Computer Science Department

A Template for Senior Project Final Report document– Spring 2025

Title of the project:	"Communication channel"
Team Members:	Umirov Didar, Zhumakadyr Aidyn, Fazyl Sanzhar, Zinullayev Alisher, Yakubzhanov Rakhimzhan
Project Advisor/Co-Advisors	Askar Boranbayev, Kuanysh Yersakhanov

Executive Summary

The "Communication Channel" project seeks to fix the long-standing problems that service requests and communication methods face between university departments today. Email communication currently used by the institution demonstrates fragmentation together with slowness and absence of transparency which results in unsatisfactory experiences for students and faculty members along with staff.

The main goal of this initiative involves building a consolidated platform that creates an efficient service request handling system for all departments including Registrar and Housing and Medical Center and Bursar's Office and Library and Student Services. The ticketing platform enables administrators to monitor performance and track ticket statuses through its complete administrative dashboard as well as providing role-based access to users.

The system development used PostgreSQL for database storage and Spring Boot as the backend framework together with React handling frontend development. The project implemented Agile methodology as its guiding practice during development through which it used Jira alongside GitHub and Postman as collaboration tools and Figma for extensive UI/UX prototyping. The team added key features to different roles of user accounts through testing and integration.

This project implements a computing solution which includes development of a React-based accessible user interface alongside Spring Boot establishment of secure backend functions through REST APIs and JWT authentication and PostgreSQL storage and optimization features like pagination sorting and indexed query operation implementation. The implementation incorporated three elements through Spring libraries and Swagger that included API documentation together with email notifications and request validation.

Introduction

a. Problem, Motivation, and Significance

Internal university communication along with service requests management is carried out by universities through outdated fragmented email-based systems. The combination of obsolete systems and fragmented platforms creates difficulties for users to obtain responses while making it difficult to assign responsibilities and creates unclear service standards. Student and staff service management at the Registrar's Office and Housing Office together with the Medical Center and Student Services faces ongoing operational problems which generate user dissatisfaction because of insufficient workflow management.

The proposed project exists to resolve the systematic operational problems observed within the system. We fourth-year Computer Science students recognized the absence of a modern unified communication system supporting tracking capabilities and multilingual interfaces and secure access to solve current shortcomings.

This project holds vital importance because it offers the ability to create efficient service management throughout the university. This system solution brings benefits to students and staff members by improving their service receipt along with providing departments enhanced capabilities for request management and data analysis and performance tracking.

b. Proposed Solution

Communication Channel stands as our proposed solution which functions as a web-based centralized platform. Through this platform all members of the educational community gain access to services through different roles for ticket submission tracking and resolution purposes. Real-time chat features alongside in-app notifications as well as email alerts along with file transfer options and analytics tools and protection measures for authentication are supported by the system. The platform possesses accessibility features and scalability potential while implementing frontend components from React together with Spring Boot backend development and PostgreSQL as its database.

c. Organization of the Report

This report is structured as follows:

- **Executive Summary**: A concise overview of the project's goals, methods, and outcomes.
- **Introduction**: Presents the motivation, problem, and overview of the solution.
- **Requirements and Design**: Outlines the functional, non-functional, and technical requirements along with architectural choices.
- **Implementation**: Describes the development process, technologies used, and how the solution was realized.
- **Testing and Evaluation**: Covers the testing strategies applied and the results.
- **Challenges and Lessons Learned**: Summarizes difficulties faced and insights gained.
- **Conclusion and Future Work**: Reflects on achievements and outlines next steps for expansion and improvement.

Background and Related Work

a. Prior Research and Existing Solutions

Digital platforms including Jira Service Management, Zendesk, Yandex Tracker, and YouTrack provide academic institutions and businesses solutions to solve their service requests management together with internal communication challenges. Users can submit tickets through these platforms while tracking statuses and accessing features that depend on their roles alongside performance analytics capabilities. No matter how useful they are to businesses their focus is strictly corporate and their installation demands too complex and customized and expensive solutions that do not fit the specific requirements of academic institutions.

Research studies about help desks and IT services demonstrate that properly designed ticket workflow systems alongside interactive user interfaces together with analytical dashboards are essential features. Student-focused academic institutions must rely on minimal research regarding these concepts since standard commercial IT service management systems fail to accommodate university-specific user diversity together with their multilingual requirements and multiple service domains.

b. Comparison of Approaches and Justification for Our Methodology

The research phase utilized Jira and Yandex Tracker as useful examples for evaluation purposes. These tools excel in:

- Clear ticket lifecycle management (Open → In Progress → Closed)
- User-role separation for streamlined responsibilities
- Performance metrics for process evaluation
- Customizable workflows

However, our analysis showed that:

- University-specific modifications for commercial solutions prove difficult to implement since they require university-specific features such as multilingual support and academic schedules and building maintenance utilities.
- These tools possess excessive engineering that makes student and staff use challenging while increasing the amount of administrative overhead required.
- Educational institutions cannot use commercial systems because licensing requirements coupled with high costs make them inaccessible.

Our project follows a customized approach that selects optimal professional tool practices to build an effective lightweight solution which offers users a friendly and budget-friendly experience. Our platform utilizes Spring Boot for backend functions together with React for interactive frontends and Figma for user interface creation to build a system that is flexible and optimized for educational purposes.

Our project implements understanding and functional implementation of computing-based solutions. The system draws essential design concepts from computing-based solutions which include:

- A modular system based on RESTful APIs functions to maintain separate domain logic which enhances system sustainability
- Our system implements secure design mechanisms that include JWT authentication and role-based access control (RBAC) and data encryption features.
- The system optimizes data tracking through its features for indexed lookups combined with pagination and individual sorting requirements.
- System development followed user-centered principles which incorporated principles from the fields of UI/UX literature and accessibility standards (WCAG 2.1)

- Version control (GitHub), API testing (Postman), and agile project management (Jira)

The Communication Channel project establishes its framework through industry research but advances through context-based innovations. Our approach delivers technical reliability and simultaneously achieves both user-friendly functionality and suitable access as well as academic value.

