# NU Housing Management System

Almaz Balgali, Nurzhan Kozhamuratov, Dana Kurasbek, Nurbek Nussipbekov, Adi Yeltay

School of Engineering and Digital Sciences, Nazarbayev University

CSCI 409: Senior Project II

Dr. Askar Boranbayev, Almas Amirbekov

April 25, 2025

**Content**

## 1. Executive summary

### a. Summary

Throughout this semester, we have been developing the Housing Management System (HMS) requested by the University Service Management (USM). The problem to be solved was to develop a modernized version of an outdated HMS that collects all information into a single system; currently, USM has several Excel files with tens of unstructured sheets storing tenants data. Key objectives include developing a new system with flexible report generation features.

To develop this new system, we read several secondary resources to identify the key features every HMS should have. Secondly, we conducted primary research by repeatedly interviewing the stakeholders (i.e., USM, DSS staff) to get a better understanding of the system and collect requirements. Overall, for the development process, we used an agile incremental approach since we communicated closely with stakeholders and adjusted the system accordingly. We also used Trello for task management, sprint planning, and team collaboration. Finally, we used different testing methods to evaluate our system.

As a result, as mentioned above, we built a web-based application, which has eliminated the need for multiple Excel sheets. The system also provides automated report generation supporting both filter-based reports (e.g., tenants by building) and chat-based reports, where the user types their queries or prompts (e.g., “show tenants who have pets”). This new HMS makes USM workers’ work easier and saves their time.

### b. Alignment with solution

Initially, during the first semester, we were told to create a role-based system that can be used by both tenants and admins. However, after meeting with USM at the beginning of this semester, we realized that only USM and DSS staff could enter into the system and use it as a centralized management database. Requirements for the project were also altered throughout the semester to add new features or after finding out new unnoticed functionality.

First of all, we researched the existing HMS, getting raw comprehension about what to build. After collecting requirements from stakeholders, we got a better understanding of the project. We created different diagrams and then started developing the system.

As our primary technology stack, we used the following tools: React.js for the frontend and SpringBoot for the backend parts. Additionally, we used the Gemini 2.0-flash model for a flexible chat-based report generation feature. This combination allowed us to create a website that fulfills the aim of creating a responsive, user friendly, easy to use system for the USM and DSS staff.

## 2. Introduction

a. Overview of the project

This project aims to streamline the work of the University Service Management (USM) and Department of Student Services (DSS) by developing an efficient and secure Housing Management System (HMS) that will keep track of living spaces on campus. It is currently difficult and time-consuming to manually manage data of residents, their rooms, and this can lead to entry errors,miscommunication, delays in meeting residents' needs, etc. Additionally, creating customized reports was complicated and time consuming. This system will make it easier to maintain housing operations by simplifying data entry, editing, and generating customized reports for administrators.

Therefore, HMS will be a web-based platform that optimizes and centralizes the management of living spaces on campus. Information on residents, dorms, rooms, apartments, and maintenance issues will be tracked and stored. Apart from that, the system will automate the creation of customized reports and allow administrators to manage resident data. It will also follow security guidelines and university policies offering safe access to personal data. The system will run smoothly on various web browsers and operating systems and provide multilingual support (i.e., English, Kazakh, and Russian).

So, to develop the system, we will implement the following features; first, it will control the distribution of rooms, dormitories, and apartments for residents (i.e., students and staff). Also, the system will generate reports for administrators about room occupancy, the need for their maintenance, etc. The reports can be customized to the needs of the administrator by applying built in filters and Gemini AI powered chat . Moreover, as mentioned earlier, we will implement multilingual support (i.e., English, Kazakh, and Russian). We will also implement password protection, role-based access control (RBAC). Roles are separated into USM and DSS staff. Different roles have different access to the functionality of the system.

b. Objectives and goals

Our objectives and goals for this semester were to refine the requirements in close contact with DSS and USM ,and completely implement the final system.

