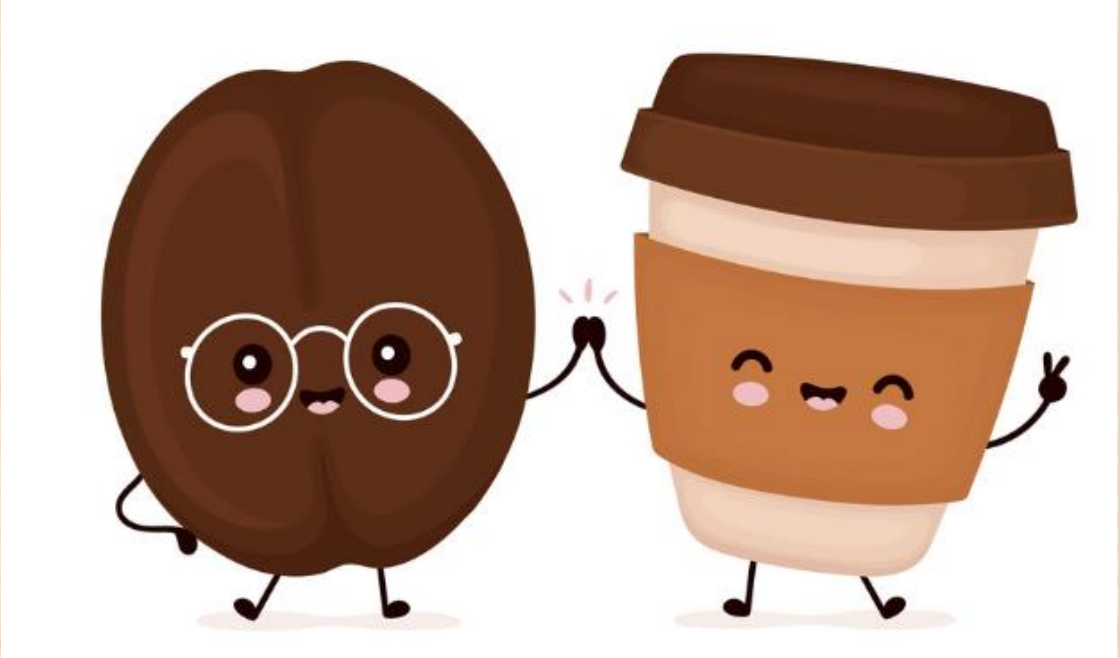
API Project: Coffee Shop API

Using Express and Sequelize

- Razib Hasan — 21I260EB

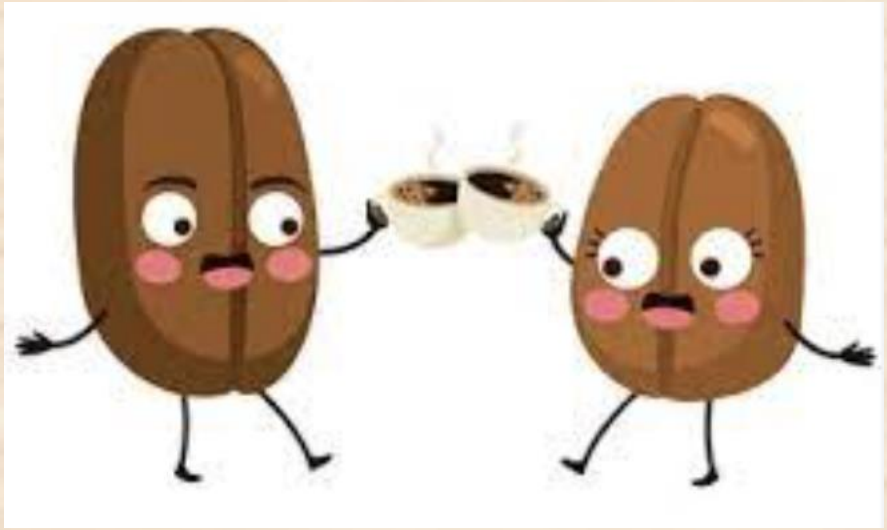
Content:

- Project overview
- Used Technologies
- Solutions
- Demo
- Challenges
- Summary



Project overview

- In this project, I have developed a robust and scalable Coffee Shop API using the Express framework for Node.js and the Sequelize ORM to interact with SQLite relational database. The goal is to provide a seamless and efficient platform for managing coffee-related information, customer details, orders, and order details.



