Long Test 2

ITMGT 45 A--The Digital Economy
Joseph Benjamin R. Ilagan

Version 1.00

Group Members

Group Name:

ID Number	Name

This Second Long Test is **group work**. You are given around **48 hours** to complete the tasks. The deadline for submission is **11:59PM** on **Saturday, 30 November 2019**.

Create a copy of this Google Doc file and use as template to answer non-coding related items. Share your new doc and send the link to jbilagan@ateneo.edu. Make sure you cc your group mates in your submission notification.

There is no single right answer to this Long Test.

Total Points: 200

Reminders:

- Before submitting your work, please make sure you have tested your application. Any work requiring a major fix in code shall get a maximum grade of 160/200.
- Submit your work in a new Github repository. Do not overwrite any of your existing group or individual repositories.

Background

Hackathon organizers have invited your group to come up with a quick prototype for a cloud kitchen concept allowing John Gokongwei Student Enterprise Center (JSEC) businesses to deliver food to anywhere within campus. You are given 48 hours to complete the prototype.

You hesitate at first because you still feel that what you've learned in ITMGT45 is not enough, but you realize that a lot of the things in the programming part thus far can be reused (and that your class instructor has agreed to coach you during the hackathon). You therefore decide to join the contest.

General Requirements and Assumptions

1. Build a quick prototype of a cloud kitchen online ordering portal showcasing candidate participating JSEC stalls.
2. Each Stall page will showcase menu items that can readily be ordered through the web app. It's best to have photos of the stalls and the food items.
3. You may include food items from more than one participating stall. Assume that a central dispatcher will consolidate all orders by a customer and deliver them in one go.
4. The customer's "food tray" (or shopping cart) must clearly show the stall offering the food item.
5. An order cannot be split across several delivery locations at this time.
6. Before checkout, the customer must specify delivery location. For now, simply allow location descriptions in free text form. Bonus (3 points) if you can incorporate a drop down list of valid delivery locations within campus.
7. Assume that the app portal operators manage the content on behalf of the stalls. In the future, designated contact persons of each stall should manage their own menu content (including advisories involving items that are out of stock).
8. The prototype must be accessible to the public and deployed on the cloud.

Hackathon Coach side-note: preserve all other functionalities of ***digitalcafe*** *(i.e. login, change password, shopping cart, checkout, …). Rename "shopping cart" as "food tray", at least on the visible pages; no need to rename Python functions as these won't be seen by the customer anyway.*

Tasks

1. Choose two to three JSEC stalls, preferably with photos of food items or stalls. If there are no ready-made photos, take a few shots of dishes during your break. No need for

these photos to be of high quality at this point.