The inputs to the system will include resident data, and room assignments. In addition, this system will process requests for maintenance services and room changes. Other inputs into the system will include updates from administrators. More precisely, resident data will consist of resident ID, first and last names, arrival date, departure date, school/department, position, type of the tenant, family information, email, pet info, spouse, relative, and visitors info. Room information will include data on room availability, the capacity of each room (ranging from 2 to 4 places), room area, the building, block, floor, and room numbers. The requests for maintenance

will include details on what needs to be fixed in the room, and the requests for room changes will consist of information about the current and desired rooms. Additionally, there will be information about the violations of the residents, like not following sanitary regulations. Such information will contain data about type of violations, actions taken, date and tenant data. We used mock data for security and privacy reasons. The website will be hosted on a hybrid server accessible only through NU network for security purposes.

The output of the project is a web application that allows administrators to manage the housing system easily. The system generates reports for administrators. In addition, an AI-based solution was integrated for analyzing provided data, and generating customized reports using chat.

The proposed solution is implemented in Java with Spring Boot, React, and PostgreSQL. Spring Boot is used for the backend as it provides security and scalability to the system; it will handle managing user accounts, generating reports, etc. React is used for the frontend because it can create a responsive UI for the system. PostgreSQL will be used for the database, i.e., store data on residents, rooms, etc. It will also contribute to security and scalability providing reliable performance. Also, we used Google Gemini for AI-based chat. On top of that, the system described will interact freely with already existing systems. We used agile development for incremental development of the project from the feedback of the USM and DSS staff.

## 3. Background and related work

We read a number of articles about criteria for developing an effective HMS and examined similar systems to understand what kind of software we might need.

Evo (2023) highlights that HMS should be automated, which is not the case for the current system; USM workers manually enter data into Excel and pull out reports. The system should also be able to keep track of the changes, such as the history of a resident’s check-ins, check-outs, and violations (Evo, 2023). Evo (2023) also points out the importance of data security; for example, freeCodeCamp (2024) describes how to hash passwords with bcrypt. Moreover, the system must be integrated, which is not the case with the current system, as workers have to manually enter data about new residents, although all of it is contained in some database.

According to Evo (2023), an effective HMS should support functions such as requesting and scheduling repairs and maintenance requests, collecting rent, etc. In addition, the performance of the system should keep up with the growing number of residents and assets and be accessible on mobile devices for the residents' convenience.

The current system is “a mix of different applications” (Lau, 2024, para. 2), with Excel, PPS (as USM workers refer to the current system), etc., compromising data integrity. As

mentioned earlier, changes to the database containing student and staff data are not reflected in the current system, although according to Lau (2024), "A truly integrated system will be able to provide your team with live information as soon as data is processed" (para. 4). To solve this problem, the author suggests including API tools in HMS.

HMS should have comprehensive reporting (Lau, 2024), i.e., reports should be able to be generated in a variety of formats. Similarly, USM workers have requested that reports be generated based on specific filters.

Lau (2024) also suggests deciding whether to develop a completely new system or modify the existing one. If the latter, it is important to determine whether there is access to the full code of the current system. It is also recommended to communicate with USM to understand their needs and requirements.

On top of everything else, the system should be user-friendly, i.e., easy to understand and use. To familiarize users with the system, Lau (2024) suggests making a copy of the system, the so-called training system, so that workers can practice without altering the real data. On the other hand, instead of developing a training system, Gupta (2024) suggests making an in-app guide when workers first use the system, user manuals, tutorials, FAQs, etc.

Among other things, to refresh our knowledge on system development, types of development, and to create diagrams such as use case diagrams, we referred to Sommerville (2016).

Our system meets both the requirements identified above (mostly) and the ones put forward by stakeholders. It is fully automated, tracking check-ins, check-outs, maintenance requests, and violations for each room and providing real-time updates.

Also, it is secure as we used JSON Web Tokens for authorization and bcrypt for password hashing. Moreover, it is fully integrated, with frontend, backend, and database components working in symbiosis. Apart from that, we followed the Evo (2023) recommendation and added maintenance management to our system, which is missing in the current system. Furthermore, our system is scalable to accommodate the increasing number of residents.

Repeating the words of Evo (2023), the system should be accessible on mobile devices for the residents' convenience. However, while our system is available on mobile devices, it is designed only for the use by USM and DSS staff as this was a requirement of the stakeholders. Our system focuses on administrative efficiency rather than resident convenience; however, residents do benefit indirectly through faster service (e.g., approval/rejection comes faster).

In addition, as mentioned in the introduction, our system has a comprehensive reporting function supporting 1) filter-based reports (e.g., tenants by building); and 2) chat-based reports, where the user types their queries (e.g., "show tenants who have pets").