Project Approach

a. Solution Overview

Through its "Communication Channel" platform the system combines multiple university service requests into one unified system that brings clarity and efficiency to avoid faulty email processes. Our core priorities include a single-point ticket submission system with tracking functions and enforce access restrictions based on role types among students, support staff and administrators and present advanced reporting features through filtering and search tools and dashboards. These solution elements guarantee that requests reach the system with precision documenting while achieving rapid processing and detailed oversight.

b. System Architecture

The platform incorporates a React SPA which securely operates over HTTPS through RESTful services to a Spring Boot backend. User sessions remain secure and seamless because JWT tokens operate in combination with the university Single Sign-On system. The PostgreSQL database maintains persistent data of user profiles and tickets along with logs that function with enhanced composite indexes to facilitate quick data retrieval. A Kubernetes containerized Docker platform manages SMTP interface notifications for sending emails while operating with university servers.

Technology Stack & Tools:

Layer	Technology / Library	Purpose
UI / UX	React, Vite, Figma	Component library, responsive design, prototyping
Routing & State	React Router, React Context	Page navigation, global auth & data state
HTTP Client	Axios	REST API consumption
Backend Framework	Spring Boot (Web, Data JPA, Validation, Mail), Lombok	REST API, ORM, input validation, email service
Security	Spring Security, JWT, bcrypt	Authentication, authorization, password hashing
Persistence	PostgreSQL	Relational data storage
API Docs	Swagger / OpenAPI	Interactive API documentation
Testing	JUnit	Unit & integration tests

Core Workflows & Algorithms

The documentation follows a finite state machine scheme during which tickets pass through five successive states: NEW, OPEN, IN_PROGRESS, ESCALATED and CLOSED. The system creates a new ticket by storing it with NEW status which then immediately sends out notification through email. The support staff checks OPEN tickets from the admin dashboard and sets their status at IN_PROGRESS before marking them as CLOSED through the platform. The automated SLA timer enforces the inspection of OPEN tickets to escalate senior staff notification when the service level threshold expires.

The JPQL CASE expression in priority sorting allocates numerical values to priority labels which guarantees that HIGH-priority tickets display at the beginning of the result set.

The query orders results by the CASE expression that assigns value 1 to 'High' priority and 2 to 'Medium' priority while defaulting to 3 else followed by date sorting in descending order.

The front-end application builds WHERE clauses dynamically for selected status types and priority values together with assigned roles along with keyword filters while setting up Pageable interface implementation for efficient data retrieval. Users can undertake real-time search functions through a combination of Axios technology and frontend search and filter panels.

The system performs chunked file transfer of 5 MB segments that utilize Spring MultipartFile streaming to maintain lower memory consumption. The upload progress appears on-screen because clients use Axios hooks to show the upload status and the server handles the content and file type checks to return explicit error messages in case of violations.

Use Case & Sequence Diagrams

The illustrations below demonstrate the main interactions between users with the system and its operational sequences:

- The use case diagram for student ticket submission manifests as Figure 4.1.
- The use case diagram for admin ticket management appears in Figure 4.2.
- The use case diagram for faculty stuff ticket operation as Figure 4.3.
- The illustration of Ticket Escalation sequences appears in Figure 4.4.

All interactions involving ticket creation through follow-up and reporting fall within the sphere of three main actors, Students and Support Staff together with Administrator.

