System Requirements Specification Index

For

EBill-Junit

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

E-BILL
System Requirements Specification

1 Project Abstract

In the world of business billing, a new challenge arises: ensuring the reliability and accuracy of the billing system. The Java-E-Bill project presents developers with a unique task: to design and implement comprehensive test cases using JUnit to validate the functionality of the billing system.

Your task is to develop a robust set of test cases that thoroughly evaluate the billing system's behavior and ensure its correctness under various scenarios.

The **Java-E-Bill** test suite should promise to provide confidence in the reliability and accuracy of the billing system, ensuring smooth operations and customer satisfaction.

2 Code Structure

2.1 Package: Com.Ebill.Service

Resources

Class/Interface	Description	Status
EbillService(class)	• This class represents a service for calculating electricity bills based on the consumed units. • It takes the consumed units as input and calculates the bill amount according to predefined rates. • Don't modify any in this class as this is already implemented.	Already implemented.

2.2 Package: Com.Ebill.Test

Resources

Class/Interface	Description	Status
EbillTest(class)	• This class should contain JUnit test cases to verify the correctness of the calculateBillAmount() method in the EbillService class. • Each test case should instantiate the EbillService class with a specific input value representing consumed units and asserts that the calculated bill amount matches the expected value. • These test cases should ensure that the billing logic implemented in the EbillService class produces accurate results for different scenarios of consumed units. • Make sure the test cases you write achieves 100% code coverage.	To be implemented.

3 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 Application menu (Three horizontal lines at left top) → Terminal →New Terminal.**
3. **This editor Auto Saves the code.**
4. **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.**
5. **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.**
6. **To execute and run test cases:**
   mvn clean install exec:java -Dexec.mainClass="mainapp.MyApp" -DskipTests=true
7. **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.**