Dinner Download

A Food Delivery Application

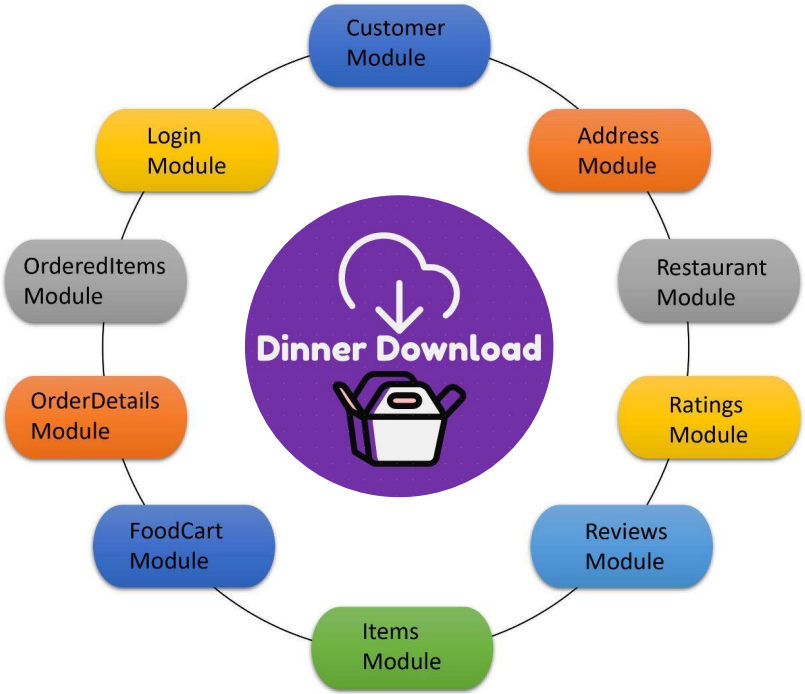
Problem Statement

To create an application for Online Food Delivery that allows restaurant owners to deliver food online as well as users to order food anytime and anywhere.

Features

- Login as Restaurant Owner or Customer
- Add, Update, Remove and Access your Data
- Add, Remove,Update and Retrieve Item Details
- View nearby Restaurants by Id,area and city
- Add or Delete Items to your cart, order food and view Order History
- Provide ratings and reviews to the Restaurants

