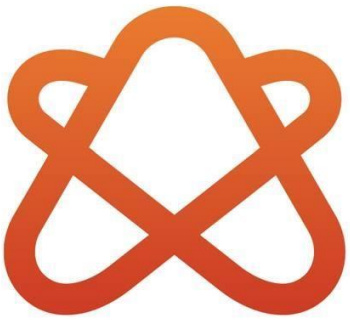
Technical Assessment – User Guide
Gapstars

June 2020

Candidate Name – Binu Lorenzuhewa



Document Control

Date	Version	Modification	Author
2021/06/19	1.0	Initial Documentation	B.K. Lorenuzhewa

Contents

Introduction 4
Setting up the project 5
Software prerequisites 5
Step 1 – Cloning the project 5
Step 2 – Open the project via IDE 5
Step 3 – Executing the database script 6
Step 4 – Changing the application configurations 6
Step 5 – Building the application 7
Step 6 – Running the application 7
Special Note 8
Testing the application 10
Testing via predefined methods 10
Testing via Swagger UI 10
Performing Assessment Task 1 – Create Two Customers 11
Performing Assessment Task 2 – Add product to first customer 11
Performing Assessment Task 3 – Calculate Amounts 11
Performing Assessment Task 4 – Add products to second customer 11
Performing Assessment Task 5 – Calculate Amounts 11
Application Features 12
Additional Features 12
Improvements and Optimizations 12

Introduction

Documentation elaborates operational and technical information of the technical assessment.

Setting up the project

Software prerequisites

Majorly below components will be required to setup and run the project.

1. Java 11
2. Maven
3. MySql Server
4. Operating System – Windows recommended.

Step 1 – Cloning the project

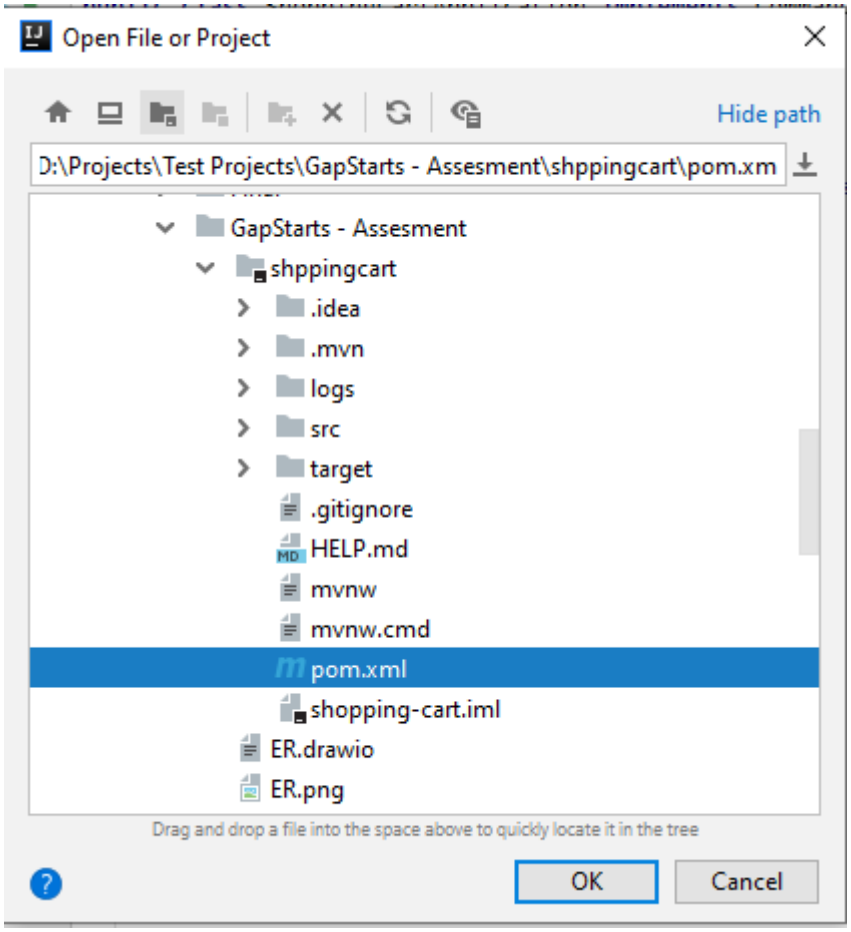
Project is pushed into GitHub Public Repository.

