Java wymagania

Notebookml

https://notebooklm.google.com/notebook/eed794c7-ddaf-4f62-b7b0-0344a00295e2

System Rezerwacji Usług Beauty (Salon Manager)

Stwórz aplikację do kompleksowego zarządzania salonem usługowym (fryzjer/kosmetyczka). System musi obsługiwać strefę publiczną (cennik, opinie), panel klienta (interaktywny kalendarz rezerwacji) oraz panel administratora. Kluczowym elementem technicznym jest algorytm rezerwacji: system musi sumować czas trwania wybranych usług, wyszukiwać odpowiednio długie luki w terminarzu oraz zarządzać cyklem życia rezerwacji (tworzenie -> potwierdzenie klienta -> zatwierdzenie administratora -> blokada edycji).

📌 Wymagania wspólne (dla wszystkich projektów)

🗄️ Model danych, Repository i JdbcTemplate

Encje JPA i relacje (4%)

- Konfiguracja encji:
  - `@Entity`, `@Id`, `@GeneratedValue`, `@Column`
- Relacje:
  - `@OneToMany` / `@ManyToOne` + `@JoinColumn`
  - `@ManyToMany` + `@JoinTable`
- Poprawne mapowanie relacji i kluczy obcych

Repository i zapytania JPA (5%)

- Repository rozszerzające `JpaRepository<T, ID>`
- Custom queries:
  - Metody typu `findBy...`
  - Adnotacja `@Query`
- Paginacja:
  - `Page<T>`, `Pageable`
- Konfiguracja w `application.yml`:
  - `datasource`, `jpa`, `hibernate`
- Inicjalizacja danych:
  - Pliki `.sql`

JdbcTemplate – zapytania SQL (5%)

- Dodanie zależności `JdbcTemplate`
- Operacje:
  - `SELECT` → `query()` + `RowMapper`
  - `INSERT / UPDATE / DELETE` → `update()`
- Użycie w:
  - warstwie serwisu **lub** dedykowanym DAO

🌐 REST API

CRUD Endpoints (7%)

- `@RestController` + `@RequestMapping("/api/v1/..." )`
- Endpointy:
  - `GET` – lista (z paginacją), pojedynczy zasób
  - `POST` – tworzenie
  - `PUT` – aktualizacja
  - `DELETE` – usuwanie
- Obsługa parametrów:
  - `@PathVariable`, `@RequestBody`, `@RequestParam`
- `ResponseEntity` + kody HTTP:
  - `200 OK`, `201 Created`, `204 No Content`
  - `400 Bad Request`, `404 Not Found`

Dokumentacja API – Swagger (5%)

- Springdoc OpenAPI
- Swagger UI dostępny pod:
  - `/swagger-ui.html`
- Kompletna i czytelna dokumentacja endpointów

⚙️ Warstwa aplikacji – Business Logic

Serwisy i transakcje (3%)

- `@Service`
- `@Transactional(readOnly = true / false)`
- Wstrzykiwanie zależności przez **konstruktor**
- Mapowanie:
  - Entity ↔ DTO
- Obsługa błędów:

- Własne wyjątki (np. `ResourceNotFoundException`)
- `@RestControllerAdvice` + `@ExceptionHandler`

Walidacja danych (3%)

- Walidacja w DTO:
  - `@NotNull`, `@NotBlank`, `@Size`, `@Email`
  - `@Valid`
- Bean Validation:
  - kontroler + serwis
- Spójne komunikaty błędów

Thymeleaf – widoki (3%)

- `@Controller`, `Model`, `@ModelAttribute`
- Szablony:
  - Listy: `th:each`
  - Formularze: `th:object`, `th:field`
- Błędy walidacji:
  - `th:errors`
- Layout:
  - `th:fragment`, `th:replace`
- Stylowanie:
  - Bootstrap 5

Operacje na plikach i eksport (3%)

- Upload plików:
  - `MultipartFile`, `enctype="multipart/form-data"`
- Zapis:
  - `Files.copy()`
- Download:
  - `Resource`, `ResponseEntity<byte[]>`
- Eksport danych:
  - CSV / PDF

🔐 Spring Security (4%)

- Konfiguracja:
  - `SecurityFilterChain` jako `@Bean`
- Autoryzacja:
  - `authorizeHttpRequests`

    - `requestMatchers`
- Logowanie:
    - `formLogin`
- Hasła:
    - `BCryptPasswordEncoder`
- Użytkownicy:
    - `UserDetailsService`
    - `loadUserByUsername`

🧪 Testowanie

Testy warstwy danych (2%)

- `@DataJpaTest`
- Minimum **10 testów CRUD**
- Testy:
    - Repository
    - Custom queries
    - `RowMapper`

Testy serwisów (3%)

- Mockito:
    - `@Mock`, `@InjectMocks`
    - `when().thenReturn()`
    - `verify()`
- Testy jednostkowe logiki biznesowej

Testy REST i integracyjne (2%)

- `@WebMvcTest` **lub** `@SpringBootTest`
- `MockMvc`:
    - `perform()`
    - `andExpect()`
- Security:
    - `@WithMockUser`
- Minimum **5 scenariuszy biznesowych**

Coverage i jakość (1%)

- JaCoCo:
    - Coverage ≥ **70%**
- Testy:

- Happy Path
- Error Cases

🎯 Wymagania specyficzne dla projektu

🌍 Strefa Publiczna i Cennik (10%)

- Dostęp **bez logowania**
- Informacje o salonie
- Cennik usług (z bazy danych):
  - Nazwa
  - Cena
  - Czas trwania
- Moduł opinii:
  - Dodawanie recenzji (treść + data)
  - Wyświetlanie posortowane chronologicznie

🗓️ Inteligentny Kalendarz i Rezerwacja (14%)

- Formularz rezerwacji:
  - Wybór **wielu usług** (np. strzyżenie + modelowanie)
- Logika systemu:
  - Sumowanie czasu trwania usług
  - Wyświetlanie dostępnych terminów
- Rezerwacja:
  - Zapis blokuje wybrany przedział czasowy

👤 Panel Klienta i Cykl Życia Rezerwacji (10%)

- Rejestracja i logowanie
- Historia wizyt
- Zarządzanie rezerwacjami:
  - Potwierdzenie przybycia
  - Edycja terminu/usług
  - Anulowanie wizyty
- Walidacja logiki:
  - Brak edycji po zatwierdzeniu przez salon

🛠️ Panel Administratora – Zarządzanie Wizytami (10%)

- Widok terminarza właściciela
- Operacje:
  - Zatwierdzanie wizyt (blokada edycji klienta)
  - Edycja wizyty
  - Usuwanie wizyty
- Powiadomienia / komunikaty dla klienta

📊 Zarządzanie Ofertą i Statystyki (6%)

- CRUD dla usług:
  - Dodawanie
  - Edycja ceny i czasu trwania
- Statystyki per klient:
  - Średni czas trwania wizyt
  - Średnia kwota zostawiana w salonie
- Implementacja:
  - SQL / JPQL queries
  -