Final Report: RA Reports

Senior Project

Adviser: Askar Boranbayev

Group 21:

Zhuldyzai Bermishtai

Aruzhan Kuzenbayeva

Magzhan Khorshat

Aniar Kaldybaiuly

Yeskendir Ybyray

Introduction

Problem description and motivation

Currently, at Nazarbayev University, the process of managing the research activities of Research Assistants (RAs), Project Administrators (PAs) and documenting the progress of the researches has been unstructured and conducted informally mainly through email messages and Google Forms updated manually. This decentralized method is also flawed in the following ways: missing or delayed reports, no easy way of identifying the status of approvals, unequal formatting, and no history of modifications. These inefficiencies hinder workflow, generate confusion and wasted time within various faculties and departments. The establishment of the RA Reports project was undertaken to meet these challenges with the objective of creating a specialized website for Nazarbayev University. This system will enhance the submission, tracking and approval of RA reports and it will offer safe, centralized and role based solutions. The platform provides specific services for two different types of users: Research Assistants, Project Administrators who manage research projects, and additional IT specialist, that is, technical admin who creates profiles for users and manages technical aspects of the system.

Solution

In order to overcome these challenges, our team designed a web-based application, namely RA Reports which is tailored for the reporting requirements of Nazarbayev University. The system provides a role-based dashboard to RAs, PAs for report submission, approval tracking, and status update. The reports are connected to the specific contracts of the projects and divided according to periods which are defined in advance to ensure that all the data that is submitted corresponds to contractual requirements. The platform also has features such as form generation, data validation, revision management, and comments. Any user actions such as submission, approval or requesting a revision will be stamped with time making it possible to determine compliance for an extended period. Notifications are sent in real time and the information is available in Kazakh, Russian and English to accommodate the conglomerate of institutions.

Report organization

- Background and related work reviews prior systems and research on research assistantship workflows and computing-based solutions
- Project approach outlines the technical methodology, architecture, and component-level design of the system

- Project execution presents the semester-wise development process, team roles, and the challenges encountered
- Evaluation discusses how the system was tested, validated, and reviewed by users to ensure its effectiveness
- Conclusion and future work summarizes key outcomes and outlines possible directions for extension or institutional adoption
- References provides full academic citations of the sources referenced throughout the report

Background and related work

Before moving on to computing-based systems, it is necessary to examine the general scientific environment within which the research assistants are placed. Based on the discussion of Yaya and Baskan (2013), RAs are exposed to administrative burden, role confusion, and reporting irregularity that may cause stress and organisational dysfunction. McDaniel (2015) also notes that RA job description includes data handling, reporting, and project coordination, but notes that these tasks are typically not accompanied by standardized technological tools. These challenges explain the need for the development of systems such as RA Reports in order to minimize such causes and enhance the clarification of the workflow to enhance the RAs, PIs, and PAs understanding.

Among the most relevant systems proposed in the literature, the most similar to RA Reports is the Research Assistant Management System (RAMS) developed by Amiruddin et al. (2019). RAMS was designed to address crucial issues on managing RA documentation, payment procedure and project issues in Malaysian universities. The authors also pointed out that paper-based submittals and isolated spreadsheets have not solved problems and created new ones, such as inaccuracy in working hour calculations, non-transparency of approvers, and a high level of burden, all of which can be seen in Nazarbayev University. RAMS maintained project-specific formatted forms for reporting and had automated ways of forwarding submissions to the administrators. In our system, we extended this core mechanism by providing dynamic forms filled with contract metadata that include project identification number, reporting period, and deliverables. As a result of focusing on RAMS in the development of our system, the principles of standardization, role differentiation, and reduced reliance on manual processes were reflected in the backend validation of data and the frontend work. Also, RAMS introduced multi-layered approval for different types of academic and financial approval. Although RA Reports does not handle payments, it has a

two-level approval that reflects NU's actual process: academic approval by PIs and checks for administration and contracts by PAs. It also means splitting the approval process more clearly and is closer to the actual structure of the university's governance than is the single-approval approach. In addition, RAMS was developed based on the Rapid Application Development (RAD) model that focuses on releasing products in phases, creating prototypes, and getting feedback frequently. Our own project adapted this approach to an agile form and implemented it with the help of sprint planning in Notion, the use of GitHub for cooperation, and Figma for the evaluation of UI/UX. These tools helped the development team to be adaptive throughout the process of development and to include feedback from NU staff as well as research groups at a crucial stage.

Yang et al. (2005) introduced a new concept of Research Assistant System (RAS) where the authors conceptualize research administration as a field of knowledge management. Unlike other more structured work-oriented systems, RAS was intended to help organizations learn, so that research institutions could build up and refer to institutional memory as well as feedback, questions, and patterns of operation. These included revision history, documentation, and shared issues list that allowed the administrators and researchers to reduce the likelihood of repeating the same mistakes in their future decisions. One of the fundamental principles of RAS is the concept of organizational memory, which goes beyond simple report submission and contains the context of administrative work. This theoretical model is different from other process-automation tools such as RAMS because it integrates reflection, and history into the system design.

Although RA Reports does not incorporate most of the principles of the organizational knowledge repository, some of them can be seen in the implementation of our system. Namely, we added the revision tracking and comment history to keep the context of approval feedback between RAs, PIs, and PAs. It promotes openness, fosters institutional responsibility and sets a foundation for future enhancements given the trends of reporting. RAS also pointed out the need for natural user interfaces and work flows tailored to a role - ideas that shaped the distinction of the RA Reports' dashboards. While user roles are not only permission based but cognitively focused in both systems, users of the interface are made aware of their responsibilities to perform tasks based on the institutional hierarchy of the system. All in all, although, despite its limitations, RA Reports does not encompass the entire range of knowledge management examined in RAS, it employs selectively some of its principles – the principles of revision history and user role – and turns into an efficient reporting tool that complies with the organizational culture at Nazarbayev University. In

addition to specific RA systems, recommendations for the design of institutional systems are available from literature related to the university digitization process.

