CSCI 409 Senior Project II – Final Project Report Spring 2025

Title of the project: NU Housing Management System

Team Members: Assyl Rakhmashev, Khafiz Khader, Bekzat Sarsenbiyev, Karassay Raushanbek, Galymzhan Turysbekov.

Project Advisor/Co Advisors: Askar Boranbayev, Almas Amirbekov, Talgat Manglayev.

1. Executive summary:

1.1. Summary

The Housing Management System (HMS) is a project focused on developing a new platform to manage housing operations at Nazarbayev University with an emphasis on scalability and efficiency. It addresses the limitations of the existing system, which relies heavily on an outdated PHP-based platform, Excel spreadsheets, and Google Forms.

The main objective of the project is to create a unified, web-based platform specifically for the university's administration to efficiently accommodate the university's faculty members and renters. Operations on the platform include managing the room assignments, tenant data, and generating tailored reports for each tenant. Additionally, the system aims to provide administrators with real-time access to occupancy status, check-in/check-out information, and maintenance records.

We applied the Agile methodology throughout all stages of the software development lifecycle. Scrum practices were employed to organize tasks efficiently, prioritize backlog items, and deliver value incrementally. This approach enabled us to structure planning, adapt to changing requirements, and continuously improve based on stakeholder feedback. As a result, our team is delivering a responsive and scalable platform, supported by a microservice-based backend and an optimized database – ensuring high system availability and performance.

1.2. Alignment with solution

The NU HMS aligns with the design, implementation, and evaluation of a computing-based solution by addressing a real-world problem through collaborative software engineering practices. During the design phase, our team identified the core functional requirements for managing buildings, rooms, tenants, assignments, and user authentication. We created system architecture diagrams, designed relational database models, and divided responsibilities between the backend and frontend teams. On the backend, we defined entities (Room, Tenant, Building, Assignment, etc.), established relationships, and enforced domain logic using enums like RoomStatus and AssignmentStatus. On the frontend, we designed responsive interfaces with forms, dashboards, and filtering views to make the system intuitive and user-friendly. In the implementation phase, the backend team developed a RESTful API using Spring Boot, with JWT-based authentication and role-based authorization. The frontend team implemented the user interface using modern web technologies and connected to backend services via HTTP calls. Data exchange between layers was managed using DTOs and form validation. The system supports all CRUD operations, dynamic filtering, and role-specific access for admins, tenants, and faculty members. For the evaluation phase, we tested backend endpoints using Postman and verified

frontend functionality through user interaction scenarios. We handled validation errors, edge cases (e.g., full rooms, invalid assignments), and ensured smooth API integration across components. The project demonstrates a complete, secure, and maintainable computing-based solution that meets both user and system requirements through effective collaboration and technical implementation.

2. Introduction

The existing platform used to manage the housing operations is outdated and usually requires manual administrative effort, often leading to delays and inconsistencies. The administration frequently encounters random system failures, challenges in updating room assignments, managing tenants, and generating reports.

Our team's motivation is to develop a more reliable solution that offers a stable platform with user-friendly interface, as well as improved scalability and maintainability. Our platform is specifically tailored to meet the housing needs of the university, improving automation, transparency, and operational efficiency.

NU HMS includes modules for room assignment, resident tracking, and maintenance requests. The report is organized into sections covering the background, design, execution, evaluation, and conclusions.

3. Background and Related Work

3.1. Existing solutions

To determine the direction of the project, our team first investigated the University's current approach, identifying its strengths, weaknesses, and key stakeholders. After contacting University Service Management (USM) and Student Housing, we collected detailed data that includes the layout of housing blocks, classification of the buildings, number and types of rooms, and other relevant structural information. This data served as a visualisation of the user interface and conceptualizing how to optimize the accommodation and assignment process. Apart from it, Student Housing demonstrated to us the existing platform, which has been in use for over a decade. By understanding the working principle of the system — particularly the check-in/check-out operations — we were able to understand the platform's functional principles and identify limitations in usability, reliability, and adaptability. Furthermore, we had continuous communication with USM, our main stakeholder. We shared our regular updates and gathered feedback to align the system with their expectations and outline the areas to improve later.

3.2. Methodology

Agile methodology and its Scrum practices suited our approach the most because we conducted regular team meetings, met often with stakeholders, and modified our software on a continuous basis in response to evolving requirements. The Agile method allowed us to divide the project into sprints, delegate and prioritize the tasks through a dynamic product backlog, and deliver functional increments at the end of each sprint. In contrast, plan-driven approaches like the Waterfall model were not appropriate for our project since they require fixed requirements and do not consider instant modifications. Changes would have been costly and time-consuming under a plan-driven approach.

Overall, Agile and Scrum enabled us to be adaptable, gather feedback continuously, and improve our system iteratively to gain the dynamic workflow of HMS development.

3.3. Existing literature

To enhance our working schedules, time management, task delegation and methodology, we analyzed the literature related to Software Project Management, particularly *Project Management: Process, Technology, and Practice*, written by Vaidyanathan, G., and *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. These resources enabled us to optimize team's cooperation, adapt new aspects of the development processes, and improve the structure and organization of our work.

