Project

In this assignment, you will create a web server with endpoints that execute server side logic.

Coding Guidelines

- When creating your solution, you must use the coding practices and conventions demonstrated in class. A solution that does not reflect what was taught in class will not be accepted (0 grade) and/or be subject to an academic integrity review.

Submission Requirements:

- ❏ Submit your code as a zip file to the project dropbox. Include your group number in the name of the zip file.   Depending on how you divide work, you may be submitting 1 zip file, or 3 separate zip files.

- ❏ A screen recording as described in the instructions. Depending on how you divide work, you may have 1 screen recording or 3 separate recordings. If your recordings are too large for the submission dropbox, then you should upload the screen recording to Microsoft OneDrive and share the link in the submission comments. Ensure you update the link sharing settings so that it is accessible by anyone in the college with the link.

Academic Integrity

- You are responsible for familiarizing yourself with the college's Academic Integrity Policy.

- This is an individual assessment

- Situations which often cause academic integrity issues:

    - Reposting any part of the assessment to online forums or homework help websites
    - Contract plagiarism: Purchasing a solution, or completing a solution for financial compensation
    - Sharing or receiving source code, references, or assistance from others

Problem Description:

Create a server-side application for a restaurant to accept, fulfill, and deliver customer orders. Specifically, the application must consist of a:

1. Restaurant website: Customers use this website to view menu items, order items, and view their order history.
2. Order processing website: The restaurant owners use this website to view incoming orders, update the status of an order, and assign an order to a delivery driver.
3. Delivery driver website: Delivery drivers working for the restaurant use this website to manage delivering the order to the customer.

Group Work and Division of Labor

Dividing Work:

You must create both the backend and frontend for these websites. As you work in your group, please ensure that each group member has opportunities to demonstrate their skills in both **server side** and **user interface**.

- Server side skills: database programming, endpoint logic, etc
- User interface: handlebars, html, css, etc

We suggest that each group member take responsibility for one of the websites listed above; and implement both the frontend and backend for that site.

How to Implement The Project

a. 3 projects, 3 server.js files, 1 database

If each group member is responsible for one of the websites, then each group member may create their own separate NodeJS + Express project. For example, you will have a RESTAURANT_WEBSITE project, a ORDER_PROCESSING project, and a DELIVERY_DRIVER project (3 projects, 3 server.js files)

All 3 projects should use the same Mongo connection (one database, 3 projects)

When submitting your code, you can submit 3 separate zip files. Include your group number in each zip file name, for example: RESTAURANT_G04.zip, ORDER_PROCESSING_G04.zip, DELIVERY_DRIVER_G04.zip

b. 1 project, 1 server.js, 1 database

If you do not want ot create 3 separate projects, your group can create 1 project with 1 server.js file. Divide the server.js file into 3 sections. One section for routes related to the RESTAURANT_WEBSITE, another section for routes related to the ORDER_PROCESSING_WEBSITE, and a 3rd section for routes related to the DELIVERY_DRIVER_WEBSITE. Each person works on their own set of routes. When complete, you can merge your routes into the appropriate section of your final server.js file.

For your user interface, create 3 sub folders, example: (views/restaurant, views/orderProcessing, views/deliveryDriver). Each group member can place their website's UI in those folders.

If you are taking this approach, then submit 1 zip file with your group #, example: PROJECT_04.zip

Database Requirements:

Database must be implemented using **Mongo**. We suggest creating 1 **Mongo Atlas** account, and then adding your group members to the account so each person can manipulate the same database. Here is documentation on how to add additional users to a Mongo Atlas project:

- https://www.mongodb.com/docs/atlas/access/manage-project-access/#add-users-or-teams-to-a-project

At minimum, your database must consist of the following collections. You may add additional collections as needed.

orders_collection

- Each order must have a customer name, delivery address, items ordered, date/time of order, and status
- Order statuses include:
    - READY FOR DELIVERY: the restaurant has finished preparing the order and is waiting for the delivery driver to pickup the order. This is the **initial (default) status** of an order.
    - IN TRANSIT: The delivery driver has picked up the order and is delivering it to the customer
    - DELIVERED: The order is delivered to the customer.

drivers_collection

- Each driver has a username, password, full name, vehicle model, color, and license plate.

menu_items_collection:

- A collection of items sold by the restaurant.
- Each menu item has a name, image, description, and price.
- You can pre-populate this collection with data.

