Senior Project II Final Report - Spring 2025

Title of the project:	"NUverse" - NU Corporate web-portal
Team Members:	Ablaikhan Orynkul, Batizhamal Rauiya, Bekzhan Abdullayev, Yesfir Yermekbayeva
Project Advisor/Co-Advisors	Askar Boranbayev

Executive summary

The main goal of our "NUVerse" project is to create a secure web portal with user-friendly interface for the Nazarbayev University community. The system addresses the limitations of the existing digital platform by providing dynamic news and event modules, a searchable phonebook, and role-based content access.

The project was developed with initially defined tech stack: Spring Boot for backend part, React.js for the frontend services and PostgreSQL as a database system for data management. We used Agile methodology for work flow control as it lets us divide tasks into sprints and complete them step by step.

This semester, we implemented new functionality including event calendar, phonebook service and birthday widget. It completes our project functionality complementing the functionality we completed last semester - authentication and authorization, news publication and home page with widgets. The project aligns closely with computing-based solution principles—starting from identifying real needs, designing scalable architecture, to delivering a secure and tested MVP that integrates with university infrastructure.

Introduction

It is obvious modern universities mostly rely on their web platforms for event coordination, resource management, etc. At Nazarbayev University, we have the current system (my.nu.edu.kz) which provides

all basic needs for students and staff but still has some limitations. These limitations lead to the need for a new system.

The “NUVerse” project was initiated to solve this issue and fulfill this need by developing a centralized and secure digital platform. The main advantage of our system is enhanced user experience with features like news publishing, event scheduling with calendar, phonebook services, multilingual support and role-based content access. These main features aim to support students in their daily tasks and be more united with the NU Community.

The motivation and requirements of this project were developed by stakeholder feedback and institutional needs for a reliable web platform that aligns with university’s policy and goals. Our portal is designed to ensure security and compliance with national data protection laws, with multilingual accessibility in English, Kazakh, and Russian.

The report begins by presenting relevant background and related work, followed by a project approach which includes a detailed description of the solution, third-party components and team work of our group. The next part is project execution which summarizes the project's progress over the past two semesters with a focus on challenges, solution and teamwork decisions. This is followed by an evaluation section about effectiveness assessment of the developed system and, finally, report concludes with key findings and possible future work.

Background and related work

- Discuss prior research, existing solutions, and related work.
- Compare and contrast various approaches to justify your selected methodology.
- Demonstrate an understanding of computing-based solutions by analyzing existing literature.

As the digital ecosystem within higher education evolves, the need for centralized platforms that support seamless, secure communication across student and staff communities has become increasingly

pronounced. The NUverse project emerged from this demand at Nazarbayev University, aiming to deliver a comprehensive and user-centric web portal.

a. Existing University Portals and Limitations

Nazarbayev University currently uses an internal system (my.nu.edu.kz) for academic and administrative services. While functional, the system lacks extensibility and interactivity. It does not support real-time news publishing, self-service profile management, or event coordination features—all of which are essential for a modern university environment.

In comparison, systems like Blackboard or Canvas offer some of these features but are generally academic-focused, not community-oriented. NUverse is designed to bridge that gap, providing a unified digital space tailored to both operational and social aspects of university life.

b. Authentication and Access Control

Modern identity management systems increasingly rely on **Single Sign-On (SSO)** for seamless and secure access. Microsoft Azure Active Directory has emerged as a standard in academic institutions. Its integration allows institutions to enforce security policies and manage user roles centrally. NUverse adopts MS Azure SSO to enable role-based access control (RBAC), ensuring granular visibility and control over features based on a user's role (student, staff, or administrator).

c. Content and Community Management Systems

Various CMS platforms (like Drupal or WordPress-based intranet solutions) support news and file publishing, but lack the academic context, integration with institutional databases (like 1C), and real-time notifications needed in a university setting. NUverse incorporates event management, multilingual content publishing, and document uploads, combining CMS capabilities with institutional alignment.

d. Security, Localization, and Compliance

In line with literature on secure web development, NUverse implements **SSL encryption**, **session timeout control**, and **compliance with Kazakhstan's data protection regulations**. Moreover, research

underscores the importance of **cultural and linguistic localization**. NUverse ensures full support for English, Kazakh, and Russian, not just in interface elements, but also in data input/output and content management.

e. **Methodological Justification**

Our team followed an iterative Agile methodology, aligning well with best practices in full-stack development projects. This approach, validated by case studies from industry and academia, allowed the team to incorporate feedback quickly and maintain project momentum. Tools such as Jira, GitHub, and Figma were employed to ensure seamless collaboration and traceability.