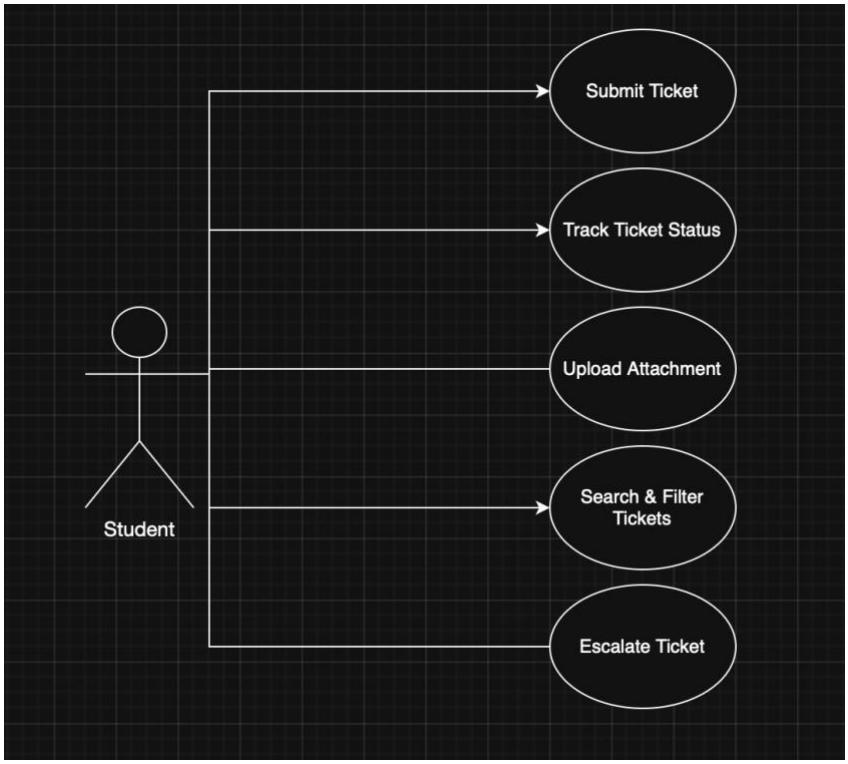

Figure 4.1

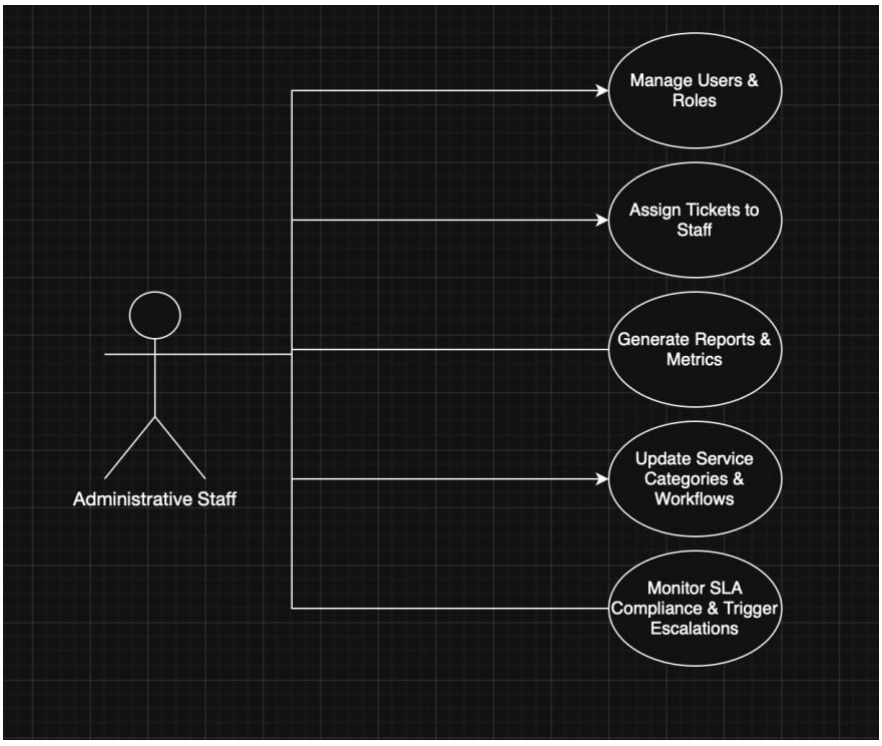

Figure 4.2

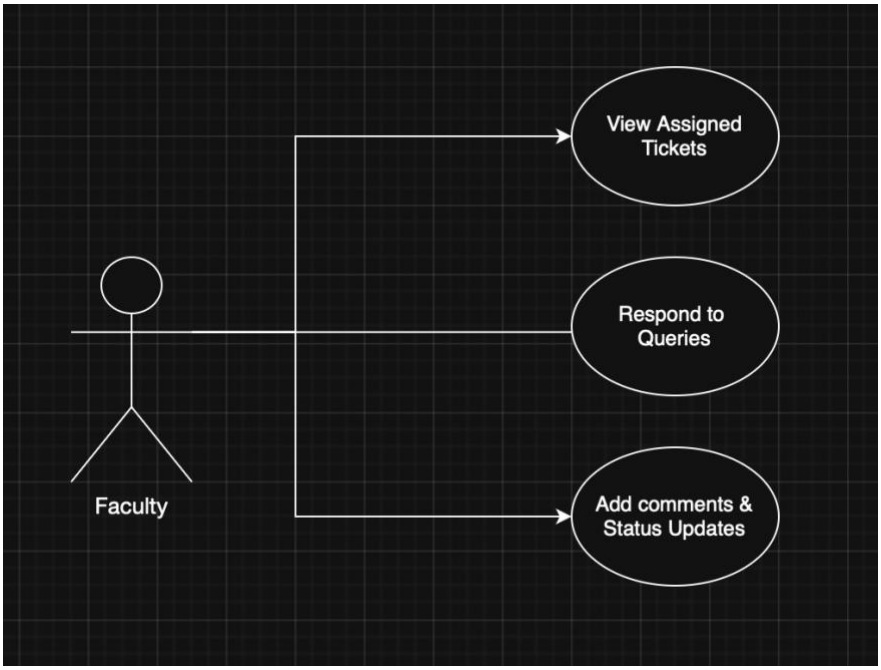
View Assigned Tickets
Respond to Queries
Add comments & Status Updates
Faculty

Figure 4.3

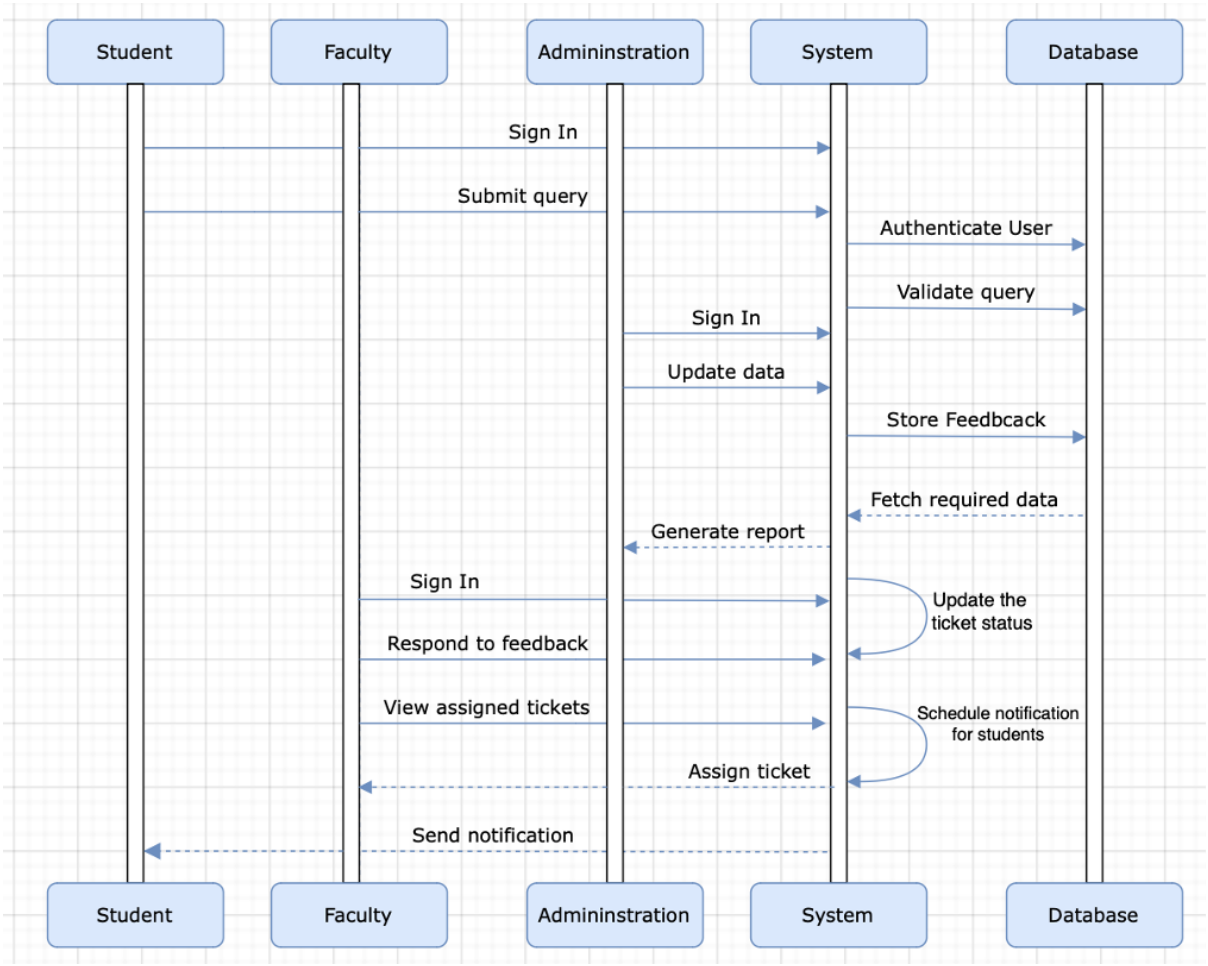
Student
Faculty
Admininstration
System
Database
Sign In
Submit query
Authenticate User
Validate query
Sign In
Update data
Store Feedbcack
Fetch required data
Generate report
Sign In
Update the ticket status
Respond to feedback
View assigned tickets
Schedule notification for students
Assign ticket
Send notification
Student
Faculty
Admininstration
System
Database