Clone URL: https://github.com/BynuLorenz/shopping-cart.git

Step 2 – Open the project via IDE

Assumptions: Java, Maven and MySql are already configured.

Go to Clone location and go to folder Code. Open folder shopping-cart. Open pom.xml via IDE.



Step 3 – Executing the database script

Go to clone location.

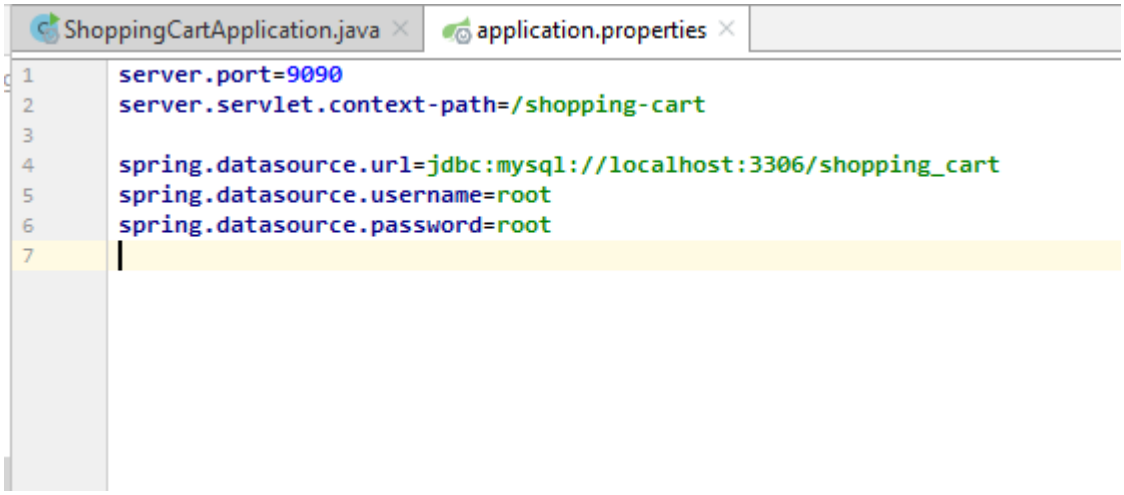
Open folder Docs, execute file “shopping_cart.sql” in mysql server.

Step 4 – Changing the application configurations

Open file “application.properties” in resource folder of the project.

Amend below properties with local machine configuration

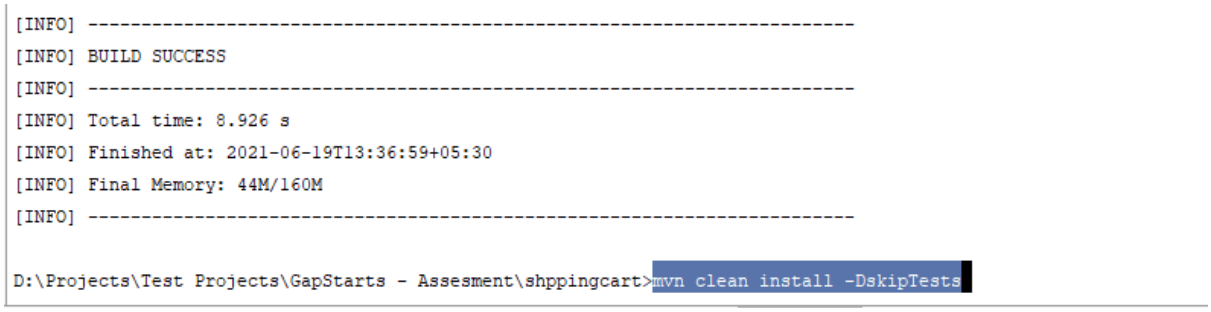
- spring.datasource.username
- spring.datasource.password



