A Project Report On

Software Testing and Quality Assurance (Mini Project I)

SUBMITTED BY

Tejas Dahad	**Roll No:41171**
Varun Karwa	**Roll No: 41174**
Prachi Wagh	**Roll No: 41176**

CLASS: BE-1

GUIDED BY

Prof M.S.Chavan

DEPARTMENT OF COMPUTER ENGINEERING

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

DHANKAWADI, PUNE-43

SAVITRIBAI PHULE PUNE UNIVERSITY

2021-22

Title:

Create a small web-based application by selecting relevant system environment/platform and programming languages. Narrate concise Test Plan consisting features to be tested and bug taxonomy. Prepare Test Cases inclusive of Test Procedures for identified Test Scenarios. Perform selective Black-box and White-box testing covering Unit and Integration test by using suitable Testing tools. Prepare Test Reports based on Test Pass/Fail Criteria and judge the acceptance of application developed.

Problem Definition:

Perform Desktop Application testing using unittest library in java.

Objective

We are going to learn how to Prepare Test Cases inclusive of Test Procedures for identified Test Scenarios. Perform selective Black-box and White-box testing covering Unit and Integration test by using suitable Testing tools. Prepare Test Reports based on Test Pass/Fail Criteria.

Theory

Test Plan for Application Testing

The Test Plan document is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents. The focus of the test is what to test, how to test, when to test, and who will test. Test plan document is used as a communication medium between test team and test managers.

A standard test plan for Application Testing should define following features;

- Define the scope of testing
- Define objective of testing
- Approach for testing activity
- Schedule for testing
- Bug tracking and reporting

UNIT TESTING : UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

INTEGRATION TESTING : INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing

Extreme Programming & Unit Testing

Unit testing in Extreme Programming involves the extensive use of testing frameworks. A unit test framework is used in order to create automated unit tests. Unit testing frameworks are not unique to extreme programming, but they are essential to it. Below we look at some of what extreme programming brings to the world of unit testing:

· Tests are written before the code

· Rely heavily on testing frameworks

· All classes in the applications are tested

· Quick and easy integration is made possible

Unittest library java

The unittest unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

Application Tested :

A healthcare system is built to support addition of patients, doctors, functionalities, etc.
These operations are tested using some test cases.

Test Plan :

The main aim is to check if we are getting correct output for each given input and if the test is being executed correctly .

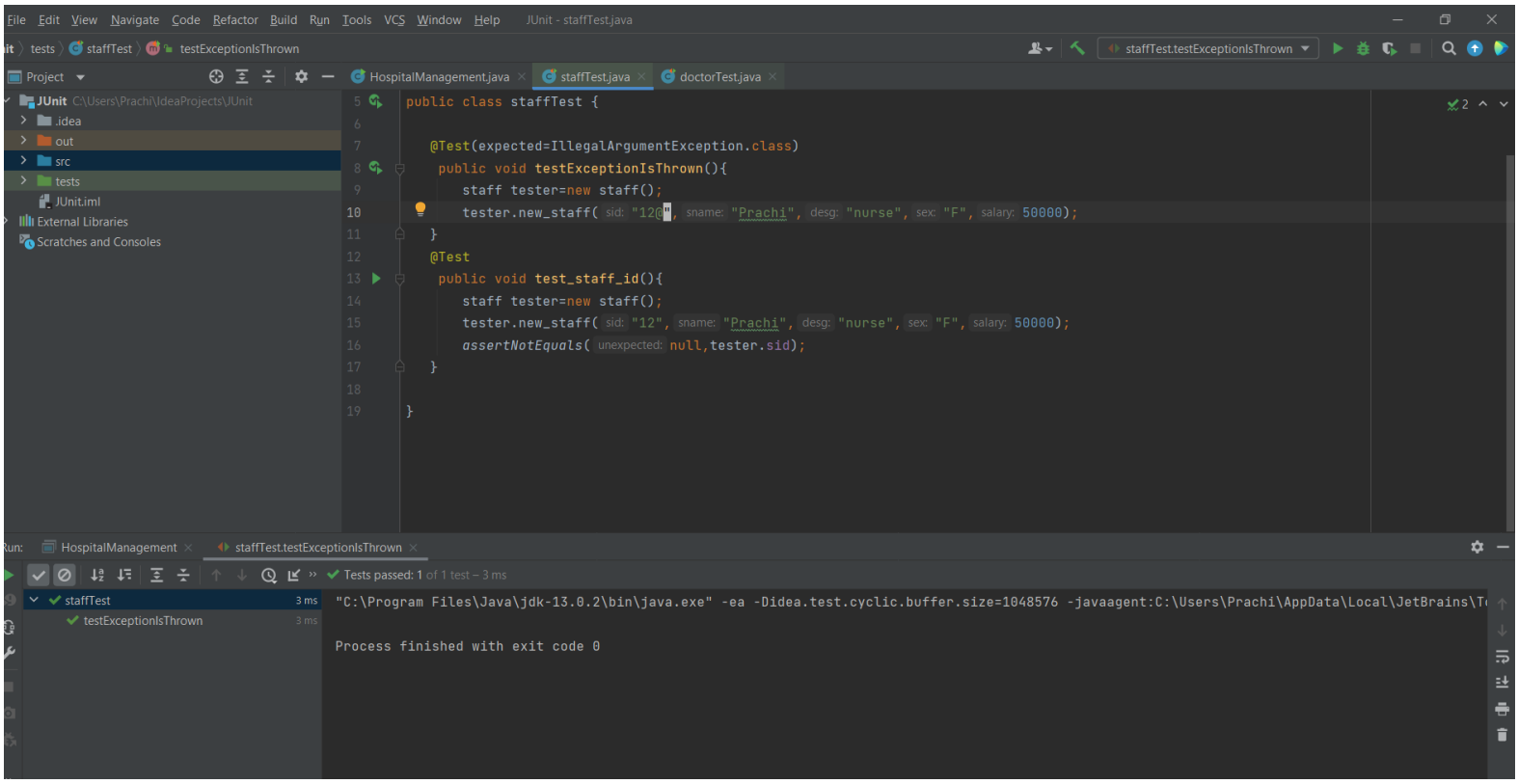
- We start with giving varied inputs to perform CRUD operations.
- Test cases include authentication for different values for Doctor salary, id and patient name fields for our database.
- Finally we get the test reports, which indicate the overall performance of our tests, i.e. either test pass or test fail.