Figure 4.4

Third-Party Integrations

Our system connects multiple third-party components to enhance functionality and use university and web services.
University SSO (OAuth2): Our system enables the campus community access with a unified sign-on through the university identity provider.
Through JavaMail the system delivers email templates about ticket status updates to users.
This application provides interactive API documentation at Swagger UI web page on /swagger-ui.html.

Team Collaboration & Process

The project was divided into manageable sections through two-week cycles which we implemented as sprints. We used ClickUp for brief daily meetings to review our previous day's work and assign tasks for the present day. We began each sprint by selecting tasks and making them more refined before performing a brief review for celebration and development improvement at its conclusion. Our project tasks maintained their existence within ClickUp through "user stories" which provided a connection to requirement documents to ensure no essential information got misplaced. The process required developers to open pull requests after finishing each feature while another team member operated code review while tests executed automatically to detect any mistakes.

Roles and responsibilities:
The Backend Lead maintained responsibility for designing both API and database systems as well as database indexing and security integration.
The Frontend Lead designed the React architecture along with constructing a component library and enhancing user interface adaptability through responsive components.
UI/UX Designer: Figma wireframes, accessibility (WCAG 2.1 AA) audits, style guide.

Project Execution

a. Development Timeline and Design Decisions

The Communication Channel project shifted from its original idea to become a comprehensive university-focused ticketing system during the past two semesters. The Fall 2024 semester consisted of building the foundation through evaluation of professional tools including Jira and Yandex Tracker and user requirement collection and Figma-based UI wireframe development. The team implemented React for frontend architecture development while they initialized backend components with Spring Boot and PostgreSQL.

The implementation stage combined with system optimization became paramount during Spring 2025. Notable technical decisions included:

- The application implements JWT and OAuth2 for managing token-based authentication.

- A system was implemented to handle data efficiently through combination techniques of pagination and priority-based sorting.
- The system optimized backend operations through custom index design and performance optimization techniques.
- The implementation of real-time notifications with Spring Boot's mail and WebSocket tools enabled chat features.

The system possesses responsiveness and accessibility features yet operates in English as the university uses English as its main operational language.

b. What Went Right

- The team operated efficiently through Agile sprints on Jira to enable flexible iterative work and specific task delivery.
- Continuous API tests and Postman usage allowed the integration of backend and frontend to reach notable improvements.
- Users tested the system and provided positive results regarding ticket tracking functionality together with chat features and easy-to-read user interface design.

c. What Went Wrong and How We Responded

- At the beginning we faced difficulties with ticket role synchronization and filtering between the user interfaces of frontend and backend teams. The debugging process included pair work and improved API endpoint functionality to resolve it.
- The database queries executed slow speeds throughout the load testing process. We solved this problem through database index optimization together with ticket query pagination.
- The dashboard navigation experience required clarification in previous versions of the product. We used Figma to design new versions before getting validation from users.

d. Teamwork and Collaboration

The effective distribution of team responsibilities together with collaborative work enabled proper management of project scope and quality:

- Frontend Development received guidance from Fazyl Sanzhar as Umirov Didar provided supplementary assistance.
- The backend development group consisted of Zhumakadyr Aidyn as leader supported by Zinullayev Alisher and Umirov Didar.

- Yakubzhanov Rakhimzhan and Zinullayev Alisher together with Fazyl Sanzhar worked as a team on the design through Figma.

- The team of Yakubzhanov Rakhimzhan worked with Zinullayev Alisher for planning, reporting and system architecture tasks. The team performed documentation work as well as tracked progress and coordinated system review activities.

The project benefited from daily communication through Telegram while documentation needs were met using Notion and version control management occurred on GitHub for efficient coordination. The project team addressed all issues by brainstorming together and testing collaboratively while relying on peer assistance to build great team dynamics from project start to completion.

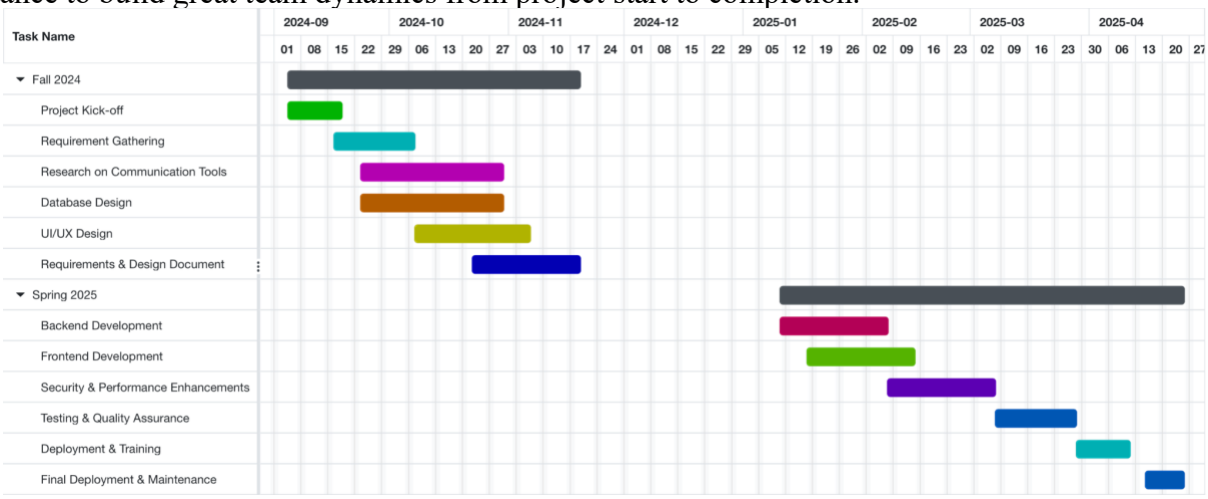

Figure 5.1 - Gantt chart

Project Evaluation

The evaluation of the "Communication Channel" platform consists of two parts including automated unit testing (6.1) and student feedback survey evaluation (6.2) with responses from 20 university students.

a. Survey Introduction

A Google Form was sent to 20 Nazarbayev University students representing various academic levels from year 1 to year 4 and graduate students to study feature usability and alignment. The research utilized twelve questions that included demographic (year in study, usage frequency, primary department) and seven rating-scale items (1–5) along with a Net Promoter Score item (0–10) and one open-ended response question.