Finally, we developed a new system from scratch instead of modifying the existing one. Its interface is user-friendly and mostly intuitive, so we did not add an in-app guide or FAQ. However, we will do some live tutorials or record one for USM workers.

## 4. Project approach

### a. Solution overview/Software architecture

The Nazarbayev University Housing Management System (HMS) is a full-stack web application designed to improve and streamline accommodation housing management at our university. It consists of a Java Spring Boot backend, a React-based frontend, and a PostgreSQL database. The system supports multilingual interfaces (English, Kazakh, Russian), user authentication, room and tenant tracking, report generation, and our special integration with Gemini 2.0-flash.

We chose ReactJS because of its component-based architecture and responsiveness. It also facilitates multilingual support and an intuitive UI. As for the backend, in addition to Java Spring Boot, we also used RESTful APIs, JWT authentication, Personal Data Bcrypt, and PostgreSQL database. The latter was used for data persistence; it also supports schema flexibility and relational data integrity.

### b. Algorithms/Workflows

Basic CRUD and search/filter functionalities are implemented. Also, we used Google's lightweight Gemini 2.0-flash model integrated via state-of-the-art Model Context Protocol (MCP). The model is used for unstructured report generation where users can generate reports through easy-to-use our custom-made chat window using system prompts.

There are several workflows in our system: Login, Tenant Management, Room Management, Report Generation, to name a few. Login is role-based, with USM administrators having access to all features while DSS administrators able access only General, Complexes, Tenants pages. Apart from that, we created a separate admin role for the IT department and our own use. In Tenant Management, admins can add/edit resident profiles, assign rooms, and manage contracts (check-in and check-out procedures). They can also search for basic tenant information like full name, ID, etc. as well as specific information like family members, pets, and violation history. On the other hand, Room Management allows for tracking room occupancy, maintenance status, etc. Finally, in Report Generation, admins can generate reports—both structured (via customly filters) and unstructured (via AI-powered chat).

