Nazarbayev University

Computer Science Department

Final Report – Spring 2025

Title of the project:	NU Housing Management System
Team Members:	Yergali Tangirbergen, Temirlan Buzhukov, Ayazhan Mukhtar, Kanagat Kurmambayev
Project Advisor/Co-Advisors	Askar Boranbayev

Executive Summary (10%)

The Housing Management System (HMS) project was developed by Team 40 to address the inefficiencies and errors in the current manual processes used to manage university housing at Nazarbayev University.

The problem lay in the absence of an integrated, scalable platform to handle resident data, room allocation, furniture tracking, billing, and maintenance requests. Manual operations led to frequent delays, miscommunication, and inaccuracies.

The key objectives were:

- Create a secure, scalable, multilingual platform for administrators and residents.
- Automate room assignments, maintenance tracking and reporting.
- Ensure compliance with data protection standards, including GDPR and Kazakhstan's Personal Data Law.

The methodology followed an Agile Scrum framework, with two-week sprints, early UI/UX prototypes, continuous integration using Docker, and backend/frontend development based on Spring Boot, React, and PostgreSQL.

The main results achieved:

- A fully functioning backend with RESTful APIs.
- A user-friendly React frontend for residents and administrators.
- Database schema and services covering residents, rooms, leases, maintenance, and billing.
- Containerized deployment using Docker and initial AWS setup.
- Early implementation of multilingual support and secure authentication.

The project fully aligns with the design, implementation, and evaluation of a computing-based solution, showcasing full-stack system design, secure software development practices, and real-world software engineering principles.

Introduction (10%)

Managing university housing had become increasingly **cumbersome, inefficient, and error-prone** due to reliance on **manual, disconnected processes**. Without a unified platform, tasks such as resident tracking, room allocation, and maintenance management suffered from frequent miscommunication and long processing times, negatively impacting both staff and residents .

The **motivation** behind the project was to streamline and automate all housing-related operations, minimize errors, reduce administrative burden, and offer a **seamless, secure, and scalable experience** for users across the university.

The **significance** of the Housing Management System is multifaceted:

- Improves operational efficiency and transparency.
- Protects sensitive resident data in line with GDPR and Kazakhstan’s data protection laws.
- Supports the university's growing population with scalable architecture.

- Provides multilingual accessibility for English, Kazakh, and Russian speakers.

The **proposed solution** is a **three-tier web system**:

- **Frontend**: Developed with **React**, offering dynamic, role-specific interfaces.
- **Backend**: Developed with **Spring Boot**, exposing RESTful APIs for housing operations.
- **Database**: Powered by **PostgreSQL**, storing residents, properties, billing, and reports.
- **Deployment**: Dockerized applications hosted on scalable cloud infrastructure.

This report is organized into the following sections: background and literature review, project and product description, system design, current implementation, future plans, and ethical and legal considerations.

Background and Related Work (15%)

Prior Research and Existing Solutions

Managing housing resources has long relied on manual or fragmented systems, as documented in studies on higher education housing inefficiencies. Many universities experience issues such as:

- Delays in maintenance handling,
- Miscommunication during check-ins/check-outs,
- Inefficient utility billing,
- Difficulty maintaining accurate furniture inventories.

Solutions like ASHAMS (Abdus Salam Hall Accommodation Management System) demonstrate that **web-based accommodation platforms** significantly reduce administrative burden and errors(Bhowmik & Riaz). Furthermore, modern CRM-like housing management systems are often cited for improving transparency, especially when integrating payment portals and real-time tracking.

Comparison of Approaches

- **Manual Systems**: High error rates, slow response, lack of real-time updates.
- **Fragmented Software Solutions**: Partial automation but often lack integration (e.g., separate systems for billing and room assignment).
- **Fully Integrated Web Systems (Our Approach)**: Seamless management of all tasks, secure access control, compliance with data protection standards, multilingual support.

Our project combines the best elements of scalable modern web systems—full CRUD APIs, responsive interfaces, modular microservices architecture—while specifically adapting to **Kazakhstan's regulatory** and **multilingual environment**(Martysheva A., 2024).

Alignment with Computing-Based Solutions

The NU Housing Management System is designed and developed by applying principles from computing disciplines:

- **Software engineering** (Agile development, modular design),
- **Database management** (PostgreSQL relational schemas with data backup),
- **Cybersecurity** (SSL/TLS encryption, RBAC access control, audit trails),
- **Scalable systems architecture** (Docker containerization, AWS hosting plans),
- **Compliance with legal frameworks** (GDPR, Kazakhstan Personal Data Law).

By integrating existing best practices and extending them with domain-specific customizations, the HMS project not only solves existing university housing issues but also stands as a robust computing-based system ready for real-world deployment.

Project Approach (20%)

Frontend Development

1. Overview

The frontend of the Housing Management System (HMS) is developed using React.js, a modern JavaScript library for building user interfaces. The frontend interacts with the backend API to manage housing assignments, payments, incidents, and maintenance requests. The project structure is organized into several functional pages and components, each serving a specific purpose. The system is designed to offer a responsive and intuitive user experience.

2. Frontend Roles

Use case

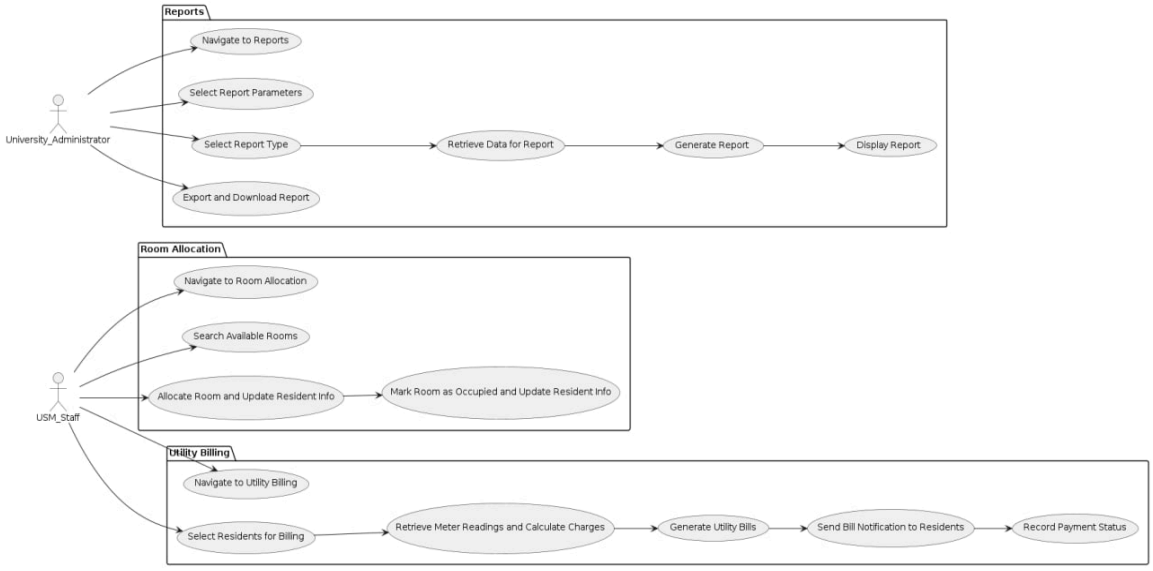


Figure 1. Use case diagram

1. Room Allocation
   - Primary Actor: USM Staff
   - Preconditions: Resident data must be available in the system.
   - Main Scenario:
     - The staff searches for available rooms in the system.
     - Based on resident preferences, the staff assigns a room to the resident (e.g., gender, special needs, room availability).
     - The system updates the room’s status and the resident's assignment.
   - Postconditions:
     - The room is marked as occupied.
     - The resident’s profile is updated with the room details.
2. Generate Utility Bills

- Primary Actor: USM Staff
- Preconditions: Utility meter readings must be entered and updated in the system.
- Main Scenario:
    - The staff selects a resident or a group of residents for billing.
    - The system calculates the utility charges based on the room's size and usage.
    - The system generates a bill and notifies the resident.
- Postconditions:
    - The resident receives the utility bill.
    - The payment status is updated and recorded in the system.

3. View Reports
    - Primary Actor: University Administrator
    - Preconditions: The system must have up-to-date data on residents, rooms, and utility consumption.
    - Main Scenario:
        - The administrator selects the type of report to be generated (e.g., room occupancy, utility consumption).
        - The system generates and displays the requested report, which can include data visualizations such as tables or charts.
    - Postconditions:
        - The administrator reviews the generated report and can export it if needed (e.g., as PDF or Excel).