4. Project Approach

4.1 Solution overview

Our project is a web-application called Housing Management System designed to streamline and automate housing assignments for tenants (faculty or renters) across multiple building types such as apartments, townhouses, and cottages. The backend was built using Java with Spring Boot, and PostgreSQL served as the relational database. The frontend was developed using React, leveraging react-hook-form for validation and react-router for client-side navigation.

The system follows a layered software architecture:

- **Controller Layer**: Handles incoming HTTP requests and maps them to service logic.
- **Service Layer**: Contains business logic, such as assignment validation and check-in workflows.
- **Repository Layer**: Manages data persistence using Spring Data JPA.
- **Security Layer**: Secured endpoints with JWT-based authentication.

Frontend and backend communicate over RESTful APIs, with structured DTOs ensuring smooth and secure data exchange. The application contains different pages/sections where a user can navigate through. For example, Login Page, Blocks Page, Rooms Page, Tenants Page, and etc.

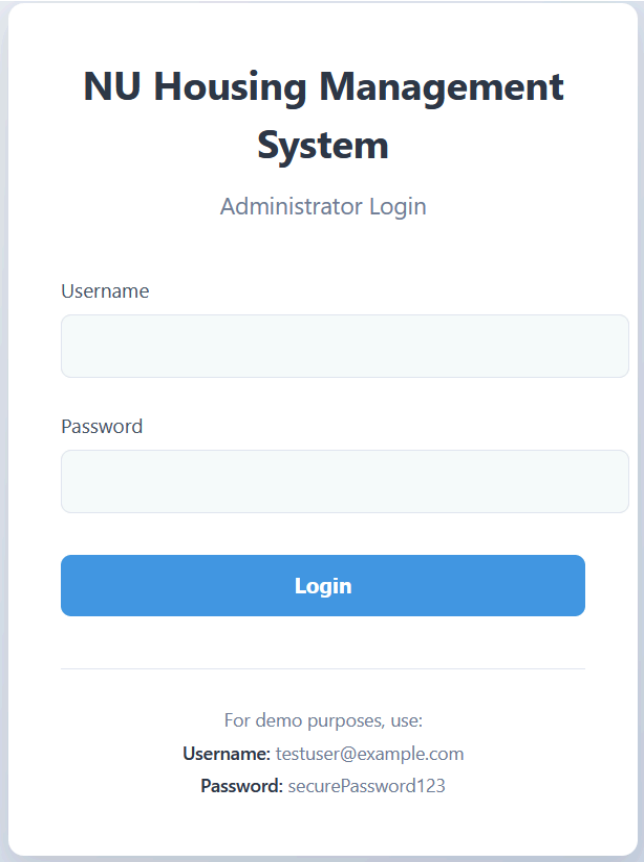

Figure 1. Login Page

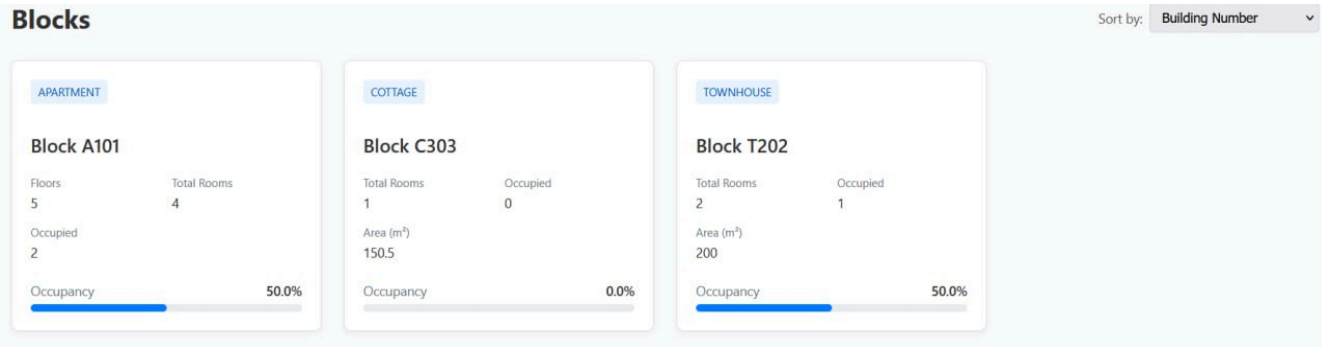

Figure 2. Blocks Page

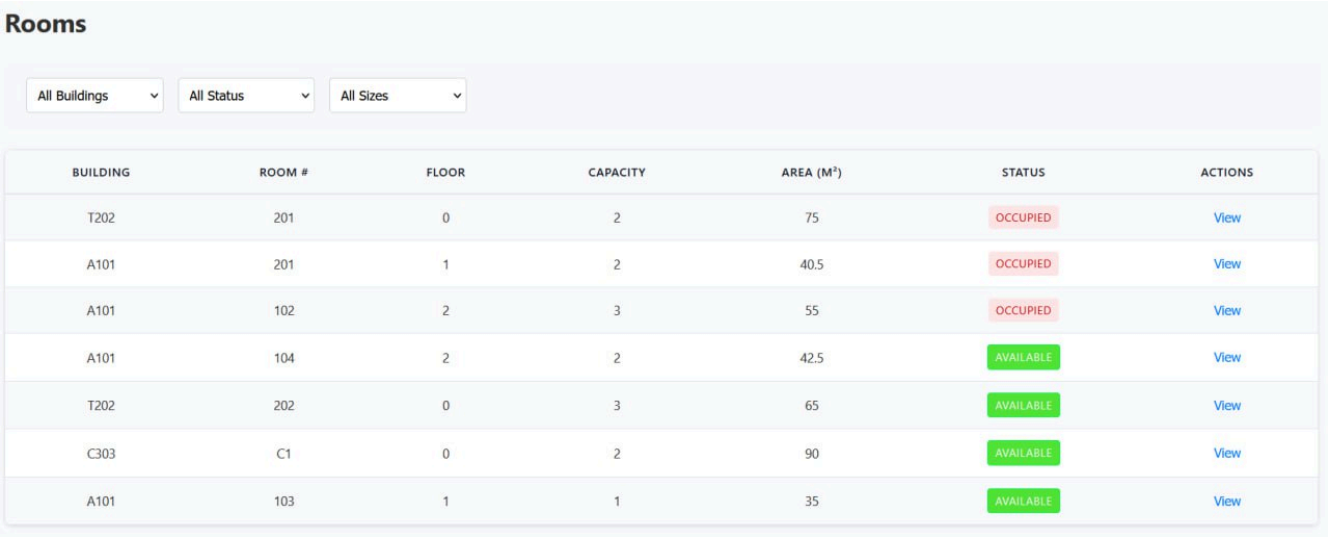

Figure 3. Rooms Page

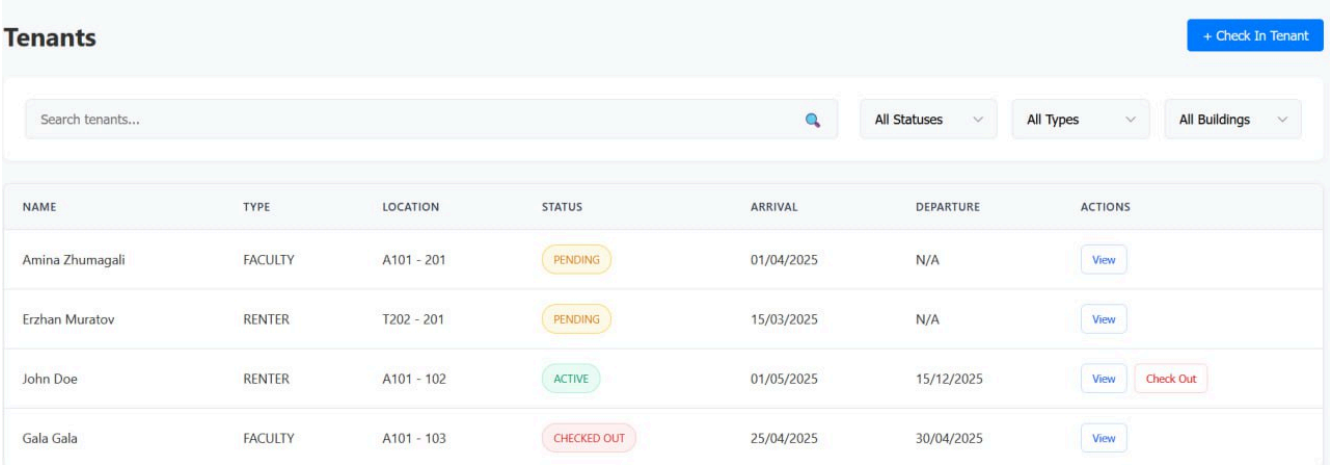

Figure 4. Tenants Page

4.2 Key features and workflows:

The Housing Management System offers a range of core features. One of them is user authentication, implemented using JWT (JSON Web Token). Upon successful login, a secure token is issued and required for accessing protected endpoints, with user roles (e.g., admin or staff) determining the scope of access. This ensures both security and proper access control throughout the application.