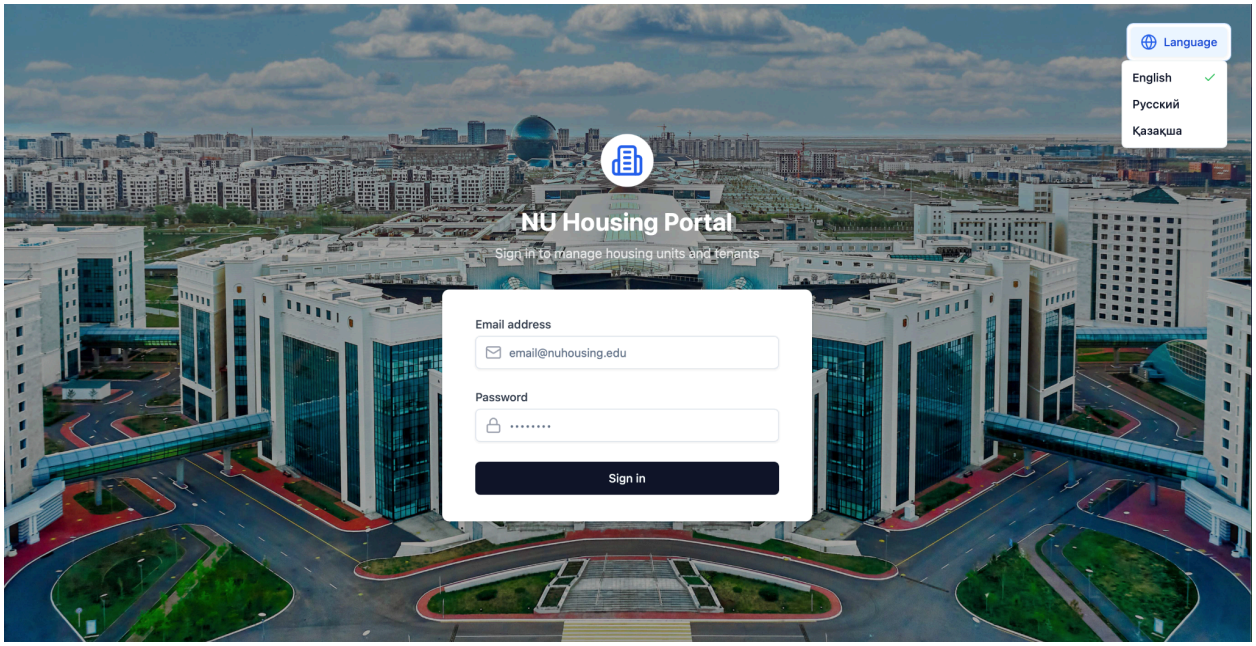


Figure 1. Login page

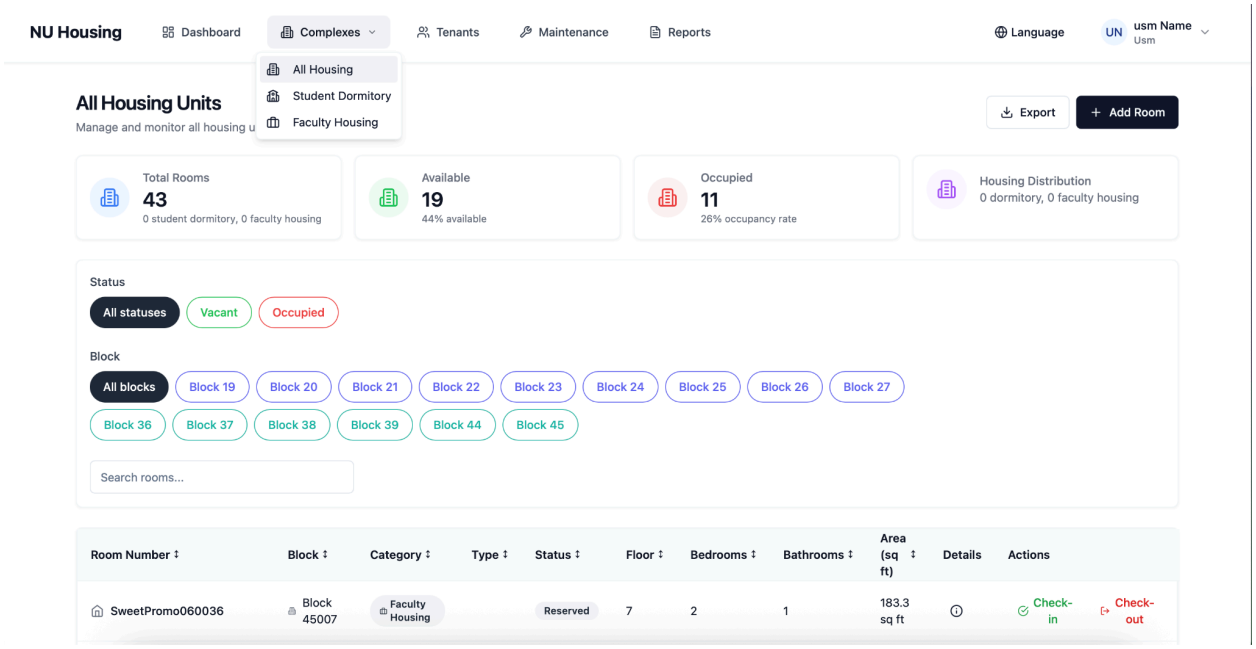


Figure 2. Complexes page.

