Assessment Question:

You are tasked with building an event management API using Flask. The application allows users to:

1. **Create an Event**: The event must include the venue, date, ticket categories (with price), seat limits, and event details.
2. **Book Tickets**: Users can book tickets for available events, ensuring that the event is not in the past and that the event is not sold out.

---

1. Event Model and API Design:

Models (Use any ORM)

- Event: The event model should contain:
    - id: Unique identifier
    - name: Name of the event
    - venue: Location where the event will be held
    - date: Date of the event (must be a future date)
    - ticket_categories: A list of ticket categories, each containing:
        - name: Category name (e.g., General, VIP)
        - price: Price of the ticket in that category
        - seats_limit: Limit of seats for that category
        - seats_sold: Number of seats already sold
    - details: Further description or details about the event

API Endpoints:

1. **POST /events**: Create a new event. The request body should include details about the event, venue, date, ticket categories, etc.
2. **GET /events**: List all upcoming events.
3. **POST /events/{event_id}/book**: Book a ticket for a specific event. The request body should include the ticket category, number of tickets, and any other relevant details.

---

2. Business Logic and Validation:

1. **Event Creation**: Validate the event date to ensure it's set to a future date. Prevent the creation of events with a date in the past.
2. **Ticket Booking**:
    - Ensure that tickets can only be booked for events that are not in the past.
    - Ensure that the number of tickets requested does not exceed the available seats for the selected ticket category.
    - Prevent booking if the event is sold out (i.e., if the number of seats sold equals or exceeds the seat limit).
    - Provide appropriate error messages when a booking attempt fails due to the event being in the past or sold out.

3. Example Data and Scenarios:

Creating an Event:

- POST /events:


```
{
  "name": "Music Concert",
  "venue": "Stadium A",
  "date": "2025-06-15T18:00:00Z",
  "ticket_categories": [
    {
      "name": "General",
      "price": 50,
      "seats_limit": 100,
      "seats_sold": 0
    },
    {
      "name": "VIP",
      "price": 150,
      "seats_limit": 50,
      "seats_sold": 0
```

```
    }
  ],
  "details": "A great music concert."
}
```

Booking Tickets:

- POST /events/1/book:

```
{
  "ticket_category": "General",
  "quantity": 2
}
```

Invalid Booking:

- Booking for a past event
- Booking exceeding the available seats
- Booking when the event is sold out

---

4. Flask Code Structure and Implementation:

Provide a solution that includes:

1. Event and Ticket Models using SQLAlchemy.
2. Endpoints for creating an event, listing events, and booking tickets.
3. Validations for the booking API to ensure only future events are bookable, and to prevent overbooking.
4. Error handling to return appropriate status codes and messages for invalid scenarios (e.g., past event, sold-out event).

Deliverables:

1. Python code with the Flask app and routes.
2. Model definitions (any ORM of choice).
3. Unit tests (Optional but highly encouraged) to test API behaviors, such as booking for past events or sold-out events.