BSSE Software Testing

Meeting Planner Test Plan

Document Information

- **Title:** Meeting Planner Test Plan
- **Version:** 1.0
- **Date:** 28th March, 2025
- **Project:** BSE3211 - Unit Testing Exercise

Names	Registration Number
NANKYA SHADIA	22/U/6646
EFRATA ARON	22/X/5249/PS
AHAABWE DERRICK	22/U/5665
ABINSINGUZA LASSE	22/U/5018/PS
NTULUME WILSON	22 /U/6739

Table of Content

1. Introduction

This document outlines the formal test plan for the Meeting Planner application. It defines the scope, responsibilities, testing strategies, test phases, test cases, risk mitigation, and acceptance criteria to ensure successful validation of the application's features and functionality.

2. Testing Team & Responsibilities

Class	Tester	Key Responsibilities
Person	Ahaabwe Derrick	Person entity management, personal agenda generation, vacation booking
Meeting	Effie	Meeting scheduling, conflict detection, attendee management
Calendar	Lasse	Meeting addition, agenda generation, availability checks
Room	Shadia Nankya	Room availability, booking management, room-specific agendas
Organization	Wilson	System-wide user and room management, search functionality

3. Test Approach

- **Unit Testing:** Each developer tests individual classes and methods they are responsible for.
- **Integration Testing:** Tests interactions between interconnected modules (e.g., Person ↔ Meeting, Room ↔ Calendar).
- **System Testing:** Full end-to-end workflow validation.
- **Regression Testing:** Re-run test cases after defect fixes to ensure consistency.

4. Test Timeline & Deliverables

Phase	Activity	Start Date	End Date	Deliverables
1	Test Planning	Mar 26	Mar 27	Test Plan Document

2	Environment Setup	Mar 31	Apr 01	Configured Development & Test Environment
3	Test Case Design & Execution	Apr 02	Apr 07	Test Cases, Execution Logs, Defect Reports
4	Regression & Final Testing	Apr 08	Apr 10	Final Test Summary Report

5. Test Cases

Phase 3: Test Case Design & Execution (Apr 02 – Apr 07)

Person Class – Ahaabwe Derrick

- testDefaultConstructor() – Validate default constructor initializes the name to an empty string.
- testConstructorWithName() – Check constructor with name argument to ensure proper initialization.
- testGetName() – Validate the getName() method returns the correct name.
- testAddMeeting() – Validate the addition of a meeting to the person's agenda and check if they are marked as busy during the meeting.
- testAddMeetingConflict() – Ensure a conflict exception is thrown when adding overlapping meetings to the same person.
- testPrintAgendaMonth() – Verify the monthly agenda formatting contains the correct meeting details.
- testPrintAgendaDay() – Check the daily agenda output for formatting and meeting details.
- testIsBusy() – Validate that the isBusy() method correctly identifies if a person is busy during a meeting time.
- testGetMeeting() – Ensure retrieving a meeting by date correctly returns the meeting object.
- testRemoveMeeting() – Check that a meeting can be removed and the person's busy status is updated accordingly.
- testMultipleMeetings() – Verify multiple non-overlapping meetings are correctly handled and displayed on the agenda.

- testEmptyAgenda() – Ensure the agenda for a day with no meetings returns an empty string or a relevant message.
- testAddValidMeeting() – Check adding a valid meeting and retrieval by date.
- testAddMeetingConflictDiagnostic() – Ensure proper diagnostics and exception handling when adding conflicting meetings.
- testAddTwoNonConflictingMeetings() – Verify adding two non-overlapping meetings works correctly without any exceptions.
- testAddMeetingWithNoRoom() – Test adding a meeting without a room and validate the meeting details.
- testAddMeetingWithEmptyAttendees() – Ensure a meeting can be added with empty attendees and validate the agenda.
- testAddBackToBackMeetings() – Check for conflicts when trying to add back-to-back meetings, and ensure proper handling of the conflict.
- testAddMeetingOnDifferentDay() – Ensure meetings on different days do not conflict and are displayed on the correct day's agenda.
- testPrintAgendaValid() – Check the generation of a valid agenda for a specific day and verify all meeting details are present.
- testPrintAgendaNoMeetings() – Ensure agenda displays correctly when there are no meetings scheduled for a day.
- testPrintAgendaMonthWithMultipleDays() – Verify the agenda correctly includes multiple meetings on different days within a month.
- testPrintAgendaFormat() – Ensure the agenda formatting matches the expected structure for meeting details.

