Final Report - NU Corporate Web Portal

Group members: Assylzhan Omarov, Marlen Nuraly, Alisher Kunbolsyn, Dinmukhamet Batmanov, Nazira Ibrasheva

Advisers: Askar Boranbayev, Almas Amirbekov

Executive Summary

The NU Corporate Web Portal project addresses the issue of having a centralized, university-level solution for handling institutional data, events, and services for students, teachers, and administrative staff members at Nazarbayev University. The initial objective was to develop an extensible and secure full-stack application that simplifies event planning, communication, and access to resources based on the user's role.

The system includes news moderation, file storage, venue booking, and responsive design with Spring Boot and PostgreSQL as its base, and deployment on Docker via Render. The user interface is role-based, serving students, DSS staff, teachers, and general staff with similarly focused access and features.

Introduction

Nazarbayev University previously lacked a unified platform for extracurricular activities. This generated decentralized workflows and additional load for university employees, poor transparency and accessibility for students. This project responds to that shortcoming by generating a usable, responsive and centralized portal with role-specific functionality. Students are able to submit news and arrange events; DSS is able to manage them; all users are able to book venues and make payments, see event calendar, campus news, a university phonebook, and static informative content through the Infocenter.

Background and Related Work

University organizations leverage learning management systems (i.e., Moodle, Canvas) or intranet sites with low functionality. They are not normally provisioned with built-in support for automatic event moderation, real-time conflict checking of venues, or QR systems campus-wide. Inspired by users' experiences on Google Calendar, Eventbrite, and admin inner dashboards, the framework was intended to give an integrated experience to end users while also allowing modular administrative control of staff as well as DSS. Compared to the platforms currently available, our solution integrates communication, booking, users' identity, and announcements from institutions into one UI. Spring Boot, JWT, and PostgreSQL were employed because of their scalability, security, as well as a large community base. Third-party QR APIs were used to avoid reimplementing common utilities and focus on integration and domain workflows.

Project Approach

All team members collaborated using Jira, GitHub, and weekly meetings. Each module was tested in isolation and integrated into the Render deployment pipeline. We implemented a responsive frontend using React.

Backend:

- Spring Boot REST API with 3-tiered architecture
- PostgreSQL for relational data (hosted on Render)
- JWT-based authentication via Spring Security
- Docker-based deployment
- Azure Blob Storage to store files

Frontend:

- Responsive UI using component-based design
- Role-specific navigation for students, faculty, DSS, and staff

- Pages include: event calendar, news feed, my page (personal account info), venue reservation planner, suggest event / suggest news (students), event & news moderation (DSS), phonebook (with contact-saving), static infocenter for useful resources

Project Execution

Assylzhan Omarov - project manager and backend developer.

Marlen Nuraly - backend developer.

Alisher Kunbolsyn - frontend developer.

Dinmukhamet Batmanov - frontend developer.

Nazira Ibrasheva - UI/UX designer and front-end developer.

During the first semester, our focus was on planning, design, and role allocation. We first decomposed the problem space and derived key system requirements from our project proposal. The team collaborated to make UML diagrams, identify entities, and map user stories and use cases for the different roles, such as students, DSS, faculty, and staff. At an early stage, the workload was divided: some members worked on backend architecture and security, and others established the groundwork for frontend design, refining and polishing mockup designs for better planning. This stage was crucial to keep all members updated about the scope of the system as well as divide the workload evenly among the members.

The second semester marked the beginning of actual development. We started with setting up the Spring Boot project structure and integrating JWT-based authentication for role-based access. Implementation of entities and business logic began in parallel with database modeling. From a front-end perspective, we employed React due to its component-based design, which supports reusable and modular UI components to promote easier development of a dynamic and responsive user interface. The ecosystem of React

allowed us to achieve efficient state management, role-based routing, and instant UI updates that were critical for functionalities. To connect the React frontend and the Spring Boot backend, we employed RESTful API communication by using fetch call, authorizing using JWT tokens that had been stored in localStorage. This allowed for a stable and secure transmission of data from frontend to backend, where checks for user roles and data to be displayed were directly tied to API returns. In order to upload, save and display images and files, we used Azure Blob Storage. To deploy, we hosted the frontend on GitHub Pages, which gave us a free, stable solution to serve the static React build with version control and continuous integration through GitHub Actions. The backend on Render was linked via HTTPS endpoints, enabling real-time interactions such as form submission and data fetch to be tested. Besides, Swagger UI (on the backend) also played a critical role in frontend development. It provided a live, interactive API documentation that allowed the frontend team to navigate endpoints, understand required request formats, and test responses without backend code access. This improved development speed and accuracy between teams significantly and allowed consistent integration between modules.

We encountered several challenges, like database schema mapping issues, especially with Optional types, file naming conflicts during the Docker builds, and difficulties connecting to the PostgreSQL database on Render. We tackled these issues through team discussions, searching online documentation, and trying different testing approaches. For example, we dealt with QR code integration problems by enhancing our service layer checks and using a dependable public API.

Evaluation

To find out how much effect the NU Corporate Web Portal has had in addressing the problems as presented in the introduction, we conducted a combination of realistic use-case

demonstrations and constrained user testing with our potential users that represent the many roles within our system. Rather than relying on internal development testing, we had five volunteer testers act like students, DSS staff, and faculty, and undertake a representative set of tasks. These included: submitting an event, checking available venues and detecting time conflicts, scanning a QR code, accessing the My Page dashboard for personalized data, submitting a news item and moderating it. Each participant rated their experience for usability and intuitiveness:

- 4 out of 5 users rated the event planning and submission feature as easy to use
- All testers were able to scan the QR code successfully and verified that the system blocked duplicate scans
- Students noted the usefulness of the sneak peek at venue availability, stating that it prevented double booking
- A recommendation for improvement was to create a more graphically interactive event calendar for easier browsing.

Conclusion and possible future work

The NU Corporate Web Portal successfully addressed the essential need for a centralized, role-based communication and service system for Nazarbayev University. The project's key contributions include event planning, venue booking, news sharing, file management. Through collaboration and problem-solving, the team delivered a computing-based solution that can scale and be used in the real-world environment. Future enhancements could focus on integrating developing analytics dashboards for administrators, improving the calendar’s visual UI/UX, and adding features such as multi-language support and accessibility improvements.

References

Spring Boot Documentation. (n.d.). *Spring Boot*. Retrieved from https://spring.io/projects/spring-boot

PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. Retrieved from https://www.postgresql.org/docs/

Render Documentation. (n.d.). *Deploying Web Services*. Retrieved from https://render.com/docs

goQR API. (n.d.). *QR Code Generator API*. Retrieved from https://goqr.me/api/

GitHub. (n.d.). *senior-project-java-backend Repository*. GitHub. Retrieved from https://github.com/Merkel00/senior-project-java-backend

Docker. (n.d.). *Docker Documentation*. Retrieved from https://docs.docker.com/