With regard to the software methodology, Abbas et al. (2020) present the use of agile frameworks within university software projects where the requirements change in the middle of the development cycle because of academic terms, staff changes, or user identification. In order to implement this recommendation, we divided the development into sprints, and checked the designs and scope with the team frequently. As for the architecture, we adhered to such principles as component isolation and modular service design. Authorization (Spring Security + JWT), data validation, approval routing and notification services are developed as decoupled modules to the main application. This is not only beneficial in terms of fast development but also scalability: the new functionality such as analytics or mobile notifications can be introduced without redesigning the existing solution. React front-end development also made it possible to render different roles with independent dashboards and PostgreSQL allowed to create structured databases with relations between project, report, and user where different relations are important to track through time. It is a highly responsive and easy-to-maintain application that complies with the academic structure and the modern software engineering practices.

Project approach

System architecture

For the purpose of implementing the RA Reports System, this application can be divided into three-tier web-based architecture which is the frontend layer, the backend service layer and the relational database layer. This architecture allows for building blocks of functionality and easy access control and also clean lines of responsibility between the user interface, the business logic and the storage backend. Figure 1 shows a high-level architecture diagram of the system components and how they are related.

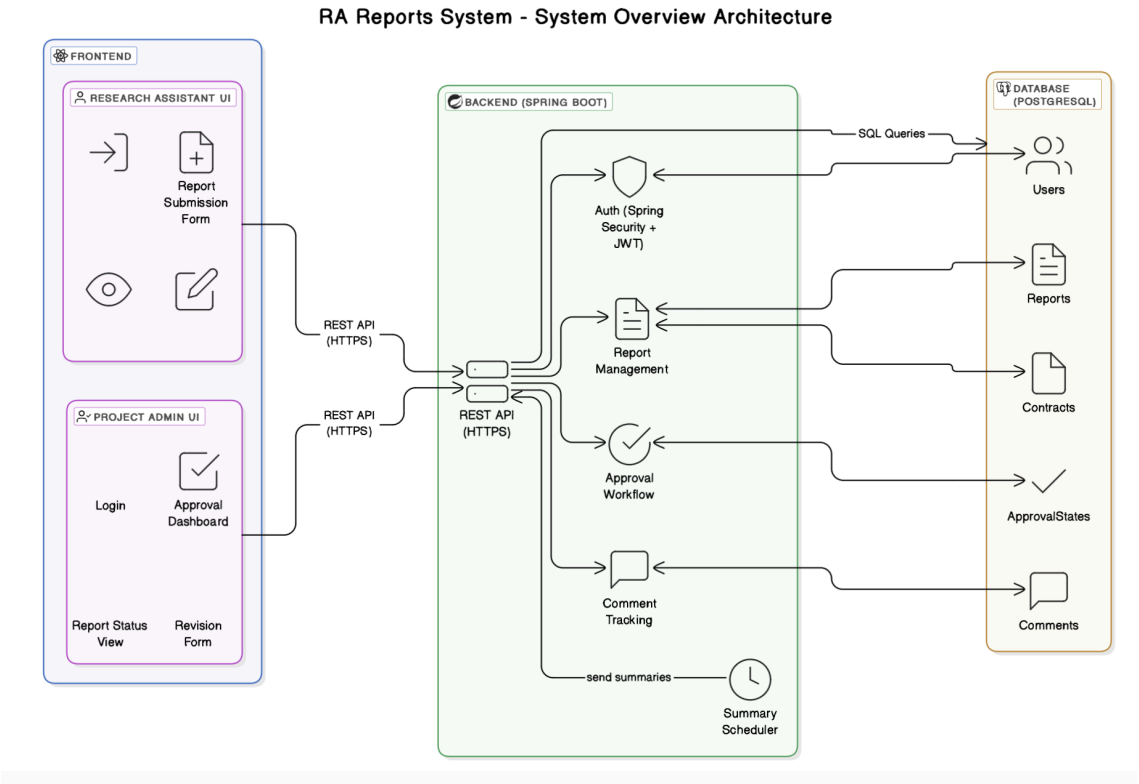


Figure 1. System architecture diagram

For the frontend layer, the user interface is built using React.js and is separated into two role-specific environments:

- Research assistant UI: login interface, report submission form (linked to specific contract), status tracker and revision interface
- Project administrator UI: login interface, approval dashboard, feedback and comment form, report status tracking interface

The backend service, developed using Spring Boot, handles the application's core logic and routes frontend requests to appropriate services. It contains the following key components:

- Authentication & authorization module: uses Spring Security and JWT for login, session control, and enforcing role-based access.
- Report management module: handles report creation, editing, submission, and contract-based validation.
- Approval workflow module: enables PAs to approve or reject reports, track approval stages, and initiate revision cycles.
- Comment tracking module: allows PAs to leave structured feedback on each report and store revision histories.

- REST API layer: exposes endpoints that the frontend consumes, such as /login, /submitReport, /getReportsByAdmin, /approveReport.
- Summary scheduler: reserved for potential future development to generate report summaries.

The backend interacts with a structured relational database managed in PostgreSQL. The schema includes five key tables: users, contracts, reports, approval status, comments

Functionalities and user roles

The RA Reports System consists of two user roles, namely Research Assistants (RAs) and Project Administrators (PAs), and their activities are guided by the following workflow demonstrated at Figure 2. RAs log in, complete and submit reports tied to specific contracts, and then track submission status. If a report is rejected, the RA receives in-system feedback and resubmits after editing. PAs log in to view all submitted reports linked to their assigned contracts. They review, approve, or reject reports, add comments when revisions are needed, and manage the report history. Notifications are displayed in the RA's dashboard to prompt action.

