# Software Testing

## ***Presentation: 1.1 Introduction to Software Testing***

### *Overview*
This presentation gives the introduction of Software Testing, common software problems, best practices to be followed during testing, objectives of a tester and categories of defects, etc.

### *What is Software Testing?*
- Software Testing is a process of evaluating a system
  - by **manual** or **automatic** means
  - verify that the **system** satisfies the specified **requirements**
  - identify differences between **expected** and **actual** results
- Testing analyzes a program with the intent of finding problems and errors that measures system functionality and quality.
- Testing includes inspection and structured peer reviews of requirements and design, as well as execution test of code.

### *Need for Software Testing*
- It helps to deliver quality software products that satisfy users requirements, needs and expectations.
- If it is not done properly, it may cause mission failure and may have negative impact on operational performance and reliability.
- If done poorly, defects are found during operation which may lead to high maintenance cost and customer dissatisfaction.

### *Common Software Problems*
Some of the most common software problems which commonly appear in a wide variety of applications and environments are as follows:-

- Incorrect calculations
- Incorrect data edits
- Inadequate software performance
- Confusing or misleading data
- Software that is difficult to use
- Inconsistent processing
- Incorrect data and file handling
- Inadequate security controls
- Inability to handle production data capacities

- Incorrect calculations
    - This is seen whenever mathematical functions and operators are involved.

- Incorrect data edits
    - This happens when the software does not apply existing data edits correctly.
    - For example, a data edit may be coded to restrict the entry of the day of the month greater than “31”, but may allow the entry of invalid dates like Feb 30th.

- Inadequate software performance
    - This refers to slow system response times and transaction throughput rates which refers to the amount of transactions that can be processed in a given time period.

- Confusing or misleading data
    - This means that the data shown to users may be correct, but the users might not fully understand how to interpret the data.

- Software that is difficult to use
    - Sometimes a software may be complicated to use, difficult to navigate and may require several steps to perform simple tasks.

- Inconsistent processing
    - This refers to software that only works correctly in one environment and cannot be transported and used in another platform.

- Incorrect data and file handling
    - This refers to the software incorrectly retrieving data from files or tables.
    - This could include recovering the wrong data from the right source or right type of data from the wrong data source.

- Inadequate security controls
    - This means that unauthorized access to the system is not properly controlled and detected.
    - In addition, people may also be able to perform transactions above the authorization levels appropriate for their job functions.

- Inability to handle production data capacities
    - This refers to the software's incapability to process data at the level required by the organization.
    - An example for this is the case of the year 2000 computing problem, where dates in the year 2000 and beyond are incorrectly recognized as being in the early 1900's.

***Objective of Software Tester***

- The goal of a software tester is
    - to find bugs
    - find them as early as possible
    - Make sure that they get fixed.

***Best practices to be followed during Testing***

- Develop Master Test Plan so that resource and responsibilities are understood and assigned as early in the project as possible.
- Verify that all project deliverables and components are complete.
- Demonstrate and track true project progress.
- A risk prioritized list of test requirements and objectives are developed and maintained.
- Conduct reviews as early and as often as possible to provide developer feedback and get problems found and fixed as they occur.
- Testing and evaluation work is given to every member so as to generate team responsibility among all.
- Maintain standards for designing and developing major test ware components and procedures.

***Introduction to DEFECTS***

- A defect is defined as
    - A variance from the desired product quality.
    - Any situation where the system does not behave as indicated in the specification.
- A program P is considered accurate with respect to a specification, if and only if:
    - For each valid input, the output of P is in accordance with the specification S.

## *Error, Defect and Failure*

- **Error**
    - Mistake made by programmer.
    - Human action that results in the software containing a fault/defect
- **Defect/Fault**
    - Symptom of an error in a program
    - Condition that causes the system to fail
    - Synonymous with bug
- **Failure**
    - Incorrect program behavior due to fault in the program.
    - Failure can be determined only with respect to a set of requirement specification.

## *Categories of defects*

Defects generally fall into the following three categories:-

- Wrong
    - The specification has been implemented incorrectly.
    - This defect is a variance from customer/user specification.
- Missing
    - A specification or wanted requirement is not in the built product. This can be
        - a variance from specification,
        - an indication that the specification was not implemented
        - A requirement of the customer identified during or after the product was built.
- Extra
    - A requirement included into the product that was not specified.
    - This is always a variance from specification.

## *Key Points*

- **Software Testing** is a process of evaluating a system by identifying differences between expected and actual results.
- Software testing is important to deliver a quality software product that satisfies user’s requirements.
- If done poorly, defects are found during operation which may lead to high maintenance cost and customer dissatisfaction.
- The goal of a software tester is to find bugs, find them as early as possible, and make sure that they get fixed.
- Develop Master Test Plan so that resource and responsibilities are understood and assigned as early in the project as possible.
- Verify that all project deliverables and components are complete.
- Maintain standards for designing and developing major test ware components and procedures.

## ***<u>Presentation: 2.1 Introductions to Testing Techniques</u>***

### *Overview*

This presentation covers the different Technique s in Software Testing. It includes the classification of testing techniques and different types of testing under those classifications and it also describes the test case design techniques.

### *Testing Techniques*

Testing Techniques can be classified into two categories:

- Static testing
- Dynamic testing

Dynamic Testing is further classified into three categories:

- White Box testing
- Black Box testing
- Grey Box testing

#### Static Testing

- Static Testing is a testing technique which is performed without executing the application being tested.
- The errors in the application are found by checking the syntax or reading the code manually, hence it is also called **"Dry Run Testing"**
- It is an inspection of software artifacts such as requirement specification, design and code
- It is done by the following methods
  - Review
  - Inspection
  - walkthrough

#### Dynamic Testing

- Dynamic testing is testing technique which is performed by executing the application being tested
- Input is given and the output is checked by comparing the actual result with the expected result
- It is further divided into three categories
  - White box Testing
  - Black box Testing
  - Grey box Testing

## White Box Testing

- White box testing is a process of testing every statement in the code and ensures that all statements and conditions have been executed at least once.
- It examines the program structure, hence it is called as **structural testing**
- It is also referred as Glass box testing, Logic Driven testing, Path Oriented testing.
- This technique is typically applied at the Unit Test level, so it is performed by the Developers
- This Testing is done by the following testing methods:
    - Statement coverage
    - Path testing
    - Condition testing
    - Loop testing

## Black Box Testing

- Black box testing is a process of checking the functionality of the application against specified requirements
- It is used to find the following incorrect or missing
    - Implementations of functions
    - External database access
    - Performance errors
    - Initialization and termination errors
- It is also referred as Functional testing, Behavioral testing or Opaque testing
- This is done by the test engineers
- This Testing is done by the following testing methods
    - Functional Testing
    - Integration Testing
    - System Testing
    - Acceptance Testing

