

FOOD DELIVERY TASK -4



Chat Application and Website

To create a to do planner

LMS Username	Name	Batch
2113a53222	DEEKAN S	A53
2113a53213	Naveen D	A53
2113a53229	Kishore kumar	A53
2113a53224	Aravind gosh	A53
2113a53218	BALAMUGURANR	A53





Task 4:: Backend (Module 4)

Do database modelling and create models

- Design schema for all the data to be stored
- Start mongodb local server and point the backend to the server
- Create mongoose schema at the backend
- Run test queries to set up the database

Create Various APIs to ensure data flow within the website

- Define all the routes for the backend
- Add authentication middleware
- Add controllers for all the paths to handle api request
- Create Environment variables for all authentication keys
- Note: Always handle all possible cases with the request

Evaluation Metric:

100% Completion of the above tasks

Learning outcome

- Understanding Nosql databases modeling
- Querying and filtering mongodb
- Understanding various req methods
- Getting familiar with cookies
- Server side authentication



```
import java.util.ArrayList;import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class FoodDeliveryApp {
  // Store all the restaurants in a HashMap
  private Map<String, Restaurant> restaurantMap;
  // Store all the orders in a HashMap
  private Map<String, Order> orderMap;
  // Store all the customers in a HashMap
  private Map<String, Customer> customerMap;
```

```
// Constructor public FoodDeliveryApp()
restaurantMap = new
HashMap<>();
 orderMap = new
HashMap<>();
customerMap = new
HashMap<>();
 // Add a new restaurant to the app
public void addRestaurant(String restaurantName, Restaurant restaurant)
restaurantMap.put(restaurantName, restaurant);
  // Add a new customer to the app
public void addCustomer(String customerId, Customer customer)
customerMap.put(customerId, customer);
```

```
// Create a new order
public void createOrder(String orderId, String customerId, String restaurantName,
List<FoodItem>foodItems)
Customer customer = customerMap.get(customerId);
Restaurant restaurant = restaurantMap.get(restaurantName);
 Order order = new Order(orderld, customer, restaurant, foodItems);
orderMap.put(orderId, order);
// Get a list of all the orders for a particular restaurant
public List<Order> getRestaurantOrders(String restaurantName)
List<Order> restaurantOrders = new ArrayList<>();
for (Map.Entry<String, Order> entry: orderMap.entrySet())
 Order order = entry.getValue();
 if (order.getRestaurant().getName().equals(restaurantName))
restaurantOrders.add(order);
 return restaurantOrders;
```

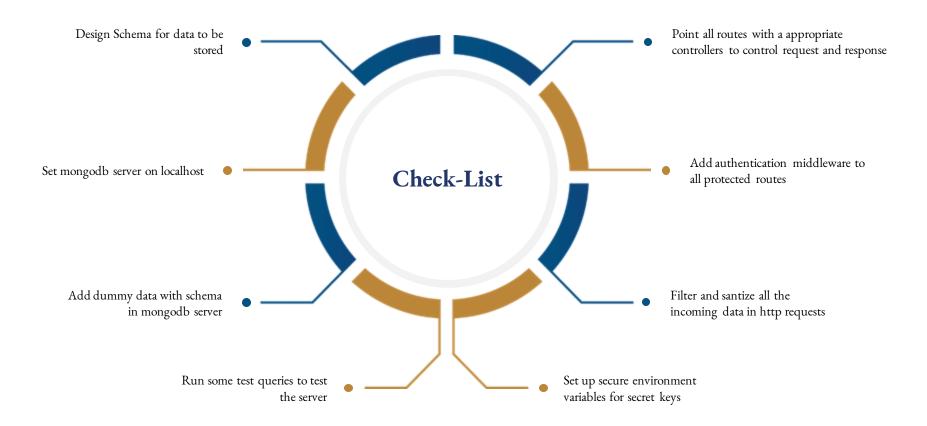
```
// Get a list of all the orders for a particular customer
public List<Order>
getCustomerOrders(String customerId)
  List<Order> customerOrders = new
ArrayList<>();
   for (Map.Entry<String, Order> entry : orderMap.entrySet())
Order order = entry.getValue();
 if (order.getCustomer().getId().equals(customerId))
 customerOrders.add(order);
return customerOrders;
```



```
// Update the status of an order
public void updateOrderStatus(String orderId, OrderStatus status)
 Order order = orderMap.get(orderId);
   order.setStatus(status);
   // Get the status of an order
 public OrderStatus getOrderStatus(String orderId)
Order order = orderMap.get(orderId);
   return order.getStatus();
```



Assessment Parameter







hank