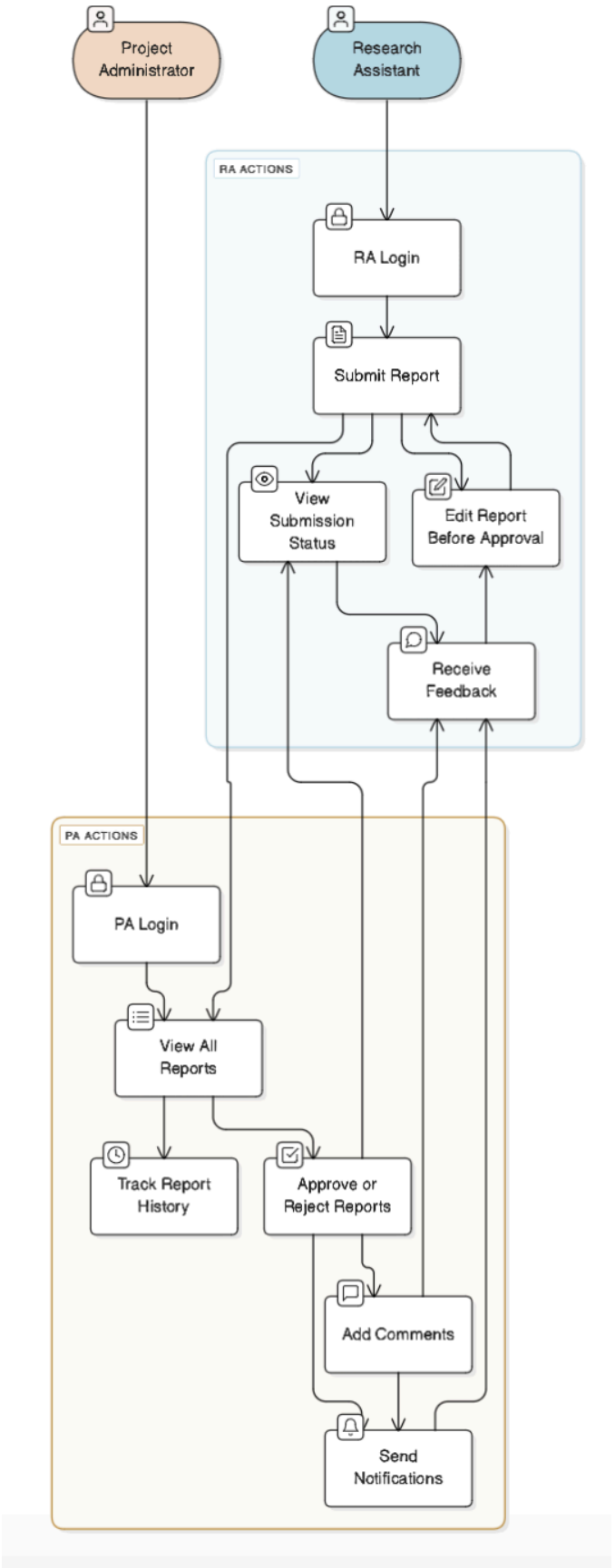


Figure 2. Flow chart diagram

Database design and entity relationships

RA Reports System is based on the PostgreSQL relational database that consists of five base objects that enable role-specific functionality, reporting flow, and data tracking. The ERD of the entities mentioned is depicted in the following UML Entity- Relationship Diagram (Figure 3).

- User: this table maintains all the users of the system who have authenticated themselves as either a Research Assistant or a Project Administrator. Every user has a role that defines his or her access rights and the specific views of the system that the user has.
- Contracts: they can be defined as project based work relations where the work is to be done under the supervision of a particular PA. Every contract is associated with one administrator and may include several reports provided by RAs at different times.
- Reports: this table is the main table that collects all the submitted research assistant reports. Every report is associated with a particular contract, produced by the RA and revised by the supervising PA. The report's status changes its status depending on the stage of the approval process it is in.
- Approval stages: the table records the status change of each report at a given period. It allows tracking of changes in status over time like "Rejected", "Pending", "Request for revision", "Approved".
- Comments: include suggestions that the PA makes to the RA about certain reports. They enable multiple rounds of review and enhance the organization of the communication process within the platform and link feedback to the content and versions.

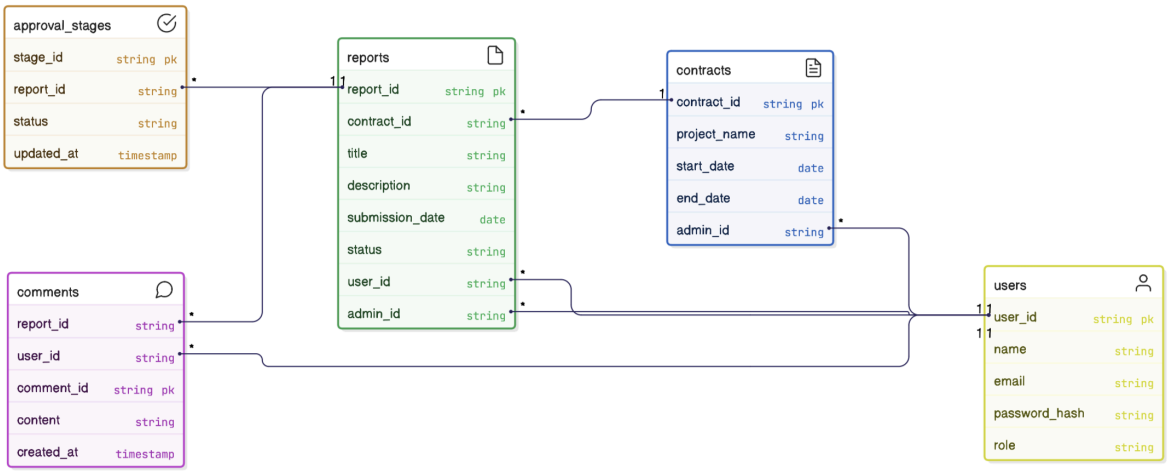


Figure 3. Database design and entity relationships

Frameworks and third-party tools utilized:

- React.js – used when developing the front-end of the application and the user interface of both RA and PA dashboards
- Spring Boot – the main backend framework that was used to develop REST APIs and business logics
- Spring Security + JWT – implements authentication and token-based authorization as well as role authorization (RA / PA)
- PostgreSQL – serves as the system for managing the relations between data and organizing structured data storage
- Figma – during the design phase, it is utilized in prototyping as well as in visualization of user interfaces and the flow
- GitHub – served as the version control system and team collaboration platform during development.
- Notion – used for project planning, sprint tracking, internal team documentation

Project execution

Development phases

The initial phase (first semester) was devoted to problem analysis, requirement gathering, and system design. During this stage, the team:

- Conducted needs analysis based on research assistant reporting practices at Nazarbayev University.
- Designed the system architecture and drafted UI mockups using Figma.
- Defined user roles and core functionalities (login, report submission, approval).
- Planned database schema and relationships.

The second semester was dedicated to full system development and testing. Key steps included:

- Building the frontend in React and structuring role-based components.
- Implementing the backend with Spring Boot, including REST APIs, authentication (Spring Security + JWT), and service modules (report handling, approval, comments).
- Developing and deploying the PostgreSQL database schema.
- Conducting internal testing and validation of workflows, including role-based access control, status changes, and revision cycles.

Key functionalities developed

The login page (Figure 4) is the first interface that is accessed by all the users in the system and contains two fields for the email and password. After a successful login, the user is redirected to different dashboards depending on the role assigned to him: either RA or PA. Session management is implemented with the help of JWT tokens for the purpose of security.

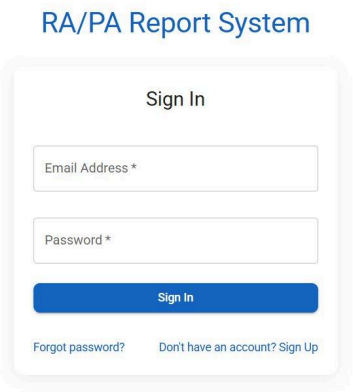


Figure 4. Login

Report submission page (Figure 5): RAs can write a new report by choosing an active contract and inputting structured data for title, content, and time period. After the report is submitted, the status of that report automatically changes to “Submitted” and it enters the approval process.

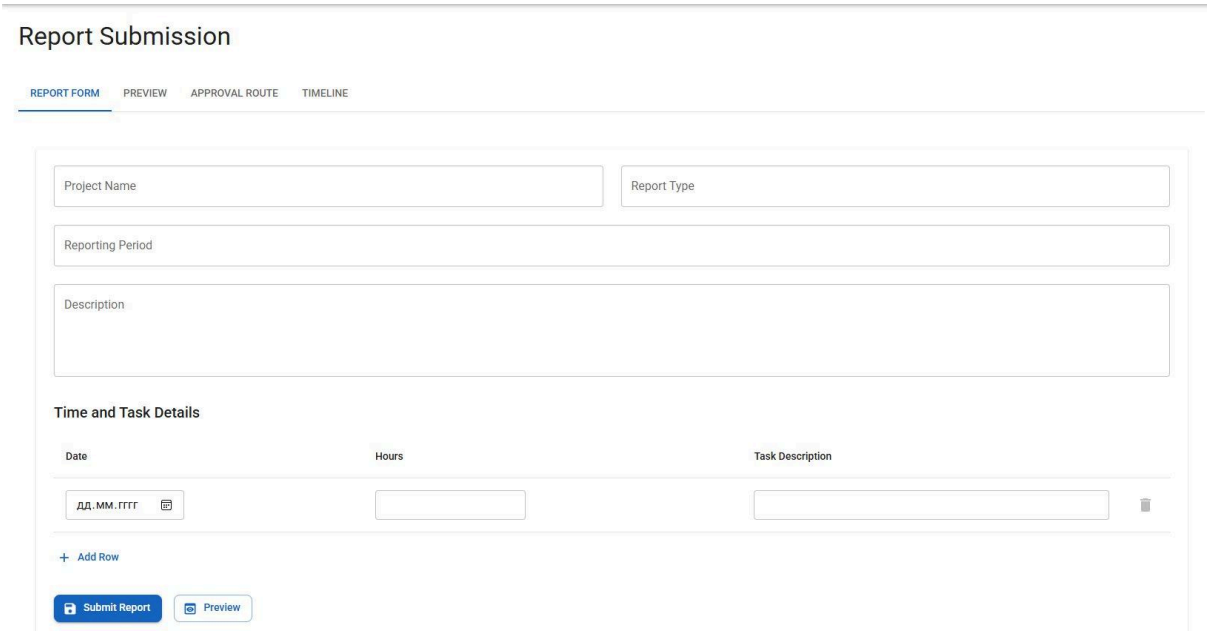


Figure 5. Report submission

Technical admin dashboard (Figure 6): technical admin can register new users (RA, PA) as shown in Figure 7, and manage their profiles. In order to authorize research projects into the system, admin can create contracts (Figure 8).