Meeting Class – Efrata

- testMeetingCreationValidParams() – Create a meeting with valid data
- testTimeConflictDetection() – Detect overlapping meetings
- testInvalidTimeValidation() – Handle invalid time inputs
- testDefaultConstructor() – Validate default constructor values
- testMonthDayConstructor() – Validate constructor with only month and day

- `testMonthDayDescriptionConstructor()` – Constructor with description formatting
- `testFullConstructor()` – Validate constructor with all parameters
- `testAddAttendee()` – Add an attendee to a meeting
- `testRemoveAttendee()` – Remove an attendee from a meeting
- `testMultipleAttendees()` – Manage multiple attendees
- `testValidTimeRange()` – Ensure valid time range is set
- `testInvalidTimeRange()` – Handle reversed time range (e.g. start > end)
- `testInvalidDate()` – Handle invalid day values (e.g. Feb 35)
- `testToStringWithNullValues()` – Check string output with null room/description
- `testTimeBoundaries()` – Verify earliest and latest valid times
- `testMeetingEquality()` – Check equality based on meeting properties
- `testEmptyMeeting()` – Ensure meeting with minimal info initializes properly
- `testDescriptionUpdate()` – Verify updating and clearing description

Calendar Class – Lasse

- `testAddMeeting_holiday()` – Add holiday and mark as busy
- `testAddMeeting_normalMeeting()` – Add regular meeting and check busy hours
- `testAddMeeting_conflict()` – Detect conflict between overlapping meetings
- `testClearSchedule()` – Clear schedule for a specific day
- `testInvalidDate()` – Handle invalid month input
- `testInvalidTime()` – Handle invalid hour input
- `testGetAndRemoveMeeting()` – Retrieve and remove a meeting
- `testVacationBlocksInCalendar()` – Full-day vacation block logic
- `testMultipleMeetingsSameDay()` – Add non-overlapping meetings on the same day
- `testMeetingRemoval()` – Remove meeting and verify calendar updates
- `testClearScheduleAgain()` – Clear multiple meetings on the same day
- `testGetMeeting()` – Retrieve specific meeting details
- `testPrintAgendaMonth()` – Generate agenda for entire month
- `testPrintAgendaDay()` – Generate agenda for specific day
- `testInvalidDateAgain()` – Detect invalid calendar date (e.g., Feb 30)
- `testInvalidTimeAgain()` – Detect invalid hour range (> 24h)

Room Class – Shadia Nankya

- `testConstructorWithID()` – Verify that the room ID is correctly set through the constructor
- `testDefaultConstructor()` – Ensure the default constructor sets the room ID to an empty string
- `testAddValidMeeting()` – Add a valid meeting and check if the room is marked busy
- `testIsBusyNoMeetings()` – Confirm the room is not busy when no meetings are scheduled
- `testIsBusyWithMeeting()` – Check that the room is busy during a scheduled meeting
- `testGetMeeting()` – Retrieve a meeting from the room's agenda
- `testRemoveMeeting()` – Remove an existing meeting and confirm the room is available
- `testPrintAgendaMonth()` – Print the room's agenda for a full month
- `testPrintAgendaDay()` – Print the room's agenda for a specific day
- `testMultipleMeetingsDifferentDays()` – Add meetings on different days and confirm they're all stored
- `testInvalidMeetingParameters()` – Try to add a meeting with invalid date/time and expect an exception
- `testMidnightMeeting()` – Add a meeting from midnight to 1 AM and verify it's recorded
- `testEndOfDayMeeting()` – Add a meeting from 11 PM to midnight
- `testLastDayOfMonthMeeting()` – Add a meeting on the last day of the month
- `testFirstDayOfMonthMeeting()` – Add a meeting on the first day of the month
- `testRemoveNonExistentMeeting()` – Try to remove a meeting that doesn't exist (handles exception)
- `testGetNonExistentMeeting()` – Try to retrieve a meeting that wasn't scheduled
- `testSameTimeDifferentDays()` – Add meetings at the same time on different days
- `testSameDayDifferentMonths()` – Add meetings on the same day but in different months
- `testPartialOverlapMeeting()` – Add a meeting that partially overlaps with another (expect exception)

Organization Class – Wilson

- `testConstructor()` – Verify that employees and rooms are initialized with correct default values
- `testGetEmployee_Valid()` – Retrieve an existing employee by name
- `testGetEmployee_Invalid()` – Attempt to get a non-existent employee (expect exception)
- `testGetEmployees()` – Confirm the full list of employees is returned correctly
- `testGetRoom_Valid()` – Retrieve a valid room by ID
- `testGetRoom_Invalid()` – Attempt to get a non-existent room (expect exception)
- `testGetRooms()` – Confirm the full list of rooms is returned correctly
- `testGetEmployee_Null()` – Try to fetch a null employee name (expect specific exception message)
- `testGetRoom_Null()` – Try to fetch a null room ID (expect specific exception message)
- `testGetEmployee_EmptyString()` – Attempt to retrieve an employee with an empty name (expect exception)
- `testGetRoom_EmptyString()` – Attempt to retrieve a room with an empty ID (expect exception)
- `testGetEmployee_CaseSensitivity()` – Validate that employee name lookup is case-sensitive
- `testDuplicateEmployees()` – Add a duplicate employee and ensure the original is still returned