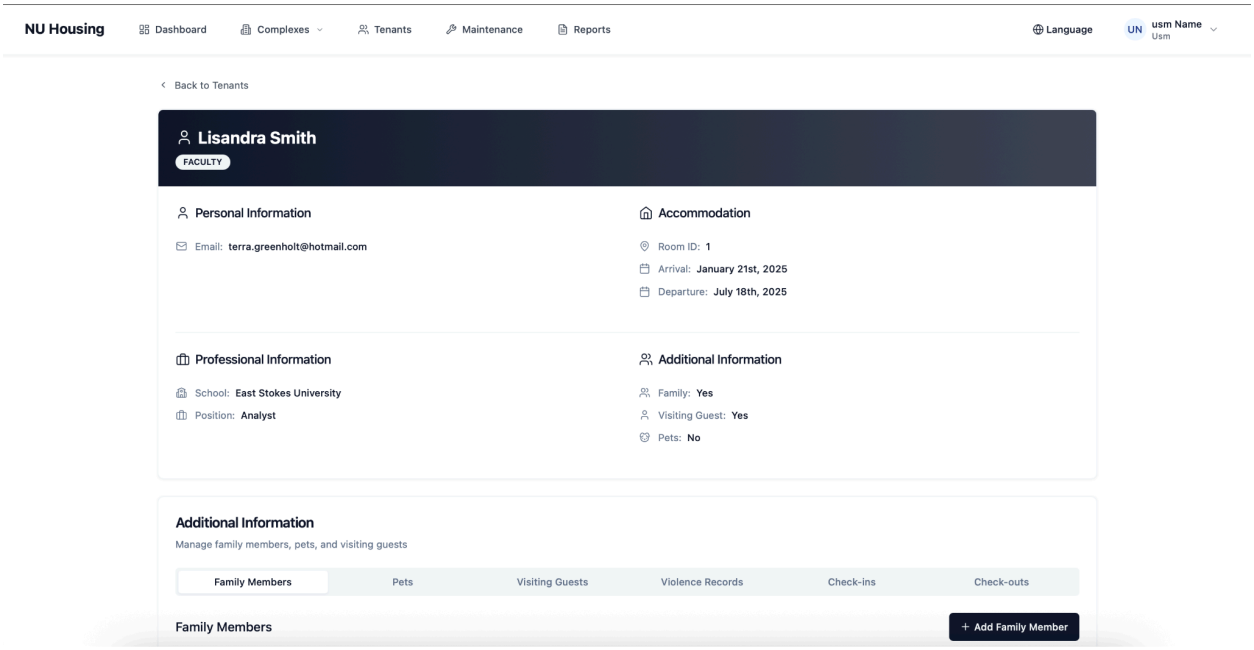


Figure 3. Tenant information page.

## c. Roles and features/Diagrams

As for the roles, as mentioned earlier, there are three of them: Admin (full CRUD access, report generation, contract management, integration settings, tenant access, complex access, check-in and check-out), USM (view personal data, request maintenance, violation tracking, report generation, tenant access, complex access, check-in and check-out), and DSS (has access to public data, can add users to the system). For the record, the latter does not have access to report generation or maintenance.

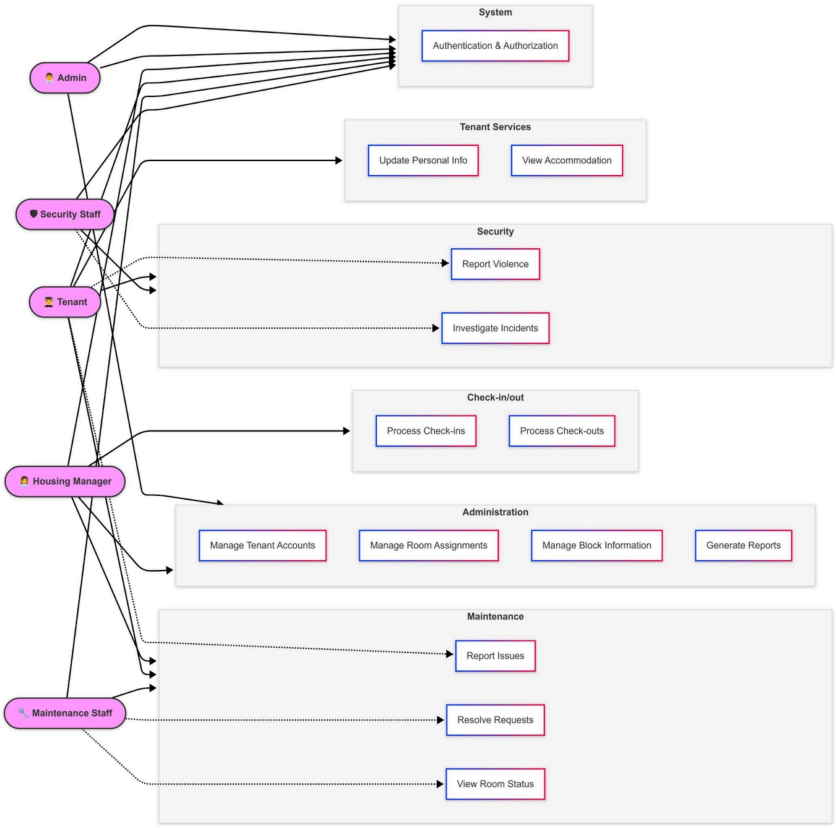


Figure 4. Use-case diagram

Figure 5. UML diagram.