2. Reuse your existing products collection in MongoDB and modify the structure and contents.

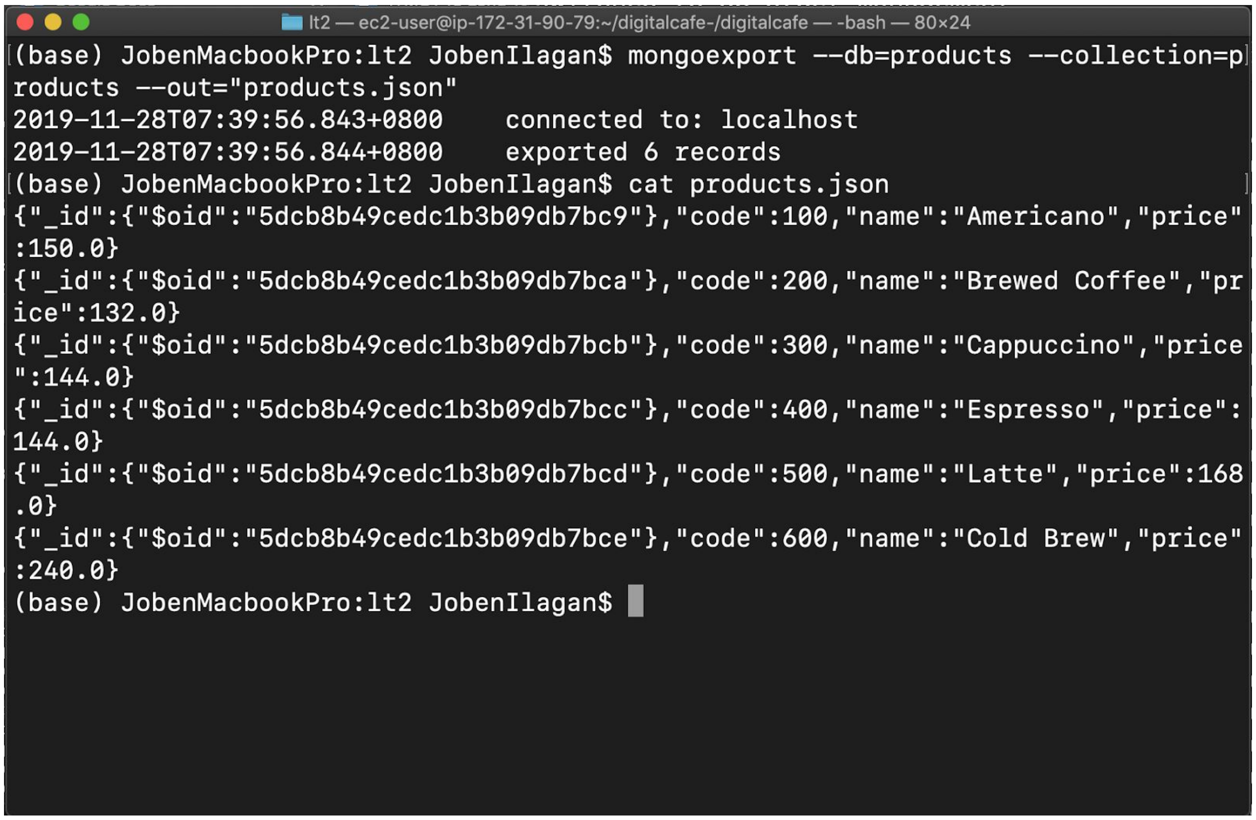
   To make things easier, back up your existing **products** database via mongoexport, keep the resulting .json files, and then drop the database. Here is the command to drop a database:
   https://www.tutorialkart.com/mongodb/mongodb-delete-database/

   Note that if you make a mistake in not backing up your data, the results may be catastrophic.

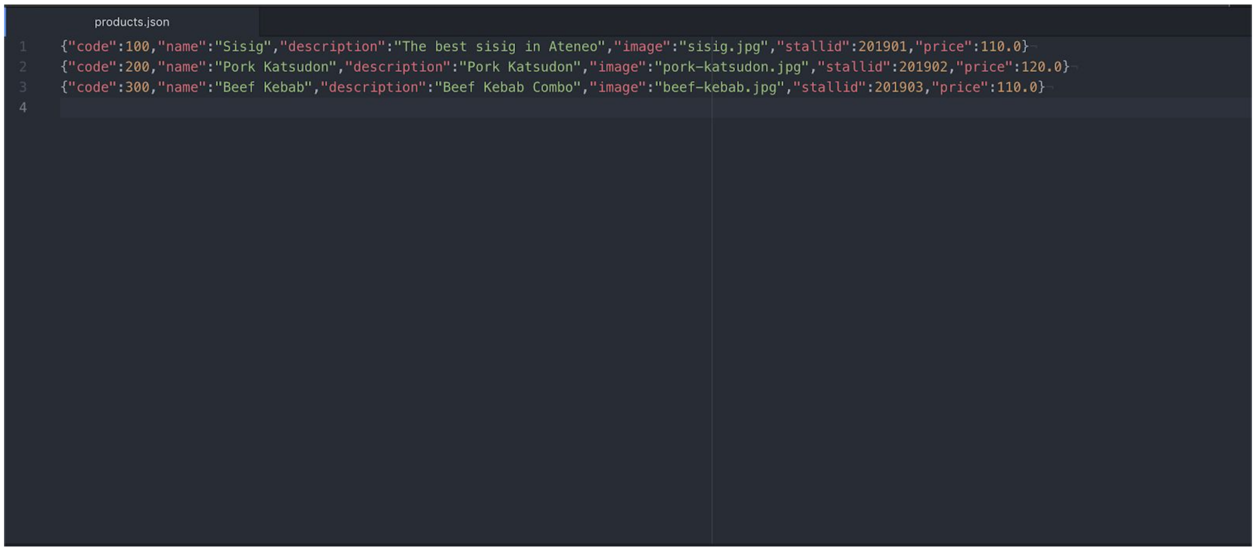
   If you prefer just to drop one collection at a time, you may do so as well.