Sequence Diagram:

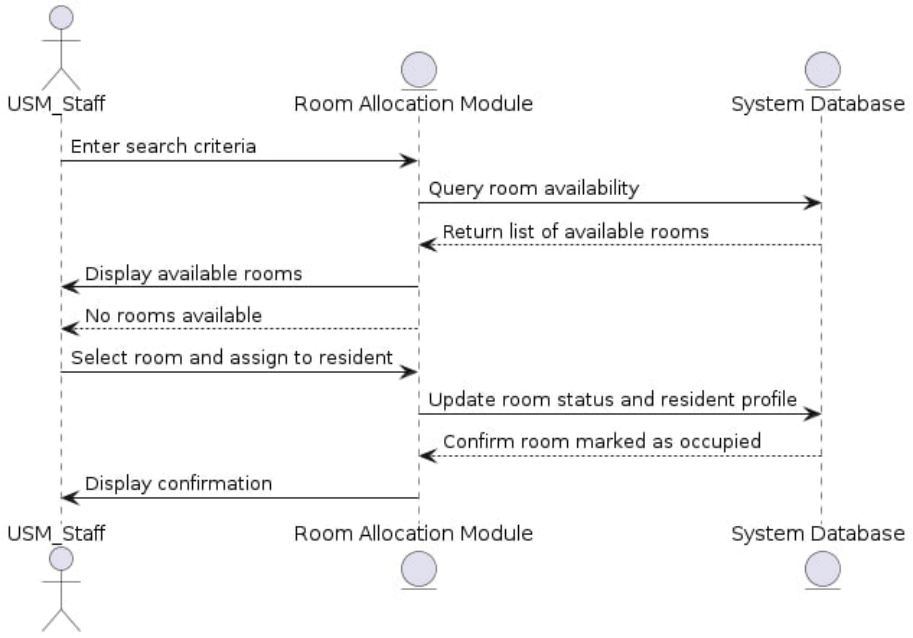


Figure 2. Sequence Diagram

System UI

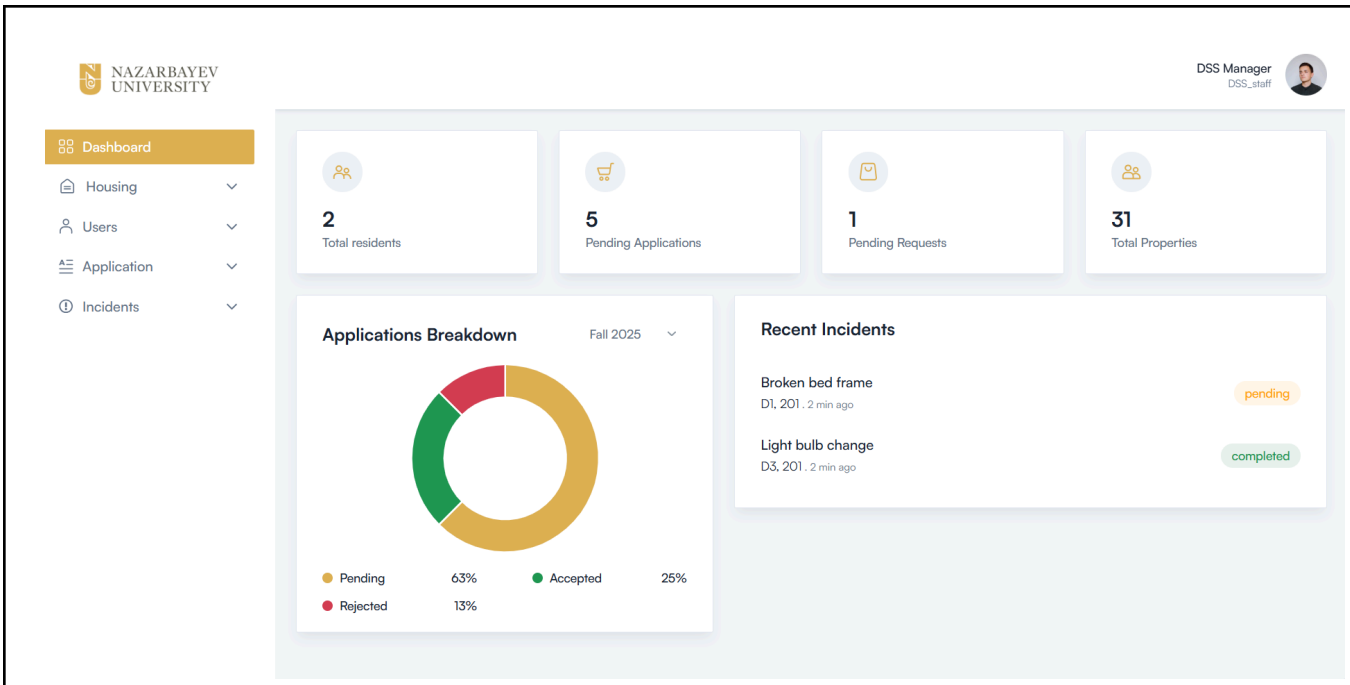


Figure 3. DSS_Staff - Dashboard

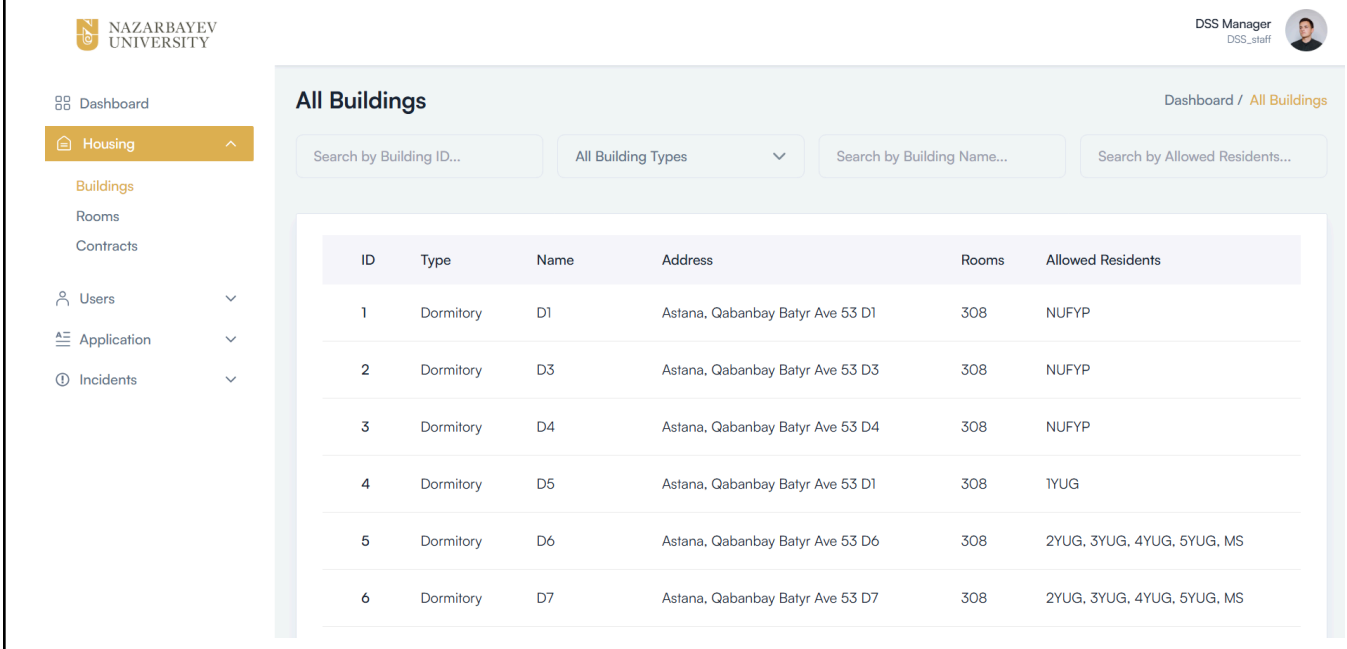


Figure 4. DSS_Staff - Buildings

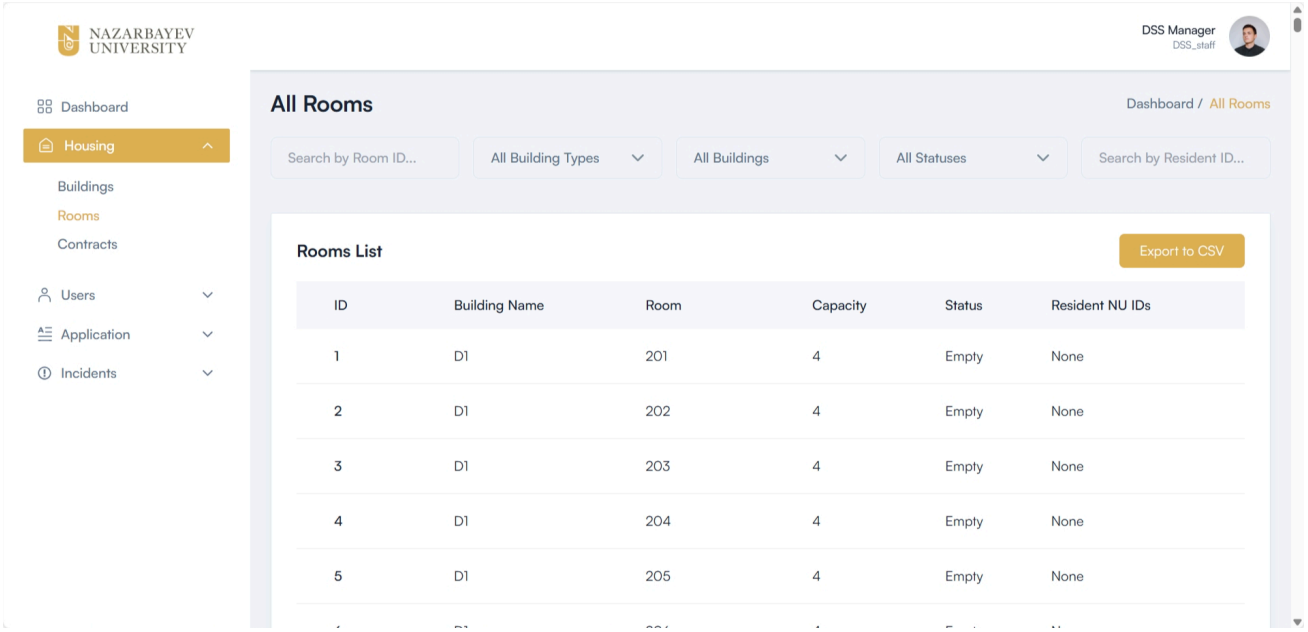


Figure 5. DSS_Staff - Rooms

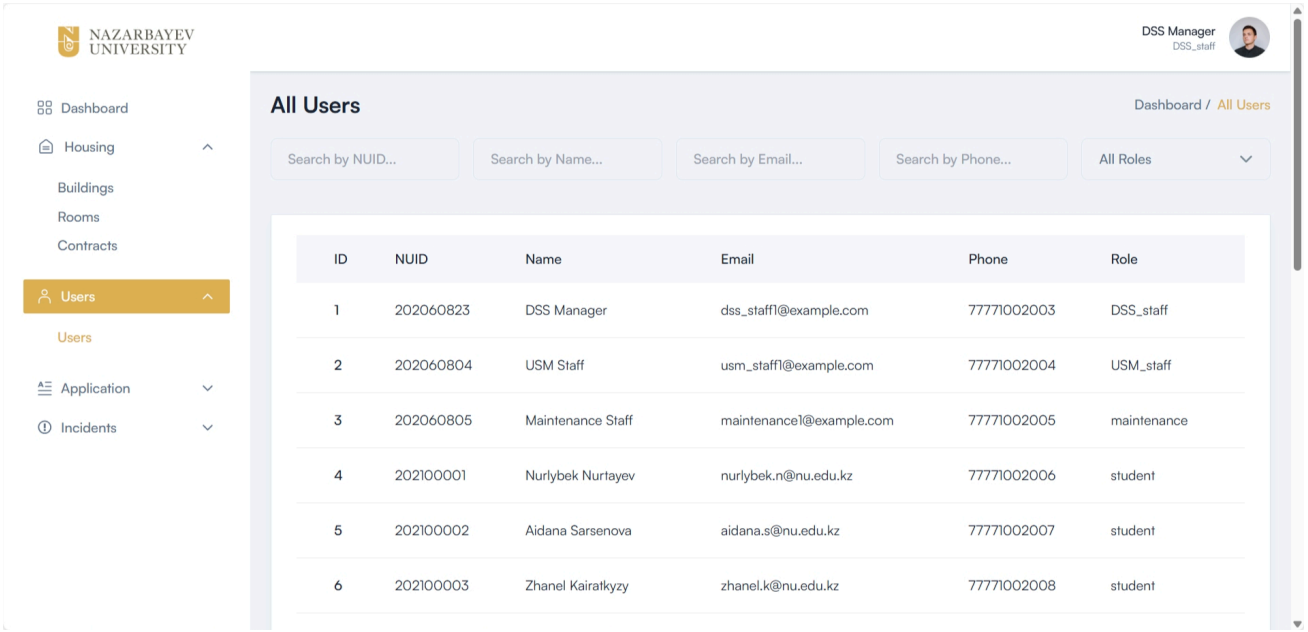


Figure 6. DSS_Staff - Users

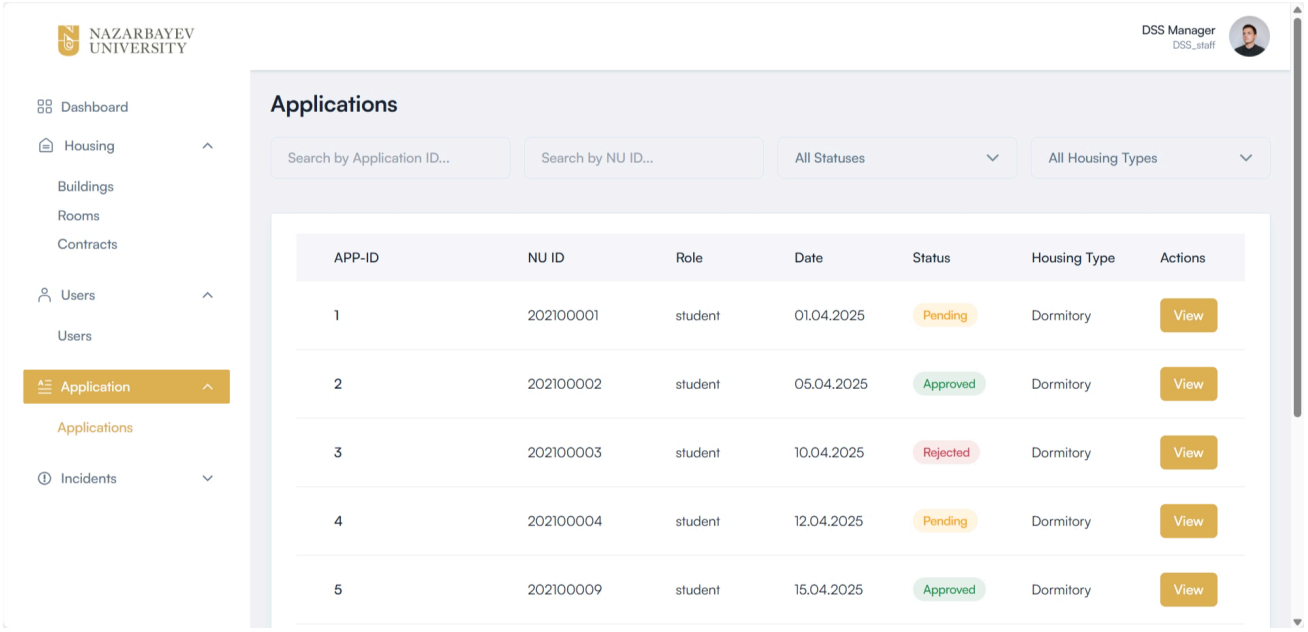


Figure 7. DSS_Staff - Received Residence Applications

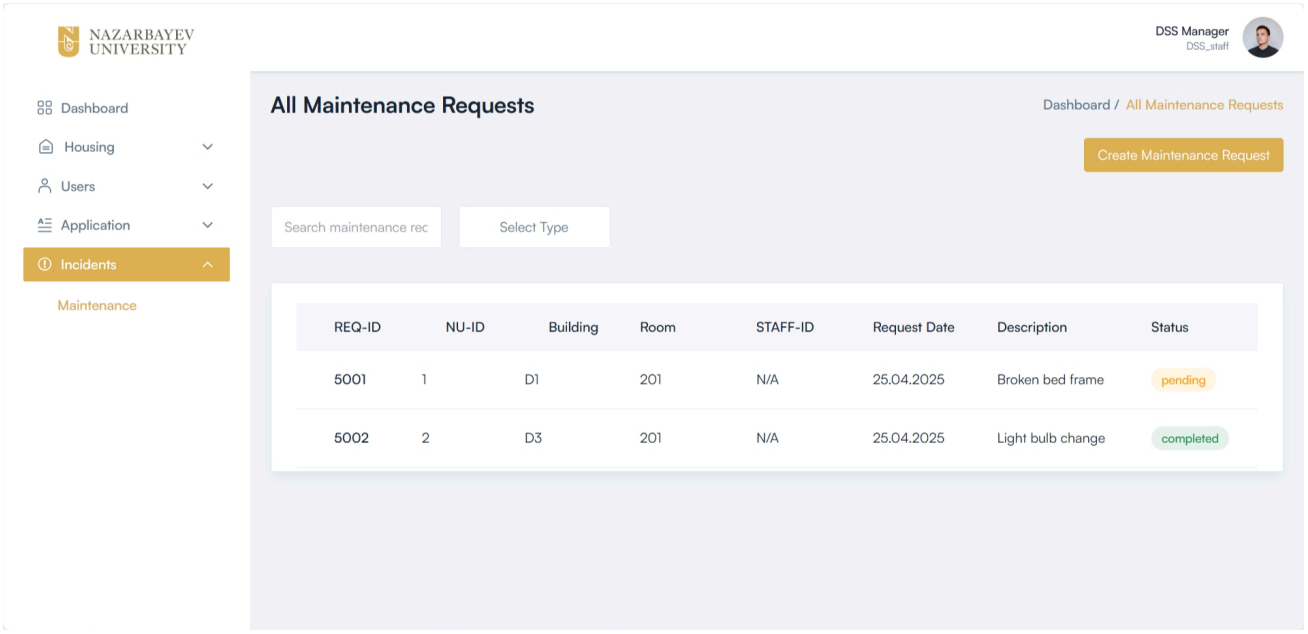


Figure 8. DSS_Staff - Maintenance Request

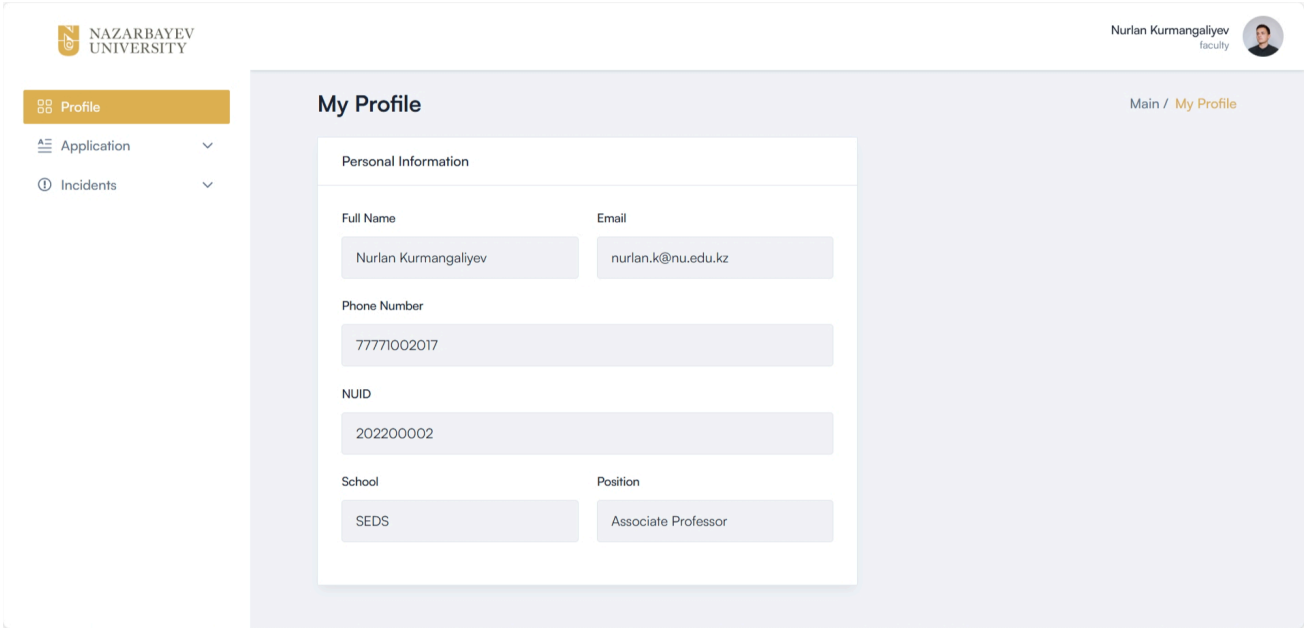


Figure 9. Faculty - Profile

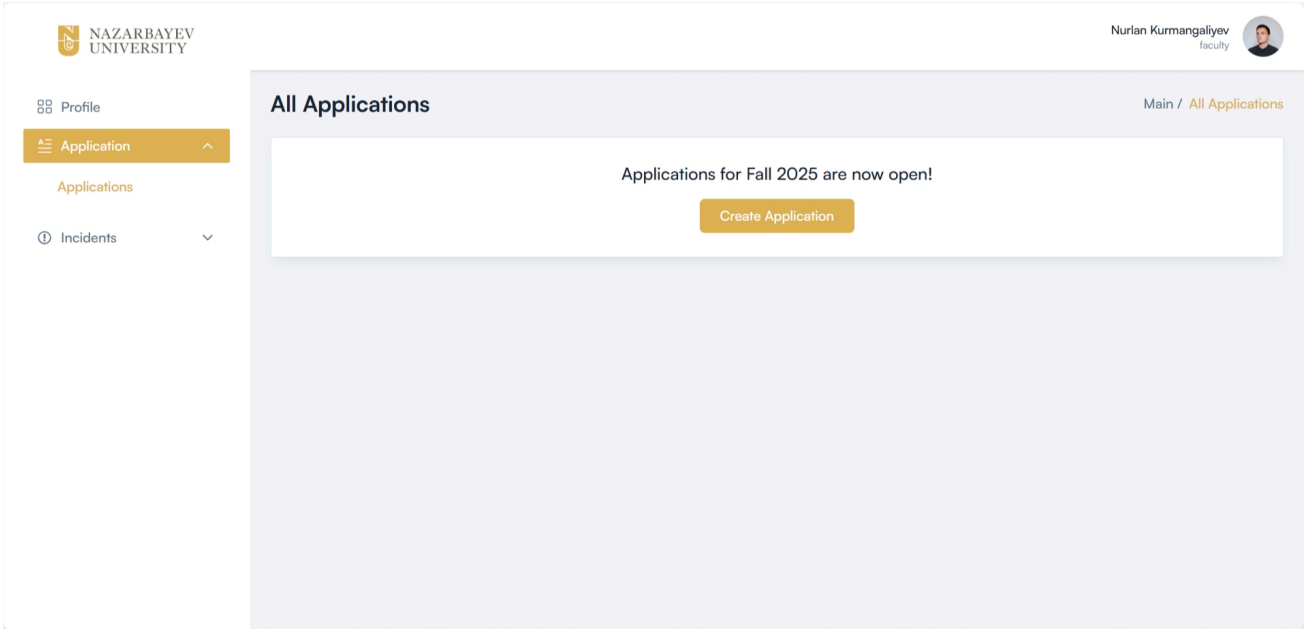


Figure 10. Faculty - Application page

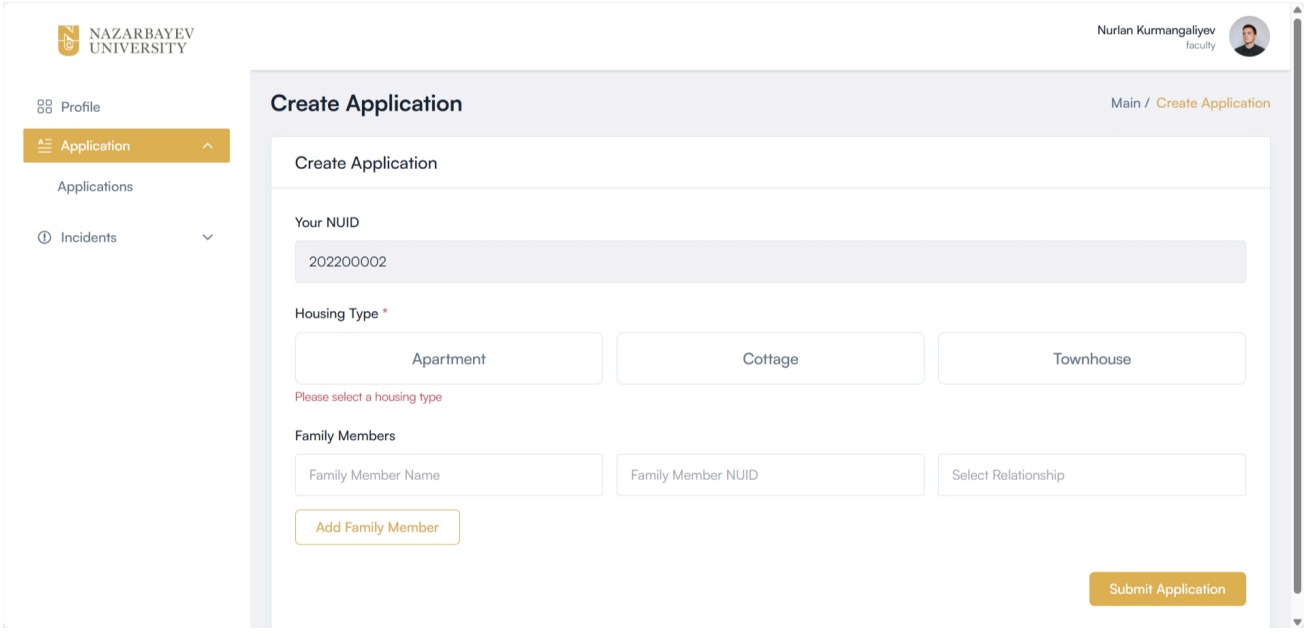


Figure 11. Faculty - Application Form

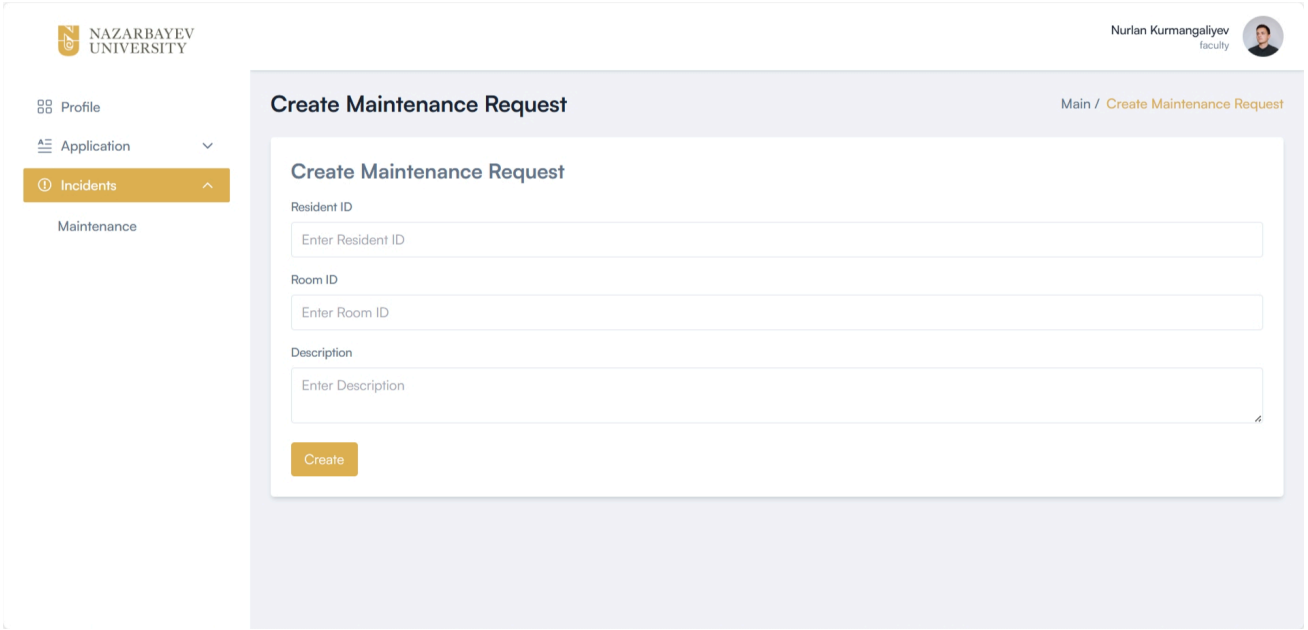