Project approach

- Provide a detailed description of the solution, including software/hardware architecture, algorithms, workflows, roles, features, tools, and use case diagrams.
- Explain any third-party components that the group did not implement themselves used and their integration.
  Show how the project team functioned effectively to develop a computing-based solution.

Our solution is to build a centralized, multilingual platform that will be used by Nazarbayev University staff and students. This app has features such as news publication, profile management, calendar events, a university phonebook, and a birthday widget. The emphasis was on security, scalability and accessibility.

Software Architecture:

- **Frontend**
  The Next.js was used for server-side rendering and routing functionality which provides both fast performance and search engine optimized pages. For rendering we utilized React Server Components and Server actions. Our decision for design styling went to Tailwind CSS for providing a utility-first framework that results in clean, responsive and consistent UI. Form validation and type safety were implemented with assistance of TypeScript and Zod library. The

application also supports language switching between Kazakh, Russian, and English through the implementation of react-i18next as an internationalization method.

- **Backend**

  Java Spring Boot was the main tool to develop all backend services which offer RESTful APIs for frontend communication. User profile management runs securely on the backend system together with role-based access control features which protect data privacy and authorize proper access between various user types.

- **Database**:

  A PostgreSQL database is used to store user-related metadata, logs, and content management data. The database implements encryption methods for sensitive data while role-based access controls restrict sensitive data access permissions.

Workflows:

1. **User authentication:**

   Users sign in using email and password. Based on their roles, they are redirected to their dashboards.

2. **Dashboard interaction:**

   In the dashboard tab, they will see the latest news published on the platform, this week's calendar that shows appointed meetings and events, and 'today's birthdays' component with the list of users who have birthdays.

3. **News section:**

   In the news section, the list of the news will be available, providing the data about category and publication date of each publication. Moreover, the user can publish the new one by tapping on the "Publish news" button. It will show the form with such input fields as title, category, content and attachments.

4. **Calendar section:**

   This section is for displaying the events that the user has for the whole month or different periods. Creating event functionality is also available in this page.

5. **Phonebook section:**

   Phonebook section is mainly for admin for changing the user data and also for the students to contact professors, university staff, and students that changed their data to be visible.

6. **Users section:**

   In this page of our web application, the user can see all the users and search. Furthermore, individual profiles can be seen by clicking on the needed user card. All the detailed information is provided there. Admins can use user addition functionality and change the data of the users.

Third-Party Components and Integrations:

- **NextAuth.js**: Used for secure authentication, integrated using modern server actions and middleware.
- **Tailwind CSS + Hero Icons**: Utility-first CSS framework paired with accessible, responsive icons for building modern UIs.
- **React-i18next**
- **Zod**: Provides type-safe schema validation for form inputs and server-side logic.
- **React-quill**: Integrated as a rich text editor for content creation within the platform.
- **Lombok:** reduces boilerplate in Java classes.
- **SpringDoc:** generates OpenAPI documentation for RESTful APIs using Spring Boot annotations.
- **Liquibase:** manages versioned database migrations and schema changes across environments.
- **JJWT (Java JWT):** used for securely creating and validating JSON Web Tokens for authentication and authorization flows.
- **ModelMapper:** simplifies mapping between DTOs and entity objects, streamlining data transfer between layers.

Team Collaboration:

The team functioned efficiently and productively. We used agile methodology for development, and divided all the time we had for two-week sprints to understand how much progress was made during that time.

