CMPS 350 Web Development Fundamentals Spring 2021

Final Lab Exam

Student Name	
Student Id	
Email	

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1) The exam duration is **120** minutes. So, read the exam questions carefully and plan your time accordingly.

2) Push your code to GitHub regularly to avoid unpleasant surprises, as your computer might hang or shutdown ☹.

3) The Exam is an open book; however, in case of plagiarism, both parties will receive **- 0 points**. Hence do not share or receive any code from anyone.

4) Once you complete the Exam,

a. Demo your work before leaving the Exam.

b. You should add the screenshots of the output to the testing sheet provided to you

c. You should push your code and testing sheet to your GitHub repo under the Final Exam

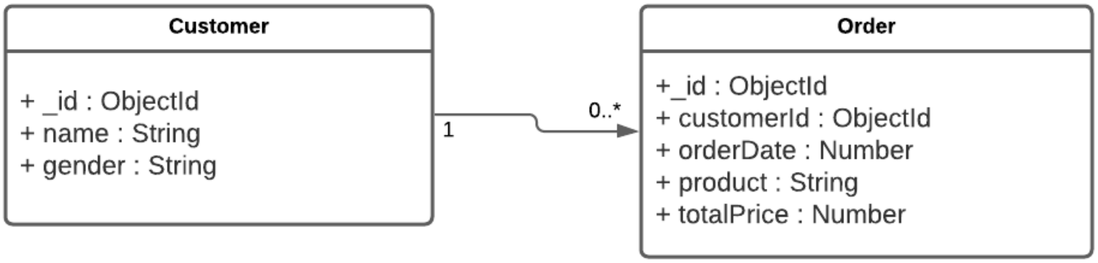
GOOD LUCK ON YOUR EXAMINATION!

Preparation

1. Sync cmps350-lab repo to get the Final Exam files
2. Download the Final Exam's **OrderTrackerApp-BaseCode** from blackboard under Assignments
3. The project contains the public folder [HTML pages, CSS, and client-side JavaScript]. Also, it has the **package.json** file, so do not create a new one.
4. Run **npm install** to get all the dependencies needed to complete the Exam. You should not install any extra packages or remove the **package.json** file.
5. Make sure you run the Mongo DB server [**mongod** / **mongo**] on your terminal. You are also allowed to use atlas but with your own risk as it might not work sometimes.

1. Creating the OrderTrackerApp's Server and APIs [80%]

1.1. **Creating the models** : **[15%]:** You should create two models named [**Customer** and **Order**] with the following properties.



Make sure you write proper validation for all the fields.

➔ All the fields are required.

➔ Gender can be either a **"Male or Female";** otherwise, throw an error

➔The product field can only have the following products (**iPhone**, **iPod**, **MacBook**, **Watch**, **TV**). If the user gives any other product name, you should throw an error.

➔**totalPrice** should be a number greater than 0

1.2. **Creating the repository**: [**30%]** Create a class named **OrderTrackerRepo** under **repository/order-tracker-repo.js** file and implement the following methods

Method	Description
getCustomers()	Return all the customers inside the customers collection
addCustomer(customer)	Adds new customer to the database customer collection
deleteCustomer(customerId)	• Removes the matching customer from the customers collection. • It should also delete all the related orders for that customer
getCustomerOrders(customerId)	Returns all the Orders of the given customer. You should open the orders collection and filter all the orders with the given customer Id
addOrder(order)	Adds new order to the database orders collection
updateOrder(updatedOrder)	Updates specific order
deleteOrder(orderId)	Deletes specific order from the orders collection
getOrder(orderId)	Return a single order that has the same order id

1.3. **Create the services**: **[15%]** Create **order-tracker-service.js** file and implement **OrderTrackerService** class with methods to handle the requests listed in the table-1 below. The handlers should use the provided methods from the **OrderTrackerRepo** class.

1.4. **Create the router** : **[10%]** Create **router.js** file and implement handlers for the routes listed in the table below. The router should use the methods from the **OrderTrackerService** class you implemented above.

Table 1 : APIs Routes

HTPP Verbs	**URL**	**Functionality**
GET	/api/customers	Return all customer
POST	/api/customers	Adds new customer
DELETE	/api/customers/?customerId=	Deletes specific customer from the orders collection that has the same order Id
GET	/api/customers/:customerId/orders	Gets all the orders that have the given customer Id
POST	/api/customers/:customerId/orders	Adds new order to the orders collection
PUT	/api/customers/:customerId/orders	Updates specific order
GET	/api/customers/:customerId/orders/?orderId=	Return a single order that has the same order id and customerId
DELETE	/api/customers/:customerId/orders/?orderId=	Deletes specific order from the orders collection that has the same cusromerId and orderid

1.5. Create the Server: Create **app.js** **[10]** file, and import express package, instantiate an express app, configure it, then start it at port 8080. You should also initialize/connect to your database. Make sure the name of your database is **order-tracker-yourname.db**

Important: Test your API's using Postman before moving to the next step and take all the necessary screenshots of the Postman

2. Connecting the Client App to the Server [20%]

Your task is to **implement the TODOs mentioned in the App. Do not change the function names or parameters.** You only need to implement the todos given to you inside the **public/js/repository/order-tracker-repo.js** file on the client side. And make the **client-side** communicate with the Server.