Figure 12. Faculty - Maintenance Request Form

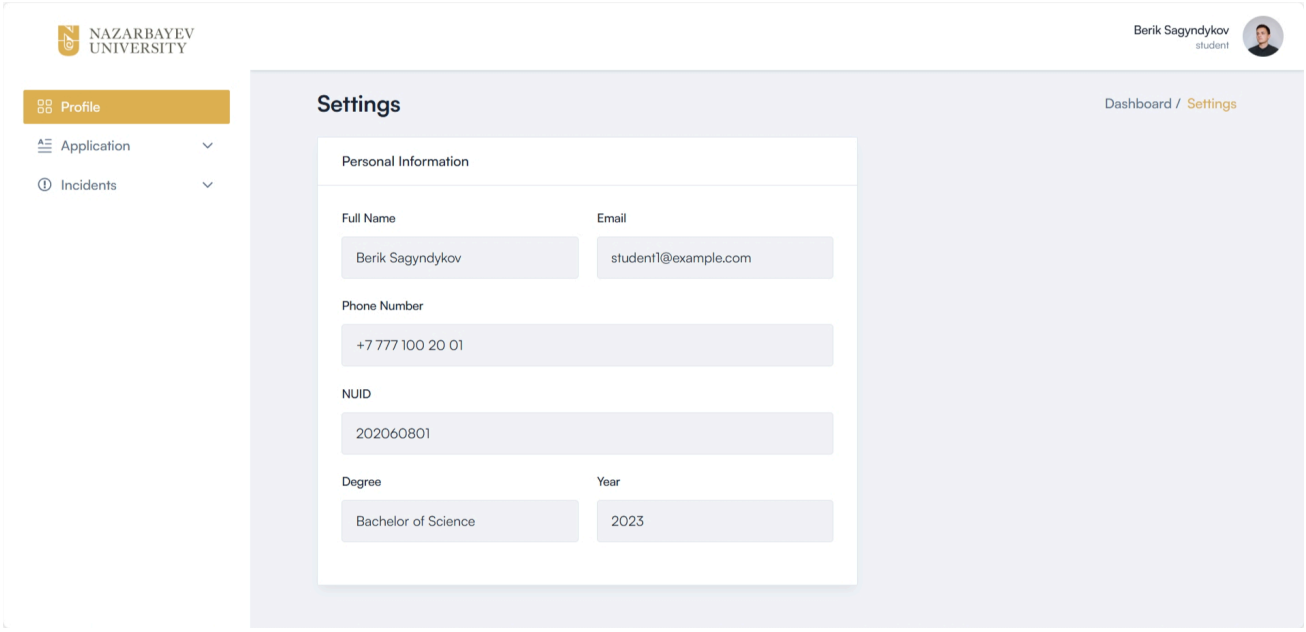


Figure 13. Student - Profile

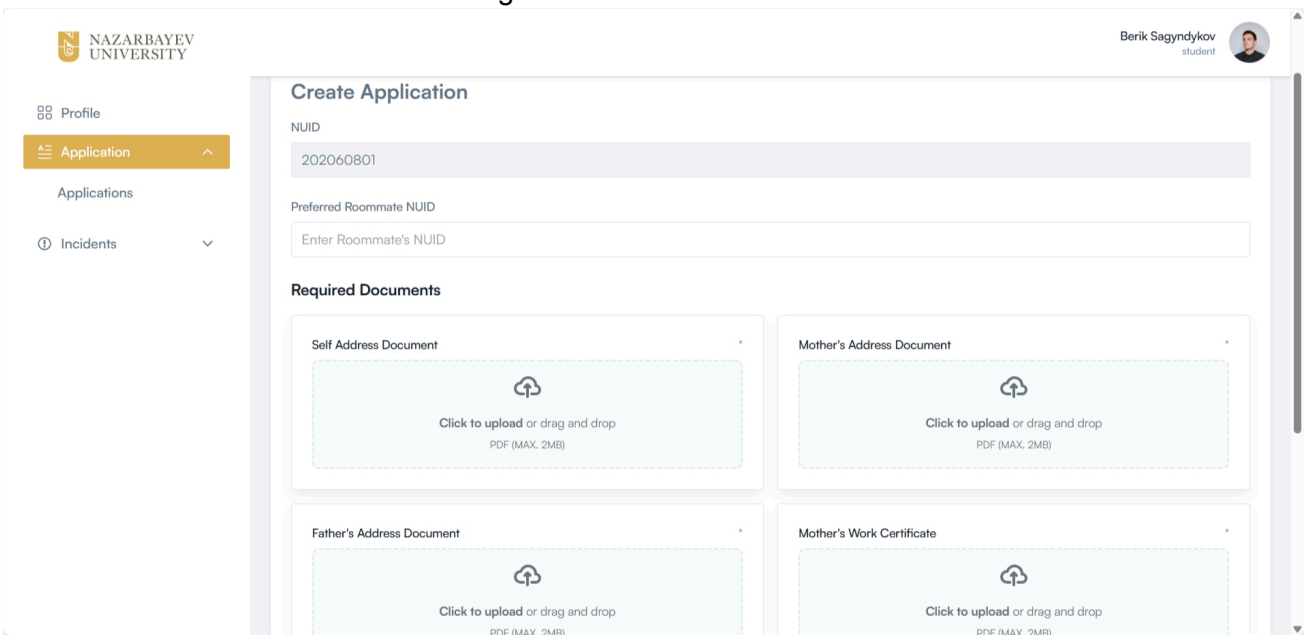


Figure 14. Student - Application Form

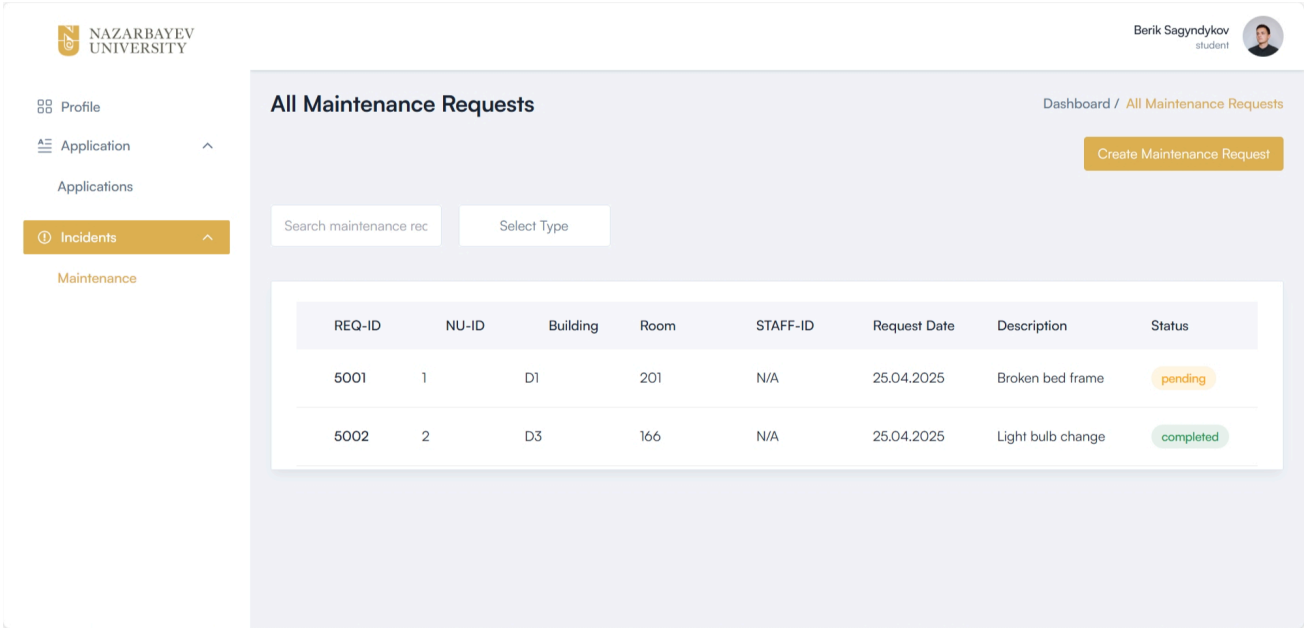


Figure 15. Student - Maintenance

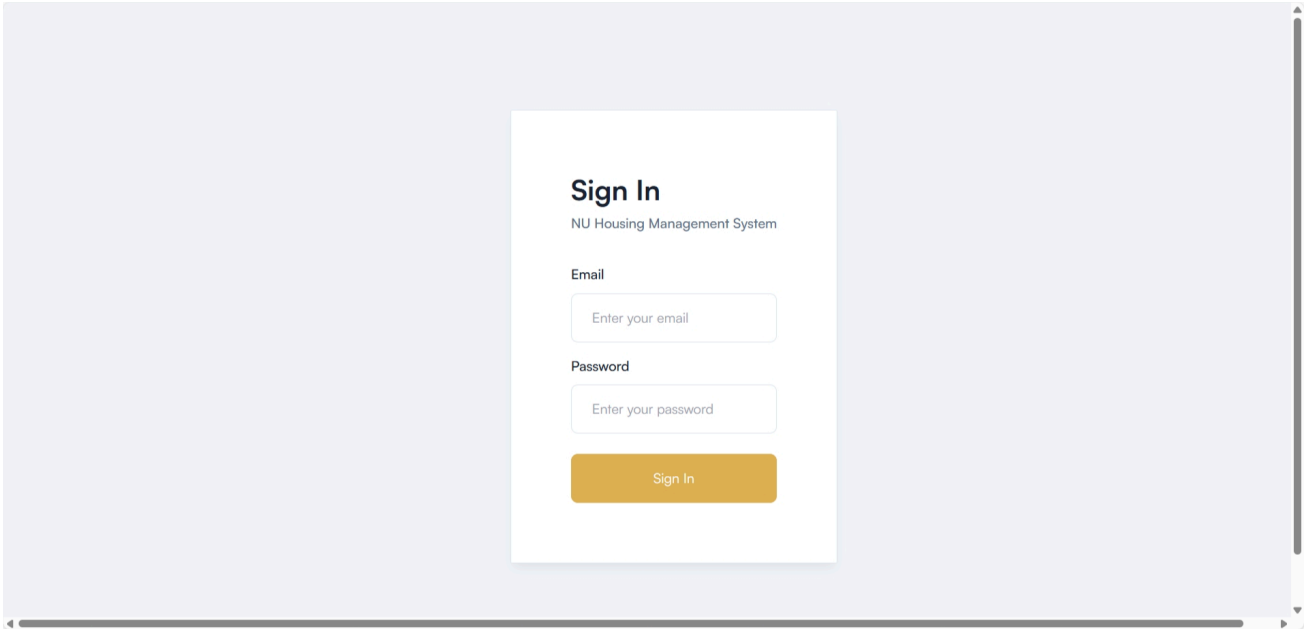


Figure 16. Login Page

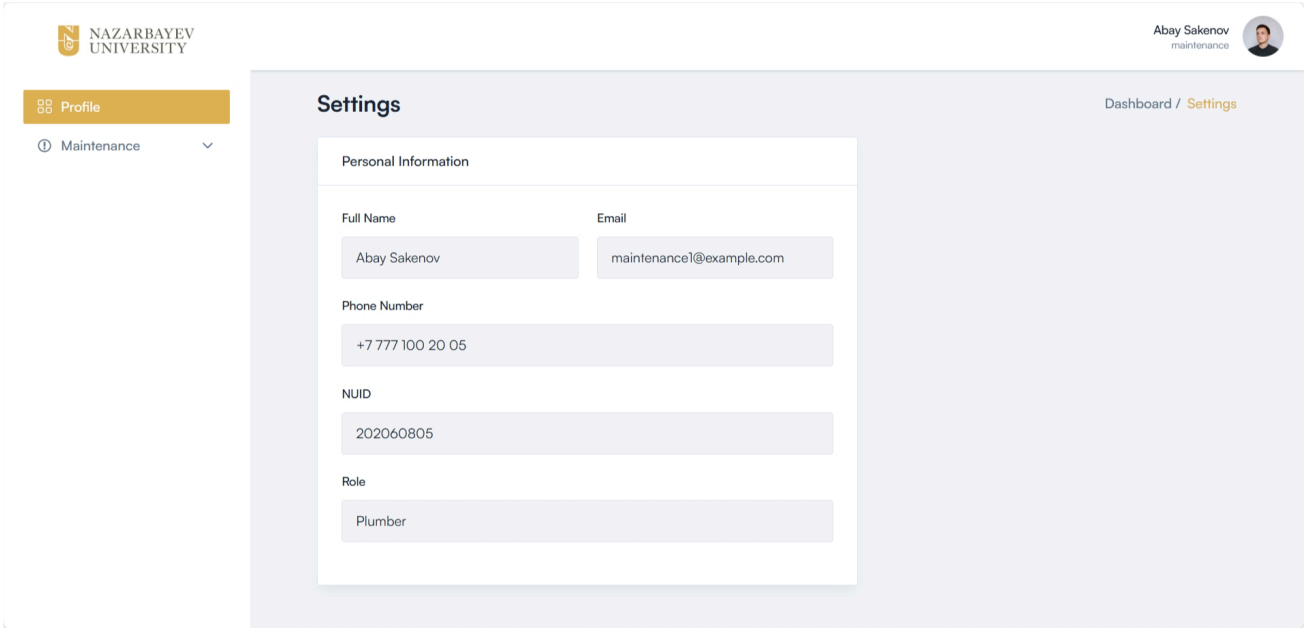


Figure 17. Maintenance - Profile

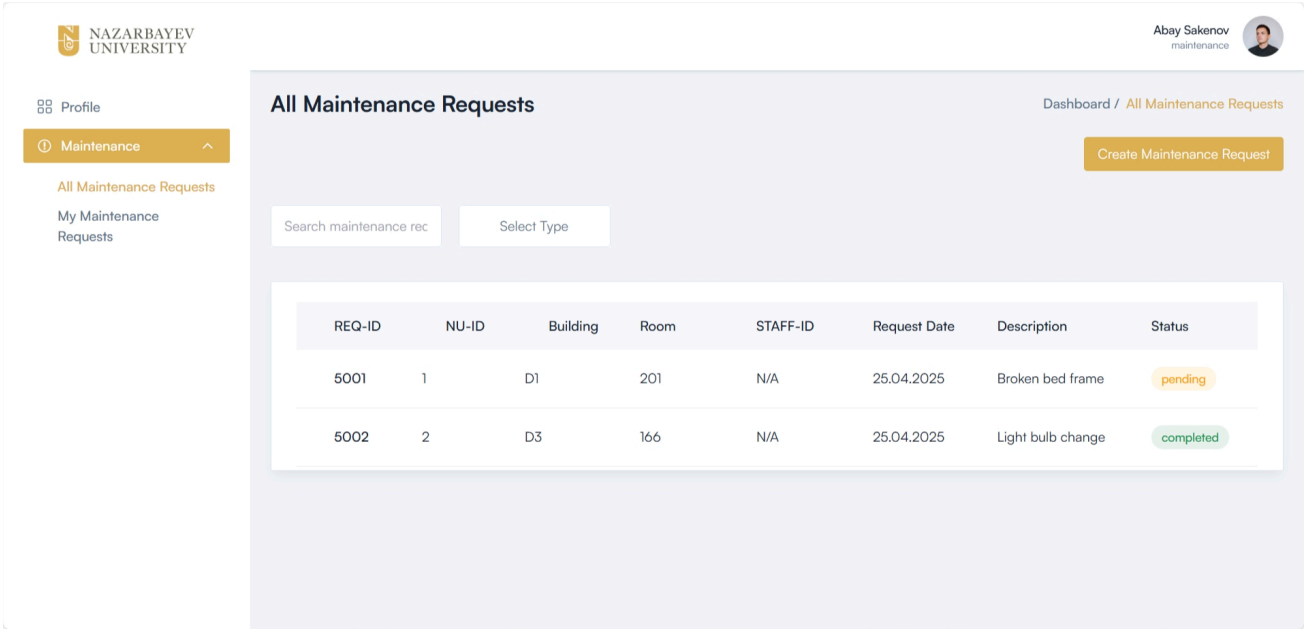


Figure 17. Maintenance - All Requests

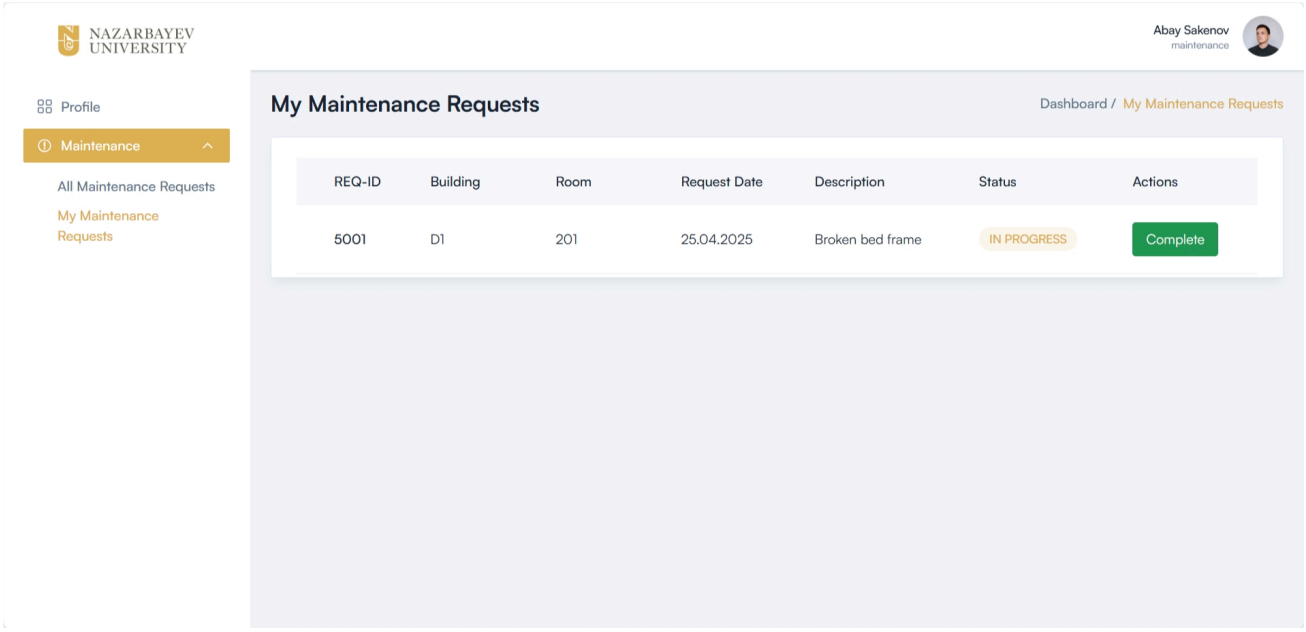


Figure 18. Maintenance - My Requests

A. Pages and Components The frontend is structured around various pages and components, each representing different aspects of the housing management system. These pages include:

- FormNewBuilding.tsx: Handles the creation of new buildings in the system. It includes a form that allows administrators to input building details (e.g., name, location, capacity).
- Housing: Manages housing assignments, room availability, and assigning rooms to residents.
- Incidents: Related to incidents or issues requiring maintenance or attention. Residents and admins can view, report, and manage incidents (e.g., broken plumbing, heating issues).
- UIElements: Contains reusable components such as buttons, form elements, dropdowns, and layout elements, ensuring consistency across the UI.
- Users: Manages user profiles, roles, and authentication. Admins can create or manage user accounts, and residents can update their personal information.
- Settings.tsx: A page for managing system settings, accessible primarily to administrators. It includes user permissions, housing rules, and system-wide configurations.

B. Key Features and Functionalities Implemented

- Room Assignment: The frontend allows residents to view available rooms and get assigned based on availability and preferences (e.g., gender, special needs).

    - Executed: Room assignment functionality is integrated with the backend, and users can view available rooms and submit preferences.
- Payment Management: The frontend enables residents to make payments for housing (e.g., rent, utilities) through an integrated payment gateway.
    - Executed: Payment UI has been developed. Users can view payment history and upcoming payments, though full payment integration is still pending.
- Maintenance Requests: Residents can submit maintenance requests specifying issues and room details.
    - Executed: The functionality for submitting maintenance requests is live, with backend integration for request tracking.
- Incident Reporting: Residents and admins can report and track incidents, such as broken appliances or infrastructure issues.
    - Executed: The UI for incident reporting is implemented, though the full workflow for assigning incidents to maintenance staff is still being refined.
- User Profiles: Residents can manage their profiles and view housing information, such as room assignments, payment status, and maintenance requests.
    - Executed: Profile management is implemented, but role-based access control (e.g., admin vs. resident) requires further adjustments.
- Responsive Design: The frontend is fully responsive, supporting mobile, tablet, and desktop devices.
    - Executed: The application is fully responsive, and UI elements adjust correctly across different devices using Tailwind CSS.

