Graduation Project Report

for

A Software for Summer Internship

Prepared by Umutcan Celik, Mert Damburacı,

Efekan Uysal, Ali Fırat Özdemir

Middle East Technical University Northern Cyprus Campus

Computer Engineering

Supervised by İdil Candan

01.06.2025

Contents

1 Introduction

1.1 Purpose

The main purpose of this project is to digitalize and improve the summer internship process for METU students and related units. Traditional methods such as paper-based documents, manual communication, and time-consuming approval steps create confusion and waste time. This system solves these problems by offering an organized and user-friendly platform that supports five different user roles.

Students can view and apply for open internship offers through the system. Since only METU students can access these offers, their chance of getting accepted increases. The system also includes a recommendation feature that analyzes the student's uploaded CV and suggests past internships based on similar profiles. Students can also submit their internship reports online. Since companies evaluate these reports through the system, there is no need for physical documents, which makes the process easier for both sides.

Companies can manage all internship steps online and avoid using old methods such as collecting documents manually. They can publish internship offers on the platform, where only METU students can apply. This helps them reach successful and qualified candidates without posting the offer on multiple websites, saving time and effort.

Coordinators do not need to assign instructors one by one anymore, thanks to the automatic assignment feature. However, they can still assign manually if needed. When coordinators publish announcements or internship forms, the system sends instant email notifications to users. Reminder emails about report deadlines are also sent automatically to both students and companies. In short, coordinators no longer need to send emails manually.

Student Affairs receives automatic email notifications when a new student is added to the system, so there is no need to refresh the page constantly. They can also download student data as Excel files using the automatic export feature instead of writing everything manually.

Instructors can grade students directly in the system instead of filling out separate Excel sheets. When they give feedback, students are notified automatically by email. If a student's score reaches a certain level, the system calculates and assigns the final letter grade automatically. Instructors are also notified when they are assigned to a student by the coordinator.

To summarize, this project reduces workload, saves time, and makes the internship process more practical and transparent for every user group. It brings all steps of internship management into a single system and improves efficiency for both students and institutional units.

1.2 Scope

This project is designed to improve and digitalize the internship process for computer engineering students at METU Northern Cyprus Campus. In the current system, many steps are done manually, such as sending emails, filling Excel files by hand, and constantly checking for updates. These cause time loss, confusion, and communication problems. Our system offers a user-friendly, online platform to solve these issues. In the future, it can be adapted for other departments or universities as well.

Goal:

To make the internship process simpler, more trackable, efficient, and transparent by improving the current methods.

Objectives:

- Ease of Use: Make the registration and starting steps easier and faster for students, companies, and coordinators.
- Automatic Notifications: Send emails automatically to students, coordinators, companies, instructors, and student affairs during different steps.
- Report Upload and Evaluation: Let students upload reports online and allow instructors to review and give feedback without physical meetings.
- Monitoring and Statistics: Help coordinators track internship status, approvals, and evaluations through the system.
- Internship Cancellation and Flexibility: Allow students to cancel their internship before it starts, helping them adjust their academic plans.
- Accessibility and Compatibility: The system works as a web application and can be used on computers, tablets, and smartphones for all users.

1.3 Identification of constraints

This project is a software-based system, so it needs to be evaluated with respect to some realistic constraints. Below are the key constraints considered during the development process and how the system addresses them:

Economic Constraints:

The system was developed with limited student resources and time. Open-source tools and free services such as GitHub and PostgreSQL were used to avoid extra costs. Users are not required to pay any fees, which increases accessibility for both students and institutions.

Ethical Constraints:

The platform is restricted to METU students only. Personal information such as email addresses and CVs is protected and accessible only by authorized users. Data privacy and user confidentiality are among the most important ethical concerns [1].

Social Constraints:

The goal is to ensure usability for all students. Since only METU students can apply to internship offers, companies receive applications from within the university, which allows internal competition. The simple and accessible design supports students with different levels of digital literacy.

Sustainability:

A modular code structure has been adopted, allowing the platform to be extended for other departments or institutions. It can also be improved with new features in the future to ensure long-term sustainability [2].

Manufacturability:

The system is built using open standards. It is well-documented, easy to maintain, and designed in a clean structure, making it suitable for future development by others.

Environmental Constraints:

As a fully digital platform, the system eliminates the need for paper and physical documentation, indirectly contributing to environmental protection.

Health & Safety:

There are no physical health or safety risks since the platform is used online. It also removes the need for in-person visits by allowing remote access to all functions.

Political Constraints:

The project is developed for educational purposes and intended for internal university use. It is not affected by political restrictions.

Conclusion:

This platform has been planned and developed with consideration of economic, ethical, social, environmental, and technical constraints. It aims to offer a secure, sustainable, and accessible solution that supports all users in the internship process.

1.4 Related Work

Internship management systems are important digital tools that help students organize their internship process and help universities follow and approve all required steps. These systems usually include form submission, report upload, supervisor assignment, and feedback collection. For coordinators, they provide control over internship approvals, deadlines, and tracking student

progress. At METU Northern Cyprus Campus, the current internship system offers these basic features through an online platform [1]. *You can see the existing system example in Figure 1.*

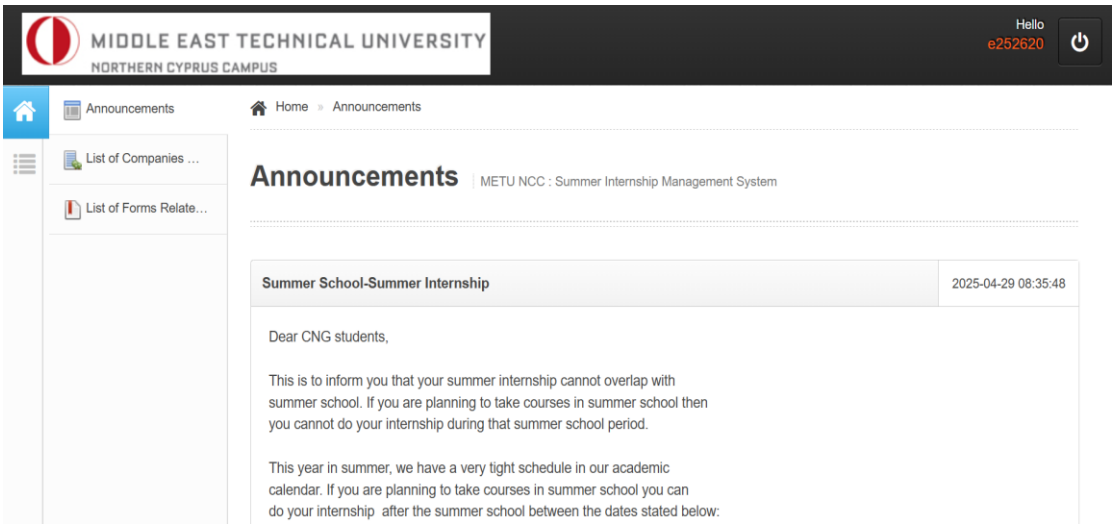


Figure 1: Example of existing system

This project is developed based on the current system, but it includes many improvements to make the experience faster, more interactive, and easier for all users. According to the literature, traditional internship systems often depend on manual email communication, physical documents, and have limited real-time interaction. Ismaili (2018) explains that these systems make it hard to track student progress and slow down communication between students and institutions [2].

Some of the improvements in our system are:

- **Students** can view internship offers, filter them by country or company, upload CVs, fill forms, apply for positions, receive notifications, and submit their internship reports online.
- **Coordinators** can publish announcements, assign instructors, set deadlines, and evaluate student reports.
- **Company users** can assign internship supervisors, evaluate students, and send feedback through the system.
- **Student Affairs** can approve internship forms and complete administrative steps like insurance checks online.

The system also includes automatic email notifications, reminders, direct feedback submission, and the ability to upload revised reports. These features reduce the need for manual tracking and make communication between all users more efficient.

Compared to traditional systems, this project offers a user-friendly interface, a centralized structure, and access levels for different roles. It is not only designed for METU students, but it is also scalable and flexible enough to be used in other universities in the future.

1.5 Product Overview

1.5.1 Product perspective

The Summer Internship System is a web-based platform developed to manage all internship-related tasks in one place. It supports students, coordinators, company representatives, and student affairs in handling the full internship process digitally.

The platform works independently but is directly integrated with Google Mail. This integration allows the system to automatically send email notifications to users about important updates such as approvals, deadlines, and status changes. It ensures faster communication and smooth workflow.

The platform is accessible on computers, tablets, and smartphones. Each user group has specific access:

- Students can see internship offers, apply, upload reports, and receive updates.
- Coordinators can manage announcements, assign instructors, approve forms, and set deadlines.
- Companies can evaluate students, assign supervisors, and confirm internships.
- Student Affairs can handle approvals, review insurance, and export reports.

This integration with Google Mail ensures all users stay informed without delays. The system structure and the mail connection are shown in Figure 2.

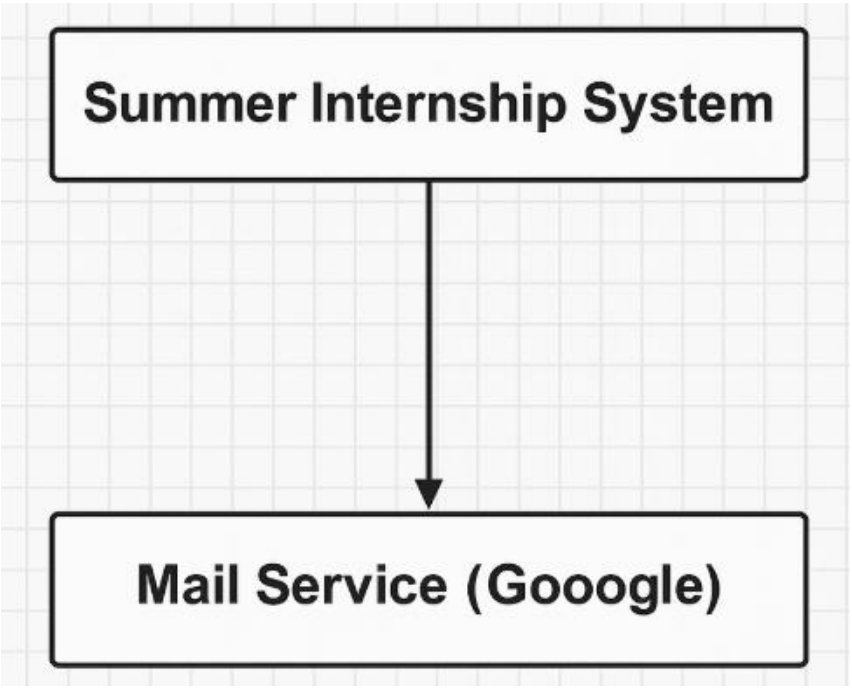


Figure 2: Product Perspective Block Diagram

1.5.1.1 System Interfaces

Google Mail Interface

The Summer Internship System is directly integrated with Google Mail. This interface allows the system to send automated emails to users for important updates. These include:

- Approval or rejection of internship applications
- New internships offer notifications
- Report submissions and evaluation results
- Reminders for upcoming deadlines

The system sends these emails automatically, ensuring users receive timely and clear information. The Google Mail interface is one-way (System → Google Mail), used only for sending data. It is a core part of the system, not an assumption, and is fully implemented.

1.5.1.2 User interfaces

The Summer Internship System provides clear and simple interfaces for each user type. Every user has access to different pages based on their role and what they are allowed to do. Below are the main pages with short explanations:

1. Welcome Page

This is the home page of the system. Students can see sections like Announcements, Forms, Manage Internships, Browse Internships and Open Offers. The layout helps users quickly go to the page they need.

→ Shown in Figure 3

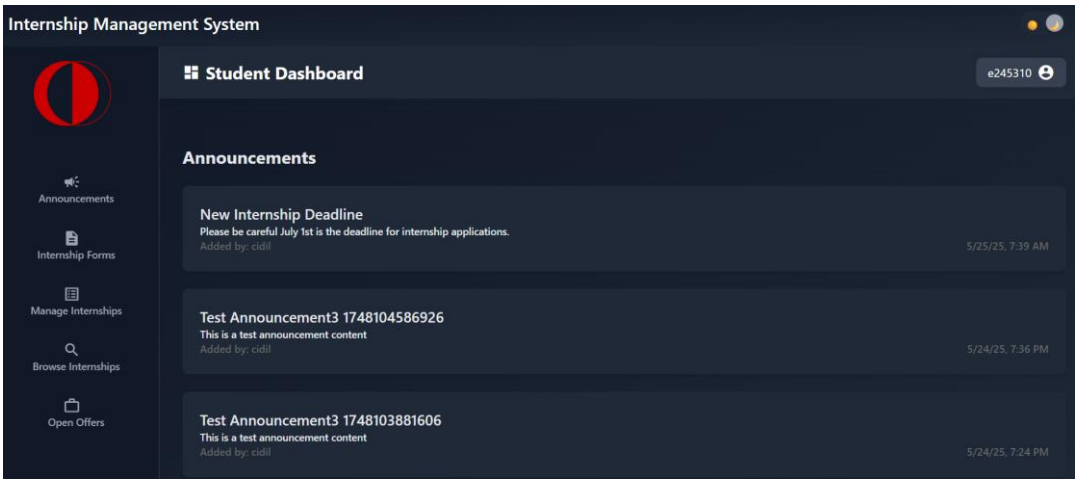


Figure 3: Welcome Page

2. Manage Internships Page

This page shows a summary of the student's past internships. It includes details such as company name, year, supervisor, report status, and grade. Students can also upload their reports.

→ Shown in Figure 4

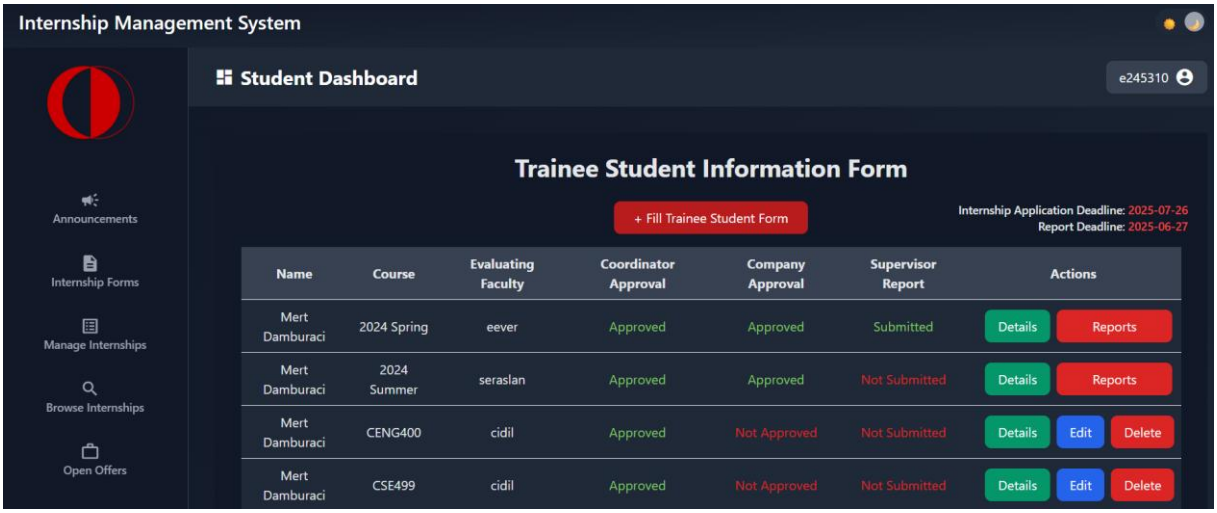


Figure 4: Manage Internships Page

3. Browse Internships Page

Students can see internships done by other students. Each entry shows company info, contact email, and rating. There are buttons to apply or view the company profile.

→ Shown in Figure 5

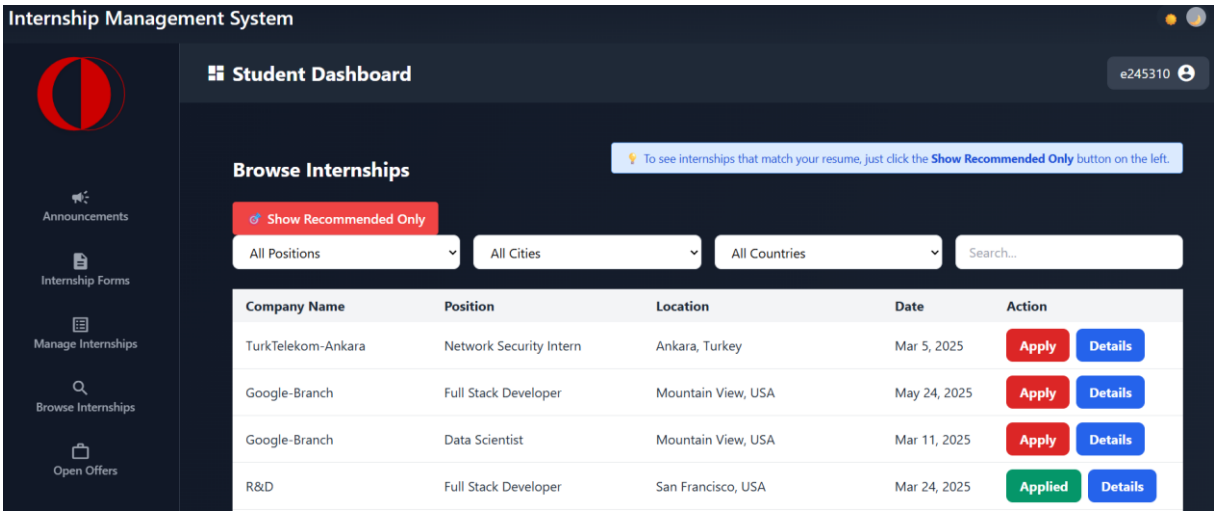


Figure 5: Browse Internships Page

4. My Resume Page

Students can upload their resume in PDF format. The page supports drag-and-drop and file browsing. It is easy to use and fast.

→ Shown in Figure 6

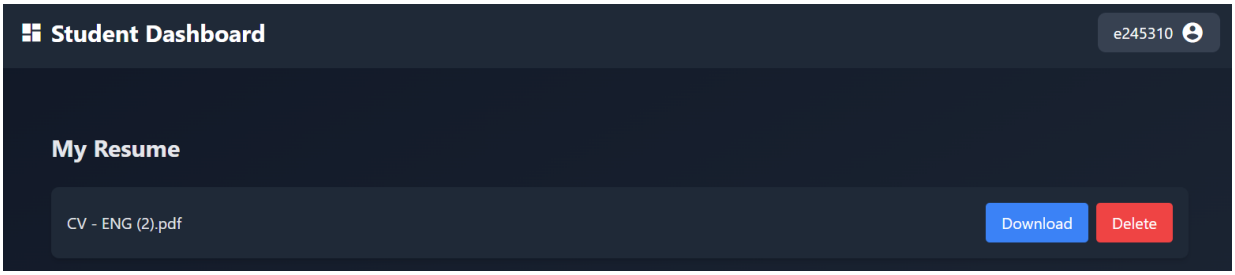


Figure 6: My Resume Page

5. Forms Page

This page shows forms that students and staff may need. Users can search, filter, download, add to favorites, or view details of each form.

→ Shown in Figure 7

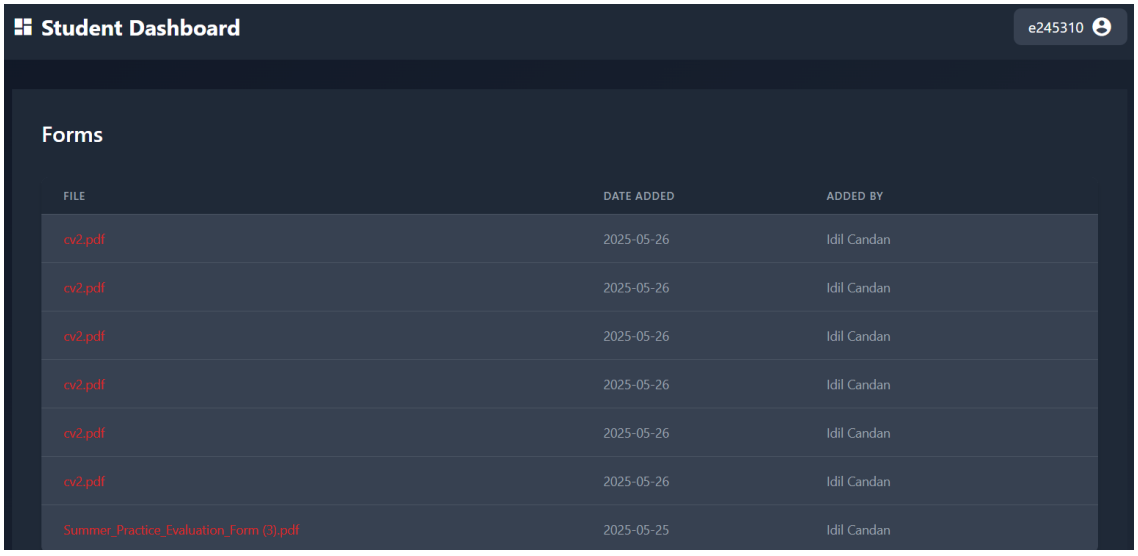


Figure 7: Forms Page

6. Evaluate Intern Student Page

Company users see students doing internships at their company. They can approve or reject them, assign supervisors, and evaluate students.

→ Shown in Figure 8

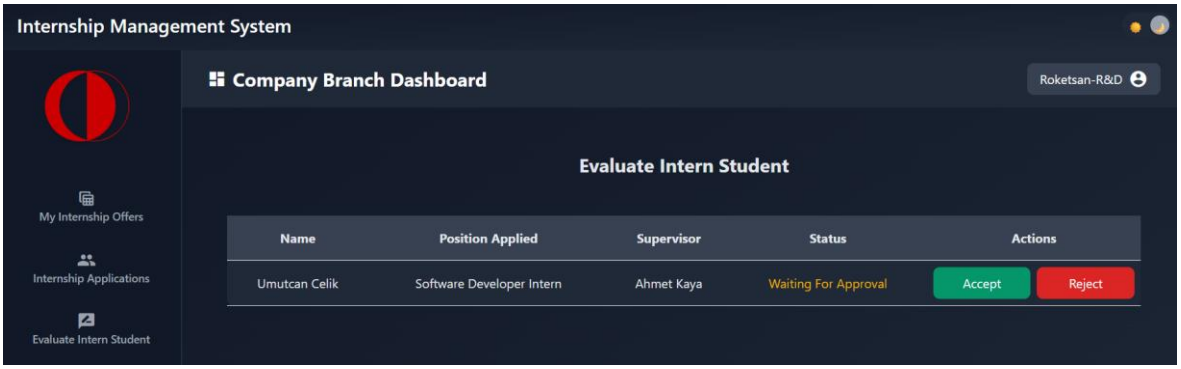


Figure 8: Evaluate Intern Student Page

7. Intern Acceptance Form

When a company clicks "Accept", a form opens to enter supervisor and internship details. The student is only accepted after this form is filled.

→ Shown in Figure 9

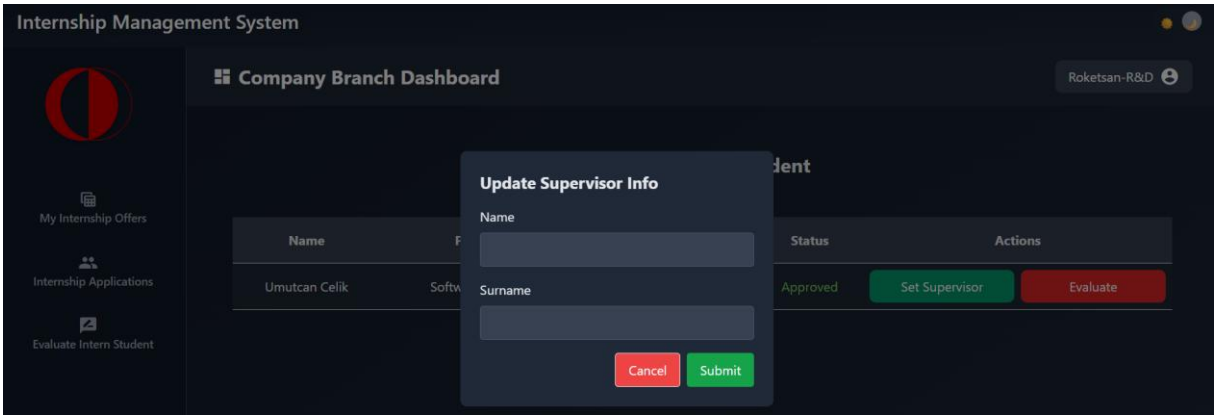


Figure 9: Intern Acceptance Form

8. Student Affairs Page

Staff from Student Affairs can see all internship information. They can approve health reports, generate Excel reports, and check approval statuses.

→ Shown in Figure 10

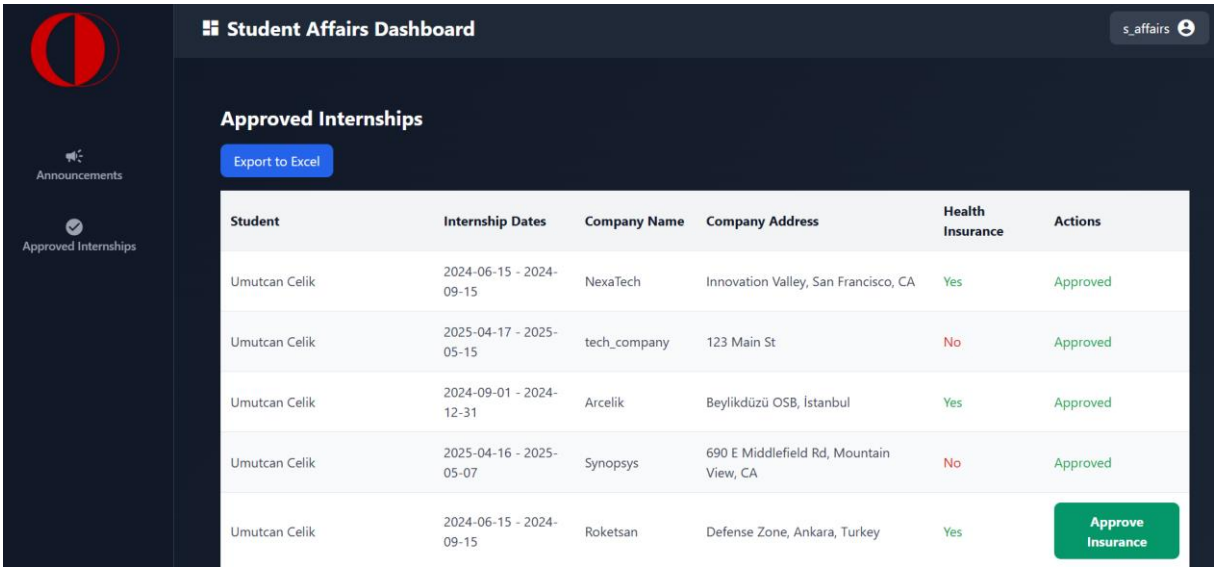


Figure 10: Student Affairs Page

The system interface is designed to be simple, accessible, and responsive. It works well on phones, tablets, and computers. All users can easily use it without advanced technical skills.

1.5.2 Product functions

The Summer Internship System is designed to support all users involved in the internship process. Below is a summary of the main functions, grouped by user roles:

Students:

- View and filter internship offers.
- Fill and submit the internship application form.
- Upload a resume (PDF format).
- Submit internship reports and check evaluation status.
- View their past internships and other students' shared experiences.
- Receive notifications through the system and email.

Coordinators:

- Publish announcements.
- Assign instructors to students (automatically or manually).
- Approve or reject internship applications.
- View uploaded reports and check their status.
- Set deadlines for reports and tasks.

Company Representatives:

- Review applications and accept students.
- Enter company supervisor information.
- Fill out evaluation forms for students.
- Track student reports submitted through the system.

Instructors:

- Grade students through the system.
- Provide written feedback.
- Access all data of assigned students.
- Let the system automatically calculate the final letter grade.

Student Affairs:

- Check health insurance status and required documents.
- Track internship submissions and approvals.
- Export data as Excel files for archiving.
- Manage approval steps within the system.

The system also sends automatic email notifications to all users. These emails inform them about application status, report evaluations, new announcements, and upcoming deadlines.

1.5.3 Identified stakeholders and design concerns

The system is used by different types of users with different technical backgrounds. For this reason, the interface was designed to be simple, easy to understand, and usable on all devices. The main user groups are explained below:

Students

- They are computer engineering students.
- Their technical skills are at a medium level.
- Most of them are familiar with forms, file uploads, and using systems like old ones.
- The interface for students is mobile-friendly and has clear buttons and guided messages.

Coordinators

- They are academic staff with high computer literacy.
- They are responsible for announcements, instructor assignments, and form approvals.
- The system gives them admin-level access with a control panel that includes advanced features.

Company Branches

- They are users from outside the university.
- Their technical knowledge may vary, and some may use the system for the first time.
- The interface is very simple, with helpful labels and a guided structure.
- Login, reviewing applications, and filling forms are made easy.

Instructors

- They are academic users who grade students and give feedback.
- They can see the information of students assigned to them.
- The interface shows only the necessary details, using clean and focused layouts.

Student Affairs

- They are administrative personnel.
- They use the system to check insurance, approve internships, and export Excel reports.
- Since they manage many tasks at once, the system allows batch actions on one screen.
- Forms are simple and contain warning messages to guide them.
- Based on these user characteristics, the system was designed to be clear, supportive, and accessible for everyone.

1.5.4 User characteristics

The users of this system share some common characteristics, regardless of their specific roles. These shared traits were carefully considered during the interface design process.

Most users are part of a university environment and have at least basic experience using online platforms such as student portals, email, or learning systems.

The system does not require advanced technical knowledge. Users with basic internet skills can easily navigate the platform.

To improve accessibility, the interface uses large readable fonts, clear section titles, and responsive layout design that works well on both desktop and mobile devices.

Visual accessibility was considered by using strong color contrast and consistent button styles.

Tooltips and short information boxes are added to help users understand specific actions without needing extra training.

In case of uncertainty, automatic email notifications and clear on-screen messages help guide the user through the process.

Additional help materials such as guides and FAQs are planned to support first-time users, especially for external users like companies.

This approach ensures the system is accessible, user-friendly, and inclusive for users with various experience levels and usage environments.

1.5.5 Limitations

The system has some limitations that may affect how it works in certain situations. These limitations are explained below:

User Input Errors

If users enter incorrect or missing information (for example, in trainee forms), it may delay the approval process or cause confusion for coordinators.

Internet and Device Issues

The system depends on users having stable internet access. Slow internet or old devices may reduce performance or cause delays.

Limited Resources

Since this is a student project, it has time and budget limits. Some extra features could not be added due to these constraints.

Privacy and Security Rules

The system must follow data protection laws. Any mistake in handling personal data may lead to legal problems and damage user trust.

External Email Services

The system uses Google Mail for sending emails. If this service is down or unavailable, email notifications will not be delivered, and users may miss important updates.

1.5.6 Assumptions and dependencies

This section explains the key assumptions and dependencies that may affect how the system works. These are not guaranteed conditions but are expected to be true during the system's operation.

- Internet Access

It is assumed that all users will have access to a stable internet connection while using the system. A slow or unstable connection may cause delays in loading pages or sending data.

- Email Service Availability

The system depends on external email services (such as Gmail) to send automatic notifications. If these services are unavailable, email delivery may be delayed or interrupted.

- Browser Compatibility

The platform is designed to work on modern web browsers like Google Chrome, Firefox, or Microsoft Edge. Users are expected to use up-to-date versions for the best experience.

- Device Access

It is assumed that users will access the system through internet-connected devices such as desktop computers, laptops, tablets, or smartphones. The system is responsive and works across different screen sizes.

- External Service Reliability

The system relies on reliable performance from external services (e.g., host provider and mail service). Any major downtime in those services may affect user access and system notifications.

- User Awareness

It is assumed that users will follow the system's guidance, such as warning messages, form instructions, and email notifications, to complete their tasks correctly.

2 Specific requirements

2.1 External interfaces

System Interface	Functionality	Input	Output
Mail Service	Send Notification Email	Mailing address to be notified Email content	Confirmation Result Code: 0: success 1: unknown problem

Explanation: This table defines our Mail Service external interface which is to notify all the actors in the system about progresses. All the 19 of the mail functionalities are tested and resulted successfully.

2.2 Functions

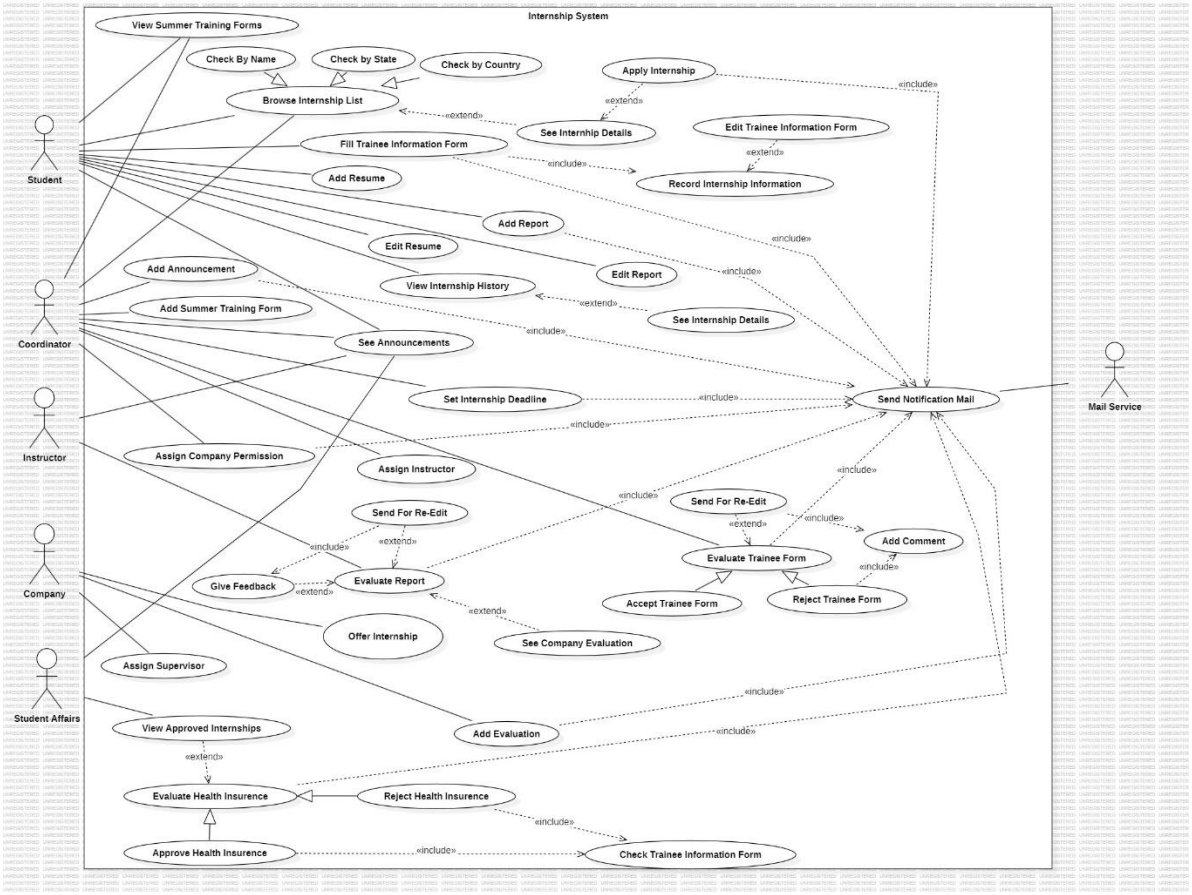


Figure 11: Use-case Diagram

Detailed Explanation for Readers:

Our system has 5 actors and all of them have some unique functionalities and use cases. Below some major use cases are explained.

1) Student:

- View Summer Training Forms: Students can view trainee forms so that they can easily manage their internship process.
- Browse Internship: Students can browse other students' internships. They have 3 filtering options, and they can apply for an internship.
- View Internship History: Students can view their past internships and the details of it.
- Add Resume: Students can add their resume to the system.

2) Coordinator:

- Add Summer Training Forms: Coordinator can upload trainee forms to the system for students to view and fill.
- Add Announcement: Coordinator can add an announcement and edit an existing announcement but edit use case is not shown in diagram as it is not so important.
- Set Internship Deadline: Coordinator can set deadlines for internships. Which means in that deadline students need to fill in their trainee forms.
- Assign Instructor: Coordinators can assign instructors to evaluate student's internship records.
- Evaluate Trainee Forms: Coordinators can evaluate student's trainee form to approve their internship information.

3) Instructor:

- Evaluate Report: Instructors can evaluate student's report with options such as re-edit, accept and reject.
- See Announcement: Instructors can view announcements that are published by Coordinator.

4) Company:

- Offer Internship: Companies can offer an internship position through system for students to apply.
- Add Evaluation: Companies evaluate the intern student at the end of their internship.
- Assign Supervisor: Companies can assign supervisors to the students to evaluate process.

5) Student Affairs:

View Approved Internships: Student Affairs can see approved internships so that they can evaluate Health Insurance process with accept and reject options.