User Interface Guidelines

- All user interfaces must be created using EJS (*or)* HTML files + fetch() request.
- UI must be reasonably polished, with pleasing typography, fonts, colors, images, and layouts.
- 3rd party icon libraries are permitted, but CSS styling frameworks like Bootstrap are NOT allowed.

1. Restaurant Website:

Create a website that customers use to view information about the restaurant and order items.

At minimum, the website must show:

- Name of restaurant
- Restaurant location and hours
- Menu with items and pricing.
- Order status form
  → Used by customers to check the status of their order
- Order form
  → Used by customers to place an order with the restaurant
  → You may assume that customers will only ever order ONE of each menu item. (assume quantity = 1)

Menu:

- At minimum, your restaurant must have 4 menu items with different prices
- Menu items must be stored in your database
- Website must programmatically display the menu items based on the database. For example, if the price of a menu item in the database changes, then the website should show the updated price.

Order form:

- Form is used by customers to place an order with the restaurant. You must include all form fields necessary to capture the information that the Order Processing and Delivery websites need to process and deliver the order.
- Assume the customer will only ever order maximum ONE of each menu item (quantity = 1)
- After an order is placed, the application must provide a receipt and an **order confirmation number**. (**HINT**: You can use the order's document._id value as your order confirmation number)

Order status form:

- Form is used by customers to check the status of the order
- Customer check status by entering their order id into a form
- If order id can be located, display its current status
- If no order is found, display an error message.

2. Ordering Processing Website

Create a website that is used by the restaurant to manage orders and update an order status.

<u>List of current orders</u>

- Displays a list of all orders received by the restaurant.
- Each item in the list should show:
    - Customer name
    - Customer delivery address
    - Date of order
    - Total price of the order (order total)
    - Order Status (READY FOR DELIVERY, IN TRANSIT, DELIVERED)
    - If the order is assigned to a driver, show the driver name and license plate.
    - If the order was delivered, provide a way to view the photo of delivery.
- The user can search for all orders from a specific customer.

<u>List of completed orders</u>

- The list should show all orders that are completed.

<u>Order Status</u>

Here are the possible order statuses:

- READY FOR DELIVERY: The order is waiting for a driver
- IN TRANSIT: The driver has picked up the order and is delivering it to the customer
- DELIVERED: The order was delivered.

3. Driver Delivery website

Create a website that allows users to register as a delivery driver for the restaurant.

- Provide appropriate user account and login mechanisms.

After registration / login, drivers can view:

- A list of open orders
- Delivery fulfillment page

List of open orders

- Displays a list of orders available to be delivered (status = READY FOR DELIVERY)
- Driver can select an order to deliver. Once a driver has selected an order, the order status should update to IN TRANSIT
  - The order should be removed from the list of available orders
  - Other logged in drivers will no longer be able to select the order.
  - Changes to the order must be reflected on the Order Processing website.

Delivery fulfillment page

- Provide a page for the driver to update the order status after the order is delivered.
- When the order is delivered, the driver must update the order status to DELIVERED. Changes to the order must also be reflected on the Restaurant and Order Processing websites.
- In addition to updating the status, the driver should upload a photo as proof of order delivery.
  - Here is an example of how to upload an image to Mongo: https://www.geeksforgeeks.org/upload-and-retrieve-image-on-mongodb-using-mongoose/ . You will need to update this example code to use let/const instead of var; and for-of loops instead of the forEach syntax.

END OF ASSESSMENT