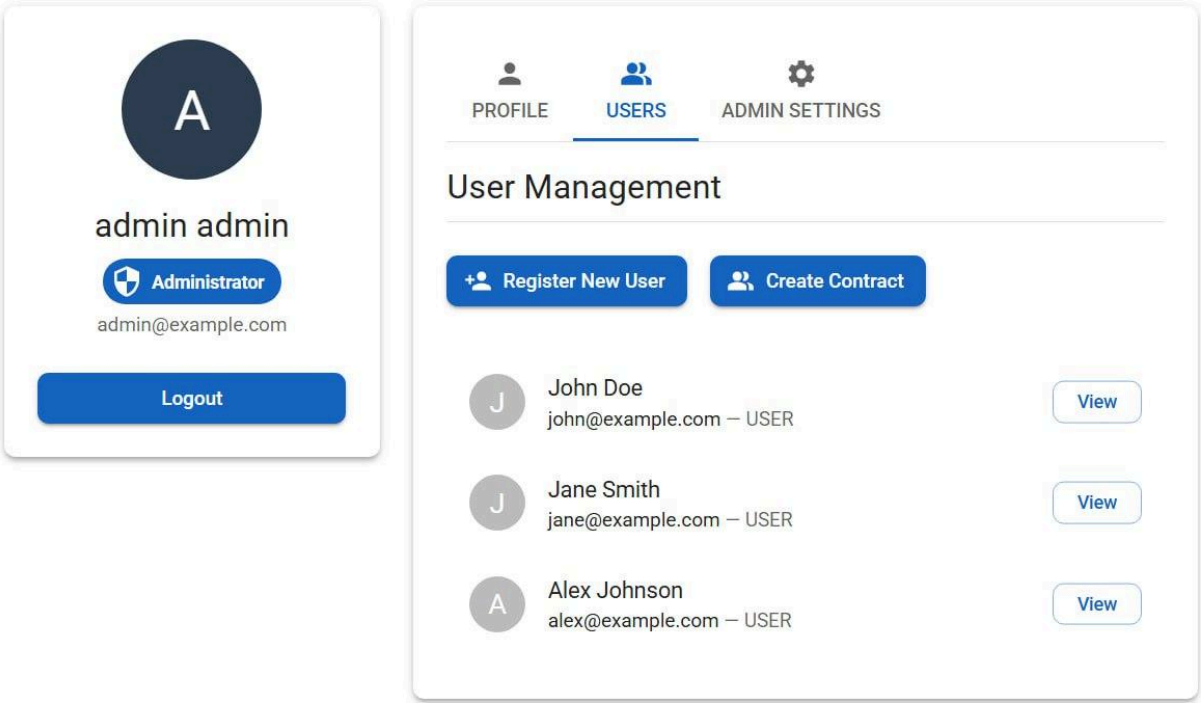


Figure 6. Technical admin dashboard

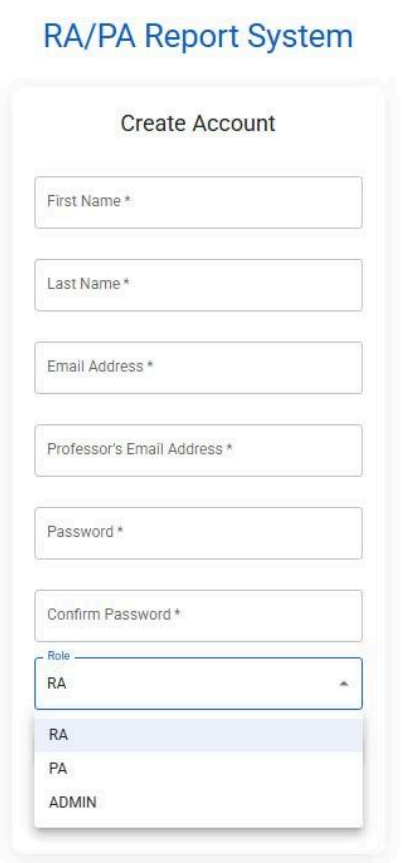


Figure 7. User account creation

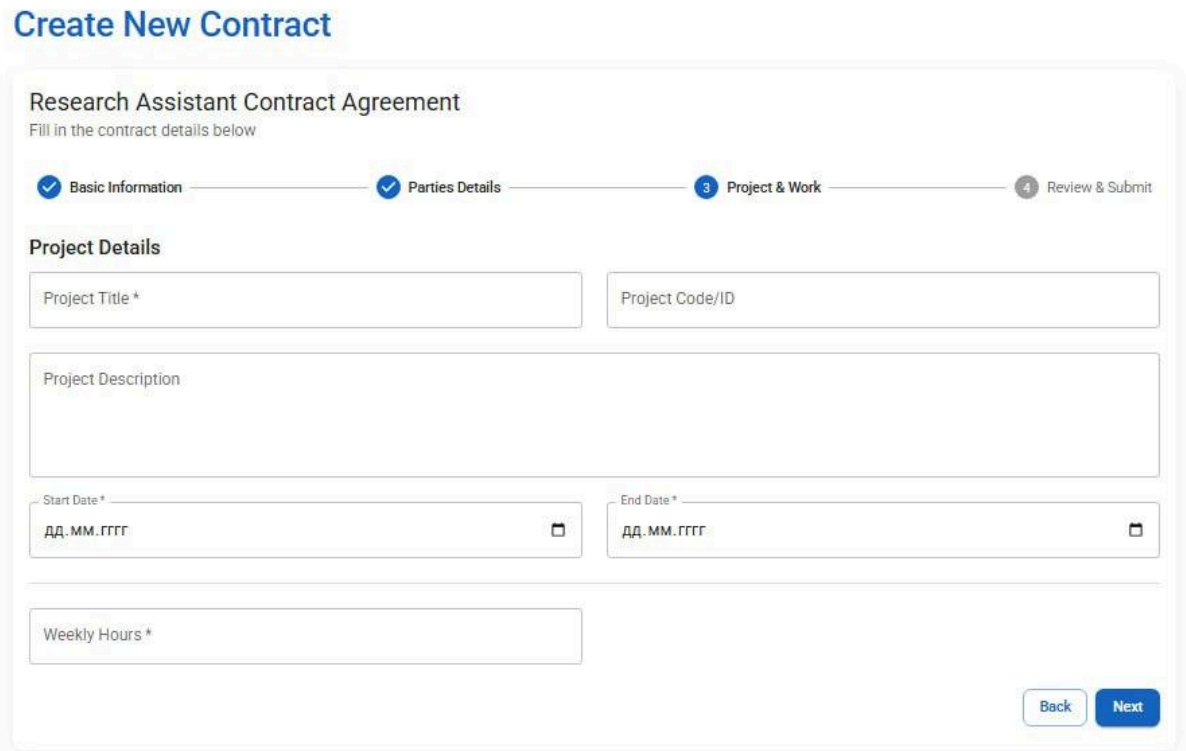


Figure 8. Contact creation

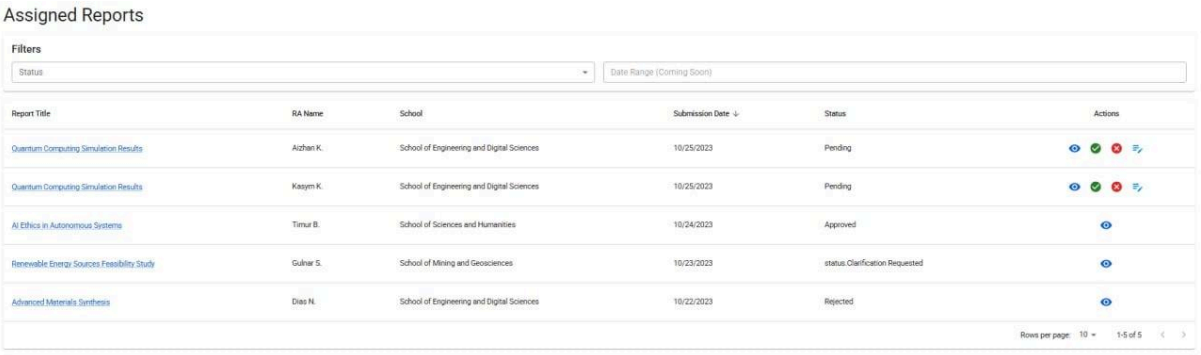


Figure 9. Project Administrator's panel

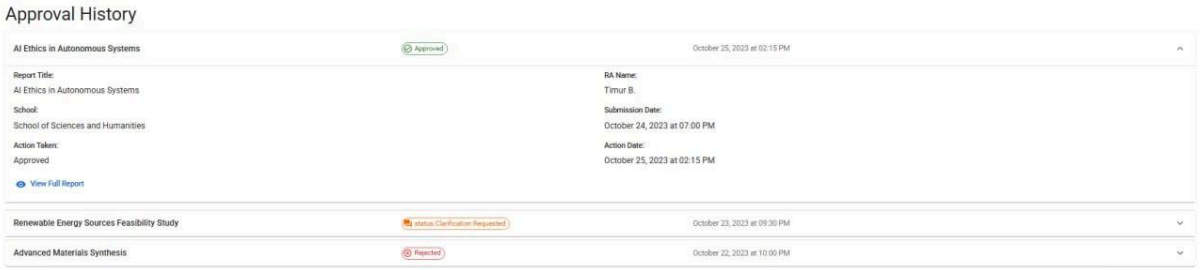


Figure 10. Approval of reports from PA side

Role distribution

UI and role-based dashboards and front-end development were performed by Aruzhan Kuzenbayeva and Zhuldyzai Bermishtai. The work on backend as well as database design and integration was carried out by Aniar Kaldybaiuly, Yeskendir Ybyray, Magzhan Khorshat. The team members worked closely with each other and all the work was done on GitHub and Notion where the work distribution was also planned, the progress was reported, and the integration of each part of the system was also synchronized.

Changes in the project

- Frontend framework: Angular -> React. Originally Angular was chosen for the frontend but was changed to React.js at the end of the first semester. React provided more flexibility, easier learning curve, and a non-invasive state management for individual components — making it easier to create role-based dashboards and forms. It helped to speed up the iterations in the frontend and at the same time do this without much compromising the functionality.
- Language. The original proposal was made with an idea of dynamic language support for Kazakh, Russian and English. However, it was practically impossible to achieve and test a complete internationalization during the project period. Therefore, the final system was released in English only, although the backend and the frontend were developed in a way that would allow for localization in the future.

- Removing external system dependencies. Integration with document management systems (1C & EDMS) were discussed but excluded as they may involve external APIs and data security issues, as well as institutional access.
- Internal notifications instead of email. Email based notifications were thought of at the beginning, but were later downgraded and replaced with the status bar and real time feedback on the dashboard. The inclusion of the email functionality would have added more complications like the configuration of SMTP, management of the credentials, and third-party authentication layers that were not relevant for the pilot.