   https://docs.mongodb.com/manual/reference/method/db.collection.drop/

   Feel free to use the mongoexport output json file as a template for your new **products** collection. Here's an example of how you could go about it:



```
lt2 — ec2-user@ip-172-31-90-79:~/digitalcafe-/digitalcafe — -bash — 80×24
(base) JobenMacbookPro:lt2 JobenIlagan$ mongoexport --db=products --collection=p
roducts --out="products.json"
2019-11-28T07:39:56.843+0800    connected to: localhost
2019-11-28T07:39:56.844+0800    exported 6 records
(base) JobenMacbookPro:lt2 JobenIlagan$ cat products.json
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bc9"},"code":100,"name":"Americano","price"
:150.0}
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bca"},"code":200,"name":"Brewed Coffee","pr
ice":132.0}
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bcb"},"code":300,"name":"Cappuccino","price
":144.0}
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bcc"},"code":400,"name":"Espresso","price":
144.0}
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bcd"},"code":500,"name":"Latte","price":168
.0}
{"_id":{"$oid":"5dcb8b49cedc1b3b09db7bce"},"code":600,"name":"Cold Brew","price"
:240.0}
(base) JobenMacbookPro:lt2 JobenIlagan$
```



```
{"code":100,"name":"Sisig","description":"The best sisig in Ateneo","image":"sisig.jpg","stallid":201901,"price":110.0}
{"code":200,"name":"Pork Katsudon","description":"Pork Katsudon","image":"pork-katsudon.jpg","stallid":201902,"price":120.0}
{"code":300,"name":"Beef Kebab","description":"Beef Kebab Combo","image":"beef-kebab.jpg","stallid":201903,"price":110.0}
```

Notice that we took out the **"_id"** attribute. We won't need this because MongoDB will generate new ones after importing.

3. Create a new MongoDB collection **stalls**. You may also use **branches** as template, but if you prefer to set up from scratch, here is a sample stall document structure:

   `{"stallid":201901,"name":"Kusina"}`

   The choice of **stallid** values is all up to you, but make sure that the stallids you reference in the **products** collection are consistent with the stalls they do reference.

4. We need a page showing a list of Stalls. Each stall entry is clickable. This time, however, each stall will show specific menu items that are orderable.

   *Coach notes: combine Branches and Products functionality from **digitalcafe**.*

5. Create a new flask application folder named **cloudkitchen**.

   *Coach notes: The structure will be similar to your old **digitalcafe** project folder. This time, add two subfolders directly under **cloudkitchen**:*
   - ***images** (this will contain all your web app photos including product and stall shots)*
   - ***styles** (this will contain all your local css files)*

6. Create a group Github repository and commit your application there. Please paste the URL of your Github repository below. Include an **exported_data** folder (which includes your exported json data files) as part of your repository.

Paste URL of your Github Repository here

7. Deploy your web application and MongoDB database to EC2. Use Gunicorn to deploy your flask app. No need to reinstall MongoDB, but make sure you create the necessary databases and collections. If you have an existing gunicorn process, kill it first.
   Paste screenshots of the following:
   - Command to start gunicorn from your EC2 instance (whether from Terminal in Mac OS or PuTTY in WIndows)

     Paste screenshot here

   - Home page of your web app (please include the URL bar in your screenshot)

     Paste screenshot here

8. Paste the URL of your web app deployed to EC2 below.

Paste URL of your web app here

Citations

Please state all cited work and references below. If work used in this test turns out to be copied from various sources without proper citations, you will get an automatic zero.

Paste citation links and descriptions here.

Rubric, Grades and Feedback

Do not write anything below this line.

	Highest Possible Points	**Your Score**
Functionality *(Overall requirements fulfilled, etc.)*	140	
Overall Quality *(no show-stopping errors, computations are consistent, etc.). Note that this portion will*	40	

automatically be zero if there are show-stopping bugs requiring a fix during grading.		
Usability and aesthetics	10	
Packaging of work submission *(instructor notified via email and groupmates cc'd, screenshots provided, etc.). It would help a lot of you include a side presentation deck explaining your work, but this is not required.*	10	
	Total (out of 200)	
	Letter Grade Equivalent	
General Comments		