# Conference room reservation system

One of the core problems which people face at every company is unavailability of conference rooms. So we thought of building one conference room management system. System should have the following features.

Assumptions:

1) Conference rooms are scattered across multiple buildings and multiple floors and each floor can have multiple conference rooms, but each conference room should be uniquely identifiable.
2) Booking will be done for slots in hours. Hours will be taken in 24 hours format eg. Book from 1am to 3am {1:3}, from 12pm to 1pm {12:13}
3) Each conference room can be booked given that no one has already booked for that slot.
4) System will be used only by one user only.
5) Booking can be done for 1 day only.

Features:

1. User should be able to list down all the conference rooms available in any building. Eg: Alpha building has conference rooms with names: a1, a2, a3, a4 etc.
2. User should be able to find suitable rooms to book for a given slot.
3. User should be able to cancel existing booking and rooms should be free to be booked again for that slot.
4. List down all the bookings made by the current user.
5. Command which will be executed to perform the above actions is listed below.

**Note* All the input params given below are for demonstration purposes only, user can create his/her own building name and floor name and conference rooms.**

Commands:

1) **ADD BUILDING <building_name>** //Adds building in the system
    1. Eg: **ADD BUILDING** flipkart1
    2. Output: Added building flipkart1 into system.
2) **ADD FLOOR <buildingName> <floorName>**
    a) **Eg.** ADD FLOOR FLIPKART1 FirstFloor
3) **ADD CONFROOM <buildingName> <floorName> <conferenceRoomID>**
    a) Eg: **ADD CONFROOM flipkart1 firstfloor c1**
    b) Above commands adds **c1** conference room in **firstfloor** of building **flipkart1**
    c) Output: Added conference room c1 at firstfloor in flipkart1
4) BOOK <SLOT> <BUILDING> <FLOOR> <ROOM ID>

a) Book the given Conference room for a given slot, in the given floor of the building.
b) Eg: **BOOK 1:5 FLIPKART1 SEVENTH C1**

5) CANCEL <SLOT> <BUILDING> <FLOOR> <ROOM ID>
   a) Cancels the slot booked for the floor in a given building.

6) LIST BOOKING <BUILDING> <FLOOR>
   a) Should list down all the bookings made by current user
   b) Output format: **<SLOT> <FLOOR> <BUILDING> <roomName>**
   c) **2:6 THIRDFLOOR FLIPKART1 c1**
   d) **6:10 THIRDFLOOR FLIPKART1 c2**

**7) SEARCH <SlotName> <BuildingName> <FloorName>**
   a) Search should return possible available rooms for given parameters.
   b) **SEARCH 2:3 flipkart3 seventhFloor** -> Search conf rooms available for booking from 2->3 in **flipkart3** building and **seventhFloor** floor
   c) If no room is available for booking for given slot, search will return "**No Rooms available**"

**Bonus Functionality:**
Existing search will return empty results if no rooms are available for a given slot . Users want a suggestion functionality using which users can get a list of possible future slots **[Size limit to 3]** which can be booked.
Eg. Assume no room is available for a slot then search will return an empty result while SUGGEST command will return the next 3 suggestions.
Command: **SUGGEST** 3:10

**Expectation**

1. **Code should be Demo able and functionally complete.**
2. **Code should fail gracefully with a proper error message for corner/invalid cases.**
3. **Code should be modular, try thinking in terms of Object Oriented Design.**
4. **Input can be taken from the command line or in the main function.**
5. **Do not use any database or NoSQL store, use in-memory data-structure.**
6. **Do not create any UI for the application.**
7. **Write a driver class for demo purposes. Which will execute all the commands at one place.**
8. **Please prioritize code compilation, execution and completion.**
9. **Work on the expected output first and then only work on bonus features.**