Technologies Used

```
npm install express sqlite3 sequelize
```

SQLite Database:

- A lightweight and efficient relational database for storing coffee details, customer information, orders, and associated order details.

Express Framework:

- Utilized for building a flexible and performant HTTP server.
- Streamlined the creation of API endpoints and handling HTTP requests.

Sequelize :

- Employed for seamless communication with a relational database.
- Enabled the definition of data models, associations, and facilitated database synchronization.

Solutions Implemented

from terminal

(http get)

```
// GET Requests: curl http://localhost:8080/

Coffee endpoint:
# Get all coffees
curl http://localhost:8080/coffees
# Search by name
curl http://localhost:8080/coffees?name=Espresso
# Search by ID
curl http://localhost:8080/coffees?id=1

Customers endpoint:
# Get all customers
curl http://localhost:8080/customers
# Search by name
curl http://localhost:8080/customers?name=Maria

Orders endpoint:
# Get all orders
curl http://localhost:8080/orders
# Search by date
curl http://localhost:8080/orders?date=2023-12-01

Order Details endpoint:
# Get all order details
curl http://localhost:8080/orderdetails
# Search by coffee ID
curl http://localhost:8080/orderdetails?coffeeID=1
```

Solutions Implemented from terminal (http post)

```
POST Requests:
Coffee endpoint:
# Create a new coffee
curl -X POST -H "Content-Type: application/json" -d '{"CoffeeName":"NewCoffee", "Pr

Customers endpoint:
# Create a new customer
curl -X POST -H "Content-Type: application/json" -d '{"FirstName":"John", "LastName

Orders endpoint:
# Create a new order
curl -X POST -H "Content-Type: application/json" -d '{"CustomerID": 1, "OrderDate":

Order Details endpoint:
# Create a new order detail
curl -X POST -H "Content-Type: application/json" -d '{"OrderID": 1, "CoffeeID": 2,
```

Solutions Implemented

from terminal (update)

PUT Request: # Update Coffee with ID 2 curl -X PUT -H "Content-Type: application/json" -d '{"CoffeeName":"UpdatedCoffee"}' http://localhost:8080/coffees/2

Inspection from sqlite3

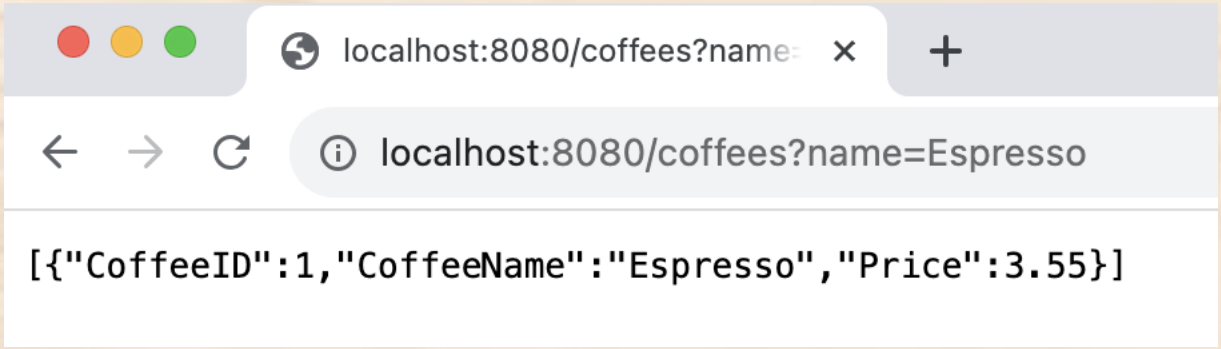
```
sqlite> .table
Coffees       Customers     OrderDetails  Orders
sqlite> SELECT * FROM Coffees;
CoffeeID  CoffeeName      Price
--------  --------------  -----
1         Espresso        3.55
2         American Latte  5.55
3         Cappuccino      6.55
4         Americano       4.55
5         Mocha           7.55
6         Special Coffee  3.55
sqlite>
sqlite> SELECT * FROM Customers;
CustomerID  FirstName  LastName  Email                          Phone
----------  ---------  --------  -----------------------------  ----------------
1           Maria      LastName  maria.lastname@example.com     +358 40 123 4567
2           Sophia     LastName  sophia.lastname@example.com    +358 40 234 5678
3           Elena      LastName  elena.lastname@example.com     +358 40 345 6789
4           Isabella   LastName  isabella.lastname@example.com  +358 40 456 7890
5           Olivia     LastName  olivia.lastname@example.com    +358 40 567 8901
sqlite>
sqlite> SELECT * FROM Orders;
OrderID  CustomerID  OrderDate   TotalAmount
-------  ----------  ----------  -----------
1        1           2023-12-01  8.97
2        2           2023-12-02  14.97
3        3           2023-12-03  23.97
4        4           2023-12-04  11.98
5        5           2023-12-05  20.97
sqlite>
sqlite> SELECT * FROM OrderDetails;
OrderDetailID  OrderID  CoffeeID  Quantity  Subtotal
-------------  -------  --------  --------  --------
1              1        2         2         9.98
2              2        1         1         4.99
3              3        3         3         17.97
4              4        5         1         6.99
5              5        4         2         7.98
sqlite>
```