2.3 Usability Requirements

Faceted Search & Filtering

- Requirement: Users shall be able to narrow internship listings by location, field, or company via an intuitive, faced search interface.
- Justification: Reduces time and effort spent scrolling long lists, helping users find relevant opportunities faster.

Real-Time Application Status

- Requirement: Each application entry shall display its status (Submitted, Under Review, Accepted, etc.) without requiring page reloads.
- Justification: Keeps students informed briefly, eliminating uncertainty and support inquiries.

Contextual Guidance

- Requirement: Key screens (e.g. application form, report upload) shall include inline help (tooltips or brief instructions).
- Justification: Supports first-time users by clarifying unfamiliar steps, thereby reducing form errors and frustration.

Immediate Visual Feedback

- Requirement: All user actions (apply, upload, send message) shall trigger unobtrusive feedback (toasts, spinners, success/error badges).
- Justification: Confirms that the system received the user's input, preventing duplicate actions and confusion.

Consistent Layout & Typography

- Requirement: The portal shall maintain uniform navigation, headings, and font styles across dashboard, search results, and messaging screens.
- Justification: Helps users build a mental model of the interface quickly, reducing cognitive load and navigation errors.

2.4 Performance requirements

The performance of the internship management system is crucial for ensuring a smooth and efficient experience for all users, including students, coordinators, instructors, company supervisors, and student affairs personnel. The following are the key performance requirements for the system:

- **Response Time:**

The system must provide a response time of under 3 seconds for all user interactions (e.g., form submissions, page loading, viewing announcements, and uploading reports) under normal load conditions (500 concurrent users).

Justification:

- The **3 s** limit is drawn from usability research (e.g., Jakob Nielsen's guidelines) showing that response times beyond 3 s break users' flow and cause frustration.
- **500 concurrent users** represent our estimated peak load during semester for start/end bursts in case of crowd universities.
- Slow responses can delay critical tasks like internship approvals and report submissions; staying within 3 s preserves a seamless experience.

Scalability

- **Requirement:** The system should scale up to handle **1,000 concurrent users** with no more than **10 %** degradation in response times.
- **Justification:**
    - Doubling the 500-user baseline gives headroom for unexpected surges (e.g., deadline rushes).
    - Ensures the platform remains stable during peak submission and evaluation periods without performance collapse.

- **Throughput:**

The system must process at least 50 transactions per minute, including form submissions, report uploads, and notifications.

Justification:

- Assuming 500 active users, even a 10 % action rate per minute yields ~50 ops/minute.
- Sustaining this throughput prevents backlogs when many users act simultaneously.

- **System Availability:**

The system must maintain an uptime of 99.9%, which translates to no more than 8 hours and 45 minutes of downtime per year.

Justification:

- A "three-nines" SLA is standard for academic services, ensuring near-continuous access across global time zones.
- Critical tasks (deadline submissions, approvals) demand the system be reliably online.

- **Data Processing Time:**

Bulk data operations, such as generating reports or processing multiple student applications, should not exceed 30 seconds.

Justification: Coordinators expect sub-minute turnaround when reviewing large datasets. Longer wait interrupt decision workflows and reduce operational efficiency

- **Notification Latency:**

Notifications (email, SMS, etc.) should be delivered within 5 minutes of the triggering event (e.g., form approval, deadline reminders, report submission).

Justification: Timely notifications are essential to ensure that all stakeholders (students, supervisors, coordinators) are kept informed about critical tasks and deadlines. Delayed notifications could result in missed deadlines or bottlenecks in the approval process.

- **Load Testing:**

The system should undergo periodic load testing to ensure that it can handle peak usage scenarios without a drop in performance.

Justification: Ensuring optimal performance under high load conditions is essential for maintaining system reliability, especially during the internship application and evaluation periods, when the number of active users can spike.

By meeting these performance requirements, the internship management system will ensure efficient operations, minimal downtime, and a seamless user experience across all stages of the internship process. These requirements are based on the expected volume of users and system transactions, and they are designed to prevent performance bottlenecks or user frustration.

2.5 Logical database requirements

You can examine the EER Diagram from Figure 12.

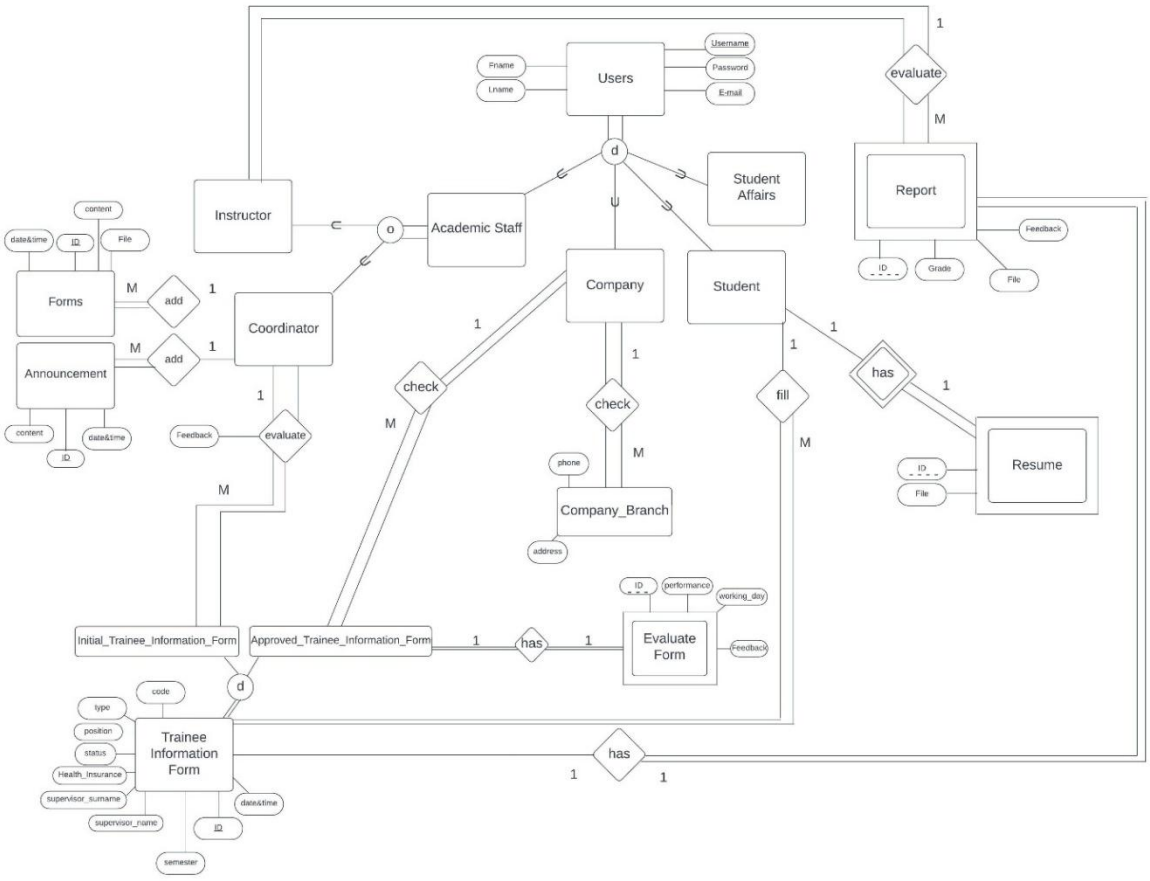


Figure 12: EER Diagram of Project

Design Requirements of EER Diagram are explained below.

Evaluate Form as a Weak Entity:

The Evaluate Form is a weak entity because it depends on the Trainee Information Form for its identification. This ensures that evaluations are tied to specific trainees, emphasizing their dependency and purpose in assessing internship performance.

Users and Roles:

The Users entity is generalized into multiple roles: Student, Coordinator, Instructor, Academic Staff, and Student Affairs. This structure allows for shared attributes like username, email, and password, while also accommodating role-specific functionalities, ensuring efficient user management.

Trainee Information Form:

This form captures detailed internship data, including supervisor information, health insurance, and semester. It serves as the foundation for monitoring and evaluating student internships.

Branch-Level Design:

The separation of Company and Company branch ensures scalability and precise tracking of internship locations, enabling companies to have multiple branches while maintaining clear records.

Approval Process:

Reports uploaded by students undergo a structured approval process, with statuses tracked as either approved or rejected. This ensures transparency, accountability, and structured feedback for students.

Forms and Announcements:

Coordinators have the flexibility to add timestamped forms and announcements, ensuring the system remains dynamic and adaptable to updates and communication needs.

Resume and Report Integration:

Resumes and reports are assigned unique IDs and are directly tied to evaluation processes, facilitating structured feedback and supporting students' academic and professional growth.

2.5.1 Relational Database Diagram Assumptions

1. We assume that each user has a unique username for easy identification within the system.
2. We believe that the coordinator can add multiple forms and announcements. These are recorded with date and time for tracking purposes.
3. Forms and reports submitted by students are assumed to be subject to review by instructors.
4. We ensure that every report has a unique ID and includes a grade as part of its evaluation.
5. Each evaluation form is designed to include details about the trainee's working days, performance, and feedback from relevant parties.
6. We assume that all announcements and forms are logged in with accurate date and time stamps for consistent record-keeping.
7. Each resume uploaded to the system must have a unique ID, ensuring proper distinction between different uploads.
8. The relationship between students and the companies they intern with is recorded in the system. A student can intern with multiple companies but can only intern at the same company once.
9. All reports and forms uploaded by students go through an approval process, where they are either approved or rejected based on evaluations, with feedback provided for clarity.
10. Company branches have unique identifiers to track their address and other details, ensuring accurate connections with students and their forms.
11. The system supports linking the initial trainee information forms to approved trainee information forms, reflecting the progression from submission to approval.

2.6 Software system attributes

Reliability

1. The system will automatically apply regular software updates to ensure ongoing compatibility with evolving technologies and maintain peak performance.

2. To mitigate risks of failure, all critical users and internship data will be securely stored using robust backup mechanisms to prevent data loss.

3. The system will efficiently manage up to 10,000 concurrent users and multiple simultaneous transactions without degradation in performance, ensuring responsiveness during peak periods.

4. In case of incorrect data entry, the system will provide clear error messages and simple correction workflows to facilitate resubmission without user frustration.

Availability

1. Planned maintenance operations will be communicated to users in advance, with service interruptions restricted to scheduled windows, typically no more than 2 hours per month.

2. For unexpected system errors, users will be presented with meaningful and descriptive error messages to guide them through recovery or next steps.

3. Automatic and redundant data recording mechanisms will ensure that users can recover their previously saved data even after crashes or interruptions.

4. The system targets 98% annual availability, meaning downtime will not exceed approximately 7.3 days per year. While this is acceptable for academic or internal-use platforms, industry best practice for critical systems is often 99.5% or higher (e.g., <1.8 days of downtime per year). Given the system's educational focus, the 2% target is reasonable and balances effort with practical needs.

Security

1. All personal and internship data will be encrypted both at rest and in transit to prevent unauthorized access.

2. Sensitive actions (e.g., profile updates, internship application submissions) will require an extra verification step, such as a one-time code sent via email.

3. The system will continuously monitor user activity, flagging anomalies such as repeated failed login attempts or suspicious access patterns for investigation.

4. Users will be periodically prompted to update their security credentials and contact information, ensuring continuous data integrity and compliance.

Maintainability

1. The system's codebase will be well-commented and documented, enabling easy understanding and modification by future developers.

2. Its modular architecture will incorporate internal fallback mechanisms, ensuring that a fault in one component doesn't impact overall system functionality.

3. Version control (e.g., Git) and peer code review processes will be strictly followed to ensure high code quality and minimize bugs during updates.

4. Clean coding practices and adherence to consistent coding standards will support long-term maintainability and facilitate ongoing development.

Portability

1. The system will be developed using platform-independent technologies to ensure compatibility across major operating systems (Windows, macOS, Linux).

2. It will support deployment on both local servers and cloud-based infrastructures (e.g., SQLPostgre), offering flexibility in hosting options.

3. The codebase will avoid hardcoded dependencies and environment-specific configurations, allowing seamless transfer between development, testing, and production environments.

4. The system will be tested on multiple browsers (e.g., Chrome, Firefox, Safari) and devices (e.g., desktop, tablet) to ensure consistent user experience and interface responsiveness.

2.7 Supporting information

This system is developed as part of a university internship management platform. It is intended for use by students, company representatives, and academic coordinators to facilitate internship processes such as application submission, evaluation, and report tracking.

Supporting standards and references:

IEEE 830-1998 – Recommended Practice for Software Requirements Specifications.

OWASP Guidelines – For secure coding practices.

PostgreSQL Documentation For understanding data storage and query functionalities used in the backend.

Spring Boot and Angular Official Docs For understanding the technologies used in system implementation.

The system follows GDPR-compliant data handling policies to ensure user privacy and secure data management. The software design also considers usability principles to make the system intuitive for users with varying technical backgrounds.

3 Software Estimation

General System Characteristics	**Degree of Influence (DI)**
Data communications	5
Distributed processing	4
Performance	4
Heavily used configuration	3
Transaction rates	5
Online data entry	2
End-user efficiency	3
Online updates	3
Complex processing	4
Reusability	4
Installation ease	3
Operational ease	3
Multiple sites	2
Facilitate change	1
Total of Degree of Influence:	46
Value Adjustment Factor (VAF):	1,11

VAF = ((TDI*0.01) + 0.65)) We used this formula to calculate VAF.

Program Characteristics	Low Complexity	Medium Complexity	High Complexity	Total
Inputs	1	2	0	10
Outputs	2	3	1	29
Inquiries	1	3	1	20
Logical Internal Files	2	2	0	14
External Interface Files	1	1	0	6

We summed up the totals for all program characteristics to obtain the **Unadjusted Total Function Points (UFP),** which is **79**.

Unadjusted Total of Function Points (ATFP): 79

Finally, we calculated the **Adjusted Total Function Points (ATFP)** using the formula: **ATFP = UFP × VAF**.

Adjusted Total of Function Points: **84.53**

KDSI Calculation:

Programming Language	Percentage	Language Unit Size	Total KDSI
Java	50%	153	26
TypeScript	20%	80	10
SQL	15%	72	5.4
Angular	15%	65	4.2

Then formula for calculating KDSI is:
KDSI = (ATFP * Language Unit Size) / 1000

1. **Java**:
   KDSI = (153×50)/1000=26
2. **TypeScript**:
   KDSI = (80×20)/1000=10
3. **SQL**:
   KDSI = (72×15)/1000=5.4
4. **Angular:**
   KDSI = (65×15)/1000=4.2

Development Type	Semi-Detached
Estimated Effort in Man-Months	**17.54**
Estimated Development Time (Months)	**6.75**
Estimated Team Size	**3**

***Basic COCOMO Parameters**:*

a = 3.0, b = 1.12, c = 0.35 our KDSI 45.6

Effort in Man-Months (MM):
$MM=3.0\times(45.6)^{1.12}$ **MM ≈ 17.54**

Development Time (in Months):
$Time=2.5\times(17.54)^{0.35}$ **Time ≈ 6.75**

Estimated Team Size:

Team Size=(17.54/6.75) **Team Size ≈ 3**

3.2 Jones' Estimation

Adjusted Total of Function Points (ATFP)	**84.53**
Kind of Software	**Systems**
Software Organization's Skills/Abilities	**Average**
Estimated Effort in Man-Months	**17.54**
Estimated Development Time in Months	**6.75**
Estimated Team Size	**3**

Our project software kind is systems because software components. We plan to integrate advanced backend systems with TypeScript, SQL, and Angular for web-based functionalities, alongside database interaction. We conducted calculations using both COCOMO and Jones' estimation methods, and the results were consistent. The estimations demonstrate a reasonable level of accuracy, validating our approach.

4 Architectural Views

The architecture of software systems will be described based on the Logical View, Process View, Development View, and Deployment View, as suggested by Kruchten (1995)[1]

4.1 Logical View

The logical view shows the key abstractions in the system as object classes.

[1] P. B. Kruchten, "The 4+1 View Model of architecture," in IEEE Software, vol. 12, no. 6, pp. 42-50, Nov. 1995, doi: 10.1109/52.469759.

4.1.1 Class Diagram

You can see the class diagram of the project from figure 13.

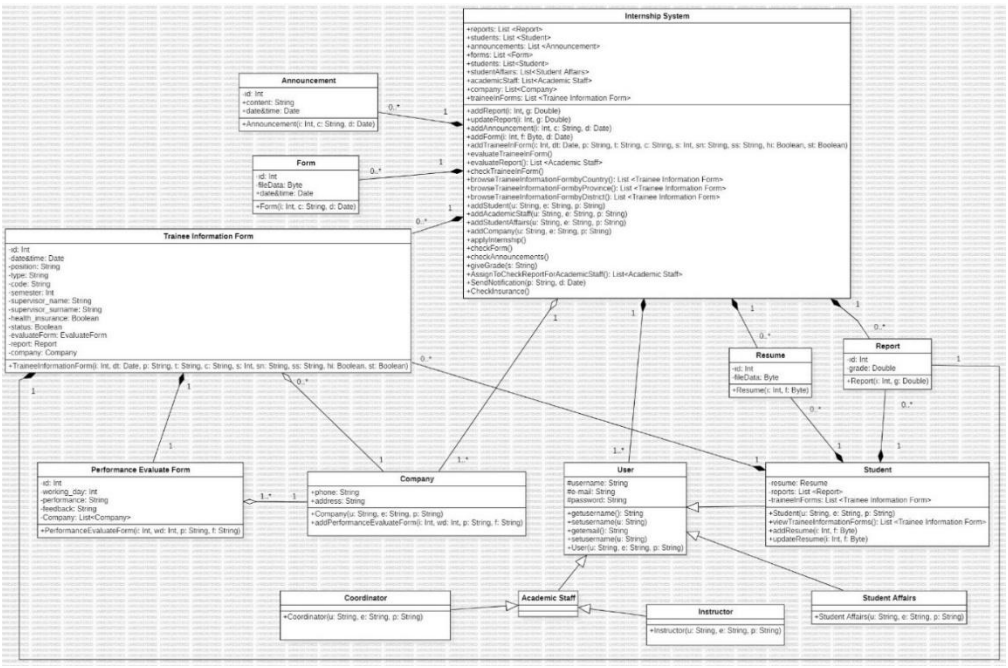


Figure 13: Class Diagram of Project

You can also examine Design Rationale below.

User Class Design:

The User class is the base for all user roles (Student, Coordinator, Instructor, Academic Staff, Student Affairs), allowing shared attributes like username, email, and password. This inheritance structure simplifies the addition of role-specific methods, such as evaluateReport() for instructors.

Trainee Information Form Dependency:

The Trainee Information Form class connects students with their internship details. It is associated with both companies and reports, ensuring that every internship is documented comprehensively.

Performance Evaluate Form as a Weak Entity:

The Performance Evaluate Form class is closely tied to the Trainee Information Form and depends on it for its identification. This highlights the subordinate nature of evaluations to internship records.

Resume and Report Integration:

The Resume and Report classes are explicitly linked to students, ensuring that all submissions are traceable to the respective user. Reports include a grading mechanism for instructors, while resumes focus on student profiles.

Coordinator Responsibilities:

The coordinator class includes methods for adding forms, announcements, and managing internship records. This role ensures system updates and administrative control.

Announcement and Form Accessibility:

Announcements and forms are designed to be dynamic, with timestamps and visibility across user roles, ensuring effective communication.

Company Association:

The Company class includes attributes like address and phone for detailed information. It is linked to both Performance Evaluate Form and Trainee Information Form, ensuring a comprehensive record of internships.

Scalability and Maintainability:

The class diagram is structured to allow future expansion, such as adding new user roles or extending functionalities like advanced feedback systems. Each component is modular and maintainable.

4.2 Process View

The process view shows how the system is composed of interacting processes.

4.2.1 Activity Diagram

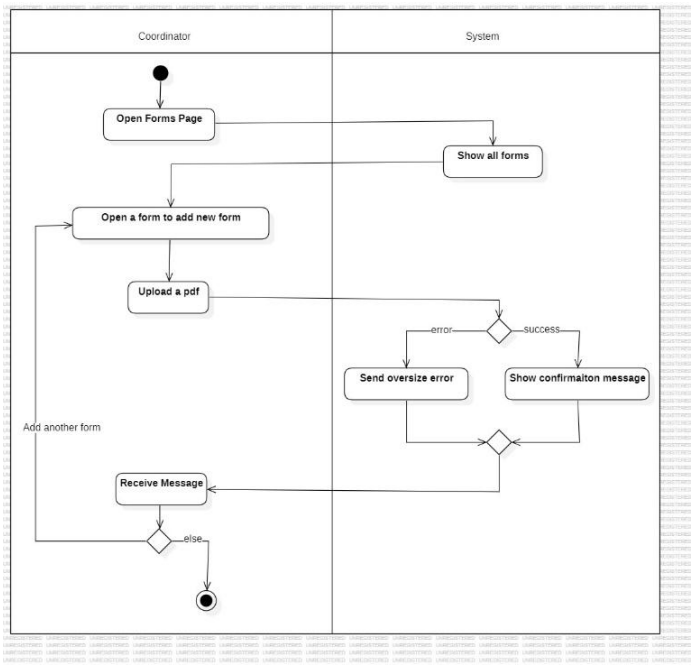


Figure 14: Add Form – Activity Diagram

Design Rationale for Add Form

In the "Add Form" process, we aimed to simplify how coordinators upload and manage forms in the system. Key considerations include:

Ease of Use: Coordinators can quickly access the Forms Page and follow a simple process to upload a new form in PDF format.

Error Handling: To prevent technical issues, the system checks the size of the uploaded PDF and notifies the coordinator if it exceeds the limit. This ensures the system remains efficient.

Feedback: After a successful upload, the system provides a confirmation message to reassure the coordinator that the form is accessible to users.

You can see the details of this process in figure 14

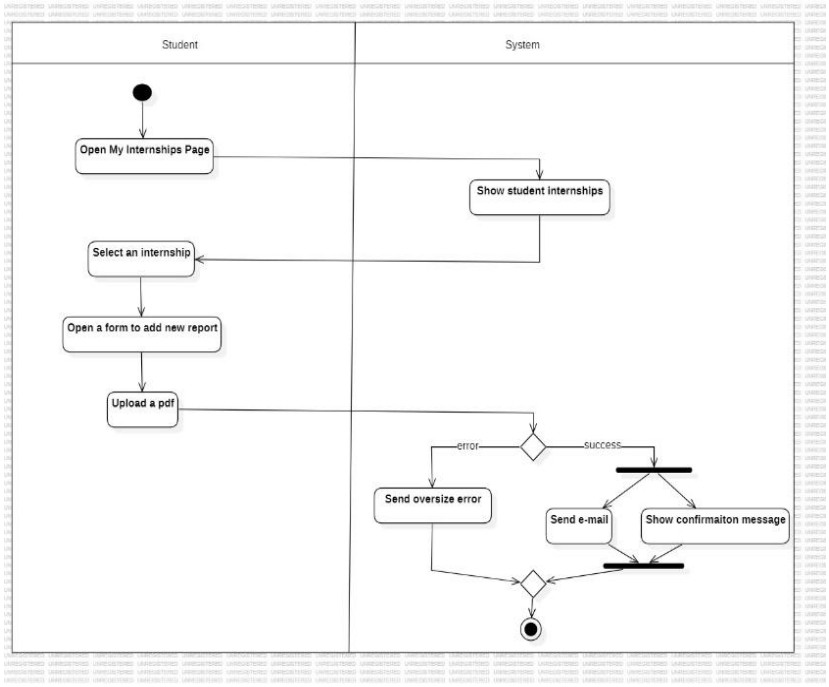


Figure 15: Add Report – Activity Diagram

Design Rationale for Add Report

For the "Add Report" process, we designed a clear and efficient workflow to enable students to upload their reports seamlessly. We considered the following:

Student-Centric Design: Students can access their internship details easily and select the relevant internship to upload their report.

Validation: The system checks the uploaded report for size issues and provides immediate feedback if an error occurs.

Notifications: Once the report is successfully uploaded, the system sends an email to the coordinator, ensuring they are informed about the new report submission.

Confirmation: A success message reassures students that their report has been added to the system.

You can see the details of this process in figure 15.

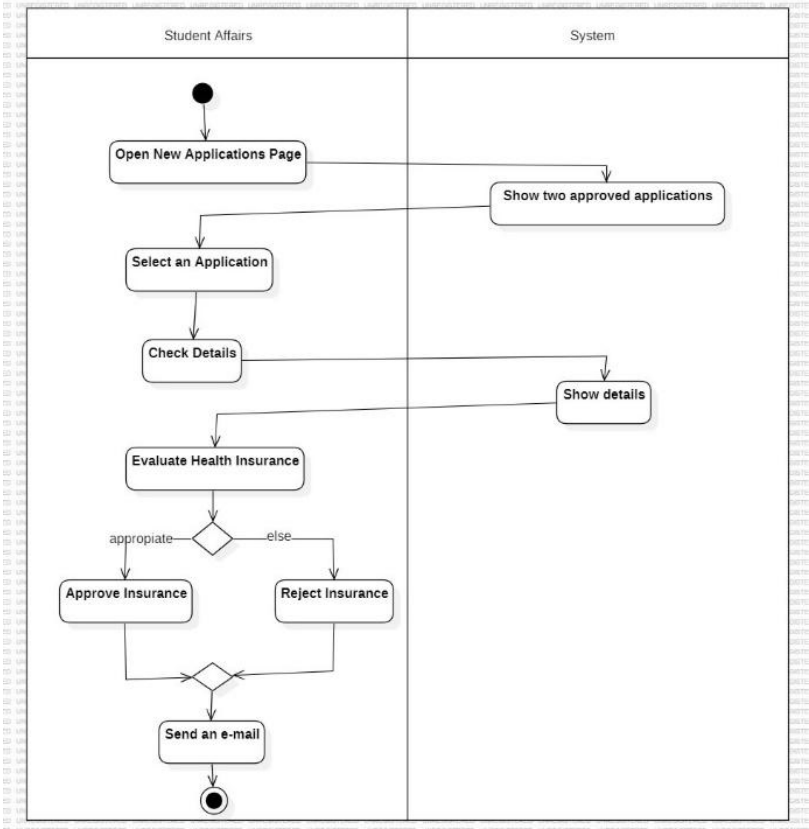


Figure 16: Evaluate Health Insurance – Activity Diagram

Design Rationale for Evaluate Health Insurance

The "Evaluate Health Insurance" process was designed to streamline how Student Affairs checks and approves insurance details. We focused on:

Transparency: Student Affairs officers can view both company and coordinator approvals before evaluating the application. This provides a clear context for decision-making.

Decision Workflow: Officers can approve or reject insurance details based on their review, ensuring accuracy and compliance.

Notifications: The system automatically informs students about the decision, maintaining effective communication.

You can see the details of this process in figure 16.

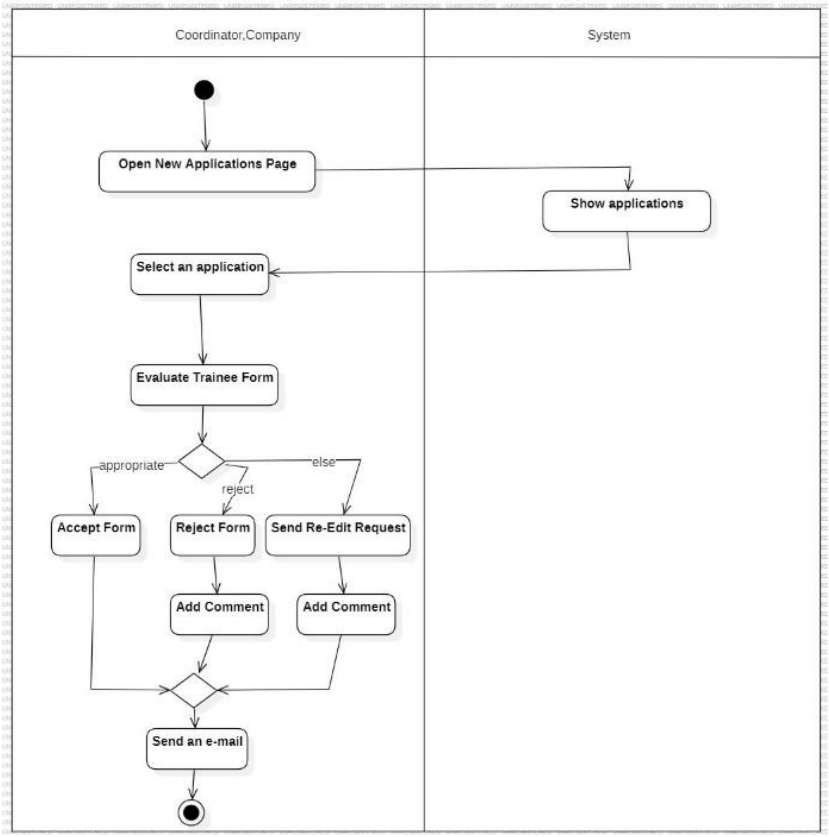


Figure 17: Evaluate Trainee Form – Activity Diagram

Design Rationale for Evaluate Trainee Form

In the "Evaluate Trainee Form" process, we aimed to facilitate how coordinators and companies review applications. Key considerations include:

Efficient Review: Coordinators and companies can access all applications in one place and check trainee forms systematically.

Error Handling: If a form is incomplete or incorrect, users can reject it or request changes, ensuring that only valid forms proceed.

Communication: Coordinators and companies can add comments during evaluation, which are sent to the student along with the result. This provides constructive feedback for improvements.

Automation: The system sends emails with results to students, ensuring they stay informed about the status of their applications.

You can see the details of this process in figure 17.

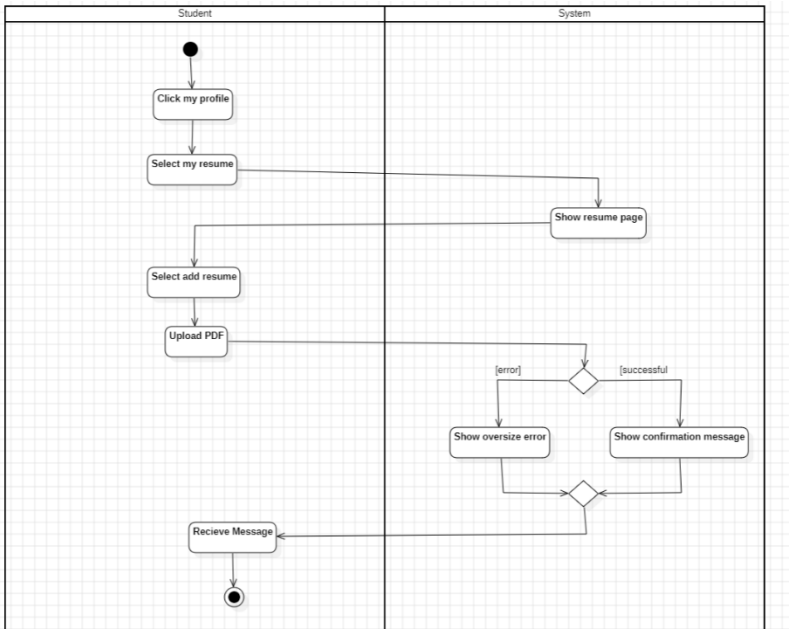


Figure 18: Add Resume – Activity Diagram

Design Rationale: Student clicks my profile button, then selects my resume page. System shows the page. Student selects add resume button and uploads resume as a PDF file. If PDF files are oversized, then the student faces errors. If it is not, it gets a confirmation message. You can see from Figure 18.

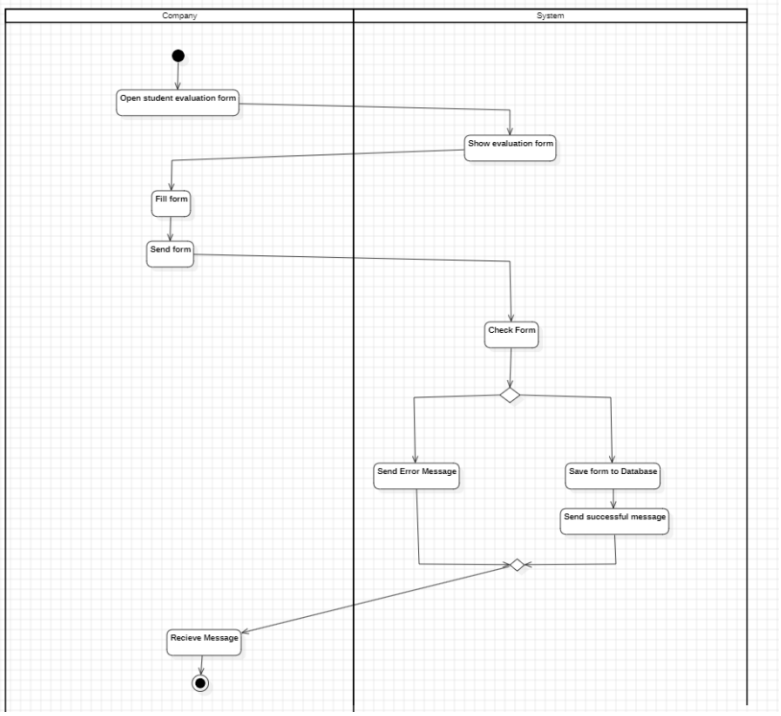


Figure 19: Add Evaluation – Activity Diagram

Design Rationale: After login the system Company opens student evaluation form. System shows it. Company evaluator fills and sends the form. System checks if it is error or not (such as dates are not same as trainee form) If it is error they face with error message. If it is not the report will be saved and the company will get a successful message. You can see from Figure 19.

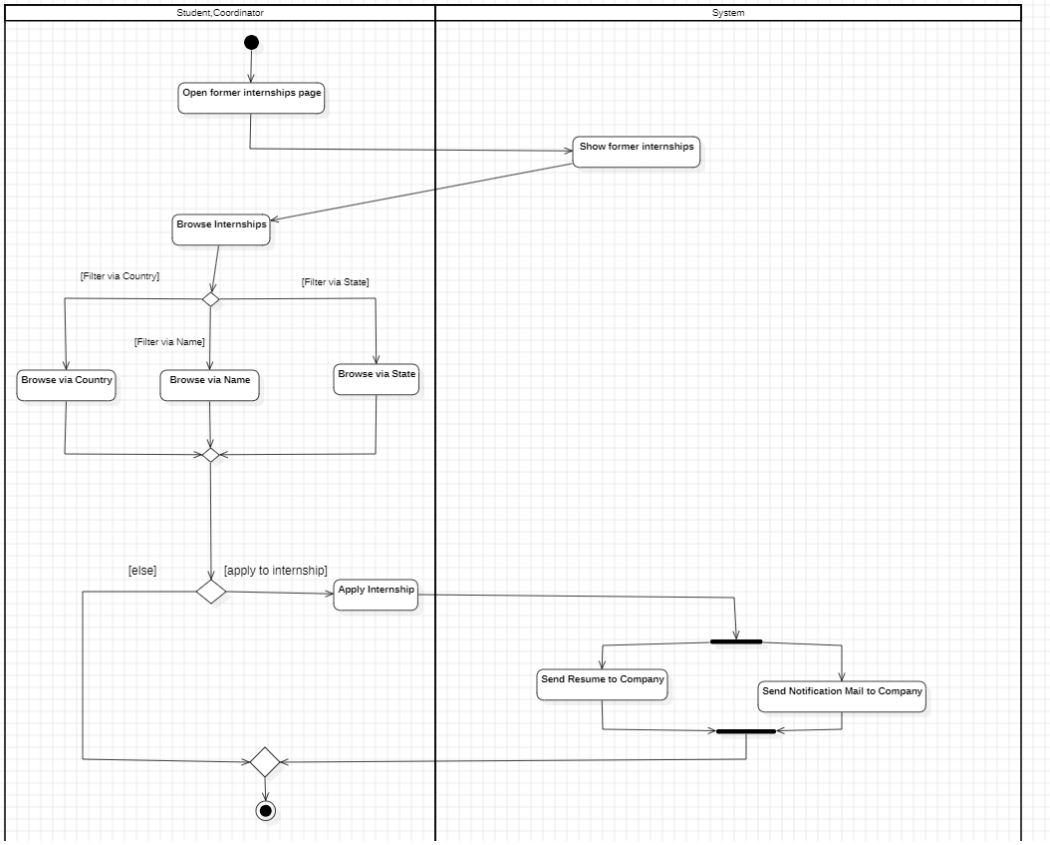


Figure 20: Browse Internships – Activity Diagram

Design Rationale: In this diagram browsing internships are shown. Students and Coordinator can both do it. First, they need to go to the former internships page, which system will provide. Then they can start to browse. After that they can use filtering such as via country, state and company name. Also, students can apply for the internships they see with apply button. If they do their already uploaded resume will be sent to company interface and Company will be notified via mail. Of course, if students do not have uploaded pdf they cannot click apply button. You can see from Figure 20.

