Тестове завдання “Бронювання переговорних кімнат”

(Meeting Rooms Booking)

Мета

Перевірити навички **C#/.NET**, **ASP.NET Core**, **EF Core**, **PostgreSQL**, **Git**, ООП/SOLID/Patterns, а також розуміння **Clean Architecture / Modular Monolith / Vertical Slice** і побудови **робочих процесів (workflows)**.

Опис

Реалізуйте API для сервісу **бронювання переговорних кімнат** в компанії. Співробітник створює запит на бронювання, система перевіряє конфлікти, після чого запит проходить процес підтвердження.

Функціональні вимоги

1. **Довідники**
   - **Room**: назва, місткість, локація, активна/неактивна.
   - **TimeSlot** визначається `startAt`, `endAt` (UTC).
   - Валідації: `endAt > startAt`, тривалість не більше N годин (наприклад, 4).
2. **BookingRequest**
   - Створення запиту на бронювання кімнати на певний слот.
   - Перелік учасників (emails) — мінімум 1.
   - Опис/мета зустрічі (обов’язково).
   - **Правила конфліктів**: в одній кімнаті не може бути перетинів по часу для підтверджених бронювань (і/або для запитів в обробці — на ваш вибір, але пояснити в README).
3. **Workflow / процес**
   - Стани: `Draft` → `Submitted` → `Confirmed` або `Declined` → (опційно) `Cancelled`.
   - `Submitted` можливий лише якщо:
     - кімната активна,
     - слот валідний,
     - є учасники,
     - немає конфлікту.
   - `Confirmed/Declined` можливий лише для `Submitted`.
   - `Cancelled` можливий лише для `Confirmed` (з причиною).
   - Ведіть **історію переходів** (хто/коли/з якого стану в який + причина).

4. **Ролі (спрощено)**
   - “Employee”: створює/відправляє/скасовує (свої).
   - “Admin”: підтверджує/відхиляє.
   - Реалізацію доступу можна спростити (наприклад, `X-UserId`, `X-Role`), але перевірка прав має бути.

API (мінімум)

- `POST /rooms` — створити кімнату (admin)
- `GET /rooms` — список кімнат (фільтри: локація, місткість, active)
- `POST /bookings` — створити запит (Draft)
- `POST /bookings/{id}/submit` — відправити
- `POST /bookings/{id}/confirm` — підтвердити (admin)
- `POST /bookings/{id}/decline` — відхилити (admin, причина)
- `POST /bookings/{id}/cancel` — скасувати (employee, причина)
- `GET /bookings/{id}` — деталі + історія
- `GET /bookings?from=...&to=...&roomId=...&status=...` — пошук

Технічні вимоги

- **.NET 8+ (LTS)**, **ASP.NET Core Web API**.
- **EF Core** + **PostgreSQL**, міграції.
- Архітектура на вибір:
  - **Clean Architecture** (API/Application/Domain/Infrastructure),
  - або **Modular Monolith** (Rooms module, Bookings module),
  - або **Vertical Slice** (feature slices на рівні endpoint’ів/команд).
- Доменно-орієнтована модель: правила та переходи станів — у домені, не в контролерах.

Вимоги до якості (ООП/SOLID/Patterns)

- Чисті інваріанти й захист від некоректних переходів.
- Патерни за потреби: State, Domain Events, Policy/Specification.
- Обробка помилок через коректні HTTP-коди + `ProblemDetails`.
- Логування ключових подій.

БД та продуктивність

- Конкурентність: мінімально обґрунтуйте підхід проти “подвійного підтвердження” при одночасних запитах (optimistic concurrency, транзакція/лок, унікальні обмеження/перевірки).
- Перевірка перетинів слотів повинна бути ефективною (не тягнути всі бронювання в пам’ять).

Git вимоги

- Репозиторій з історією: **5–10 логічних комітів** (feature-by-feature).
- README з інструкцією запуску та коротким поясненням архітектури.

Результати

- Репозиторій/архів.
- `README.md`:
  - запуск, міграції, конфігурація
  - коротко: як організована архітектура (модулі/слайси/шари)
  - приклади API-запитів (curl)