## Grey Box Testing

- A Greybox testing is a combination of Black box and White box testing
- It is not fully black box testing because the tester should have the knowledge of internal design of the code
- The application is also tested for its behavior and functionality which forms which is white box testing
- It is also called as “translucent testing”

## *Test Case Design Techniques*

- The design techniques for a test case are used to build test data
- These techniques determines whether combinations of inputs and operations produce expected results
- The following are the design techniques
  - Error Guessing
  - Equivalence Partitioning
  - Boundary Value Analysis

## Error Guessing

- Error Guessing is a testing technique which involves making a list of the errors expected to occur in a particular area of the system and then designing a set of test cases to check for these expected errors

- It is more an art than science but is very effective if a tester familiar with the history of the system

## Equivalence Partitioning

- Equivalence Partitioning involves partitioning the input values into number of equivalence classes

- An equivalence class is a subset of data that is representative of a larger class

## Boundary Value Analysis

- This technique is used to derive the test data from the boundaries of the input data
- These boundaries are known as input classes
- The boundaries are decided based on the following
    - the minimum and maximum boundary values
    - the maximum value plus or minus one
    - the minimum value plus or minus one

## *Key Points*

- Static testing is a process of checking for syntax and other errors by manually reading the code.
- Dynamic testing is the process of evaluating a system or component based upon its behaviour during execution.
- White box testing is the process of testing each and every statement of the product or application
- White box testing is done by the developers
- Black box testing is the process of executing test cases with the intention of finding the defects and check the functionality of the application against the requirement specifications
- Black box testing is done by the testers
- Grey box testing is a combination of white and black box testing

## ***<u>Presentation: 3.1 Testing Levels</u>***

### *Overview*

This presentation covers the different levels involved in Software Testing. It includes Functional testing, Integration testing, System testing and Acceptance testing.

### *Functional Testing*

- In functional testing the individual units or components are tested to verify if they function correctly.
- Functional Testing is also called as Component **testing.**
- Different types of input (test data) are given for testing the functionality of the component.

### *Integration Testing*

- Integration testing is a process of combining the modules and testing the data flow between the modules
- Integration testing verifies that the combined units function together properly
- This can facilitate finding problems that occur at interfaces or communication between the individual components
- It is divided into two types
    - Incremental Integration testing
    - Non-Incremental Integration testing

#### Incremental Integration Testing

- Incremental integration testing is a process of testing in which
    - each unit is integrated one at a time into the system
    - the data flow between the modules is tested as soon as a new module is integrated
- This process consists of systematically combining the unit tested modules one by one and testing the combined modules
- Incremental integration testing is divided into two types
    - Top-down testing
    - Bottom-up testing

## Top-down Testing

- Top-down testing is a process of incrementally adding the modules from parent to child and test the data flow between the modules
- This process works downward through the software hierarchy (top-to-bottom).
- It emphasizes on interface testing.

## Bottom-up Testing

- Bottom-up testing is a process of incrementally adding the modules from child to parent and test the data flow between the modules
- It emphasizes on module functionality

## Non-Incremental Integration Testing

- Non-Incremental integration testing is a process of testing the product by integrating every unit, all at a time into the system and testing the data flow between the modules
- This method can be used to save time in the Integration process
- It is also called as **Big bang method**

## *System Testing*

- System testing verifies whether the entire application is working according to the software requirements
- It is also referred to as Product testing and End-to-End testing
- System Testing involves some or all of the tests listed below
  - Compatibility testing - Testing functionality on different software/hardware environments
  - Usability testing – Testing the user-friendliness of the application
  - Performance testing – Testing the performance of the application in terms of response time etc.
  - Security testing – Testing the Authorization and Authentication levels of the application

## *Acceptance Testing*

- Acceptance testing is a process in which testing of the application is done by the end user or client
- The client tests whether the application meets the requirements given by him
- It is classified into two types
    - Alpha testing
    - Beta testing

## Alpha Testing

- Alpha testing is done by potential user / customer or an independent testing team at the developer's site along with the developers or testers
- It is done before beta testing

## Beta Testing

- Beta testing is done by potential user / customer or independent testing team at the customer's real time environment without any developer or tester involvement
- It is done before releasing the product to the market

## *Key Points*

- **Functional testing** is a process of testing each and every unit or component
- **Integration testing** is a process of testing the each unit in isolation, integrate the modules, then test the data flow between the modules.
- **Top-down and bottom up** testing are the types of Incremental integration testing.
- Non incremental integration testing is called as **Big bang method**.
- **System testing** is a process of testing the application as a whole against the requirement specification.
- **Acceptance testing** is a process of testing the application by the end user
- **Alpha and beta testing** are the types of Acceptance testing which are done by the customers

## ***Presentation: 4.1 Types of Testing***

### *Overview*

This presentation covers the different Techniques in Software Testing. It includes Smoke testing, Compatibility testing, Usability testing, Performance testing, Exploratory testing, Recovery testing, Installation testing, Regression testing.

### *Smoke Testing*

- Smoke testing is a test done before the application goes into the actual testing phase.
- The tester touches all areas of the application without getting in too deep
- This test reveals any simple failures and ensures that the system is ready for further testing
- For Example :
    - Smoke test is done to check whether the link or button does some action. If the answer is "No" then the test condition fails

### *Compatibility Testing*

- Compatibility testing is a process of testing the functionality of an application on different hardware and software environments
- The application is tested for the environments which are specified by the client
- For Example
    - Test the application on different operating systems such as Windows 2000 and XP, Macintosh, Linux and UNIX.
    - Test the web application on different browsers such as Internet explorer, Mozilla Firefox, Netscape navigator and Opera.

## *Usability Testing*

- Usability testing is done to test the user friendliness of the software
- The following areas are tested for usability
  - Time taken to accomplish a task
  - Accuracy of Data entered
  - Emotional response of the user (confident or stressed after using the application)
- Accessibility test is a type of Usability test which verifies that the application can be used by physically handicapped (deaf, blind, mentally retarded)

## *Performance Testing*

- Performance testing is a group of tests which test the following aspects
  - Response time
  - Execution time
  - Volume
  - Scalability

## *Types of Performance Testing*

- The different types of Performance tests are as follows
  - **Load testing** - The application/product is tested against fixed number of users in order to find out the stability and response time of the product
  - **Stress testing** – is done to evaluate if the application is stable at peak load i.e. beyond the limits of its specified requirements
  - **Volume testing** – is done to test the stability of the system by processing huge amount of data
  - **Soak testing** – is done by applying significant load over a extended period of time, to discover how the system behaves

## *Exploratory Testing*

- Exploratory testing is an approach of software testing where there is simultaneous exploring the application, preparing the test design and test execution
- When performing this test there are no exact expected results, the tester decides what will be verified
- It is done when requirements are incomplete or there is a lack of time
- Ad-hoc testing is a part of Exploratory Testing where the test is run once, unless a defect is discovered.

## *Security Testing*

- Security testing is a process of testing a application/product how well it is protected from unauthorized entries.
- It is used to find out all the potential loopholes and weaknesses of the system.
- Through this testing we can prevent the loss/theft of highly sensitive information from the outsider.
- It ensures that the system and application used by the organization are secure and not exposed to any type of attack.

## *Recovery Testing*

- Recovery testing is a process of testing the application in order to check how well the application recovers from any type of crash or hardware failure.
- It is tested for the system’s ability to recover at varying degrees of failure

## *Installation / Uninstallation Testing*

- Installation testing is defined as test which occurs outside the development environment
- This test is performed whenever a full, partial or upgrades of the application are installed/uninstalled
- It is also referred to as Implementation testing & Software deployment testing.

## *Regression Testing*

- Regression Testing is retesting the application after the bug fixes just to make sure that the bug has been fixed and its impact on the rest of the application
- It is divided into three types
  - Unit regression testing - Re-executing the same test cases on a particular unit or module due to requirement change
  - Regional regression testing - Re-executing the same test cases in two or three modules due to requirement change
  - Full regression testing - Re-executing the test cases in all modules due to a requirement change

## *Key Points*

- Smoke testing is a process of testing the application whether it is testable or not.
- Compatibility testing is a process of testing the functionality of an application on different hardware and software environments.
- Usability testing is a process of testing the user friendliness of the software.
- Performance testing is a process of testing the stability and response time of application by applying load.
- Exploratory testing is a type of testing where simultaneous exploration and testing is done
- Regression testing is a process of retesting the test cases after the bug fixes just to make sure that defect fixed or not
- Exploratory testing is a type of testing where understand the product and test the product.
- Ad-hoc testing is nothing but test the application randomly using commonsense or own ideas.
- Recovery testing is a process of testing the application in order to check how well it is recovering from any type of crash or hardware failure.

## ***Presentation: 5.1 Introductions to Software Testing Life Cycle (STLC)***

### *Overview*

This presentation involves the software testing life cycle, and the activities involved in each stage of the life cycle.

### *Software Testing Life Cycle (STLC)*

- The phase in which software testing starts till it ends is called software testing life cycle.
- In general software testing starts from the requirements analysis phase and ends with deployment phase.

### *Stages of STLC*

- Software testing life cycle contains the following stages
  - Requirement Analysis
  - Test plan Preparation
  - Test cases Preparation
  - Prepare RTM
  - Test execution
  - Defect tracking
  - Test execution report

### Requirements Analysis

- Going through the requirement documents to understand the expectations of customers is called Requirement analysis.
- In this process the tester becomes familiar with the requirements.
- The test scenarios are derived from the Requirements.

### Test Plan Preparation

- Test plan is a document which contains the following information
  - Objectives
  - Scope of Testing
  - Approach to be followed
  - Schedule
  - Outputs or deliverables
  - Risks and Contingency Plans
  - Estimates
  - Roles and Responsibilities

## Test Case Preparation

- A Test case is a set of test conditions which the tester uses to determine whether a requirement or Use Case has been satisfied in the application.
- They are derived from the Test Scenarios
- A Test case consists of
  - Header - test case name, project name, severity.
  - Body - description, input, expected result.
  - Footer - author, reviewer name.

## Prepare Requirement Traceability Matrix (RTM)

- Requirement Traceability Matrix (RTM) is a document which maps the test cases with the requirement.
- This document is used to
  - validate that the requirements are correct and complete
  - Check if all the requirements are addressed by some components of the application
  - Check if all the application components are addressing some requirements

**Illustration of RTM:**

| S.No | Requirements | Testcases |
|---|---|---|
| 1 | Login Page | Testcase001 |
| 2 | Home Page | Testcase002 |
| 3 | Inbox Page | Testcase003 |
| 4 | Compose Page | Testcase004 |

## Test Execution

- Test Execution is done by executing the test cases against the application / product or executing the test scripts by using an automation tool.
- Read the description in test steps which is given in test cases and apply the test data on the application.
- Once executed the test cases, will get the actual results.
- Compare the expected result with actual result then fill the status of the test case.

### Defect Tracking

- Defect Tracking is a process of finding defects in an application by inspection, testing or as feedback from customers.
- A defect arises when the expected result and actual results are not the same.
- The defects are documented in the System Investigation Request (SIR).
- The defects are fixed and the tester re-tests the application to ensure full resolution of the defects.

### Test Execution Report

- All executed test cases will be delivered to the client. This is called as test execution report.
- The test execution report includes information like tester's name, module name, build no, date of execution, actual result and status.
- Along with test cases, test plan, requirement traceability matrix and defect report will be delivered to the client.

## *Key Points*

- Software testing life cycle provides the standard procedure for testing processes.
- The test plan provides all the information about the processes.
- The Requirement traceability matrix is used to find out any missing requirements
- All the deliverables will be delivered in the test execution report.

## ***<u>Presentation: 6.1 Introductions to Test Plan</u>***

### *Overview*
This presentation gives the introduction of Test Plan, the testing environment, testing methodologies, the deliverables of the test team, etc.

### *Test Plan*
- Test Plan describes the intended scope, approach, resources, and schedule of the testing activities.
- The Test Lead prepares the test plan.
- The purpose of test plan is to manage, organize, and schedule the testing activities effectively.

### *Contents of Test Plan:*

1. Objective
2. Scope
3. Testing Environment
4. Deliverables
5. Roles and Responsibilities
6. Assumptions
7. Risks
8. Contingency Plan
9. Approach
10. Effort Estimation
11. Schedule
12. Testing Methodologies
13. Defect Tracking
14. Entry and Exit Criteria
15. Test Automation
16. Templates

## Objective & Scope

- The Objective explains the need and importance of test plan.
- The Scope talks about the features which need to be tested and the features which need not be tested.
- For Example
  - Features to be tested would be User required modules with clear priority levels defined (H, M, and L).
  - Features not to be tested can be features which are not part of the current release

## Approach

- Approach describes the strategy which is appropriate for a particular level.
- It can include information on tools to be used, metrics, software and hardware details, Regression Testing approach

## Testing Environment

- Testing Environment describes the resources required for the testing environment.
- The requirements for a Test environment can be special hardware requirements, specific versions of supporting software, specific range of test data used.

## Testing Methodologies

- Testing Methodologies provide the information about the testing methods to be used.
- These methodologies may vary from project to project

## Deliverables

- Deliverables are the outputs which are delivered at the end of each stage of testing
- Some of the deliverables are test plan, test cases, requirement traceability matrix, defect reports, test execution report, metrics.

## Roles and Responsibilities

- It describes the people and their tasks
    - Test manager
        - Write or review test plan.
        - Estimation, resource management and people management
    - Test Lead
        - Write or review test plan.
        - Monitor the work done by the Test engineers and make sure that all are computing their work within schedule.
        - Consolidate the report sent by the test engineers and communicate it to management/Customer.
    - Test Engineer: "XXX"
        - Review test plan
        - System study, write test scenarios, write test cases for "pqr" and "xyz" modules.
        - Prepare requirement traceability matrix.
        - Execute test cases of "pqr" and "xyz".
        - Communicate the defect report to development team and track the defects.
    - Test Engineer: "XYZ"
        - Set up the test environment and install the product.
        - Identify the scenarios for automation.
        - Automation of identified regression test cases.
        - Execution and maintenance of automated test cases.

## Assumptions

- Assumptions documents the prerequisites of the test, which if not met could have a negative impact on the test.
- All the planning is based on assumptions.
- For example test budget, availability of testing environments.

## Risks and Contingency Plan

- Risks in the Testing Process can be defined as follows
    - Lack of resources when training begins
    - Delay or Lack of Training on the application or tools used
    - Requirements getting changed
    - Application being delivered late
- Contingency plan is prepared as a part of analyzing risks and used whenever the risk occurs.

## Effort Estimation and Schedule

- **Effort Estimation** is estimation of time, resource, and cost for testing the entire project.
- **Schedule** is planning the start date and end date of all the testing activities

## Entry and Exit Criteria

- Entry and Exit Criteria are the inputs and outputs for the different stages of Testing.
- For example: In Functional testing
    - The entry criteria would be
        - Test environment should be ready.
        - White box testing should be completed
        - Test cases should be ready with test data.

    - Exit criteria would be
        - Percentage of test execution i.e. the number of test cases which are passed.
        - Number of defects sorted by severity levels.

## Defect Tracking

- **Defect Tracking** is done by using Defect Tracking tool or Test Management tool.
- The defect report is the primary document used to track and resolve defects discovered during testing
- Some of the attributes mentioned in the tool are severity, priority, status, etc., of the defect.
- These attributes can have the following values
    - Severity: critical/ major/ minor
    - Priority: high/ medium/ low
    - Status: open/ assigned / fixed / closed

## Test Automation

- Test automation is the part of the test plan which defines the tools to be used for Testing, Defect tracking and other testing activities in the test plan
- It also describes automation framework.

## Templates

- The **templates** are the structure or skeleton of the deliverables.
- It provides the standard format for all the deliverables.

## *Key Points*

- The test plan is a document which defines all the testing activities.
- Testing methodologies and automated tools to be used are mentioned in test plan.
- Roles and responsibilities of each member of a team are clearly mentioned in the test plan.
- Effort Estimation is estimation of time, resource and cost for testing the entire project.
- Defect tracking clearly describes about the status, severity and priority.

## ***Presentation: 7.1 Introductions to Test Cases***

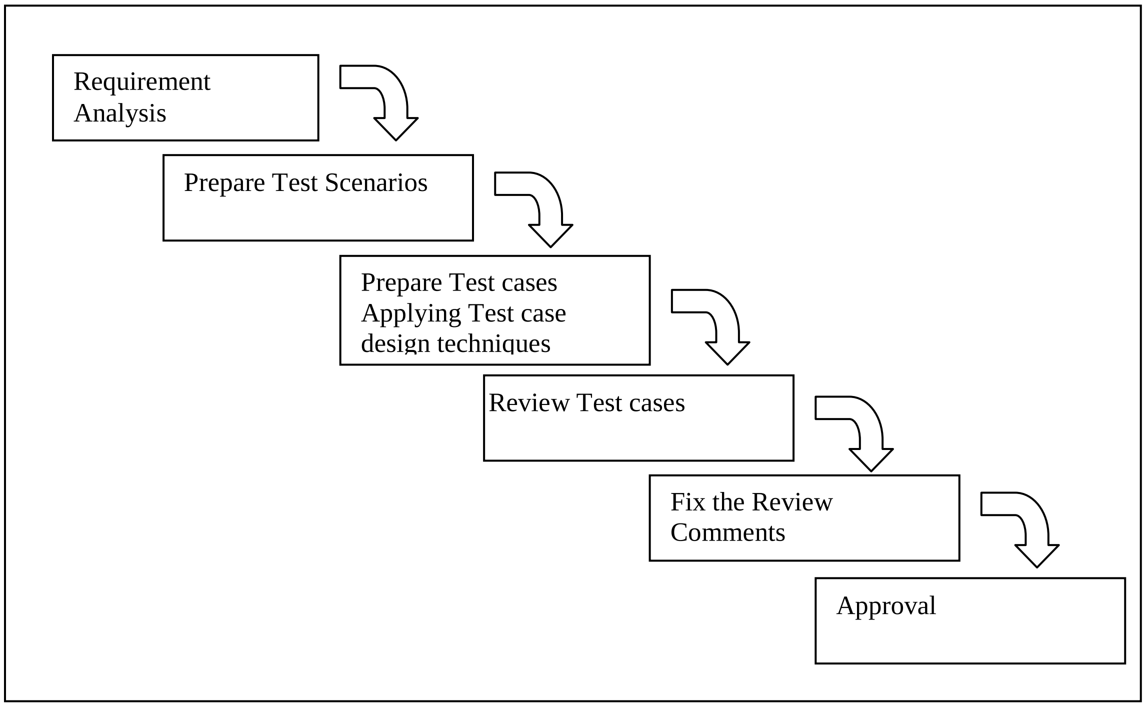
### *Overview*

This presentation gives the introduction of Test cases, review test cases and contents of test cases, etc.

### *What is test case?*

- **The test case is a combination of description, input and expected result developed for a particular intention to confirm the compliance of a requirement.**
    - Descriptions are step by step instructions or sequence of steps which contains conditions to achieve the perfect requirement.
    - Inputs are different possibility of test data.
    - Expected results are the outcome of the conditional descriptions based on the different inputs.
- The actual result and status can be obtained after executing the test cases.

### *Process of writing Test cases*



## Model of Test case Template

Test case id/name:
Requirement no:
Project name:
Module name:
Severity:
Pre-condition:
Test data:
Test case type:
Brief description:

| Step no | Description | Input | Expected result | Actual result | Status | Comments |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |

Author:
Reviewed by:
Last modified date:

## Contents of Test case

- **Test case id/ name:** Describes the name of the test case according to the company standard. E.g. cbo_finance_001.
- **Requirement no:** Describes for which requirement we prepared this test case. E.g. 100.1
- **Project name:** Describes the name of the project.
- E.g. Citibank Online.
- **Module name:** Describes the name of the module. E.g. Finance.
- **Severity:** Describes the test case impact of the business. E.g. critical/major/minor.
- **Pre-condition:** Describes the pre-conditions which is to be followed according to the requirement.
- **Test data:** Describes the test data to be used while executing the test case. E.g. username, password.
- **Test case type:** Describes the type of the test case. E.g. Functional/ Integration/ System test case.
- **Brief description:** Describes the explanation about the test case.

- **Author:** Describes the tester name who has written the test case.
- **Reviewed by:** Describes the tester name who has reviewed the test case.
- **Last modified date:** Describes the date of the last modification in the test case.

### Review Test cases

- Once the test case has been prepared by the tester then the review has to be done by one of his peer who could be a colleague. This type review is also called as Peer Review.
- The reviewer documents the review comments which can be any discrepancies in the test case.
- The tester fixes all the review comments and gets the test case approved by the test lead before execution of the test.
- The test lead plans the peer reviews.
- A review can be done by the test lead, developer or customer depending on the project plan

### Sample of a Test case Review Template

| Serial no. | Test case name | Step no. | Comments | Severity | Author |
|---|---|---|---|---|---|
| 1 | CBO_Loan_IR | 5 | Test data is wrong | Major | fixed |
| | | | | | |
| | | | | | |

### *Key Points*

- The **test case** is a combination of description, input and expected result developed for a particular intention to confirm the compliance of a requirement.
- The actual result and status can be obtained after executing the test cases.
- Test case design techniques should be followed when developing the test cases.

Test scenarios are the step by step instruction of a particular function.

## ***Presentation: 8.1 Introduction to Software Bug Life Cycle***

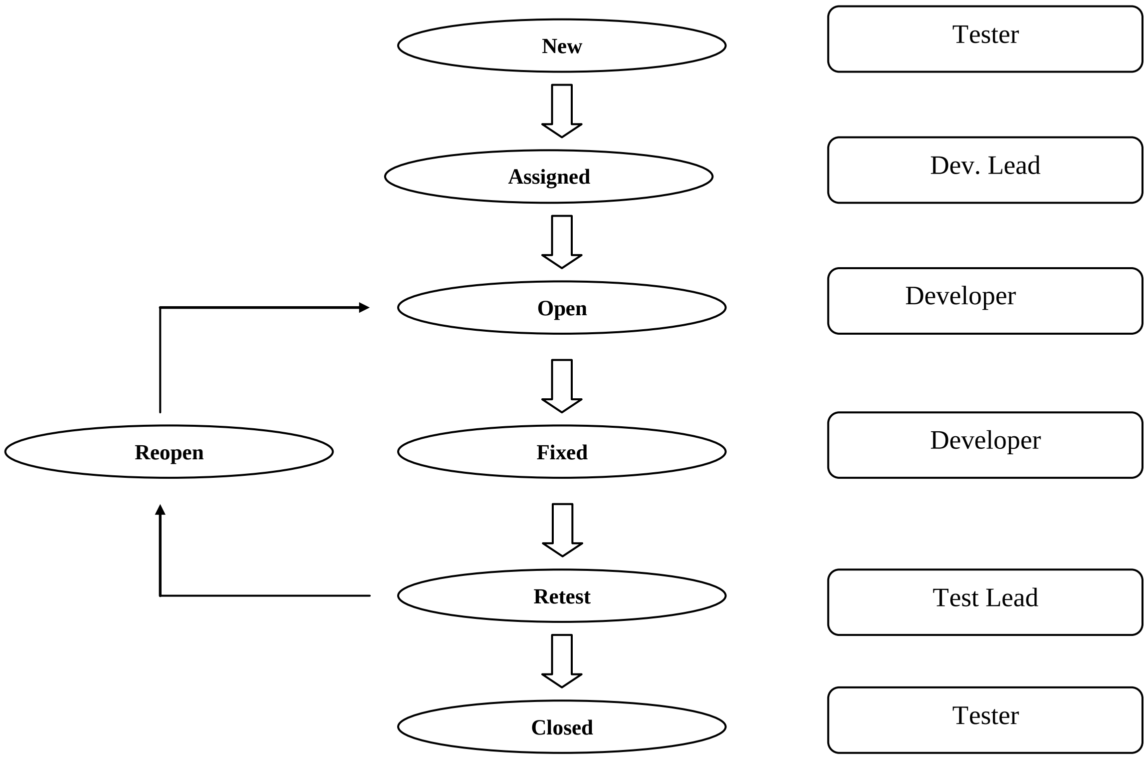
### *Overview*

This presentation gives the introduction of Software Bug Life Cycle, states of bug and defect report, etc.

### *Software Bug Life Cycle*

- **The duration between the detection of bug and closing the bug successfully is** called as software bug life cycle.
- Testers, test lead, developers and development lead play vital role in this life cycle

### Flow diagram of Software Bug Life Cycle



## States of Bugs

The different states of bug can be new, assigned, open, fixed, retest and closed.

- **New**
  - The bug is in the “New” state when the bug is detected the first time.
  - The tester logs the bug with the status as ‘New’ in the defect report
- **Assigned**
  - Here the bug is assigned to the developer to fix it.
  - The development lead logs the status as ‘Assigned’ in the defect report
- **Open:**
  - The developer changes the status as ‘Open’ when he starts fixing the bug.
- **Fixed:**
  - Once the developer fixes the bug, he changes the status as ‘Fixed’ which is reviewed by the development lead and it is forwarded to test lead.
- **Retest:**
  - The test lead changes the status as ‘Retest’ and sends it to tester to retest to check whether the bug is fixed.
- **Closed:**
  - The tester checks whether the defect is fixed or not. If yes then the status is changed to ‘Closed’.
- **Reopen:**
  - If the defect is not fixed, the tester changes the status to ‘Reopen’.
- **Rejected:**
  - The test lead reviews the bug and if the bug is not valid then the state is changed to ‘Rejected’.

## Defect Report Template

Defect Name/id:
Release name:
Project name:
Module name:
Status:
Severity:
Priority:
Test case id/name:
Brief description :
Detail description:
Following are the steps to reproduce the defects
1.
2.
3.
4.
5.

Expected result :
Actual result:

## Contents of Defect Report

- **Defect id/name:**
  - Describes the name of the defect according to the company standard.
  - The naming convention varies from company to company.
  - Eg: cbo_hr_def_001.
- **Release name:**
  - Describes the name of the Release.
  - Eg: Release_1, Release_2, Release_3
- **Project name:**
  - Describes the name of the project.
  - Eg: CitiBankOnline.
- **Module name:**
  - Describes the name of the module.
  - Eg: Sales.

- **Status:**
  - Describes the status of the bug.
  - Eg: New, Open, Fixed, Closed.
- **Severity:**
  - Describes the impact of the defect on the business.
  - Eg: critical/major/minor.
- **Priority:**
  - Describes the priority or urgency of fixing the defect.
  - Eg: High/Medium/Low.
- **Test case id/name:**
  - Describes the name of the test case according to the company standard.
  - Eg: cbo_finance_001.
- **Environment:**
  - Describes the environment of where the defect was found like operating system and browser name.
  - Eg: Win XP, IE 6.0
- **Test case type:**
  - Describes the type of the test case.
  - Eg: Functional/ Integration/ System test case.
- **Detailed description:**
  - Describes the steps on how the defect was detected.
- **Expected result:**
  - Describes the expected result based on client requirements.
- **Actual result:**
  - Describes the actual result after the test condition was executed.

## Severity

- **Severity indicates the Impact of the defect on the business.**
- It can be specified as
    - Blocker or Show stopper
    - Critical
    - Major
    - Minor
- Blocker or show stopper means login page not allowing the user to login and hence the user is not able to proceed to the next step.
- The test plan has a mention of the appropriate severity according to the situation.

## Priority

- **Priority is used to indicate the precedence of the defect to be fixed.**
- It is specified as
    - High / Medium / Low
    - P1 / P2 / P3 / P4
- The defect will be fixed based on the priority level according to the time basis.

## *Key Points*

- The duration between the bug detection and the bug being closed is called as **software bug life cycle**.
- The software bug life cycle contains the different states of bug such as new, assigned, open, fixed, retest and closed.
- **Severity** indicates the Impact of the defect on the business.
- **Priority** is used to indicate the precedence of the defect to be fixed.

## ***Presentation: 9.1 Verification vs. Validation***

### *Overview*

This presentation gives the introduction of Verification methods & its activities and also Validation methods & its activities, etc.

### *Verification Overview*

- It is the process of finding whether the product is developed correctly.
- Verification is a process of finding whether work product complies with the design specifications.
- The work product is assessed by one or more persons to check the correctness.

### **Verification Methods**

The following are some of the Verification methods:

- Inspection
- Walkthroughs
- Buddy Checks

The methods above are explained in the following slides

### **Inspection**

- The main goal of inspection is to find the defect and to convey the information about the product.
- It involves a team of 3-6 members
- It is headed by a unbiased moderator
- Each member will be viewing the product from different angle.
- It needs individual preparation by the members before going to the meeting.
- The presenter is not the author of the document
- Presenter reads the document, while others view it from different perspectives.
- This team finally comes out with a inspection report and the facts on errors.

## Walkthroughs

- In walkthrough the participants attend the meeting where the presentation about the product is given, and they give their comments, in form of reports.
- It gives an chance for the participants to familiarize with the product
- This team is led by the creator of the document who would also present the work product
- The team size usually is between 2-7 members.

## Testing Environment

- Testing Environment describes the resources required for the testing environment.
- The requirements for a Test environment can be special hardware requirements, specific versions of supporting software, specific range of test data used.

## Buddy checks

- Buddy check is an informal peer review
- It is performed by someone other than the creator
- One team member goes through the documents prepared by another member in order to find out if there are any bugs which the author couldn't find previously.
- This testing is done in situations where it is not suitable or impossible to conduct inspection or formal review.

## Verification Activities

Verification activities include verifying the following documents

- Software Requirement Specification (SRS)
- Functional Design
- Internal Design
- Code
- Test Plans

## *Validation Overview*

- This is the process of finding whether we are developing the right product
- This is the process to check whether the product meets the specified requirement mentioned in the SRS and the functional design.

## Strategies for Validation

- Requirement based tests
  - Related to Requirement document
  - Black box method is used
- Function based tests
  - Related to Functional design
  - Black box method is used
- Internal/Logic based tests
  - Checking the code
  - White box method is used

## Validation Methods

The following are some of the Validation methods:

- Equivalence partitioning
- Boundary-value analysis
- Error guessing
- Syntax testing

## Validation Activities

- Unit testing
- Integration testing
- Usability testing
- Function testing
- Acceptance testing
- etc.,.

## *Key Points*

- Verification is the process of finding whether the product is developed correctly.
- Verification is done by using methods like inspection, walkthroughs, buddy checks, etc.,.
- Validation is the process of finding whether we are developing the right product
- Verification activities include verifying the following documents
  - Software Requirement Specification (SRS)
  - Functional Design
  - Internal Design
  - Code
  - Test Plans
- Validation activities some of the following:
  - Unit testing
  - Integration testing
  - Usability testing etc..

## ***Presentation: 10.1Activities (Writing Test cases)***

### *Overview*

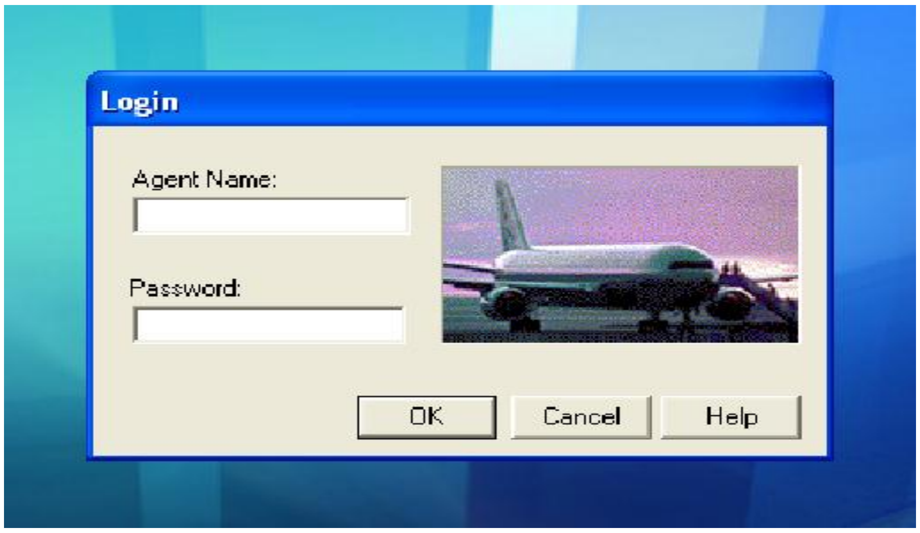
This presentation gives the introduction of writing functional, integration and system test cases.

### *Writing Test cases*

- We are using the Flight Reservation System for writing test cases.
- The test cases should be written in the excel sheet.
- The test case template should be in the same format which is specified in module 7.
- Use the following snap shots and write the test cases.

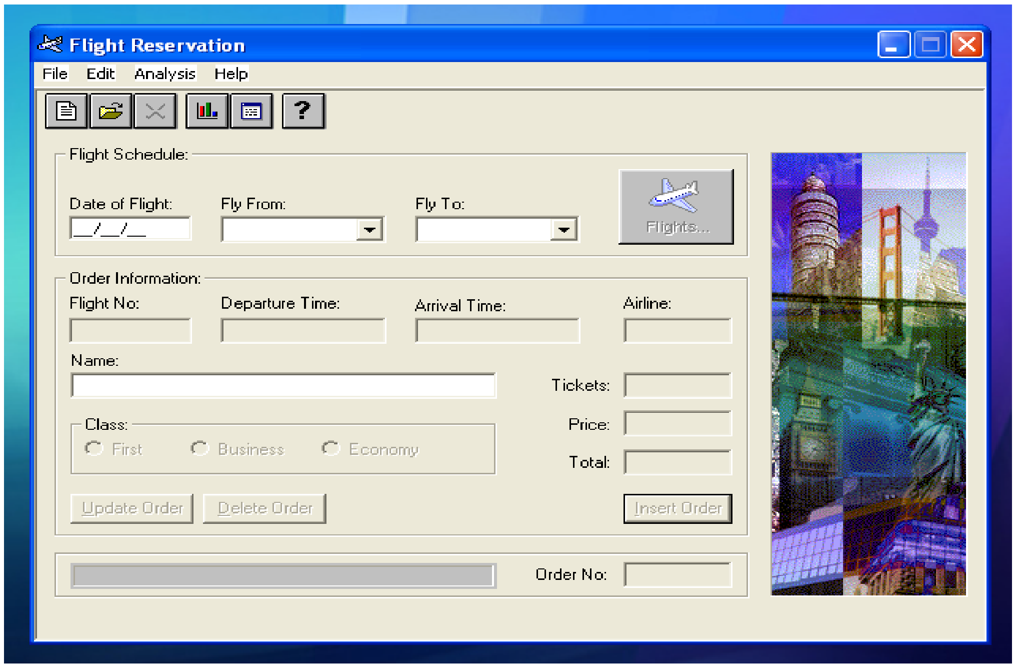
### Functional Test case

- Validate the Agent name and password using different possibilities of inputs.
- Validate the date of flight using multiple possibilities of test data.



Specifications:

- Agent Name:
    - Must be 4 - 30 characters long
    - Special characters not allowed
    - Start with alphabets
    - Alphanumeric characters allowed
- Password:
    - Should be mercury.
    - Should not be blank spaces
    - Special characters not allowed

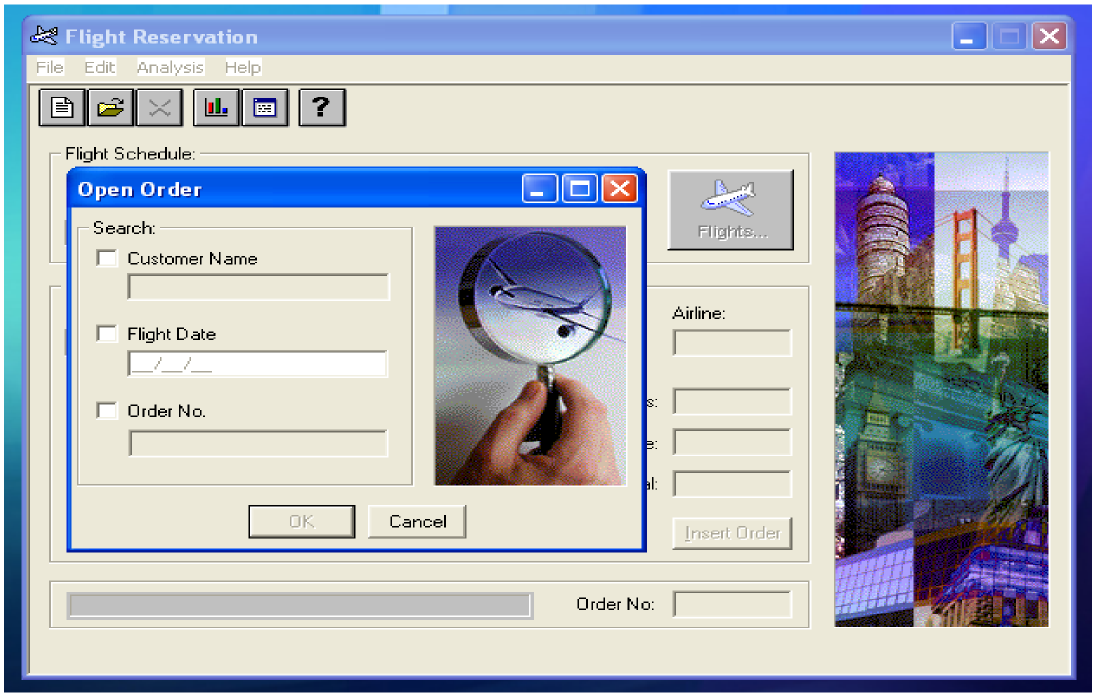


Specifications:

- Date of Flight:
    - Date should be greater than the system date
    - Date of the year should be less than 2038.
    - Check the date with different possibilities.

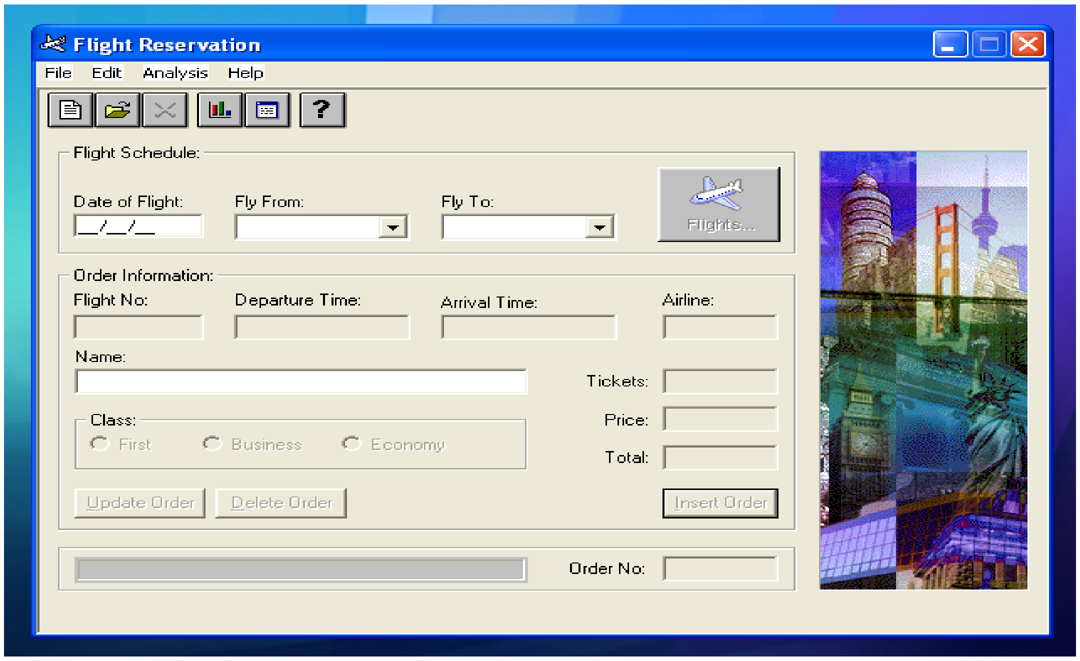
## Integration Test case

- Validate the order no in the Flight reservation page by giving the order no, in the open order page.
- Validate the customer name in the Flight reservation page by giving the customer name, in the open order page.



_Specifications:

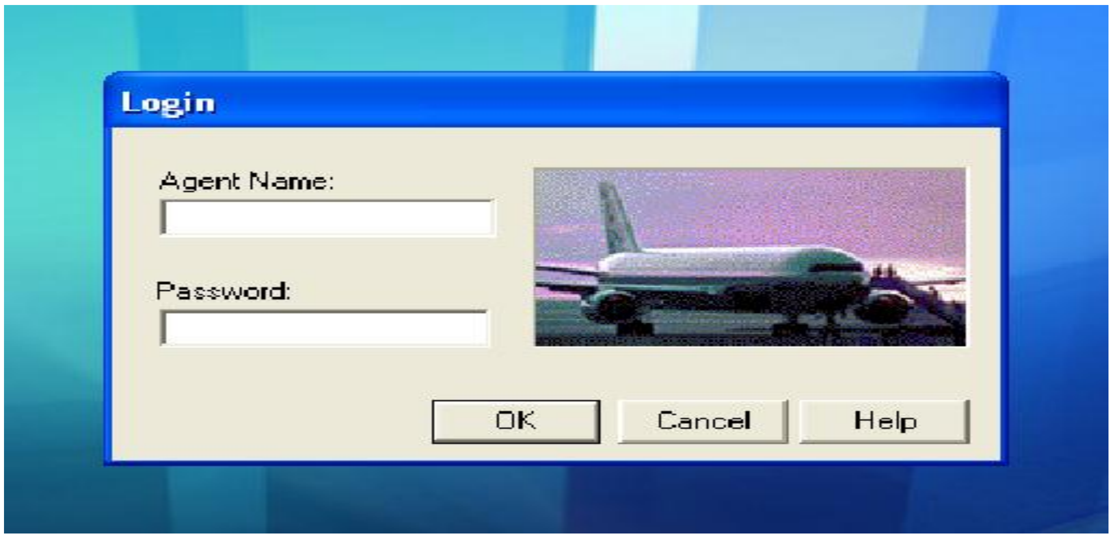
- Open order
- Enter order no / customer name
- Check whether you are getting the appropriate data.



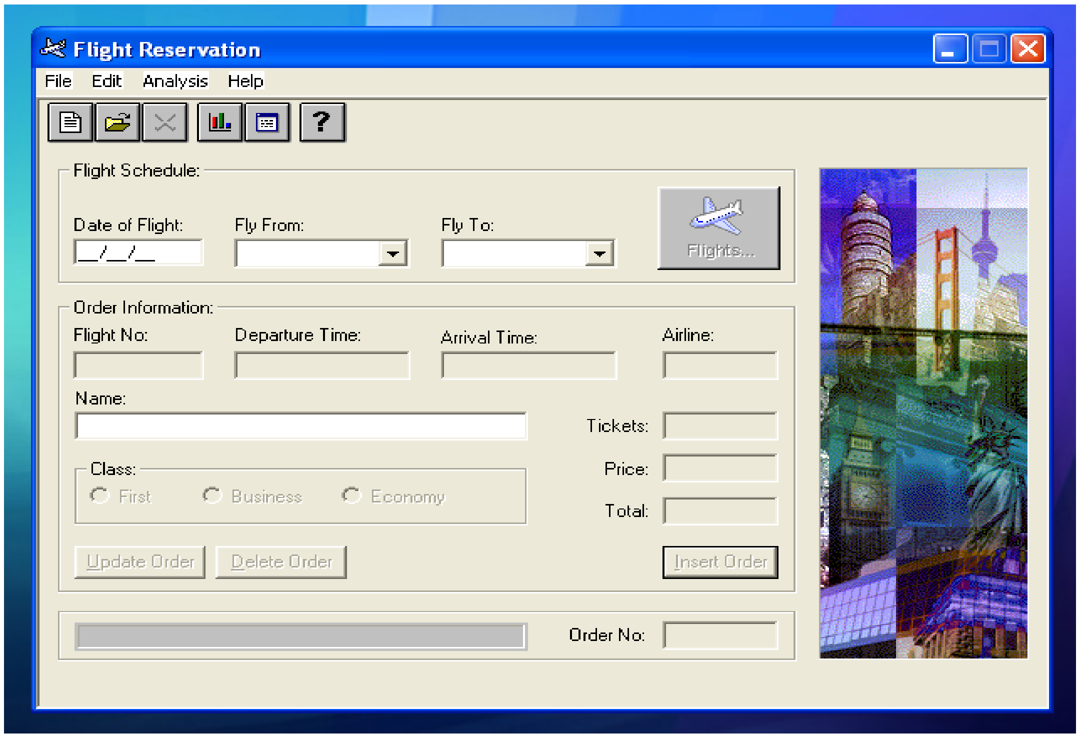
- Check the order no / customer name in the flight reservation page.
- Check whether the required order no / customer name is displayed.

## System Test case

- Validate the following requirement.
  - Enter user name and password.
  - Insert a new order
  - Check whether you are getting the order number once you have successfully inserted the order.



- Enter Valid Agent Name and Password.
- Click ok



- Enter Valid Date of Flight, Fly from, Fly to and select the Flight.
- Enter the Name of the passenger.
- Click Insert order.
- Check whether you are getting the order no.

## *Key Points*

- Test cases are specifically written for functional, integration and system testing.
- A functional test case contains many steps for verifying each component in detail.
- Integration test cases concentrates on data flow or interface design.
- System test cases concentrate whether on whether they are satisfying business requirements or not.