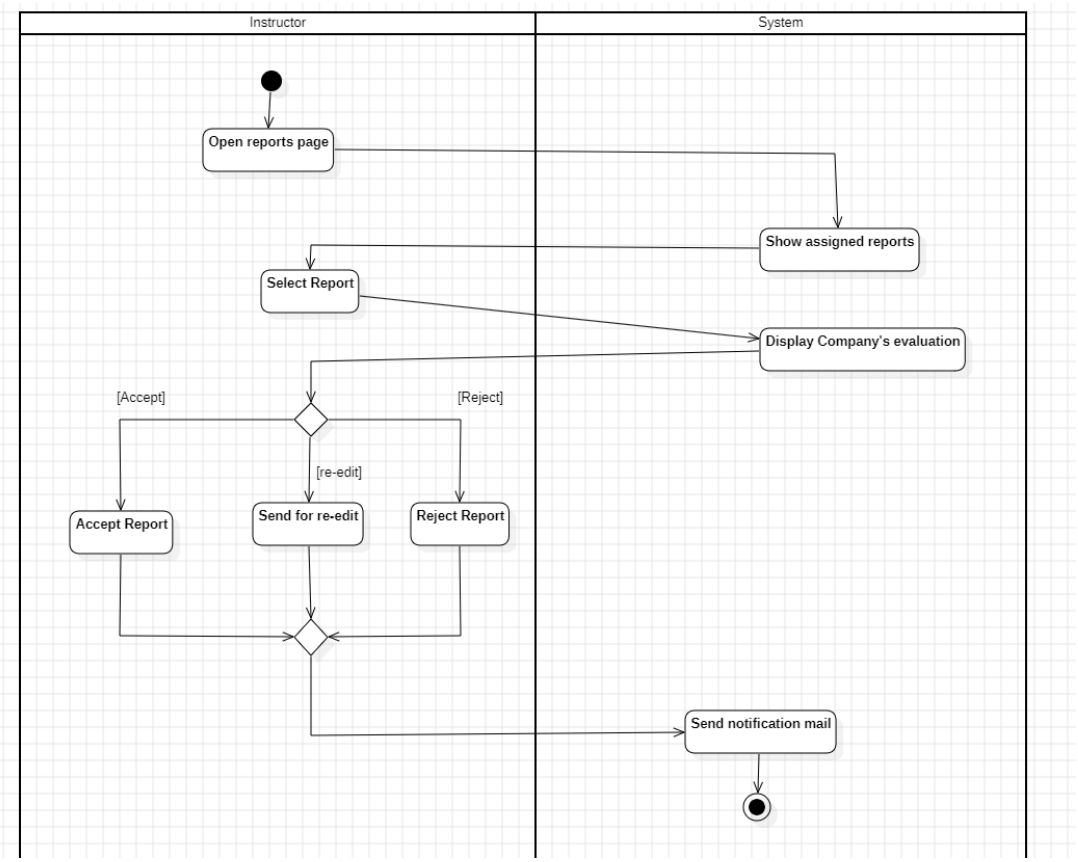


Figure 21: Evaluate Reports – Activity Diagram

Design Rationale: Evaluating reports are Instructors job. Instructors must open the reports page. When they open, they will see their assigned reports by Coordinator. Then they will select the report of any student, and the system will provide the Company's Evaluation then they start to evaluate it. They have 3 options after that. First, they can Accept Report. Second, they can send report back to students for re-editing purposes. Third they can reject the report. At the end notification mail will be sent to students after either taking feedback or report. You can see from Figure 21.

4.2.2 Sequence Diagrams

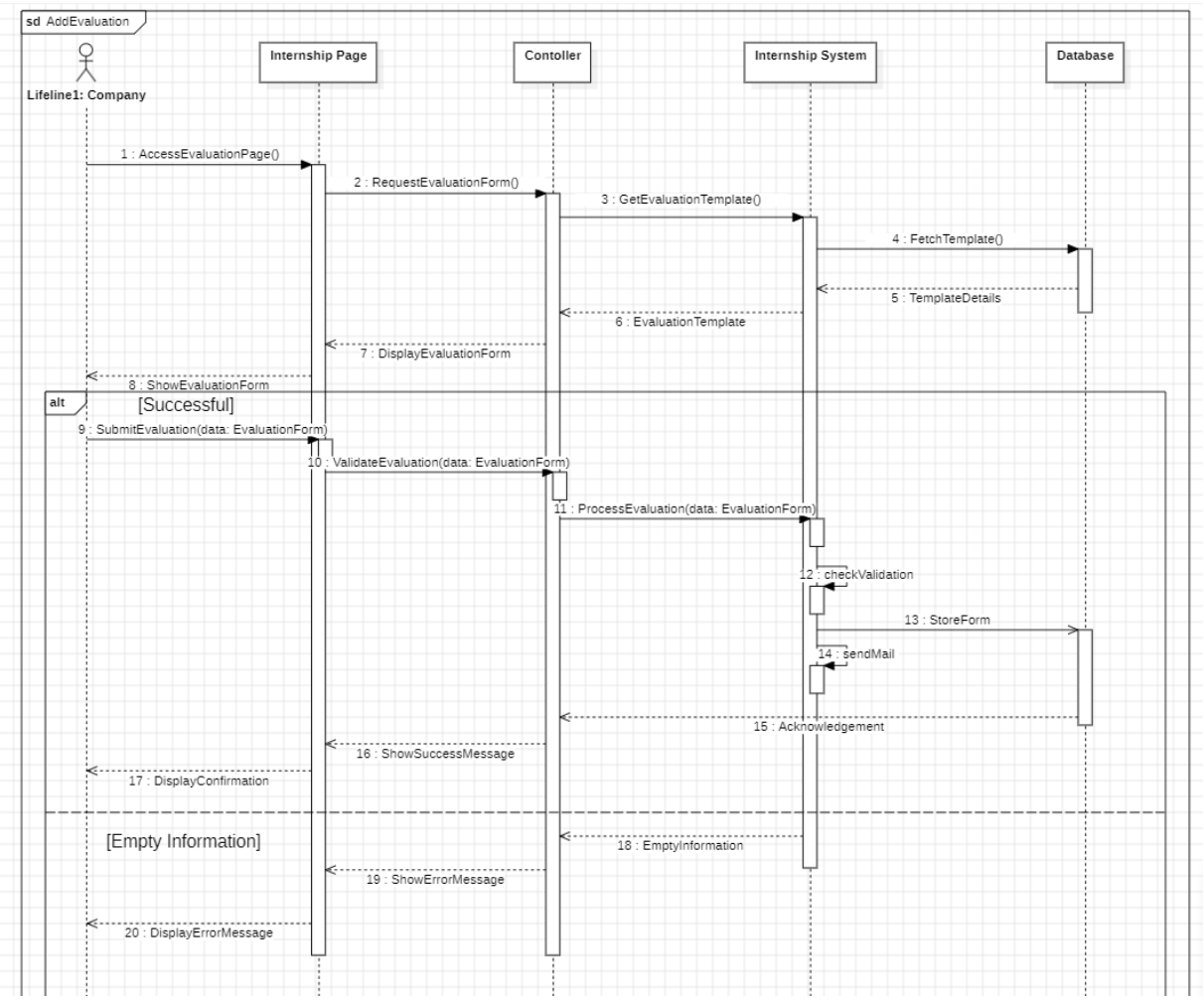


Figure 22: Sequence Diagram of AddEvaluation

Design Rationale: This is the company's evaluating an intern student process. At the end of the internship Company's must evaluate the student. First, they need to access the evaluating page. Since our project design is related to Model View Controller (MVC) pattern [6], we have made our sequence diagrams related to this pattern. After accessing the evaluation form with controller and system. Company workers must fill in the form with no empty information. If they try to submit with incorrect or missing information such as wrong working days, they will be warned with error message. If they submit correctly, the system will save the form to the database and send notification mail to students and coordinators. You can see from Figure 22.

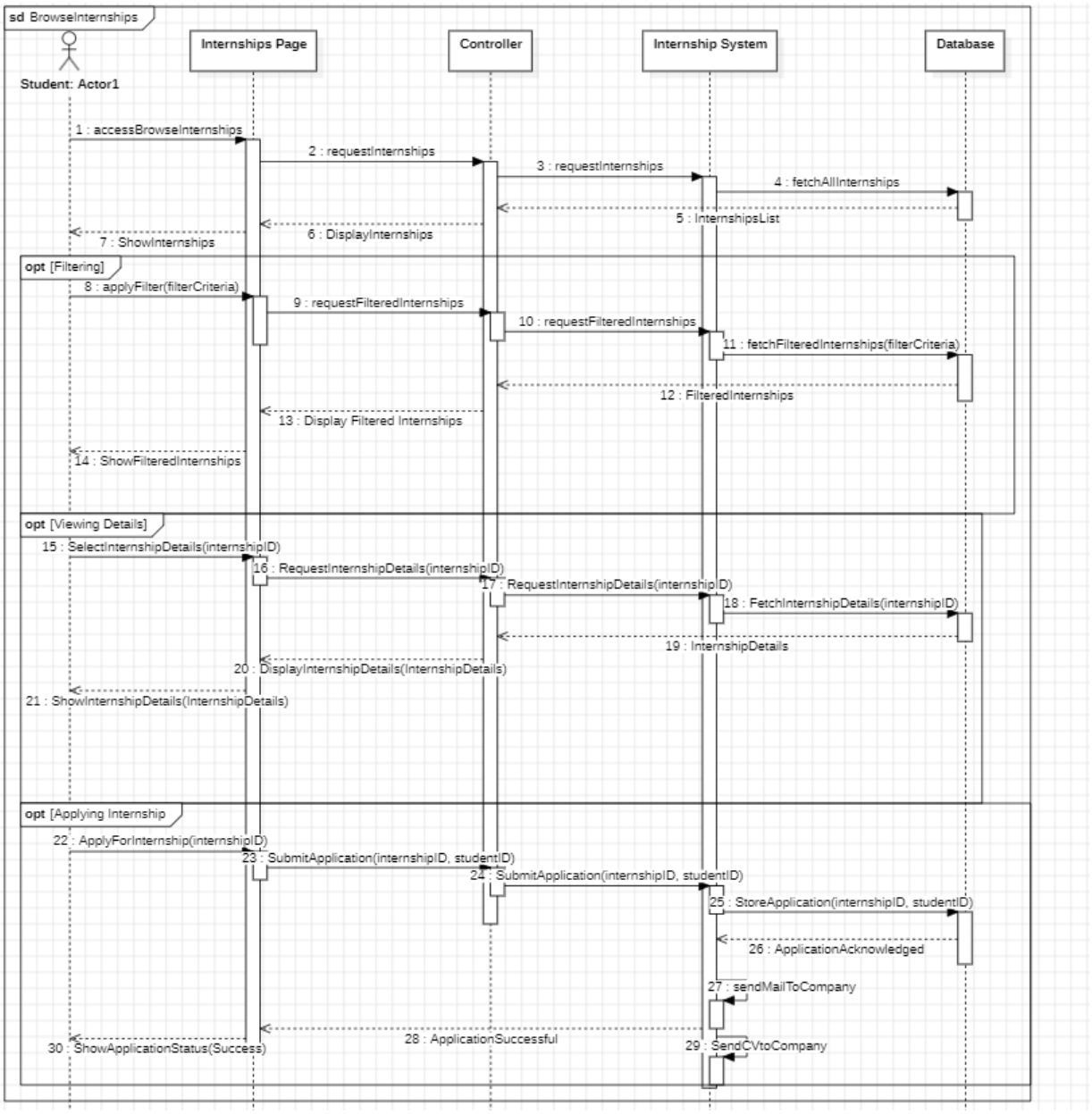


Figure 23: Sequence Diagram of Browse Internships

Design Rationale: This diagram is about browsing internships in our system. Students can access this via internships page. After that MVC pattern do his work to bring internships to student from database. After viewing internships student can use filtering with 3 options: Country, State and Name. Since all of the filtering methods are same and for sake of simplicity in our sequence diagram, we show them as one filtering. After that student have options such as viewing detail of the internships and applying an specific internship. If they view detail MVC pattern will do his work. If they apply for an internship system will send notification mail to company. Also system will send student's resume to company interface. You can see from Figure ..

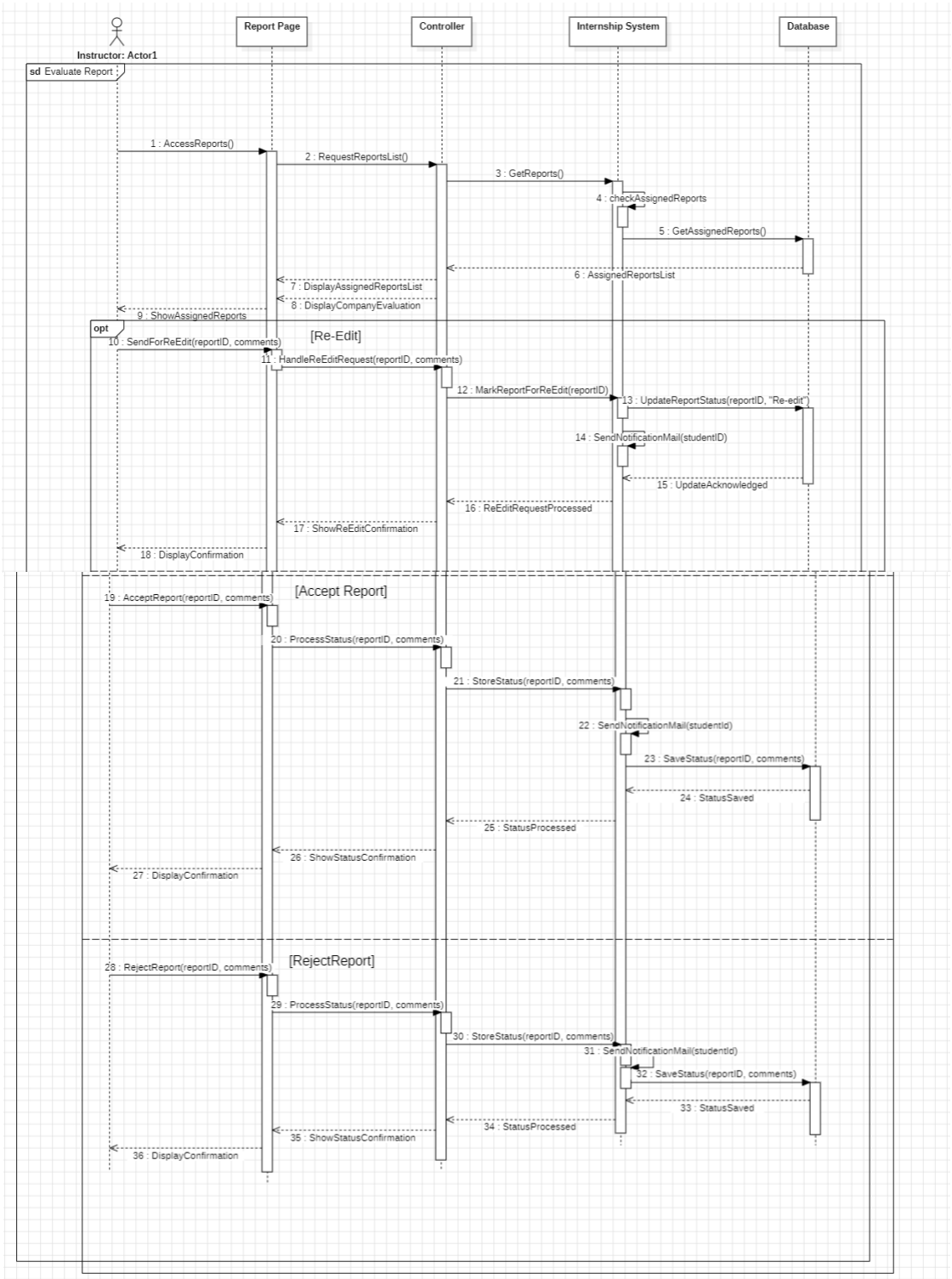


Figure 24: Sequence Diagram of Evaluate Report

Design Rationale: The sequence diagram above is about Evaluating Report of the students. Instructors are responsible for this. They first need to open the reports page, after that system will show assigned reports to the instructors. Then they have 3 choices which are: Send for Re-edit; they send report back to student and students will get notify in the process. Accept Report; they approve the report with grade feedback, and they can also see the Reject Report. You can see from Figure 24.

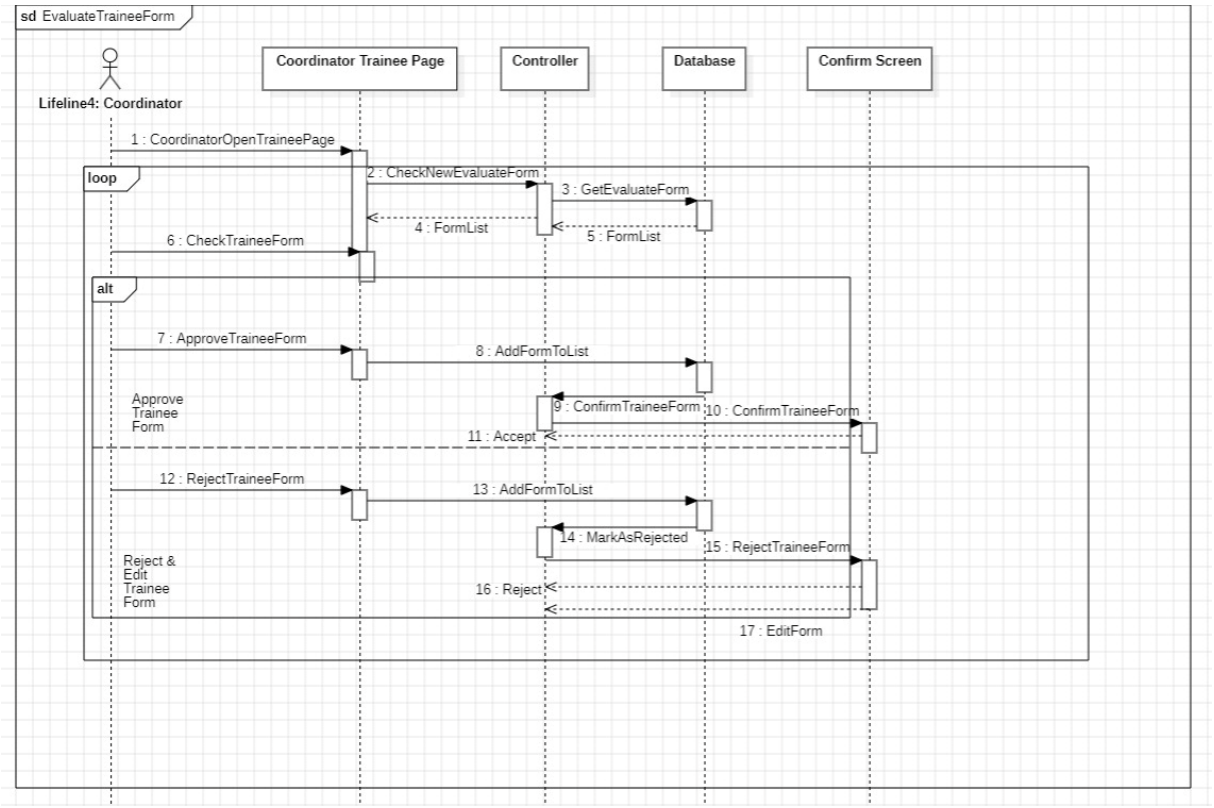


Figure 25: Sequence Diagram of Evaluate Trainee Form

Design Rationale: The sequence diagram describes how the coordinator evaluates trainee forms. The process involves the following steps:

Check New Forms: Coordinator checks new evaluation forms. The system retrieves the list of forms from the database.

Approve Form: If a form is approved, it is added to the approved list, confirmed, and displayed on the confirmation screen.

Reject Form: If a form is rejected, it is marked as rejected in the database. The coordinator can edit the form before confirming the changes.

The system ensures proper validation and provides notifications for approval or rejection, streamlining the form evaluation process.

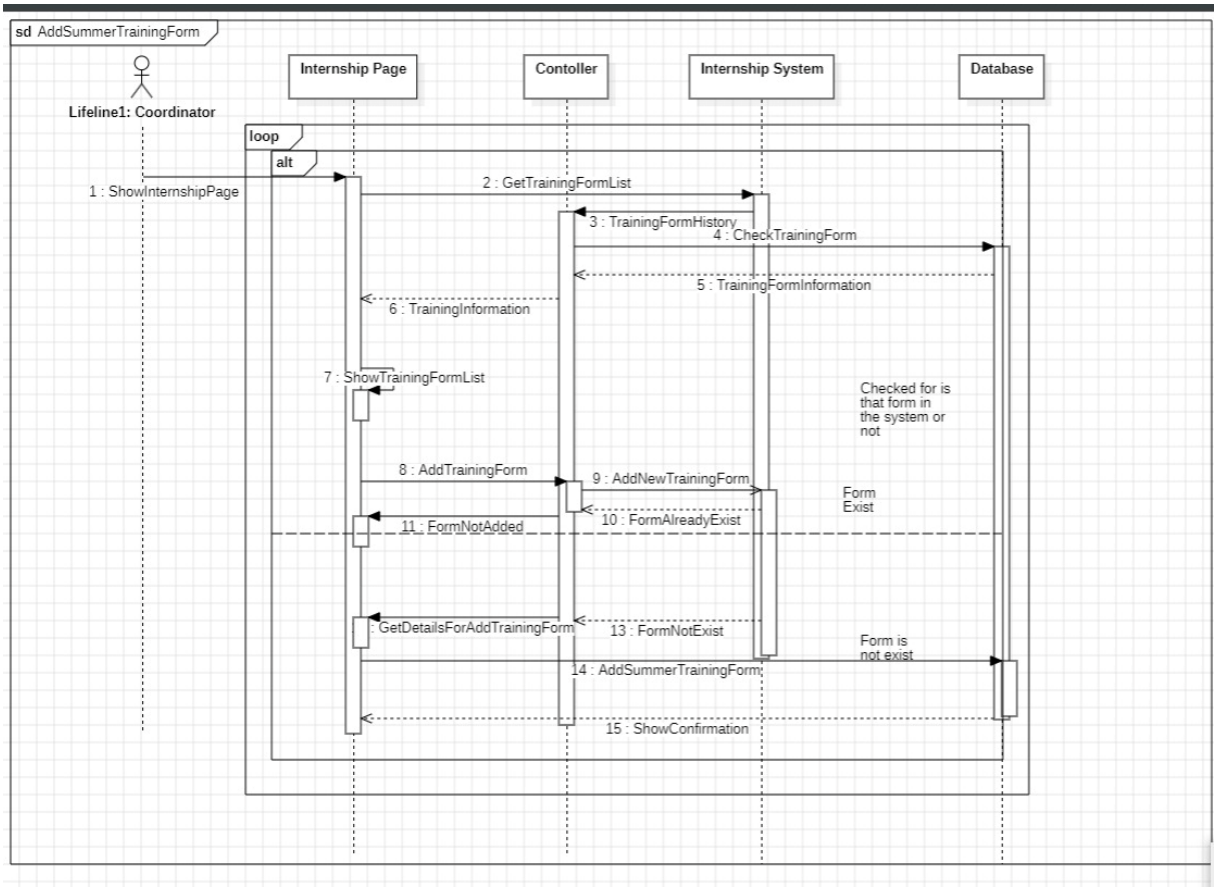


Figure 26: Sequence Diagram of Add Summer Training Form

DesignRationale:

The sequence diagram illustrates the Add Summer Training Form process initiated by coordinator. The process begins when coordinator navigates to the Internship Page and requests the Training FormList. The system then retrieves the list of training forms and checks the existence of the requested form in the database. If the form already exists, the system displays a notification indicating the form's presence. If the form does not exist, the coordinator proceeds to add a new training form. Once the details of the new form are provided, the system validates the input and adds the Summer Training Form to the database. Finally, a confirmation message is displayed, ensuring the successful addition of the form. This process ensures that duplicate forms are avoided, and new forms are systematically added, maintaining data integrity within the system.

4.2.3 Data Flow Diagrams

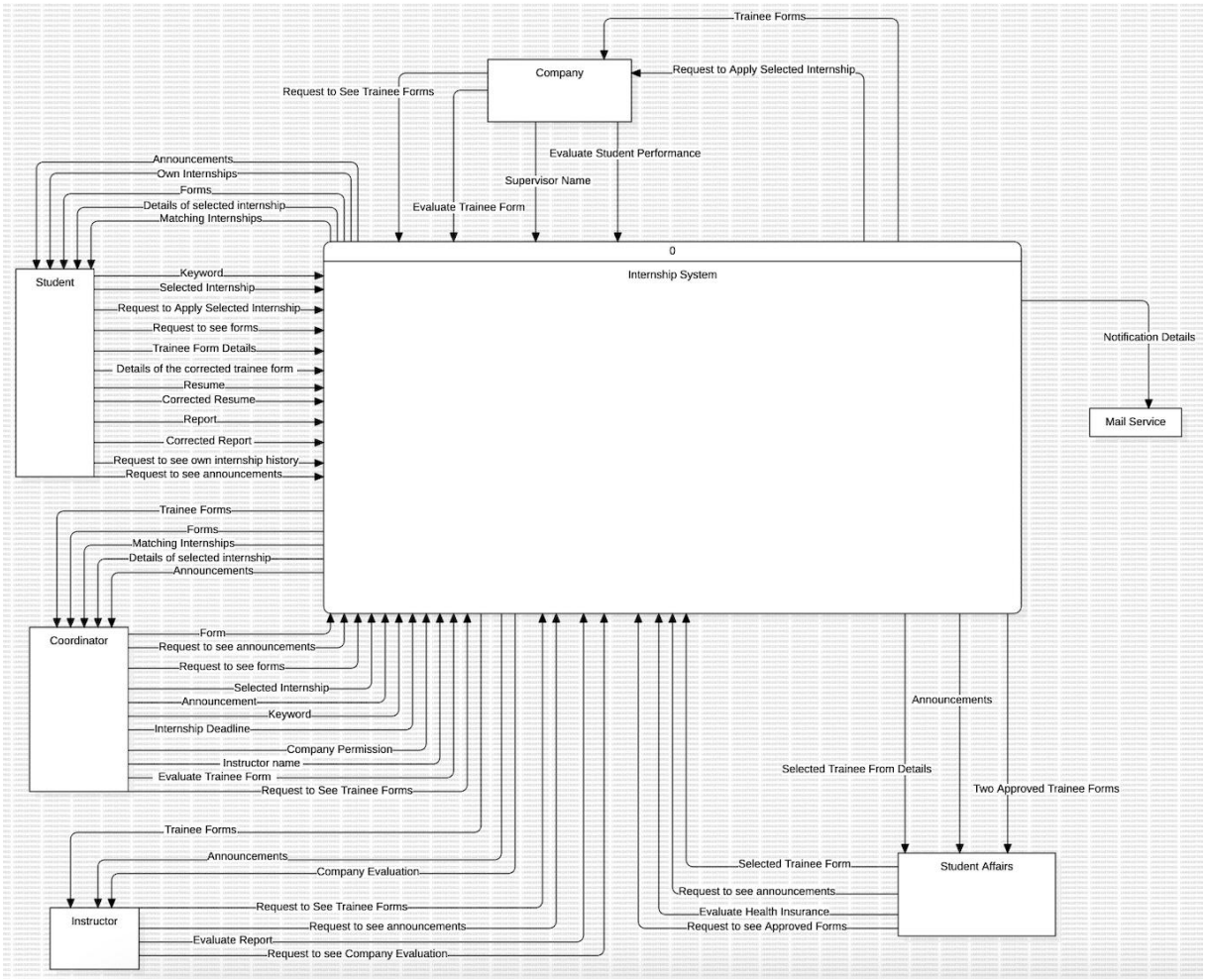


Figure 27: Context Level Diagram – DFD

In this diagram, we showed the main users of our internship system and their tasks. As students, we can apply for internships, see announcements, and add or update our internship forms. Coordinators help us by adding announcements, approving our forms, and assigning instructors to guide us. Instructors check our reports and evaluate them, and they also review feedback from companies about our performance. Companies use the system to evaluate us and share their feedback. Student Affairs supports us by checking our forms and making sure our health insurance is valid. We also added the Mail Service, which sends emails to inform us about important updates in the system. You can see from Figure 27.

Design Rationale for Context Level Diagram

When designing the Context Level Diagram, we aimed to define the overall framework of the system and identify the main user groups. At this level:

Main Purpose of the System: Our goal was to streamline the internship process for all users, including students, coordinators, companies, instructors, and student affairs, ensuring a more efficient and user-friendly experience.

Simple and Inclusive Approach: We focused on showing the interactions between the system and its users in a general way, highlighting each user group's main tasks and connections with the system.

Clear Role Definitions: We ensured that user roles were clearly distinguished based on their responsibilities in the system. For example:

- Students can apply for internships, view announcements, and update their internship forms.
- Companies provide feedback and evaluate performance.
- Coordinators create announcements, approve forms, and assign instructors.

Defining Boundaries: We illustrated the system's boundaries and its interactions with external components like the Mail Service. This helped clarify how the system integrates with external processes.

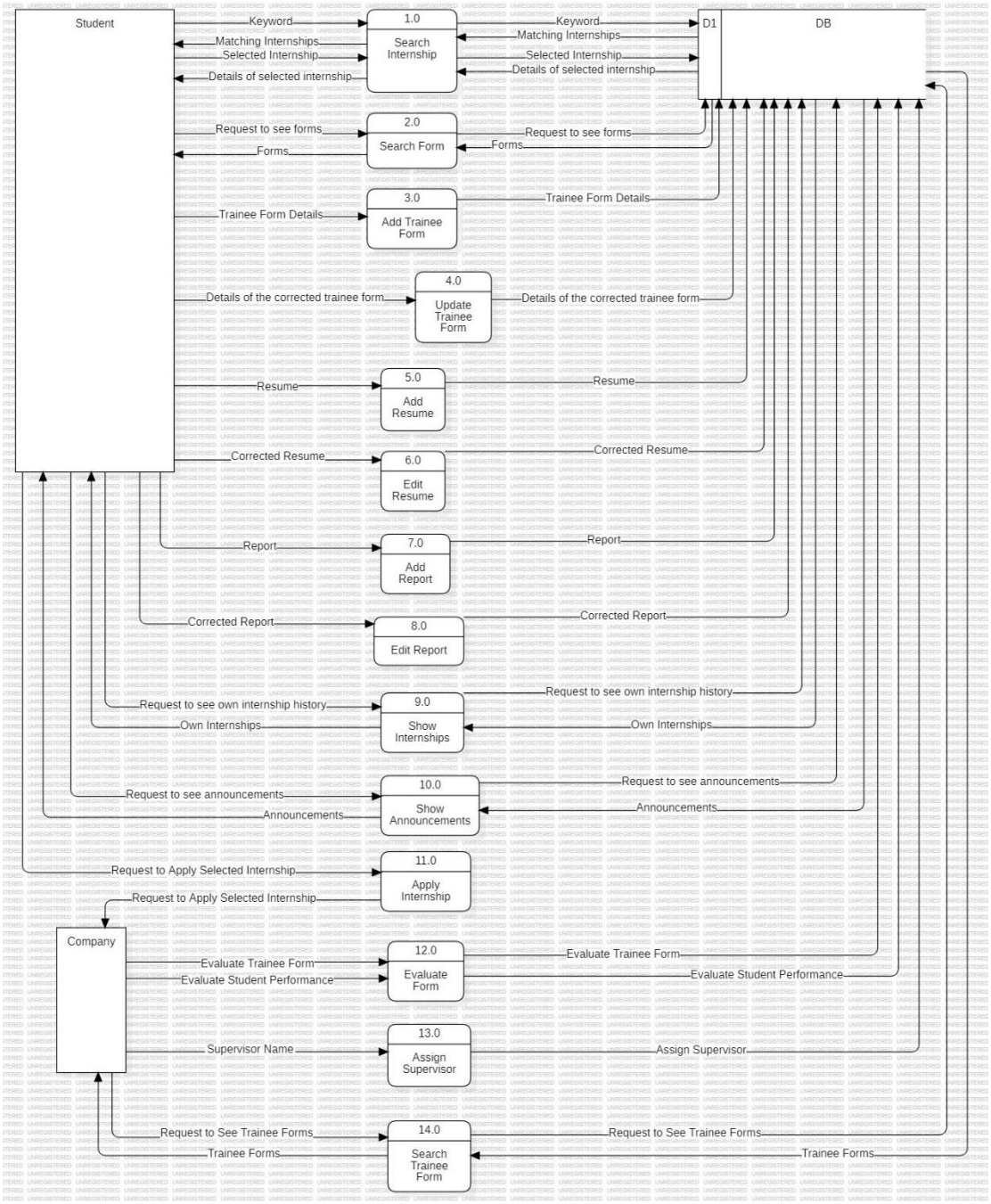


Figure 28: Level 0(1) – DFD

At this level, we explained the actions that both students and companies can do in our system. As students, we can search for available internships (1.0), add our trainee forms (3.0), or update them when needed (4.0). We can also see announcements about internships (10.0) and apply for internships that interest us (11.0). On the other side, companies evaluate us based on our internship performance and add this feedback to the system (6.0). All these actions are saved in the database, so everything stays up to date. For example, when we apply for an internship, the application is recorded in the system and can be seen by the coordinator. Similarly, companies' evaluations are saved and used by instructors to assess our overall performance. This helps create a smooth and organized internship process for both students and companies. You can see from Figure 28.

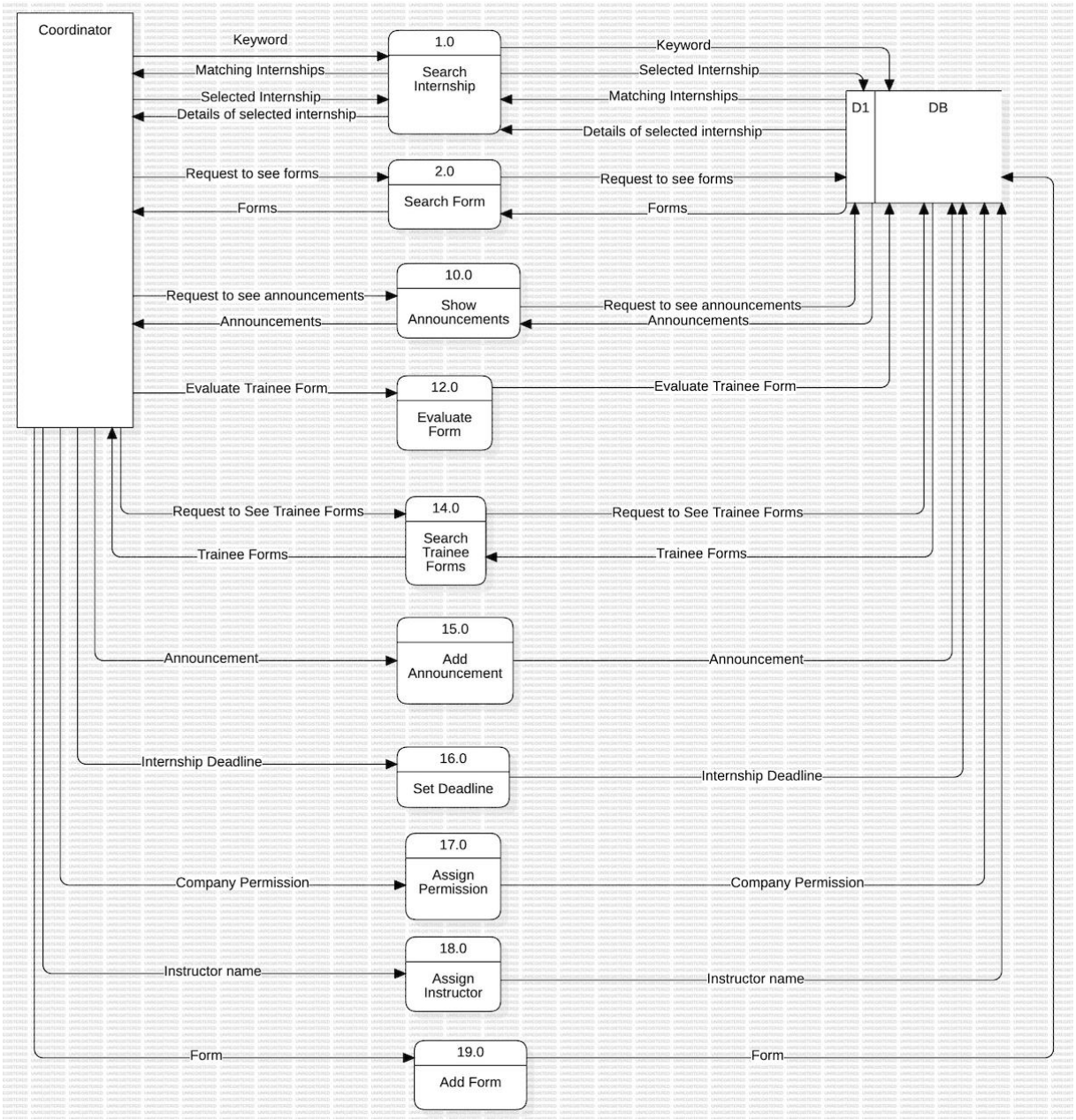


Figure 29: Level 0(2) – DFD

At this level, we focused on the coordinators' tasks. Coordinators can search for internships in the system (1.0) and add new forms to update information (19.0). They can also create announcements to share updates with students and instructors (14.0). Another task is assigning instructors to students for evaluating reports (17.0). For example, when a coordinator adds an announcement, it becomes visible to all students and instructors. These tasks help coordinators manage the internship process effectively. You can see from Figure 29.