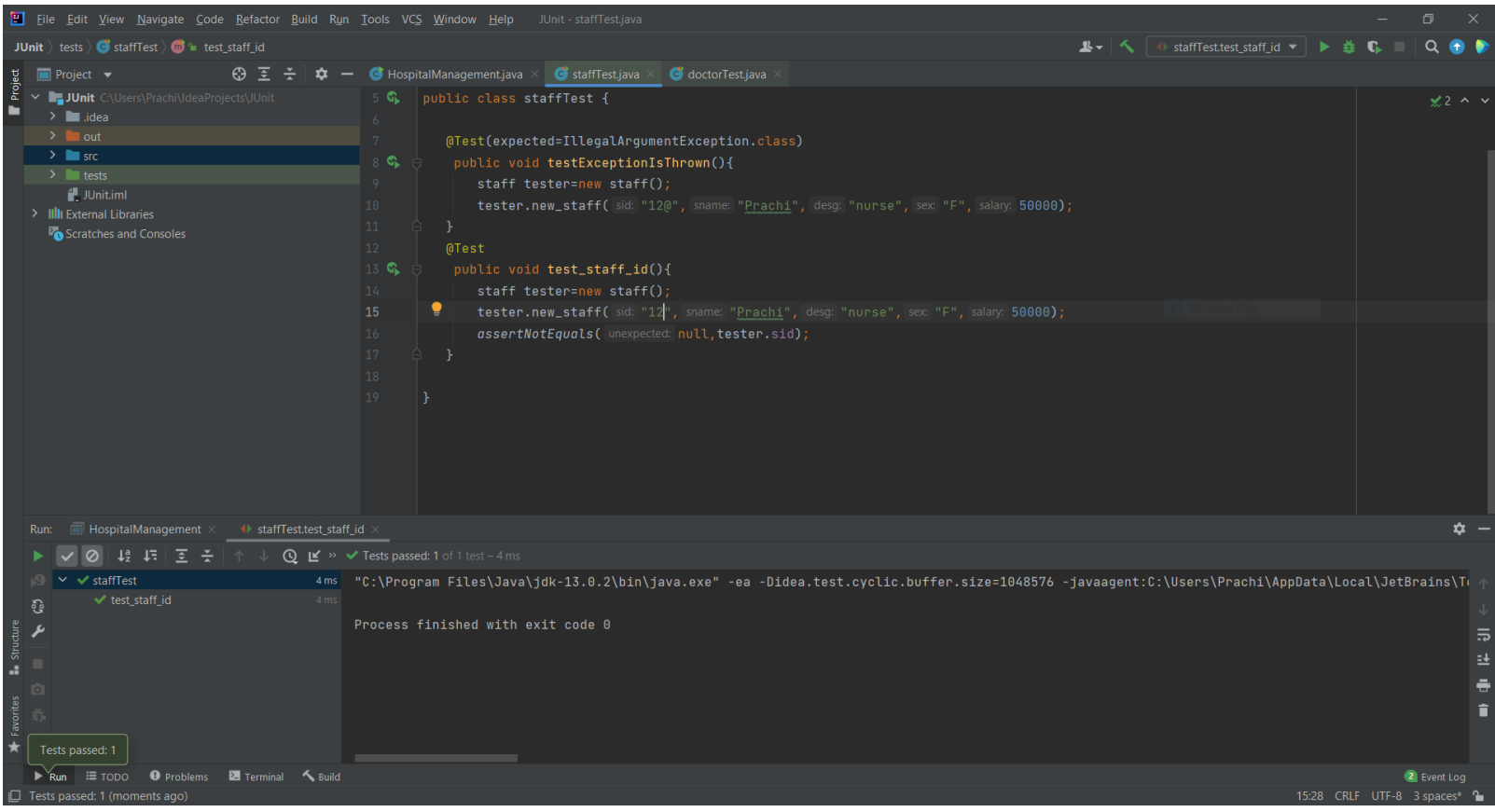
SAMPLE UNIT TESTS:

