Sprint 3: Access Database

Project Group 8: Yao-Fu Yang, Qiushi Li, and Rong Chang
Trello board: https://trello.com/b/61ULBOVX/grocery-store-data-support
Git directory: cs5500_sum2020_group8/GrocerySprint3

User stories

As a store manager, I want to make the database available over the web that allows us to make the queries, specifically:

- I want to access the data through a curl or postman queries.
- I need to be able to update the data through a curl or postman queries.
- I want to have URL format instructions for accessing and updating the database.

Functionality of the Product

At this phase, we have mainly focused our work on the implementation of RESTful requests. The database currently could be accessed by GET queries. The data could also be inserted or updated by POST queries.

In addition, we are also moving forward our work to support the data presentation with a certain order requested by the user. We will keep our work in the next week to allow the user to update or request the data through a user-friendly form. Those further expectations will make this grocery visit data more accessible.

URL Format Instruction

1. Accessing the data in the visits collection:
    a. all visits: *localhost:8080/visits/all/unordered*
    b. all visits, ordered by entry time: *localhost:8080/visits/all/ordered/entry*
    c. a single visit by visit ID: *localhost:8080/visits/single/{visitID}*
    d. visits for specified event and entry time periods:
        localhost:8080/visits/partial/prefix/{idPrefix}\{start}\{end}
        prefix: N: normal daily visit entries
                H: holiday related visit entries
                W: weather related visit entries
                M: meal (lunch / dinner) rush time related visit entries
                S: senior discount event related visit entries
        entry time in date/time interval: formatting example as "*2020-07-08T06:00*"
    e. visits with an entry time in the interval:
        localhost:8080/visits/partial/entry/interval/{start}\{end}
    f. visits with a leave time in the interval:

localhost:8080/visits/partial/leave/interval/{start}|{end}

g. visits with a duration in the interval: example as "*20*"
localhost:8080/visits/partial/duration/interval/{min}|{max}

h. visits with a duration greater than equal to the min:
localhost:8080/visits/partial/duration/LTE/{max}

i. Subset of visits with a duration less than equal to the max:
localhost:8080/visits/partial/duration/GTE/{min}

2. Insert visit to data:
localhost:8080/visits/add/single?visitID={param}&entryTime={param}&leaveTime={param}&duration={param}&holiday={param}&dayOfWeek={param}

URL Tests

For this sprint, we were able to implement and demonstrate the above URL instructions and have chosen Postman as the main API to create requests. We did not create any tests to assess the functionality of our requests and opted to check the return values of each request in order to assess correct functionality. We also re-populated the visits collection with different data to confirm that the functionality persisted with a different datasets inside the visits collection. Of note, we focused more on the GET requests instead of the POST request for this sprint, but anticipate adding more POST functions in sprint4. We also started linking these requests to a form on the web server through http, but have not yet completed this functionality.

Code Quality

To assess our code quality, we generated a CodeMR report for the new classes that were created inside the project GrocerySprint3, which implements the URL instructions and GET/POST requests. The Visit class had a size triggering low-medium level indication, but all of the other categories were at low levels and we did not run into any quality issues. We did not include the CodeMR report for the code for data generation (in Project GroceryStoryTraffic) as we made minimal modifications to the code here and the reports are the same as the last sprint.

In this respect, we are satisfied with the quality of our code for the classes inside GrocerySprint3. We plan to revisit the code in GroceryStoryTraffic for more refactoring but we do not anticipate a large change in CodeMR qualities as most of the indications across the board are at low-medium levels or lower. We have some functionality we would like to extend in sprint4 that may decrease the quality initially, but we plan to iterate through again to fix any quality issues that come up.

Of note, we kept the data generation/data storage functions (inside GroceryStoryTraffic) separate from the data-access requests (inside GrocerySprint3) due to how our projects were built. We feel that attempting to combine the classes in both projects together would likely cause numerous compatibility issues, and have opted to prioritize implementing extra functionality instead.