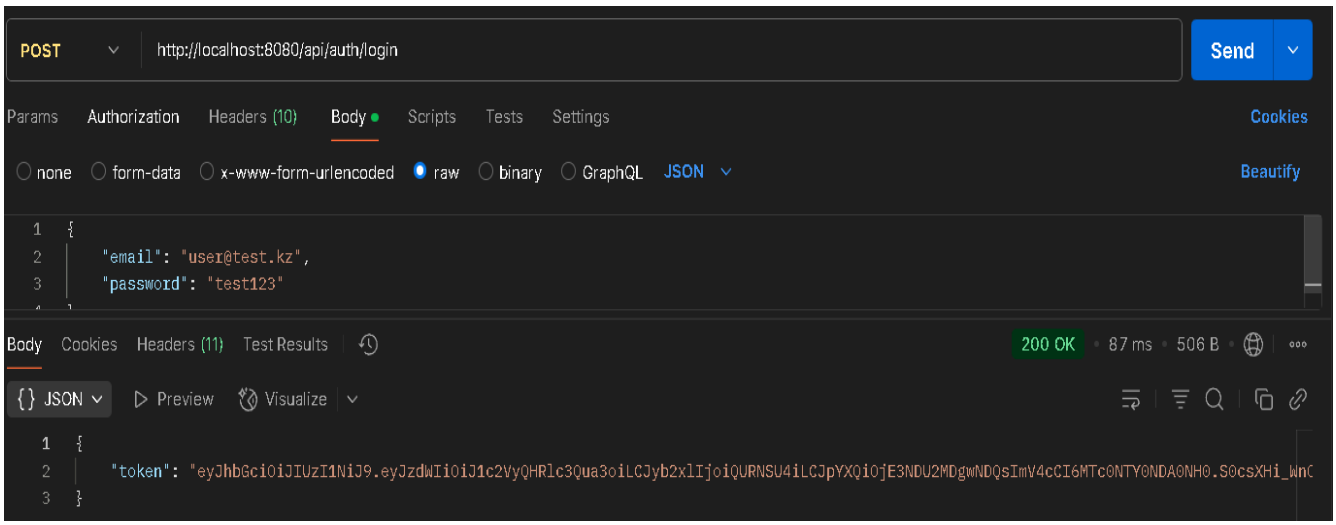

Figure 5. JWT token access

The tenant check-in and check-out workflows are built around the Assignment entity. When checking in, the system creates a new tenant record and associates it with a room, marking it as occupied. Conversely, during check-out, the system marks the assignment as completed and frees up the room, maintaining accurate room availability status at all times.

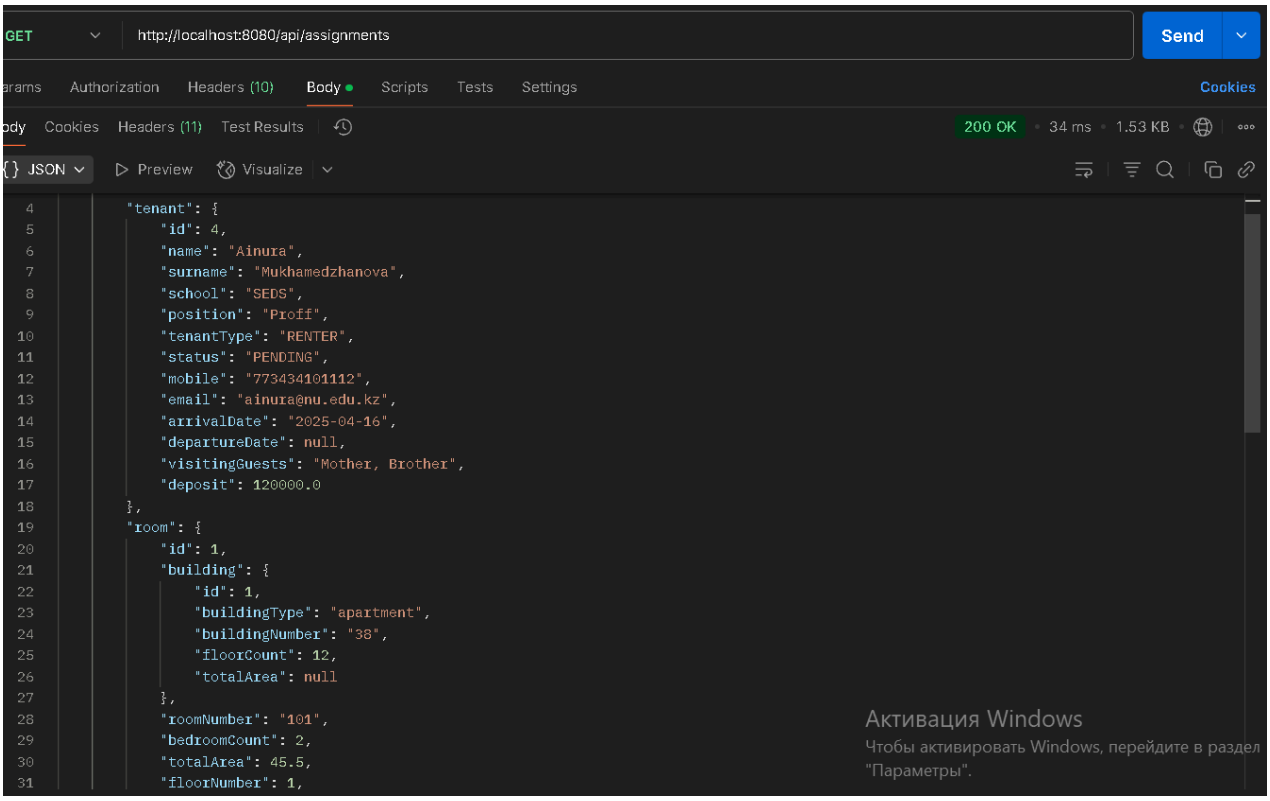

Figure 6. Assignment GET response

Another important component is family member management, which allows the registration of dependents related to a tenant. This feature is particularly useful in contexts where tenants may live with spouses or children and is fully integrated with the tenant profile.

Additionally, the maintenance management feature allows administrators or staff to create and track maintenance requests for specific rooms. The form includes fields for selecting the room, tenant (optional), maintenance category, and priority level. Users must also provide a short description of the issue and can include additional notes for clarity. Once the form is submitted by clicking "Create Request," the system saves the request with the selected priority level and displays it in the table below for further action or monitoring.