Challenges and how we dealt with them

One major issue was the ability to track the dependencies and versions of the different files that were being created simultaneously. Since, there were cross-developer modifications done on the frontend and backend at the same time, merge conflicts were often encountered particularly when working with models or routes. Such conflicts sometimes lead to system-breaking bugs that would slow down work being done. To address this issue, the team changed their Git workflow to use new branches for each task and code reviews before merging to the master branch. This also helped in avoiding cases of conflict and made it easier for the different components to be incorporated.

Another problem was that the levels of testing were not standardized. During local testing, team members used different operating systems and database configurations, and this affected the consistency of the systems, for instance, data was not saved as it should or authentication was not successful at some instances. To mitigate this, the team decided to have common development tools, added Docker to PostgreSQL, and created a shared space to put the environment setup guide. This helped in making certain that all the members were testing in a similar environment.

Also, an important problem was to make progress on a regular basis, since team members had high workloads in other courses and graduate applications. Due to this, the team employed weekly meetings and set milestones to ensure that progress was achieved on a weekly basis. If someone was overloaded with other studies, tasks were reassigned so that work could always go on and everyone would have a fair share of the workload.

Evaluation

To assess the efficiency of the RA Reports System, functional testing as well as the feedback of users of the system were gathered. The objective was to find out whether or not the developed system was sufficient in solving the problem of unstructured and unorganized

reporting between the Research Assistants and Project Administrators. The assessment was done based on test case verification and real users' feedback collected through a set of questionnaires.