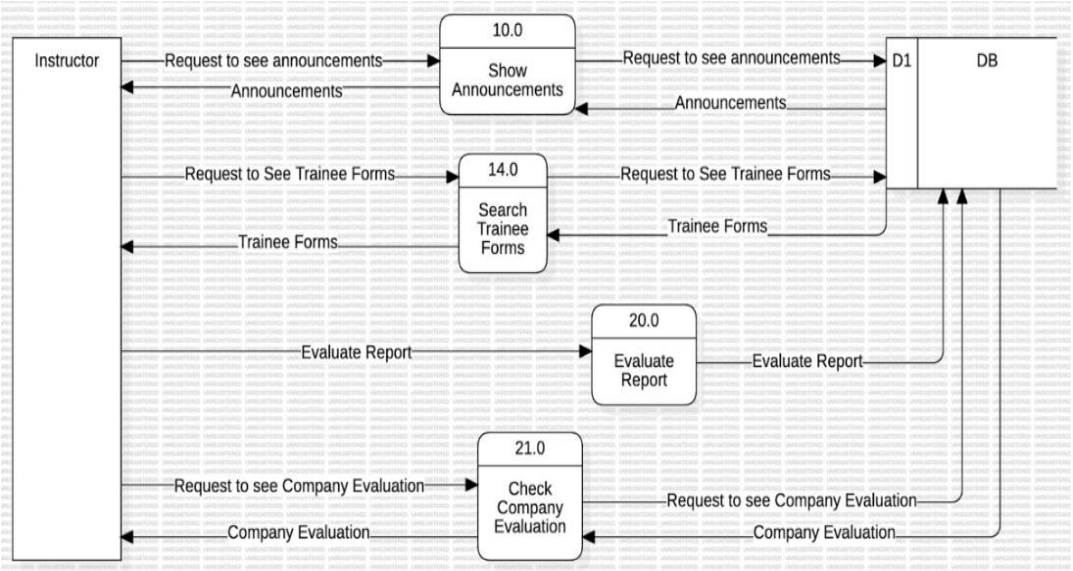


Figure 30: Level 0(3) – DFD

At this level, we explained the tasks of instructors. Instructors can review and evaluate students' internship reports (20.0). They can also check the feedback provided by companies about the interns (21.0). Additionally, they can see announcements shared by coordinators or other users in the system (10.0). For example, when an instructor evaluates a report, the feedback is saved in the system and students can see it. These tasks allow instructors to give useful feedback and assess students' performance during the internship. You can see from Figure 30.

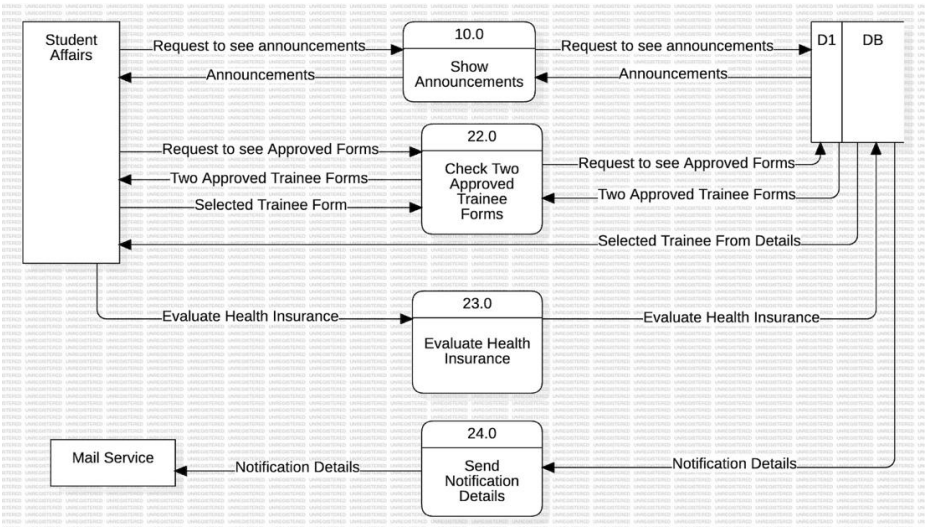


Figure 31: Level 0(4) – DFD

At this level, we described the tasks of Student Affairs. They check if the internship forms submitted by students are correct and approve them. They also evaluate if the students' health insurance meets the requirements for internships. For example, when a form is approved, this information is saved in

the system and can be viewed by both the student and the coordinator. Checking health insurance ensures that students are fully prepared for their internships. These tasks help complete all formal requirements for the internship process. You can see from Figure 31.

In our system, notifications are sent automatically after specific processes. These notifications help users stay updated. We decided not to show this process in the Data Flow Diagram (DFD) because it would make the diagram more complicated. Instead, you can refer to Figure 32 For the detailed table explaining how the notification process works.

No	Event / Condition	Recipient(s)	Description
1	When a student applies for an internship	Company	Application information should be sent
2	When a student fills a new internship form	Coordinator and Company	Should be sent to both at the same time
3	When a student uploads the internship report for the second time	Instructor	"Uploaded for the 2nd time" warning
4	When the coordinator publishes an announcement	Students, Instructor, Student Affairs	Announcement notification
5	When the coordinator sets an internship deadline	Students	Reminder mail
6	When a student adds a new company	Company	Password and system access info should be sent
7	When the instructor gives feedback on the student's report	Student	Feedback notification
8	When the coordinator or company approves/rejects the internship form	Student	Application status notification
9	When the company adds an evaluation	Student	Notification that company evaluation is added
10	When it's time to remind the company to fill the evaluation form	Company	Reminder mail (can be scheduled automatically)
11	When Student Affairs approves/rejects the health insurance on the internship report	Student	Health insurance status notification
12	When it's time for the company's annual activity check	Company	"Are you active?" themed reminder mail
13	As the student's final report upload deadline approaches	Student	Deadline reminder
14	When the instructor gives the student's grade	Student	Grade information should be sent

15	When an instructor is assigned to a student	Student	Assigned instructor info should be sent
16	When a new internship form is approved	Student Affairs	Notification email that the form is approved
17	When a student applies to an open internship posting	Company	Informative email should be sent to the company
18	When a company posts a new internship offer	Students	Relevant students should be notified
19	When a company approves or rejects an application	Student	Approval/rejection result should be emailed

Figure 32: Notification Table

Design Rationale for Level 0 Diagram

For the Level 0 Diagram, we focused on breaking down the main tasks performed by each user group identified in the Context Level Diagram. At this level:

Breaking Down Main Tasks: We detailed the primary tasks for students, companies, coordinators, instructors, and student affairs. For instance:

- Students can view announcements (10.0), apply for internships (11.0), and update their forms (4.0).
- Companies provide feedback on internship performance (6.0).
- Coordinators create announcements (14.0) and assign instructors (17.0).

Illustrating Data Flows: We demonstrated the flow of information between different components of the system, linking user actions to their corresponding processes. This clarified how the database updates and how users stay informed about one another's actions.

Maintaining Consistency: We emphasized that every action in the system has a corresponding result saved in the database. For example:

- A student's application is visible to the coordinator for further action.
- Feedback provided by companies is accessible to instructors for evaluation purposes

Managing Complexity: We chose not to include detailed processes, like notifications, in the diagram to avoid overcomplication. Instead, we explained these processes in other sections of the document.

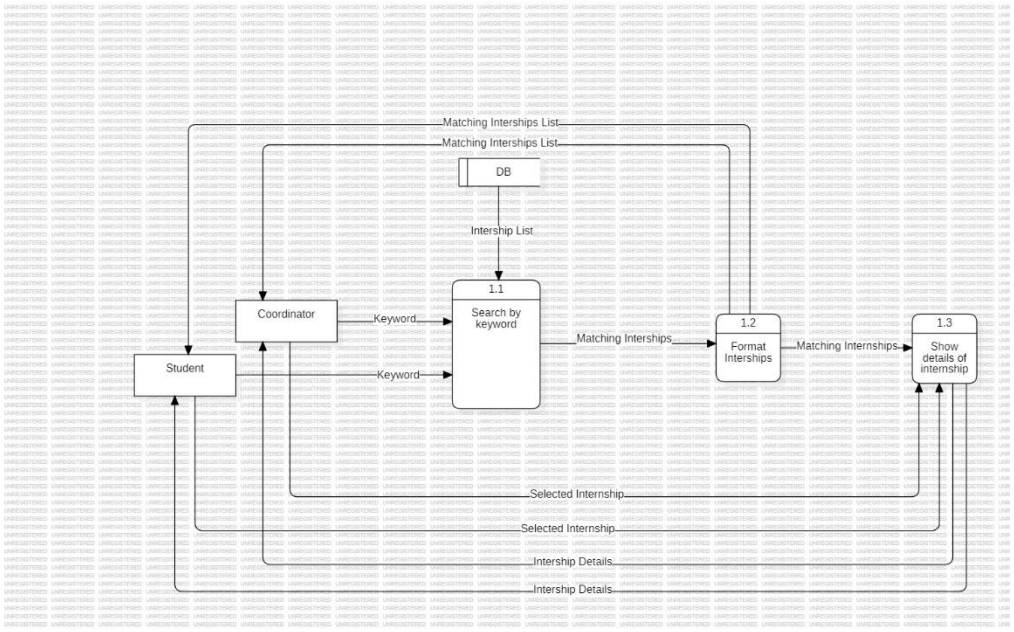


Figure 33: Level 1(1) – DFD

In the diagram in Figure 33, we have described the process by which students and coordinators can search for internships using keywords. The system fetches relevant internships from the database based on the input keyword. These internships are then formatted into a list of matches, which is displayed to the users. From this list, users can select an internship to view its detailed information.

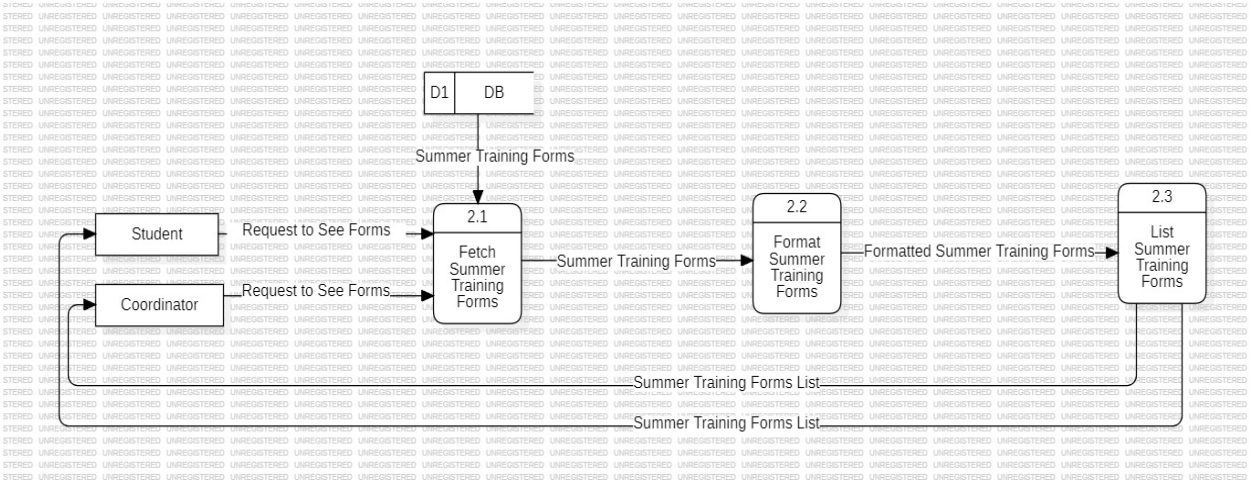


Figure 34: Level 1(2) – DFD

In this diagram, we have described how students and coordinators can view the summer training forms. The Coordinator or Student sends a request to fetch forms. The system retrieves the forms from the database. The form objects are formatted through the Format Summer Training Form process to make it suitable for display. The formatted forms are listed and sent to the Coordinator or Student. You can see from Figure 34.

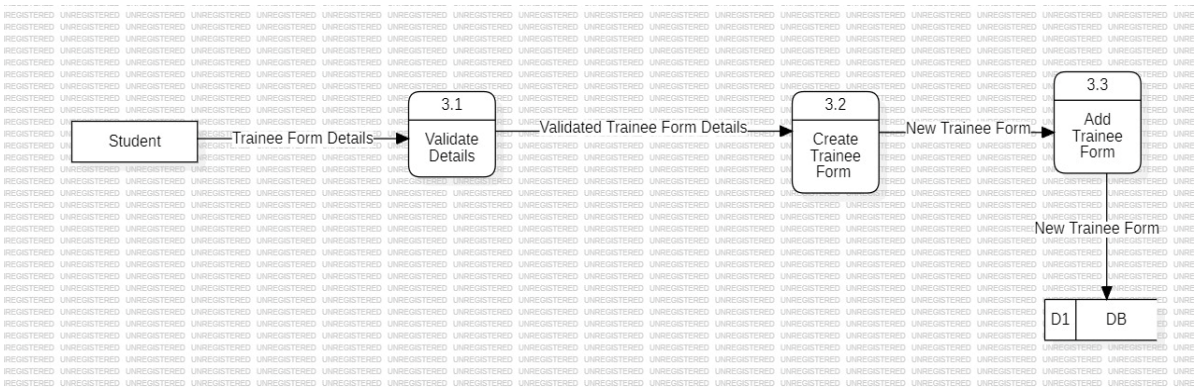


Figure 35: Level 1(3) – DFD

In this diagram in Figure 35, Student provides Trainee Form details, which are passed to the Validate Details process. This process ensures the input is valid before proceeding. The validated data is then sent to the Create Trainee Form process, which generates a trainee form object. The form is then passed to the Add Trainee Form process, which saves the information into the database.

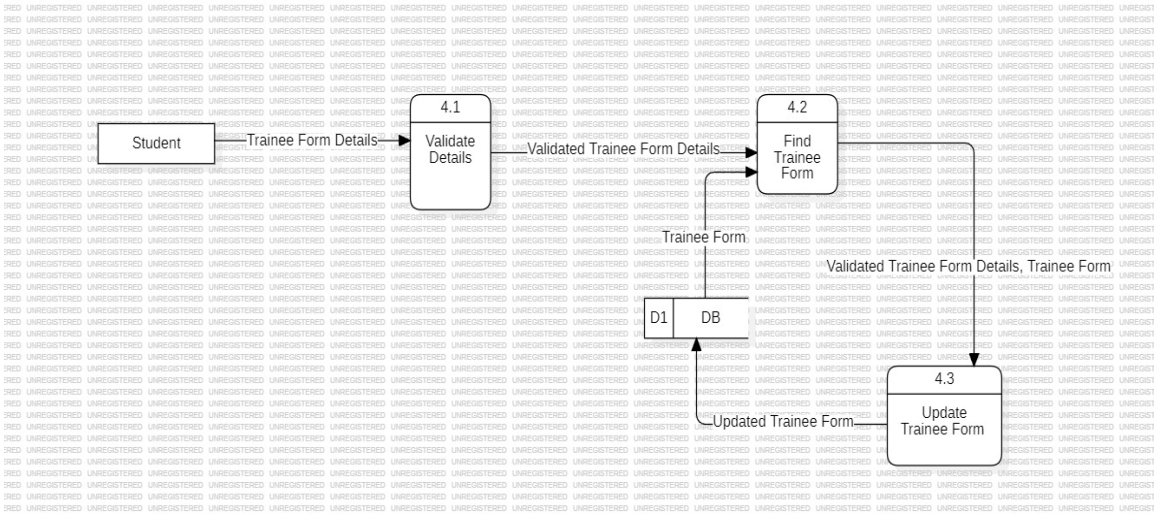


Figure 36: Level 1(4) – DFD

In this diagram, we have described how students can update their trainee forms. The process begins when the student provides their details to be validated via Validate Details process. Once validated, the system proceeds to Find Trainee Form process, which retrieves the trainee form to be updated from the database. Finally, Update Trainee Form process modifies the trainee form based on the updated information, and the updated form is saved back into the database. You can see from Figure 36.

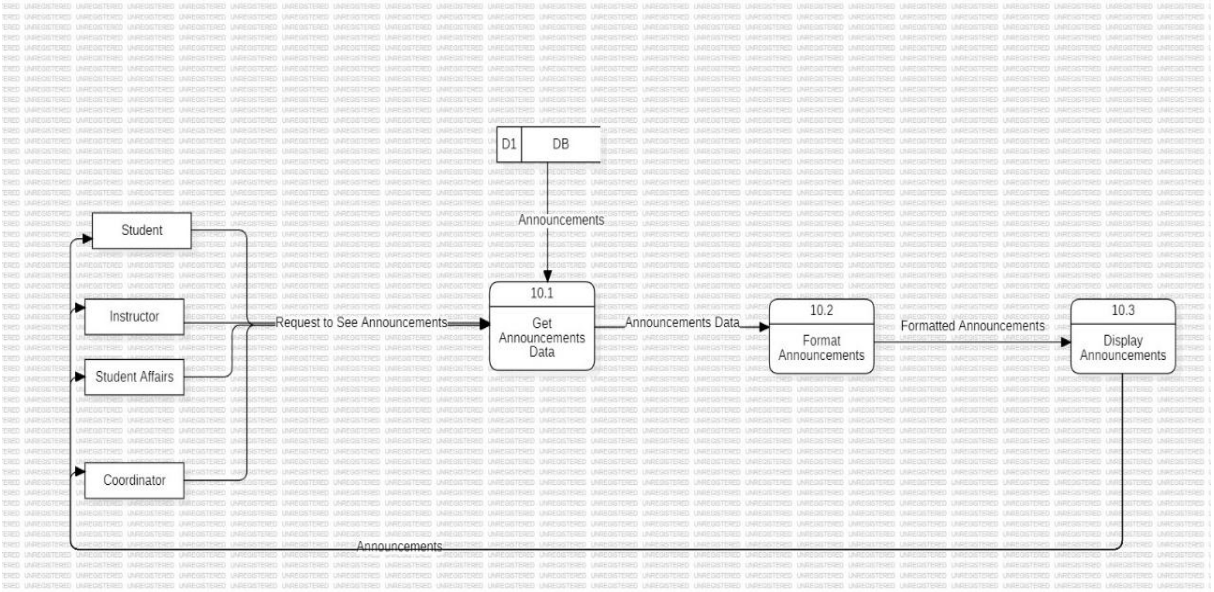


Figure 37: Level 1(10) – DFD

In this diagram, we have described how students, instructors, student affairs and coordinators can view the announcements. First, they interact with the system and Get Announcements Data process runs, which retrieves data from the database containing announcements. This data is then formatted through Format Announcements process, where the raw data is organized into a structured format suitable for display. Finally, the formatted internship details are presented back to the student through the Display Announcements process. You can see from figure 37.

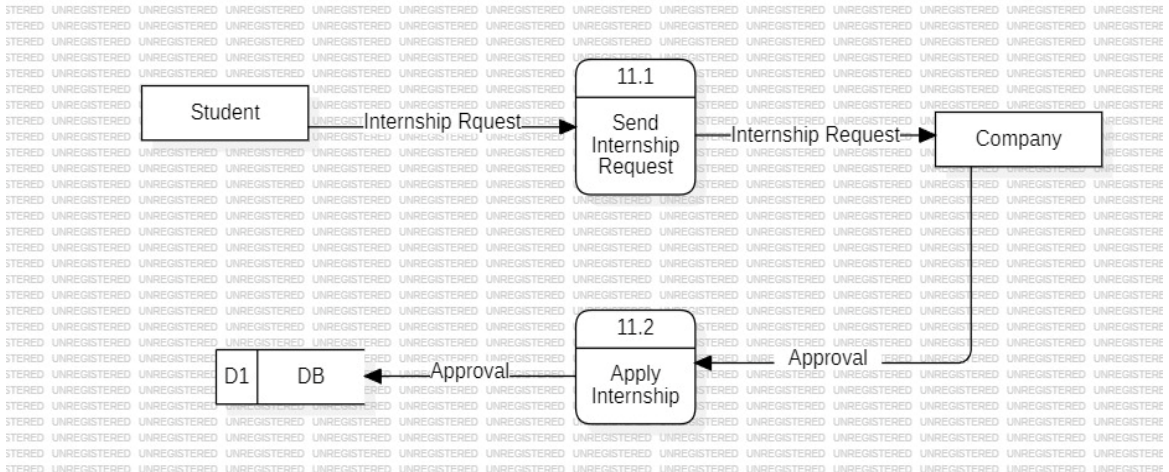


Figure 38: Level 1(11) – DFD

In this diagram in Figure 38, we have described how students can apply for an internship and how the company approve internships via the internship system. Student initiates the process by sending an internship request to a Company. The company processes this request and applies for the internship, storing the relevant data in the database.

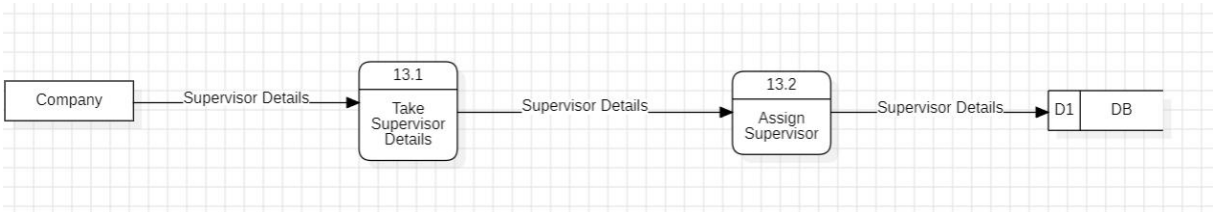


Figure 39: Level 1(13) – DFD

In this diagram, we have described how companies can fill supervisor details and assign a supervisor for an internship. The company provides the supervisor. Then this information flows to the Assign Supervisor process, where the supervisor details are being directed to the database. You can see from Figure 39.

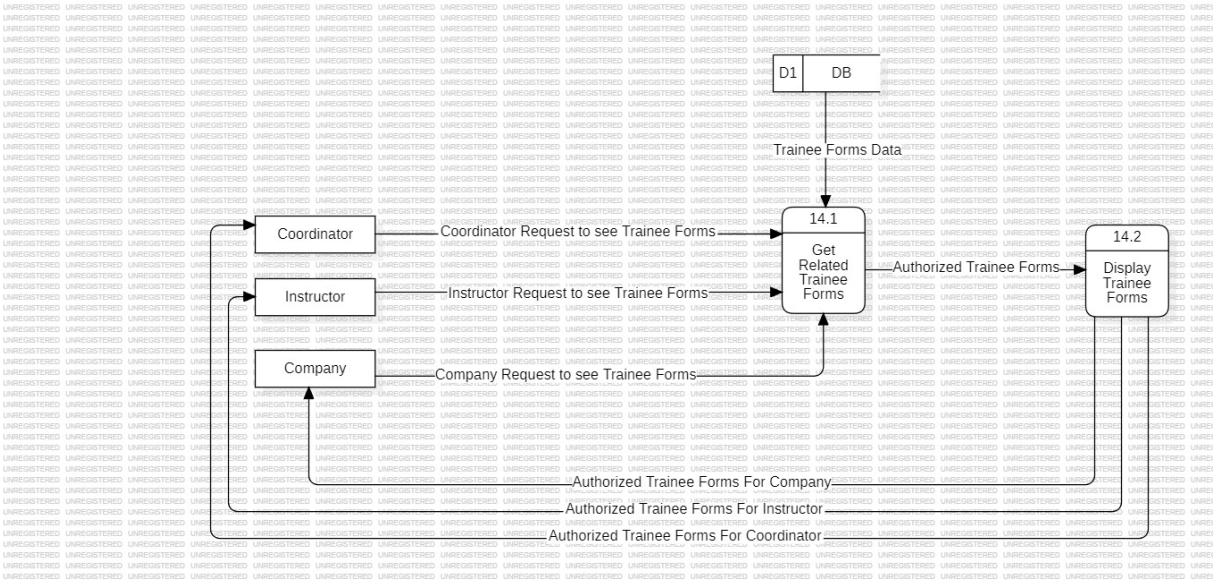


Figure 40: Level 1(14) – DFD

In this diagram in Figure 40, we have described how companies, instructors and coordinators can view trainee forms that they have access to. First, they send a request to reach their Trainee Forms. These forms are fetched by Get Related Trainee Forms process. Then these forms are directed to display trainee forms that displays the Authorized Trainee Forms for a specific company, instructor or coordinator.

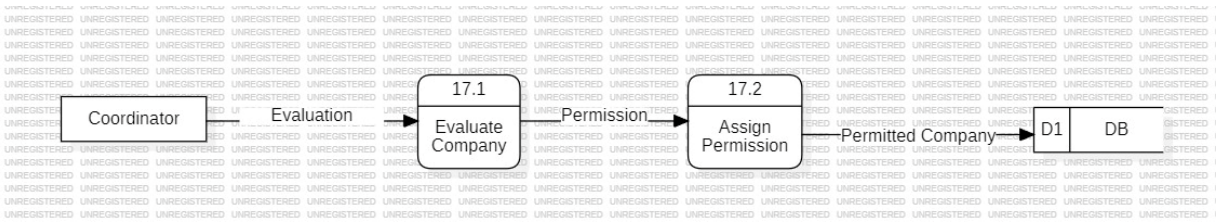


Figure 41: Level 1(17) – DFD

In this diagram in Figure 41, we have described how coordinators can evaluate and approve a company into the internship system. When coordinator use the system Evaluate Company process lets coordinator to evaluate the company. This process takes the evaluation input and directs permission to Assign Permission process. This process saves the company details as a permitted company in the database.

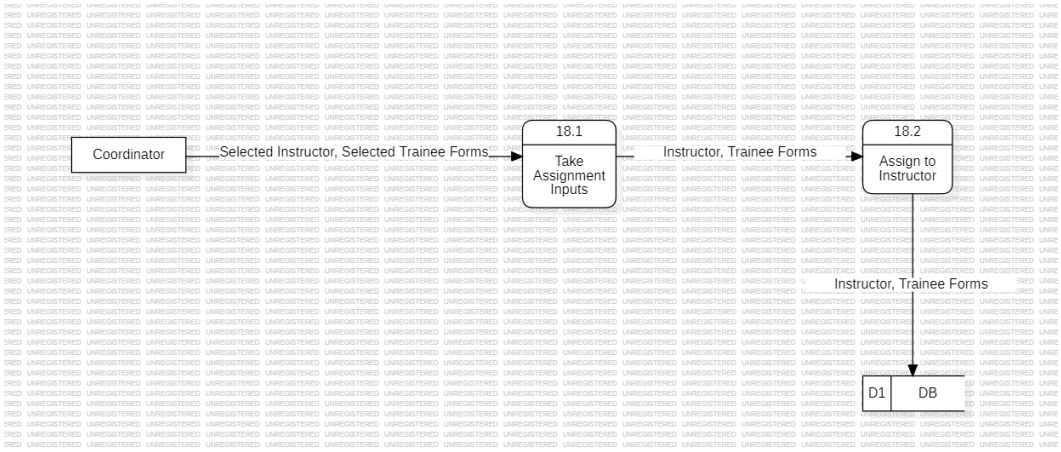


Figure 42: Level 1(18) – DFD

This diagram in Figure 42, shows how we have described how coordinators can assign instructors to trainee forms. Coordinator selects an instructor for a trainee form and Take Assignment Inputs process runs. Then Selected Instructor and Trainee Form being conveyed to the Assign to Instructor process to be saved in the database.

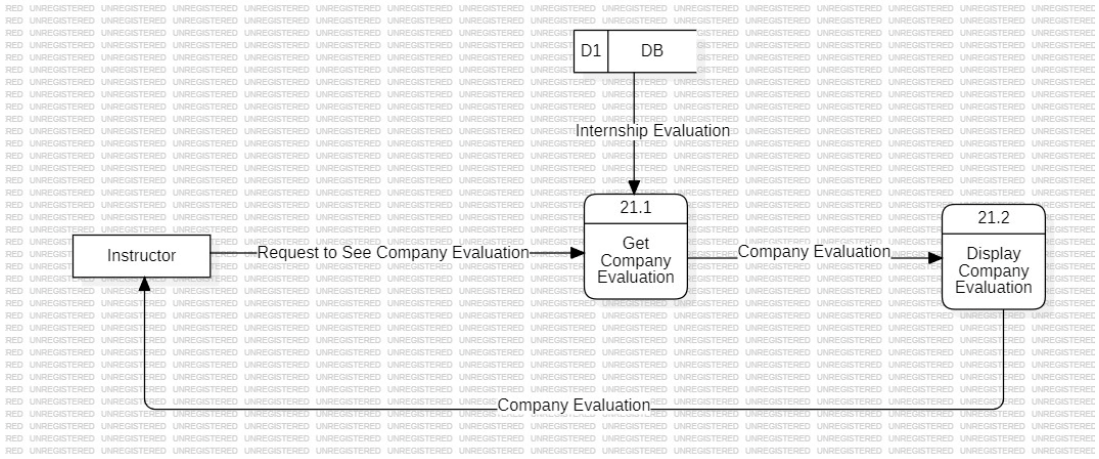


Figure 43: Level 1(21) – DFD

In this diagram, we have described how instructors can view company evaluations. The instructor interacts with a system to obtain and view company evaluations. The process begins when the instructor initiates a request to Get Company Evaluation, which involves fetching data from the database. Once the data is retrieved, it is passed to the Display Company Evaluation process, which formats and presents the evaluation to the instructor. You can see from Figure 43.

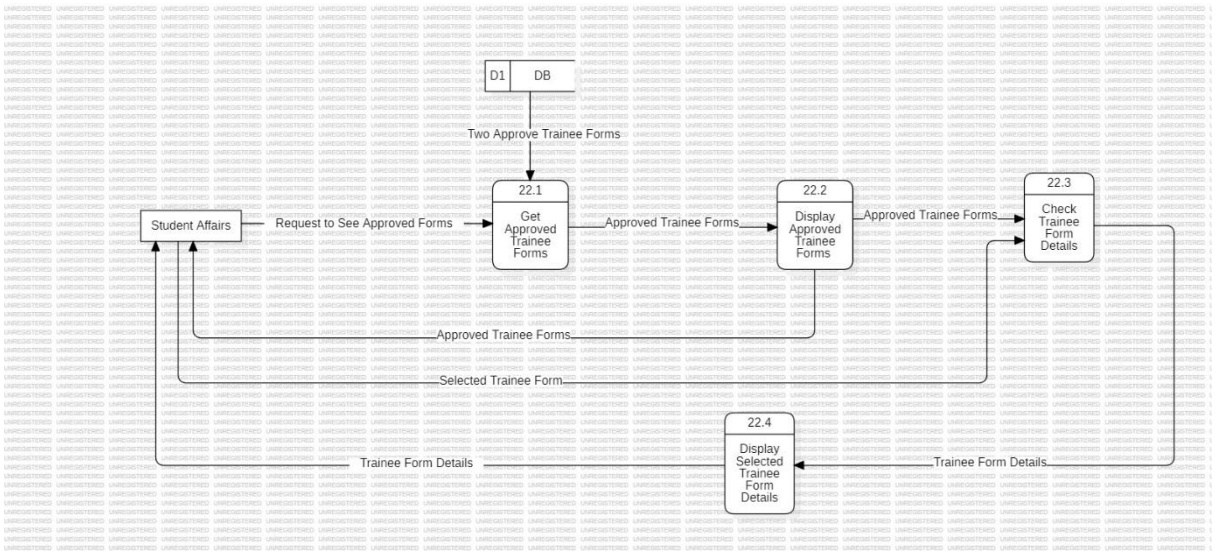


Figure 44: Level 1(22) – DFD

In the diagram in Figure 44, we have described how student affairs can view the approved trainee forms and select a trainee form to see its details. Student Affairs initiates the process by requesting forms from the Get Approved Trainee Forms process, which retrieves data from the Trainee Forms the database. The system then proceeds to Display Approved Trainee Forms, displaying approved trainee forms. Student Affairs can also select a trainee form to run Check Trainee Form Details process. The system concludes by sending Display Selected Trainee Form Details back to Student Affairs, providing the necessary details for review.

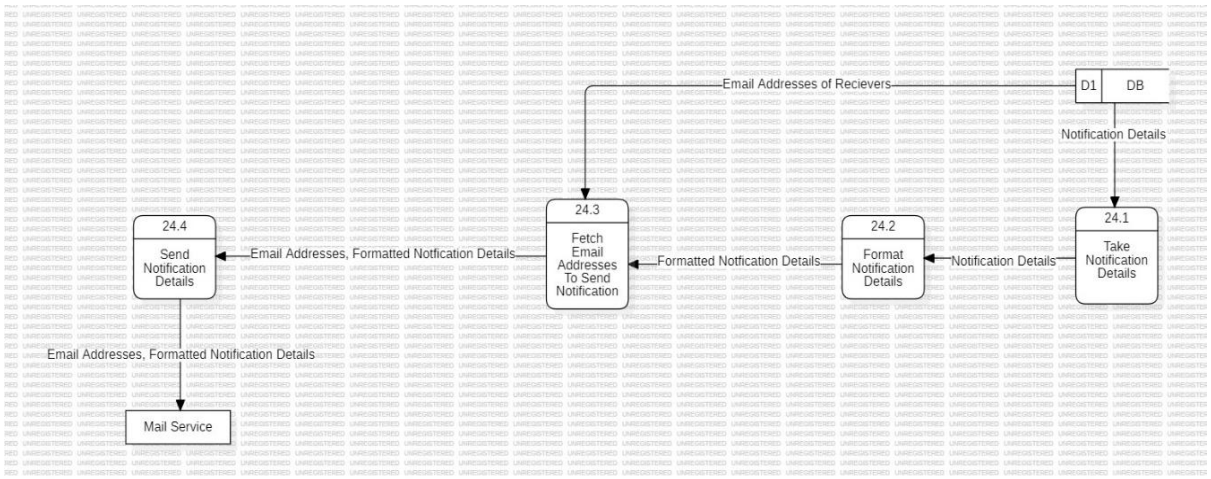


Figure 45: Level 1(24) – DFD

As shown in Figure 45, in this diagram, we have described how notifications are created and sent through the mail service with the details provided. When sending a notification, notification details is first retrieved from the database. These details are processed by the Take Notification Details process, which passes them to the Format Notification Details process for formatting. Once formatted, the Fetch Email Addresses to Send Email process retrieves email addresses to send email from the database. The formatted notification details, along with the fetched email addresses, are then passed to the Send Notification Details process. Finally, the notification details are forwarded to the Mail Service which is responsible for delivering the emails. This data flow diagram highlights how notifications are managed and sent systematically.

Design Rationale for Level 1 Diagram

For the Level 1 Diagram, we focused on breaking down the process in more detail than the Level 0 This level of DFD helps developers understand how the system works internally and provides a foundation for implementation. At this level:

Detailing Processes: We broke down the main process into smaller, manageable sub-processes.

Identifying Sub-Processes: We decomposed the high-level processes into its sub-processes.

Maintaining Context: We ensured it aligns with the system goals and the Level 0 DFD, providing consistency.

Facilitate Understanding: The diagram serves as a bridge for system analysis by providing more functions while avoiding overwhelming detail.

4.3 Development View

The development view shows how the software is decomposed for development.

4.3.1 Component and Deployment Diagram

The Internship System stores its data in a centralized database and retrieves it when needed. Forms and announcements can be added and managed directly within the system.

There are five different types of users interacting with the system: Coordinator, Student, Company, Instructor, and Student Affairs. All these users can be contacted through the integrated Mail System, which connects to an external Mail Provider.

Trainees (students) are able to manage their internship processes by:

- Filling out and submitting Trainee Information Forms,
- Uploading and updating their Resume, and
- Uploading their Internship Reports, which are subject to review by instructors or coordinators.

The Resume component is directly associated with the Student and allows them to maintain their personal and academic background for application purposes. It is accessed and stored via the Internship System.

The Report component enables students to submit internship reports, which may be evaluated by instructors or authorized staff. This data is also routed through the Internship System to ensure consistency and accessibility.

In addition, companies have access to Performance Evaluation Forms to evaluate student performance during the internship period.

This structure ensures centralized control, clear role-based responsibilities, and smooth communication between all entities in the system.

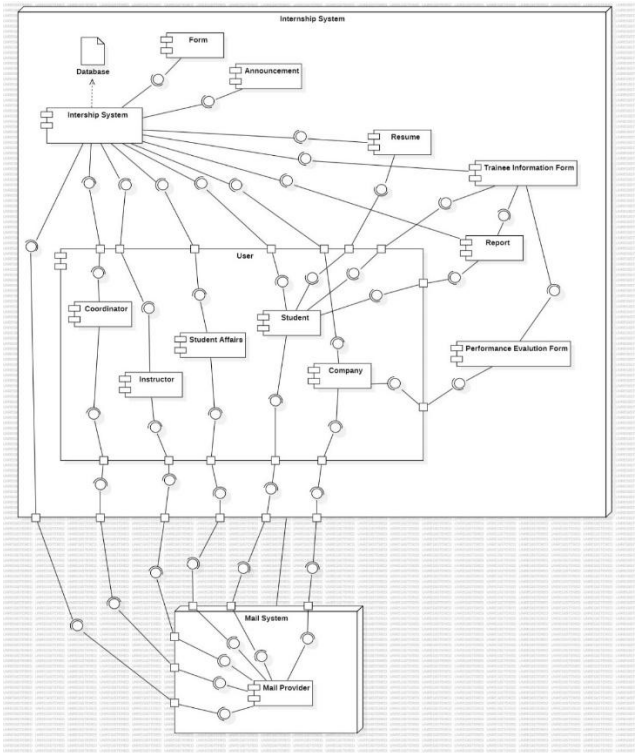


Figure 46: Component and Deployment Diagram

5 Software Implementation

We implemented our software using the following technologies:

PostgreSQL: PostgreSQL is an open-source relational database management system. In our project, PostgreSQL was used to store and manage application data securely and efficiently. Its support for advanced SQL features, trigger functions for automated data changes, indexing, and data integrity made it a reliable choice for our DBMS. We used PostgreSQL 15 version of PostgreSQL in our project.