C. Third-Party Libraries and Integrations

- React.js: Used for building the frontend UI, enabling dynamic components and easy maintenance.
    - Executed: Fully integrated for the entire frontend.
- Tailwind CSS: Provides utility-first CSS classes for rapid UI development.
    - Executed: Tailwind is set up and applied to ensure consistent styling across pages.
- Axios/Fetch: Used to make HTTP requests to the backend API for data retrieval.
    - Executed: API calls are set up for interacting with the backend (e.g., room assignments, payments, maintenance)
- React Router: Manages the routing and navigation of the application.

    - Executed: React Router is implemented to navigate between pages (e.g., dashboard to settings).
- Formik / React Hook Form: These libraries handle form data and validation.
    - Executed: Forms for room assignment, incident reporting, and user profile updates are managed with Formik or React Hook Form.

BackEnd Development

Software/Hardware Architecture

The backend of the Housing Management System (HMS) is a robust, scalable Spring Boot application that provides a RESTful API to manage university housing operations, including properties, users, leases, maintenance requests, application forms, and reports. It follows a layered architecture to ensure separation of concerns and maintainability, consisting of:

Layer	Description
Framework	Spring Boot 3.x with Spring Data JPA and Hibernate for ORM.
Database	PostgreSQL 16, supporting advanced features like JSONB for report storage and enums for status/role management.
API Layer	RESTful endpoints with JSON responses, documented using OpenAPI/Swagger (`@Tag`, `@Operation`).
Authentication	Session-based authentication (`HttpSession` in `DashboardController`), with potential for future JWT/Spring Security integration.
Deployment	Dockerized using Docker and Docker Compose, utilizing a custom network (`hms_network`) and persistent volume (`hms_postgres_data`).

The backend leverages inheritance hierarchies (`BaseUser`, `BaseProperty`) to ensure reusability and consistency across user and property operations. This containerized model allows the backend

to be deployed both on-premises and in cloud environments like AWS ECS, Google Cloud Run, or Azure, with scaling opportunities via Kubernetes.

Algorithms, Workflows, and Services

The backend defines clear business workflows and algorithms that encapsulate university housing processes:

- **Lease Management**: Implemented through `LeaseService`, linking users and properties with full lifecycle support including creation, renewal, termination, and family member management.
- **Maintenance Requests**: Tracked and managed by `MaintenanceRequestService`, supporting submission, assignment to staff, status updates, and completion tracking.
- **Dashboard Metrics**: Provided by `DashboardService`, returning real-time counts of active leases and other user-specific data.
- **Property and User Management**: CRUD operations, type-based filtering, counting, and complex queries are shared across services like `StudentService`, `HousingManagementService`, and `DormitoryRoomService`.

Each service is implemented using transactional boundaries (`@Transactional`), consistent CRUD patterns, and Data Transfer Objects (DTOs) for clean communication.

Roles, Features, and Use Case Overview

The HMS backend supports diverse user roles mapped to entity classes inheriting from `BaseUser`:

Role	Features
Student	Apply for housing, view roommates, manage leases.
Teacher	Manage personal and family member data.
Housing Management Staff	Manage properties and leases, assign blocks.
Maintenance Staff	Handle maintenance requests.
Department of Student Services (DSS)	Review and process housing applications.

Core backend features include:

- Managing multiple property types (DormitoryRoom, Cottage, CampusApartment, OffCampusApartment, Townhouse).
- Full lease lifecycle management (creation, renewals, terminations).
- Handling student housing applications and file uploads.
- Managing maintenance request workflows.
- Generating custom reports stored as JSONB.

Representative workflows supported:

- Student submits housing application → DSS reviews → Lease created.
- Tenant submits maintenance request → Assigned to staff → Completed and marked paid.

Third-Party Components and Their Integration

Component	Purpose	Integration
Spring Boot 3.x	Application framework and DI container	Maven dependencies
Spring Data JPA + Hibernate	ORM and database access	Annotated entities, repositories
PostgreSQL 16	Relational DB with JSONB support	Docker service, JDBC integration
Swagger (OpenAPI)	API documentation	`springdoc-openapi-ui`, annotations
Docker & Docker Compose	Deployment and orchestration	Dockerfile, docker-compose.yml
JUnit 5 + Testcontainers	Testing environment	Unit and integration tests

These components were used out-of-the-box without reimplementation. Integration involved configuration via properties files (`application.properties`) and environment variables (e.g., `SPRING_DATASOURCE_URL`).

Project Team Functioning and Effectiveness

The backend team worked in a highly structured and effective process:

- **Version Control**: GitHub was used for managing the source code with dedicated branches and pull request reviews.

- **Issue Tracking**: GitHub Projects (Kanban) were used to track features, bugs, and sprints.
- **Development Practices**:
    - Consistent usage of services and DTO patterns.
    - Code modularity and reusability through `BaseUser` and `BaseProperty` inheritance.
    - Swagger annotations ensured API clarity and maintainability.
    - Weekly sprint reviews and retrospectives for continuous improvement.
- **Deployment and Testing**:
    - The backend was containerized and tested via local Docker Compose orchestration.
    - Future-proofing considered scaling via Kubernetes, AWS ECS deployments, and integrations with monitoring tools like Prometheus and Grafana.

Thus, the project team delivered a scalable, modular, production-grade computing-based solution, meeting both functional and non-functional requirements.

Use Case Diagrams

Lease Workflow (Student Housing Process)

```
Unset
[Student] --> (Submit Housing Application)
(Submit Housing Application) --> [ApplicationFormController]
```

```
[ApplicationFormController] --> [DSS Staff]
[DSS Staff] --> (Review Application)

(Review Application) --> (Create Lease)
(Create Lease) --> [LeaseController]
```

Maintenance Request Workflow

```
Unset
[Tenant] --> (Submit Maintenance Request)
(Submit Maintenance Request) --> [MaintenanceRequestController]

[MaintenanceRequestController] --> [Maintenance Staff]
[Maintenance Staff] --> (Accept and Assign Request)

(Accept and Assign Request) --> (Mark as Completed)
(Mark as Completed) --> [MaintenanceRequestService]
```

Project Execution (15%)

Over the course of two semesters, our team worked on developing the NU Housing Management System (HMS). The project moved through several stages — from planning and individual module development to integration and final deployment. We started by carefully outlining our goals, both functional and non-functional, to match the real needs of university housing. These early requirements helped guide our decisions throughout the entire process.

Fall 2024

During the first semester, our focus was entirely on preparation. We spent time understanding what the stakeholders needed, outlining clear system requirements, and planning how the system should work. We also worked alongside another team that was building a similar housing solution — this collaboration helped us align our plans, especially when it came to the user interface and overall system design.

We defined user roles, mapped out workflows, and created ER diagrams to guide our architecture. This phase helped us figure out exactly what the project should include, focus on the most important parts, and build it in a way that would be easier to update and grow later on.

Spring 2025

The actual building phase began in Spring 2025. With four people on the team, we divided responsibilities based on our strengths and project needs. Three of us focused on backend development, while one teammate took the lead on frontend integration and helped bridge the gap between both ends.

Backend:

Our team implemented a **modular PostgreSQL schema** that models core entities such as users, leases, and properties. At the center of the property structure is a shared property_base table, with additional tables branching off for specific types like dormitories, apartments, cottages, and off-campus housing. These are all connected through foreign keys, which helped us keep things organized and easy to expand.

For users, we created a central user_base table and added role-specific tables for students, faculty, and staff. This setup gave us the flexibility to assign roles and handle user-specific actions more easily. Leases act as the bridge between users and properties, storing key info like move-in and move-out dates, contract status, and payment details — all with proper checks in place to keep the data reliable.

Some of the challenges we faced:

Challenge: Real Excel files from the university contained inconsistent formats (e.g., block names in different languages, missing room numbers, or mixed values).

Decision: We implemented fallback logic in Python scripts to match and clean data, used mappings for block names, and added placeholder data when testing data were missing. Although importing was a small part of our backend scope, this decision ensured cleaner development datasets.

Challenge: Disagreements within the team

Decision: Lots of meeting with the team, lots of discussions.

This backend foundation served as the system's core, enabling frontend features like lease views and resident search.

Frontend

While the backend was the main focus, one of our team members worked on building frontend pages — mainly for displaying lease information and user profiles — using a React framework. That person also coordinated with the other team to make sure the frontend communicated correctly with our backend APIs and followed the agreed design.

We used **Jira** for sprint planning and progress tracking. Our team maintained weekly sync-ups, frequently reassigning tasks based on complexity and workload:

Evaluation (20%)

To evaluate the Housing Management System (HMS), a survey was conducted with 14 participants. The survey aimed to assess usability, performance, and overall experience with the system.

In total, 14 participants underwent the survey, representing various user roles:

- 10 students: These participants assessed the system from a student perspective, evaluating features like room assignments, maintenance request submissions, and general usability.

- 1 faculty member: Provided feedback on the system from an administrative standpoint, including contract management and room allocation.

- 2 DSS (Department of Student Services): Focused on their roles in managing housing assignments, tracking resident data, and ensuring smooth operations.

- 1 maintenance staff member: Provided feedback on the handling of maintenance requests, task assignments, and incident tracking.

Key Survey Insights:

- 85% of participants found the system easy to navigate.

- 50% felt neutral about the system's response time, while the other 50% were satisfied.

- 85% rated their overall experience with the HMS as better than the previous methods (Google Forms for students, Excel for administrators).

The complete set of questions, along with visual summaries of the users' responses, can be found in the Appendix.

Guidance from Ayagoz Kulekeeva

During the development of the Housing Management System, Ayagoz Kulekeeva, the Housing Management General Manager, provided significant guidance to ensure that the system would meet the real needs of the department. Throughout the semester, Ayagoz Kulekeeva continuously navigated us by offering valuable feedback on what was truly necessary for the project. She highlighted that payment processing was not required to be integrated into the system since it is handled separately through my.nu.edu.kz.

