Kryteria oceny projektu z Technik tworzenia skalowalnych aplikacji webowych

1. Działające MVP + kompletność scenariuszy (60%)

- Instalacja aplikacji z repo z gotowością do demonstracji
- Kompletność kluczowych procesów end-to-end (bez „martwych ekranów" i brakujących kroków).
- Poprawne CRUD + walidacje dla podstawowych encji domenowych.
- Czytelny, spójny UI i logiczne flow użytkownika.
- Obsługa błędów „użytkowych" (np. brak uprawnień, brak danych, konflikty terminów).

Dodatkowe punkty

- Statusy i workflow (np. „wniosek → zaakceptowany → anulowany").
- Raporty/eksport (CSV/PDF) albo przynajmniej widok „do wydruku" dla najważniejszych danych.
- Dobre UX: potwierdzenia, komunikaty, „empty states", sensowne domyślne filtry.

2. Skalowalność i poprawność współbieżności (10%)

- Backend stateless (brak trzymania sesji w pamięci pojedynczej instancji lub jasny mechanizm współdzielenia).
- Wydajne listy: paginacja + filtrowanie po stronie backendu, a nie „pobieramy wszystko i filtrujemy w JS".

Dodatkowe punkty

- Indeksy w bazie dla kluczowych zapytań + uzasadnienie (choćby 2–3 najważniejsze).
- Cache (np. Redis / pamięć aplikacji) dla słowników lub dostępności.
- Prosty test obciążeniowy (np. 50 równoległych zapisów) + wyniki.

3. Bezpieczeństwo, role, audyt (10%)

- RBAC: jasno zdefiniowane role i uprawnienia (minimum 3 role).
- Autoryzacja na poziomie API (nie tylko „ukryty przycisk w UI").
- Bezpieczna walidacja danych wejściowych (DTO / schema validation).

Dodatkowe punkty

- Logowanie / nieudane logowanie (bez haseł).
- Logowanie kluczowych operacji: utworzenie/edycja/usunięcie (rezerwacji/terminu)
- Ochrona przed typowymi problemami (przejście OWASP ZAP Baseline Scan):
  SQL injection, XSS, CSRF (jeśli są cookie).

4. Testy + jakość kodu + dokumentacja API (10%)

- Czytelna struktura projektu (warstwy/moduły), spójne nazewnictwo.
- Obsługa błędów i kody HTTP zgodne z semantyką (400/401/403/404/409/500).

Dodatkowe punkty

- Dokumentacja API (OpenAPI/Swagger) aktualna względem implementacji.
- Testy integracyjne API + baza (np. zapis/rezerwacja + constrainty).

5. Obserwowalność, dokumentacja architektury, konteneryzacja (10%)

- Konfiguracja przez env vars (bez sekretów w repo).
- Dokumentacja architektury: 1–2 diagramy.

Dodatkowe punkty

- Docker dla backendu + DB (i ewentualnie frontu) + docker-compose.
- Prosty monitoring (np. Prometheus/Grafana) albo gotowość na to (metryki).

Poszczególne oceny będą zróżnicowane w zależności od roli i stopnia zaangażowania w projekcie.
Proszę przygotować plik z rolami w projekcie i opisem zrealizowanych zadań.