**Full-Stack Developer Test: Conference Management System**

**Objective**

Develop a **Conference Management System** that allows users to create, manage, and track conference events and participants. The application should follow **functional programming principles**, be **fully Dockerized**, and use **modern front-end technologies**. The UX and design won't be evaluated in detail, using a UI framework like Bootstrap, Tailwind, or MUI is preferred.

**Technology Stack**

**Backend (Java/Kotlin)**

* **Programming Language:** Java / Kotlin
* **Framework:** Spring Boot (Java/Kotlin)
* **Database:** MySQL or PostgreSQL or MongoDB or SQLite
* **Build Tool:** Maven, Gradle
* **Testing:** use your preferred framework
* **Containerization:** Docker & Docker-Compose

**Frontend (Your Choice)**

* **Framework:** React / Vue / Next.js
* **Styling:** Bootstrap, Tailwind, MUI, AntD (preferred)
* **Testing:** UI testing with **Playwright**
* **Optional:** Use admin UI frameworks like **React-Admin, Refine.app, or Retool** to save development time

**Infrastructure**

* **Pack backend, frontend, and database into a Docker container using Docker Compose.**

**Functional Requirements**

**1. Conference Management**

**Create a New Conference**

* **Form Inputs:**
  + name (max 150 characters)
  + dateTime (conference start time)
  + roomId (conference room selection)
* **Validation Rules:**
  + Ensure all required fields are provided.
  + A conference cannot overlap with another conference in the same room.
  + A conference name should be unique for the same date.
  + Conferences must be scheduled at least 24 hours in advance.
* **UI Behavior:**
  + Present a clear form with inline validation feedback.
  + Use a simple <select> dropdown for room selection.

**Cancel a Conference**

* **Action:**
  + Allow cancellation from either the conference list (dashboard) or a detailed view.
  + A conference **cannot** be canceled if it has registered participants.
* **User Confirmation:**
  + Include a confirmation prompt before finalizing the cancellation.
* **Feedback:**
  + Display a clear success or error message after cancellation.

**Check Conference Room Availability**

* When creating a conference, verify if the room is already booked.
* If a chosen room cannot accommodate additional participants, show an error message.

**2. Conference Participant Management**

**Add a Participant to a Conference**

* **Form Inputs:**
  + fullName (string, required)
  + birthDate (date, required)
  + conferenceId (the selected conference)
* **Validation Rules:**
  + Ensure all required fields are provided.
  + A participant must be **at least 18 years old** to attend.
  + A participant **cannot register twice** for the same conference.
* **UI Behavior:**
  + Use a minimal, clear form that provides immediate inline feedback.

**Remove a Participant from a Conference**

* **Action:**
  + Allow removal from the participant list.
* **User Confirmation:**
  + Include a confirmation step to avoid accidental removals.
* **Feedback:**
  + Provide clear notifications upon success or error.

**3. Room Management (Optional – Pre-seeded Rooms Required)**

**CRUD Operations**

* **Create a Room:** Add new conference rooms (name, location, max seating).
* **Read/View Rooms:** Display a list of available rooms.
* **Update a Room:** Enable editing of room details.
* **Delete a Room:** Allow removal with confirmation.
* **Initial Data Requirement:** At least **two conference rooms should be pre-seeded** when the project starts.

**4. UI-Specific Functionalities**

**Conference List/Dashboard**

* **Display:** Show a list of conferences with basic details (name, date/time, room, participant count).
* **Navigation:** Quick access to conference creation, cancellation, and participant management.

**Overall UI Behavior**

* **Simplicity:** Keep the design intuitive and avoid unnecessary complexity.
* **Use UI frameworks** like Bootstrap, Tailwind, MUI, or AntD.

**Business Rules Summary**

1. **Conference Name and Date Validation:** No two conferences should have the same name on the same date in the same room.
2. **Room Capacity Check:** Only rooms with at least **10 available seats** can be booked.
3. **Age Requirement:** Participants must be **18+ years old** to register.

**Data Model (Suggested Schema)**

**Conference**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | UUID | Unique identifier |
| name | String(150) | Conference name (max 150 chars) |
| dateTime | DateTime | Conference start time |
| roomId | UUID | Reference to the conference room |

**Participant**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | UUID | Unique identifier |
| fullName | String | Person’s full name |
| birthDate | Date | Date of birth |
| conferenceId | UUID | Reference to the conference |

**Conference Room**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | UUID | Unique identifier |
| name | String | Room name (e.g., "Room 101") |
| location | String | Floor or specific location |
| maxSeats | Integer | Maximum seating capacity |

**Non-Functional Requirements**

Follow **functional programming principles** (immutability, pure functions).  
Implement **input validation** to prevent issues (e.g., overbooking).  
Ensure **proper error handling** (e.g., prevent duplicate registrations).  
Use **Swagger/OpenAPI** for REST API documentation.  
**Dockerize** the entire application with **Docker-Compose**.

**Evaluation Criteria**

Functional programming approach.  
Code clarity, structure, and maintainability.  
Proper use of REST API and data validation.  
Effective implementation of automated testing (REST-assured for API, Playwright for UI).  
Ability to set up a fully Dockerized full-stack application.