GitGrade Analysis Report

Repository: Kartikey1405/CampusConnect

Language: TypeScript

Stars: 0 | Forks: 0

Final Grade: 85/100

Executive Summary:

This project showcases a solid architectural foundation as a full-stack monorepo, leveraging a modern React + TypeScript frontend and a robust Spring Boot backend. The chosen technology stack is well-suited for a comprehensive event management system, indicating a strong understanding of current development practices. While the core logic appears to be adequately designed, the project's operational maturity and deployment consistency require significant improvements to move beyond its foundational state.

Improvement Roadmap:

1. Implement Monorepo Orchestration [Architecture]

Currently, the services are treated as independent silos. You need a `docker-compose.yml` at the project root to define and link the backend and frontend services, ensuring a unified development and deployment environment. This isn't optional for a monorepo; it's fundamental.

2. Containerize Frontend Application [Infrastructure]

The frontend, like the backend, needs its own `Dockerfile` to ensure environment consistency and simplify deployment. Don't rely on host-specific Node/Bun installations; containerize it properly with a multi-stage build for optimal image size and security.

3. Establish Comprehensive CI/CD [DevOps]

Where's your CI? Create a `.github/workflows/main.yml` (or similar) to automate builds, run tests (if you ever write any), and perform linting for both frontend and backend on every push. Manual verification is for hobby projects, not enterprise-level systems.

4. Centralize Environment Configuration [Security]

Managing sensitive credentials and varying configurations across local, dev, and production environments for two services is prone to error. Implement a robust strategy using externalized configuration (e.g., Docker secrets, Kubernetes secrets, or a dedicated configuration service for advanced setups) rather than scattered `.env` files or directly in `application.properties` for non-dev environments.

5. Introduce Database Migration Tooling [Database]

Relying solely on Spring Data JPA's DDL auto-generation in production is a recipe for disaster. Integrate a proper

database migration tool like Flyway or Liquibase to manage schema evolution across environments safely and systematically. You'll thank me when your database isn't a mess after the third feature release.

6. Standardize Frontend Package Manager [Development Workflow]

You have both `bun.lockb` and `package-lock.json` present. This is a clear sign of inconsistency and will lead to 'works on my machine' issues. Pick one - preferably Bun, given `bun.lockb` - and remove the other, ensuring all developers use the same package manager. Consistency is key, not an afterthought.

7. Document Backend API Contracts [Documentation]

Frontend developers shouldn't have to guess what your API endpoints expect or return. Integrate OpenAPI/Swagger (e.g., Springdoc OpenAPI) into your Spring Boot backend to automatically generate and host API documentation. This improves collaboration and reduces integration errors significantly.