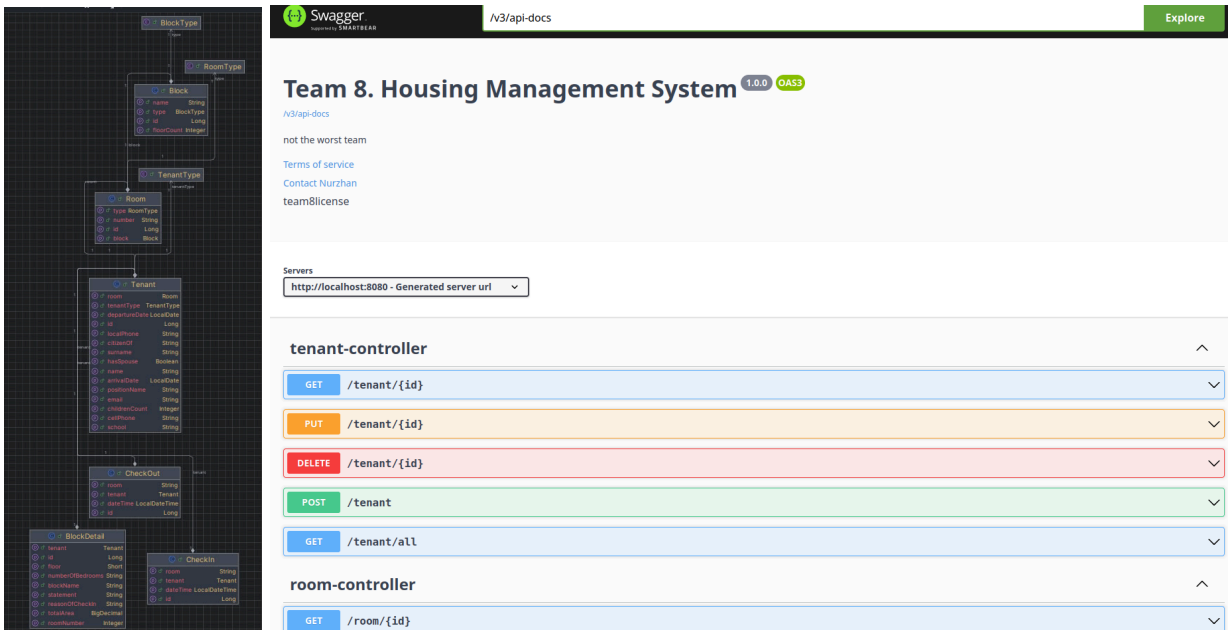


Figure 6. Backend API that is documented with Swagger.

**d. Third-party components**

We used some third-party tools into our project to improve its efficiency, functionality, and completion time. If we had to name the most significant components used, those would be Swagger and Google's Gemini.

Swagger was employed in the backend to document and interact with the APIs built using Spring Boot. Swagger was used as a user-friendly web interface that enabled us to visualize API endpoints, test them in real time, and validate the proper functionality of endpoints. Google's Gemini 2.0-flash model was the core of the AI-powered report generation. AI-powered chat enabled end-users, such as USM and DSS staff, to interact with the system through a user-friendly chat-based interface. Users could ask questions or request summaries in all three languages, and Gemini would return easy-to-understand responses from the HMS database. The prompt engineering included filtering instructions, context about the dataset, and the desired format of the response.