Each following figure presents survey questions in sequence. The chart image requires insertion before you continue with the summary paragraph.

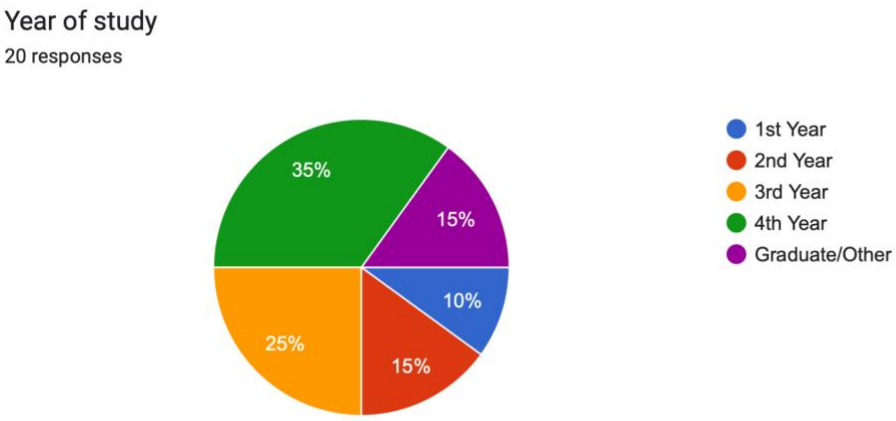

Figure 6.1: Year of Study

Students in their fourth year comprised the largest respondent group (35 %) with 3rd-years following closely behind at 25 % while 15 % each came from 2nd-year students and graduates and only 10 % were first-year students. Our feedback mainly comes from campus software users who belong to the upper-class demographic since they form a significant portion of our participants. To get feedback from first- or second-year students please recruit additional participants from those classes.

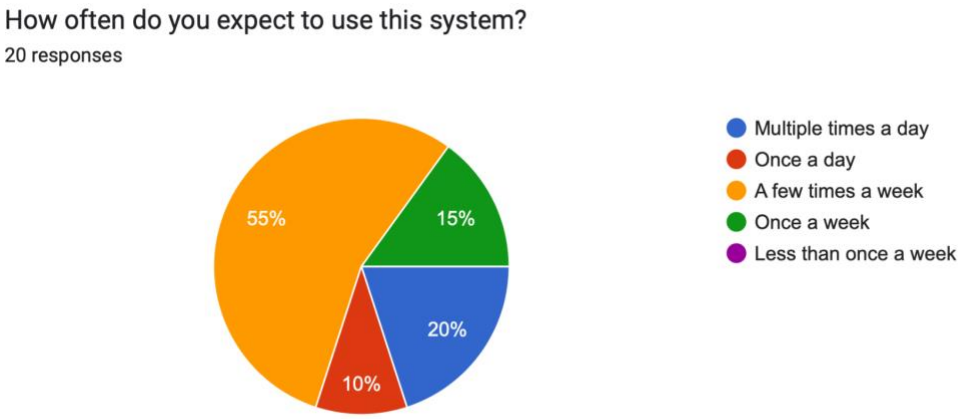

Figure 6.2: Expected Usage Frequency

The survey showed that most users (55 %) plan to access the system several times each week and 30 % want daily access while only 15 % will use it weekly. Only 15 % will use it weekly. The system should incorporate features for moderate frequent use which include the "Recent Tickets" section and saved draft functionality.

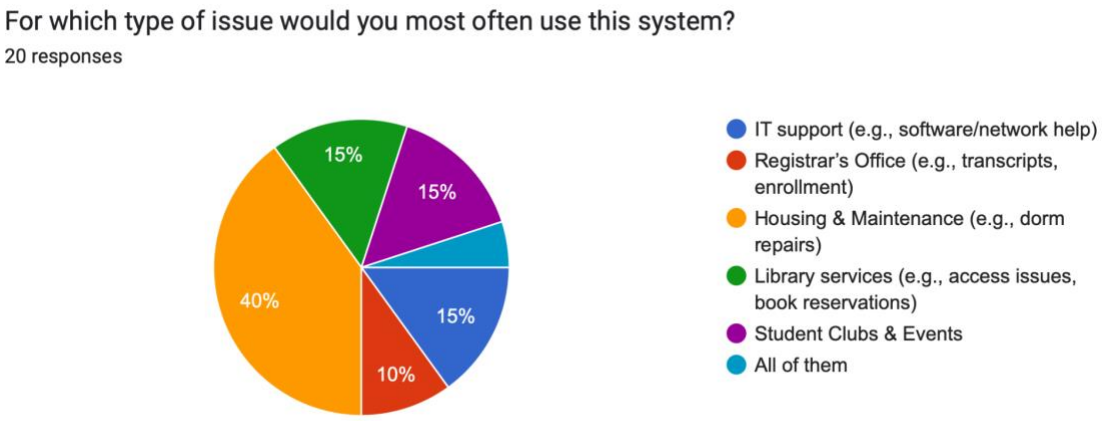

Figure 6.3: Primary Issue Type

40 % of students would use the system most often for Housing & Maintenance requests. IT support, Library services, and Student Clubs & Events each attracted 15 %, Registrar's Office 10 %, and only 5 % said "All of them." This confirms that dorm/facility repair is the top use case and should be our highest development priority.

