Problem Statement

Make an API that facilitates a conference booking application. Following Endpoints are expected to be present. Assume that all user actions are happening in UTC timezone only.

Add Conference: Add a new conference. It takes following parameters as input

1. Name: Any Alphanumeric String. Spaces are the only special character allowed
2. Location: Any Alphanumeric String. Spaces are the only special characters allowed
3. Topics: Comma Separated Strings. Strings can be Alphanumeric with spaces allowed. Maximum 10 Strings
4. Timing
    a. Start Timestamp
    b. End Timestamp
    c. Conditions: Start Timestamp must be before End Timestamp. Duration should not exceed 12 hours
5. Available Slots: A integer greater than 0

Expectations: Conference Name should be unique globally unique in system. Trying to add a conference with the same name as existing one should result in error. There is no way to edit an added conference. The Timezone for Start and End Timestamp should be UTC.

Add User: Add a new user to the system. It takes following parameters

1. UserID: An Alphanumeric string with no special characters allowed
2. Interested Topics: Comma Separated Strings. Strings can be Alphanumeric with spaces allowed. Maximum 50 Strings

Expectations: UserID would be unique in the system. No endpoint to edit an existing user is needed.

Book a Conference: Book a conference slot for a given user. It takes the following parameters

1. Name: Name of the conference
2. UserID to book conference for

This endpoint returns a booking ID that can be used to track status of the booking. It will also be used to confirm booking in case of waitlist and/or cancel the booking.

The conference and user should already exist in the system. If they don't then throw an error. The booking should only be allowed if there are any remaining slots for the particular conference and the user doesn't have any other overlapping conference booked already. A successful booking leads to the number of slots getting reduced by 1.

If booking is successful, return a success booking message to the user.

If all the seats are already booked, then add the user to a waitlist for this conference. The waitlist should work on a first come first serve basis manner. Whenever a new slot becomes available (due to existing booking getting canceled) then the first waiting user gets a chance to confirm their booking within the next 1 hour (From the time when this option became available to them). Users should use **Confirm Waitlist Booking** Endpoint for this purpose.

If that user doesn't confirm his booking and or cancel within 1 hour then he is moved to the end of the waitlist and the next waiting user gets a chance to confirm their booking. At a given moment there can be multiple people who can confirm their booking. (This can happen for example if there are simultaneous cancellations).

Also any user can anytime cancel their booking or get removed from the waitlist using **Cancel Booking** Endpoint.

If the user already has an active (non-cancelled) booking for this conference then don't allow a new booking and also return an error with the ID of the existing active booking.

Lastly if any user books a conference/confirms his booking from the waitlist, then they also get removed from the waitlist of all the other overlapping conferences.

Users can re-book the same conference even if they have previously canceled their booking (confirmed/waitlisted) given that they satisfy all other constraints. This booking should have a separate ID.

One more thing is that booking for a conference or booking confirmation is only allowed before the conference start timestamp. Once a conference has started then, all the waitlisted booking gets auto canceled and no more changes to any booking of that conference is allowed (This includes cancellations / new bookings / booking confirmation).

Booking Status: Returns Booking Status (Confirmed/Waitlisted/Canceled) for a given booking ID. If status is waitlisted then also indicate whether the user can confirm their booking now and till what time.

Confirm Waitlist Booking: Takes Booking ID as parameter. Return Success Message if given booking is in the waitlist and there is any empty slot for this booking ID. Else returns error.

Cancel Booking: Takes Booking ID and cancels booking or removes from waitlist. If the given booking is already in a canceled state then return Error.

Things to look out for (in order or priority):

1. Complete as many features as possible
2. API Design and proper use of HTTP Codes
3. Design of Entities and their relationships
4. Input Validation in APIs

Good to have (not necessary, implement in any order):

1. Storing data in a persisted datastore
2. Making sure to ensure concurrency (prevent race conditions) to avoid double booking and related edge cases
3. Additional endpoints that can be implemented
    a. Search Conference Endpoint: Search for present/past conferences based on location, topics, name, date time, duration. Design the API as seems fit
    b. Suggested Upcoming Conference Endpoint: Takes a User ID and returns top 10 suggested upcoming conferences based on topics that the user is interested in. The ranking algorithm for this is left upon interviewee to decide