BranDon Brown

COP 3060

# Individual Reflection

Our team built a full-stack campus resource management system with multi-level access. By intentionally switching from our usual roles, Cameron and I increased our overall output and improved the final product. He focused on the web interface and documentation, while I managed the backend and testing. Our complementary strengths helped us deliver a robust, extendable, and user-friendly system.

I handled most of the backend work, including the Spring Boot service layer, repository interactions, validation rules, and REST endpoints for Categories, Locations, and Resources. I configured the MySQL database and ensured clean JPA mappings, refined controller responses, optimized exception handling, and aligned CORS settings with our workflow. I also wrote unit and integration tests to maintain stability as features grew. While Cameron led the frontend, I assisted with Axios issues, form validation alignment, and API response checks. Documentation was shared by myself which contributed backend sections and technical reviews.

This project greatly strengthened my teamwork skills in a full-stack environment. Cameron and I divided responsibilities clearly, frontend and documentation for him, backend and testing for me, while staying flexible and trusting each other's strengths. Regular check-ins and open communication helped us integrate features smoothly, from database updates to frontend state changes. Through this collaboration, I gained confidence in backend architecture, deepened my knowledge of Spring Boot 3 and JPA, improved my API testing skills, and developed a stronger understanding of how frontend and backend must work together to create a seamless user experience.

Because our system stores campus resource data and user-generated entries, we discussed the ethical responsibility of handling information securely and transparently. Even though it doesn't include sensitive personal data, we still applied best practices such as validation, structured error handling, and careful data access. We also reflected on the use of AI tools for documentation and code scaffolding. While AI helped clarify concepts and suggest structure, we ensured that all core logic, especially database operations and API design, was fully understood and implemented by us.

Our system includes user login and full CRUD functionality for Categories, Locations, and Resources. Once authenticated, users can organize campus resources, manage data, and update entries easily. We also integrated the OpenWeatherMap API to display real-time weather conditions for campus areas, adding helpful context, such as clear sky or fog alerts, that may affect resource availability and support future notification features. Frontend and backend communication uses Axios with standard RESTful endpoints (GET, POST, PUT, DELETE) for all entities, keeping interactions predictable. Axios also retrieves external weather data directly from OpenWeatherMap.

Our system uses a clean full-stack structure, a React frontend for routing, forms, and validation, a Spring Boot backend with controllers, services, repositories, and strong validation, and a MySQL database storing categories, locations, and resources. This modular design enables smooth parallel development.

A key challenge was inconsistent validation between the backend and early frontend forms, causing rejected submissions and confusing errors. I standardized backend validation and improved error messaging, and Cameron updated the frontend to match these rules. This alignment eliminated invalid requests and made the system far more reliable.

I learned the importance of planning early, documenting endpoints clearly, and maintaining organized code to support collaboration. Looking ahead, we aim to add a notification system that alerts users to important campus events or weather conditions affecting resources. Overall, this project strengthened my commitment to clean, maintainable code, and continuous improvement.