The development team consisted of four members, each contributing to different aspects of the project based on their expertise:

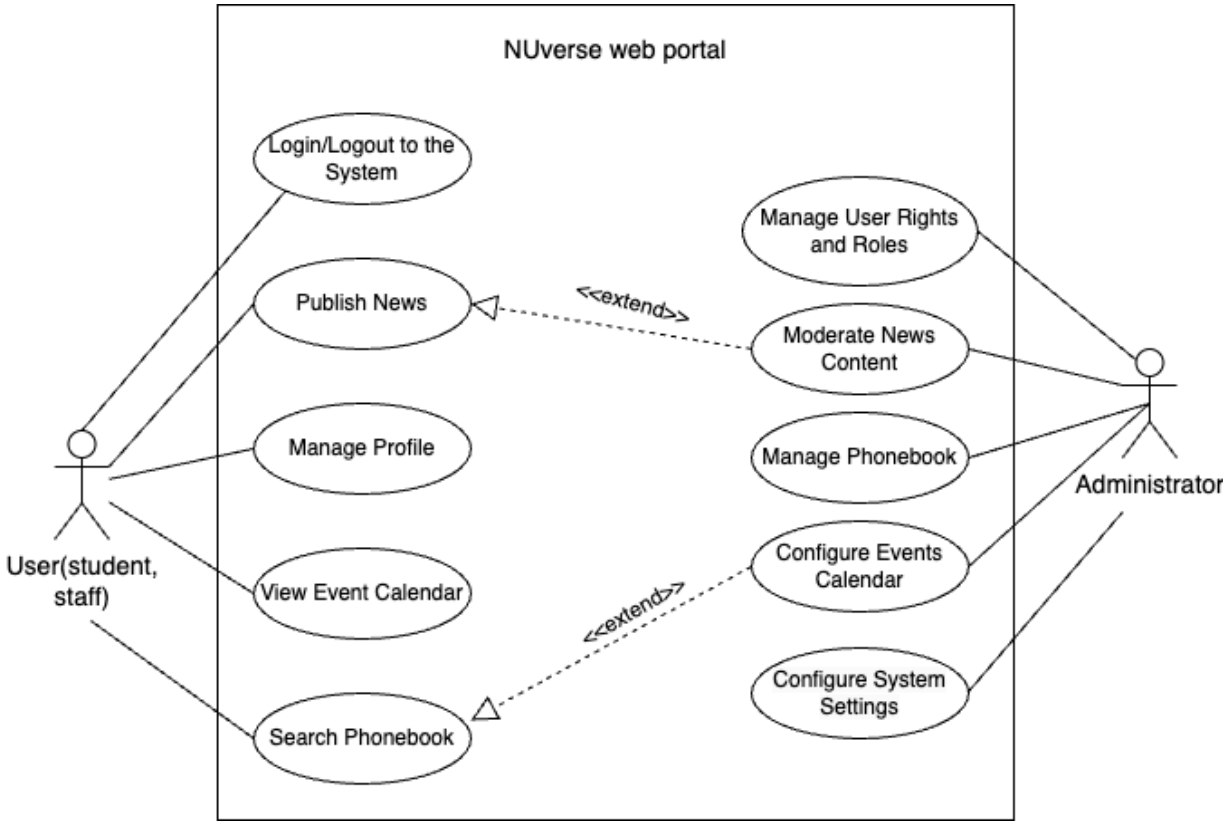
- **Batizhamal**: Focused on frontend development, implementing core UI pages. Handled the deployment process, ensuring the application was properly built, tested, and deployed in a production-ready environment.
- **Bekzhan**: Focused on frontend development, implementing core UI components, state handling, and integrating frontend logic with backend services.
- **Ablaikhan**: Was responsible for backend development using Java Spring Boot, building RESTful APIs, handling business logic, and integrating with the PostgreSQL database.
- **Yesfir**: Acted primarily as the project manager, overseeing task distribution, sprint planning, and team coordination. Additionally, Yesfir contributed to development work when needed, assisting across both frontend and backend tasks.

