Re-Exam Spring 2025 – Car Rental System

Technologies & Guidelines

- Use Java, Javalin, JPA with Hibernate, PostgreSQL, and REST
- Do not use Spring Boot, Spring Data, Spring Security, or any other external frameworks not taught in this course
- Duration: 5 hours
- Submit a .zip file of your project and a GitHub link (do not push until the end)

Car & Booking Domain

The system you will build is a car rental system. It consists of two main entities: Car and Booking.

- A Car has: make, model, year, licensePlate, pricePerDay
- A Booking has: startDate, endDate, customerName, customerEmail, and is linked to one Car
- One car can have multiple bookings, but each booking belongs to only one car

Task 1: Setup (5%)

Requirements

- Create a new Java project with Javalin and JPA
- Document your steps and assumptions in a README.md file

Task 2: JPA & Persistence (25%)

User Stories

- As a customer, I want to browse all available cars so I can choose one to book
- As a customer, I want to book a car for specific dates with my contact information
- As a customer, I want to view my past bookings

Actions

- Create JPA entities: Car and Booking, with a OneToMany relationship
- Implement DTOs and DAOs using a shared CRUD interface
- Add methods: addCarToBooking(int bookingId, int carId) and getBookingsByCar(int carId)
- Populate the database with example cars and bookings

Task 3: REST API with Javalin (25%)

User Stories

- As a customer, I want to view a car’s details including past bookings
- As a customer, I want to update or cancel my booking
- As an admin, I want to add, update, or delete cars in the system

Actions

- Implement REST endpoints for CRUD operations and booking-car linking
- Follow REST conventions (e.g., GET /cars, POST /bookings, PUT /bookings/{id}/car/{carId})
- Document the API responses in README.md with test results

Task 4: REST Error Handling (5%)

User Stories

- As a customer, I want to receive a clear error message if a booking or car doesn't exist

Actions

- Return JSON-formatted errors for not found or invalid operations

Task 5: Streams & Queries (10%)

User Stories

- As a customer, I want to filter cars by model or year to quickly find the one I want
- As an admin, I want an overview of how many days or how much revenue each car has generated

Actions

- Use streams or queries to summarize booking info per car
- Choose one output format (total days or total price)

Task 6: REST Testing (15%)

Requirements

- Use RestAssured to test all endpoints
- Setup Javalin and test data in @BeforeAll and @BeforeEach
- Verify booking responses include car info and maintenance tips

Task 7: Security (15%)

User Stories

- As a user, I want to log in so I can manage my bookings
- As an admin, I want to secure car management operations so only admins can access them

Hints

- Use JWT-based auth with role-based access control
- Protect POST, PUT, DELETE routes for cars/bookings
- Make sure unauthenticated test requests return 401
- Document how to fix or bypass failed test cases in README.md