XenPlan — Advanced Java Mini-Project (Spring Boot & Vaadin)

Event Reservation Management System

Deadline: 31/12/2025
Technologies: Java 17+, Spring Boot, Vaadin, H2 Database

---

1. Introduction

Context

You are tasked with developing a complete web application for managing reservations of cultural events (concerts, theatre, conferences, sports, etc.).

The platform allows **organizers** to create and manage events, and **users** to browse and reserve seats through a modern, responsive web interface.

Learning Objectives

This project allows you to practice: - Advanced Java concepts (Streams, Optional, Lambdas, Generics) - Spring Boot and its ecosystem - Vaadin for UI development (Java-based frontend) - Data persistence using JPA / Hibernate - Exception handling and data validation - Unit and integration testing - MVC architecture and design patterns

---

2. Technical Architecture

- **Backend:** Spring Boot 3.x
- **Frontend:** Vaadin 24.x (Java)
- **Database:** H2 (embedded mode)
- **ORM:** Spring Data JPA
- **Build tool:** Maven or Gradle
- **Java version:** 17 or higher

---

3. Entity Modeling

3.1 User Entity

Attributes: - `id` (Long, auto-generated) - `lastName` (String, required) - `firstName` (String, required) - `email` (String, unique, valid email format) - `password` (String, required, minimum 8 characters) - `role` (Enum: ADMIN, ORGANIZER, CLIENT) - `registrationDate` (LocalDateTime) - `active` (Boolean) - `phone` (String, optional)

3.2 Event Entity

Attributes: - `id` (Long, auto-generated) - `title` (String, required, 5–100 characters) - `description` (String, max 1000 characters) - `category` (Enum: CONCERT, THEATRE, CONFERENCE, SPORT, OTHER) - `startDate` (LocalDateTime, required, must be in the future) - `endDate` (LocalDateTime, required, after startDate) - `venue` (String, required) - `city` (String, required) - `maxCapacity` (Integer, required, > 0) - `unitPrice` (Double, required, $\geq$ 0) - `imageUrl` (String, optional) - `organizer` (ManyToOne relationship with User) - `status` (Enum: DRAFT, PUBLISHED, CANCELLED, FINISHED) - `createdAt` (LocalDateTime) - `updatedAt` (LocalDateTime)

3.3 Reservation Entity

Attributes: - `id` (Long, auto-generated) - `user` (ManyToOne relationship with User) - `event` (ManyToOne relationship with Event) - `numberOfSeats` (Integer, required, > 0) - `totalAmount` (Double, automatically calculated) - `reservationDate` (LocalDateTime) - `status` (Enum: PENDING, CONFIRMED, CANCELLED) - `reservationCode` (String, unique, auto-generated, format: EVT-XXXXX) - `comment` (String, optional)

Relationships

- A user can create multiple events (organizer)
- A user can make multiple reservations
- An event can have multiple reservations
- Proper cascade and orphan removal management is required

---

4. Repository Layer

UserRepository

- Find user by email
- Find active users by role
- Check if an email already exists
- Search users by first or last name (case-insensitive)
- Count users by role

EventRepository

- Find events by category
- Find published events between two dates
- Find events by organizer and status
- Find available events (published and not finished)
- Count events by category
- Find events by venue or city
- Search events by title keyword
- Find events within a price range

ReservationRepository

- Find reservations by user
- Find reservations by event and status
- Calculate total reserved seats for an event
- Find reservation by code
- Find reservations between two dates
- Find confirmed reservations by user
- Calculate total amount spent by a user

---

5. Service Layer

UserService

- User registration (with validation and password hashing)
- Authentication (email/password verification)
- Profile update
- Password change
- Account activation/deactivation
- User statistics (events created, reservations, amount spent)
- User listing with filters

EventService

- Create event (ADMIN or ORGANIZER only)
- Update event (creator or ADMIN)
- Publish event (DRAFT → PUBLISHED)
- Cancel event (handle existing reservations)
- Delete event (only if no reservations)
- Advanced event search (multiple filters)
- Calculate available seats
- Retrieve popular events
- Organizer statistics
- Automatic detection of finished events

ReservationService

- Create reservation with checks:
- Seat availability
- Event validity (published & not finished)
- Max 10 seats
- Auto-generate reservation code
- Calculate total amount
- Confirm reservation
- Cancel reservation (refund rules)
- Retrieve user reservations
- Verify reservation by code
- Generate reservation summary
- Reservation statistics

---

6. Business Rules

- A reservation cannot exceed 10 seats
- Reservations can be cancelled up to 48h before the event
- Total reserved seats cannot exceed event capacity
- Finished events cannot be modified
- An event can only be published if all required fields are filled
- Reservation code must be unique and follow format EVT-XXXXX
- Total amount = number of seats × unit price

---

7. User Interface with Vaadin

Public Pages

- Login
- Registration
- Home page with featured events
- Event list with filters
- Event detail page

Client Interface

- Dashboard
- My reservations
- Reservation form
- Profile management

Organizer Interface

- Organizer dashboard

- My events
- Event creation/edit form
- Event reservations view

Admin Interface

- Admin dashboard
- User management
- Event management
- Reservation management

---

8. Security

- Spring Security (session-based)
- BCrypt password hashing
- Role-based access control
- Public vs protected routes
- Automatic redirection after login

---

9. Exception Handling & Validation

Custom Exceptions

- ResourceNotFoundException
- BadRequestException
- UnauthorizedException
- ForbiddenException
- ConflictException
- BusinessException

Validation

- Bean Validation annotations
- Vaadin Binder validation
- Custom validators (password strength, date checks)

---

10. H2 Configuration & Test Data

- Embedded H2 database
- H2 console enabled
- `data.sql` with:
- Users (ADMIN, ORGANIZERS, CLIENTS)
- Events (all categories, multiple statuses)

- Reservations (various states and seat counts)

---

11. Advanced Java Concepts

- Streams API (filtering, statistics)
- Optional for null safety
- Lambdas and functional interfaces
- Generics utilities
- Enums with business logic
- Design patterns: Singleton, Factory, Observer, Builder (optional)

---

12. Optional Features (Bonus)

- Advanced search with autocompletion
- Image upload for events
- Email notifications
- Charts and dashboards
- CSV/PDF export
- Google Maps integration
- Dark/light mode
- Real-time notifications
- Reviews and ratings
- Lazy loading pagination

---

13. Expected Deliverables

- Fully functional Maven/Gradle project
- Clean, structured, documented code
- README.md with setup and usage
- Database schema generated by Hibernate
- Final report (3–4 pages PDF):
- Architecture
- Technical choices
- Features
- Challenges & solutions
- Screenshots
- Future improvements

Good luck. XenPlan is not small — build it clean or it will break.