Event Planner App – Grails Design Document

Author: Kevin Farias
Project Type: Junior Developer Grails Technical Assessment
Framework: Grails 5.3.6
Language: Groovy
Purpose: Event and Attendee Management with RSVP Functionality

---

1. Overview

This web application allows users to:

- **Create**, **view**, and **manage** events.
- **RSVP** to events as attendees.
- **Differentiate** between upcoming and past events.
- **Search** for events and manage attendance caps.

The goal is to demonstrate proficiency in Grails domain modeling, controller/service layering, validation, and dynamic form rendering using GSP.

---

2. Domain **Model**

Event

Field	**Type**	**Constraints**
name	String	Required
location	String	Required
eventDate	Date	Required, must be in the future
description	Text	Optional
capacity	Integer	Optional (null = unlimited)
attendees	hasMany	Linked to Attendee (1-to-many)

Attendee

Field	**Type**	**Constraints**
fullName	String	Required
email	String	Required, must be valid email format
event	belongsTo	Associated Event

---

3. Architecture

Layers

- **Controller Layer:** Handles HTTP requests and flash messaging.
- **Service Layer:** Encapsulates validation, business rules, and persistence.
- **Domain Layer:** Enforces constraints, relationships, and core logic.
- **GSP Views:** HTML rendering using Bootstrap, conditional logic, and taglibs.

---

4. Features Implemented

Feature	Description
✅ Event CRUD	Create, read, update, and delete events.
✅ RSVP Form	Custom form with full name and email validation.
✅ Upcoming/Past Event Separation	Filtered views with date comparison logic.
✅ Attendee Count	Displays total and remaining seats per event.
✅ Input Validation	Domain-level rules prevent invalid form submissions.
✅ Delete Attendee	Admin feature to remove attendees with success message.
✅ Event Full Handling	Prevent RSVP if capacity is reached; shows friendly message.
✅ Fake Email Confirmation	Simulated email output via println() on RSVP or delete.
✅ Event Search	Filter events by name or location (case-insensitive).

---

5. Key Business Logic

Attendee RSVP

```
if (!event || event.isFull()) {
   return AppUtil.saveResponse(false, null)
}
```

- Attendee only allowed to RSVP to a future, non-full event.
- validate() ensures proper data entry before saving.

Attendee Count Display

```
(${ev.attendees?.size() ?: 0} / ${ev.capacity}) — ${remaining} seat(s) left
```

Email Format Validation

Handled via Grails domain constraints and GSP type="email" input field.

---

6. UI & UX Practices

- Responsive layout using **Bootstrap** grid.
- Buttons clearly labeled (“RSVP”, “Edit”, “Delete”).
- Radio buttons for event filter toggling (Upcoming / Past).
- Input feedback via HTML5 attributes + flash messages.
- Card-based layout for listing events.

---

7. Validation and Constraints

Type	Mechanism
Required	constraints { field blank: false }
Future Date	validator: { it > new Date() }
Email	email: true constraint
Capacity	Integer? capacity, logic in service

---

8. Project Structure (Core Files)

```
grails-app/
├── controllers/
|   └── AttendeeController.groovy
|   └── EventController.groovy
├── domain/
|   └── Attendee.groovy
|   └── Event.groovy
├── services/
|   └── AttendeeService.groovy
|   └── EventService.groovy
├── views/
|   └── event/_eventCard.gsp
|   └── event/index.gsp
```

```
|   └── attendee/_rsvpForm.gsp
|   └── shared/_messages.gsp
└── utils/
    └── AppUtil.groovy
```

---

9. Recommendations for Future Enhancements

- Add HTMX support for AJAX RSVP submission.
- Use a real mail service for email confirmations.
- Add pagination to attendees list.
- Export attendees as CSV or PDF.
- Add detailed error handling in UI, as it only displays a flash message
- Add more custom GSP or HTMX files for UI, as some of these files are the default generated by grails.
- Enable persistent data storage by switching from the in-memory H2 database to a production-grade database like PostgreSQL. This allows data to persist across application restarts and supports more advanced queries and scaling needs.
- Add unit and integration testing to ensure core functionality (like RSVP limits, validations, and service-layer logic) behaves as expected across all use cases.
- Implement admin authentication to restrict editing, deleting events, and managing attendees to authorized users only.