Mobiquity assignment

Deriving test cases

I have used a mind map to derive test cases. When the functionality involves multiple pages, multiple data transfers it is good to use a mind map so you can easily identify the scenarios that involve in the system. I used a tool called 'xmind' to create the mind map.

Advantages of creating a mind map

- System can be divided into smaller branches
    - Login
    - Logout
    - Employee Details
        - View Employee Details (Subbranches of Employee Details)
        - Create Employee
        - Edit an employee
        - Delete an employee
    - Delete Employees
- Ability to view all test cases derived in a single file
- Can be shared to a developer/BA for any suggestions/feedbacks

I have attached both PDF version and original file of the mind map that I created. I used test design techniques like boundary value analysis, equivalence partitioning, decision tables to derive the test cases.

Test Implementation

From the test cases I derived, I identified test cases that can be automated. When identifying tests to automate I focused on:

- Actual value addition to the system by automation that specific test case
- Frequency of the test case being executed in the system (Repetitive test cases)
- Return on investment
- Tests that require a lot of time in manual verification (Multiple data sets)

Selecting the framework:

I selected BDD driven approach to automate my tests. I used cucumber, java, selenium with mmvn maven. Further I used Page Object Model as the design pattern. Reasons for selecting this approach:

- Cucumber acts as the collaborative tool/framework and gives clear idea of what is happening in the system
- Selenium as the automation tool since it is more powerful and open source
- Java as the programming language since as the Automation Engineer, I am much familiar in Java
- Maven, since it provides the Project Object Model and dependency management
- Page Object model to isolate tests from pages and for better maintainability
- Since reporting can be done by the inbuilt cucumber reporter plugins

Tests covered

Since I have identified tests that can be automated from all the test cases, I have added two tags in the Mind map as;

- Automatable = Tests that should be automated
- Automated = Tests that are automatable and currently automated

Automation coverage can be calculated as:

$$\text{Automation coverage} = \frac{\text{Total Automated Test cases}}{\text{Total Automatable Test cases}} \times 100\%$$

In my scenario:

- 15 automated test cases
- 24 automatable test cases
- 15/24 X 100% = 62.5%

Automation coverage is 62.5%

Please refer CafeTown.pdf for mind map in PDF version

Please refer CafeTown.xmind for original version (Need to download xmind software)

Some technical decisions I made while doing the project,

- Sharing state between two different cucumber JVM step classes using pico container.
- Using data tables and List Web elements for handling the multiple user inputs
- Implement the code in way that can be expanded in the future
- Strong assertions for user inputs