllAboutAustin website user, I want to be able to change the importance of the factors that are taken into account when ranking the zipcodes, so that the ranked list of Austin zip codes is personalized to my preferences and perceived importance.
- User Story 10
    - A professor who studies traffic patterns in cities wants to quickly be able to glance at data of Austin and understand where the traffic is heavy and where the traffic is light, in a format that is easy to comprehend.
    - User Story: As a curious user, I want to be able to see Austin data concerning food/traffic/education in a format that is quick and easy to understand, for example, a heat map, so that I can quickly comprehend the data on a citywide level.

Updated Use Case Diagram:



Tools, software, and frameworks used:

- Front End
    - npm: Node Package Manager that works with Javascript (React).
    - React: A frontend javascript library for building user interfaces.
        - react-router-dom: A package that makes routing available for React.
        - axios: A package for HTTP request in React
        - React-redux: A package for state management for React Application
    - Bootstrap 4: A front-end Web framework that provides HTML and CSS-based design templates to beautify the user interfaces.

- Back End
    - Spring Boot framework: a Java backend framework. Used for developing RESTful API connecting to our MongoDB database.
    - Postman: A tool for testing our RESTful API.
    - Lombok: A java library that to facilitate writing standard functions in our Java classes.
- Database
    - MongoDB / MongoDB Atlas: A NoSQL database, Atlas allows us to easily set up and run a cloud database.
    - Morphia: A JVM Object Document Mapper for MongoDB, which allows us to easily interact with the MongoDB cluster in Java.
    - SODA, Socrata Open Data API: Data gathering API for Austin Government data.
    - JSON.simple: Java library to streamline processing of JSON objects from online sources.
    - Maven: Project build and dependency management.

Testing:

Data Collection Application: We created JUnit tests for individual parts of the data collection application, testing individual modules at a time. Zip codes used for testing were: with data, without data, and invalid names. All of the data collectors were tested to ensure they provided non-empty data in a valid format. Test classes are in the Git repository at https://github.com/Iucundus/AustinData/tree/master/DataCollection/src/test/java/database/datacollection

RESTful API: We used Postman for our API testing. We varied the parameters and zip code searches to ensure that the API gave correctly formatted data under all supported forms of request.