Session 3

Static and Dynamic Testing:

Static Testing Techniques:

1. **Review**

Conducts on Documents to ensure Correctness and Completeness.

Review could be done by anyone yet its planned.

Types of Reviews:

1. Requirements Review (To check on the requirements and verify them)
2. Design Review (To review the diagrams and pictures)
3. Code Review (To review the written code)
4. Test plan reviews, Test case review etc.
5. **Walkthrough**

It’s an informal review, i.e. having no plan and could be done at any time by anyone.

Author reads the documents and code and discuss with peers (here more than one person is required which is unlike the Review)

1. **Inspection**

Its more formal type of review, only concerned people are called through email and having a specific schedule.

In which 3-8 people sit in the meeting

(It involves 3 types of people:

1. Reader, generally the person who’d written the document will read.
2. Writer, the person who writes all the points discussed in the meeting.
3. Moderator is the Organizer between other groups

Dynamic Testing Techniques:

1. Unit
2. Integration
3. System
4. User Acceptance Testing (UAT)

**QA vs QC vs QE**

|  |  |
| --- | --- |
| QA (Quality Analysis) | QC (Quality Control) |
| Process related (they design the process) it could be said that, they are involved in every stage of SDLC | Actual testing of the software, they are only engaged in Testing phase |
| Analyst people | Only Testers |
| Focuses on building in quality (by following certain processes) | Focuses on testing for quality. |
| Prevents the defects | Detects the defects |

QE (Quality Engineer):

People who write the code for testing the software, i.e. Automation testers.

Levels of Testing:

1. **Unit Testing**

* A unit is a single component/module of software
* (ex. A login is one unit, delete email button is one module like that)
* A white box testing technique, Conducted by Developers.
* Techniques:
  + Basic path Testing (each line should be executed at least once
  + Control Structure Testing
  + Conditional coverage (checks for the conditions like if, else)
  + Loop coverage
  + Mutation Testing (It relates to testing according to multiple set of data),

for ex. Login credentials like valid and invalid combinations.

1. **Integration Testing**

* It is performed between two or more modules.
* It’s concerned about the flow of data between different modules.
* Tho it’s a White Box Testing, Testers could also perform it on UI/Application level.
* Types of Integration Testing:
  + **Incremental Integration Testing** and **Non**-**Incremental Integration Testing**
    - Incremental: Incrementally adding the modules testing data flow bw modules
    - It having 2 approaches i) Top Down & ii) Bottom-Up
    - Top Down (To ensure the newly added module is the child of previous module)
    - Bottom-Up (The module which is added should be the parent of the previous module)
    - Another is Sandwich/Hybrid Approach (mix of both)
    - Non-Incremental: It integrates all modules in a single shot and then tests.
    - But it is used less because of several reason like, we might miss data bw some of modules, it becomes difficult to find the root cause of defect.

1. **System Testing**

* Testing the overall functionality of the application with respect to client requirements
* It’s black box technique method.
* Conducted by Testing team.
* 4 area of concerns:
  + User Interface Testing
  + Functional Testing
  + Nonfunctional Testing
  + Usability Testing

1. **UAT (User Acceptance Testing)**

Here customers are actively involved along with Testers for the guidance.

It having 2 level:

Alpha and Beta Testing

Alpha Testing is carried out development and testing environment.

Beta Testing is about using the product as a customer.

**Positive and Negative Test Cases**

* Positive test cases verify that when the function under test is exercised using valid input, the input is accepted and the user is able to proceed with the flow.
* Negative testing, in opposition to positive testing, uses unexpected conditions to ensure the functionality does not do something that it should not do.
* Here’s a table format for **positive** and **negative test scenarios** for a **login page**:

| **Test Type** | **Test Scenario** | **Expected Result** |
| --- | --- | --- |
| **Positive Test Scenarios** | **Valid Username and Password** | User successfully logs in and is redirected to the dashboard/home page. |
|  | **Remember Me Checkbox** | User is logged in and remains logged in on subsequent visits. |
|  | **Password Field Masking** | Password is displayed as asterisks or dots (masked input). |
|  | **Case Sensitivity** | Login is successful with correct case-sensitive password, even if the username is case-insensitive. |
|  | **Successful Logout** | User successfully logs out and is redirected to the login page. |
|  | **Valid Email as Username** | User successfully logs in using a valid email address as the username. |
|  | **Login After Password Reset** | User can log in with the new password after a password reset. |
| **Negative Test Scenarios** | **Invalid Username and Valid Password** | Error message displayed: "Invalid username or password." |
|  | **Valid Username and Invalid Password** | Error message displayed: "Invalid username or password." |
|  | **Empty Username and Password** | Error message: "Please enter username and password." |
|  | **Username with Special Characters** | Error message or rejection of special characters in the username. |
|  | **Password Too Short** | Error message: "Password too short" (based on system’s minimum length requirements). |
|  | **Invalid Email Format** | Error message: "Please enter a valid email address." |
|  | **Account Locked After Multiple Failed Attempts** | Error message: "Too many failed login attempts. Please try again later." |
|  | **Password Field Not Masked** | Password is visible in plain text instead of masked input. |
|  | **Session Timeout After Inactivity** | Session expires, user is redirected to the login page. |
|  | **Login Button Disabled with Empty Fields** | Login button remains disabled until both username and password are entered. |

**Functional vs Non-Functional Testing:**

| **Functional Testing** | **Non-functional Testing** |
| --- | --- |
| It verifies the operations and actions of an application. | It verifies the behavior of an application. |
| It is based on requirements of customer. | It is based on expectations of customer. |
| Functional testing is easy to execute manually. | It is hard to execute non-functional testing manually. |
| Functional testing is based on the business requirement. | Non-functional testing is based on the performance requirement. |
| Ex. Unit, Integration testing | Ex. Performance, Load, Scalability Testing |

**How to decide which sdlc should be used?**

Choosing the right Software Development Life Cycle (SDLC) model depends on factors such as project requirements, size, client collaboration, development team, risk tolerance, time, budget, regulatory constraints, and stakeholder expectations.

**Boundary-value analysis**

Boundary-value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range.

Ex. Any values between 18 and 65 are valid values, while anything below the minimum or above the maximum is invalid.

Here’s a **short and basic manual test case sheet** for the **login page** of GitHub (<https://github.com/login>):

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| TC001 | Valid login with correct username and password | 1. Navigate to GitHub login page. 2. Enter a valid username. 3. Enter the correct password. 4. Click "Sign in". | Username: valid\_userPassword: valid\_password | User successfully logs in and is redirected to the dashboard. |
| TC002 | Invalid login with incorrect username and correct password | 1. Navigate to GitHub login page. 2. Enter an incorrect username. 3. Enter the correct password. 4. Click "Sign in". | Username: invalid\_userPassword: valid\_password | Error message: "Incorrect username or password." |
| TC003 | Invalid login with correct username and incorrect password | 1. Navigate to GitHub login page. 2. Enter a valid username. 3. Enter an incorrect password. 4. Click "Sign in". | Username: valid\_userPassword: invalid\_password | Error message: "Incorrect username or password." |
| TC004 | Login with empty fields | 1. Navigate to GitHub login page. 2. Leave both fields empty. 3. Click "Sign in". | Username: (empty)Password: (empty) | Error message: "Please fill in both fields." |
| TC005 | Login with valid credentials and "Remember me" checked | 1. Navigate to GitHub login page. 2. Enter valid credentials. 3. Check "Remember me". 4. Click "Sign in". | Username: valid\_userPassword: valid\_password | User stays logged in on subsequent visits. |