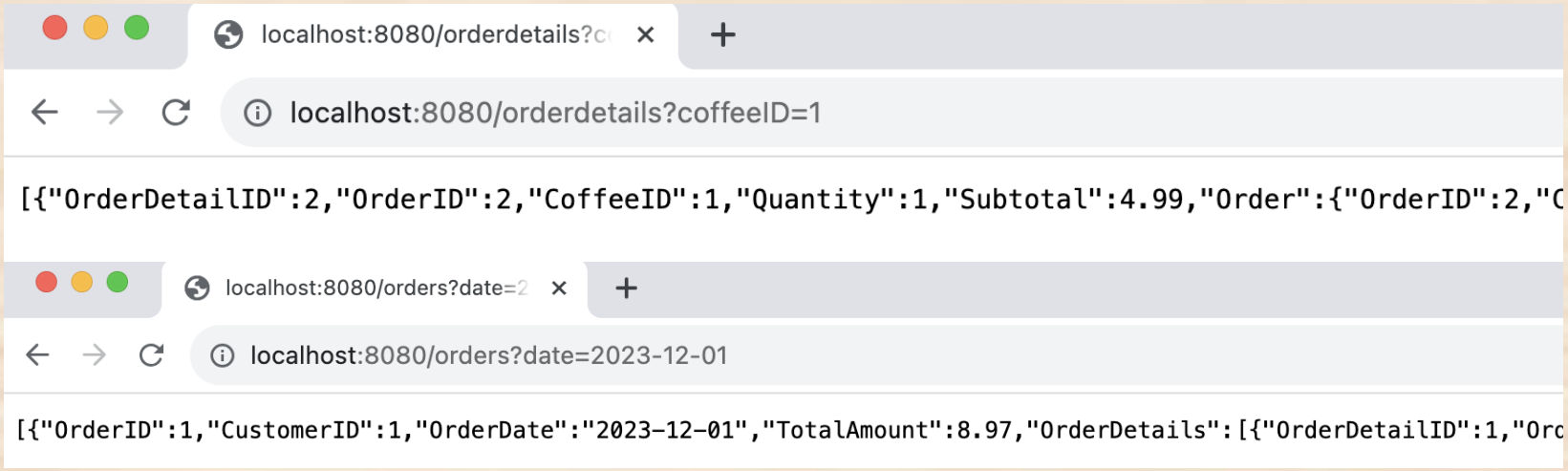
cURL Demo

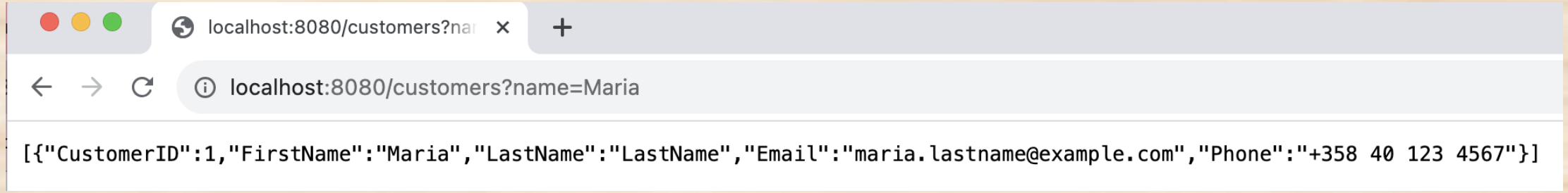
2n — -bash — 106×34

```
358 40 123 4567"},{"CustomerID":2,"FirstName":"Sophia","LastName":"LastName","Email":"sophia.lastname@exam
ple.com","Phone":"+358 40 234 5678"},{"CustomerID":3,"FirstName":"Elena","LastName":"LastName","Email":"el
ena.lastname@example.com","Phone":"+358 40 345 6789"},{"CustomerID":4,"FirstName":"Isabella","LastName":"L
astName","Email":"isabella.lastname@example.com","Phone":"+358 40 456 7890"},{"CustomerID":5,"FirstName":"
Olivia","LastName":"LastName","Email":"olivia.lastname@example.com","Phone":"+358 40 567 8901"}](base) Mac
Book-Pro:2n razibhasan$ curl http://localhost:8080/customers?name=Maria
[{"CustomerID":1,"FirstName":"Maria","LastName":"LastName","Email":"maria.lastname@example.com","Phone":"+
358 40 123 4567"}](base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$ curl http://localhost:8080/orders
[{"OrderID":1,"CustomerID":1,"OrderDate":"2023-12-01","TotalAmount":8.97,"OrderDetails":[{"OrderDetailID":
1,"OrderID":1,"CoffeeID":2,"Quantity":2,"Subtotal":9.98}]},{"OrderID":2,"CustomerID":2,"OrderDate":"2023-1
2-02","TotalAmount":14.97,"OrderDetails":[{"OrderDetailID":2,"OrderID":2,"CoffeeID":1,"Quantity":1,"Subtot
al":4.99}]},{"OrderID":3,"CustomerID":3,"OrderDate":"2023-12-03","TotalAmount":23.97,"OrderDetails":[{"Ord
erDetailID":3,"OrderID":3,"CoffeeID":3,"Quantity":3,"Subtotal":17.97}]},{"OrderID":4,"CustomerID":4,"Order