Tools Used:

- Git + GitHub as the main version control system and code reviews
- Figma for UI/UX designs
- Jira was used for task tracking
- Telegram/Discord for communication
- Postman for API testing

Code Reviews and Pair Programming: Conducted weekly to ensure code quality and knowledge sharing.

Use case diagram:



Project execution

- Describe what happened over the course of the last two semesters, including design decisions made, changes in the project, what went wrong/right, how you dealt with the problems you encountered, etc.
  Highlight teamwork aspects, such as how responsibilities were divided, collaborative problem-solving strategies, and leadership roles taken during the project

The NUverse project was executed across two semesters, moving from concept and design to implementation and testing. Each phase was marked by careful planning, continuous integration, and collaborative problem-solving.

The Fall semester laid the foundation for the system architecture. Following stakeholder interviews and documentation analysis, we translated requirements into technical components.

Key Milestones:

- Defined the tech stack: **Java Spring Boot, PostgreSQL, React**
- Developed initial **REST APIs**, schema models, and UI mockups
- Built MVP with:
    - MS Azure SSO authentication
    - Role-based permissions (student, staff, admin)
    - Home dashboard with widgets (news, birthdays, calendar)
    - News publishing form and preview

Technical Challenges:

- Designing RBAC with fine-grained permission control
- Backend delays due to data model complexity (resolved using mock databases via **Neon**)
- Deciding localization strategy: chose hybrid model (i18n on frontend, backend error localization)

In Spring semester, building on the MVP, the team implemented advanced features and began preparing the system for pilot deployment.

Newly Implemented Modules:

- **Multilingual interface** with live language switching
- **Phonebook directory** with access filtering by role
- **Event calendar** with resource booking and approval workflows
- **Admin panel** for user and access management

Integration & Optimization:

- Established RESTful data exchange with 1C and HR systems
- Refined backend endpoints and frontend responsiveness
- Implemented real-time notifications and session timeout logic

- Finalized CI/CD workflow for deployment

Step-by-Step Execution Plan

Step 1: Project Planning and Requirement Analysis

- Created a kickoff document, defined the project scope, and reviewed stakeholder needs.
- Resulted in initial and refined requirements documents capturing both functional and non-functional requirements.
- Tools Used**:** Google Docs, Jira for sprint planning and requirements tracking.

Step 2: Technology Stack Evaluation and Setup

- **Frontend:** React.js with i18n for localization and responsive design.
- **Backend:** Java Spring Boot with Spring Security for access control.
- **Database:** PostgreSQL, mocked early on using Neon.
- **Tools Used:** Figma, Postman, pgAdmin, GitHub, Prometheus, Grafana, Vercel.

Step 3: Authentication and Authorization Implementation

- Integrated MS Azure SSO.
- Created a robust role and permission system on backend.
- Developed a login UI that dynamically adjusted views based on roles (student, staff, admin).
- Enforced secure session handling and HTTPS.

Step 4: UI/UX Design and Dashboard Development

- Designed home page with three widgets: news feed, upcoming events (calendar), and today's birthdays.
- Used mocked data for initial development with a plan to connect APIs later.

Step 5: News Module

- Created a dynamic form for news publishing.

- Supported attachments: docx, xlsx, pptx, pdf, and image files.
- Backend supported full CRUD operations for news.

Step 6: Localization Strategy and Integration

- Resolved internal debate on text storage with a hybrid approach.
- Frontend used i18n (React), backend provided localized error messages.
- Supported English, Kazakh, and Russian with real-time switching.

Step 7: Modular Backend Development

- Implemented modules for authentication, news, events, and profile management.
- REST APIs were built and tested using Postman.
- Token-based authentication was established via Spring Security.

Step 8: Testing and Mock Data Integration

- Used Neon to simulate data during early development.
- React components tested using mocked JSON data.
- Monitored backend performance using Prometheus Grafana.

Step 9: Role-Based Feature Display and Admin Panel Planning

- UI rendered different features and views based on user roles.
- Initial admin panel planning began, aimed at managing users and published content.