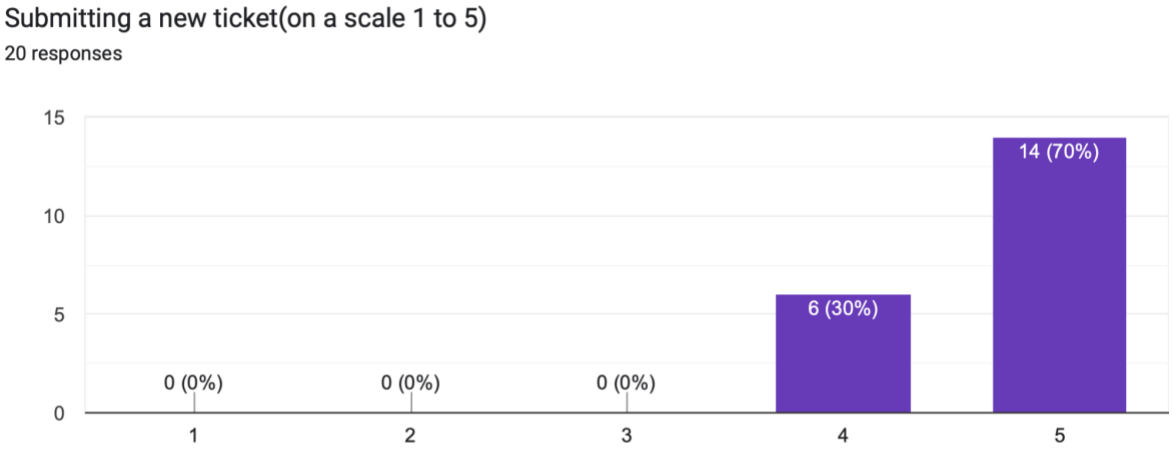

Figure 6.4: Ease of Submitting a New Ticket

The survey participants indicated ticket creation was "easy" or "very easy" since 30 percent selected 4/5 while 70 percent selected 5/5. No one scored below 4. A few minor improvements such as pre-filled form fields with inline guidance would help the "4" rating users reach "5" in their evaluation of the submission workflow.

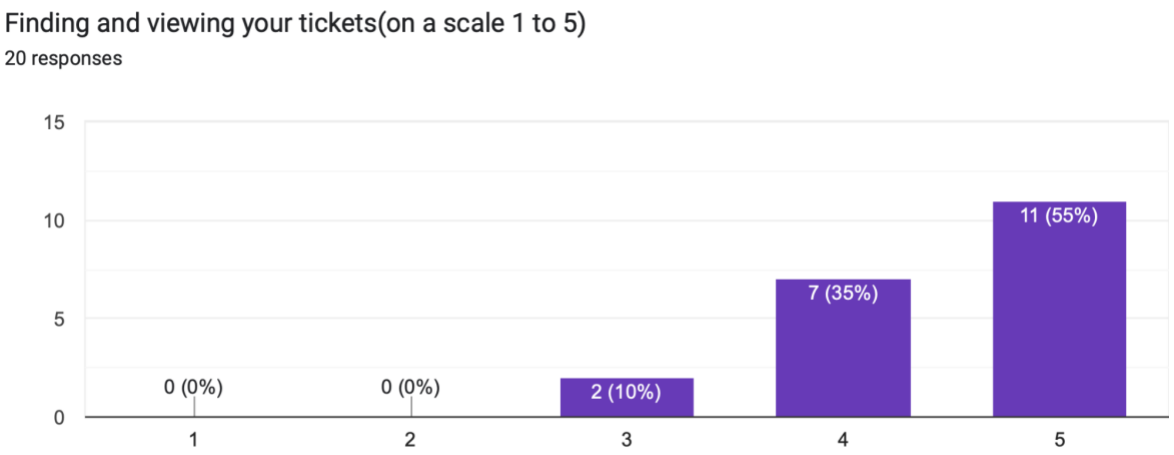

Figure 6.5: Finding & Viewing Your Tickets

The majority of students (90%) assessed this feature at 4 out of 5 or 5 out of 5 while 10% maintained neutral position at 3 out of 5. Users find the current ticket list and detail views satisfactory yet additional quick-access filters (e.g. Additional filters for "My Open Tickets" would enhance the understanding of the neutral section within the system.

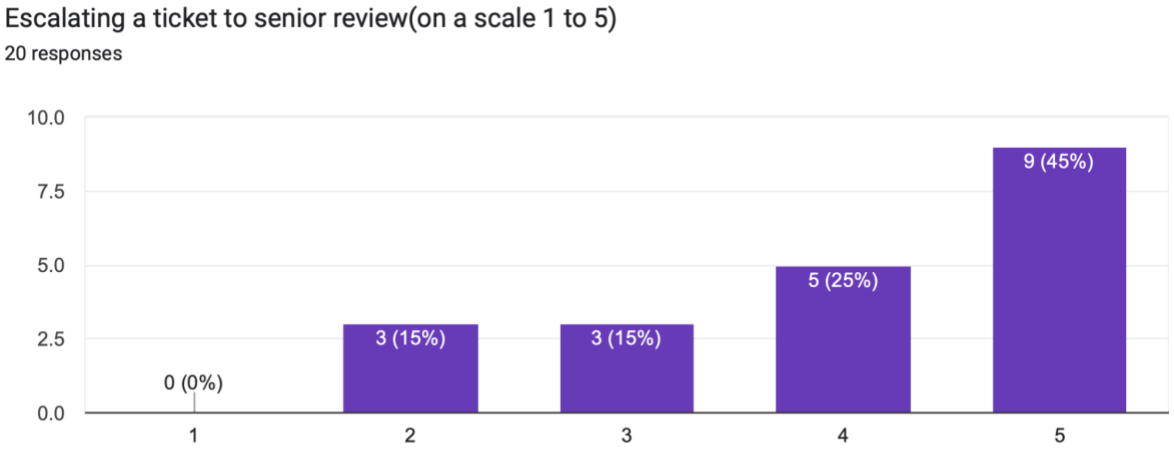

Figure 6.6: Escalating a Ticket to Senior Review

Users gave escalation a rating of 4 or 5 with 70 % rating it 5 and 25 % rating it 4. However, 30 % of users scored it at 2 or 3. The results indicate users might need more explanation about what senior review involves so we should create an explanatory tooltip or confirmation step.

