**Exploratory Testing Session Report - Meeting Planner Application**

**Date:** 2024-04-01 **Tester:** Bard **Application:** Meeting Planner (Code Snippets Provided)

**Scope:**

This exploratory testing session focused on the core functionalities of the Meeting Planner application, primarily the Person, Room, and Meeting classes. Key areas explored included adding/removing meetings, attendee management, and basic conflict detection.

**Observations:**

1. **Meeting Creation and Time Conflicts:**
   * Successfully created meetings for single days and full days.
   * Verified that adding a meeting with overlapping times for a person throws a TimeConflictException.
   * Tested adding meetings with invalid parameters (e.g., negative start/end times) and observed appropriate behavior.
2. **Meeting Management:**
   * Added and removed attendees from meetings using the Person and Meeting classes.
   * Explored the toString method to verify meeting details are displayed correctly.
3. **Room and Calendar Integration (Limited):**
   * Due to the use of mocked Calendar objects in the provided test plan, a thorough exploration of room and calendar interactions wasn't possible. However, attempts to add meetings to rooms were made, assuming they would delegate to the Calendar class for conflict checking.
4. **Missing Functionalities:**
   * The ability to edit existing meetings (changing time, attendees, etc.) was not explored as the code snippets didn't show such functionality.
   * Features like searching for meetings, filtering by attendee/room, or exporting agenda data weren't covered in this session.

**Overall Findings:**

The core functionalities of adding/removing meetings, managing attendees, and basic conflict detection seem to work as expected. However, a more comprehensive test suite covering editing meetings, searching/filtering, and room/calendar interaction is needed for a complete picture.

**Recommendations:**

* Implement unit tests for all functionalities, including editing meetings, searching/filtering, and room/calendar interaction.
* Consider using dependency injection to improve the testability of calendar integration.
* Explore user interface (UI) testing tools and techniques to simulate user interactions and identify potential UI-related issues.

**Next Steps:**

* Based on the findings, further exploratory testing sessions can delve deeper into missing functionalities like meeting editing and search.
* Develop a more comprehensive test suite using JUnit and mocking frameworks for complete code coverage.

**Disclaimer:**

This exploratory testing session was based on limited code snippets. A more comprehensive assessment would require access to the entire application codebase and UI.