Functional testing

The system was thoroughly tested on the manual level to guarantee that all the essential features operated correctly. Specific test cases for the RA and PA, as well as technical IT administrator user roles were created, which include login authentication, submitting a report, approving and rejecting a report, and revising a report. These test cases ensured the users could perform their respective roles without submitting erroneous results and the integrity and security of the data during the operations. Details on test cases and result status are given in Table 1.

Test case	Role	Expected result	Test status
Login with valid credentials	RA/PA	User is authenticated and redirected to dashboard	Pass
Login with invalid credentials	RA/PA	Error message displayed, access denied	Pass
Submit report with valid data	RA	Report is saved with status "Pending"	Pass
Submit report with missing required fields	RA	Submission is blocked, error shown	Pass
Display of report status (Pending, Rejected.)	RA/PA	Status updates reflect correctly	Pass
Create contract with valid data	Technical Admin	PDF file of contract is generated	Pass
View list of submitted reports	RA	Only own reports are shown with correct statuses	Pass

View assigned reports list	PA	Reports tied to assigned contracts are shown	Pass
Approve a report	PA	Report status updates to “Approved”	Pass

Table 1. Test cases

User feedback

In order to assess the effectiveness of the RA Reports System, the team administered questionnaires to the RAs and PAs after the implementation of the system: 14 RA responses and 8 PA responses, which were related to the core functions and the satisfaction level of the users (see Appendix). Specifically, 71.4% of the RAs mentioned that they understood the process of submitting the report and 64.3% said that they could easily revise and resubmit the reports. Regarding the status tracking and feedback functionalities, 57.1% of the respondents found it useful or very useful, and 92.9% of the participants rated the dashboard usability as “4” or “5”. Such responses show that RAs agreed that the system was easy to use and its approval process was easily understandable. PAs also gave positive responses. 75% of them described the ease in accessing and reviewing the assigned reports as easy while 62.5% of them said that the system facilitated approval and feedback. Also, 62.5% of the participants declared that status labels and report history features were enough, and 100% of participants evaluated the dashboard as ‘4’ or ‘5’ in terms of clarity. Most importantly, 75% of the PAs considered that the system could substitute traditional tools like e-mail or spreadsheets.

Moreover, users provided some recommendations on further possible improvements for the application. Some of the RAs and PAs recommended for enhancing the accessibility through the inclusion of multiple languages and the incorporation of automated emails to notify users of status changes. These suggestions reveal the potential development and enhancement of the system to enhance its responsiveness to the expectations of the users in the future.

Unit Testing

Using Jest and React Testing Library, the team achieved:

- 95% coverage for authentication services
- 92% for data services
- 88% for UI components

All critical operations such as user login, data handling, and role-based access control performed as expected.

API Testing

Postman was used to validate endpoint reliability, with:

- 99% success for authentication APIs
- 99% for report management
- 99% for contract-related operations

All endpoints responded within 200ms, meeting performance expectations

Conclusion

The RA Reports System was created in response to a pressing need within the institution – the absence of an effective, unified, and efficient system for submitting and tracking Research Assistant report submissions. As a result of the designed and scoped approach, the role-based architecture, and the requirement to continuously develop the system over two semesters, the system now offers a straightforward and reliable computing-based solution to the administrative needs of Nazarbayev University. Some of the major accomplishments of the project are secure role specific dashboards, reports submission and review revision cycle, comment based feedback system and status tracking system. The functionalities test of the system and users' feedback pointed out that the system achieves its primary goals. Research Assistants stated that the program is easy to use for input and output while Project Administrators noted that the reports could easily be reviewed, commented on and approved. Positive responses by the two groups of users also endorse the functionality, feasibility, and applicability of the system within institutions.

There are several more improvements that could make the system more robust and capable of scaling up to the size of institutions. They include making the application support multiple languages to accommodate a large number of users and linking it with other financial programs like 1C to facilitate automation of the payment of the research assistant. Also, it is convenient to use email notifications and integrate the platform with the university identity management system. Therefore, the RA Reports System illustrates how a specific, role-based application can enhance academic reporting activities. It creates the basis for future institutional implementation and can serve as an example of additional enhancements in other administrative and research-related applications.

References

Abbas, Y., Martinetti, A., Rajabalinejad, M., Schuberth, F., & Van Dongen, L. a. M. (2022). Facilitating digital collaboration through knowledge management: a case study. *Knowledge Management Research & Practice*, *20*(6), 797–813. https://doi.org/10.1080/14778238.2022.2029597

Amiruddin, N. I., SaDan, S. A., Mohamed, H., & Adzmi, N. H. a. M. (2019). The Research Assistant Management System (RAMS). *International Journal of Advanced Science Computing and Engineering*, *1*(3), 169–178. https://doi.org/10.62527/ijasce.1.3.23

Kovaleva, N. N., Eresko, P. V., Izotova, V. F., & Gafarov, Y. R. (2022). Optimizing the implementation of university digitalization practices. *The European Proceedings of Social & Behavioural Sciences,* 353–359. https://doi.org/10.15405/epsbs.2022.01.57

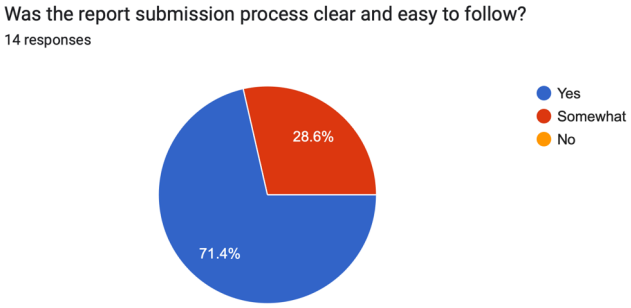
McDaniel, P. (2015). *What Does a Research Assistant Do?* Classroom. http://classroom.synonym.com/research-assistant-do-2624.html

