CSCI 409 Senior Project II - Final Project Report

Web Application for Alumni

Group Members: Abylaikhan Anapiya, Raiymbek Meirambek, Aibek Zhakhan

Advisors: Askar Boranbayev, Kuanysh Yersakhanov

Date: April 25, 2025

1. Executive Summary

The challenge of managing sizable alumni datasets and improving communication between the university and its graduates is addressed by the Web Application for Alumni, which was created for Nazarbayev University. With features like token-based authentication through Laravel Sanctum, complete CRUD operations, and automated/manual backups, the project sought to provide a scalable, safe, and user-friendly platform. React with Inertia.js for the frontend, PostgreSQL for the database, PHP/Laravel with Laravel Breeze for authentication scaffolding, and an admin panel for data management are all features of the system, which is hosted on an Nginx server. Alumni can access news, events, user stories, and personal information through the alumni user panel.

With Jira for task management and GitHub for version control the development process was iterative and employed Agile methodologies. Performance testing, usability testing with students and alumnis, security audits, and backup testing to guarantee adherence to data protection guidelines were all part of the evaluation process. The outcomes validate a strong solution that satisfies all requirements and greatly enhances alumni engagement and data management. This computing-based solution offers stakeholders quantifiable value while being in line with institutional requirements and contemporary web development practices.

2. Introduction

A centralised, effective system is required to manage records and encourage continued engagement due to the alumni network's explosive growth at Nazarbayev University. Today its

number is around 10339, and the number still keeps growing by thousands annually, yet no centralised platform to connect those Alumni students after graduation. However, there could be number of advantages from it for Universities, for instance:

- Alumni Donations and Fundraising Opportunities may bring some external funds;
- Maintaining lifelong engagement can bring more opportunities for faculty / Alumni;
- Successful Alumni stories may drive others and in general good marketing tool;

The shortcomings of manual processes and legacy systems led to data inconsistencies, inefficiencies, and limited interaction capabilities. The Web Application for Alumni addresses these problems by providing a secure, scalable web platform that streamlines administrative tasks and enhances alumni connectivity with interactive features.

The project is important because it can improve data management, ensure security, and strengthen university-alumni ties, which will benefit stakeholders like the IT Department, Alumni Office, and graduates. The solution is a multi-layer web application that meets functional and non-functional needs like load performance and scalability.

This report is structured as follows: Section 3 reviews related work, Section 4 details the project approach, Section 5 describes execution, Section 6 presents evaluation results, Section 7 discusses conclusions and future work, and Section 8 lists references.

3. Background and Related Work

Alumni management systems are essential for universities to maintain relationships with graduates, track professional achievements, and facilitate networking. We have reviewed different existing solutions for reference:

- **UniAlumni**: A Themeforest template solution offering robust alumni management and scalable user engagement system, which is probably most suitable for NU, but its functionality extends our needs, so that some of its features would be unused. In general, great reference for review and idea.
- **AlumniQ**: A cloud-based platform with features like event management and user profiles but lacking deep integration with university systems.
- **Alumnius (open-source solution)**: These provide basic CRUD functionality but often lack scalability, security features, and support for complex requirements like bulk data imports. Good idea for visualisation of Alumni data for its users, but we've decided not to display any personal data of our platform users.

Our approach leverages PHP/Laravel for its rapid development capabilities and robust security features, PostgreSQL for efficient data management, and React for a dynamic, responsive frontend. Unlike UniAlmuni, our solution is cost-effective and tailored to Nazarbayev University's needs. Compared to AlumniQ, it includes real-time synchronization with events for Alumni. The use of Laravel Sanctum for authentication and role-based access control addresses security gaps in open-source alternatives, justifying our methodology as a balanced, institution-specific solution. And one of the main advantages of our system is its robustness and scalability, so that in future more features for Alumni engagement could be added if required.

4. Project Approach

4.1 System Architecture

The application employs a multi-layer architecture:

- **Frontend**: React, integrated with Inertia.js, provides a responsive, component-based single-page application interface, enabling dynamic user interactions and seamless navigation.
- **Backend**: Laravel, enhanced with Laravel Breeze and Sanctum, handles authentication and data processing. Inertia.js bridges the backend and frontend, allowing Laravel controllers to serve data directly to React components.
- **Database**: PostgreSQL manages alumni records, supporting complex queries and scalability for large datasets.
- **Server**: Nginx hosts the application, ensuring reliable performance and SSL encryption for secure communication.

Backend-Frontend Integration:

- Laravel Breeze provides authentication scaffolding (login, registration, password reset), generating secure backend routes and controllers. It integrates with Inertia.js to deliver authentication data to React components without traditional API endpoints.
- Inertia.js enables a single-page application experience by allowing Laravel to render initial pages server-side and handle client-side navigation. For example, when an admin performs a CRUD operation, a React component sends a request via Inertia.js, the Laravel controller processes it, and Inertia.js updates the component with new data, avoiding full page reloads.
- React components (e.g., AdminPanel.jsx, UserProfile.jsx) render dynamic UI elements, such as forms for CRUD operations or news feeds, using data provided by Inertia.js. This setup ensures a fast, modern user experience with minimal API complexity.

4.2 Key Features

- **Admin Panel**: Supports full CRUD operations, data filtering, and search. Administrators can manage roles and manage events, website contents.
- **Alumni User Panel**: Allows alumni to view and update personal information, access user stories, and stay informed about news and events.
- **Authentication**: Sanctum-based login with session expiry and password reset functionality. Role-based access control restricts actions based on user type (Admin, Alumni).
- **Data Backup**: The system implements both manual and automated backup mechanisms to ensure data integrity and support disaster recovery. Manual backups are performed using PostgreSQL's pg_dump tool, allowing administrators to create database snapshots during periodic checkups or before major updates. These snapshots are exported as .sql files and stored securely on a university-managed file server. Automated backups are facilitated by Laravel's task scheduler, which runs a custom Artisan command (backup:database) daily to execute pg_dump and save compressed backups to a designated server directory. The spatie/laravel-backup package was integrated to enhance automation, enabling compressed backups and optional cloud storage integration. Restoration is supported using psql or pg_restore to recover data from backups in case of system failure.
- **Integration**: The system ensures seamless integration between the Laravel backend and React frontend via Inertia.js, providing a cohesive user experience without reliance on external university systems.

4.3 Tools and Third-Party Components

- **Laravel Packages**: Laravel Breeze for authentication scaffolding, Laravel Sanctum for token-based authentication, spatie/laravel-backup for automated backups, Inertia.js for backend-frontend integration.
- **React Libraries**: React Router (via Inertia.js) for navigation, Material-UI for UI components.
- **Development Tools**: Jira for sprint management, GitHub for version control.
- **Third-Party Integration**: PostgreSQL drivers for Laravel, Nginx modules for SSL configuration.

4.4 Use Cases

- **UC1: Admin Data Management**: Administrator logs in, performs CRUD operations, and views filtered data.
- **UC2: Alumni Profile Access**: Alumni logs in, updates personal information, and views news/events.
- **UC3: Backup and Restoration**: Administrator triggers manual backup; restores data using psql in case of failure.

4.5 Team Roles

- **Raiymbek Meirambek**: Frontend development (React), UI/UX design, and integration with backend APIs.
- **Aibek Zhakhan**: Backend development (Laravel), API design, and deployment on Nginx.

- **Abylaikhan Anapiya**: Project management, database design and management (PostgreSQL), system integration.

The team collaborated effectively, using Agile sprints to align development with stakeholder feedback and project milestones.

5. Project Execution

5.1 Timeline and Milestones

The project spanned Fall 2024 and Spring 2025:

- **Fall 2024**:
    - Set up development environment (initially Spring, later Laravel with Breeze and Sanctum, PostgreSQL, React with Inertia.js, Nginx).
    - Designed system architecture and database schema.
- **Spring 2025**:
    - Implemented basic CRUD operations and token-based authentication using Laravel Breeze and Sanctum.
    - Developed advanced features (user stories, news/events, backup mechanisms).
    - Integrated React frontend with Laravel backend via Inertia.js.
    - Conducted performance optimization, security enhancements, and testing.
    - Deployed the application and gathered user feedback.

5.2 Design Decisions

- **Laravel Breeze with Sanctum and Inertia.js**: Chosen for rapid authentication setup and seamless backend-frontend integration, reducing API complexity.
- **React over Angular**: Selected for its flexibility and ecosystem, suitable for rapid UI development with Inertia.js.
- **PostgreSQL**: Preferred for its performance with large datasets and support for advanced querying.

5.3 Challenges and Solutions

- **Challenge**: Initial use of Spring Boot for backend development. The team spent time exploring Spring Boot, but its complex configuration requirements and steep learning curve slowed progress, risking project delays given the tight timeline and the team's limited experience with Java.
    - **Solution**: Switched to Laravel, which offered a simpler setup with tools like Breeze and Sanctum, leveraging the team's PHP expertise. This allowed rapid development of authentication and CRUD functionality, recovering lost time and aligning with project deadlines.
- **Challenge**: Ensuring smooth Inertia.js integration with React and Laravel.
    - **Solution**: Standardized Inertia.js responses and used React's state management to handle dynamic updates efficiently.
- **Challenge**: Configuring Laravel Sanctum for secure token-based authentication.
    - **Solution**: Combined Breeze's authentication scaffolding with Sanctum's token management, using middleware to enforce role-based access control.

5.4 Team Collaboration

Responsibilities were divided based on expertise, with regular reviews in Jira to track progress. Weekly meetings with team members ensured our project development process' alignment with its requirements. Collaborative problem-solving, such as pair programming for complex API calls and backend/frontend integrations easied development process and enhanced code quality.

5.5 What Went Right/Wrong

- **Right**: Agile methodology enabled iterative improvements, and stakeholder feedback refined features like the alumni user panel.
- **Wrong**:
    - The team wasted time exploring Spring Boot as the backend framework early in Fall 2024. Spring Boot's complex configuration and steep learning curve slowed progress and proved incompatible with the project's tight timeline and the team's familiarity with PHP. Switching to Laravel, with its simpler setup and rapid development tools like Breeze and Sanctum, allowed the team to recover lost time and meet deadlines.
    - Initial underestimation of Inertia.js setup complexity delayed Sprint 4. This was mitigated by reallocating resources and extending testing.

6. Evaluation

6.1 Evaluation Methods

The system was evaluated to validate its effectiveness in solving the alumni management problem:

- **Performance Testing**: Tested with moderate user loads to ensure response times in seconds for CRUD operations and searches. While not tested with 1,000 concurrent users, PostgreSQL's scalability suggests it can handle such loads.
- **Usability Testing**: Conducted with 30 students using questionnaires to assess ease of use and feature accessibility.
- **Security Testing**: Penetration testing and vulnerability scans to ensure compliance with data protection standards.
- **Functional Testing**: Verified all features (CRUD, authentication, etc.) against requirements using automated unit tests (PHPUnit) and manual testing.

6.2 Results

- **Performance**: CRUD operations and search queries completed within 2 seconds under tested loads. PostgreSQL's design supports scalability for higher loads (e.g., 1,000 users), though not explicitly tested.
- **Usability**: More than 90% of students found the user panel easy to navigate. Feedback highlighted the news/events section as particularly engaging.
- **Security**: No critical vulnerabilities were found; SSL encryption and JWT ensured secure communication and access control.
- **Functionality**: All required features were successfully implemented, with 100% test case coverage for critical components.

6.3 Analysis

The evaluation confirms that the system meets its objectives of efficient data management and enhanced engagement. Performance metrics validate reliability, while usability feedback underscores user satisfaction. Security measures, including Sanctum's token-based authentication, align with institutional policies, ensuring data protection. The backup mechanism, combining manual checkups with automated scheduling, ensures data integrity and supports disaster recovery, as evidenced by successful restoration tests. The inclusion of user stories and events significantly increased alumni interaction, as noted in user feedback.

7. Conclusion and Future Work

The Web Application for Alumni successfully delivers a scalable, secure, and user-friendly platform for Nazarbayev University. It streamlines administrative tasks, ensures data integrity, and fosters alumni engagement through organized events / news, stories. Key contributions include a robust multi-layer architecture with seamless backend-frontend integration via Inertia.js and reliable backup mechanisms.

Future enhancements could include:

- Implementing two-factor authentication (2FA) for enhanced security.
- Implementing bulk data import/export for enhanced administrative efficiency.
- Integrating with external university systems for data sharing, if required in the future.
- Implementing Alumni programs (mentorship, career advancements, etc.)
- Verified Alumni event organization alongside admin
- Directory of verified Alumni for other employers to review

The project's phased approach and rigorous evaluation ensure its readiness for production use, with potential for further scalability and feature expansion.

8. References

- UniAlumni – University Alumni Template (2023). https://themeforest.net/item/unialumni-university-alumni-html-template/22050026
- AlumniQ - Efficient, effective alumni engagement systems. https://www.alumniq.com
- Alumni US – all best US alumni graduates directory. https://alumnius.net
- From first graduates to leaders: 15 years of NU (2025). https://nu.edu.kz/news/from-first-graduates-to-leaders-15-years-of-nu
- Laravel Documentation. (2024). https://laravel.com/docs
- React Documentation. (2024). https://react.dev/
- PostgreSQL Documentation. (2024). https://www.postgresql.org/docs
- Spatie Laravel Backup Documentation. (2024). Available: https://spatie.be/docs/laravel-backup
- Inertia.js Documentation. (2024). Available: https://inertiajs.com/
- Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine.
- OWASP. (2024). Web Security Testing Guide. https://owasp.org/www-project-web-security-testing-guide/