CS29006 Software Engineering

Test Plan

For

Factory Service Simulation Software

(FSSS)

- Group 12
  - Ayush Pattnayak(19CS10014)
  - Kamalesh Garnayak(19CS10074)
  - Sayantan Saha(19CS30041)

1. References

The websites/documents used as go-to references are:

- IEEE-829 standard
- TutorialsPoint/TestPlan
- Google
- Stack Overflow

2. Introduction

2.1. Objectives

This Test Plan document is the Master Plan for the Factory Service Simulation Software (FSSS). It intends to lay out all the testing strategies involved in verifying the software. By means of testing, we can quickly iron out any bugs/errors at the time of software release.
The contents of this document are to be interpreted in the pre-implementation stage. It addresses the testing strategies to be used. As mentioned in the 'References' section, all IEEE Standard-829 (1987) specifications are adhered to appropriately.

2.2. Scope

The scope of testing extends throughout the life cycle of the product. The software may be tested whenever any issues are found at a later stage.
The testing process may also be updated if needed.

3. Test Items(Functions)

3.1. Unit Testing

Unit testing aims to find the program errors such as wrong logic, wrong output in UI and test the individual classes/functions part of the code. All the classes and the GUI output are to be verified through unit testing.

Process for Construction of object:

We consider some general cases of Machine Type and Adjuster for each Service Manager. The quantity can be chosen in any manner as long as all the constraints are being checked.

3.1.1. Class MachineType:-

Constructor scenarios:

- Object with a valid name, MTTF, repair time and number of machines
- Invalid name
- MTTF input is negative or zero (sign handling)
- Number of machines is > 1000 or <= 0
- Repair time input has an upper bound of Simulation time.

Methods:

- Comparing equality of string input name with Get_Name output
- Comparing equality of integer input MTTF with Get_MTTF output
- Comparing equality of integer input Time_Repair with Get_Time_Repair output
- Comparing equality of integer input no_machines with Get_machine_no output

3.1.2. Class AdjusterType:-

Constructor scenarios:

- Object with valid name and Arraylist of repairable machines
- Invalid name
- List of machines contains unavailable machines

Methods:

- Comparing equality of string input name with Get_Name output
- Comparing equality of Arraylist input Repair_Machine with Get_Repair_machine output

3.1.3. Class Machine:-

Constructor scenarios:

- Object with a valid name, MTTF, repair time and number of machines
- Invalid name
- MTTF input is negative (sign handling)
- Number of machines is >1000 or <=0
- Repair time input is > Simulation time or is <=0

Methods:

- Comparing equality of string input name with Get_Name output
- Comparing equality of integer input MTTF with Get_MTTF output
- Comparing equality of integer input Time_Repair with Get_Time_Repair output
- Comparing equality of integer input no_machines with Get_machine_no output
- Comparing equality of string input ID with Get_Id output
- Comparing equality of integer input of Working Status with Get_Working_Status output
- Comparing equality of integer input of Repairing Status with Get_Repairing_Status output
- Comparing equality of integer input of Working days with Get_Working_days output
- Comparing equality of integer input of Total Working days with Get_total_Working_days output
- Comparing equality of integer input of Required days with Get_req_days output
- Checking that MTTF, Working days, Required days and Time_Repair are lesser than Total Working Days and >=1

3.1.4. Class Adjuster:-

Constructor scenarios:

- Object with a valid name, ID and Arraylist of repairable machines
- Invalid name
- ID input is negative (sign handling)

- List of machines contains unavailable machines

Methods:

- Comparing equality of string input name with Get_Name output
- Comparing equality of integer input ID with Get_Adjuster_no output
- Comparing equality of Arraylist input Repair_Machine with Get_Repair_machine output
- Comparing equality of integer input of Working Status with Get_Working_Status output
- Comparing equality of integer input of Working days with Get_Working_days output
- Comparing equality of integer input of Required days with Get_req_days output
- Checking that Working days and Required days are lesser than Total Working Days and >=1

3.1.5. Class ServiceManager:

Constructor scenarios:

- Object with valid username and password of repairable machines
- Password has both alphabets and numeric characters

Methods:

- Comparing equality of string input username with Get_UserId output
- Comparing equality of string input password with Get_Password output

3.2. Interface Testing

The Factory Service Simulation software project’s user interaction is limited to a local GUI, where he/she can perform various functions like giving input, changing input, getting simulation results, etc. Within this, there are two stages: (i) testing Login page with various combinations of Username/Password (ii) Testing the GUI

Simulation output after proper input is supplied. All this is done in the backend, with appropriate mouse-click-events being checked for correctness too.

The interfaces being tested here are:

- ServiceManager
- Adjuster
- AdjusterType
- PageAddAdjuster
- PageDelAdjuster
- PageEditAdjuster
- Machine
- MachineType
- PageAddMachine
- PageDelMachine
- PageEditMachine
- FirstPage
- SecondPage
- ThirdPage
- ShowAdjuster
- ShowMachine
- SelAdjuster *(Selecting adjuster from database)*
- SelMachine *(Selecting machine from database)*
- AdjUtil *(Adjuster utilisation)*
- MacUtil *(Machine utilisation)*
- Utilisation

4. Features to be tested

(Use Case Testing)

1. When an account is being created and any of the three fields: ‘Enter User Id’,’Enter password’ or ‘Confirm Password’ is left empty, “Incomplete registration” should be shown.
2. If Confirm password input does not match with Enter password field, “Password does not match” should be informed to the user.
3. When there is no account in the database and the Service manager tries to log in, a “Create Account” message should be shown.

4. Every company has a single service manager only who can handle the factory. So if a second account is being created, we interpret that the service manager of the company intends to refresh the details present previously. Therefore all of the company's previous data will be lost.
5. When the username entered by the manager is present in the system, but the password is incorrect, "Wrong password" should be shown.
6. For all the buttons on the home page, if any field is left empty, there should be an error message.
7. For 'Add machine'/'Add adjuster', if the MTTF field/ 'Repair time' / 'Quantity' is given a negative number or zero as input, it should be shown invalid. These also have an upper bound value of 1000 days / 100 days / 1000 each.
8. For 'Edit machine', if the ''Quantity' is given a negative number or zero as input, it should be shown invalid. Again it also has an upper bound of 1000. This holds for both database and simulation.
9. For 'Add machine'/'Add adjuster' if a new machine/adjuster is added with the same name as an already existing one, an error message will be shown, i.e."Machine(adjuster) already exists". This holds for both database and simulation.
10. When no machines are present, clicking 'Add adjuster'/'Delete adjuster' should give an error.
11. In the Simulation specification, if the quantity of selected machine/adjusters is entered to be greater than the already present machine/adjuster quantity in the database, "Select a valid number of machines" error should appear.
12. When machines/adjusters are not populated, 'Simulate' should give an error.

5. Testing Approach

The aim of testing is to compare the present and ideal versions of the software, identify the features that may be improved and evaluate the achieved functionalities. The test plan has the following components:

- Purpose for this level of test
- Management and technical approach
- Pass/Fail criteria
- Hardware/Software requirements(covered in SRS)

5.1. White Box Testing;

The UI is bypassed in white box testing. The focus is on the input and corresponding output at the local level, and it is verified with the expected output. Only the code and its structure are being checked through WBT, while the program verification is at standby.
All conditions and all their corresponding values are taken as test cases to cover all possibilities.

5.2. Black Box Testing:

Black box testing is more user-centric. It involves verifying whether the correct output is visible to the user or not at the GUI level. All possible inputs for this testing method are obtained by Error guessing strategy and getting all input corner cases.

Subsection 3.1(Unit testing) involves white box testing methods, and 3.2 (Interface testing) involves black box testing.

6. Pass/Fail Criteria

6.1. Suspension criteria

The test is considered suspended if any of the following is encountered:
1. The program crashes
2. The software produces incorrect output/error message
3. The software takes more than expected time to produce the output

6.2. Resumption criteria

In case the testing fails at an early stage, there is no point in continuing further tests; the errors in logic will be ironed out, and updated testing will be done.

6.3. Approval criteria

The test is considered approved if

1. The software produces the correct output as per the client's demand,
2. The necessary error messages whenever the user inputs corner cases.

The test suite elaborates on the expected results for configuration management.

7. Glossary

Service Manager: One who maintains machine and adjuster queue and assigns machines to the adjusters. He/she can get statistics of machine and adjuster utilisation.
Adjuster: One who can repair inoperative machines.
Machine Queue: One queue which contains all the inoperative machines in order.
Adjuster Queue: One queue which contains all the available free adjusters ready to repair faulty machines.
FSSS: Factory Service Simulation Software
'Del': Delete chosen machine/adjuster from the database.
Firstpage: Create account/Login page
Secondpage: Home page, shown after being logged in