DORCAS TABITHA SINDE

CS 320

10/13/22

JOURNAL PAPER WK 7

Software testing is a fundamental pillar in the modern development initiatives and played a

major role in the successful achievement of my project .There are various types of testing

categories that can be employed depending on the project requirements and its main

purpose.

The main purpose of the project as mentioned in the rubric was to verify that

the **Contact class** meets the requirements through JUnit tests, verify the **ContactService**

**class** meets the requirements through JUnit tests, verify the **Task class** meets the requirements

through JUnit tests, verify the **TaskService class** meets the requirements through JUnit tests,

verify the **Appointment class** meets the requirements through JUnit tests, verify

the **AppointmentService class** meets the requirements through JUnit testsEnsure the **test**

**coverage** for the java files has 80% coverage or higher. While working on this project I

employed both black box and white box categories since they are both specifications based, or

structure based. Black box techniques I employed included equivalence partitioning to test for

both valid and invalid inputs, decision tables to test conditions and actions ,and state

transitioning testing that tested events that change the state and generate outputs ,using cases that

are made from test cases and boundary values to test boundaries. I used structured based

techniques a lot in the coverage tests to analyze components and when applying the if then

statements , to break down the tests into sections to be tested. Structure based techniques were

used in this project to explore system or component structures at several levels.

However, there are other manual testing techniques that I did not employ in my project which

includes unit testing where each component or individual units of the software shall be tested

which aims in checking internal data structures ,logic boundary conditions for input and output

data as per the design . Another manual technique I failed to employ was system technique that

tests the fully integrated application to evaluate the systems compliance with its specified

requirements. Integration testing involves testing the interface between the two software units

and lastly acceptance testing that is done by the end users along with the testers to validate the

functionality of the application.

The overall quality of the Junit tests is about 50% and was technically sound so it made it easy to

write the codes that would run a good test that will meet the requirements. However, I did not as

much learn how to test for a failed code and fix any associated bugs which is important but I

ensured the code was efficiently written by writing all my class service classes in different

approaches. Using the appointment service class as an example of how I followed the rubric

while writing my Junit test cases. As per the requirements;

The appointment object shall have a required unique appointment ID String that cannot be longer than 10 characters. The appointment ID shall not be null and shall not be updatable.

The appointment object shall have a required appointment Date field. The appointmentDate field cannot be in the past. The appointmentDate field shall not be null. *Note: Use java.util.Date for the appointmentDate field and use before(new Date()) to check if the date is in the past.*

The appointment object shall have a required description String field that cannot be longer than 50 characters. The description field shall not be null.

public class Appointment {

//representing my required variables

private String appointmentID;

private String date;

Date date1 = new Date();//new date check

private String description;

//initializing my constructor

public Appointment(String appointmentID, String date, String description) {

if(appointmentID.length()<=10&&appointmentID != null) {//appointment object shall have an id string not longer than 10 characters

this.appointmentID = appointmentID;

I also made sure the appointment was deleted ;

public boolean deleteAppointment(String ID) { //initializing the delete function

boolean erase = false;

for(Appointment appointment : appointments) {

if(appointment.getID().equalsIgnoreCase(ID)) {

appointments.remove(appointment); //initializing my remove function

erase = true;

break;

}

}

return erase;

}

Working on these projects has definitely changed my mindset on effective methods to

implement before writing a code, by applying greater cautions on any bugs that needs to

be fixed to ensure the codes run effectively . I can say I applied less attention in making

global variables private instead of public and did not take into consideration the different

possible values of testing, or checking for unnecessary objects that did not need to be

included within some of the service classes.