INFO30005 Deliverable 2 – Mockup App Server

Requirements

This deliverable is based on materials in lectures weeks 1 to 5 and workshops weeks 1 to 6.

For this deliverable, your group needs to present your live server (hosted on Heroku) which supports the features listed below, and is linked to your group's Git repository.

This deliverable is server-side - you are required to deliver routes and database access only. The user interface (HTML pages etc) is not required for this deliverable.

Features to deliver

Your group needs to implement routes to support the following **customer** features:

1) View menu of snacks (including pictures and prices)
2) View details of a snack
3) Customer starts a new order by requesting a snack

Your group needs to implement routes to support the following **vendor** features:

1) Setting van status (vendor sends location, marks van as ready-for-orders)
2) Show list of all outstanding orders
3) Mark an order as "fulfilled" (ready to be picked up by customer)

For simplicity, implement both customer and vendor features in the same server. Use different routes to separate these. For example, customer routes could be under [server-name]/customer and vendor routes under [server-name]/vendor.

Database

To support the above features, your group is required to use MongoDB running on Atlas. When marking the above functionalities, we will also check whether the database is updated appropriately.

Note that your database will need to contain sample data that is not directly manipulated by the routes listed above. For example, for customers to order snacks, you (the developer) must have already entered the list of available snacks into the database. (This list is in the business requirements.) There are no routes yet that allow vendors or anyone else to add these items to the database. For such data, your group can enter values directly into the database through a MongoDB administrator interface such as Compass.

Submission

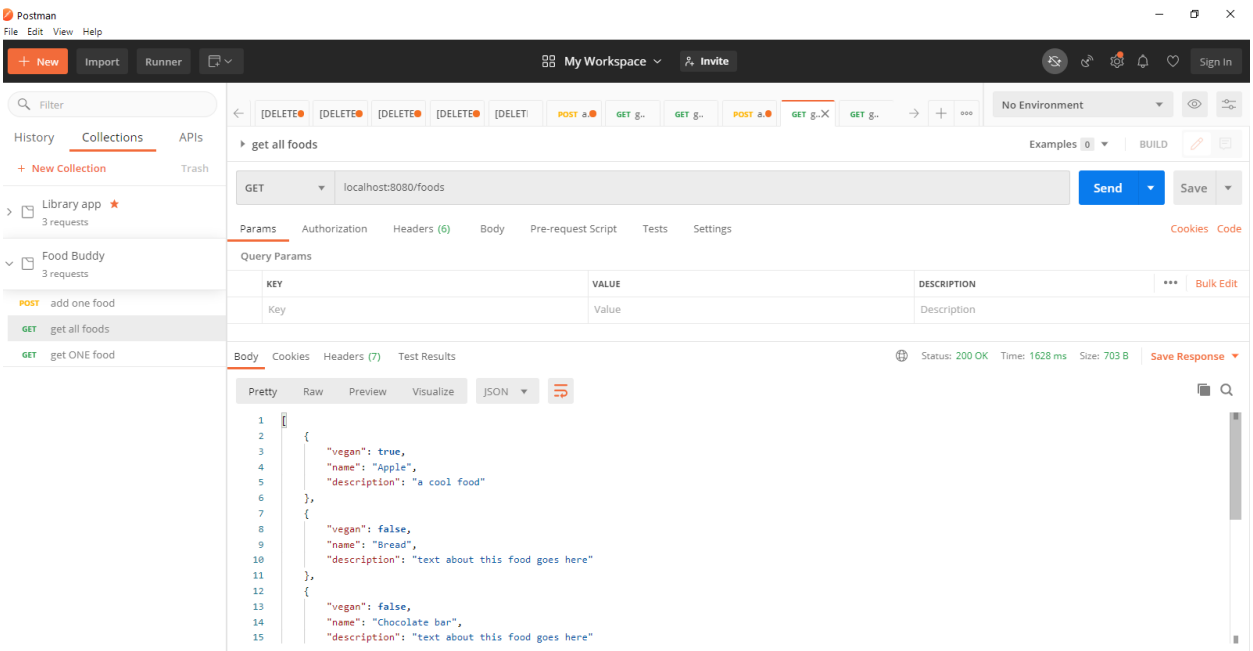
Closer to the submission date, we will open the submission link. One of your group members will submit via the LMS (i.e. we expect one submission per group):

1. The URL for your live website.
2. The commit id that you want us to mark. You may continue working on your project after submitting the deliverable. The commit id will tell us which commit we should use for marking.
3. Your exported requests from Postman, allowing your tutor to access each route.
4. In the README file of the Git repository, describe how your Postman requests work. See comment below regarding how we will test your routes.
5. Provide the access details to your database (in the README file).

Please keep track of the contribution of each group member to the submission. At the end of semester, each student will be offered the opportunity to comment on the contributions made by fellow group members.

How staff will test your routes

Staff will test your routes using Postman (https://www.postman.com/downloads/). For routes that need the user to submit data, include in your Postman requests some sample values that staff can use to test your routes. For example, to display all documents from a collection, Postman could send a request like the following.



Marking Rubric

Criteria	Excellent	Good	Inadequate	Not Submitted (Marks: 0.0)
Online	The website(s) is hosted and can be accessed through a URL. 5			The website is not online. 0
Implementation	The website is well- built, and all concerns are well-separated with good level of modularity. 10	The website is generally well-built, but there are issues with the implementation with level of modularity. 5	The website is implemented in a single file, that is, there is low level of modularity. 3	The website was not implemented using Node.js or No repository link was submitted 0
Functionality	The group implemented RESTful routes for all required functionalities. 15	The group implemented RESTful routes for most of the functionalities; a few were missing. 10	The group implemented RESTful routes for around 50% of the functionalities. 5	The website was not implemented using Node.js, or No repository link was submitted 0
Repository	Professional code quality, consistent naming standards, Good level of commenting, git history with reasonable commit messages, 10	The code is mostly adequate, but includes flaws 6	Inadequate code quality 4	The code is not in the repository 0