Spring Boot: Spring Boot is an open-source Java framework and an extension of the Spring framework, widely used for building web applications. In our project, Spring Boot was used to develop the backend services, offering features such as dependency injection, REST API support for communication between the frontend and backend, and seamless integration with PostgreSQL databases and security modules. Maven was used to handle project dependencies. We used version 3.3.6 of Spring Boot.

Hibernate / Spring Data JPA: Hibernate is an object-relational mapping (ORM) tool for Java, and Spring Data JPA is a part of the Spring ecosystem that simplifies database access by creating JPA-based repositories. We used Hibernate and Spring Data JPA to manage database interactions efficiently. These tools allowed us to reduce the raw SQL codes we needed and let us focus on the business logic while ensuring data consistency and transaction management.

Angular: Angular is a framework of TypeScript used for front-end web applications developed by Google. Angular is a component-based framework. Each component is an independent UI block with its own template, style, and logic units. We also used services which are used to provide data sharing between frontend and backend, working as singletons. This structure enhances code modularity and reusability. 19.0.4 version of Angular was used to develop the front-end of our application.

Bootstrap: Bootstrap is a popular front-end CSS framework that provides pre-designed components and responsive grid systems. In our project, Bootstrap was used to design UI components quickly.

TailwindCSS: TailwindCSS is another popular CSS framework that allows for rapid UI development using utility classes. It was used in conjunction with Bootstrap to enhance the customization of our front-end design. It offered our front-end team the control over styling without writing many custom CSS from scratch.

JUnit / Mockito: JUnit is a widely used testing framework for Java, and Mockito is a mocking framework that enables the creation of mock objects for unit testing. In our project, JUnit and Mockito were used to write and execute unit tests for backend services, ensuring code correctness, reliability, and facilitating test-driven development.

Playwright: Playwright is an end-to-end testing framework developed by Microsoft, supporting multiple browsers and platforms. We used Playwright to automate integration testing for the front-end, enabling us to simulate user interactions and verify application behavior.

Google Cloud SQL: Google Cloud SQL is a relational database service provided by Google Cloud Platform. It supports PostgreSQL, MySQL, and SQL Server, offering automated backups, replication, failover, and patch management. In our project, we used Google Cloud Console SQL with PostgreSQL to host and manage our database in the cloud. This allowed us to have high availability, scalability, and secure access over the database.

6 Software Testing

6.1 Unit Testing

6.1.1 Test Procedure

In our project, we performed unit testing on service classes to check if the methods work correctly. JUnit 5 and Mockito were used. With Mockito, we could mock repository classes, so we didn't need to use a real database. We mainly tested methods that save, get, or delete data. For example, in the test for `saveAnnouncement()`, we checked if the method returns the correct announcement object after saving. We used `@Mock`, `@InjectMocks`, and @ExtendWith(MockitoExtension.class)` to prepare the testing environment. All unit tests were written by the backend team and executed successfully.

Umutcan performed all the tests in the unit testing.

6.1.2 Test cases

ID	***Description***	***Steps***	***Test Data***	***Pre-condition***	***Expected Output***	***Passed /Failed***
TC1	Test saveAnnouncement() method	1. Create a mock Announcement object 2. Mock repository save method 3. Call saveAnnouncement() 4. Check returned result	Content: "Important Message"	Repository mocked with Mockito	Returned Announcement has correct ID and content	Passed
TC2	Test getApprovedTraineeInformationForms() method	1. Mock findAll() method 2. Call the service method 3. Check list size	Two mock form objects	Repository mocked with Mockito	Returns list of 2 forms	Passed

TC3	Test updateFormStatus() when form exists	1. Mock findById() with existing form 2. Call updateFormStatus() 3. Check returned boolean and form status	Status = "Approved"	Form found in repository	Returns true, status updated	Passed
TC4	Test approveInsurance() logic	1. Mock findById() and user repo 2. Call approveInsurance() 3. Check fields and verify calls	Form with ID, username	Valid internship form in DB	Sets approval true, logs it, sends mail	Passed
TC5	Test sendVerificationEmails() method	1. Create mock branches 2. Mock findAll() 3. Call method 4. Check email service	2 mock branches	Repository mocked	Email sent for each branch	Passed
TC6	Test saveCompanyBranch() method	1. Create a mock CompanyBranch 2. Mock save and email functions 3. Call saveCompanyBranch() 4. Verify result	CompanyBranch with email and username	Repository and EmailService mocked	CompanyBranch saved and email sent	Passed
TC7	Test createEvaluationForm() method	1. Mock findById 2. Call createEvaluationForm() 3. Verify save and email methods	Working Day: 20 Performance: "Excellent" Feedback: "Well done"	Trainee form mocked	Evaluation saved and email sent	Passed

TC8	Test addForm() method	1. Create mock AcademicStaff 2. Mock academicStaffRepository.findById() 3. Call addForm() 4. Mock formRepository.save() 5. Check return	File: file.pdf Content: "form content"	AcademicStaff exists in repository	Form is saved and returned with ID 1	Passed
TC9	Test updateInitialFormStatus() for company mail	1. Mock findById() for form 2. Mock last approved form 3. Mock token creation 4. Call method	Status: "Company Approval Waiting"	Trainee form and approved form exist	Status updated, token created, email sent to company	Passed
TC10	Test applyForInternshipOffer() method	1. Mock findByUserName() 2. Mock findById() for InternshipOffer 3. Call applyForInternshipOffer() 4. Verify save method	Username: "student1" Offer ID: 101	Student and internship offer exist	Application is saved successfully in repository	Passed
TC11	Test createAllEvaluations() method	1. Mock findById() for report 2. Prepare mock DTO with 11 scores & comments 3. Call createAllEvaluations() 4. Verify saves & email	Report ID: 1 Username: "student1"	Report exists in DB	11 evaluations saved and email sent to student	Passed

TC12	Test addReport() for second report upload	1. Mock findById() for form2. Mock existing report3. Prepare DTO with file4. Call addReport()5. Verify save and email call	Username: "student1 "Form ID: 1File: report.pdf	Form status is "Approved"1 report exists	Second report saved, email sent to instructor	Passed
TC13	Test addResume() method	1. Mock findByUserName() 2. Create a mock resume file 3. Call addResume() 4. Verify resumeRepository.save() method	Username: "student1 " File: "This is the resume content."	Student exists in repository	Resume is saved and returned successfully	Passed
TC14	Test createPasswordResetToken() method	1. Call createPasswordResetToken() 2. Verify token is saved	Username: "testuser "	-	Token is generated and saved with correct data	Passed

6.1.3 Test Results

- TC1: The test for the saveAnnouncement() method passed successfully. No errors occurred during the execution. The output matched the expected result.

- TC2: The test for getApprovedTraineeInformationForms() passed without errors. It returned the correct list size.

- TC3: The test for updateFormStatus() returned true and the form status was updated as expected.

- TC4: The test for approveInsurance() updated the approval status, saved the log, and triggered the mail function.

- TC5: The test for sendVerificationEmails() passed successfully. The email service was called exactly two times as expected.

- TC6: The test for the saveCompanyBranch() method passed successfully. The method saved the branch and triggered the email service without throwing any exception. The output matched the expected behavior.

- TC7: The test for the createEvaluationForm() method passed successfully. The evaluation form was saved, and the email was sent without any errors.

- TC8: The test for the addForm() method passed successfully. It saved the form using the mocked repository and returned the saved form object.

- TC9: The test for the updateInitialFormStatus() method passed successfully. The method updated the form status, created a reset token, and sent the email to the company without any errors.

- TC10: The test for the applyForInternshipOffer() method passed successfully. The student and internship offer were mocked, and the application was saved using the repository without any errors.

- TC11: The test for the createAllEvaluations() method passed successfully. The method saved 11 evaluation items and sent the email without any errors.

- TC12: The test for addReport() passed successfully. When a second report was uploaded, it saved the report and triggered an email notification to the instructor.

- TC13: The test for the addResume() method passed successfully. The method retrieved the student, created a resume object, and saved it using the mocked repository. No exceptions were thrown, and the result matched the expected behavior.

- TC14: The test for the createPasswordResetToken() method passed successfully. The method generated a valid token and saved it to the repository with correct expiration time.

6.1.4 Test Log

All unit tests were run using IntelliJ IDEA. The tests were executed with JUnit 5 and Mockito. The console output and test result screen showed whether each test passed or failed. Screenshots for each test are given below.

- TC1: Tested the saveAnnouncement() method using a mocked repository. The service method was called and the console printed the correct ID and content. Test method name: testSaveAnnouncement_ReturnsSavedAnnouncement() (see Figure 47).

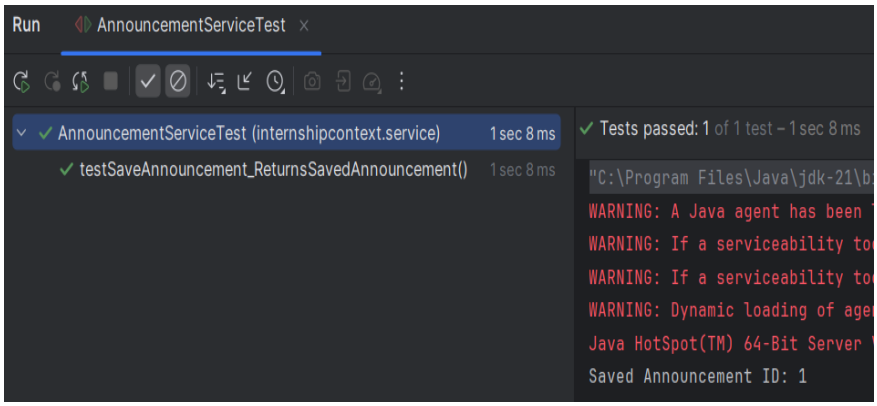


Figure 47: Unit test result for saveAnnouncement() method

The following test cases were executed, and their results are shown in Figure 48:

• TC2: Tested getApprovedTraineeInformationForms() with a mocked repository. Confirmed that a list of 2 items was returned.

• TC3: Called updateFormStatus() after mocking findById(). Checked that it returned true and correctly updated the form status.

• TC4: Called approveInsurance() on a mocked internship. Verified that the approval flag was set, the log was saved, and an email was sent to the user.

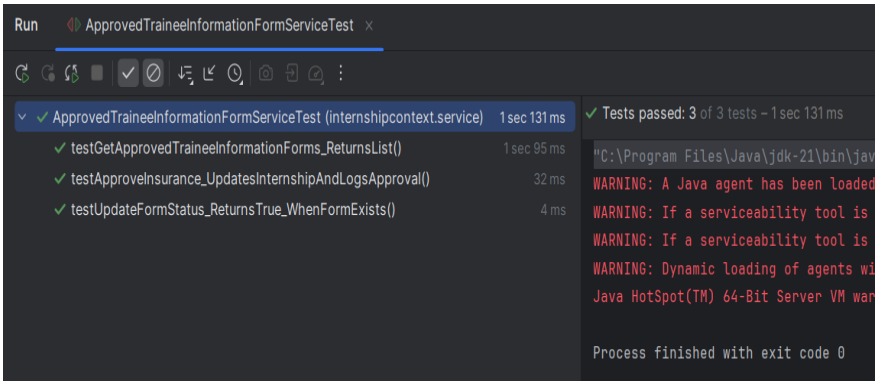


Figure 48: Test Results – ApprovedTraineeInformationFormService

- TC5: Tested sendVerificationEmails() method with two mock company branches. Verified that emailService.sendEmail() was called twice with correct subject and content (see Figure 49).

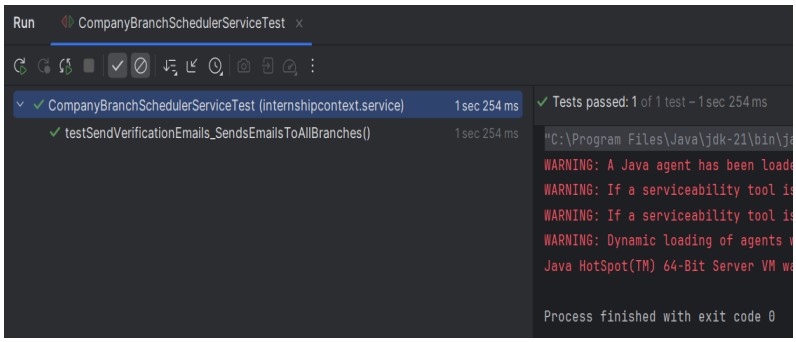


Figure 49: Test Result – CompanyBranchSchedulerService

- TC6: Tested the saveCompanyBranch() method. It saved a mock CompanyBranch and called the email sending function. Verified that the method worked as expected and did not throw an exception. Test method name: testSaveCompanyBranch_SavesAndSendsEmail() (see Figure 50).

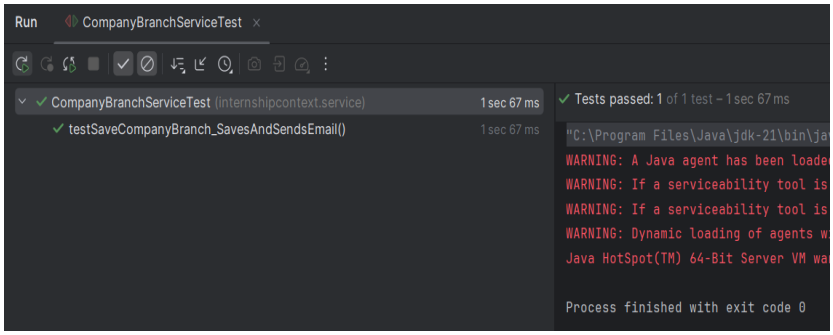


Figure 50: Result of saveCompanyBranch() unit test

- TC7: Tested the createEvaluationForm() method with a mocked trainee form and email service. The method saved the evaluation and triggered the email sending. Test method name: testCreateEvaluationForm_SavesEvaluationAndSendsEmail() (see Figure 51).

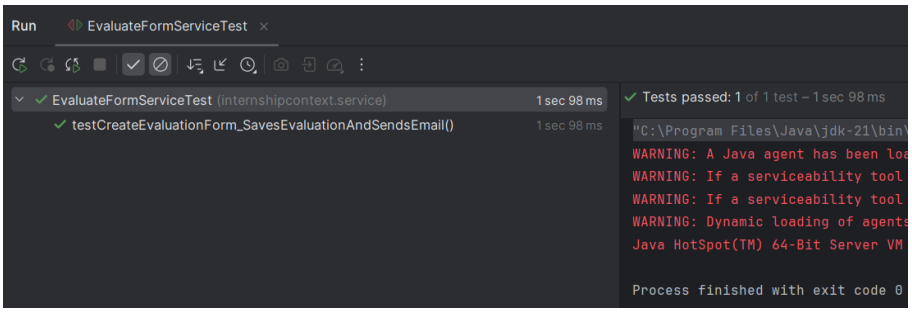


Figure 51: Unit test result for createEvaluationForm() method

- TC8: Tested the addForm() method using a mocked AcademicStaff and repository. Confirmed that the form was saved and the returned object had a valid ID. Test method name: testAddForm_SavesAndReturnsForm() (see Figure 52).

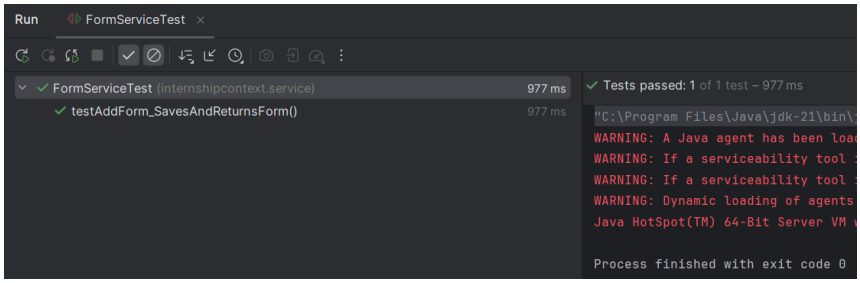


Figure 52: Unit test result for addForm() method

- TC9: Tested the updateInitialFormStatus() method with a mocked trainee form and company branch. Verified that the method updated the status, generated the token, and triggered the email service correctly. Test method name: testUpdateInitialFormStatus_SendsCompanyEmail() (see Figure 53).

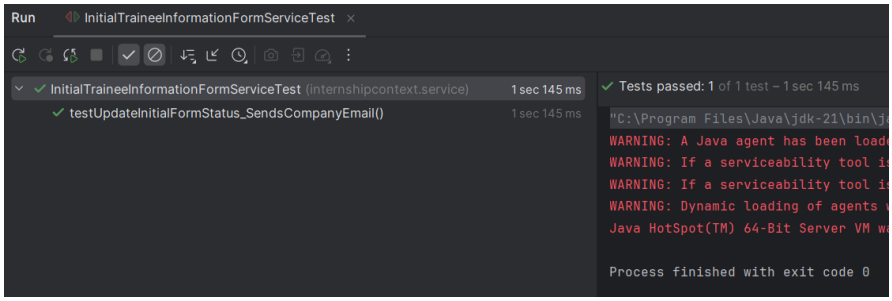


Figure 53: Unit test result for updateInitialFormStatus() method

- TC10: Tested applyForInternshipOffer() with mocked student and internship offer. Verified that the repository's save() method was called correctly. Test method name: testApplyForInternshipOffer_SavesApplication() (see Figure 54).

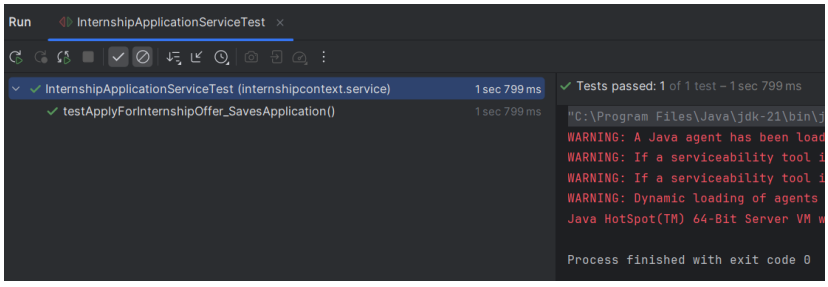


Figure 54: Unit test result for applyForInternshipOffer() method

- TC11: Tested the createAllEvaluations() method using a mocked report and evaluation repository. Verified that 11 evaluation entries were saved and the email service was called correctly. Test method name: testCreateAllEvaluations_SavesAndSendsEmail() (see Figure 55).

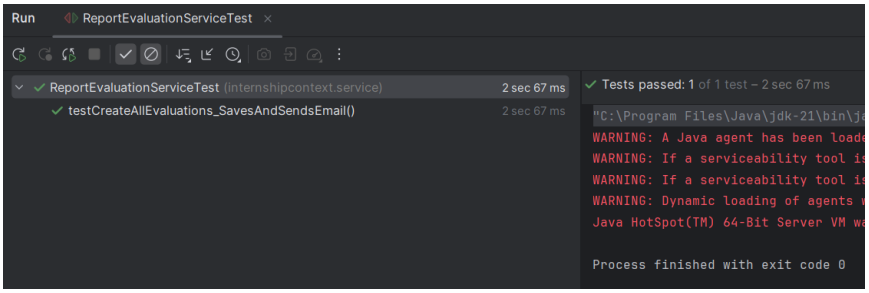


Figure 55: Unit test result for createAllEvaluations() method

- TC_{12}: Tested the addReport() method with mocked dependencies. Simulated a second report upload by a student. Verified that the report was saved and the instructor received a notification email. Test method name: testAddReport_SavesReportAndSendsEmailOnSecondUpload() (see Figure 56).

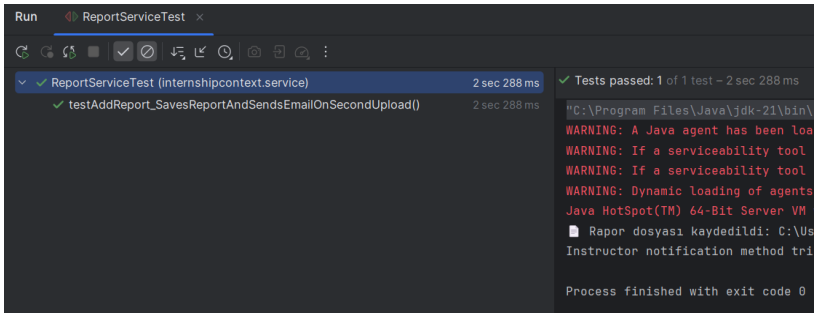


Figure 56: Unit test result for addReport() method

- TC13: Tested the addResume() method with a mocked student repository. Verified that the method fetched the student successfully, created the resume object, and saved it. Test method name: testAddResume_SuccessfullySavesResume() (see Figure 57).

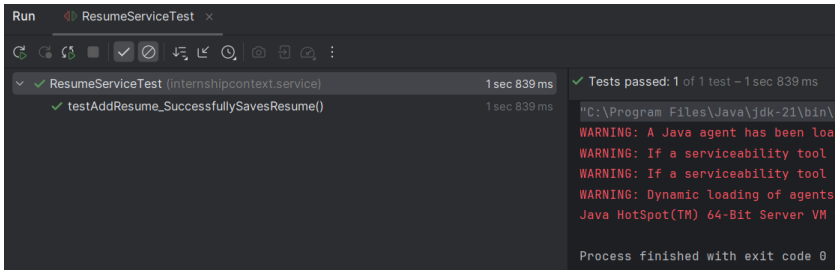


Figure 57: Unit test result for addResume() method

- TC14: Tested the createPasswordResetToken() method with a mock username. Verified that the generated token was saved using the mocked repository. Test method name: testCreatePasswordResetToken_SavesToken() (see Figure 58).

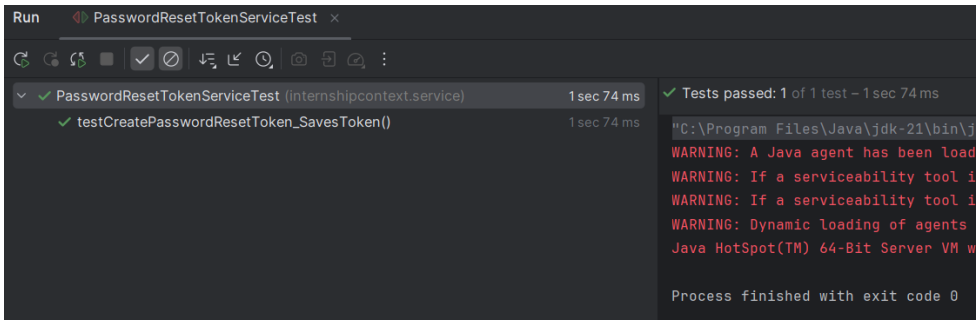


Figure 58: Unit test result for createPasswordResetToken() method

6.2 Integration Testing

6.2.1 Test Procedure

Unit and integration testing were conducted with a strong focus on ensuring the seamless interaction between frontend and backend components. The testing process primarily emphasized end-to-end (E2E) and integration scenarios to validate the system's overall behavior.

All tests were developed and executed within the IntelliJ IDE, using Playwright as the primary testing framework. Because we had tens of cases that needs integration between frontend and backend, we selected a total of 20 test cases were implemented in spec.ts files, covering key workflows and user interactions. These test cases simulated real user behavior and verified the correct functioning of various application modules in an integrated environment. Tests of more complicated and combined actions are performed within the system testing part.

Playwright provided robust capabilities for automating browser-based tests, enabling efficient validation of UI elements, API responses, and client-server communication. Each test case was carefully and specially crafted to ensure that both individual components and their interactions met the expected outcomes for each unique use case.

Additionally, during the testing process, it was also tested whether some functions that needed to send e-mails sent e-mail notifications.

Efekan performed all the tests in the integration testing.

6.2.2 Test cases

ID	Description	Steps	Test Data	Pre-condition	Expected Output	Passed/Failed
TC1	Test login integration: Successful login for different type of users	1) Navigates to localhost:4200 before test 2) Fills username/password fields and clicks login button for each test user 3) Frontend sends credentials to backend (localhost:8080/auth/verify) 4) Validates if it is successful login by checking if the redirect URL matches user type.	username: 'cidil', password: 'cidil2', username: 'e258546', password: 'efekan1', username: 'eever', password: 'eever1', username: 's_affairs', password: 'student_affairs1', username: 'Google-Branch', password: 'Google-Branch1',	Frontend and Backend are both should be running.	Url to change as http://localhost:4200${user.expectedRoute} Where expected route is different for every user for example: for student /student and for instructor /instructor etc.	Passed
TC2	Test login integration: Invalid credentials show error message	1) Navigates to localhost:4200 before test 2) Fills username/password fields and clicks login button for invalid credentials. 3) Frontend sends credentials to backend (localhost:8080/auth/verify) 4) Verifies error message displays	Username: 'invaliduser' Password: wrongpassword'	Frontend and Backend are both should be running.	Error message that includes 'You have entered wrong username or password' message	Passed
TC3	Test login integration: Empty form submission shows validation errors	1) Navigates to localhost:4200 before test 2) Frontend sends null credentials to backend (localhost:8080/auth/verify) 3) Verifies error message displays	Username: Password:	Frontend and Backend are both should be running.	Error message that includes 'You have entered wrong username or password' message	Passed
TC4	Test login integration: Company login link redirects and allows	1) Navigates to localhost:4200/company-login before test 2) Fills username/password fields and clicks login button	Username: 'Google-Branch', Password: 'Google-Branch1',	Frontend and Backend are both should be running.	Url to change as 'http://localhost:4200/company-branch/my-offers'	Passed

	company login	*3) Frontend sends credentials to backend (localhost:8080/auth/verify)* *4) Validates if it is successful login by checking if the redirect URL matches user type.*				
TC5	*Announcement loading*	*1) Navigates to localhost:4200/company-login before test* *2) Fills username/password fields and clicks login button* *3) Frontend sends get request to backend (localhost:8080/api/announcements)* *4) Validates by checking if the announcements were fetched and loaded correctly*	*None*	Frontend and Backend are both should be running.	*List of announcements*	*Passed*
TC6	*Announcement adding*	*1) Navigates to localhost:4200/login* *before test* *2) Fills username/password fields and clicks login button* *3) Frontend sends post to backend (localhost:8080/api/announcements)* *4) Validates by checking if the new announcement was added correctly.*	*Title:Test Announcement3 ' + Date.now()* *Content: 'This is a test announcement content'*	*Frontend and Backend are both should be running.*	*Announcement fetched and appeared in the list*	*Passed*
TC7	*Loading existing forms*	*1) Logs in as a coordinator or student* *2) Navigates to localhost:4200/coordinator/forms* *3) Sends get request to backend (localhost:8080/api/forms)*	*None*	*Frontend and Backend are both should be running.*	*Forms table is not empty and correctly filled after fetching data from the backend.*	*Passed*

		4) Verifies that forms table is correctly filled				
TC8	*Adding form*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/forms)* *3) Sends cv.pdf with a post request to backend* *(localhost:8080/api/forms)* *4) Verifies that number of forms in the table increased by one.*	*cv2.pdf*	*Frontend and Backend are both should be running.*	*Number of forms in the table increased by one.*	*Passed*
TC9	*Adding non-Pdf form*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/forms)* *3) Sends a non pdf input with post request to backend* *(localhost:8080/api/forms)* *4) Verifies that correct error message is displayed and the number of forms hasn't changed*	*testfile.txt*	*Frontend and Backend are both should be running.*	*"Only PDF files are allowed" message is displayed and the number of forms hasn't changed*	*Passed*
TC10	*Assigning an instructor to a student*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/assign-instructors)* *3) Verifies that forms loaded correctly from the backend* *4) For the first internship selects the second instructor in the menu and assigns them.* *5) Verifies the assignment.*	*None*	*Frontend and Backend are both should be running.*	*Success message is showed up and instructor assigned successfully is visible after an automatic table reload.*	*Passed*
TC11	*Update a deadline*	*1) Logs in as a coordinator*	*2024-12-31*	*Frontend and Backend*	*Report Deadline is updated as 2024-12-31*	*Passed*

		2) Navigates to (localhost:4200/coordinator/set-deadlines) *3) Sets the report deadline as 2024-12-31.* *4) Submits deadlines to backend (localhost:8080/api/deadline/add)* *5) Verifies the change in deadline.*		*are both should be running.*	*in deadlines page.*	
TC12	*Reset a deadline*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/set-deadlines)* *3) Removes the report deadline and send delete http request to backend (localhost:8080/api/deadline/delete/internship-deadline)* *5) Verifies the change in deadline.*	*None*	*Frontend and Backend are both should be running.*	*Report Deadline is changed as Not Set in deadline page.*	*Passed*
TC13	*Trainee form evaluation (coordinator):* *Load trainee forms*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/evaluate-forms)* *3) Fetches all the forms for coordinator. (localhost:8080/api/internships)* *4) Shows all forms in a table.*	*None*	*Frontend and Backend are both should be running.*	*Trainee Information forms table is not empty and correctly filled after fetching data from the backend.*	*Passed*
TC14	*Trainee form evaluation (coordinator):* *Approve trainee forms*	*1) Logs in as a coordinator* *2) Navigates to (localhost:4200/coordinator/evaluate-forms)* *3) Fetches all the forms for coordinator. (localhost:8080/api/internships)*	*None*	*Frontend and Backend are both should be running.*	*Success notification has popped up and form's coordinator approval changed as approved.*	*Passed*

		4) Finds a non-approved form in the table and approve it. *5) Sends the id of the form we want to approve to backend with a post http request. (localhost:8080/api/coordinatorApprove/approve/${id}?username=${encodeURIComponent(username)})* *6) Verifies the success notification has popped up and form's coordinator approval changed as approved in frontend.*				
TC15	*Instructor* *View: browsing reports*	*1) Logs in as an instructor* *2) Navigates to (localhost:4200/instructor/evaluate-assigned-reports)* *3) Fetches all the forms for that instructor.* *4) Checks the reports of the 5th report in the table by fetching it from the backend. (localhost:8080/reports/trainee/ + formId+ /reports)* *5) Verifies that the report information is fetched and displayed correctly.*	*5th form*	*Frontend and Backend are both should be running.*	*Report information is fetched and displayed correctly.*	*Passed*
TC16	*Instructor* *View: Viewing Company Evaluation*	*1) Logs in as an instructor* *2) Navigates to (localhost:4200/instructor/evaluate-assigned-reports)* *3) Fetches all the forms for that instructor.* *4) Finds a form that has its company evaluation filled.* *5) Verifies that the company evaluation is loaded and being displayed.*	*None*	*Frontend and Backend are both should be running.*	*Company evaluation is loaded and being displayed.*	*Passed*

TC17	*Student Affairs Insurance Approval*	*1) Logs in as student affairs* *2) Navigates to (localhost:4200/student-affairs/approved-internships)* *3) Fetches all the forms for student affairs.* *4) Finds a form that hasn't approved.* *5) Approves the form by sending post http request to backend. (localhost:8080/api/studentAffairs/approveInsurance)* *6) Verifies that it is displayed as approved and it persists after reloading the page and fetching data again from the backend.*	*None*	*Frontend and Backend are both should be running.*	*Approved text is displayed, and it persisted after reloading the page.*	*Failed*
TC18	*Uploading Resume*	*1) Logs in as a student* *2) Navigates to (localhost:4200/student /my-resume)* *3) Clicks to browse and upload cv2.pdf.* *4) Confirms upload and reload the page to verifies that resume is accessible from the backend correctly.*	*cv2.pdf*	*Frontend and Backend are both should be running.*	*Uploaded resume is displayed, and it persisted after reloading the page.*	*Passed*
TC19	*Approving Internship Application*	*1) Logs in as a company-branch* *2) Navigates to (localhost:4200/company-branch/evaluate-intern-student)* *3) Approves a non-approved form by sending a post request to backend. (localhost:8080/api/traineeFo*	*None*	*Frontend and Backend are both should be running.*	*"Internship successfully approved!" message is shown and status changed to Approved.*	*Passed*

		rmCompany/approveInternship) *4) Confirms that "Internship successfully approved!" message is shown and status changed to Approved.*				
TC20	*Assigning Supervisor*	*1) Logs in as a company-branch* *2) Navigates to (localhost:4200/company-branch/evaluate-intern-student)* *3) Clicks on Set Instructor button for 2nd form in the list.* *4) Writes Ka Ryo Ten to menu and submits it to backend.* *(localhost:8080/api/internships/updateSupervisor)* *5) After window is being closed, verifies that supervisor is changed as Ka Ryo Ten in the table.*	*Supervisor Name:* *"Ka Ryo"* *Supervisor Surname: "Ten"*	*Frontend and Backend are both should be running.*	*Supervisor is changed as Ka Ryo Ten in the table.*	*Passed*

6.2.3 Test Results

Since we usually test the integration of each feature we add to our system, only one of our test scenarios failed.

- TC1: All test users logged in successfully, each being redirected to the correct page based on their user role.
- TC2: The system correctly handled login failure and displayed the appropriate error message.
- TC3: The system displayed successfully validation errors when trying to submit an empty form.
- TC4: Successfully redirected to the correct company login dashboard after successful login as a company branch.
- TC5: Fetched announcements successfully, confirming GET request functionality.
- TC6: Added a new announcement and confirmed its appearance in the announcement list. Additionally confirmed that email notification is being sent to selected user types.
- TC7: Verified forms were correctly fetched and displayed for coordinator/student.
- TC8: PDF file successfully uploaded, and form count incremented, verifying POST functionality.
- TC9: System correctly prevented non-pdf form file upload and showed the appropriate error message.

- TC10: Instructor assignment was successful and reflected immediately in the UI with a success message.
- TC11: Deadline was set to 2024-12-31 and is being displayed correctly verifying POST functionality.
- TC12: Deadline was successfully removed and shown as "Not Set". This proved that our DELETE request functions as intended.
- TC13: Trainee Information Forms for coordinator were loaded and populated in the table without an issue.
- TC14: As a coordinator system successfully approved a form and updated the status of the form. The email notification sent to the student was also approved successfully.
- TC15: Instructor accessed details of reports uploaded by students correctly.
- TC16: Instructors can view the company evaluations of students successfully.
- TC17: The test case failed as the insurance status was not changed to "approved". However, after a page reload the status changes correctly. Although we don't know the issue yet it is possible that this indicates a possible issue in backend where backend returns an error message to frontend even after successfully changing the status. It was also noted that during this process, an email notification was successfully sent to the student stating that their insurance had been approved.
- TC18: Resume was uploaded and displayed correctly. The preservation of the resume proves that the POST request works correctly.
- TC19: Approval of internship by company was successful, status updated, and success message appeared.
- TC20: Supervisor details updated correctly and reflected in UI which shows a correctly working POST request functionality.

6.2.4 Test Log

All integration and E2E tests were run using IntelliJ IDEA. The tests were automatically executed with Playwright. The console output and test result screen showed whether each test passed or failed. Screenshots for each test are given below.

TC1-TC4:

Tested login integration by logging in as different type of users and trying to login with invalid credentials. The system gave the expected output in all cases. To check the tests performed, please check the figure below.

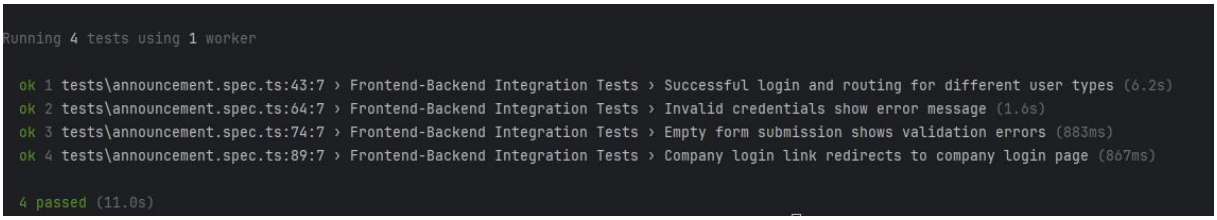


Figure 59: Login test

TC5-TC6:

Tested announcement integration with 3 cases. The first test loads the existing announcements correctly from the backend. Second case verifies that the frontend shows the announcement form when add button is clicked and third case confirms that the form that is shown actually sends http request that adds a new announcement. The system gave the expected output in all cases. You can check the figure below to see test results.

```
Running 3 tests using 1 worker

  ok 1 tests\announcement.spec.ts:19:7 › Announcements Integration Tests › Should load existing announcements (4.4s)
  ok 2 tests\announcement.spec.ts:31:7 › Announcements Integration Tests › Should show/hide add announcement form (2.3s)
  ok 3 tests\announcement.spec.ts:44:7 › Announcements Integration Tests › Should create new announcement (2.4s)

  3 passed (10.3s)
```

Figure 60: Announcement integration test

TC7-TC9:

Tested form integration with 4 cases. The first three cases check TC7, TC8 and TC9 and the last case checks if the page reloads after uploading. It is not considered a case because it is not directly related to integration. As you can see the figure below the system gave the expected output in all cases.

```
Running 4 tests using 1 worker

  ok 1 tests\files.spec.ts:24:7 › Forms Integration Tests › Should load existing forms from API (3.5s)
  ok 2 tests\files.spec.ts:42:7 › Forms Integration Tests › Should successfully upload a PDF form (12.4s)
  ok 3 tests\files.spec.ts:64:7 › Forms Integration Tests › Should show error when uploading non-PDF file (2.4s)
  ok 4 tests\files.spec.ts:88:7 › Forms Integration Tests › Should refresh forms list after upload (2.7s)

  4 passed (22.3s)
```

Figure 61: Login test

TC10:

Tested instructor assignment integration. Verified that instructor assigned as selected and success message showed up. Check the figure below.

```
PS C:\Users\uysal\Desktop\CN6491\Project\23-05-2025\GraduationProject\internship-management> npx playwright test

Running 1 test using 1 worker

  ok 1 tests\assign.spec.ts:22:7 › Manual Instructor Assignment › Should manually assign instructor to trainee form (15.4s)

  1 passed (16.7s)
```

Figure 62: Instructor Assignment test

TC11-TC12:

Tested deadline setting integration. Verified that deadlines can be set and reset correctly as selected in the frontend. You can check the figure below to see the test performed.