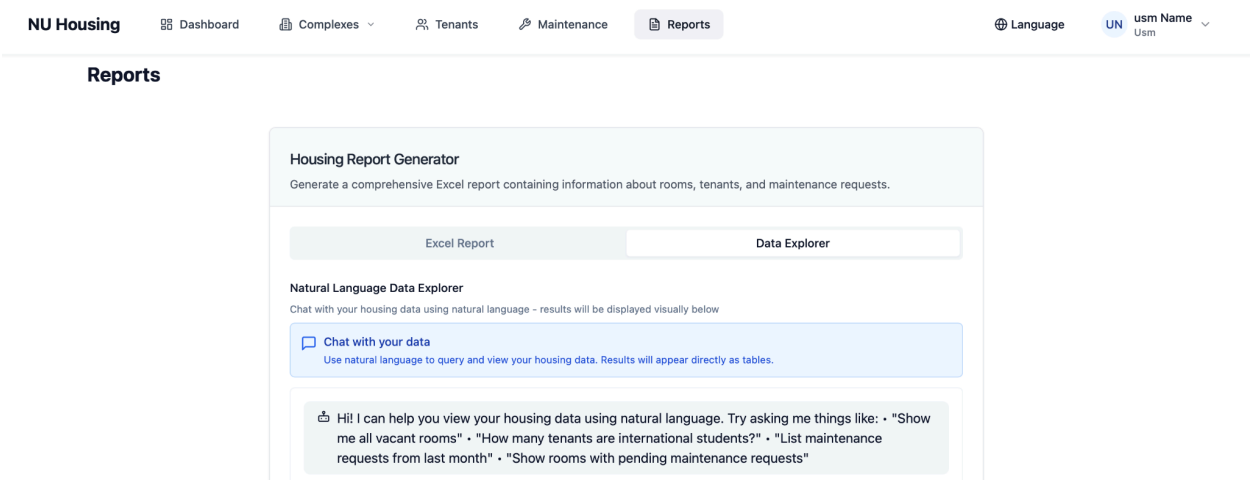


Figure 7. Chat-based report generation.

## 5. Project execution

### a. Project development

We developed the Housing Management Systems (HMS) over two semesters using the agile approach. The latter required us or was the result of constant interaction with stakeholders, and we changed the system in accordance with changing requirements. For example, we initially thought that both residents and administrators would be users, so we developed the system accordingly. However, later, after one of the consultations with USM, we realized our mistake and changed the requirements so that the system would be used exclusively by USM and DSS personnel. To identify the key features of HMS, we relied not only on customer interviews but also on secondary source analysis. The list we ended up with was: centralized data management to replace disparate Excel spreadsheets USM stuff uses; automated report generation using filter-based or AI-powered chat interfaces; multilingual support (English, Kazakh, Russian); and role-based access control for data security and appropriate access levels.

Regarding the key technologies used to implement the features mentioned above, we used React.js to build the frontend; this provided an intuitive user interface. On the other hand, Spring Boot was used for the backend to handle the business logic, API endpoints. Also, PostgreSQL was used to implement the database for reliable data storage/retrieval. Finally, AI integration was achieved using Google's Gemini 2.0-flash model, which enabled the creation of a dynamic chat-based report generation.

However, not everything went smoothly; as mentioned earlier, evolving requirements and AI integration were two major challenges we encountered while developing the system. We communicated frequently with stakeholders, which resulted in frequent changes in system

requirements, but Agile's iterative structure allowed us to remain flexible. As for the AI integration, we had to conduct multiple iterations to fine-tune the AI's responses; these iterations were manual and time-consuming.

**b. Team responsibilities**

For ensuring efficient teamwork we had frequent synchronous meetings both offline and online tracking tasks in Trello Kanban board and actively communicating via Telegram. In Telegram we asked each other about the progress and reported accordingly.

The team consisted of five members with diverse skill sets allowing us to distribute tasks fairly and work in harmony. For instance, Almaz and Adi were the front-end developers. Their task was implementing and designing the UI using React.js. Those two also ensured platform responsiveness, user-friendliness,and multilingual support capabilities. Nurzhan and Dana were back-end developers assigned to implement the server-side logic with Spring Boot. They managed database operations, implemented API routes, and provided data protection via JWT authentication and bcrypt-based hashing of a password. Lastly, Nurbek seamlessly integrated Gemini 2.0-flash into the system with the introduction of our custom chat-based reporting feature.

If we go into more detail about the apps we used to track progress, they were Trello, GitHub, and Telegram. The former was used for task management, sprint planning, and tracking progress. GitHub facilitated shared repository collaboration. Finally, as mentioned earlier, Telegram was our primary communication channel for daily updates.

**6. Evaluation**

**a. Project solution evaluation**

To evaluate the effectiveness of HMS in addressing the main problem defined in the introduction—optimizing housing operations for USM—we used different testing methods and checked with requirements and stakeholder feedback.