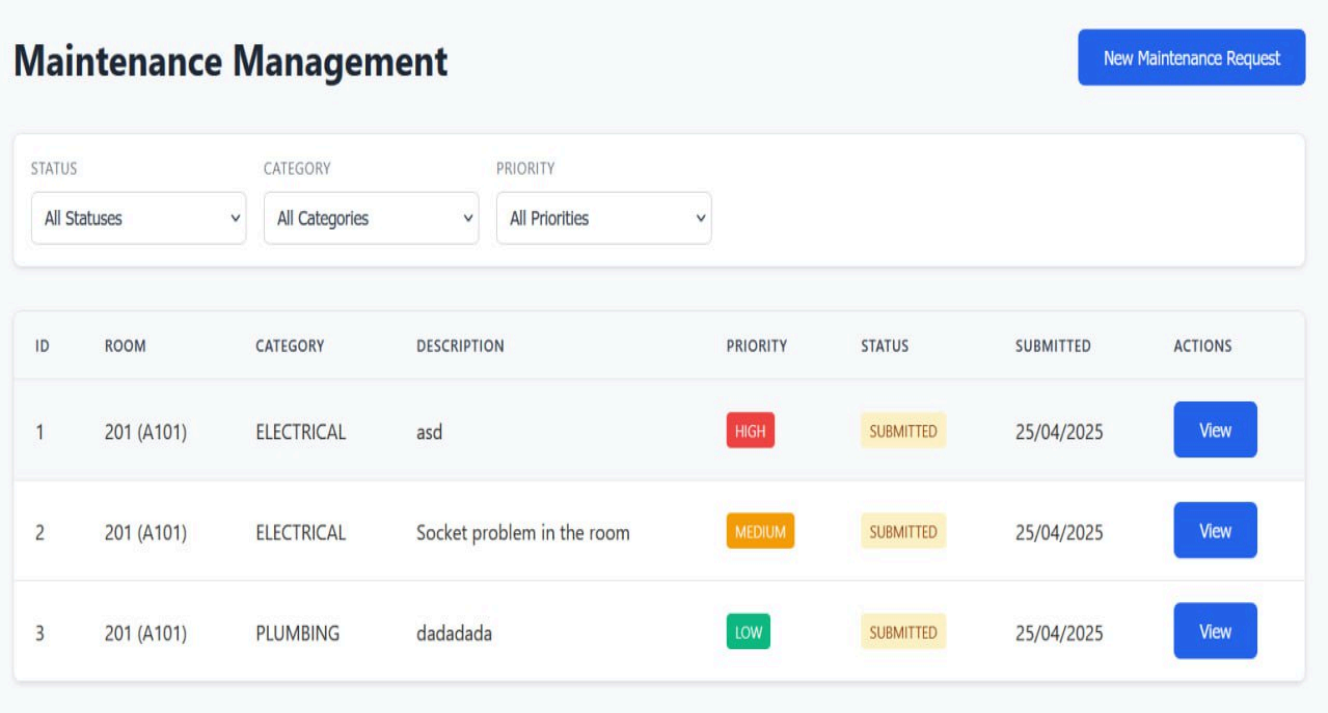

Figure 7. Maintenance Page

The system also supports dynamic filtering for both tenants and rooms. The endpoints /api/tenants/view and /api/rooms/view enable flexible, multi-parameter queries such as filtering by building, room status, tenant type, or number of bedrooms. This greatly enhances usability, allowing administrators to quickly find specific entries without manual searching. The multilingual support is also included as a feature of our software platform: administrators can switch the service language to Kazakh, Russian, or English according to their preference.

Finally, the application is containerized using Docker and deployed via Render, a cloud hosting platform. This setup ensures that both the backend and frontend are consistently deployed and can be scaled or updated easily in the future.

4.3 Third-party components

To accelerate development and ensure reliability, we integrated several third-party components that were not implemented from scratch but played a critical role in the system's functionality. For secure user sessions, we used JWT, a Java library for creating and verifying JSON Web Tokens. It allows us to generate signed tokens during login, which are then used to authenticate users across protected endpoints without maintaining server-side sessions.

Authentication and authorization are further streamlined by Spring Security, which provides robust protection mechanisms with minimal custom configuration. It integrates seamlessly with our JWT setup, allowing us to define role-based access control and secure endpoints against unauthorized access.

On the frontend, we relied on two powerful React libraries. react-hook-form simplifies form state management and validation while minimizing re-renders, ensuring efficient and responsive user interaction. Meanwhile, react-router enables dynamic navigation between components, providing a smooth single-page application experience for tasks like tenant registration, room viewing, and assignment management.

As was mentioned previously, we adopted Docker, which allowed us to package both the backend and frontend into isolated, reproducible containers. This ensured consistency across development, testing, and production environments. The final application was deployed using Render, a cloud hosting platform that supports continuous deployment of containerized applications, making it easy to host our API and frontend with minimal configuration.

4.4 Team collaboration and roles

The project team was divided into clear roles to maximize efficiency and ensure a smooth development process. Galymzhan was primarily responsible for the backend development, focusing on entity modeling, designing the main system architecture, implementing core CRUD operations, developing the JWT-based security framework, and building dynamic filtering functionalities. Karassay also helped with developing the backend architecture and contributed significantly to the deployment phase, preparing Docker configurations and deploying the system on the Render platform.

Frontend development was handled by Assyl and Bekzat, who were responsible for building user interface components, designing forms, and integrating API responses into the React-based frontend. They also worked on adjusting API requests to ensure smooth communication between frontend and backend. Khafiz contributed to both frontend development and testing activities, helping build React components, refining the integration with backend APIs, and conducting tests to ensure user flows worked as expected.