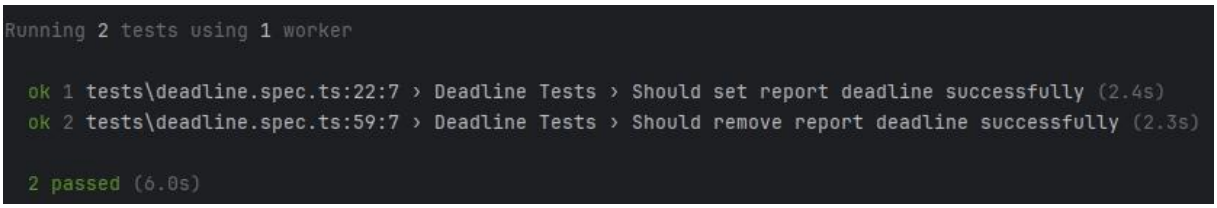


Figure 63: Deadline test

T13-T14:

Tested that the coordinator can view and approve the internship forms of the students in the frontend and that this works in an integrated way with the backend. You can see the figure below to see the console screen of the tests.

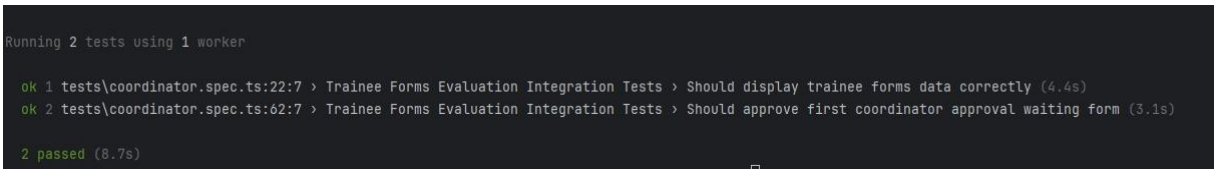


Figure 64: Coordinator Form Approval test

T15-T16:

Tested that instructors can view reports of the students and company evaluation of their internship in the frontend. You can see the tests performed below.

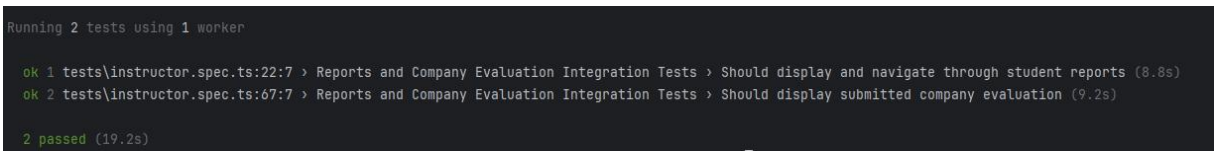


Figure 65: Instructor Report View test

T17:

Tested student affairs to approve the insurance of an internship and further checked if the change persists after reloading the page. After running the tests shown below, the approvement is being displayed in the frontend only after reloading the page.

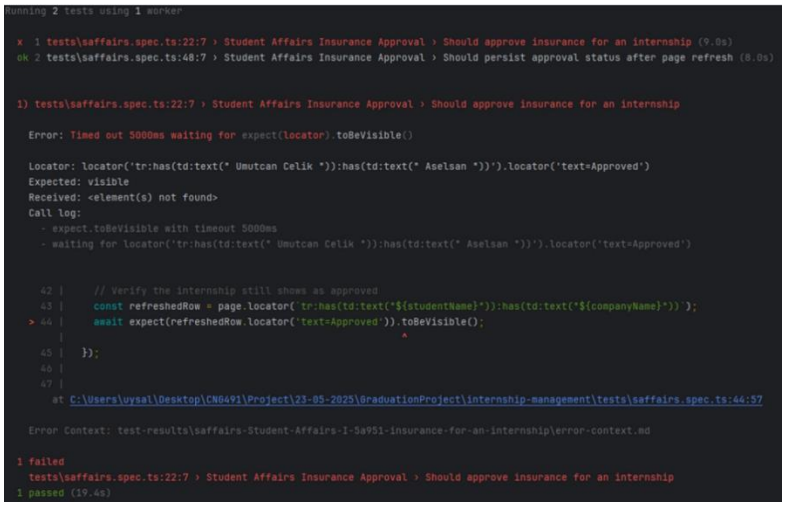


Figure 66: Student Affairs Insurance Approval test

T18:

Tested student resume upload to the system and confirmed that the uploaded resume can be fetched again from the backend after reloading the page. Below is the test we performed.

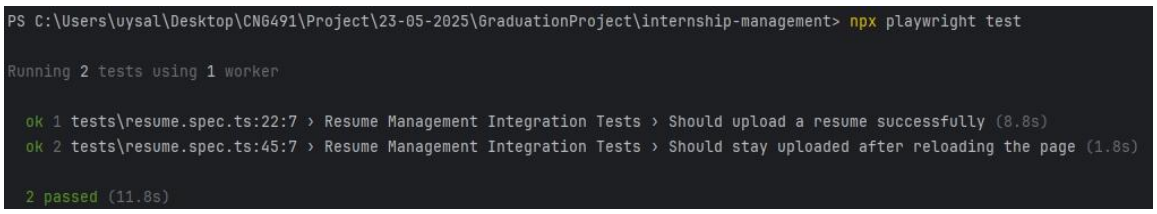


Figure 67: Resume Upload test

T19-T20:

Tested company branch to approve student internships and assigning supervisors to these internships. Verified that approvements and supervisor assignments are done correctly. You can examine the following figure to see the tests.

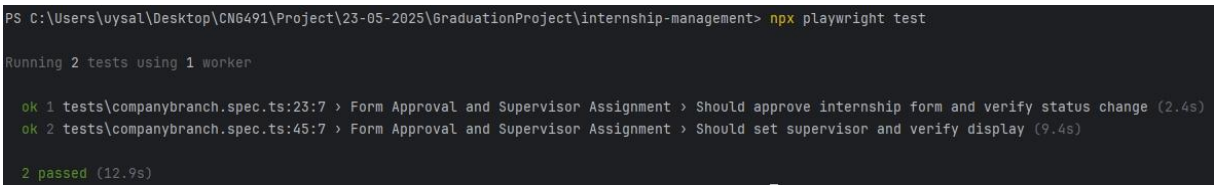


Figure 68: Company Internship Approval test

6.3 System Testing

6.3.1 Test Procedure

System testing was performed in real-time using a scenario-based approach to validate the end-to-end workflow of the system. The primary objective was to ensure that the integrated components of the system function correctly together and meet the functional requirements. The test procedure followed a structured approach based on the system's actual usage scenarios that simulates real-world usages and ensuring comprehensive coverage of all critical functionalities. All system tests were performed by Efekan.

6.3.2 Test Scenarios

Scenario 1:

a) As the new internship period approaches, the internship coordinator determines the deadlines for filling out the internship application form and submitting the report.

b) A student who finds an internship enters the trainee information form into the system.

c) The student can make changes on this form before it is approved.

d) The internship application entered by the student into the system is approved by the internship coordinator if deemed appropriate. The student gets an email notification about this.

e) The internship application approved by the coordinator is now visible to the company. The company approves the student's internship from the system. The student is notified by an e-mail.

f) The company assigns a supervisor to the student.

g) Student affairs approves the insurance. And can export to excel. The student is notified by an e-mail.

h) The student uploads the internship report to the system after the internship is completed.

i) The company evaluates the student's internship. The student is notified by an e-mail.

j) The coordinator assigns a faculty member to evaluate the student.

k) The assigned instructor sees the reports uploaded by the student and the evaluation form uploaded by the company through the system.

l) Instructor grades the last uploaded report and requests re-submission. The student is notified by an e-mail.

m) The student submits the report again.

n) Instructor approves by giving grade again. The student is notified by an e-mail.

o) The student and the coordinator can see this grade.

Scenario 2:

a) The student uploads his/her resume to the system.

b) Student uses browse Internships page to see the companies where previous students have done internship. They can also see which internships match with their skills in their resume. Student applies to a company here.

c) The company can see and approve this application and the student's resumé on the Applicants page.

Scenario 3:

a) The student uploads his/her resume to the system.

b) The company creates an internship advertisement on the open internships page.

c) Student uses Internships Offers page to applies to an internship advertisement.

d) The company can see the applicants from the open internships page and reject the student.

Scenario 4:

a) Coordinator adds an announcement. All selected user types get notified by an e-mail.

b) Students can view the announcement from Announcements page.

Scenario 5:

a) Coordinator adds a summer training form.

b) Students can view and download the summer training form.

6.3.3 Test Results

Scenario 1: Passed

Scenario 1 is a scenario that fictionalizes the main purpose of the internship system, and the current system was able to successfully fulfil all steps. The internship coordinator was able to set the application and report deadlines, and students could enter and edit their trainee information. Approved internship applications triggered email notifications to students and became visible to companies, who could then approve them and assign supervisors. Student affairs successfully processed and approved insurance and exported records to Excel, with appropriate notifications sent to students. After completing the internship, students uploaded their reports, and companies submitted evaluations, both accessible to the assigned instructors. Instructors could review the documents, assign grades, and request revisions with students notified via email. Final grades were submitted by instructors and became visible to both the student and the coordinator.

However, during testing, we noticed a small usability issue: students were unable to view the revision decision directly within the system interface. Although this doesn't affect the functionality and even though the students were notified by an email, the lack of visibility in the system can create confusion. This issue will be addressed to ensure a better user experience.

Scenario 2: Passed

The student was able to upload their resume to the system without issues. Upon navigating to the "Browse Internships" page, the student could view a list of companies where previous students had interned, along with personalized internship suggestions that matched the skills listed in their resume. The student was able to apply to a selected company directly through the system. On the company side, the application appeared correctly on the "Applicants" page, and the company was able to view and approve the student's application and resume. All functionalities worked as expected.

Scenario 3: Passed

As we already tested in second scenario the system successfully allows the student to upload their resume. The company is able to create an internship advertisement, which is then displayed on the open internships page, indicating that advertisement creation feature is functional. The student is able to view and apply to the internship via the Internships Offers page, demonstrating that application is correctly implemented. Finally, the company can view the list of applicants and reject a student through the open internships page, confirming that rejection functionality operates as expected.

Scenario 4: Passed

When the coordinator adds an announcement, all selected user types receive a notification via email. Additionally, students can access and view the newly added announcement correctly on the Announcements page which confirms that both the notification and display functionalities are working as intended.

Scenario 5: Passed

After the coordinator adds a summer training form, students can access and download the newly added summer training form correctly on the Forms page. This shows that all functionalities are working as expected.

6.3.4 Test Log

Scenario 1:

a) Coordinator sets the deadlines for filling out the internship application form and submitting the report.

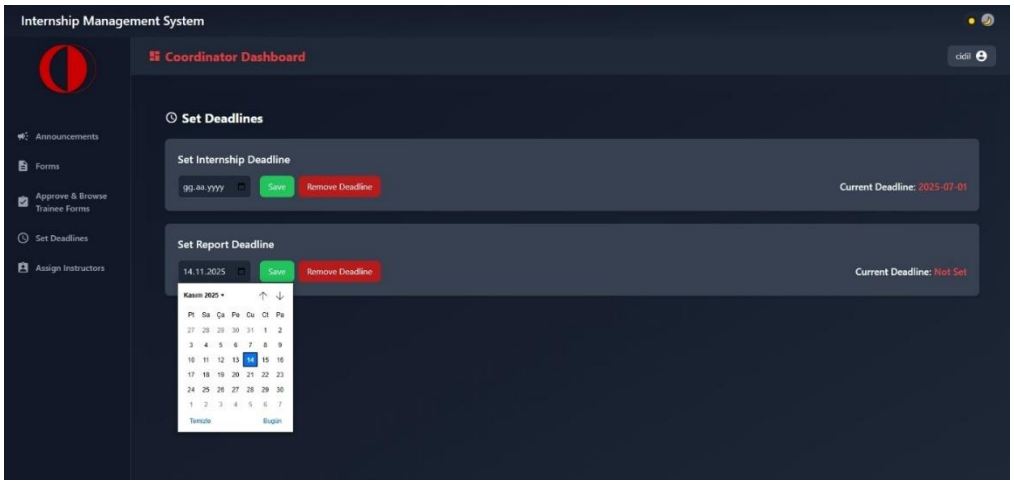


Figure 69: Coordinator Set Deadline page

b) A student who finds an internship enters the trainee information form into the system.

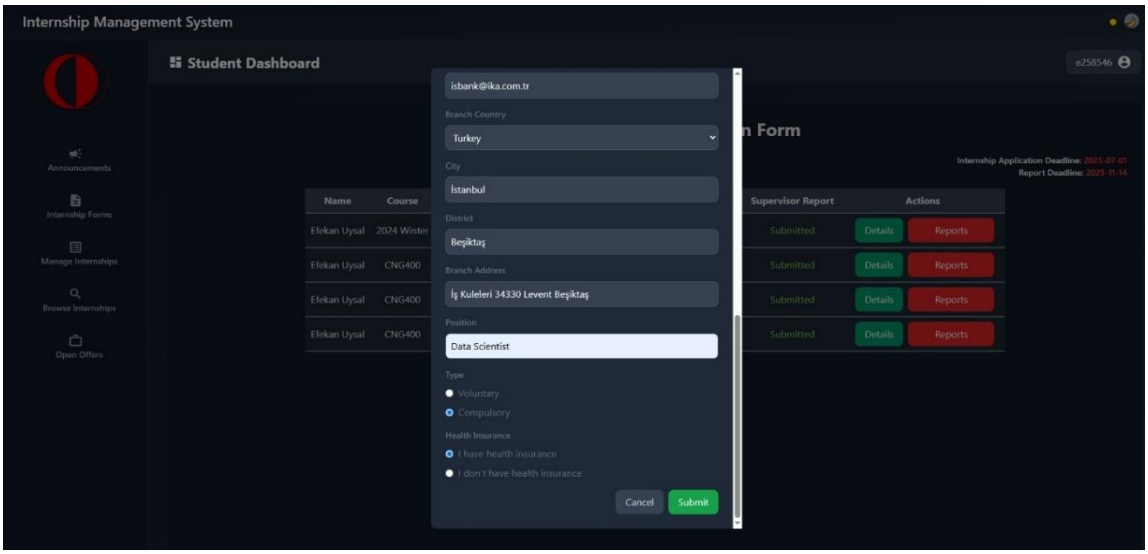


Figure 70: Student's Trainee Form page

c) The student can make changes on this form before it is approved by the company.

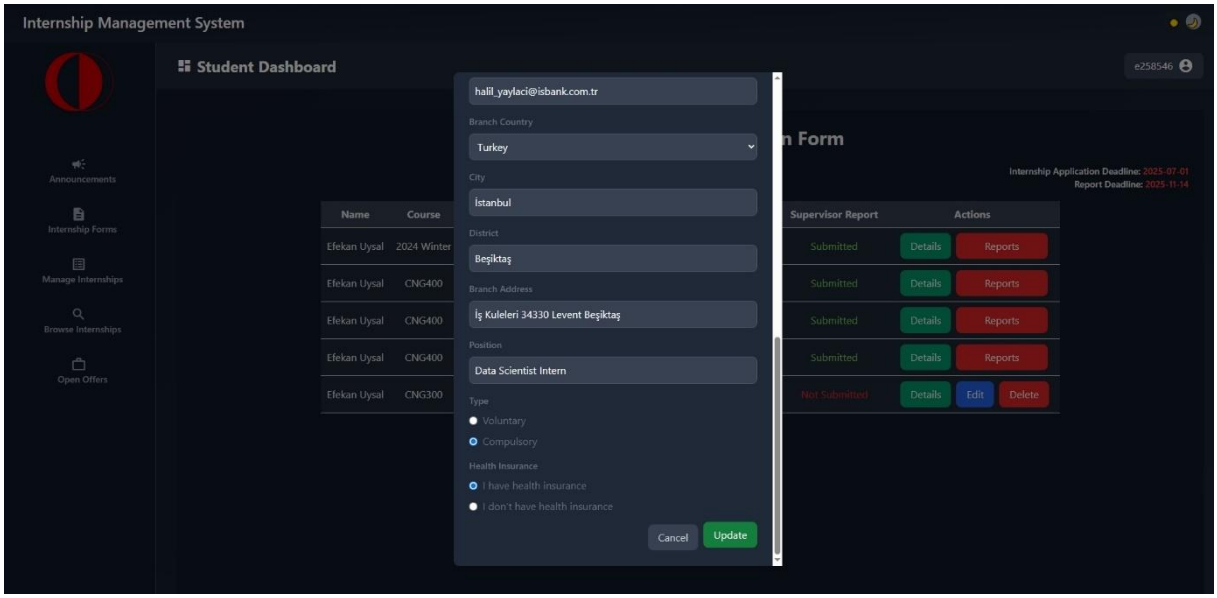


Figure 71: Student Form edit

d) The internship application is approved by the internship coordinator. Students get an email notification.

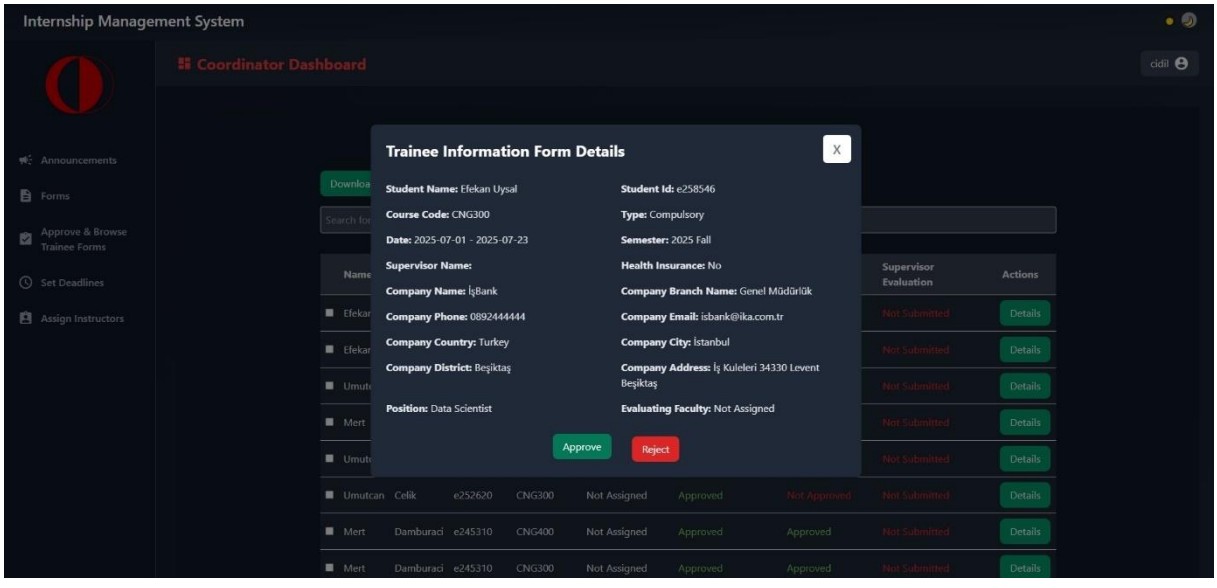


Figure 72: Coordinator Form Approval page

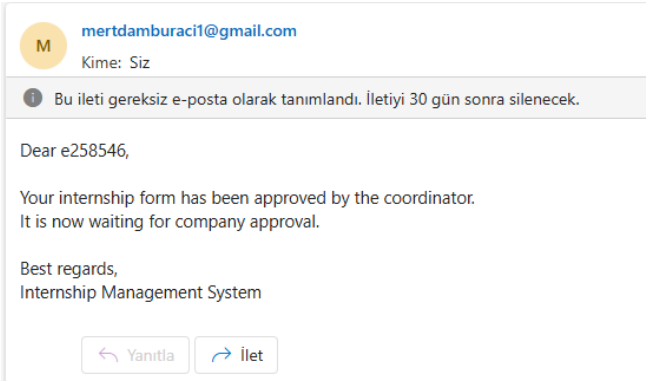


Figure 73: Form Approval Mail

e) The internship application approved by the coordinator is now visible to the company. The company approves the student's internship from the system. The student gets an email notification.

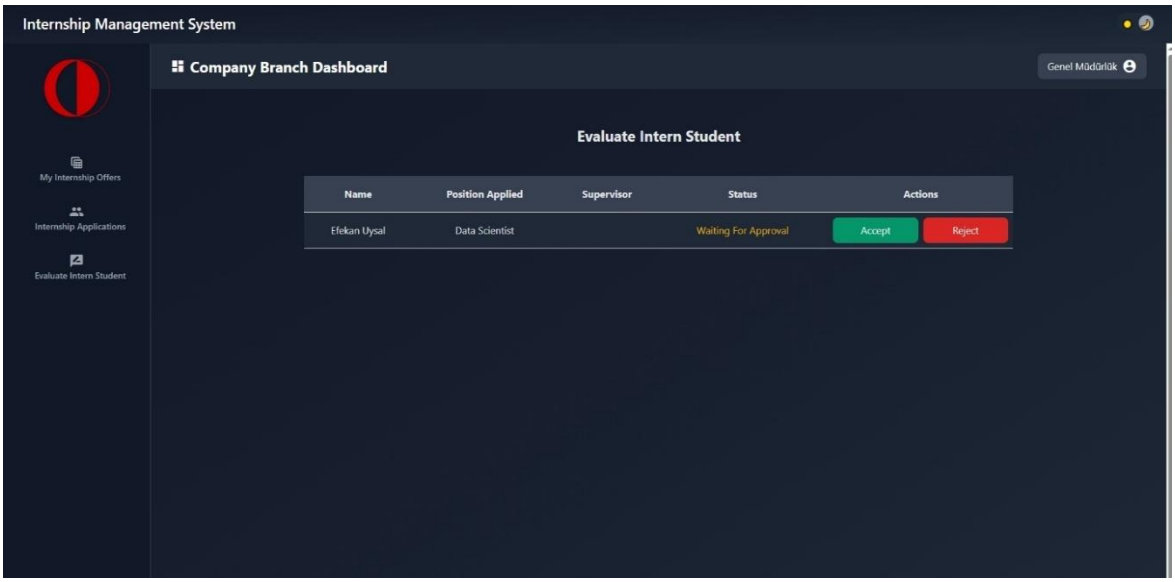


Figure 73: Company Internship Approval page

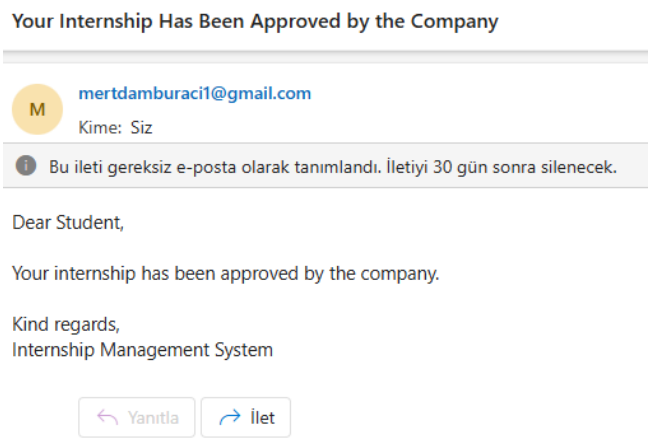


Figure 74: Company Approval Mail to Student

f) The company assigns a supervisor to the student.

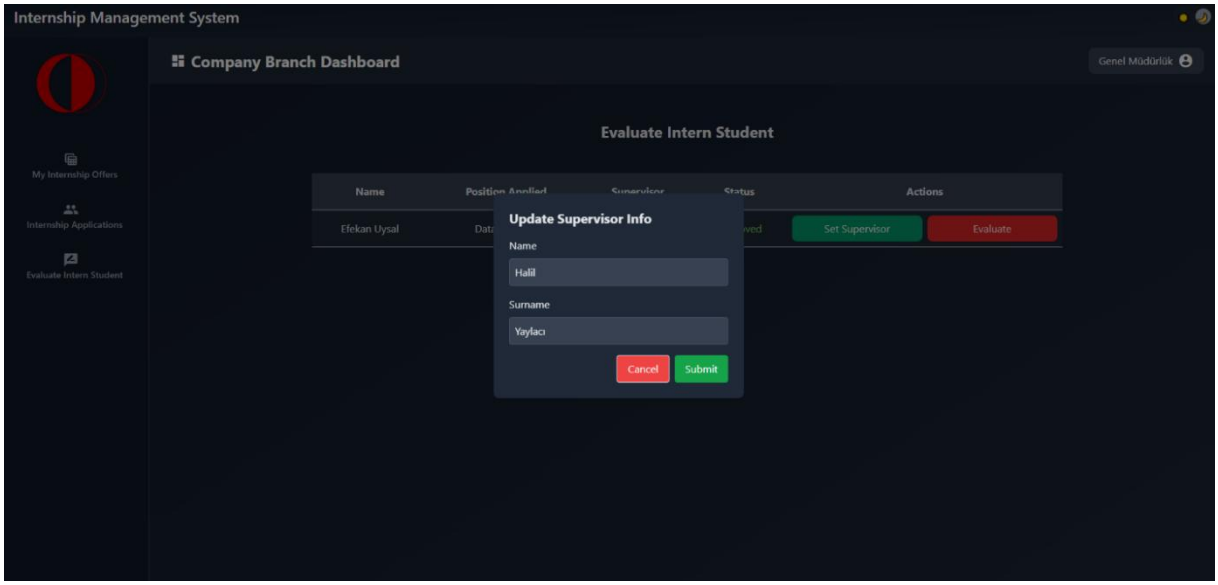


Figure 75: Company Supervisor Assign page

g) Student affairs approves the insurance. And can export to excel. The student will get an email notification.

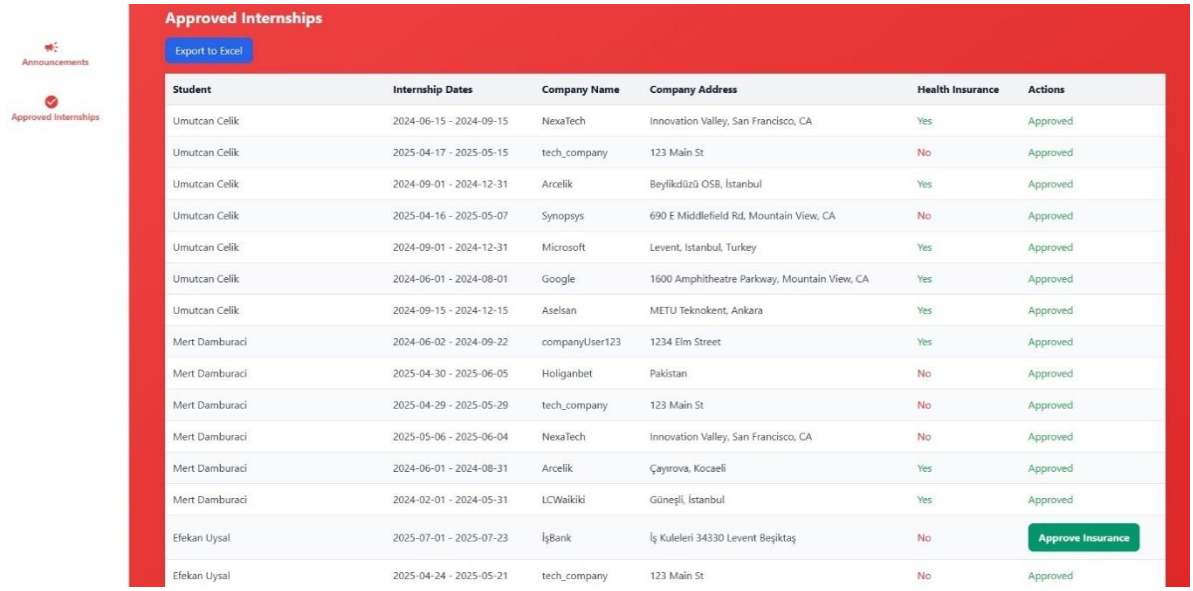


Figure 76: Student Affairs Insurance Approval page

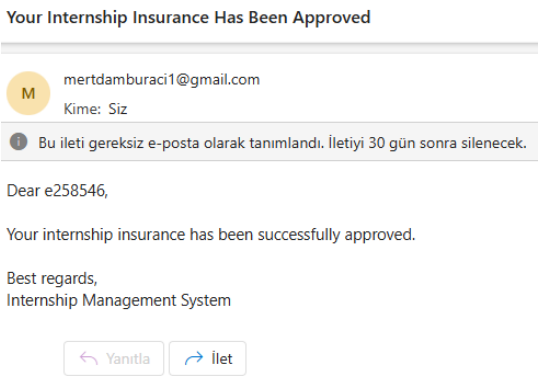


Figure 77: Insurance Approval Mail to Student

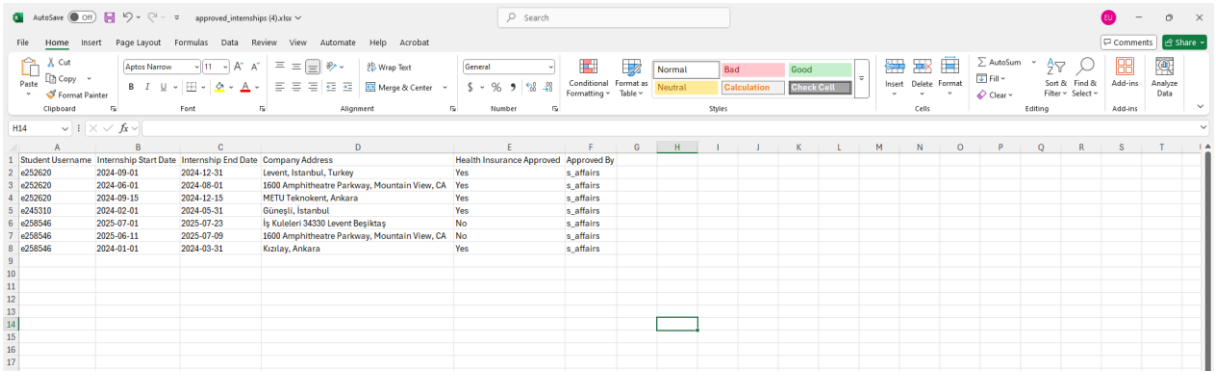


Figure 78: Student Affairs Excel Download example

h) The student uploads the internship report to the system after the internship is completed.

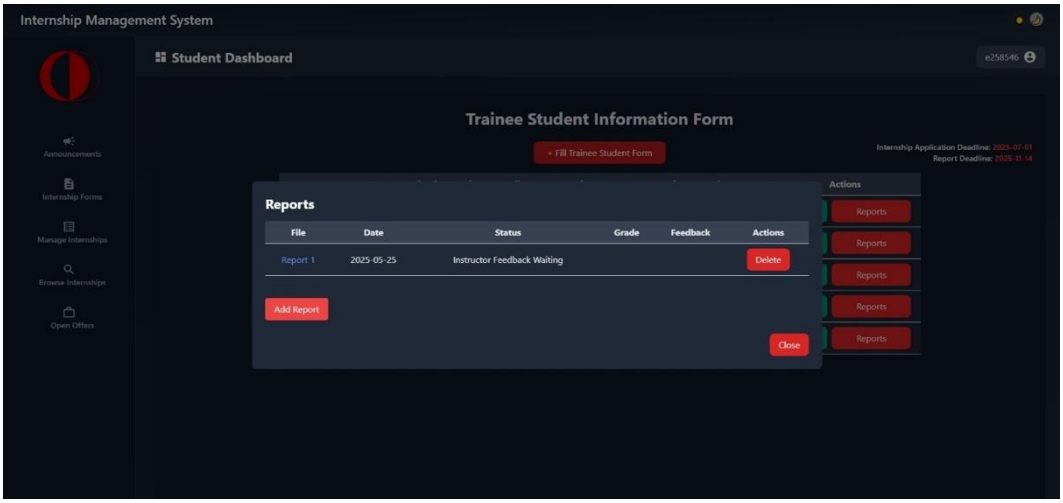


Figure 79: Student Report Upload page

i) The company evaluates the student's internship. The student gets an email notification.

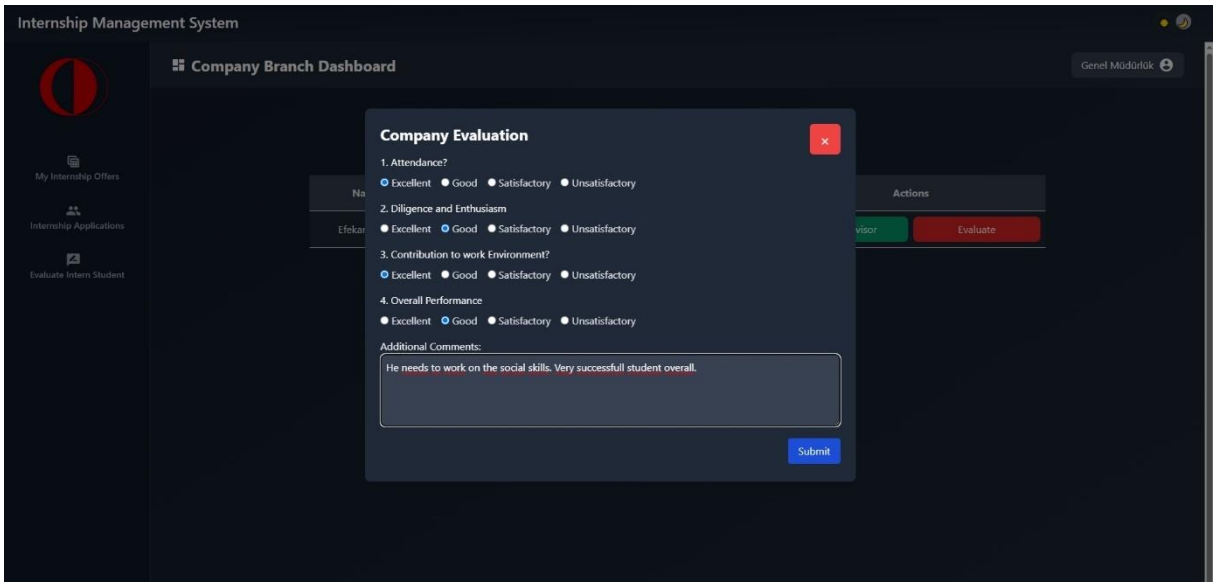


Figure 80: Company's Student Evaluation page

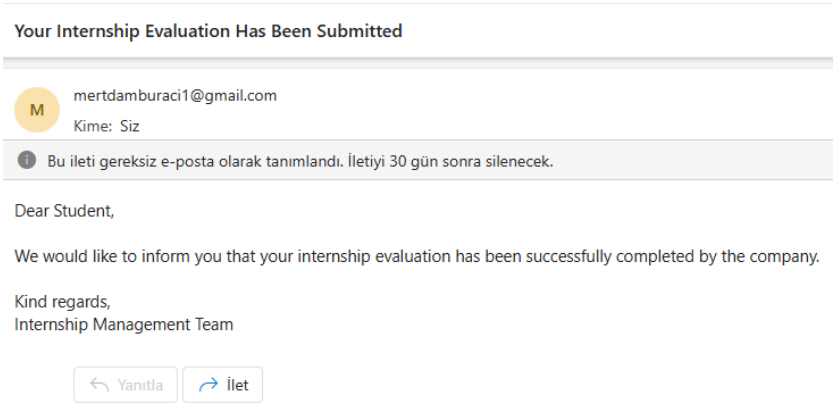


Figure 81: Company Evaluation Submit Mail to Student

j) The coordinator assigns a faculty member to evaluate the student.