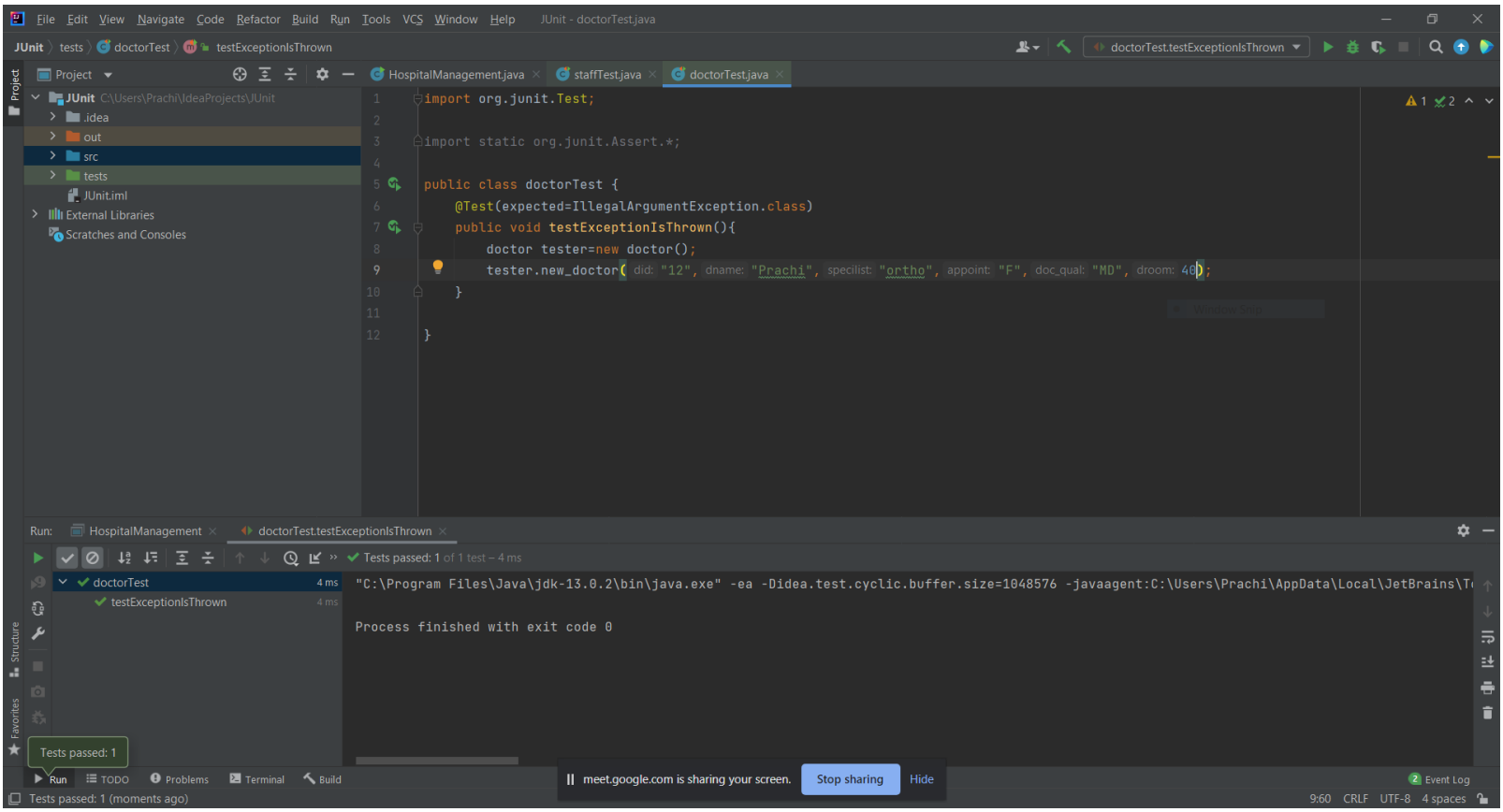
```
import org.junit.Test;

import static org.junit.Assert.*;

public class doctorTest {
   @Test(expected=IllegalArgumentException.class)
   public void testExceptionIsThrown(){
      doctor tester=new doctor();
      tester.new_doctor("12","Prachi","ortho","F","MD",40);
   }

}
```

```
import org.junit.Test;

import static org.junit.Assert.*;

public class staffTest {

  @Test(expected=IllegalArgumentException.class)
   public void testExceptionIsThrown(){
      staff tester=new staff();
      tester.new_staff("12@","Prachi","nurse","F",50000);
  }
  @Test
   public void test_staff_id(){
      staff tester=new staff();
      tester.new_staff("12","Prachi","nurse","F",50000);
      assertNotEquals(null,tester.sid);
  }
```

Screenshots of unit tests.







Conclusion:

Hence, we have successfully performed Unit testing on a HealthCare System application performing CRUD operations.