Step 10: Team Collaboration and Agile Workflow

- Followed 2-week sprint cycles using Jira.
- Communicated daily via Telegram and held weekly Google Meet calls.
- GitHub used for code management, PR reviews, and version control.

Step 11: Challenges and Course Corrections

- Faced delays in backend development due to complex database design—resolved with mock tools.
- Localization issue solved with a hybrid frontend/backend approach.
- Ensured compliance with Kazakhstan's data protection regulations.

Step 12: Final Development

- Finalized Event Calendar, Phonebook, Notifications, and Multilingual Interface.

Step 13: System Integration

- Integrated with 1C and HR systems via REST APIs for user data sync and updates.
- Tested MS Azure SSO in live university network environments.

Step 14: Security Enhancements

- Enforced password policies, session expiration, role-based access, and full SSL encryption.
- Implemented access logs and compliance checks for legal requirements.

Step 15: Testing and QA

- Conducted extensive functional testing, user acceptance testing (15 users), and load testing for 1,000+ concurrent users.
- Passed accessibility audit aligned with WCAG 2.1 AA guidelines.

Step 16: Deployment and Rollout

- CI/CD implemented via GitHub Actions.
- Frontend deployed on Vercel; backend containerized and deployed to university infrastructure.
- Soft-launched with administrators, followed by full rollout after stability confirmation.
- Monitoring set up using Prometheus Grafana.

Step 17: Documentation

- Compiled technical documentation and user manuals.

During the development process, we made several key findings:

- Problem-solving through mock tools (e.g., mocked REST APIs with Postman, mock data with Neon) allowed parallel development
- Early stakeholder alignment helped avoid misinterpretation of access control policies
- Agile cycles with 2-week sprints allowed rapid iteration and feedback incorporation
- Cross-functional collaboration ensured no component was built in isolation

Evaluation

To determine whether the NUverse project effectively addressed the problems outlined in the introduction we conducted a comprehensive evaluation process beyond standard bug detection.

Our evaluation methodology:

- Manual UI/UX Testing:

  We interacted with the platform in a deployed environment, navigating key features such as news publication, event scheduling, and phonebook access. UI/UX usability was assessed through observation focusing on intuitiveness and responsiveness.

  Moreover, we conducted manual API testing through the Postman tool to ensure that business processes functioned correctly. This was done to ensure that the system's logic aligned with project requirements. As a result the system remained stable up to 1,200 concurrent users, with average response time under 500ms.

- Unit Testing & Load Testing:

  Automated unit tests were executed on backend services to ensure correctness and reliability of individual components. Additionally, load testing simulated over 1,000 concurrent users to evaluate database and server performance.

Conclusion and possible future work

To conclude, overall key contributions include a functional role-based access system, dynamic modules for news, events, phonebook services, and personalized dashboards and a scalable architecture ensuring performance under high user loads. While NUverse project achieved its core objectives, several areas offer opportunities for future development include mobile application development, an advanced analytics dashboard, AI-powered features like chatbot integration for quick FAQs, notifications system, and integration with university services.

References

[1] PostgreSQL Global Development Group, "PostgreSQL documentation." [Online]. Available: https://www.postgresql.org/docs/. [Accessed: Nov. 17, 2024].

[2] Meta, "React documentation." [Online]. Available: https://react.dev/. [Accessed: Nov. 17, 2024].

[3] MDN Web Docs, "JavaScript documentation." Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed: Nov. 17, 2024].

[4] Open Web Application Security Project (OWASP), "OWASP secure coding practices quick reference guide." [Online]. Available: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/. [Accessed: Nov. 17, 2024].

[5] Oracle, "Java Spring Boot documentation." [Online]. Available: https://spring.io/projects/spring-boot. [Accessed: Nov. 17, 2024].

[6] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media, 2017.

[7] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.

[8] S. Stefanov, *React Up & Running: Building Web Applications*. O'Reilly Media, 2016.

[9] C. Walls, *Spring in Action*, 5th ed. Manning Publications, 2018.