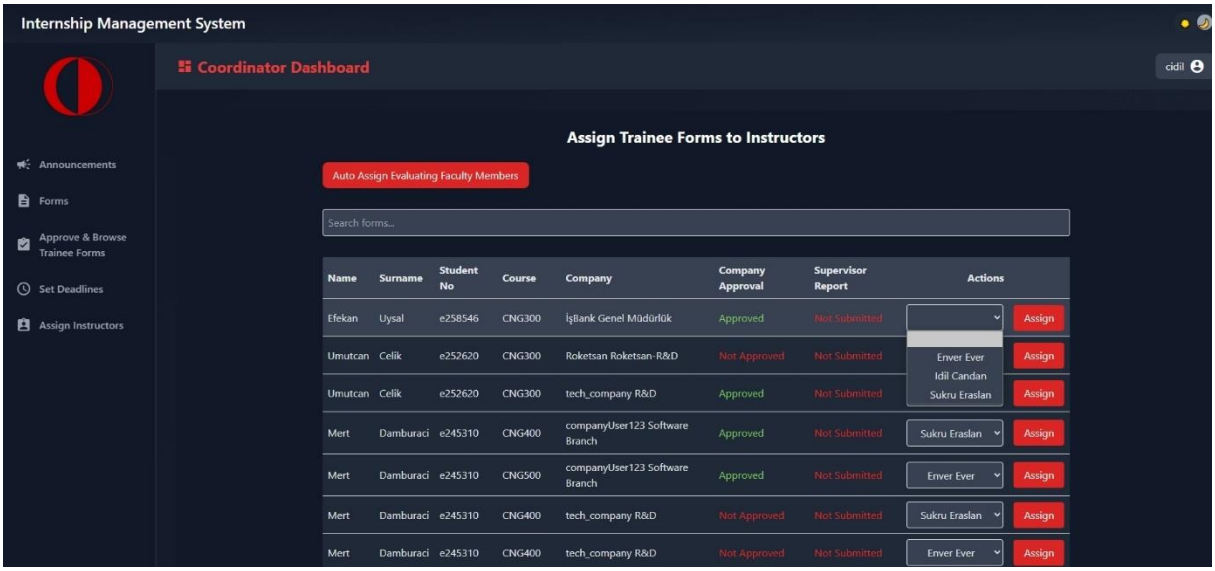


Figure 82: Coordinator's Assign Instructor page

k) The assigned instructor sees the reports uploaded by the student and the evaluation form uploaded by the company through the system.

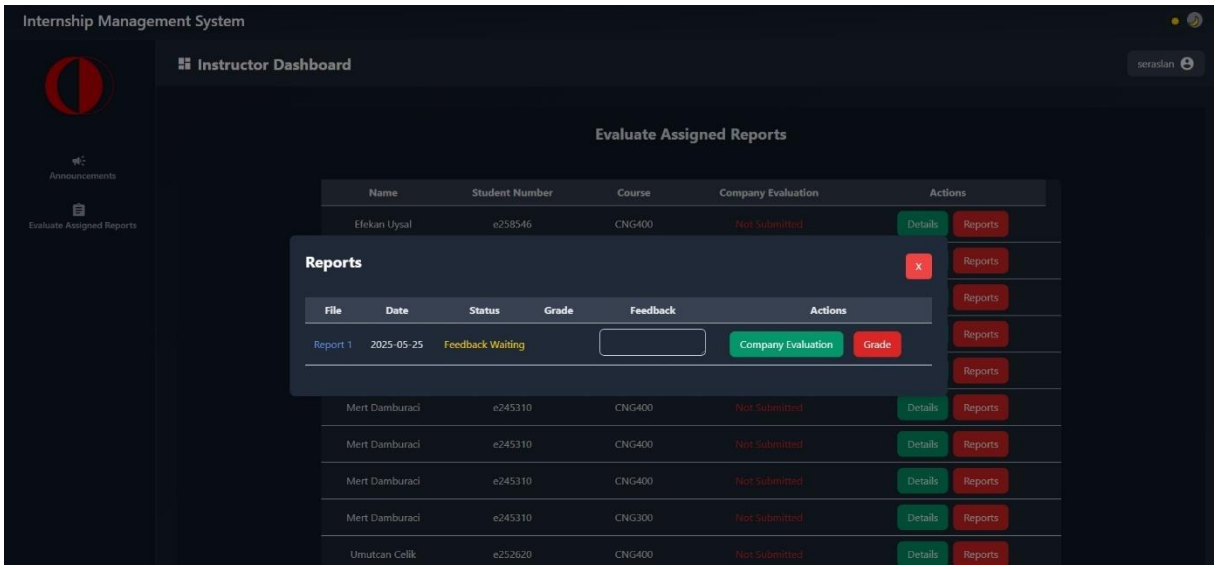


Figure 83: Instructor's Assigned forms page

l) Instructor grades the last uploaded report and requests re-submission. The student gets and email notification.

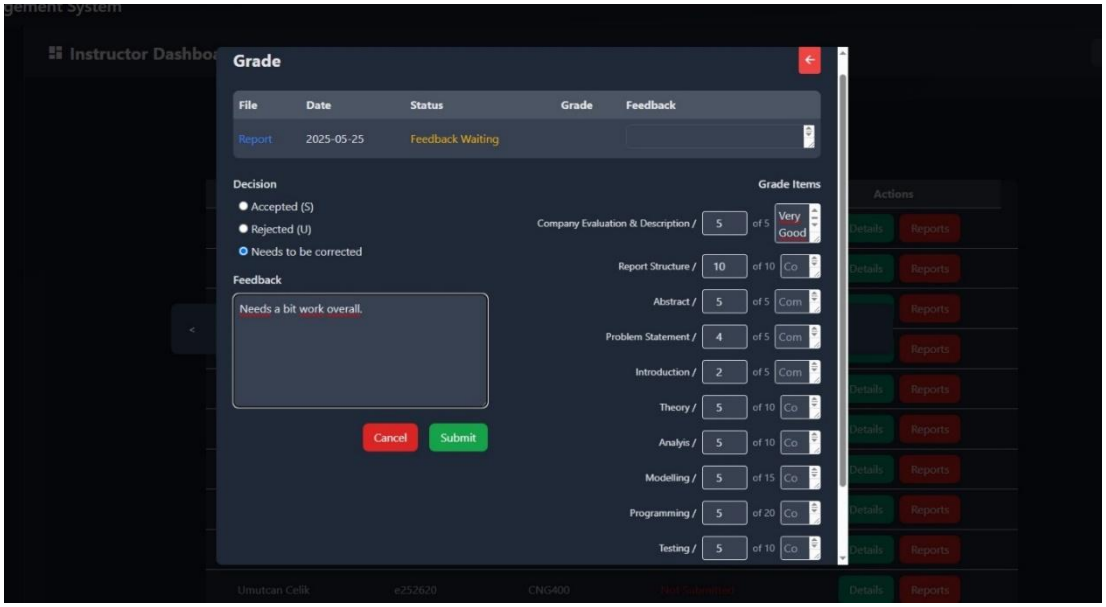


Figure 84: Instructor's Grading page

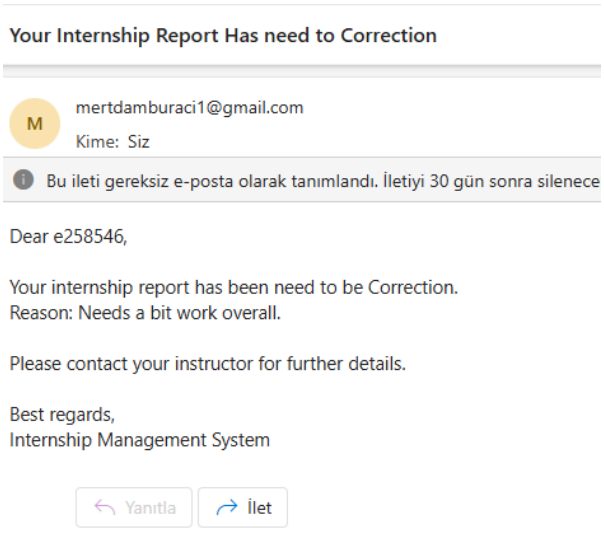


Figure 85: Student's Report Grading Results Mail

m) The student submits the report again.

n) Instructor approves by grading again. The student is notified by e-mail.

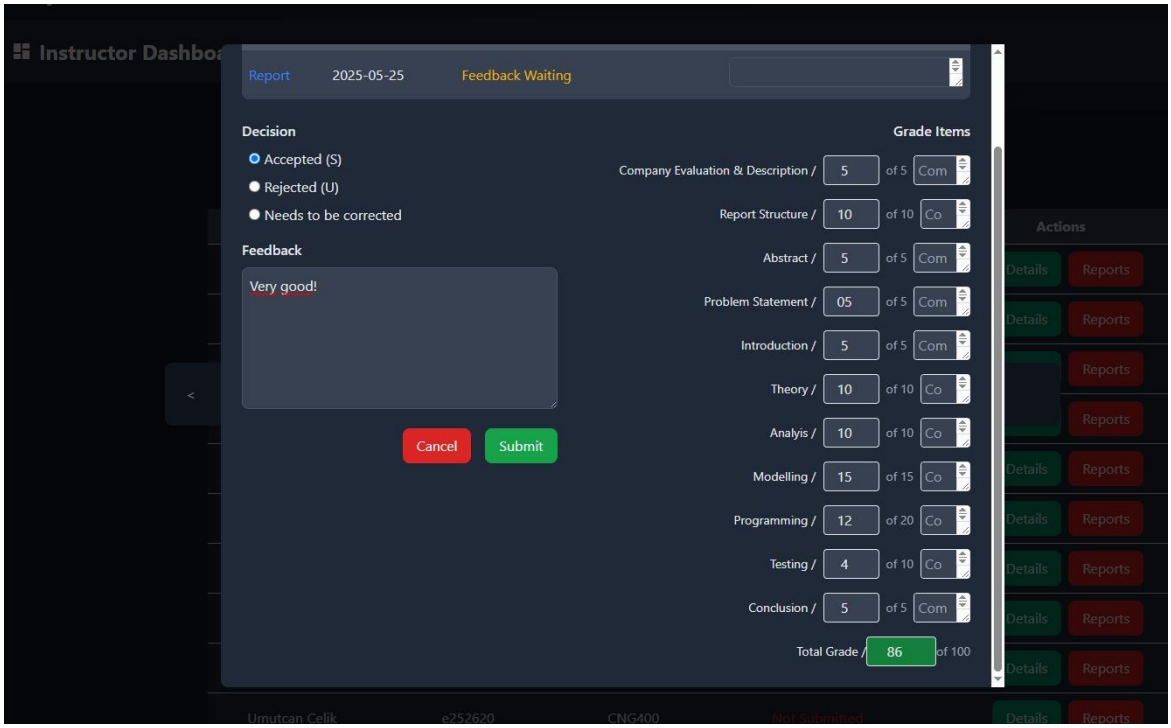


Figure 86: Instructor Re-Grade Process

o) The student and the coordinator can see this grade.

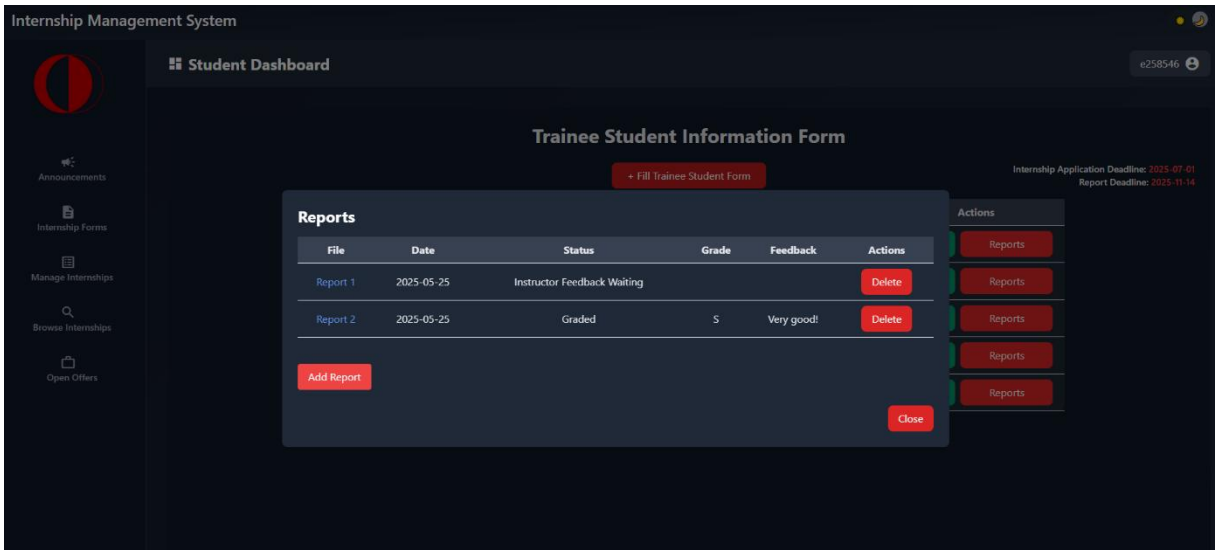


Figure 87: Student Grade and Feedback result

Scenario 2:

a) The student uploads his/her resume to the system.

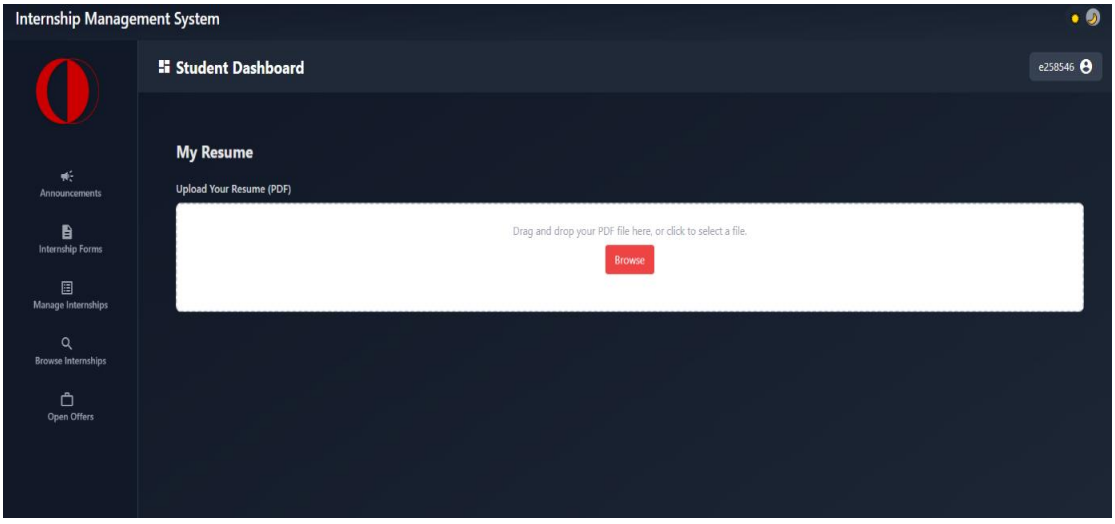


Figure 88: Student's Resume Upload page

b) Students use browsing Internships pages to see the companies where previous students have done internship. They can also see which internships match their skills in their resume. Students apply to a company here. The company will be notified by email.

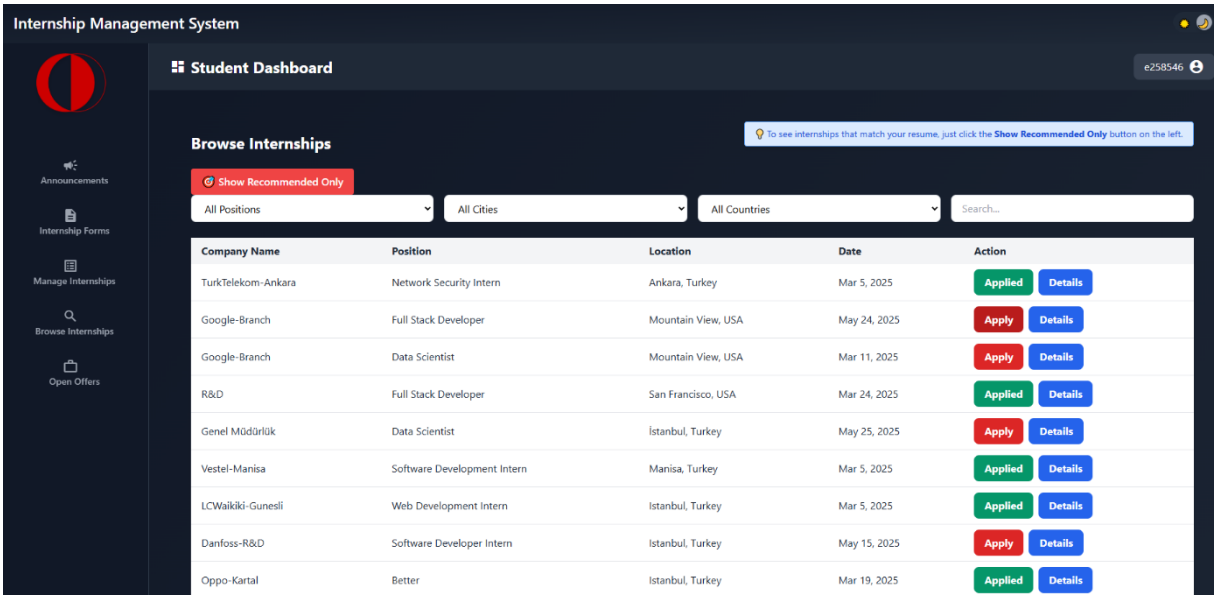


Figure 89: Student's Browse Internship page

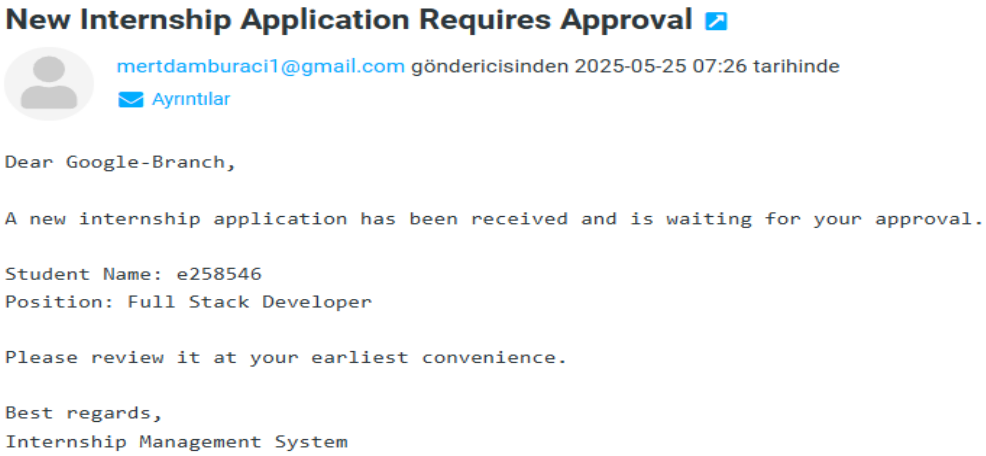


Figure 90: Company Internship Application Mail

c) The company can see and approve this application and the student's resumé on the Applicants page.

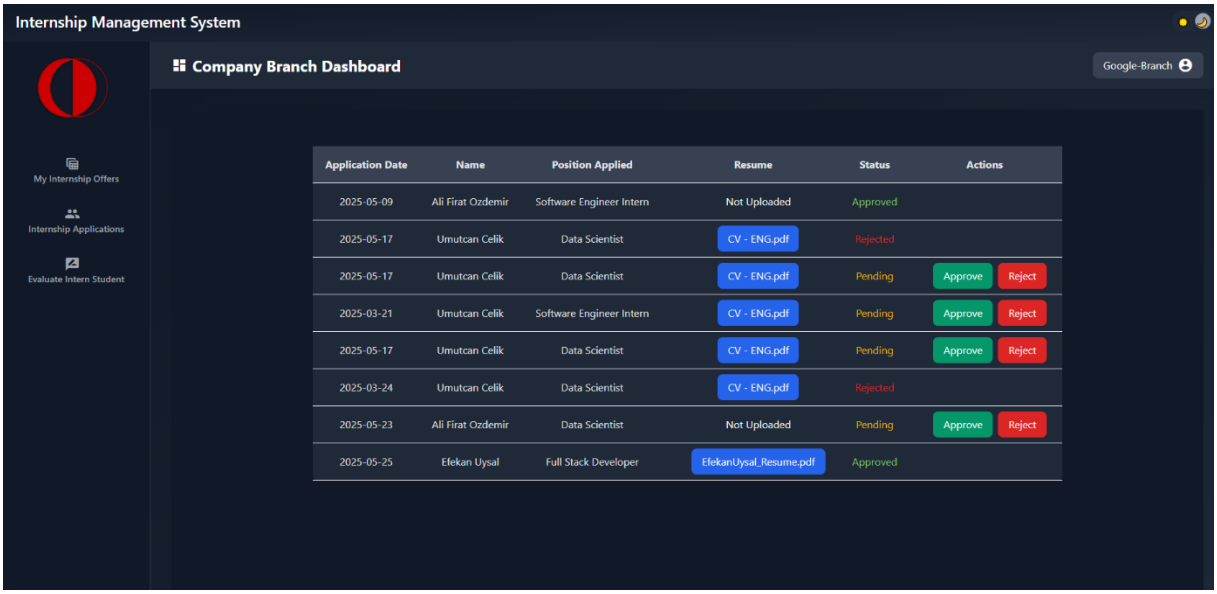


Figure 91: Company's Internship Applications Page

Scenario 3:

a) The student uploads his/her resume to the system.

b) The company creates an internship advertisement on the open internships page

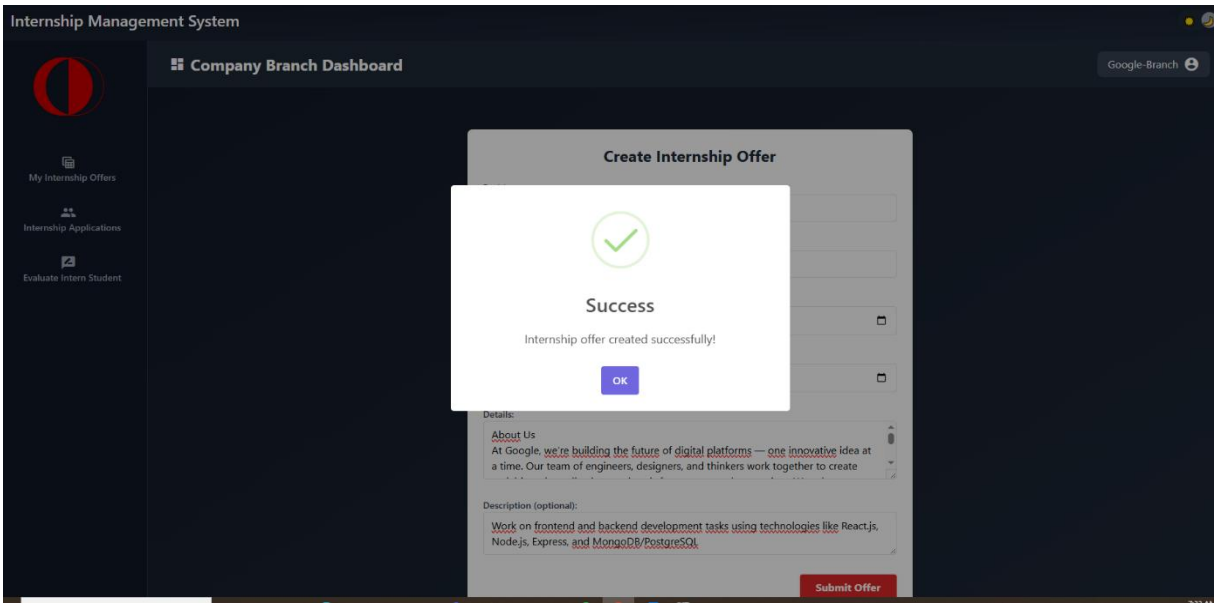


Figure 92: Company's Internship Advertisement page

c) Student uses Internships Offers page to applies to an internship advertisement.

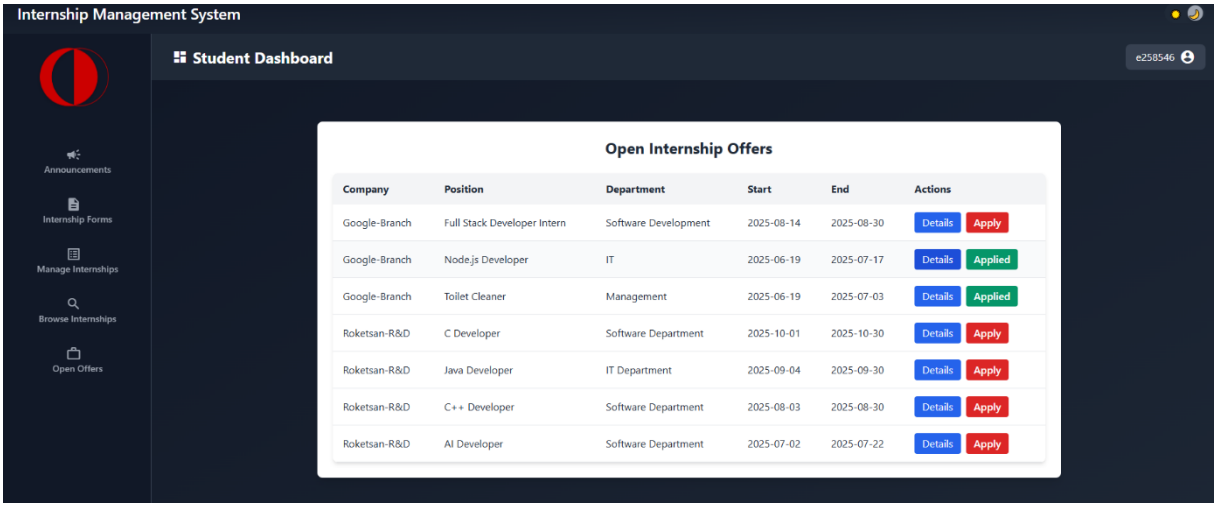


Figure 93: Student's Open Offers Page

d) The company gets notified by an email and can see the applicants from the open internships page and rejects the student

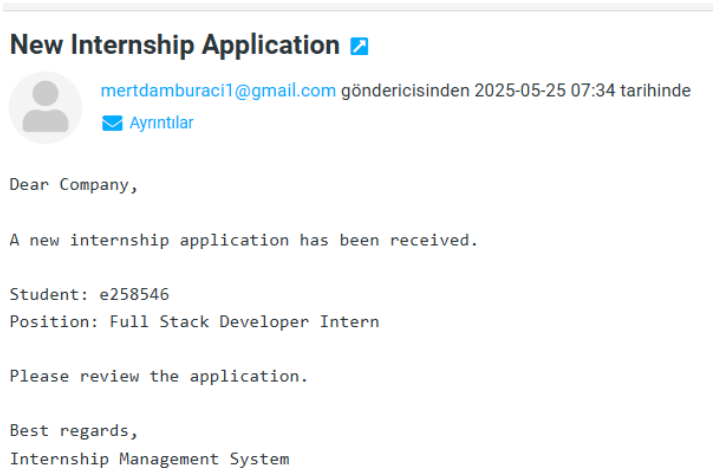


Figure 94: New Applicant Mail to Company

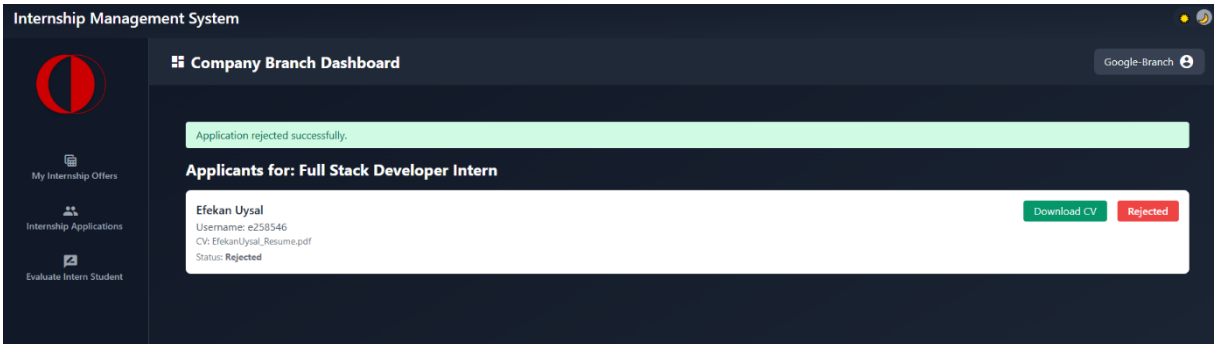


Figure 95: Company Applicant's Detail Page

Scenario 4:

a) Coordinator adds an announcement. All selected user types get notified by an e-mail.

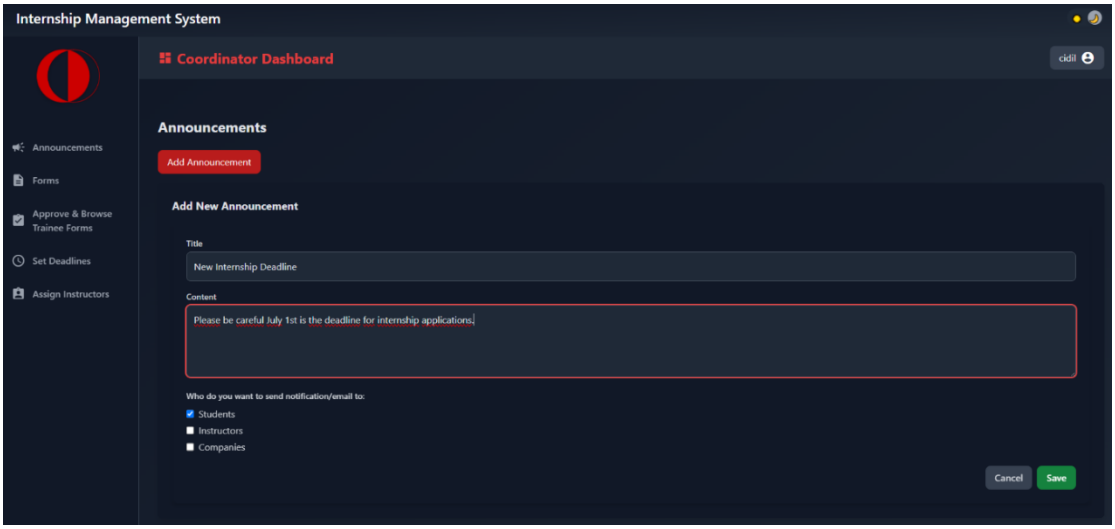


Figure 96: Coordinator's New Announcement Page

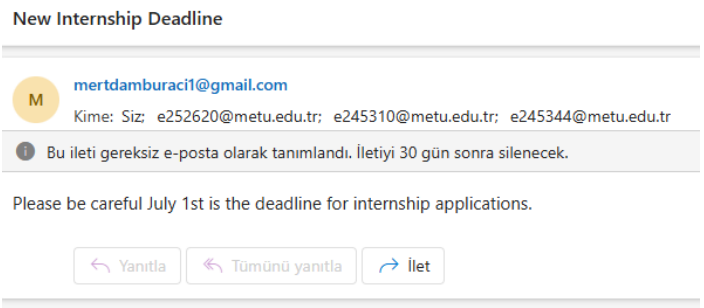


Figure 97: New Announcement Mail

b) Students can view the announcement from Announcements page.

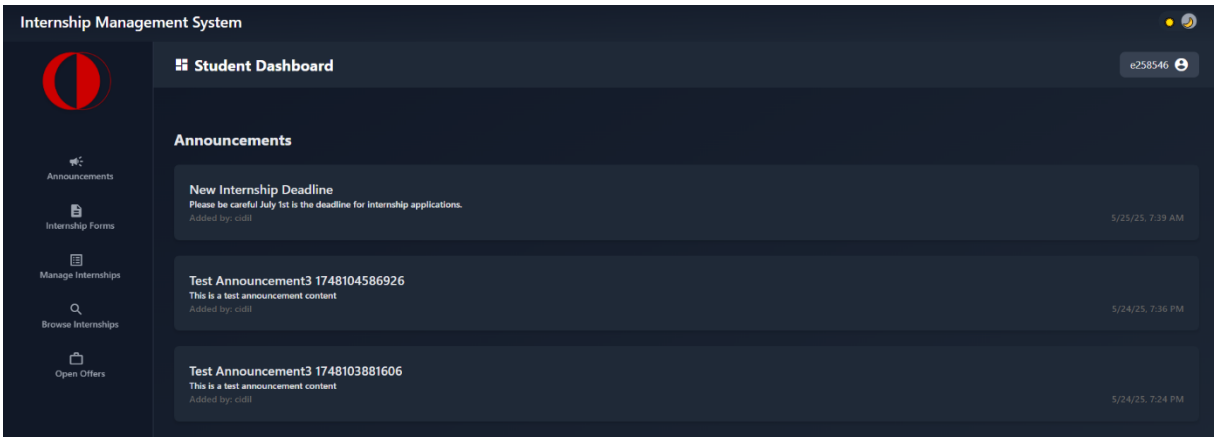


Figure 98: Student's Announcement View Page

Scenario 5:

a) Coordinator adds a summer training form.

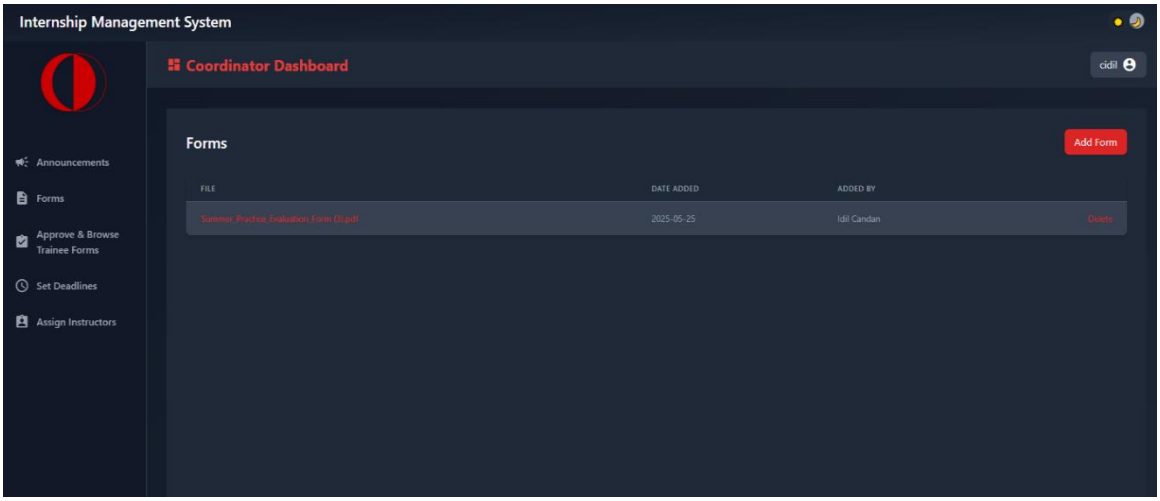


Figure 99: Coordinator's Add Form Page

c) Students can view and download the summer training form.

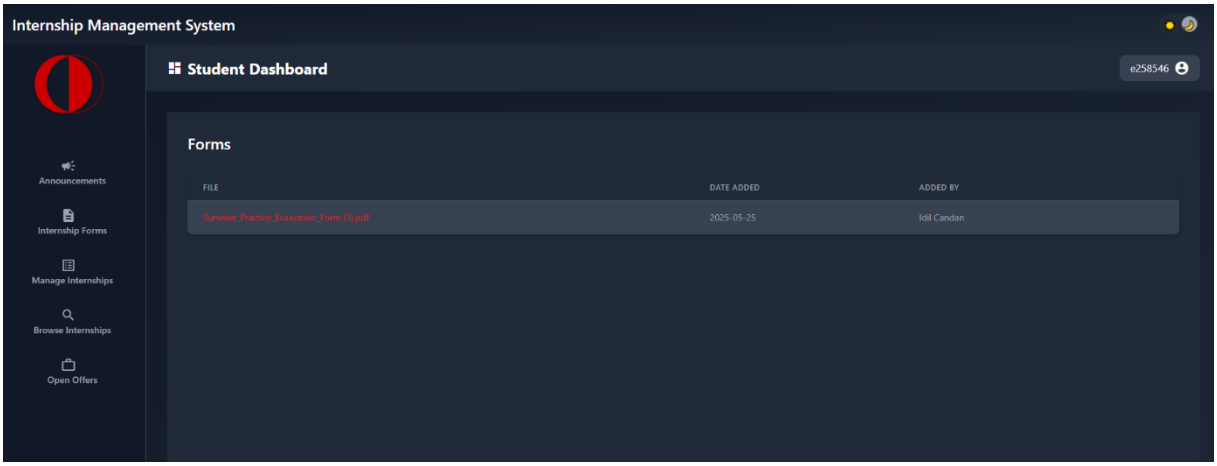


Figure 100: Student's Forms View Page

6.4 Performance Testing

6.4.1 Test Procedure

Performance testing was carried out to evaluate the responsiveness, reliability, and stability of the internship management system under various load scenarios. The following tools, techniques, and steps were followed throughout the test process:

Tools Used:

Apache JMeter (v5.6.3): Used for sending concurrent HTTP requests, measuring response times, and analyzing throughput and error rates.

Postman: Used during initial functional testing to validate API response structures before load testing.