Throughout the project, we collaborated actively using GitHub for version control, Telegram for daily communication, and Jira to organize tasks and monitor progress. We also utilized Postman collections extensively for testing and sharing backend endpoints, which helped ensure that frontend and backend development stayed synchronized. Weekly meetings and task reviews were held to align work, address any blockers early, and provide mutual support for problem-solving.

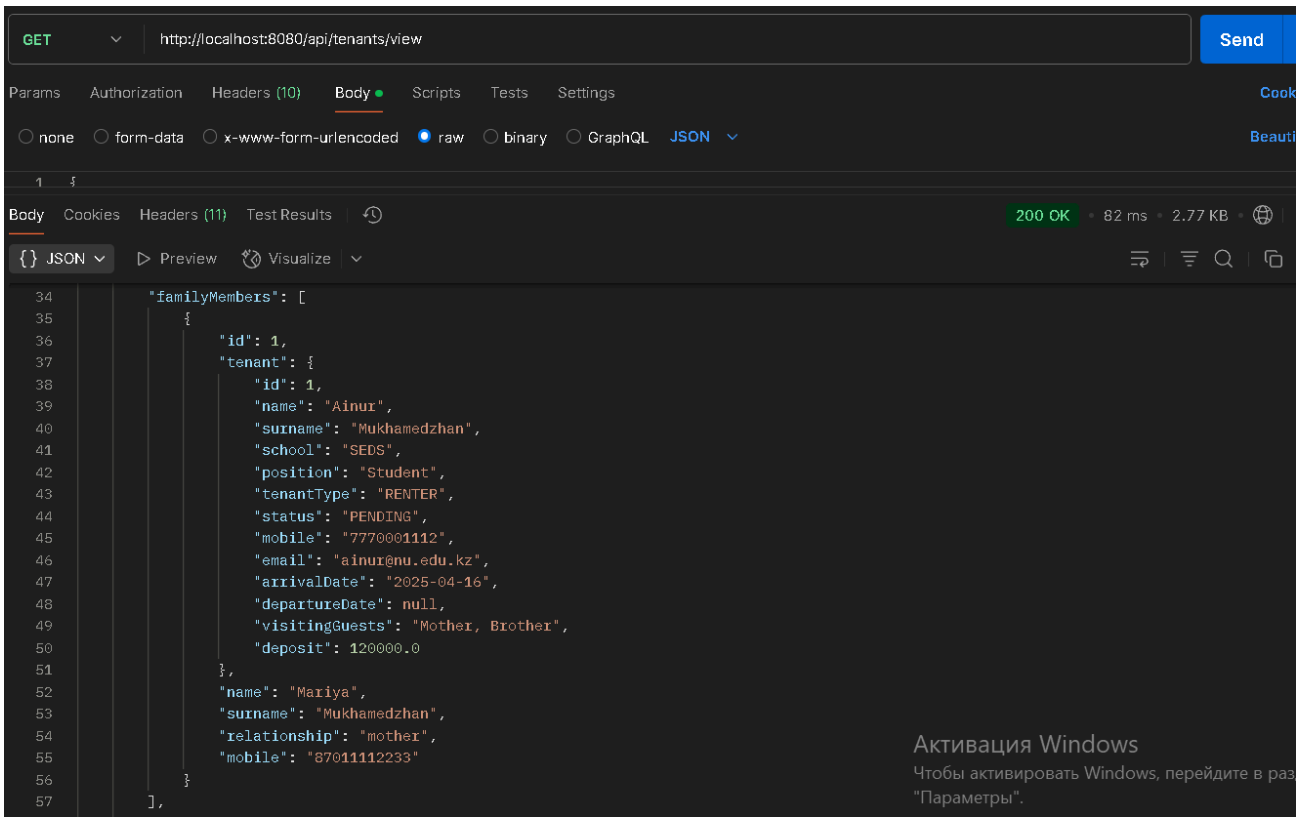

Figure 7. Testing on Postman

5. Project Execution

5.1 Development timeline & Major decisions

Our project evolved significantly over two semesters, moving from a simple housing registration system to a more complex and functional Housing Management System. Initially, we envisioned building a basic platform to register tenants and assign them to rooms. However, during our initial user requirement gathering phase, we met with representatives from USM (University Service Management) who highlighted the limitations of manual tracking and shared additional needs, such as validating room capacity, managing tenant families, and tracking active/inactive tenants over time.

As a result of these, we introduced key design modifications:

1. We added a dedicated Assignment entity to separately track tenant check-in and check-out dates, expected departure, and assignment status (e.g., ACTIVE, COMPLETED).
2. The tenant and room relationship was restructured to use relational mapping instead of embedding fields directly, which enabled more flexible and scalable filtering.
3. We also refined the use of enums to improve code clarity, database consistency, and support dynamic behavior in filtering endpoints.

Consequently, we have continued meeting with the University Service Management team to gather more requirements and to check our work. Our team followed agile and scrum **development** principles by breaking down the work into sprints, conducting regular team meetings, and iteratively reacting to input. This approach allowed us to continuously improve the system design and incorporate feedback without major errors.