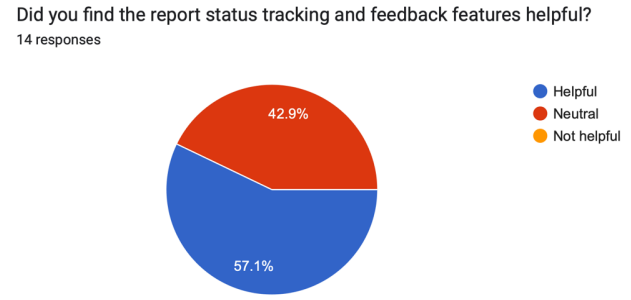
Yang, D., Tsai, C., & Huang, J. (2005). A case study on research assistant system: From knowledge management perspective. *Computer Science and Information Systems, 2*(2), 65–82. https://doi.org/10.2298/csis0502065y

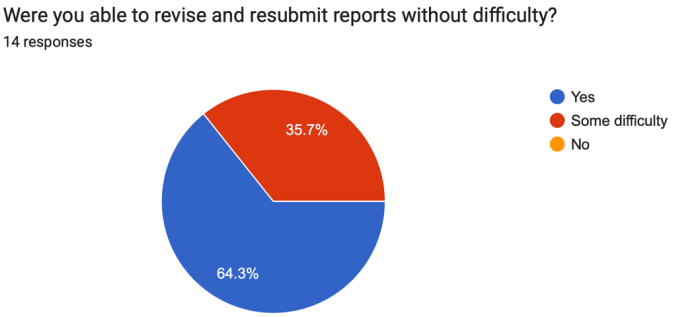
Yaya, D., & Baskan, G. A. (2013). The Opinions of Research Assistants in Education Faculties Regarding their Working Lives. *Procedia - Social and Behavioral Sciences*, *93*, 1355–1361. https://doi.org/10.1016/j.sbspro.2013.10.043

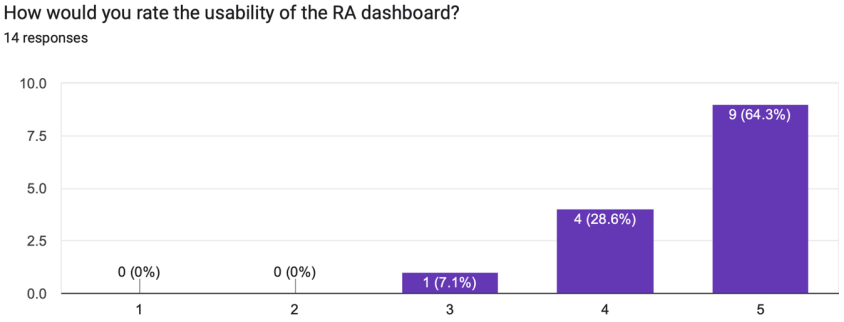
Appendix

RA questionnaire responses





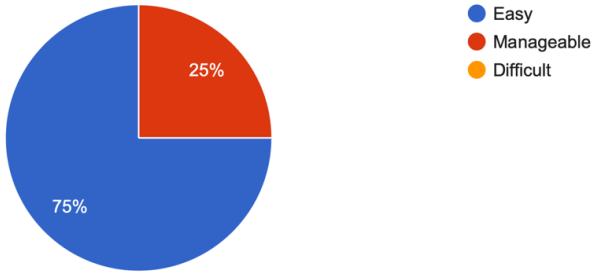




PA questionnaire responses

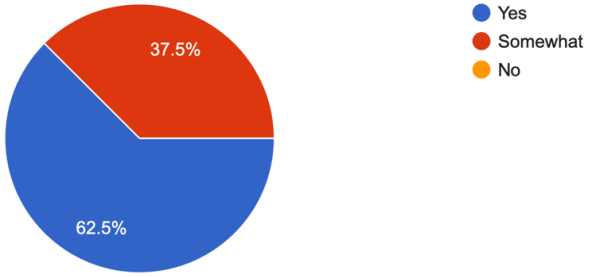
Was it easy to access and review assigned reports?

8 responses



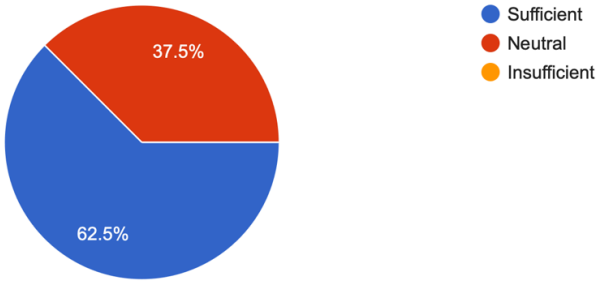
Did the system improve the approval and feedback process?

8 responses



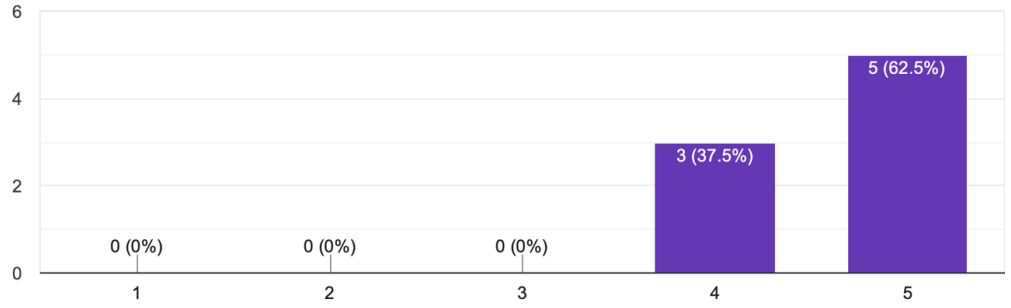
Were the status labels and report history features sufficient?

8 responses



How would you rate the PA dashboard layout?

8 responses



Can this system replace email or spreadsheet-based reporting?

8 responses