This feedback helped the team refocus the system's scope and prioritize essential functionalities, such as room assignments and maintenance request management, rather than trying to include unnecessary features. Confirmation of the three stakeholder meetings is also included in the Appendix.

Conclusion and possible future work (5%)

The NU Housing Management System (HMS) project was a semester-long journey aimed at developing a real-world, scalable solution to support the complex operations of university housing. Our goal was to create a system that could handle everything from managing residents and properties to tracking leases and handling user roles — all while balancing both backend and frontend development efforts.

From Planning to Implementation

Over two semesters, we moved from initial planning and stakeholder analysis to designing and implementing a multi-layered system that mirrors how housing works on campus. We accounted for multiple user roles — including students, faculty, admins, and support staff — and different types of housing, such as dormitories, campus apartments, and off-campus units. Our system was built to reflect the real-world challenges and needs of these interactions.

While our primary focus was on backend development, we worked closely with another team to start shaping the project into a full-stack application. On our end, we designed normalized database schemas, maintained referential integrity, and built data import tools for processing actual lease records provided by the university. We also made sure our data structures were compatible with frontend APIs, allowing for smooth integration.

Meanwhile, one of our team members contributed to the frontend side, helping to develop initial UI components for viewing leases and user profiles. This helped set the stage for a complete, interactive system experience in the future.

What We Gained

Beyond just writing code, this project pushed us to apply real software engineering practices. We learned how to plan in sprints, work as a team, communicate across functions, and iterate on our work based on changing requirements. We had to clean and structure messy data, adjust to shifting expectations, and make sure everything integrated properly — challenges that closely mirror what happens in professional development environments.

Key Contributions

- A modular and scalable PostgreSQL database schema supporting users, roles, leases, and multiple housing types.
- Backend logic that models the full lifecycle of housing assignments and lease tracking.
- Shared APIs and coordinated schema designs to support backend-frontend integration.

Future enhancements

If this system were to move toward full production use, there are several areas where further development would make a big impact:

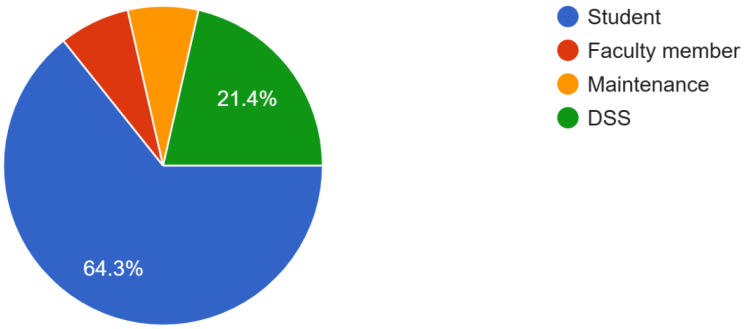
- **Complete Frontend Interface:** Build out a full UI experience for students, faculty, and admins.
- **Admin Dashboard:** Develop real-time dashboards for occupancy, revenue, and reporting.
- **Authentication and Security:** Implement full user authentication (e.g., SSO or OAuth) with secure access control.
- **Multilingual Support:** Add translations for Kazakh and Russian to make the platform more accessible.
- **Room Allocation:**

In the end, this project gave us the opportunity to build the backbone of a powerful housing management platform. We saw firsthand how software can improve processes and experiences for university housing operations. The HMS system now has a strong foundation and is ready for future development and potential real-world use.

References and Appendices (5%)

Your Role (Position)

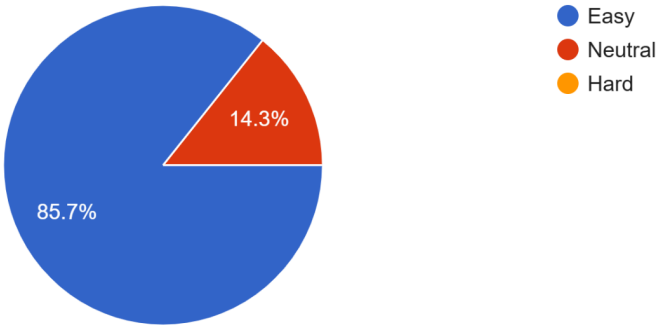
14 responses



Appendix 1. Question 1

Question: How easy was it for you to navigate through the Housing Management System to find the information you needed (e.g., room assignments, maintenance requests, etc.)?

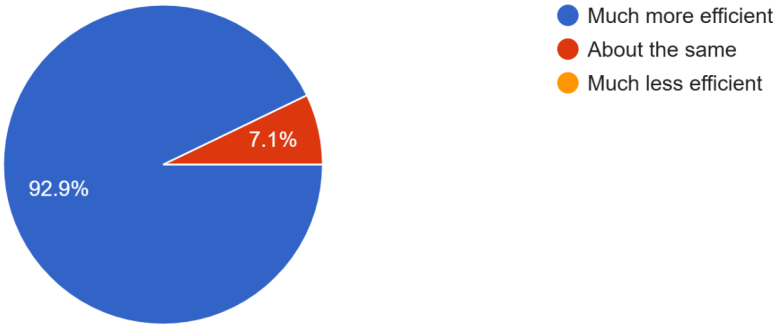
14 responses



Appendix 2. Question 2

Question: Compared to the previous system (Google Forms for students, Excel for administrators), how much more efficient do you feel the new Hous... as room assignments or maintenance requests?

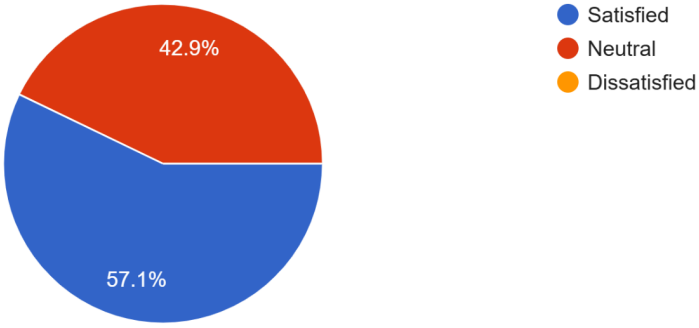
14 responses



Appendix 3. Question 3

Response Time (Performance)

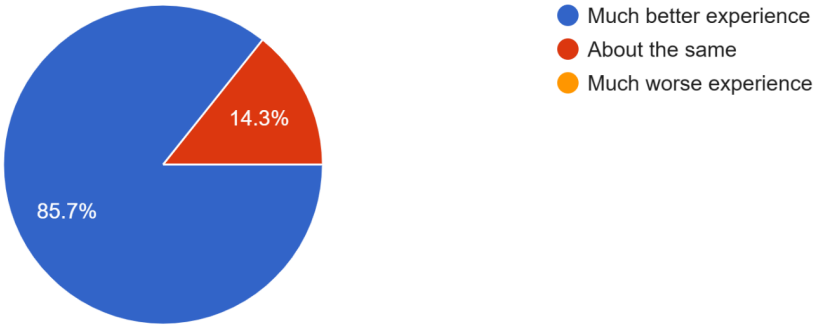
14 responses



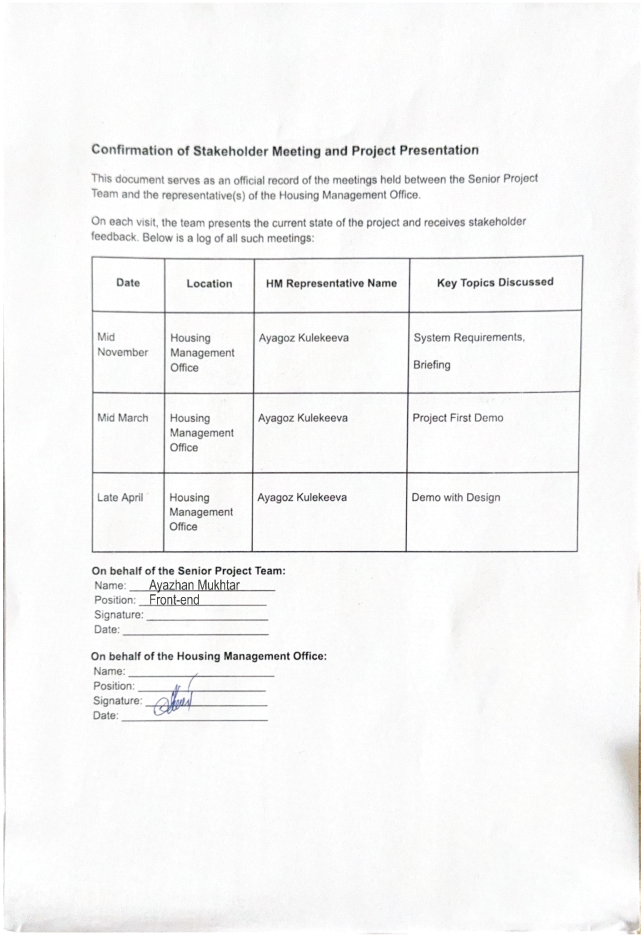
Appendix 4. Question 4

Question: How would you rate your overall experience using the new Housing Management System compared to the previous methods (Google Forms for students, Excel for administrators)?

14 responses



Appendix 5. Question 5



Confirmation of Stakeholder Meeting and Project Presentation

This document serves as an official record of the meetings held between the Senior Project Team and the representative(s) of the Housing Management Office.

On each visit, the team presents the current state of the project and receives stakeholder feedback. Below is a log of all such meetings:

Date	Location	HM Representative Name	Key Topics Discussed
Mid November	Housing Management Office	Ayagoz Kulekeeva	System Requirements, Briefing
Mid March	Housing Management Office	Ayagoz Kulekeeva	Project First Demo
Late April	Housing Management Office	Ayagoz Kulekeeva	Demo with Design

On behalf of the Senior Project Team:
Name: Ayazhan Mukhtar
Position: Front-end
Signature: _____
Date: _____

On behalf of the Housing Management Office:
Name: _____
Position: _____
Signature: _____
Date: _____

Appendix 6. Confirmation