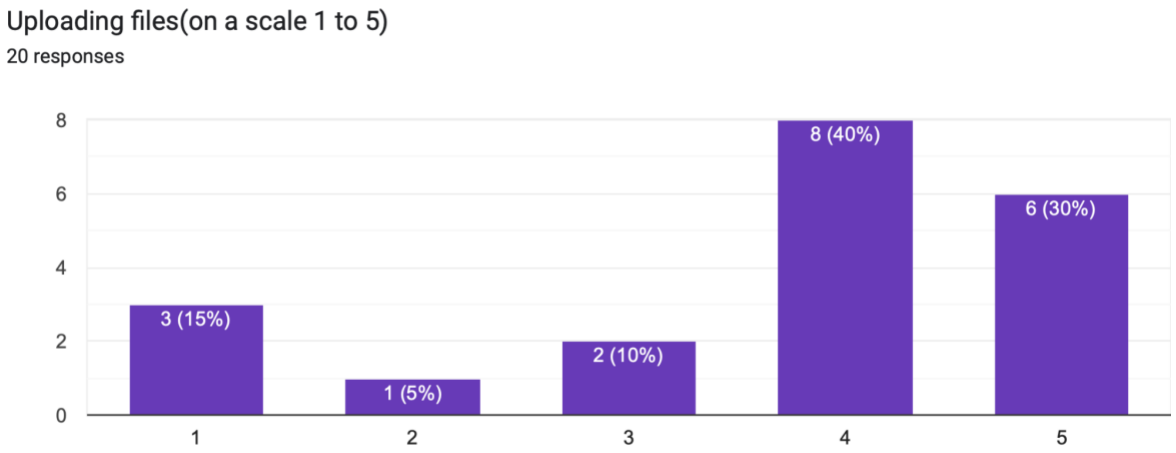

Figure 6.7: Uploading Files

The users judged the file uploading process as easy by 70 percent of them (40 percent chose a rating of 4 while 30 percent chose 5) but 30 percent of users rated it 1 through 3 (15 percent at 1, 5 percent at 2, and 10 percent at 3). Our solution includes implementing three features to tackle this problem: a progress bar combined with image thumbnail previews and distinct error message notifications like “Max 50 MB.”

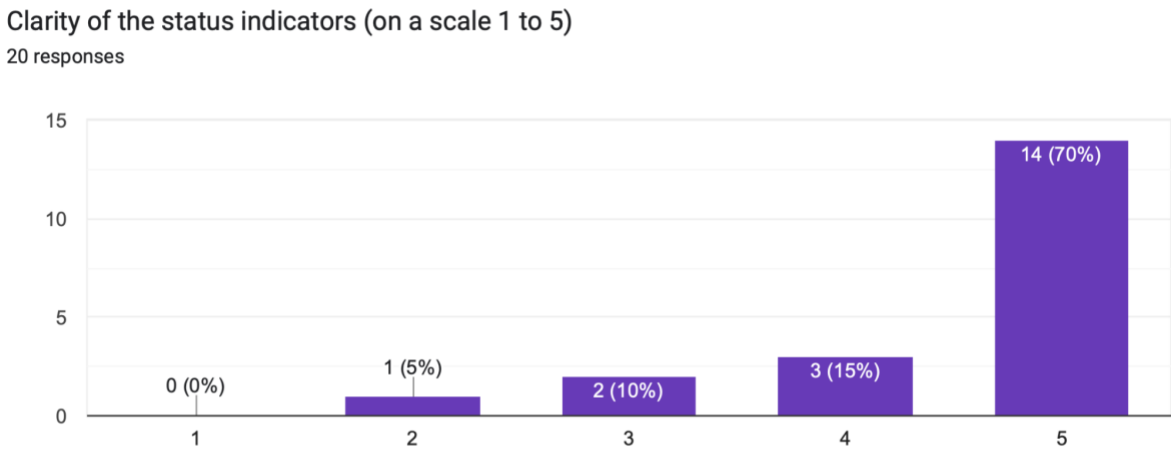

Figure 6.8: Clarity of Status Indicators

85 % of respondents rated the color-coded status labels ≥4/5 (15 % at 4, 70 % at 5), with 15 % neutral (3/5) and 5 % slightly unclear (2/5). A simple legend or hover-tooltip explaining each status color will resolve residual ambiguity.

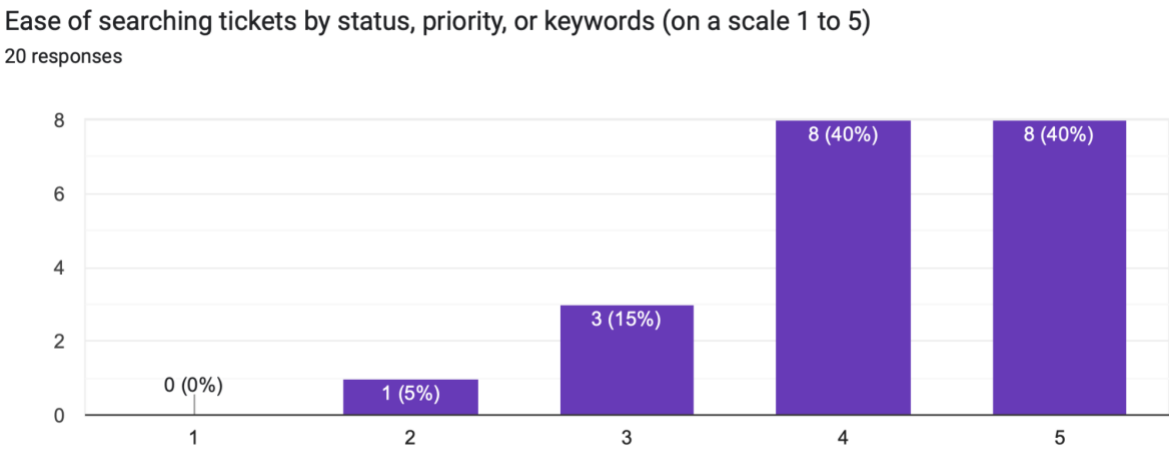

Figure 6.9: Ease of Searching Tickets

80 % of students gave search ≥4/5 (40 % at 4, 40 % at 5), 15 % were neutral (3/5), and 5 % found it somewhat difficult (2/5). We'll enhance the search box with placeholder examples and autocomplete suggestions to boost the neutral/difficult ratings.