One of the testing methods was functional testing, in which we verified that each system feature (e.g., room assignment) performed as intended across various user roles. During user testing, we asked USM staff to interact with the deployed system and perform their daily tasks on it. Their experience/feedback gave us valuable insights into usability and functionality. Next, performance analysis helped us to evaluate the system responsiveness under different load scenarios; this was to ensure consistent performance when handling real-world data volumes. Finally, we tested our AI-chat interface evaluating its accuracy, response relevance, etc. However, we faced several challenges during development; that is, legacy excel-based databases

and inaccurate AI responses. USM's legacy Excel-based database was very unstructured and required some cleaning. Apart from that, it required creating a single structure for all excel sheets, which was time-consuming. Additionally, Gemini's AI explanations were sometimes overly verbose; the system was updated with prompt engineering to shorten responses and make them more accurate.

**b. Solution validation**

The evaluation process validated the HMS solution through three key outcomes. First, alignment with stakeholders needs was confirmed by positive feedback from the USM workers and successful functional testing. The latter showed that our system met all the main requirements of USM, which are associated with role-based data access, centralized record-keeping, and maintenance history tracking. Second, the AI reports implemented using the Gemini-based assistant were evaluated using various queries (e.g., "Show students in Block C whose contracts have expired," "List maintenance requests this month"). Third, USM staff used the platform to simulate daily operations over a one-week test period. Their approval of the workflow confirmed the soundness of the system design and that it effectively replaced the previous Excel-based approach.

**c. Collected data**

Users appreciated the intuitive interface and faster access to student housing data. AI features were seen as a bonus—especially useful for non-technical staff—but some requested even simpler phrasing. Most of the feedback was used to correct the namings of pages and other details according to USM's needs and preferences.

**7. Conclusion and possible future work**

**a. Key findings**

To sum up, the system we finished developing this semester successfully solved USM's main problem of streamlining their housing management operations. While they had to do some operations manually with the old system, our system automated them. Overall, our solution was to develop a fully functional web solution integrated with AI features. It is convenient for managing housing data and generating flexible reports, which was a major requirement of USM. However, before we started building the new system, we conducted a comprehensive analysis of the old system and also read a number of secondary sources. After collecting the complete requirements, customer expectations, and the current systems in use, we started designing and implementing our HMS.

During this process, we gained many valuable insights into project development and management. We learned the importance of early planning and requirement gathering, as well as the advantages of agile methodologies over waterfall ones. However, the most special part of our HMS is the integration of Google Gemini 2.0-flash model for AI-based reporting; we have not seen such a feature in other similar systems we read about. It made the system more flexible and improved its overall usability.

**b. Future plans**

Notwithstanding that our project was a definitive success, we have several ideas about possible improvements to the system. One of the features that was initially planned was a roommate allocation system for students. Students with similar interests and lifestyles would be assigned as roommates. During the project development, it was not our and staff's main priority. Due to that, we chose not to implement it. But in the future, this can be a welcome addition to the functionality of the system. It can decrease the number of students relocating and the number of complaints to the university student housing administrators. Additionally, we can add the tenants to the system as well. They can see their utility bills and manage maintenance requests. The billing can be also done through the website. Another addition is to integrate automated email responses for maintenance and violation records of the tenants. This will improve the communication between the tenants and the administrators.

**8. References**

Evo. (2023, December 4). *Housing Management Systems: How to Effectively Manage Your Portfolio*. https://evo-pm.com/insights/housing-management-systems-how-to-effectively-manage-your-portfolio/

freeCodeCamp. (2024, April 3). *How to Hash Passwords with bcrypt in Node.js*. freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-hash-passwords-with-bcrypt-in-nodejs/

Gupta, D. (2024, June 20). *Property Management System Implementation: 9-Step Guide*. The Whatfix Blog. https://whatfix.com/blog/property-management-software-implementation/

Jeffrey, A. H., Venkataraman, R. and Heikki T. (2015). *Modern Database Management* (9th ed.). Pearson. https://industri.fatek.unpatti.ac.id/wp-content/uploads/2019/03/167-Modern-Database-Management-Jeffrey-A.-Hoffer-V.-Ramesh-Heikki-Topi-Edisi-12-2016.pdf

Lau, E. (2024, February 27). *How to successfully implement a housing management system*. OmniLedger. https://www.omniledger.co.uk/how-to-successfully-implement-a-housing-management-system/

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson. https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf