ONLINE FOOD ORDER APPLICATION

IIHT

Time To Complete: 3 hrs

CONTENTS

1 PROBLEM STATEMENT

The Online Food Order Application allows users to perform CRUD (Create, Read, Update, Delete) operations and search functionalities in different criterias on dishes and restaurants. Users can create new dishes and restaurants, update existing dish and restaurant information, delete dishes and restaurants that are no longer available or relevant, and retrieve dish and restaurant details for viewing.

2 BUSINESS REQUIREMENTS:

a)

Screen Name	Console input screen
Problem Statement	1. User needs to enter into the application. 2. The user should be able to do the particular operations 3. The console should display the menu 2) create restaurant 3) create dish 4) update restaurant 5) update dish 6) delete restaurant 7) delete dish 8) get restaurant by id 9) get all restaurant 10) get all dishes 11) search restaurants by name 12) search restaurants by location 13) search restaurants by dishname 14) search dishes by ingredients 15) remove all dishes 16) delete all restaurants 17) exit

3 IMPLEMENTATION/FUNCTIONAL REQUIREMENTS

3.1 CODE QUALITY/OPTIMIZATIONS

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

3.2 TEMPLATE CODE STRUCTURE

a. PACKAGE: COM.ONLINEFOODORDERAPPLICATION

Resources

Class/Interface	Description	Status
OnlineFoodOrderApplication.java(class)	This represents bootstrap class i.e class with Main method, that shall contain all console interaction with user.	Partially Implemented

b. PACKAGE: COM.ONLINEFOODORDERAPPLICATION.MODEL

Resources

Class/Interface	Description	Status
Dish.java(class)	This represents entity class for Dish	Partially Implemented
Restaurant.java(class)	This represents entity class for Restaurant	Partially Implemented

c. PACKAGE: COM.ONLINEFOODORDERAPPLICATION.REPOSITORY

Resources

Class/Interface	Description	Status
DishDao.java(interface)	This is an interface containing declaration of DAO method	Already Implemented
DishDaoImpl.java(class)	This is an implementation class for DAO methods. Contains empty method bodies, where logic needs to written by test taker	Partially Implemented
RestaurantDao.java(interface)	This is an interface containing declaration of DAO method	Already Implemented
RestaurantDaoImpl.java(class)	This is an implementation class for DAO methods. Contains empty method bodies, where logic needs to written by test taker	Partially Implemented

4 EXECUTION STEPS TO FOLLOW

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. To build your project use command:

```
mvn clean package -Dmaven.test.skip
```

4. This editor Auto Saves the code.
5. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
6. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

7. Default credentials for MySQL:
   a. Username: root
   b. Password: pass@word1

8. To login to mysql instance: Open new terminal and use following command:

a. sudo systemctl enable mysql

b. mysql -u root -p

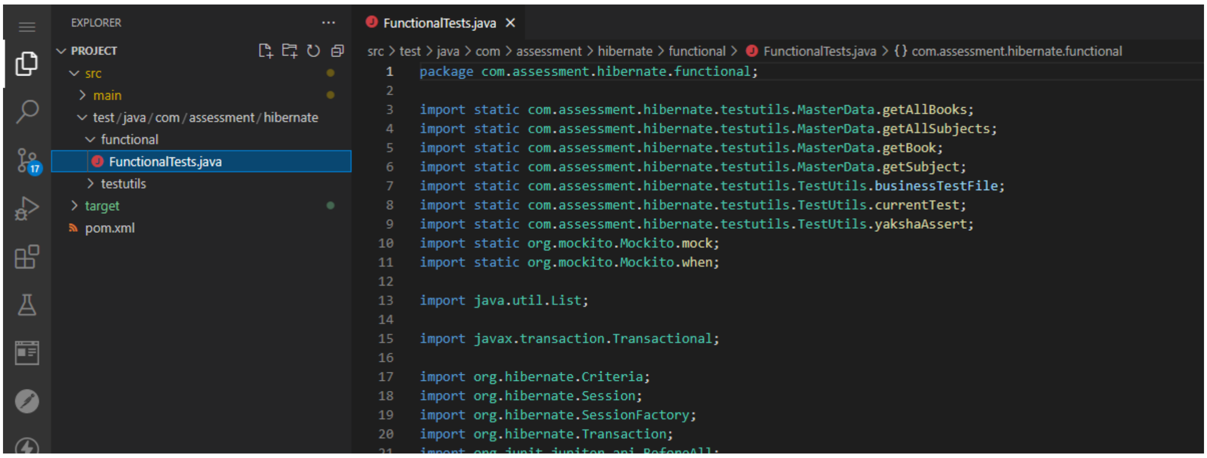
The last command will ask for password which is 'pass@word1'

9. These are time bound assessments. The timer would stop if you logout (Save & Exit) and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
10. To run your project use command:

```
mvn clean install exec:java –
Dexec.mainClass="com.onlinefoodorderapplication.OnlineFoodOrderApplication"
```

11. To test your project, use the command

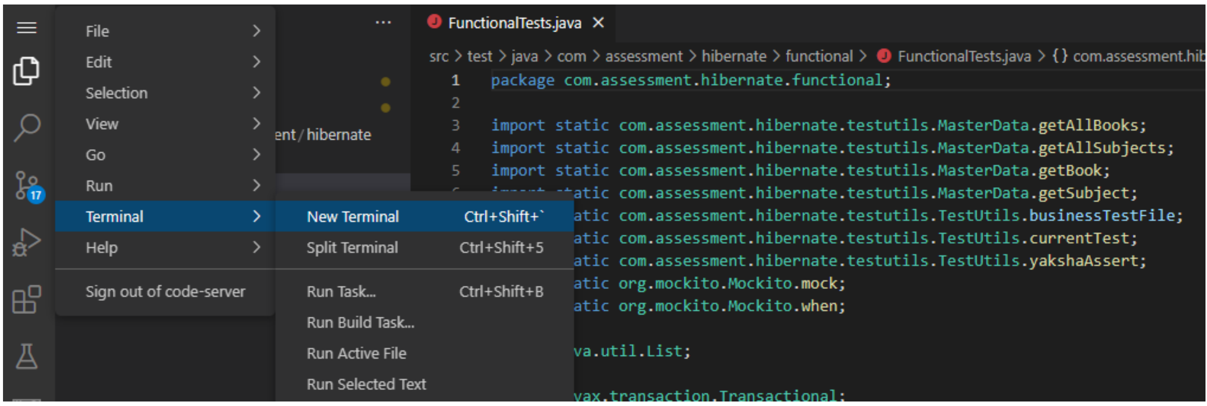
a. Open FunctionalTests.java file in editor



EXPLORER
PROJECT
src
main
test / java / com / assessment / hibernate
functional
FunctionalTests.java
testutils
target
pom.xml
FunctionalTests.java
src > test > java > com > assessment > hibernate > functional > FunctionalTests.java > {} com.assessment.hibernate.functional
17

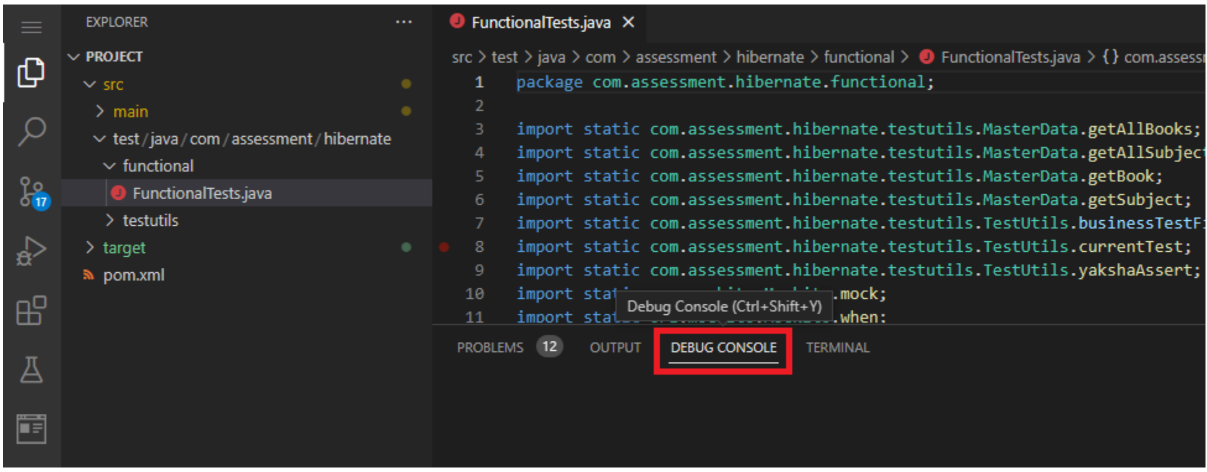
```java
package com.assessment.hibernate.functional;

import static com.assessment.hibernate.testutils.MasterData.getAllBooks;
import static com.assessment.hibernate.testutils.MasterData.getAllSubjects;
import static com.assessment.hibernate.testutils.MasterData.getBook;
import static com.assessment.hibernate.testutils.MasterData.getSubject;
import static com.assessment.hibernate.testutils.TestUtils.businessTestFile;
import static com.assessment.hibernate.testutils.TestUtils.currentTest;
import static com.assessment.hibernate.testutils.TestUtils.yakshaAssert;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

import java.util.List;

import javax.transaction.Transactional;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.junit.jupiter.api.BeforeAll;
```

b. Open a new Terminal



```java
package com.assessment.hibernate.functional;

import static com.assessment.hibernate.testutils.MasterData.getAllBooks;
import static com.assessment.hibernate.testutils.MasterData.getAllSubjects;
import static com.assessment.hibernate.testutils.MasterData.getBook;
atic com.assessment.hibernate.testutils.MasterData.getSubject;
atic com.assessment.hibernate.testutils.TestUtils.businessTestFile;
atic com.assessment.hibernate.testutils.TestUtils.currentTest;
atic com.assessment.hibernate.testutils.TestUtils.yakshaAssert;
atic org.mockito.Mockito.mock;
atic org.mockito.Mockito.when;

va.util.List;

vax.transaction.Transactional;
```

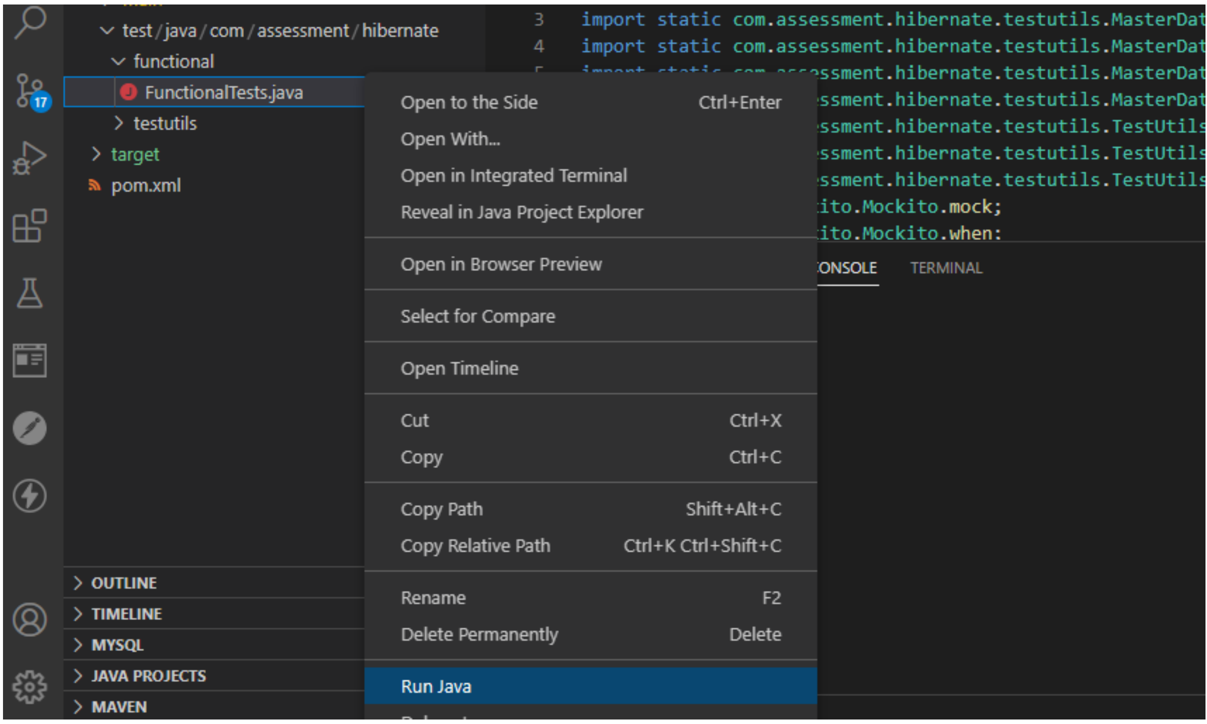
c. Go to Debug Console Tab



EXPLORER
PROJECT
src
main
test / java / com / assessment / hibernate
functional
FunctionalTests.java
testutils
target
pom.xml
FunctionalTests.java
src > test > java > com > assessment > hibernate > functional > FunctionalTests.java > {} com.assess
17
Debug Console (Ctrl+Shift+Y)
PROBLEMS 12
OUTPUT
DEBUG CONSOLE
TERMINAL