Phase 4: Regression & Final Testing (Apr 08 – Apr 10)

- Re-execute failed tests from Phase 3 after fixing defects
- Conduct end-to-end workflow test:
  Book meeting → Validate availability (room & person) → Print agenda

Deliverables:

- Final Test Summary Report
- Pass/Fail status
- List of open/closed defects

6. Test Cases (Detailed)

6.1 Person Class

Test ID	Description	Expected Result	Priority
P1	Default constructor	Name initialized to empty	High
P2	Parameterized constructor	Name set correctly	High
P3	Add valid meeting	Meeting added successfully	High
P4	Add conflicting meeting	TimeConflictException thrown	High
P5	Add back-to-back meetings	TimeConflictException thrown	Medium
P6	Add meeting with null room	Meeting added with null room	Medium
P7	Add meeting with empty attendees	Meeting added	Medium
P8	Remove meeting	Meeting removed	High
P9	Print daily agenda	Correct agenda displayed	High
P10	Print monthly agenda	Correct agenda displayed	High

6.2 Room Class

Test ID	Description	Expected Result	Priority
R1	Default constructor	ID initialized to empty	High
R2	Parameterized constructor	ID set correctly	High
R3	Add valid meeting	Meeting added	High
R4	Add conflicting meeting	TimeConflictException thrown	High
R5	Add back-to-back meetings	TimeConflictException thrown	Medium
R6	Remove meeting	Meeting removed	High
R7	Print daily agenda	Correct agenda displayed	High
R8	Print monthly agenda	Correct agenda displayed	High

6.3 Meeting Class

Test ID	Description	Expected Result	Priority
M1	Default constructor	All fields initialized	High
M2	Constructor with date	Date fields set	High
M3	Full constructor	All fields set correctly	High
M4	Invalid month (0)	IllegalArgumentException thrown	High

M5	Invalid month (13)	IllegalArgumentException thrown	High
M6	Invalid day (0)	IllegalArgumentException thrown	High
M7	Invalid day (32)	IllegalArgumentException thrown	High
M8	Invalid start time	IllegalArgumentException thrown	High
M9	Invalid end time	IllegalArgumentException thrown	High
M10	End time before start time	IllegalArgumentException thrown	High

6.4 Calendar Class

Test ID	Description	Expected Result	Priority
C1	Default constructor	Calendar initialized	High
C2	Add valid meeting	Meeting added	High
C3	Add conflicting meeting	TimeConflictException thrown	High
C4	Feb 29 in non-leap year	TimeConflictException thrown	Medium
C5	Invalid date	TimeConflictException thrown	High
C6	Remove meeting	Meeting removed	High
C7	Print daily agenda	Correct agenda displayed	High
C8	Print monthly agenda	Correct agenda displayed	High

6.5 Organization Class

Test ID	Description	Expected Result	Priority
O1	Default constructor	Employees and rooms initialized	High
O2	Get existing employee	Employee returned	High
O3	Get non-existent employee	Exception thrown	High
O4	Get existing room	Room returned	High
O5	Get non-existent room	Exception thrown	High

7. Test Data

7.1 Valid Test Data

- **Months:** 1 to 12
- **Days:** 1 to 31 (according to month)
- **Times:** 0 to 23 (24-hour format)
- **Room IDs:** LLT6A, LLT6B, LLT3A, LLT2C, LAB2
- **Employee Names:** Namugga Martha, Shema Collins, Acan Brenda, Kazibwe Julius, Kukunda Lynn

7.2 Invalid Test Data

- **Months:** 0, 13, negative values
- **Days:** 0, 32, invalid combinations (e.g., Feb 30)
- **Times:** -1, 24, non-numeric
- **Rooms/Employees:** Non-existent values

8. Test Execution Strategy

- Unit testing of individual classes using **JUnit**
- Integration testing across modules
- System testing for end-to-end flows
- Edge case and boundary validation
- Exception and error-handling verification

9. Expected Outcomes

- All valid operations execute successfully
- Invalid inputs raise appropriate exceptions
- Time and resource conflicts are properly detected
- Calendar and agenda data remains accurate
- User is informed with clear error messages

10. Test Deliverables

- JUnit test classes and test suites
- Test execution logs
- Bug and issue reports (if applicable)
- Code and test coverage metrics

11. Test Schedule

Activity	Duration
Unit Test Development	2 days
Integration Test Creation	1 day
Test Execution	1 day
Bug Fixing & Retesting	1 day

12. Risks and Mitigations

Risk	Impact	Mitigation Strategy
Incomplete test coverage	Medium	Use code coverage tools
Missed edge cases	High	Conduct peer reviews of test cases
Environment setup issues	Low	Share setup documentation among the team
Time constraints	Medium	Prioritize high-risk and critical test cases

Test run