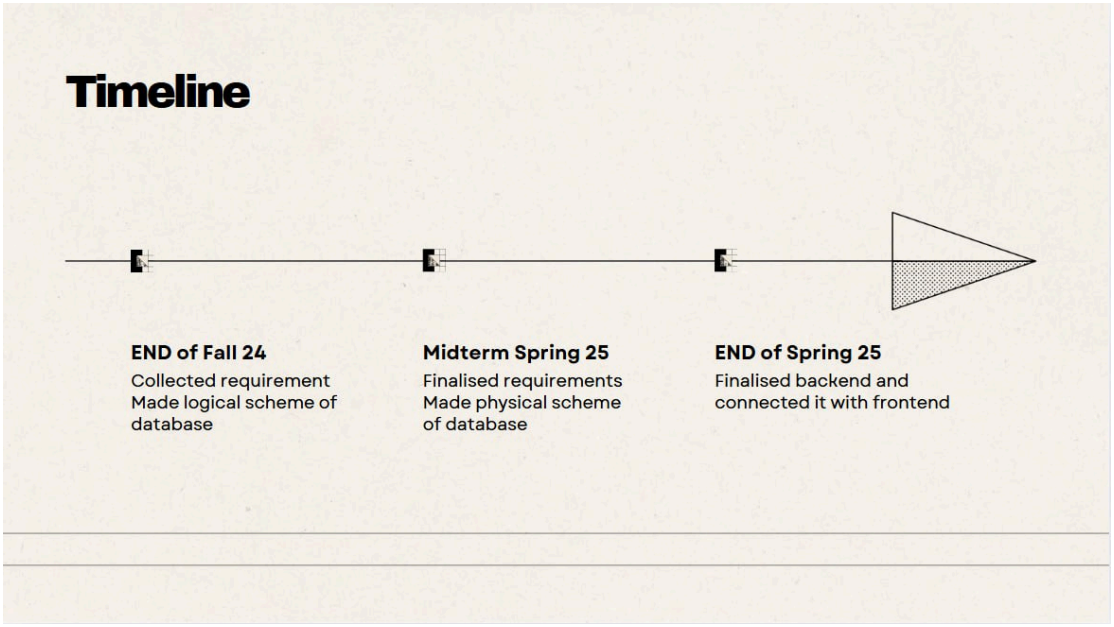

Figure 8. Project Timeline

The timeline image showing the three major milestones:

- End of Fall 2024: We collected initial requirements and designed the logical database schema. Also, we made the
- Midterm Spring 2025: Requirements were finalized in collaboration with USM, and we developed the physical database schema, mapping out all entity relationships.
- End of Spring 2025: The backend was fully implemented, with working authentication, filtering, assignment logic, and data validation. The backend was integrated with the React frontend and deployed on Render using Docker.

This structured timeline ensured that we remained focused and had clear deliverables for each phase. It also helped us stay accountable to stakeholders, such as the USM representatives, who were consulted at key checkpoints to validate that our system aligned with real user needs.

5.2 Challenges and resolutions

Throughout the project, we faced a number of technical and organisational problems that put our adaptability and collaboration skills to the test. The graphic summary identifies three primary types of challenges: late and changing requirements, difficulty linking frontend and backend, and complicated entity connections, all of which had a substantial influence on our operations.