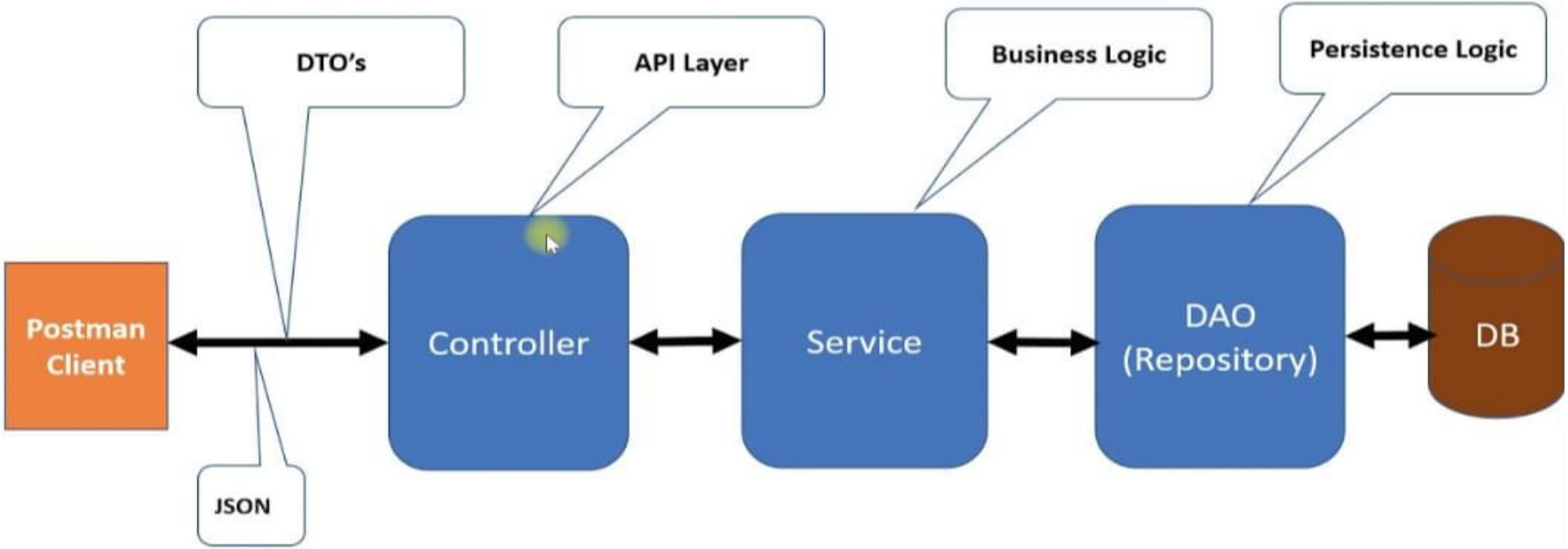
Modules



The ten core modules of the application are developed using the following technologies :

- Spring Boot
- Spring Data JPA
- Spring REST
- Mockito
- JUnit5
- Lombok
- PostgreSQL
- Postman
- SonarQube

Spring Boot Application Architecture



Users and Use Cases

Restaurant :

Restaurant is responsible for handling items and restaurant details. Restaurant owner can add, remove restaurant, get items of a restaurant, get list of restaurants by city and area, add and remove items from the restaurant and also update the restaurant details.

Use Cases :

Add Restaurant	/restaurant	Get Restaurants by Area	/restaurantsByArea/{area}
Update Restaurant Details	/restaurantDetailsUpdate	Add Items to Restaurant	/addItemsToRestaurant
Get all the Restaurant	/restaurants	Get items of Restaurant	/restaurnt-items/{rest_id}
Get Restaurants by Id	/restaurantById/{id}	Delete Restaurant	/delrestaurant/{id}
Get Restaurants by City	/restaurantsByCity/{city}		
DeleteItemsfromRestaurant	/deleteItemFromRestaurant/{rest_id}/{item_name}		

Users and Use Cases

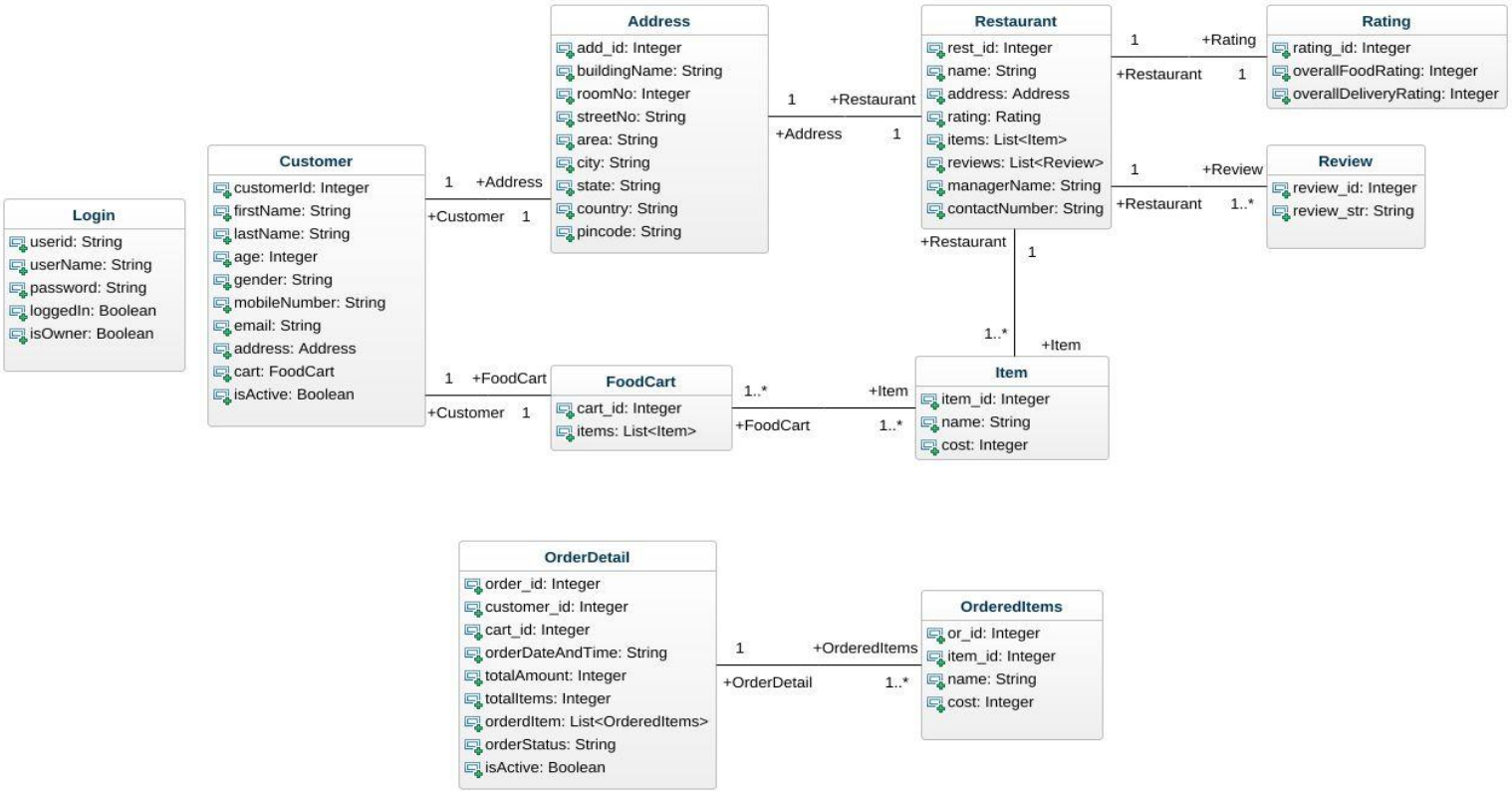
Customer :

Customer is responsible for handling customers and foodcart details. Customer can add, delete, update customer, get all customers, get customer by Id, add item, delete item from foodcart, get foodcart details and get customer by city and area.

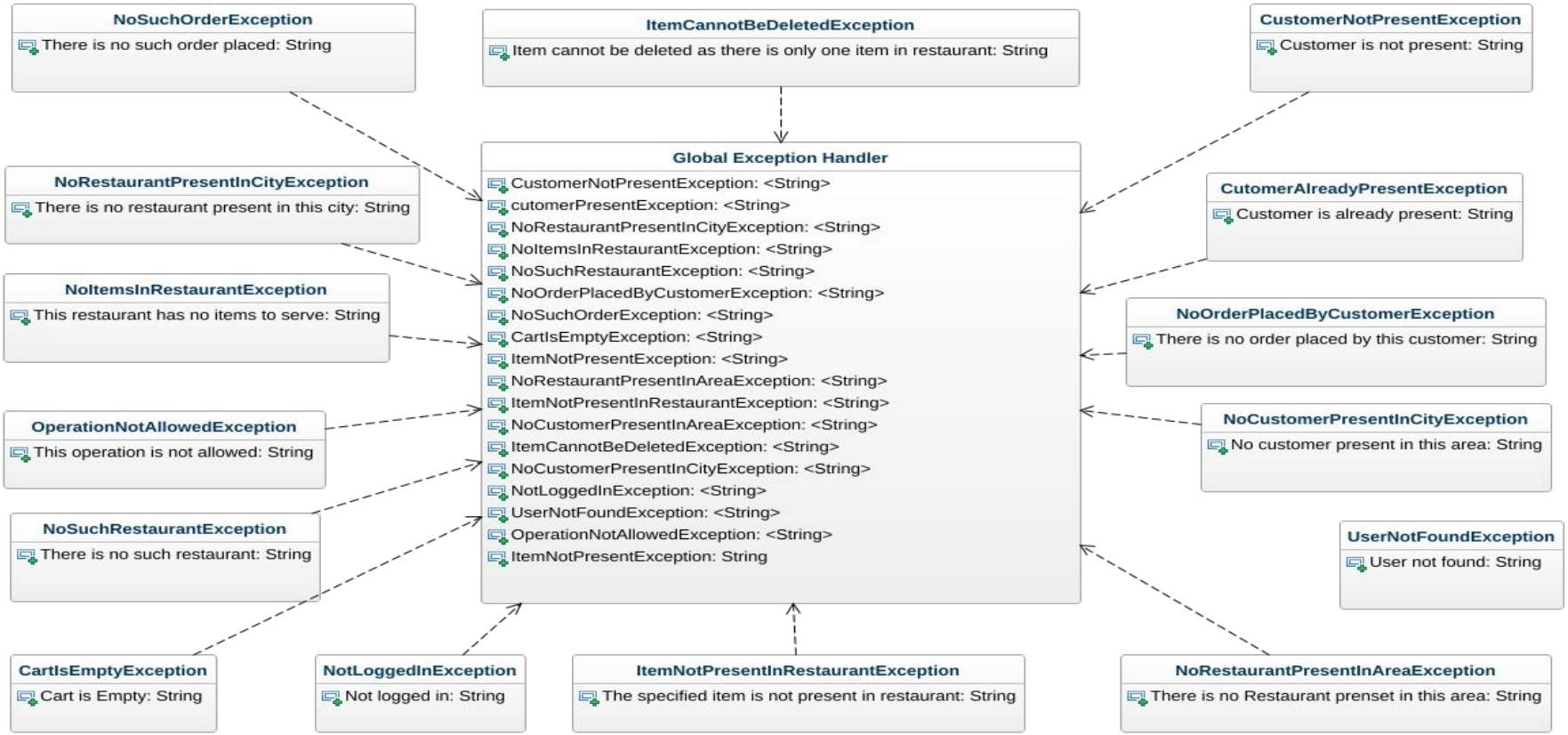
Use Cases :

Add Customer	/addCustomer
Get all Customers	/getCustomers
Get Customer by Id	/getCustomerById/{id}
Get Customer by city	/getCustomerByCity/{city_name}
Get Customer by area	/getCustomerByArea/{area_name}
Add Items to Cart	/addItemsToTheCart
Delete Items from Cart	/deleteItemsFromCart/{cust_id}/{item_id}
Get Cart	/viewCart/{id}
Update Customer	/updateCustomer
Delete Customer	/deleteCustomer/{id}
Add Order	/addOrder
CancelOrder by ID	/cancelOrder/{id}
Customer Orders	/viewOrdersByCustomerId/{id}
Restaurant Orders	/viewOrdersByRestaurant/{rest_id}

Entities


Login
userid: String
userName: String
password: String
loggedIn: Boolean
isOwner: Boolean
Customer
customerId: Integer
firstName: String
lastName: String
age: Integer
gender: String
mobileNumber: String
email: String
address: Address
cart: FoodCart
isActive: Boolean
1 +Address
+Customer 1
Address
add_id: Integer
buildingName: String
roomNo: Integer
streetNo: String
area: String
city: String
state: String
country: String
pincode: String
1 +Restaurant
+Address 1
Restaurant
rest_id: Integer
name: String
address: Address
rating: Rating
items: List<Item>
reviews: List<Review>
managerName: String
contactNumber: String
1 +Rating
+Restaurant 1
Rating
rating_id: Integer
overallFoodRating: Integer
overallDeliveryRating: Integer
1 +Review
+Restaurant 1..*
Review
review_id: Integer
review_str: String
1 +FoodCart
+Customer 1
FoodCart
cart_id: Integer
items: List<Item>
1..* +Item
+FoodCart 1..*
+Restaurant 1
1..* +Item
Item
item_id: Integer
name: String
cost: Integer
OrderDetail
order_id: Integer
customer_id: Integer
cart_id: Integer
orderDateAndTime: String
totalAmount: Integer
totalItems: Integer
orderdItem: List<OrderedItems>
orderStatus: String
isActive: Boolean
1 +OrderedItems
+OrderDetail 1..*
OrderedItems
or_id: Integer
item_id: Integer
name: String
cost: Integer

Exception class



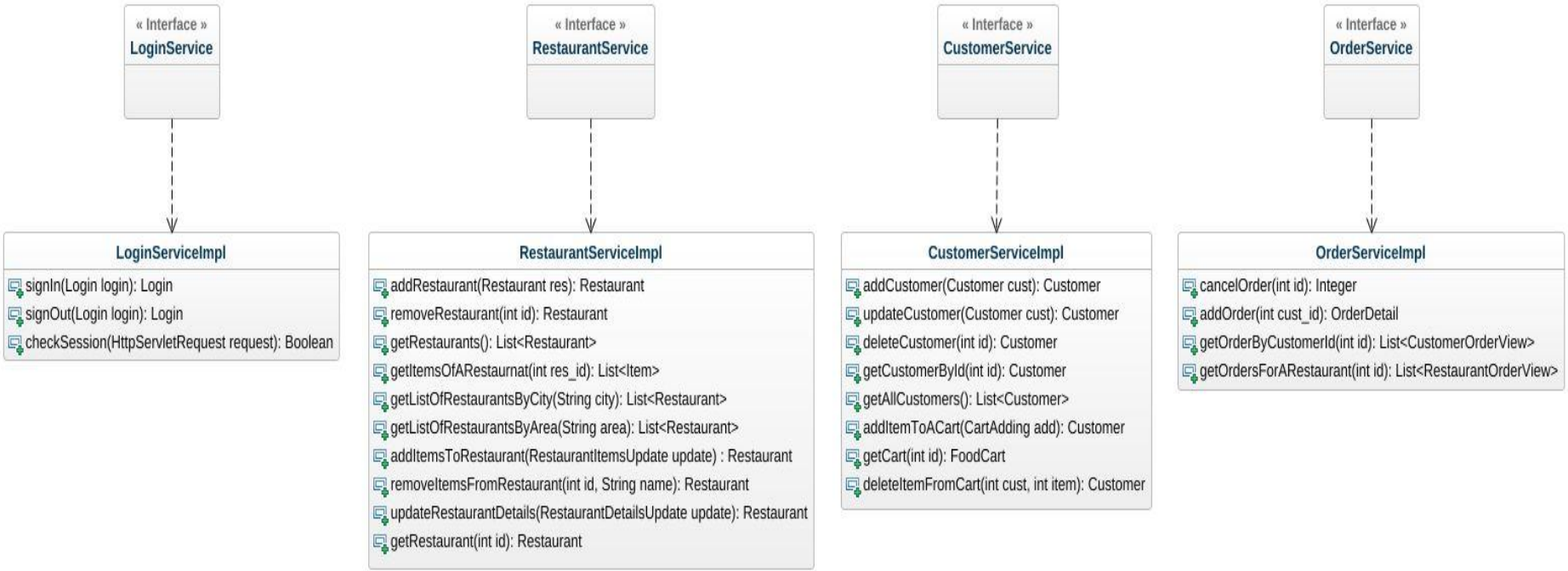
Models

AddingOrder
cust_id: Integer
CartAdding
id: Integer
list: List<Item>
RestaurantDetailsUpdate
id: Integer
name: String
managerName: String
contact: String
RestaurantItemsUpdate
rest_id: Integer
list: List<Item>

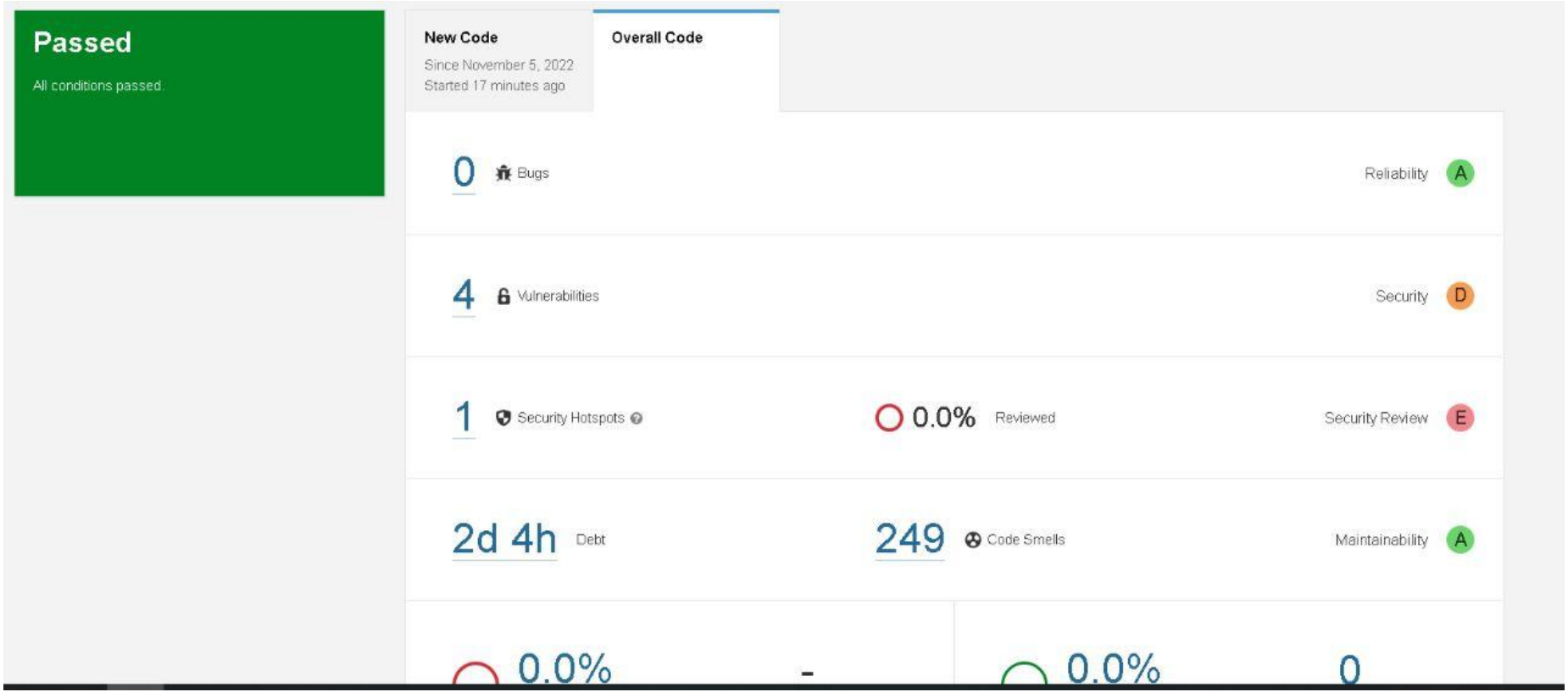
Repository

« Interface »
OrderRepository
« Interface »
LoginRepository
« Interface »
ItemRepository
« Interface »
CustomerRepository
getCustomerByCity(city: String): List<Customer>
getCustomerByArea(area: String): List<Customer>
« Interface »
RestaurantRepository
getItemsOfARestaurant(id: Integer): List<Item>
getRestaurantsByCity(city: String): List<Restaurant>
getRestaurantsByArea(area: String): List<Restaurant>

Services


« Interface »
LoginService
« Interface »
RestaurantService
« Interface »
CustomerService
« Interface »
OrderService
LoginServiceImpl
signIn(Login login): Login
signOut(Login login): Login
checkSession(HttpServletRequest request): Boolean
RestaurantServiceImpl
addRestaurant(Restaurant res): Restaurant
removeRestaurant(int id): Restaurant
getRestaurants(): List<Restaurant>
getItemsOfARestaurnat(int res_id): List<Item>
getListOfRestaurantsByCity(String city): List<Restaurant>
getListOfRestaurantsByArea(String area): List<Restaurant>
addItemsToRestaurant(RestaurantItemsUpdate update) : Restaurant
removeItemsFromRestaurant(int id, String name): Restaurant
updateRestaurantDetails(RestaurantDetailsUpdate update): Restaurant
getRestaurant(int id): Restaurant
CustomerServiceImpl
addCustomer(Customer cust): Customer
updateCustomer(Customer cust): Customer
deleteCustomer(int id): Customer
getCustomerById(int id): Customer
getAllCustomers(): List<Customer>
addItemToACart(CartAdding add): Customer
getCart(int id): FoodCart
deleteItemFromCart(int cust, int item): Customer
OrderServiceImpl
cancelOrder(int id): Integer
addOrder(int cust_id): OrderDetail
getOrderByCustomerId(int id): List<CustomerOrderView>
getOrdersForARestaurant(int id): List<RestaurantOrderView>

SonarQube


Passed
All conditions passed.
New Code
Since November 5, 2022
Started 17 minutes ago
Overall Code
0 Bugs
Reliability A
4 Vulnerabilities
Security D
1 Security Hotspots
0.0% Reviewed
Security Review E
2d 4h Debt
249 Code Smells
Maintainability A
0.0%
-
0.0%
0