Date":"2023-12-04","TotalAmount":11.98,"OrderDetails":[{"OrderDetailID":4,"OrderID":4,"CoffeeID":5,"Quanti
ty":1,"Subtotal":6.99}]},{"OrderID":5,"CustomerID":5,"OrderDate":"2023-12-05","TotalAmount":20.97,"OrderDe
tails":[{"OrderDetailID":5,"OrderID":5,"CoffeeID":4,"Quantity":2,"Subtotal":7.98}]}](base) MacBook-Pro:2n
razibhasan$
(base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$ curl http://localhost:8080/orders?date=2023-12-01
[{"OrderID":1,"CustomerID":1,"OrderDate":"2023-12-01","TotalAmount":8.97,"OrderDetails":[{"OrderDetailID":
1,"OrderID":1,"CoffeeID":2,"Quantity":2,"Subtotal":9.98}]}](base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$ curl http://localhost:8080/orderdetails
[{"OrderDetailID":1,"OrderID":1,"CoffeeID":2,"Quantity":2,"Subtotal":9.98,"Order":{"OrderID":1,"CustomerID
":1,"OrderDate":"2023-12-01","TotalAmount":8.97}},{"OrderDetailID":2,"OrderID":2,"CoffeeID":1,"Quantity":1
,"Subtotal":4.99,"Order":{"OrderID":2,"CustomerID":2,"OrderDate":"2023-12-02","TotalAmount":14.97}},{"Orde
rDetailID":3,"OrderID":3,"CoffeeID":3,"Quantity":3,"Subtotal":17.97,"Order":{"OrderID":3,"CustomerID":3,"O
rderDate":"2023-12-03","TotalAmount":23.97}},{"OrderDetailID":4,"OrderID":4,"CoffeeID":5,"Quantity":1,"Sub
total":6.99,"Order":{"OrderID":4,"CustomerID":4,"OrderDate":"2023-12-04","TotalAmount":11.98}},{"OrderDeta
ilID":5,"OrderID":5,"CoffeeID":4,"Quantity":2,"Subtotal":7.98,"Order":{"OrderID":5,"CustomerID":5,"OrderDa
te":"2023-12-05","TotalAmount":20.97}}](base) MacBook-Pro:2n razibhasan$
(base) MacBook-Pro:2n razibhasan$
```

API Endpoints:

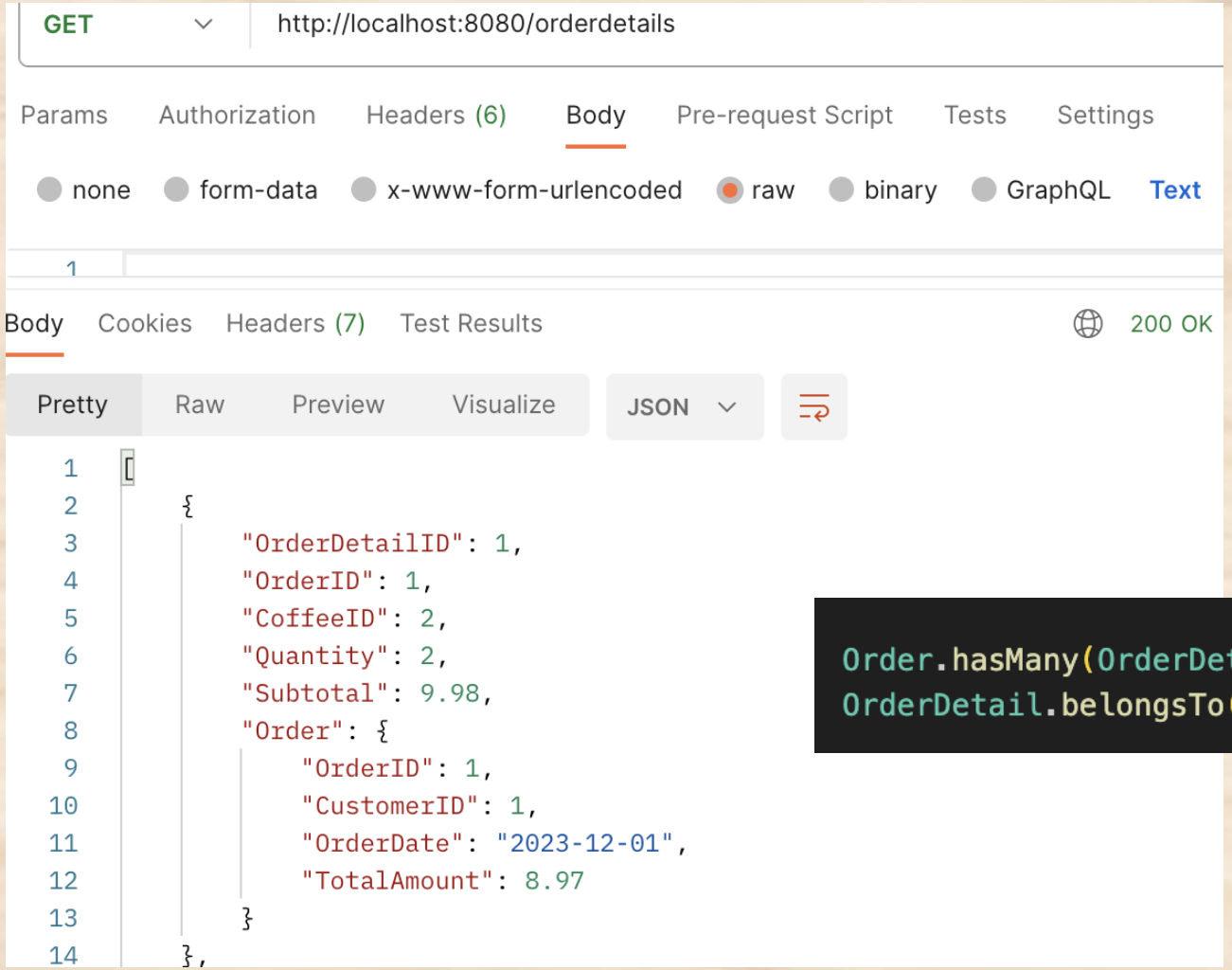






Demo - two tables joined together

- Postman Postman – (HTTP GET), (HTTP POST),(HTTP UPDATE)




```
Order.hasMany(OrderDetail, { foreignKey: 'OrderID', as: 'OrderDetails' });
OrderDetail.belongsTo(Order, { foreignKey: 'OrderID' });
```

Summary

```
GRADE +3

        - HTTP server:
          + uses Express framework.
        - Database:
          + Relational database SQLite.
          + uses Sequelize framework: class Model and methods like findAll().
          + No raw SQL queries.
          + Must use two tables joined together during API GET call.
        - API:
          + Allows searching information (HTTP GET)
            by using multiple criterias in call,
          + Allows adding information (HTTP POST).
          + Allows modifying information (HTTP UPDATE).
        - Response:
          + JSON data.
        - Error handling:
          + Proper HTTP status codes for API requests.
```

Thanks