Explanation of Modifications Required in Submission #2 to Fit the New Requirements

To incorporate the new requirements outlined for the updated Schedule Tool, the following changes would need to be made to the **requirements analysis and modeling** in Submission #2:

1. Modifications to the Class Diagram

The following updates will be necessary in the class diagram:

New Classes

- **NonCourseBlock**:
    - Represents a time block for non-course activities, such as seminars or workshops.
    - **Attributes**:
        - name (String): Name of the non-course block (e.g., "Seminar").
        - timeslot (String): The time period of the block (e.g., "F 1:00PM-2:00PM").
    - **Methods**:
        - getName(): Returns the name of the block.
        - getTimeslot(): Returns the timeslot for the block.
    - **Relationships**:
        - Associated with the Group class as a list (List<NonCourseBlock>), similar to the Course relationship.

Modified Classes

1. **Group**:
    - A Group now contains both Course and NonCourseBlock objects.
    - **Attributes**:
        - groupName (String): Name of the group.
        - courses (List<Course>): List of courses in the group.
        - nonCourseBlocks (List<NonCourseBlock>): List of non-course blocks in the group.
    - **Methods**:
        - addCourse(Course course): Adds a course to the courses list.

        - addNonCourseBlock(NonCourseBlock block): Adds a block to the nonCourseBlocks list.
        - displaySchedule(String filename): Displays the schedule for both courses and non-course blocks and saves it to a file.
    - **Relationships**:
        - Now has an additional composition relationship with NonCourseBlock.

2. **Course**:
    - No changes to structure; however, its relationship with Group is now paired with NonCourseBlock.

Diagram Notes

The class diagram will now depict:

- A **composition relationship** between Group and NonCourseBlock.
- Updates to the Group class to show relationships with both Course and NonCourseBlock.

2. Sequence Diagram Changes

The sequence diagram for **displaying a schedule** needs to be updated to include the following steps:

1. **Invocation of** addNonCourseBlock:
    - Sequence: The Main class invokes addNonCourseBlock to add non-course blocks to the Group.
2. **Conflict Checking**:
    - The checkConflicts method in Group needs to iterate over both courses and nonCourseBlocks to identify any overlapping time slots.
3. **Schedule Display**:
    - The sequence for displaySchedule will now:
        - Iterate through both courses and nonCourseBlocks.
        - Include logic for formatting and displaying both types of schedule items.

3. Updated Use Case

The existing use case for adding and managing courses needs to be expanded to include non-course blocks:

New Use Case: Add Non-Course Block

- **Actors**:
    - User
- **Description**:
    - The user inputs the name and timeslot for a non-course block.
    - The system validates the input and adds the block to the selected group.
- **Basic Flow**:

1. User selects the option to add a non-course block.
2. System prompts for the block name and timeslot.
3. User provides the input.
4. System validates the timeslot and adds the block to the group.

Modified Use Case: Check Conflicts

- **Actors**:
    - System
- **Description**:
    - The system checks for overlapping time slots between courses and non-course blocks.
- **Basic Flow**:

1. System iterates through all courses in the group.
2. For each course, it checks for overlapping times with:
    - Other courses in the group.
    - Non-course blocks in the group.
3. System reports any conflicts detected.

4. Requirement Modifications

Functional Requirements

- **New**: The system must allow users to add non-course blocks to groups.
- **Modified**: The system must check for conflicts between courses and non-course blocks in the same group.

Non-Functional Requirements

- **New**: The system must ensure that non-course blocks are displayed clearly alongside courses in both terminal output and saved schedules.

Summary of Modifications

1. **Class Diagram**:
   - Add the NonCourseBlock class.
   - Update the Group class to manage both Course and NonCourseBlock objects.
2. **Sequence Diagram**:
   - Include steps for adding non-course blocks and checking conflicts between courses and blocks.
3. **Use Cases**:
   - Add a new use case for managing non-course blocks.
   - Update the use case for conflict checking to include non-course blocks.
4. **Requirements**:
   - Update functional and non-functional requirements to reflect the ability to manage and display non-course blocks.