Flight Management System

Test plan and Test case

Version 1.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/5/2015 | 1.0 | Created the test plan and test case document | Siyuan Chen |
|  | 1.1 |  |  |
|  | 1.2 |  |  |
|  | 1.3 |  |  |
|  | 1.4 |  |  |
|  | 1.5 |  |  |
|  | 1.6 |  |  |

Table of Contents

Test Plan

# Introduction

## Purpose

This Test Plan document for the Flight Management System supports the following objectives:

• We will test the Flight Management System whether or not can successful work and those functional/nonfunctional requirements are covered by the software.

• The major test requirements are follow the main software components: Reservation System, Profile Subsystem, Service Subsystem, Reporting Subsystem

• The test strategies will based on the different user side: Administrator Side, Customers and travel agencies, Reservation Manager and Staff, Flight Manager, Services Manager. We will test the results of each functionality respectively. If any part won’t work successfully, the error information will send to the coding member and modify the program.

• Based on the above requirements, we hope to achieve all the functions perfectly.

## Background

There are four main components to the system which will be described below:

* Reservation System that manages all flight reservations, seat selection, ticketing, flight availability, flight details, rates and conditions.
* Profile Subsystem that manages individual passengers and travel agency profiles.
* Service Subsystem that manages in-flight services such as food and drinks.
* Reporting Subsystem to generate various summary reports such as Passenger Report, Cash Report and Booking Summaries.

Each user will use their unique account name and password to login the system. There will also demonstrate different interface to different type of user. And the database that has been created and connected should be successful used when call or input new data.

## Scope

The programmer should have the unit testing in the coding step to ensure the internal workings of each module can successful compile and work. When the whole software finished, we will use system testing for the function and performance.

In the whole testing, it includes all functional features based on three use case: System administrator, Project manager, Project member. And also few nonfunctional requirement such as speed and security part.

Assumptions: 1. If find any changing of interface could let user to feel more comfortable or convenient during testing, although the function can work well, the idea should be recorded and develop the software.

2. If the reaction speed of the software is too low and make user to feel impatient, although the function can work well, it should also be optimized.

Risk and contingencies : 1. The software could not finished when need to test.

2. Software might crash because of memory reasons when testing.

3. The functionality can not work successfully

## Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document  (and version / date)** | **Created or Available** | **Received or Reviewed** | **Author or Resource** | **Notes** |
| Requirements Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Functional Specification | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Use-Case Reports | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project Plan | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Design Specifications | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Prototype | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| User’s Manuals | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Business Model or Flow | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Data Model or Flow | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Business Functions and Rules | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |
| Project or Business Risk Assessment | 🞏 Yes 🞏 No | 🞏 Yes 🞏 No |  |  |

# Requirements for Test

The listing below identifies those items⎯use cases, functional requirements, and non-functional requirements⎯that have been identified as targets for testing. This list represents what will be tested.

- System administrator: 1. Login

2. Create staff profile

3. Edit staff profile

- Customers and travel agencies: 1. Login

2. Sign up

3. Make booking

4. Edit account details

5. Cancel booking

6. Edit service

7. Generate flight reports

8. Close account

- Reservation Manager: 1. Set ticket price

2. Move passengers between flights

3. Change passengers seating

4. Edit watch and no fly list

- Flight Manager: 1. Edit routes

2. Edit fleet

3. Edit flight schedule

4. Edit airports

- Services Manager: 1. Add service

2. Remove service

3. Edit service price

4. Edit service availability

- Profile Manager: 1. Edit watch and no fly list

2. Edit travel agency and customer profile

3. Close account

# Test Strategy

## Testing Types

### Function Testing

3.1.2.1 Profile Subsystem Testing

3.1.2.1.1 Administrator Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of System administrator |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. Create a staff profile  • Successfully create a staff profile  • Point out the error and doesn’t produce a new staff profile, when the user name is same with an existing one  4. Modify a staff profile  • Successfully modify a staff profile  • Point out the error and doesn’t change the profile, when the user name is same with an existing one  5. Remove a staff profile  • Successfully remove a staff profile  6. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.1.2 User Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of user |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. Create a user profile  • The user profile has been created successfully  • Point out the error and doesn’t produce a new staff profile, when the user name is same with an existing one  4. When login to the user side  • Includes the domain specific options: Make a Booking, Modify Services for Existing Booking, Cancel a Booking, Change Password, Modify Profile, Remove Profile, Request Statistical Report  5. Modify the details of profile  • Successfully modify a user profile  6. Change the password  • Successfully change the user password  • Point out the error and let user input a new password, when the new password is same with old one  7. Remove the profile  • Successfully remove the user profile  8. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.1.3 Profile manager Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of Profile manager |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. Remove the profile  • Successfully remove the user profile  4. Change the profile  • Successfully change the user profile  5. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.2 Reservation Subsystem

3.1.2.2.1 Reservation Manager and Staff Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of Reservation Manager and Staff |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. When login to the reservation manager and staff side  • Includes the domain specific options: Flight Transfer, Modify Customer Seat Selection  4.Transfer a customer from one flight to another  • Successfully transfer the flight  • Point out the error and the flight doesn’t be changed when the flight seats are not enough  5. Modify customer seat selection  • Successfully modify the seat selection  • Point out the error and the seat selection doesn’t be changed when the flight seats are not enough  6. Set ticket pricing of a flight  • Successfully set the price  7. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.2.2 Booking Process Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of Booking Process |
| Technique: | 1. The user selects the flight and book it  • Successfully book the flight  • Point out the error and the details don’t be recorded when the flight seats are not enough  • Point out the error and the details don’t be recorded when the Customer no fly status is set to “No Fly”  • Point out the error and the details don’t be recorded when the Customer passport holder status is set to false  2. The user views the booking summary  • Successfully display a booking summary to the user  3. The user selects in-flight services for a flight  • Successfully select services  4. Modify in-flight services for existing booking  • Successfully modify services  5. The user selects the flight and cancel it  • Display warning and cancellation fee, successfully cancel the flight  6. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.2.3 Flight Manager Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of Flight Manager |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. Modify details of routes  • Successfully modify details of routes: origin, destination, codeshare, number of stops  4. Modify records  • Successfully modify records: Airport Id Number,  Airport Name, City, Country, IATA Code, Latitude, Longitude, Altitude, Time Zone, DST, TZ Database Time Zone  5. Modify the airlines fleet of aircraft  • Successfully modify the airlines fleet of aircraft: Aircraft Name, Total in Service, First Class Seats Available, Business Class Seats Available, Premium Economy Seats Available, Economy Class Seats Available, Total Seats Available  6. Modify the flight schedule  • Successfully modify the airlines fleet of aircraft: Aircraft Name, Total in Service, First Class Seats Available, Business Class Seats Available, Premium Economy Seats Available, Economy Class Seats Available, Total Seats Available  7. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.3 Services Subsystem Testing

3.1.2.3.1 Services Manager Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of Services Manager |
| Technique: | 1. Login  • use correct user name and password, Successfully login  • use invalid user name and password, can’t login and displays an error message on the user interface  2. Logout  • Successfully logout  3. When login to the services manager side  • Includes the domain specific options: Modify Services  4. Modify the in-flight services available for purchase from the airline  • Successfully modify the in-flight services  5. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

3.1.2.4 Reporting Subsystem Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of user to request statistical report |
| Technique: | 1. The user requests the statistical report  • Successfully request statistical report  2. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

|  |  |
| --- | --- |
| Test Objective: | Ensure functionality of reporting system |
| Technique: | 1. The reporting manager login to the system  • Includes the domain specific options: Request Revenue Report, Request Popular Services Report, Request Popular Route Report  2. Display reports relating to (daily/monthly/annual) airline revenue  • Successfully request report  3. Display reports relating to (daily/monthly/annual) service popularity  • Successfully display the service popularity  4. Display reports relating to user (customer/travel agent) statistics and historical data  • Successfully display the statistics and historical data  5. Display reports relating to (daily/monthly/annual) route popularity  • Successfully display the route popularity  6. On occurrence of any exception during the action  • Cancel the action and doesn’t change any thing |
| Completion Criteria: | All planned tests have been executed.  All identified defects have been addressed. |
| Special Considerations: |  |

### Performance Profiling

|  |  |
| --- | --- |
| Test Objective: | [Verify performance behaviors for designated transactions or business functions under the following conditions:  • normal anticipated workload  • anticipated worst case workload] |
| Technique: | • [Use Test Procedures developed for Function or Business Cycle Testing.  • Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs.  • Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).] |
| Completion Criteria: | • [Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction.]  • [Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.] |
| Special Considerations: | [Comprehensive performance testing includes having a background workload on the server.  There are several methods that can be used to perform this, including:  • “Drive transactions” directly to the server, usually in the form of Structured Query Language (SQL) calls.  • Create “virtual” user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with “traffic”.  • Use multiple physical clients, each running test scripts to place a load on the system.  Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  The databases used for Performance Testing should be either actual size or scaled equally.] |

### Load Testing

[Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues).]

[Note: Transactions below refer to “logical business transactions”. These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

|  |  |
| --- | --- |
| Test Objective: | [Verify performance behavior time for designated transactions or business cases under varying workload conditions.] |
| Technique: | • [Use tests developed for Function or Business Cycle Testing.  • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.] |
| Completion Criteria: | [Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.] |
| Special Considerations: | • [Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  • The databases used for load testing should be either actual size or scaled equally.] |

### Stress Testing

[Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle.]

[Note: References to transactions below refer to logical business transactions.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test functions properly and without error under the following stress conditions:  • little or no memory available on the server (RAM and DASD)  • maximum actual or physically capable number of clients connected or simulated  • multiple users performing the same transactions against the same data or accounts  • worst case transaction volume or mix (see Performance Testing above).  Notes: The goal of Stress Testing might also be stated as identify and document the conditions under which the system FAILS to continue functioning properly.  Stress Testing of the client is described under section 3.1.11, Configuration Testing.] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • To test limited resources, tests should be run on a single machine, and RAM and DASD on server should be reduced or limited.  • For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume or mix. |
| Completion Criteria: | [All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions.] |
| Special Considerations: | • [Stressing the network may require network tools to load the network with messages or packets.  • The DASD used for the system should temporarily be reduced to restrict the available space for the database to grow.  • Synchronization of the simultaneous clients accessing of the same records or data accounts.] |

### Volume Testing

[Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test successfully functions under the following high volume scenarios:  • Maximum (actual or physically- capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period.  • Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.] |
| Technique: | • [Use tests developed for Performance Profiling or Load Testing.  • Multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume or mix (see Stress Testing above) for an extended period.  • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries and report transactions simultaneously for extended periods.] |
| Completion Criteria: | • [All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.] |
| Special Considerations: | [What period of time would be considered an acceptable time for high volume conditions, as noted above?] |

### Security and Access Control Testing

[Security and Access Control Testing focus on two key areas of security:

• Application-level security, including access to the Data or Business Functions

• System-level Security, including logging into or remote access to the system.

Application-level security ensures that, based upon the desired security, actors are restricted to specific functions or use cases, or are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that” user type one” can see all customer information, including financial data, however,” user two” only sees the demographic data for the same client.

System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.]

|  |  |
| --- | --- |
| Test Objective: | * Application-level Security: [*Verify that an actor can access only those functions or data for which their user type is provided permissions.]* * System-level Security: *Verify that only those actors with access to the system and applications are permitted to access them*.] |
| Technique: | * Application-level Security: [*Identify and list each user type and the functions or data each type has permissions for.]*   • [Create tests for each user type and verify each permission by creating transactions specific to each user type.]  • Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied.   * System-level Access: *[See Special Considerations below]* |
| Completion Criteria: | [For each known actor type the appropriate function or data are available, and all transactions function as expected and run in prior Application Function tests.] |
| Special Considerations: | [Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.] |

### Failover and Recovery Testing

[Failover and RecoveryTesting ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that recovery processes (manual or automated) properly restore the database, applications, and system to a desired, known, state. The following types of conditions are to be included in the testing:  • power interruption to the client  • power interruption to the server  • communication interruption via network servers  • interruption, communication, or power loss to DASD and or DASD controllers  • incomplete cycles (data filter processes interrupted, data synchronization processes interrupted).  • invalid database pointer or keys  • invalid or corrupted data element in database] |
| Technique: | [Tests created for Function and Business Cycle testing should be used to create a series of transactions. Once the desired starting test point is reached, the following actions should be performed, or simulated, individually:  • Power interruption to the client: power the PC down.  • Power interruption to the server: simulate or initiate power down procedures for the server.  • Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers.  • Interruption, communication, or power loss to DASD and DASD controllers: simulate or physically eliminate communication with one or more DASD controllers or devices.  Once the above conditions or simulated conditions are achieved, additional transactions should be executed and upon reaching this second test point state, recovery procedures should be invoked.  Testing for incomplete cycles utilizes the same technique as described above except that the database processes themselves should be aborted or prematurely terminated.  Testing for the following conditions requires that a known database state be achieved. Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed.] |
| Completion Criteria: | [In all cases above, the application, database, and system should, upon completion of recovery procedures, return to a known, desirable state. This state includes data corruption limited to the known corrupted fields, pointers or keys, and reports indicating the processes or transactions that were not completed due to interruptions.] |
| Special Considerations: | • [Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required.  • Resources from the Systems (or Computer Operations), Database, and Networking groups are required.  • These tests should be run after hours or on an isolated machine.] |

### Configuration Testing

[Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded⎯for example, applications, drivers, etc.⎯and at any one time, many different combinations may be active using different resources.]

|  |  |
| --- | --- |
| Test Objective: | [Verify that the target-of-test functions properly on the required hardware and software configurations.] |
| Technique: | • [Use Function Test scripts.  • Open and close various non-target-of-test related software, such as the Microsoft applications, Excel and Word, either as part of the test or prior to the start of the test.  • Execute selected transactions to simulate actor’s interacting with the target-of-test and the non-target-of-test software.  • Repeat the above process, minimizing the available conventional memory on the client workstation.] |
| Completion Criteria: | [For each combination of the target-of-test and non-target-of-test software, all transactions are successfully completed without failure.] |
| Special Considerations: | • [What non-target-of-test software is needed, is available, and is accessible on the desktop?  • What applications are typically used?  • What data are the applications running; for example, a large spreadsheet opened in Excel or a 100- page document in Word?  • The entire systems, netware, network servers, databases, etc. should also be documented as part of this test.] |

### Installation Testing

[Installation testing has two purposes. The first is to insure that the software can be installed under different conditions⎯such as a new installation, an upgrade, and a complete or custom installation⎯under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, etc. The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing.]

|  |  |
| --- | --- |
| Test Objective: | Verify that the target-of-test properly installs onto each required hardware configuration under the following conditions:  • new installation, a new machine, never installed previously with <Project Name>  • update, machine previously installed <Project Name>, same version  • update, machine previously installed <Project Name>, older version |
| Technique: | • [Manually or develop automated scripts, to validate the condition of the target machine⎯ new - <Project Name> never installed; <Project Name> same version or older version already installed).  • Launch or perform installation.  • Using a predetermined sub-set of function test scripts, run the transactions.] |
| Completion Criteria: | <Project Name> transactions execute successfully without failure. |
| Special Considerations: | [What <Project Name> transactions should be selected to comprise a confidence test that <Project Name> application has been successfully installed and no major software components are missing?] |

## Tools

The following tools will be employed for this project:

[Note: Delete or add items as appropriate.]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tool | Vendor/In-house | Version |
| Test Management |  |  |  |
| Defect Tracking |  |  |  |
| ASQ Tool for functional testing |  |  |  |
| ASQ Tool for performance testing |  |  |  |
| Test Coverage Monitor or Profiler |  |  |  |
| Project Management |  |  |  |
| DBMS tools |  |  |  |

# Resources

## Workers

This table shows the staffing assumptions for the project.

|  |  |  |
| --- | --- | --- |
| Human Resources | | |
| Worker | Minimum Resources Recommended  (number of full-time workers allocated) | Specific Responsibilities or Comments |
| Test Manager,  Test Project Manager |  | Provides management oversight.  Responsibilities:   * provide technical direction * acquire appropriate resources * provide management reporting |
| Test Designer |  | Identifies, prioritizes, and implements test cases.  Responsibilities:   * generate test plan * generate test model * evaluate effectiveness of test effort |
| Tester |  | Executes the tests.  Responsibilities:   * execute tests * log results * recover from errors * document change requests |
| Test System Administrator |  | Ensures test environment and assets are managed and maintained.  Responsibilities:   * administer test management system * install and manage worker access to test systems |
| Database Administratator, Database Manager |  | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities:   * administer test data (database) |
| Designer |  | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities:   * identifies and defines the test class(es) * identifies and defines the test packages |
| Implementer |  | Implements and unit tests the test classes and test packages.  Responsibilities:   * creates the test classes and packages implemented in the test model |

## System

The following table sets forth the system resources for the testing project.

[The specific elements of the test system are not fully known at this time. It is recommended that the system simulate the production environment, scaling down the accesses and database sizes if and where appropriate.]

[Note: Delete or add items as appropriate.]

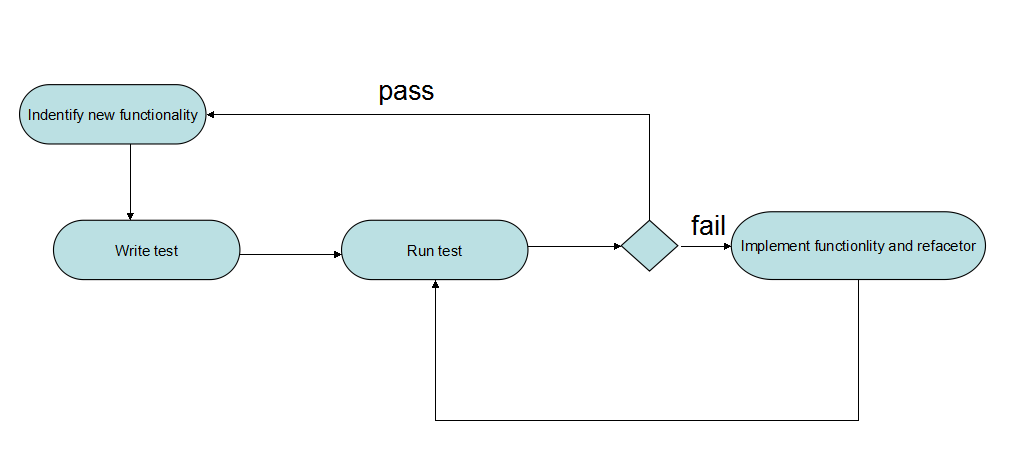
|  |  |
| --- | --- |
| System Resources | |
| Resource | Name / Type |
| Database Server |  |
| —Network or Subnet | TBD |
| —Server Name | TBD |
| —Database Name | TBD |
| Client Test PC's |  |
| —Include special configuration requirements | TBD |
| Test Repository |  |
| —Network or Subnet | TBD |
| —Server Name | TBD |
| Test Development PC's | TBD |

# Project Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Effort** | **Start Date** | **End Date** |
| Plan Test |  |  |  |
| Design Test |  |  |  |
| Implement Test |  |  |  |
| Execute Test |  |  |  |
| Evaluate Test |  |  |  |

# Deliverables

## Test Model



## Test Logs

[Describe the method and tools used to record and report on the test results and testing status.]

## Defect Reports

[In this section identify the method and tools used to record, track, and report on test incidents and their status.]

# Appendix A: Project Tasks

Below are the test related tasks:

• Plan Test

* + identify requirements for test
  + assess risk
  + develop test strategy
  + identify test resources
  + create schedule
  + generate Test Plan

• Design Test

- prepare workload analysis

- identify and describe test cases

- identify and structure test procedures

- review and assess test coverage

• Implement Test

* + record or program test scripts
  + identify test-specific functionality in the Design and Implementation Model
  + establish external data sets

• Execute Test

- execute Test procedures

- evaluate execution of Test

- recover from halted Test

- verify the results

- investigate unexpected results

- log defects

• Evaluate Test

- evaluate Test-case coverage

- evaluate code coverage

- analyze defects

- determine if Test Completion Criteria and Success Criteria have been achieved