****

Test Report

MEETING PLANNER

*Mar, 2017*

Table of Contents

[Introduction 3](#_Toc477637640)

[Test Scope 3](#_Toc477637641)

[Test Plan Details 3](#_Toc477637642)

[Test Features 4](#_Toc477637643)

[Test Strategy 5](#_Toc477637644)

**Introduction**

This document is used to define the details of the test plan of Meeting Planner System. Meeting Planner aims to facilitate its user an ability to record & maintain meetings or vacation.

**Test Scope**

This section is used to provide a high-level description of the business processes to be tested and a high-level description of the type of testing required for this project. This document also focuses on specific assumption which was recorded in Business Requirement Document.

*This test plan will test all the requirements for Meeting Planning System. This Master Test Plan covers unit, integration, user acceptance, and regression testing.*

*Broadly We will test below classes -*

1. *Calendar – Individual Calendar Function,*
2. *Meeting – Setting Meetings & Gaining Information on boundary value of inputs.*
3. *Organization – Integration of Other Classes & their functionality*
4. *Person – Collection of Users*
5. *Room – Room Controller*
6. *TimeConflict Exception – Special Exception Controller*

*Specific Assumptions:*

1. *This System only capture information for meeting in current year. Hence it does not deal with Year information.*
2. *Current Year is assumed to be Non leap year which should restrict entry for 29 Feb.*

**Test Plan Details**

This is a description of the features that will be tested within the scope of the test plan.   
The list represents what will be tested as well as the individuals/groups involved with the testing activities.

|  |  |  |
| --- | --- | --- |
| ***Module#*** | ***Feature*** | ***Test Case Concentrations*** |
| *1* | Method Level | *Check all methods of Classes at method level only.* |
| *2* | Meeting | *Check invalid inputs, Malfunctioning date, Valid Input, Overlapping Time range, Boundary value testing. Remove Meetings.* |
| *3* | Vacation | *Check invalid inputs, Malfunctioning date, Valid Input, Overlapping Time range, Boundary value testing.* |
| *4* | Room | *Invalid Room ID, Blank Room Id, Conflicts of Room. Show availability wide Rooms.* |
| *5* | Person | *Invalid Names, Empty Inputs, conflict of Schedule, Show availability wide Person.* |
| *6* | Planner Interface | *Invalid Inputs, Frequent Inputs.* |
| *7* | Combined Classes | *Dependency of two type of class over functionality like Check adding Meeting post Vacation booking etc.* |

Importantly, we will test single classes & their behavior. Later we will combine few features together and analyze their output in same class. We will also carefully analyze how system fails.

Further, we will pick up set of dependent classes and carefully track their output by combining scenario which needs those dependent classes.

**Test Features**

This is a description of the features that will be tested within the scope of the test plan.   
The list represents what will be tested as well as the individuals/groups involved with the testing activities.

|  |  |  |
| --- | --- | --- |
| BR # | Feature | Responsible party or groups |
| 1 | Booking a meeting | IT |
| 2 | Booking Vacation Time | IT |
| 3 | Check room availability | IT |
| 4 | Check person availability | IT |
| 5 | Scheduler Person | IT |
| 6 | Scheduler Room | IT |
|  |  |  |

**Test Inputs/Outputs**

This is a description of the inputs that will be used to assist with the testing effort and a list of the outputs that will be delivered from the testing effort.

|  |  |
| --- | --- |
| ***Inputs*** | ***Outputs*** |
| *Code* | *Test Results* |
| *BRD – SRS* | *Defect Log* |
| *Use Case Documents* | *Issues Log* |
|  | *Change Request Log* |

**Test Strategy**

This is a description of the recommended testing approach for this project. This describes “how” the test items and features will be tested.

*Example:*

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Comment** |
| Unit Test | *This type tests individual blocks of code and provide error free code structurally.* |  |
| Integration Test | *This type tests all changed functionality end to end.* |  |
| User Acceptance Test | *Typically performed by Business User, this tests all changed functionality of a system from the end-user perspective.* |  |
| Regression Test | *This tests the entire system to insure that any untouched functionality did not get adversely impacted by the project-related work.* |  |

**Test Case Measurement**

Assigning test result to measurable results so that we can evaluate severity & priority of errors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Severity** | |  | **Priority** | |
|  | 1 | Urgent |  | 1 | Critical |
|  | 2 | High |  | 2 | High |
|  | 3 | Medium |  | 3 | Moderate |
|  | 4 | Low |  | 4 | Low |

|  |  |
| --- | --- |
| Severity – Classification of defect (bug) to indicate the degree of negative impact on quality or the degree of impact that a defect has on the development or operation of a component or system. | |
| Urgent | The defect affects critical functionality or critical data. It does not have a workaround. Example: Unsuccessful installation, complete failure of a feature |
| High | The defect affects major functionality or major data. It has a workaround but is not obvious and is difficult. Example: A feature is not functional from one module but the task is doable if 10 complicated indirect steps are followed in another module(s) |
| Medium | The defect affects minor functionality or non-critical data. It has an easy workaround. Example: A minor feature that is not functional in one module but the same task is easily doable from another module |
| Low | The defect does not affect functionality or data. It does not even need a workaround. It does not affect productivity or efficiency. It is merely an inconvenience. Example: Petty layout discrepancies, spelling/grammatical errors |
| Priority – The level of business importance assigned to a defect. | |
| Critical | This needs to be fixed right now; everything else can wait; the build cannot be released with the defect |
| High | Should be fixed as early as possible |
| Moderate | May be fixed after the release / in the next release |
| Low | Fixing can be deferred until all other priority defects are fixed |

**Document Approval**

The signatures below acknowledge that the test plan outlined above is complete and accurate. Upon receiving written approval, the project team will proceed to the next step of the project.

If anything changes during the execution of the project, the test plan will be updated and re-approved accordingly.

|  |  |  |
| --- | --- | --- |
| **Approved by:**  **Printed name** | **Approved by:**  **Title** | **Approved By:**  **Signature/Date** |
|  | Business Representative |  |
|  | IT Representative |  |