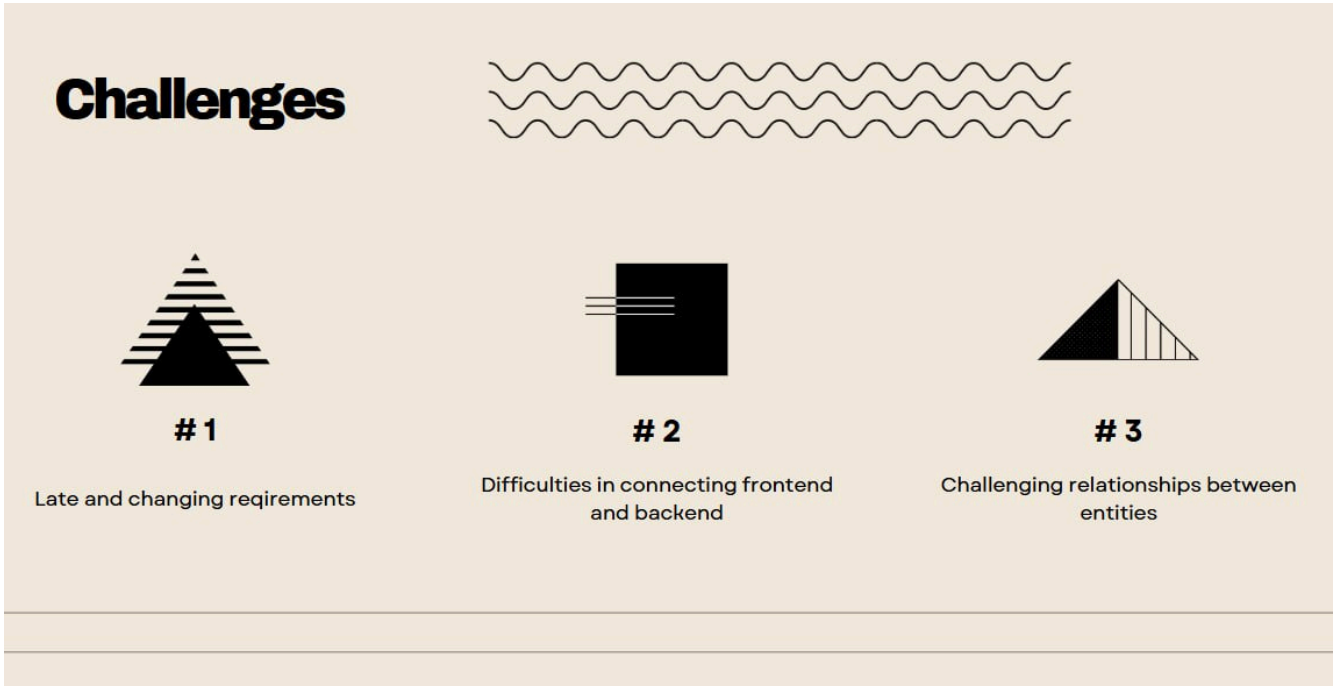

Figure 9. Major challenges

One of the primary challenges was dealing with late and evolving requirements. Initially, the project scope was limited to tenant registration. However, after meetings with the University Service Management (USM), additional requirements such as dynamic room assignment, capacity limits, and family member tracking were introduced. These late-stage adjustments required us to redesign our entity relationships and introduce new backend workflows. To manage this, we adopted agile and scrum principles, allowing us to integrate feedback and incrementally adapt the system's design. Additionally, we constantly tested the system once the requirements changed to ensure that the modifications are correctly inserted.

We also faced integration issues between the frontend and backend. While the backend followed a RESTful design, there were frequent mismatches in data structure, missing fields, and misunderstood endpoints. These were addressed through shared Postman collections, and close collaboration between frontend and backend developers during integration testing.

From the backend perspective, configuring Spring Security with JWT was initially a source of error. We encountered recurring 403 Forbidden errors even with valid tokens. These issues stemmed from misconfigured security filters and incorrect token parsing. After multiple rounds of debugging using Postman, we refined the JwtAuthFilter and role-based endpoint access to stabilize authentication. Another major backend issue was handling complex entity relationships, especially when designing associations like Tenant to Assignment, and Room to Building. At one point, circular dependencies led to stack overflow errors and serialization problems in JSON responses. We resolved this by restructuring entity mappings, using different techniques and limiting bi-directional references.

On the frontend, component structure and reusability posed significant problems. Initially, components were built without strict separation of concerns, which resulted in large, cluttered files handling multiple responsibilities. This made debugging and reusability difficult. The team resolved this by

breaking components into smaller, focused units, each responsible for a single task such as data fetching, input validation, or UI rendering.
Lastly, deployment also brought some issues. We faced inconsistencies between local and cloud environments, especially related to environment variable handling. These were resolved by properly configuring .env files for the frontend and using application.properties for the backend, along with containerizing both services using Docker. Hosting was done through Render, which simplified the deployment process once our container setup was stable.
Despite these difficulties, each incident served as a learning opportunity. Our team's communication, adaptability, and willingness to debug jointly were critical to overcoming these challenges and delivering a working, stable system before the end of the semester.

6. Evaluation

To determine whether our Housing Management System effectively addressed the real-world problem of managing university housing, we involved user testing, feedback collection, and stakeholder validation. We conducted both functional and usability testing, and invited stakeholders - particularly representatives from University Service Management (USM) and our course professor - to review the system. Additionally, we executed internal testing with our project team and a few potential users to assess how well the application met technical and user-centered goals.

One of the most meaningful parts of our evaluation process was our collaboration with USM, who acted as our real-world client. Their early feedback during requirement collection shaped many of the system's features—such as family member tracking and room occupancy limits—and later, they reviewed a working demo of the system.

They suggested adding different future features such as email notifications and room availability dashboards, which we documented as future improvements. This input confirmed that our current solution addressed the essential requirements while leaving room for future scalability.

Our professor also evaluated the system during milestone submissions and the final demo. He assessed both the technical depth and usability of the system and gave feedback about it: Suggestions from our professor included adding deployment, optimizing frontend responsiveness and backend functionality.

7. Conclusion and possible future work

In conclusion, the NU Housing Management System marks a significant step toward modernizing housing operations at Nazarbayev University. The system replaces outdated tools with a scalable and secure web-based platform tailored to the university's needs. By implementing dynamic room filtering, secure tenant assignment workflows, and family member tracking, we addressed key administrative challenges while ensuring data consistency and user-friendly interaction. The Agile approach enabled us to remain flexible, include stakeholder input, and provide a strong and practical solution within the project deadline.

The introduction of a utility billing system, which will help to automate financial tracking and lower manual overhead, is one of the key goals for future development. Enhancing the system with an analytics dashboard could also provide valuable insights into room occupancy, tenant trends, and maintenance frequency. Furthermore, expanding support for mobile responsiveness or creating a dedicated mobile app would improve accessibility and user convenience.

The existing system lays a solid basis for future expansion and has the potential to grow into a complete, university-wide housing management platform.

8. References

1. Vaidyanathan, G. (2013). *Project management: Process, technology, and practice*. Pearson Education.
2. Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® Guide)* (7th ed.). Project Management Institute.
3. Postman. (n.d.). *Overview*. Postman Learning Center. Retrieved from https://learning.postman.com/docs/introduction/overview/