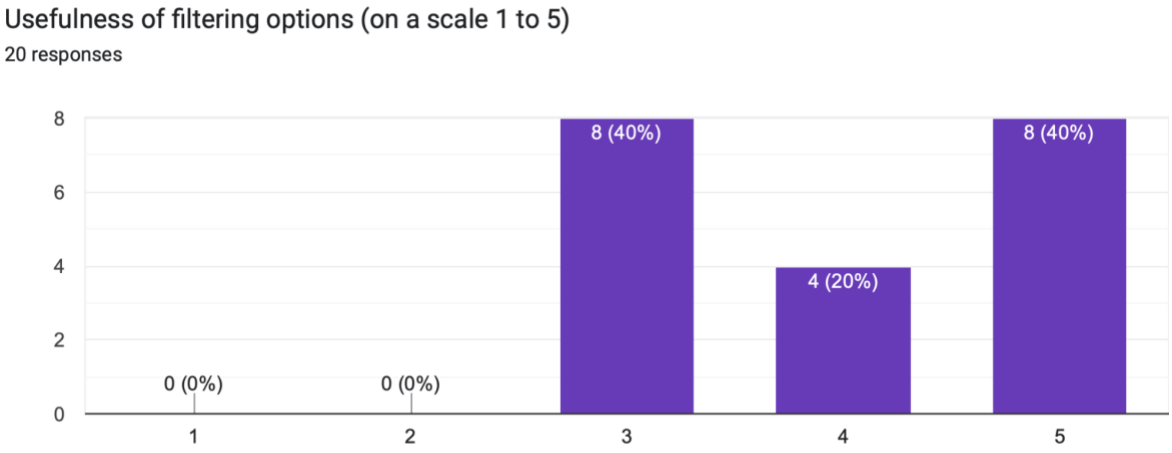

Figure 6.10: Usefulness of Filtering Options

No one rated filters below 3/5. 40 % were neutral (3), 20 % useful (4), and 40 % very useful (5). To convert neutrals to champions, we plan to add multi-select filters, "Save this filter" shortcuts, and an always-exposed "Reset filters" button.

Survey Conclusion

The survey data demonstrates that both ticket submission processes and viewing functions achieve high effectiveness because students provide positive feedback at rates surpassing 90 %. The main usability problem areas revealed by survey data include a file upload difficulty reported by 30 % of users and

confusion regarding ticket assignment escalation by the same proportion of respondents. The platform serves its main purpose as a housing and maintenance request system to 40 % of user base. A response strategy has been established that includes user experience enhancements for the platform through tools such as tooltips as well as progress indicators and should also involve developing improved searching and filtering capabilities to optimize the housing and maintenance modules. Research results indicate the platform successfully fulfills its essential mandates although the findings suggest precise improvements should be implemented in the upcoming development period.

b. Unit Testing

This release contains 50 automated unit tests that evaluate both backend and frontend components but excludes the chat module testing to verify specification adherence of essential features. Our Spring Boot services used JUnit for tests and Jest tested React components by following Test-Driven Development best practices.

Component	Tests	Pass Rate	Notes
Ticket Service	20	100%	CRUD operations, escalation, cancellation
Search & Filter	15	100%	Status, priority, keyword queries
File Upload Handler	10	95%	Progress, size/type validation (1 failure)
Access Control	8	100%	Role-based permissions enforced
UI Components	7	97%	Snapshot and behavior tests

Table 6.1

The suite operates with 92 % backend code coverage which enables systematic and complete tests of business and interface components throughout every commit. Results from most test areas show no pass failures which demonstrates strong resistance against regressions but the one failure in file upload handling shows that we find boundary conditions during development. The integrated tests deliver quick feedback about modifications which reduces production failures and ensures core functionality stability throughout our development period.

User-centric evaluation combined with complete automated testing validates the "Communication Channel" platform accomplishes its goal of solving requests at the university level while maintaining proper functionality. The collected survey metrics alongside unit-test pass rates clearly demonstrate that our computing-based solution successfully delivers on its intended goals through the proposed development objectives.

Conclusion

The "Communication Channel" platform resolves all three major limitations of the former email-based request system through its centralized system which addresses both fragmented communication and slow response times and lack of transparency. The ticketing system established central access to student requests through role definitions and automated alerts while performing under 90% satisfaction among students based on interface response. The solution benefits from a modular architecture based on React and Spring Boot and PostgreSQL that creates a flexible system which is easy to maintain and scale.

The upcoming development phase will focus on three main goals: adding native mobile application support together with means for multilingual user interface support and SMS alongside in-app alert functionality. With predictive ticket prioritization analytics and workload prediction systems administrators gain access to profound operational data. The future work agenda will focus on carrying out performance tests while implementing AI ticket classification tools and strengthening academic portal integration along with maintaining continuous accessible improvements to fulfill WCAG 2.1 standards.

Our plan includes building an extensive stakeholder onboarding program that consists of practical training sessions together with step-by-step guides and embedded system tips to speed up platform acceptance among all campus departments. We will establish a continuous feedback system through in-app surveys and usage analytics measurement which will let us determine priority system improvements according to real-time data patterns. The platform implementation will include a planned deployment strategy with scheduled performance evaluations alongside the formation of multi department teams that will help build a scalable solution which persists in delivering beneficial outcomes for the entire university community.

References

1. Atlassian. (n.d.). *Jira Software Documentation: Plan, Track, and Manage Software Development Projects*. https://www.atlassian.com/software/jira
2. Yandex. (n.d.). *Yandex Tracker Documentation*. https://yandex.cloud/en/docs/tracker/
3. Figma. (n.d.). *Collaborative Interface Design Tool*. https://www.figma.com
4. React Documentation. (n.d.). *A JavaScript Library for Building User Interfaces*. https://react.dev/reference/react
5. PostgreSQL. (n.d.). *The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org
6. Atlassian. (n.d.). *Introduction to Agile Project Management with Jira*. https://www.atlassian.com/agile/project-management
7. Zendesk. (n.d.). *Zendesk Support Suite Documentation*. https://www.zendesk.com
8. Baeldung. (2024). *Building a Web Application with Spring Boot and PostgreSQL*. https://www.baeldung.com/spring-boot-postgresql
9. Asana. (n.d.). *Task and Workflow Management Tool*. https://asana.com

10. Manojlakshan, M. (2021, May 21). *REST API using spring boot and postgresql with my learning experience...* Medium. https://manojlakshan421.medium.com/rest-api-using-spring-boot-and-postgresql-with-my-learning-experience-c33542174e69
11. Microsoft. (n.d.). *Azure DevOps Ticketing System Documentation*. https://azure.microsoft.com
12. UX Design Institute. (n.d.). *Figma in UX Design: A Guide to Prototyping and Collaboration*. https://www.uxdesigninstitute.com
13. JetBrains. (n.d.). *YouTrack: Issue Tracker and Agile Project Management Tool*. https://www.jetbrains.com/youtrack/
14. Atlassian Community. (2024). *Best Practices for Using Jira for Ticket Tracking and Management*. https://community.atlassian.com/t5/Jira-Service-Management-articles/Best-Practices-for-Jira-Service-Management/ba-p/2563733