Step 5 – Building the application

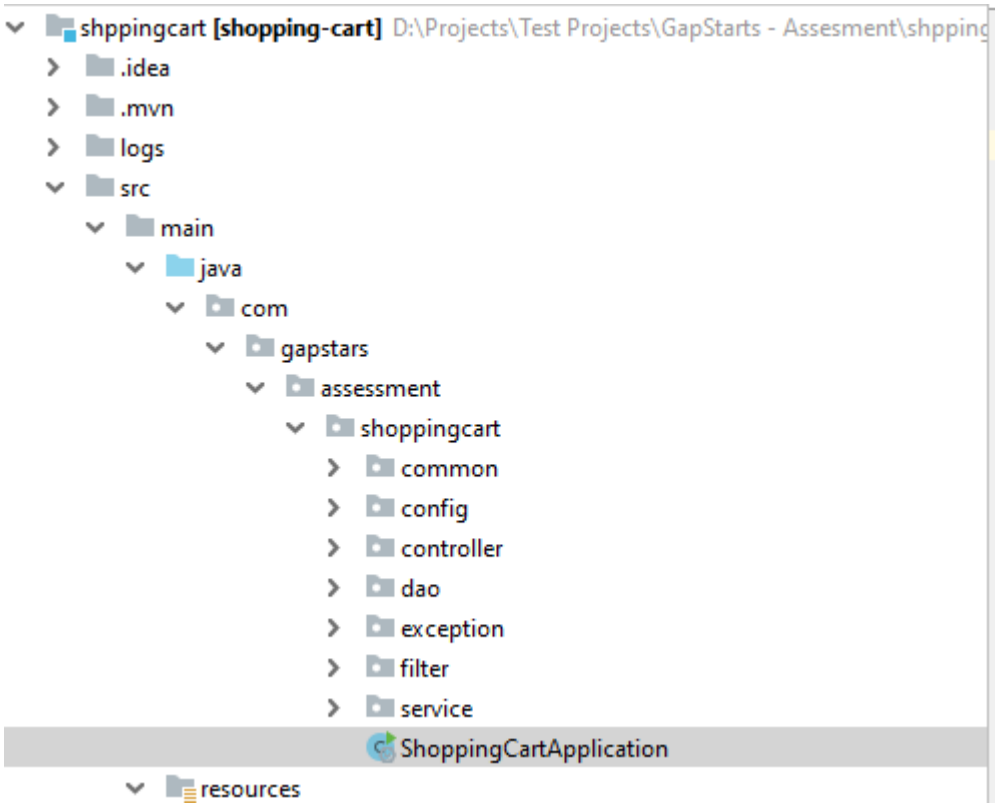
Assumptions: Java 11, Maven and MySql are already configured.

Execute command “mvn clean install -DskipTests” and application will build successfully.



Step 6 – Running the application

Run the class “ShoppingCartApplication.java”


shppingcart [shopping-cart] D:\Projects\Test Projects\GapStarts - Assesment\shpping
.idea
.mvn
logs
src
main
java
com
gapstars
assessment
shoppingcart
common
config
controller
dao
exception
filter
service
ShoppingCartApplication
resources

Special Note

With Main Application class extended to command line runner, it will automatically execute several predefine methods on each Application run. Methods are directly related to Assessment.

```
package com.gapstars.assessment.shoppingcart;

import ...

@Slf4j
@SpringBootApplication
public class ShoppingCartApplication implements CommandLineRunner {

  @Autowired( required = false )
  CustomerService customerService;

  public static void main(String[] args) {
    SpringApplication.run(ShoppingCartApplication.class, args);
  }

  @Override
  public void run(String... args)  {

      // Assessment Task 1
      createTwoCustomers();
      // Assessment Task 2
      addProductToFirstCustomer();
      // Assessment Task 3
      calculateFirstCustomerCartAmounts();
      // Assessment Task 4
      addProductsToSecondCustomer();
      // Assessment Task 5
      calculateSecondCustomerCartAmounts();
  }
```

There are five methods implemented focusing on the assessment tasks.

1. createTwoCustomers() – Assessment Task 1
2. addProductToFirstCustomer() - Assessment Task 2
3. calculateFirstCustomerCartAmounts() – Assessment Task 3
4. addProductsToSecondCustomer() - Assessment Task 4
5. calculateSecondCustomerCartAmounts() - Assessment Task 5

Also Swagger API is also integrated for API testing. Please refer to section "Testing the application"for more information.

Testing the application

Application can be tested via two methods,

1. Predefined methods
2. Swagger UI

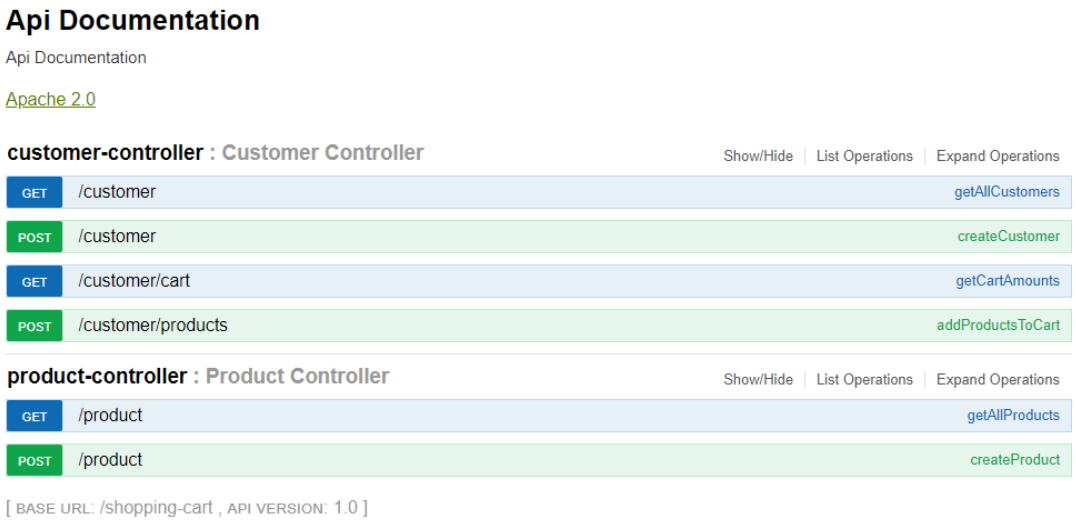
Testing via predefined methods

Please refer to previous section “Special Note”

This method executes with pre-set static values. Please refer to class “ShoppingCartApplication.java” run() method.

Testing via Swagger UI

Swagger UI URL: http://localhost:9090/shopping-cart/swagger-ui.html



Performing Assessment Task 1 – Create Two Customers

API URL	/customers
HTTP Method	POST
Request Params	• firstName • lastName
Response Params	• id - Id of created customer

Performing Assessment Task 2 – Add product to first customer

API URL	/customer/products
HTTP Method	POST
Request Params	• customerId – id of the created customer (Returned from previous API call or can be retrieved by calling API "/customer" in Customer Controller) • productIds – Existing Product Id's can be retrieved via API "/product" on Product Controller.
Response Params	• cartId – Customer cart id

Performing Assessment Task 3 – Calculate Amounts

API URL	/customer/cart
HTTP Method	GET
Request Params	• cartId – Customer's cart id (Returned from previous API call or can be retrieved by calling API "/customer" in Customer Controller)
Response Params	• status • shippingAmount – Card shipping amount • totalVatAmount – Cart total vat amount • totalCartAmount – Total price of all products • totalToBeSettled – Sum of shipping,vat,price

Performing Assessment Task 4 – Add products to second customer

Perform the same API twice as in Task 2.

Performing Assessment Task 5 – Calculate Amounts

Perform the same API as in Task 3.

Application Features

Additional Features

- Additional APIs implemented.
    - Get all Customers API
    - Get all Products API
    - Add Product API
- Product's available quantity handling implementation.
- Product Titles managed separately in both application layer and data layer.
- High-level Architecture Diagram
- ER Diagram

Improvements and Optimizations

- Swagger implementation
    - API Docs implementation URL : http://localhost:9090/shopping-cart/v2/api-docs
    - API Docs UI implementation URL : http://localhost:9090/shopping-cart/swagger-ui.html
- Application loggers
    - Rollback policy implementation
    - MDC logging with MDC Filter
- Exception Handling
    - Business Exception implementation
    - Global Exception Handler implementation
- Spring Boot Data Validation Constraint Layer implementation for API Payload Request classes.
- Application specific property file added "shopping-cart.properties"