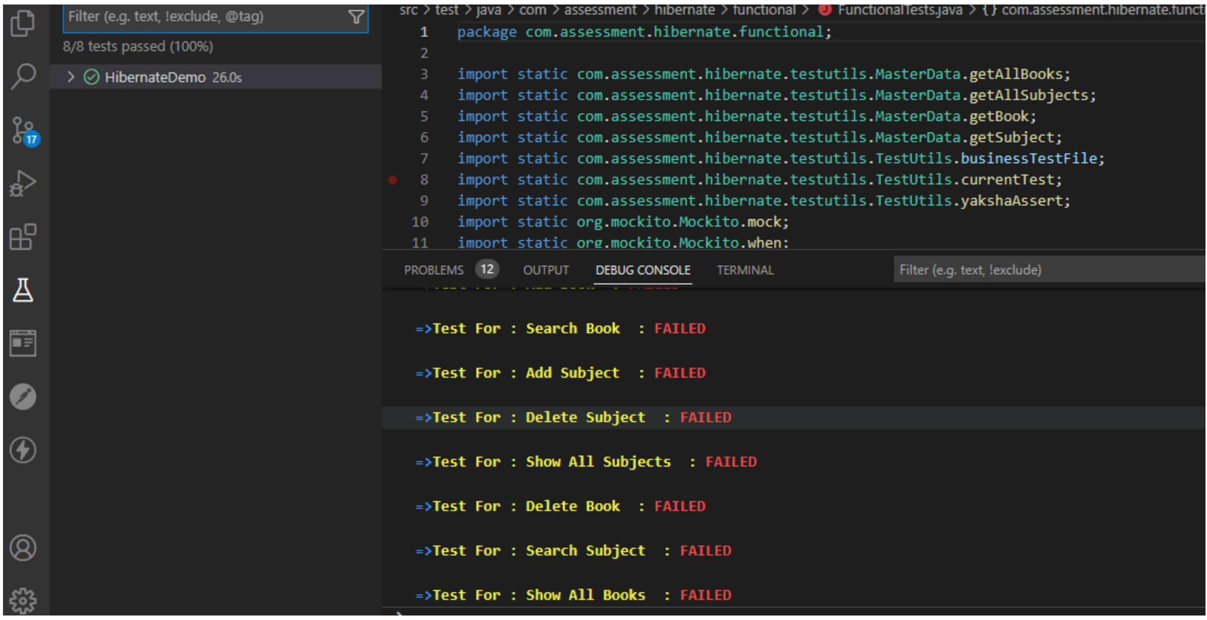
```java
package com.assessment.hibernate.functional;

import static com.assessment.hibernate.testutils.MasterData.getAllBooks;
import static com.assessment.hibernate.testutils.MasterData.getAllSubjec
import static com.assessment.hibernate.testutils.MasterData.getBook;
import static com.assessment.hibernate.testutils.MasterData.getSubject;
import static com.assessment.hibernate.testutils.TestUtils.businessTestF
import static com.assessment.hibernate.testutils.TestUtils.currentTest;
import static com.assessment.hibernate.testutils.TestUtils.yakshaAssert;
import stat                        .mock;
import stat                        .when;
```

d. **Right click on FunctionalTests.java file and select option Run Java**



```
import static com.assessment.hibernate.testutils.MasterDat
import static com.assessment.hibernate.testutils.MasterDat
ssment.hibernate.testutils.MasterDat
ssment.hibernate.testutils.TestUtils
ssment.hibernate.testutils.TestUtils
ssment.hibernate.testutils.TestUtils
ito.Mockito.mock;
ito.Mockito.when:
```

e. **This will launch the test cases and status of the same can be viewed in Debug Console**


Filter (e.g. text, !exclude, @tag)
8/8 tests passed (100%)
HibernateDemo 26.0s
src > test > java > com > assessment > hibernate > functional > FunctionalTests.java > { } com.assessment.hibernate.funct
PROBLEMS 12
OUTPUT
DEBUG CONSOLE
TERMINAL
Filter (e.g. text, !exclude)

```
package com.assessment.hibernate.functional;

import static com.assessment.hibernate.testutils.MasterData.getAllBooks;
import static com.assessment.hibernate.testutils.MasterData.getAllSubjects;
import static com.assessment.hibernate.testutils.MasterData.getBook;
import static com.assessment.hibernate.testutils.MasterData.getSubject;
import static com.assessment.hibernate.testutils.TestUtils.businessTestFile;
import static com.assessment.hibernate.testutils.TestUtils.currentTest;
import static com.assessment.hibernate.testutils.TestUtils.yakshaAssert;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when:
```

```
=>Test For : Search Book  : FAILED

=>Test For : Add Subject  : FAILED

=>Test For : Delete Subject  : FAILED

=>Test For : Show All Subjects  : FAILED

=>Test For : Delete Book  : FAILED

=>Test For : Search Subject  : FAILED

=>Test For : Show All Books  : FAILED
```

12. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**