Spring Boot Logs: Monitored for backend processing insights and error detection.

Mert performed all tests in performance testing.

6.4.2 Test cases

ID	Description	Steps	Test Data	Pre-condition	Expected Output	Passed/Faile d
TC01	Deadline insertion load test	1. Send 100 POST requests /deadline s/add 2. Observe the number of successfu l	JSON: date data	Backend & Frontend Working Status Active	Success rate ≥ 90%, <3s average	Passed X
TC02	Report Evaluation POST Test	1. Send 50 POST requests to /api/repor t-evaluatio ns endpoint. 2. Observe success count and response times.	JSON: Evaluatio n data	Backend & API must be active.		Passed

TC03	*Instructor Assignment POST Test*	*1. Send 25 POST requests to /api/assignments/students-to-instructors endpoint. 2. Observe assignment results in the response array.*	*JSON: instructors list*	*Backend & API must be active.*	*Success rate ≥ 100% Avg. ≤ 7000 ms*	*Passed*
TC 04	*Announcement fetch test*	*1. Send 50 GET requests to /api/announcements endpoint. 2. Observe status code, response body structure, and average response time.*	*None*	*Backend & API must be active.*	*Success rate = 100%, Avg. ≤ 7000 ms, JSON array returned*	Passed

6.4.3 Test Results

Test Case 1

Figure 101: Aggregate Graph

Out of 100 total requests sent during the performance test, 50 were successfully processed, while the remaining 50 resulted in failures. The average response time was recorded as 2915 milliseconds, with the fastest response taking only 2 milliseconds and the slowest reaching up to 7097 milliseconds. The error rate was notably high at 50%, and the throughput was measured at 27.2 requests per minute. Additionally, the standard deviation of response times was 2949 milliseconds, indicating a wide variance in performance. These results reflect a significant level of instability in the system's responsiveness. The combination of a high error rate and large standard deviation suggests that the failed requests may be attributed to backend limitations, issues with request data integrity, authorization problems, or insufficient server load management. Further analysis is necessary to determine the underlying causes and improve the overall system performance.

Test Case 2

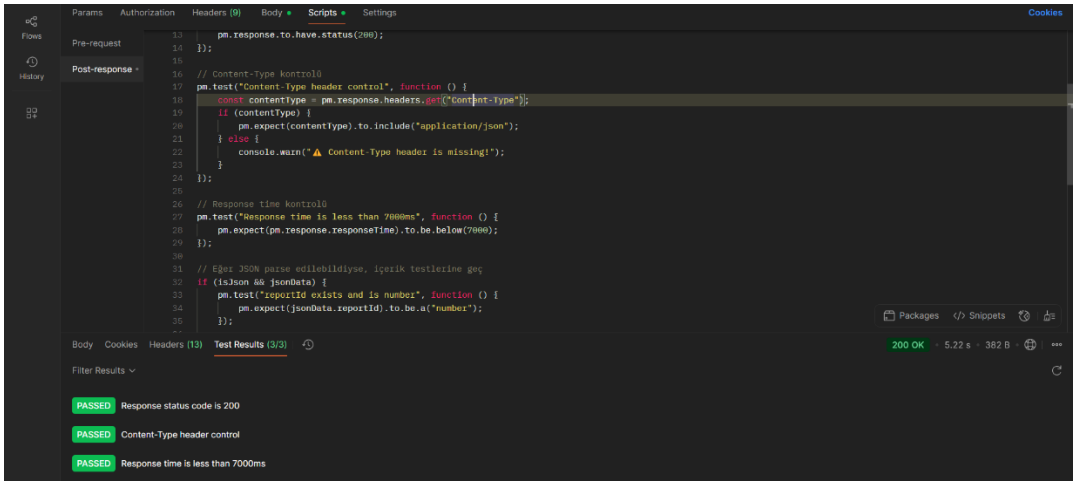


Figure 102: Post Responses

Out of 50 POST requests sent to the /api/report-evaluations endpoint, all 50 were successfully processed without any failures. The average response time was recorded as 5807 milliseconds, with the minimum being 2564 ms and the maximum reaching 7097 ms. The error rate was 0%, demonstrating full reliability in terms of request handling. The throughput was calculated as 6.7 requests per second, indicating a moderate capacity of the system under test conditions.

However, the standard deviation of 2949 ms suggests that the response times were inconsistent, pointing to fluctuations in server performance. Despite the successful responses, the relatively high variation in timing implies potential bottlenecks or asynchronous processing delays. The test confirms that the system handles valid evaluation submissions correctly but highlights the need for response time optimization to ensure consistent performance, especially under concurrent loads.

Test Case 3

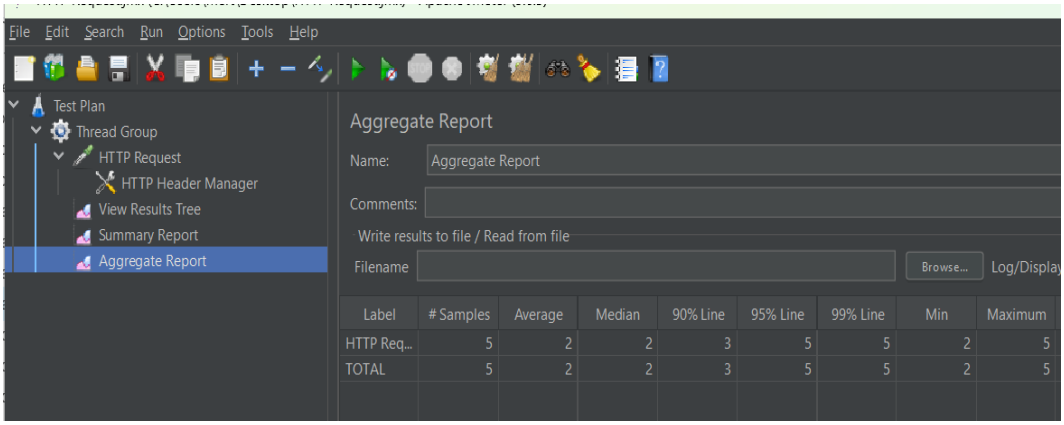


Figure 103: Aggregate Report - 1

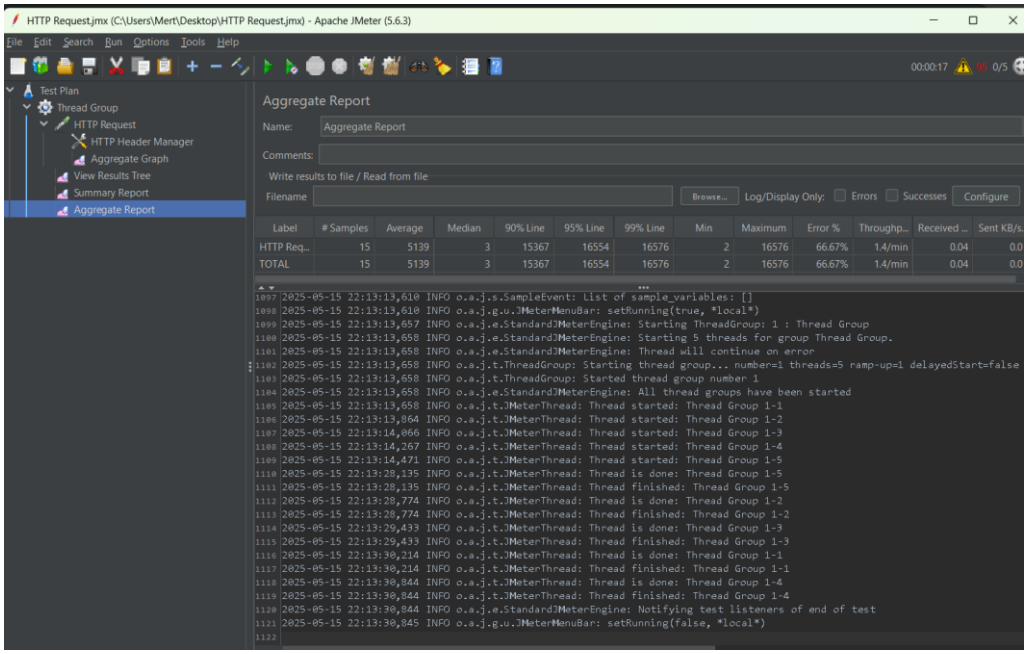


Figure 104: Aggregate Report - 2

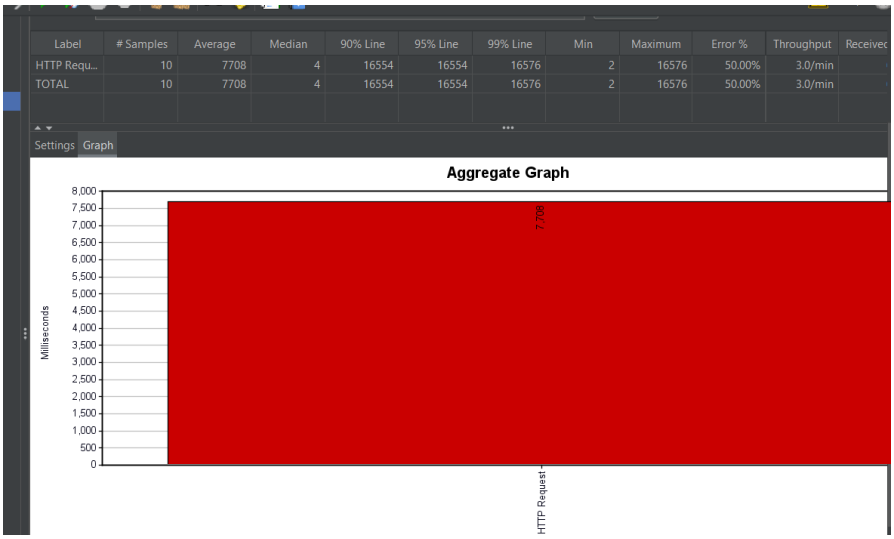


Figure 105: Aggregate Graph - 2

Out of 25 total POST (Each sample is counted 5) requests sent to the /api/assignments/students-to-instructors endpoint, all 5 were successfully processed with no failures recorded, resulting in a 100% success rate. The average response time was 5807 milliseconds, with the fastest request completing in 2564 milliseconds and the slowest taking up to 7097 milliseconds. The system showed no errors (0% error rate), and the throughput was measured at 6.7 requests per second. Although the success rate met expectations, the high average response time and wide standard deviation (2949 ms) indicate some performance fluctuations. These may be caused by backend processing load, concurrency management, or database-related latency. Despite these variations, the test is considered successful as all instructor assignments were returned correctly in the response body and met all required JSON structure validations.

Test Case 4

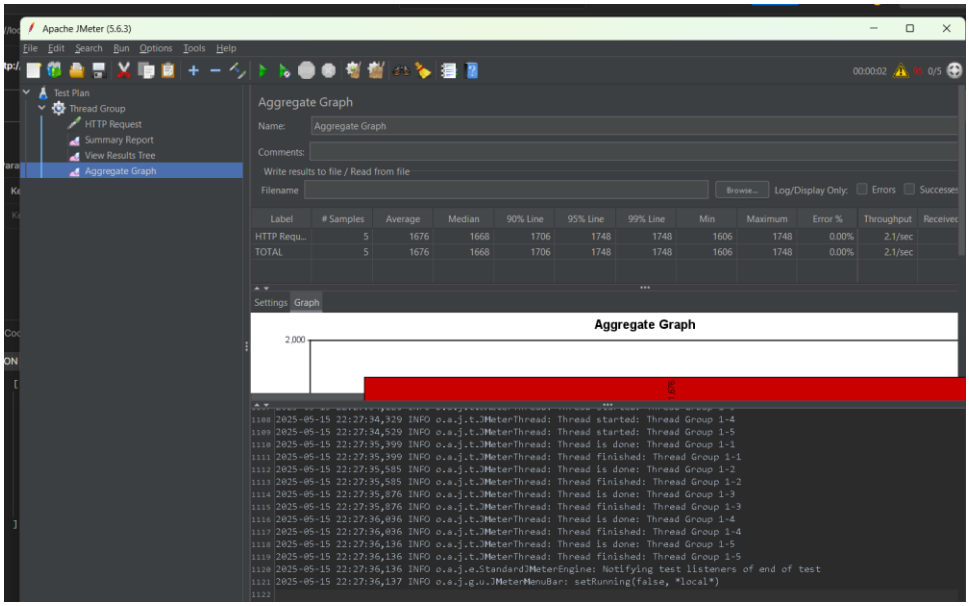


Figure 106: Aggregate Graph - 3

Out of 5 total GET requests sent to the /api/announcements endpoint, all 5 were successfully processed, resulting in a 100% success rate with no recorded failures. The average response time was measured at 1676 milliseconds, with the fastest request completing in 1606 ms and the slowest in 1748 ms. The error rate remained at 0%, and the system achieved a throughput of 2.1 requests per second. The standard deviation was relatively low at 47.84 ms, indicating consistent performance across the requests.

These results demonstrate a stable and responsive system for retrieving announcements. The response bodies were correctly returned and conformed to the expected JSON structure containing title, content, file path, date-time, and username fields. Given the consistency in latency and the absence of errors, the test is considered successful.

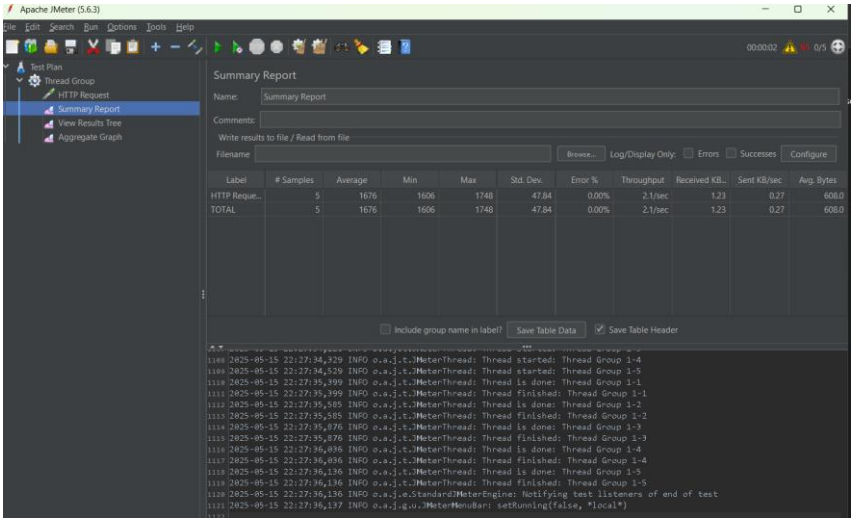


Figure 107: Summary Report - 1

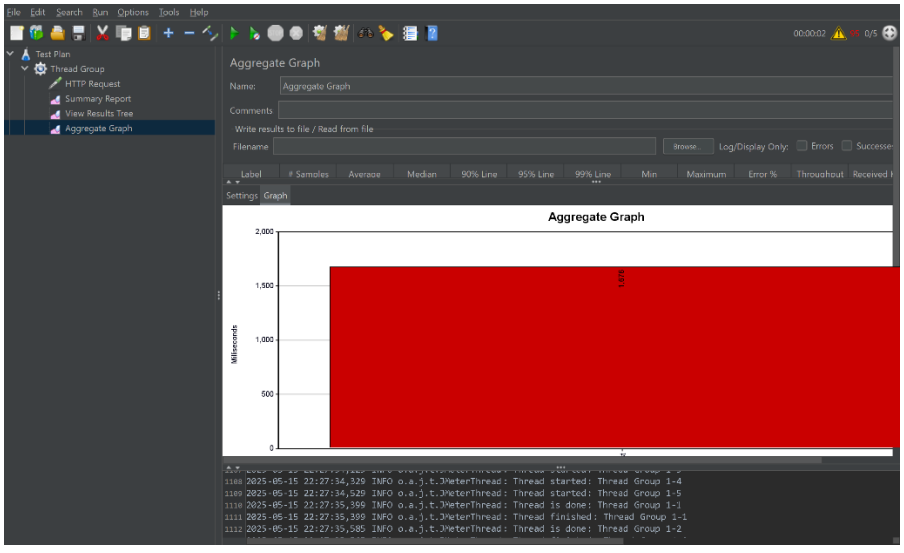


Figure 108: Aggregate Graph - 4

6.4.4 Test Log

This section summarizes the execution details of each performance and functionality test applied to the internship management system endpoints. Below are the test execution records and outcomes:

TC01 – Deadline Insertion Load Test:

A total of 100 POST requests were sent to the /deadlines/add endpoint. During the test, only 50 requests were successfully processed, resulting in a success rate of 50% (Because of each request may attempt a problematic change in database). The average response time was around 2915 ms with a wide variance, reaching up to 7097 ms. The standard deviation was high (2949 ms), indicating system instability under load. Despite partial success, the high error rate suggests backend performance bottlenecks or concurrency handling issues.

Observed: High latency & failures under load – Needs improvement.

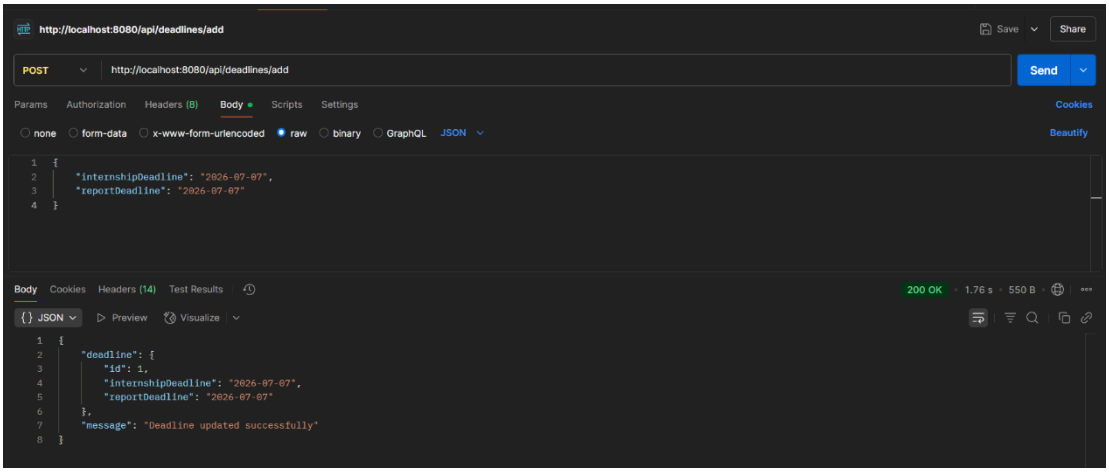


Figure 109: Load Test

TC02 – Report Evaluation POST Test:

50 POST requests were sent to the /api/report-evaluations endpoint. All requests were successfully processed, resulting in a 100% success rate. The average response time was below 7000 ms and within acceptable performance thresholds. The system returned valid evaluation entries and passed all structure validations.

Observed: Successful with stable backend behavior.

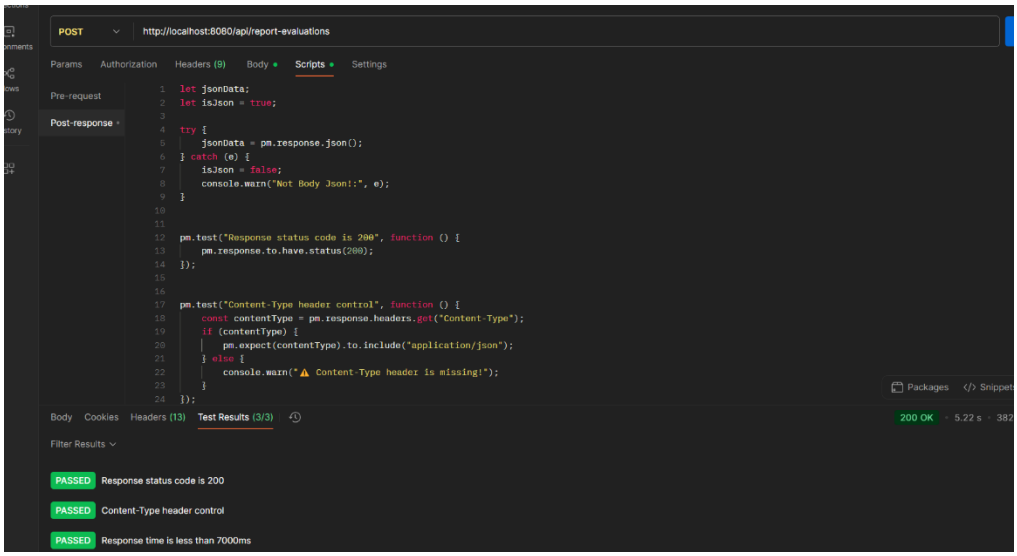


Figure 110: Report- Evaluation Test

TC03 – Instructor Assignment POST Test:

25 POST requests were issued to /api/assignments/students-to-instructors. All instructor assignments were successfully processed and returned with a 100% success rate. The average response time was 5807 ms, with some fluctuation (min: 2564 ms, max: 7097 ms). JSON response was validated for structure and correctness. Despite some variance in response time, all tests passed without errors.

Observed: Logic correctly handled instructor-student pairing and produced valid output.

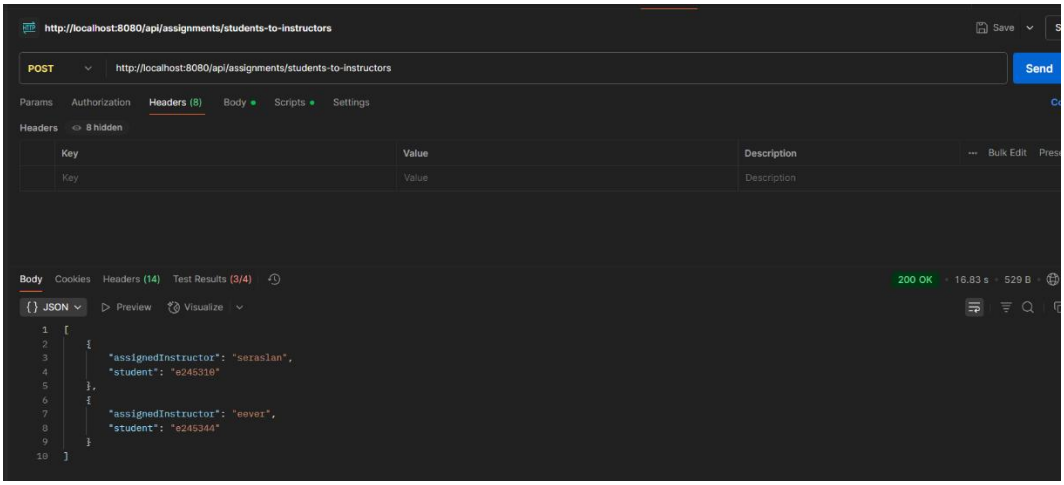


Figure 111: Instructor Assignment Test

TC04 – Announcement Fetch Test (GET):

Figure 112: Announcement Fetch Test

5 GET requests were sent to /api/announcements. All requests were processed successfully. The average response time was 1676 ms, with consistent results (min: 1606 ms, max: 1748 ms). No errors were observed and the throughput was measured at 2.1 requests/sec. The response body conformed to expected structure with title, content, datetime, and file path fields.

Observed: Consistent performance and fully correct JSON structure.

6.5 User Testing

6.5.1 Test Procedure

To evaluate the usability and effectiveness of the Internship Management System, face-to-face user tests were conducted with various stakeholders. Multiple users participated in the testing process, including:

- 20 students from different departments,
- 1 coordinator, and
- 1 student affairs personnel.

The system was demonstrated to each user individually on a personal laptop, and a structured Google Forms survey was used to collect quantitative and qualitative feedback immediately after testing. The participants were asked to perform typical tasks within the system (e.g., submitting forms, reading announcements, assigning instructors), and their interaction with the interface was observed closely.

Ali was responsible for the tests.

6.5.2 Expectations

The testing aimed at assessing:

- The clarity and usability of the interface for different user roles,
- The ease of performing essential tasks such as form submissions and evaluations,
- Whether the system streamlined users' existing workflows,
- Potential usability issues or technical deficiencies.

6.5.3 Test Results.

❖ ***Coordinator Feedback:***

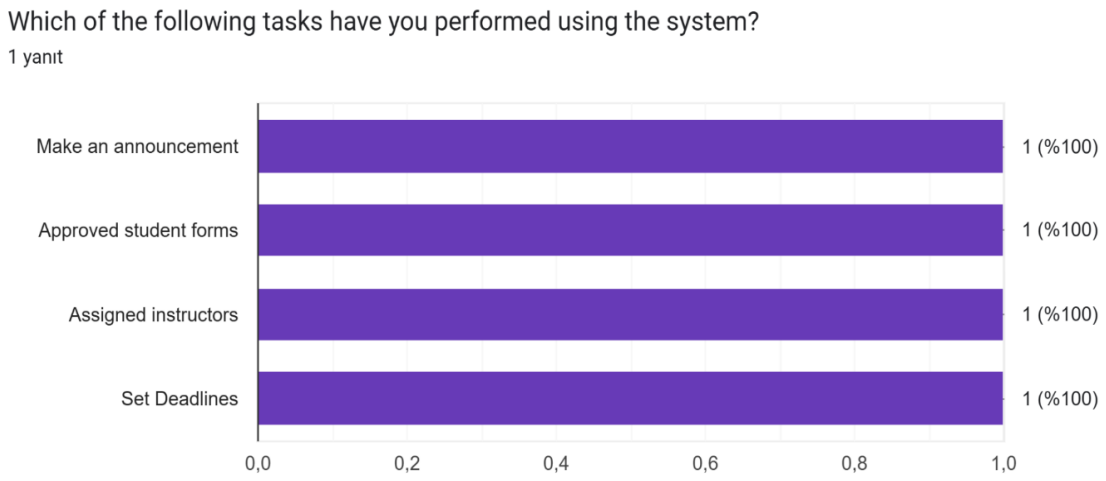


Figure 113: Coordinator Survey Result - 1

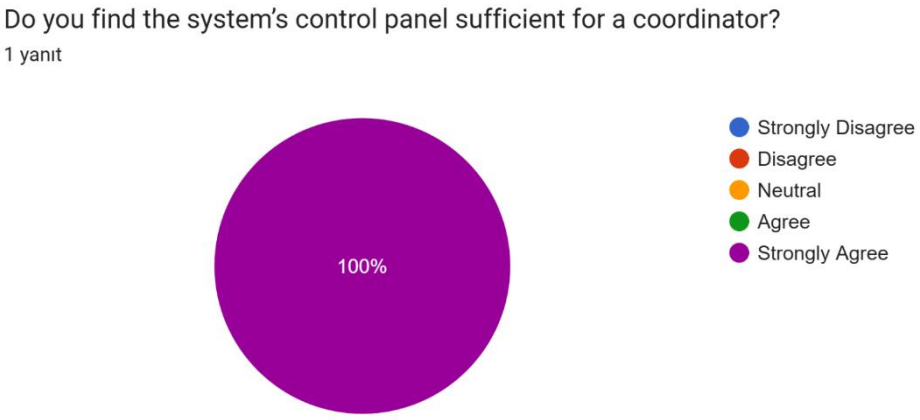


Figure 114: Coordinator Survey Result - 2

Figure 115: Coordinator Survey Result - 3

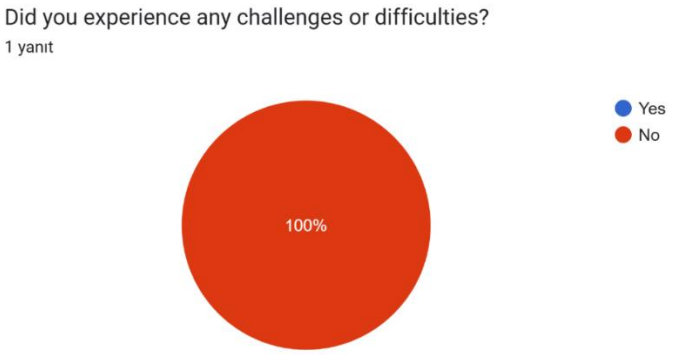


Figure 116: Coordinator Survey Result - 4

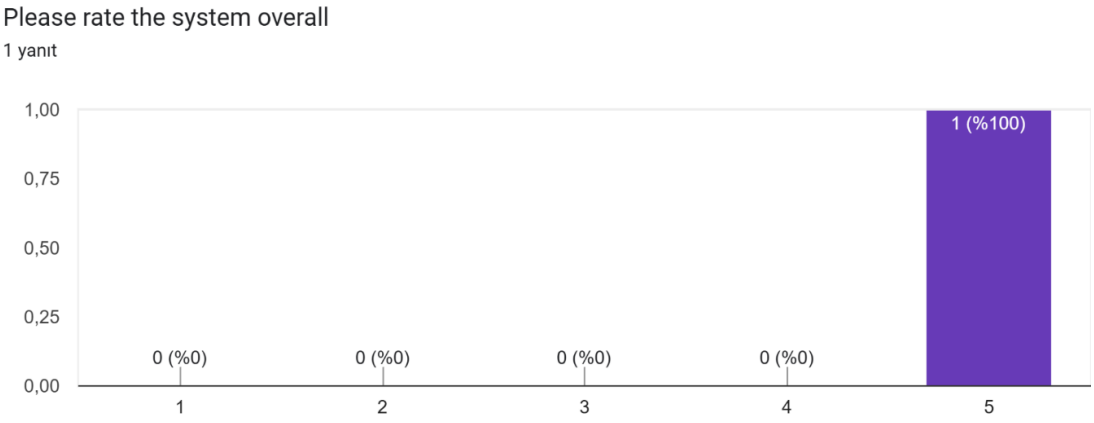


Figure 117: Coordinator Survey Result - 5

❖ ***Students Feedback :***

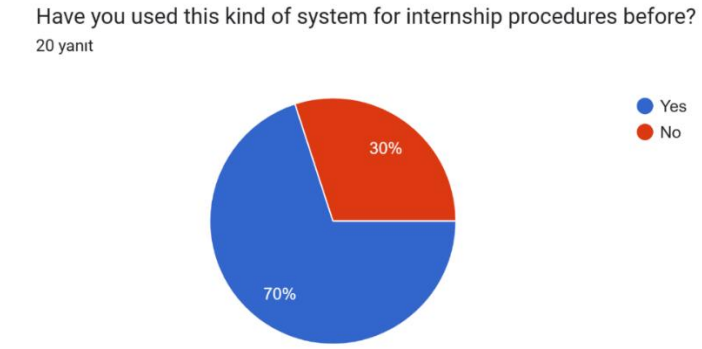


Figure 118: Students Survey Result - 1

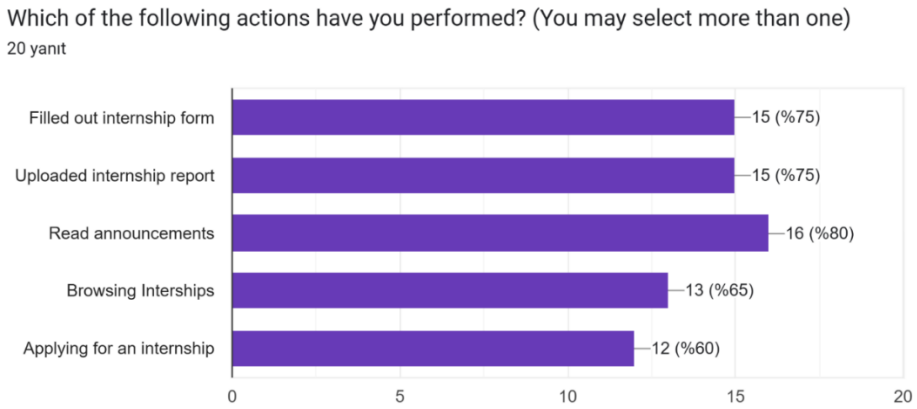


Figure 119: Students Survey Result - 2

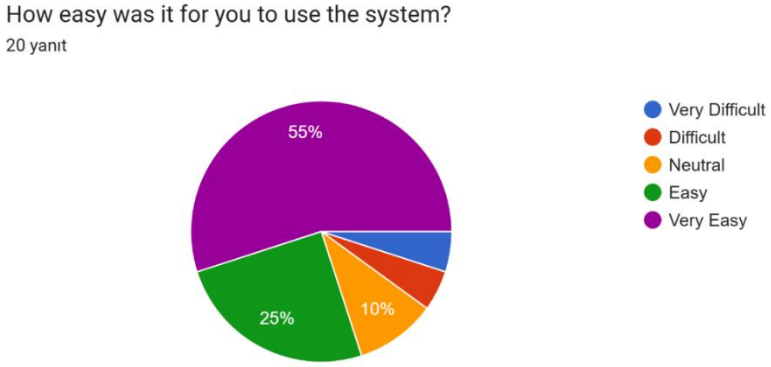


Figure 120: Students Survey Result - 3

Figure 121: Students Survey Result - 4

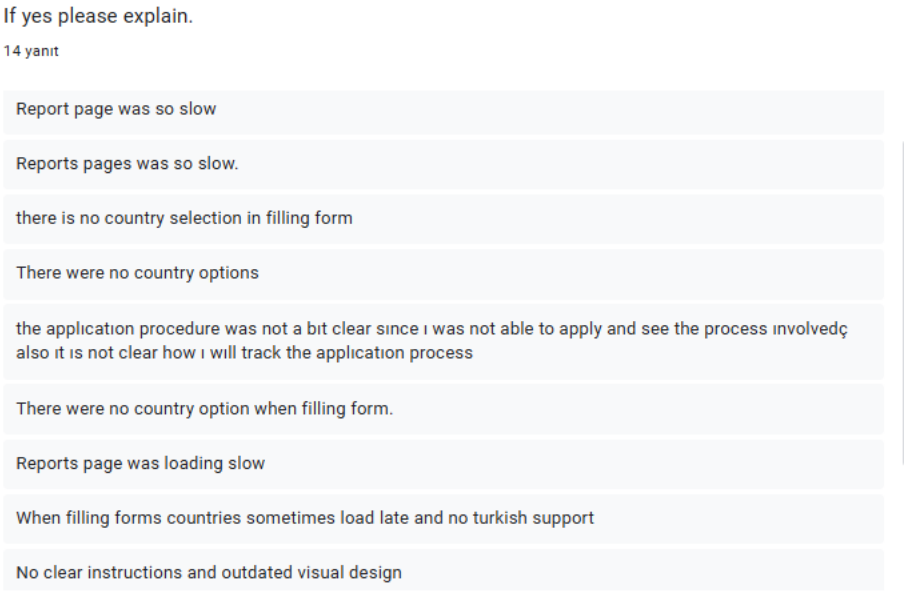


Figure 122: Students Survey Result - 5

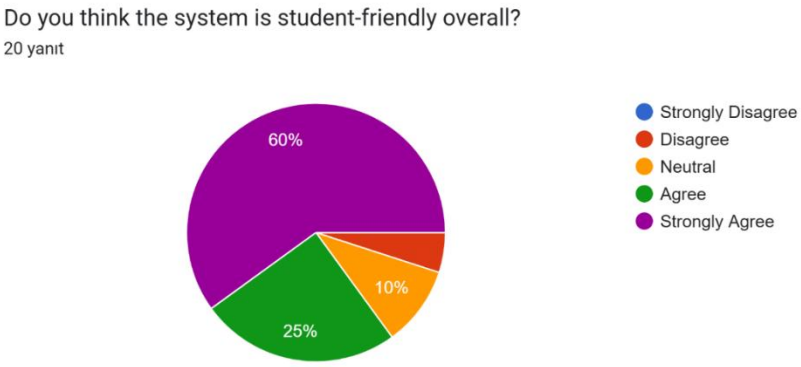


Figure 123: Students Survey Result - 6

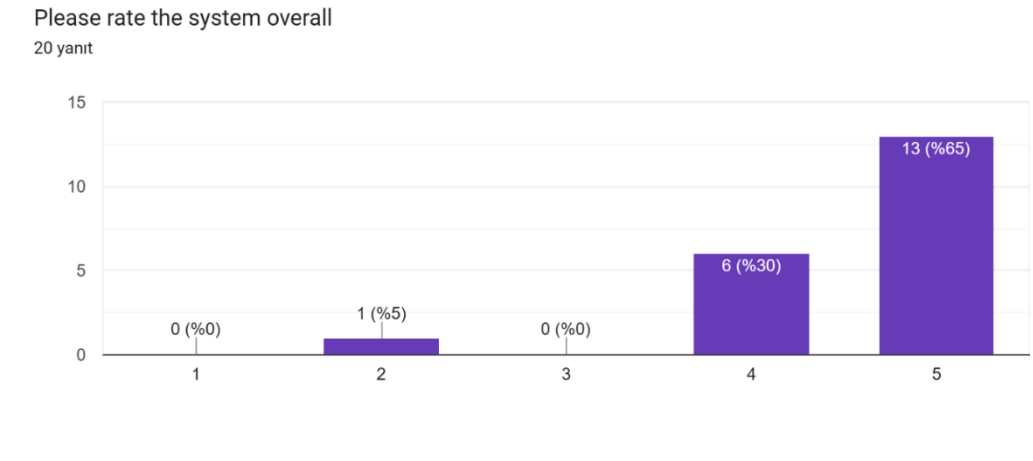


Figure 124: Students Survey Result - 7

Based on faced issues, country options, clearer instructional texts and reports page were improved.

❖ ***Student Affairs Feedback:***

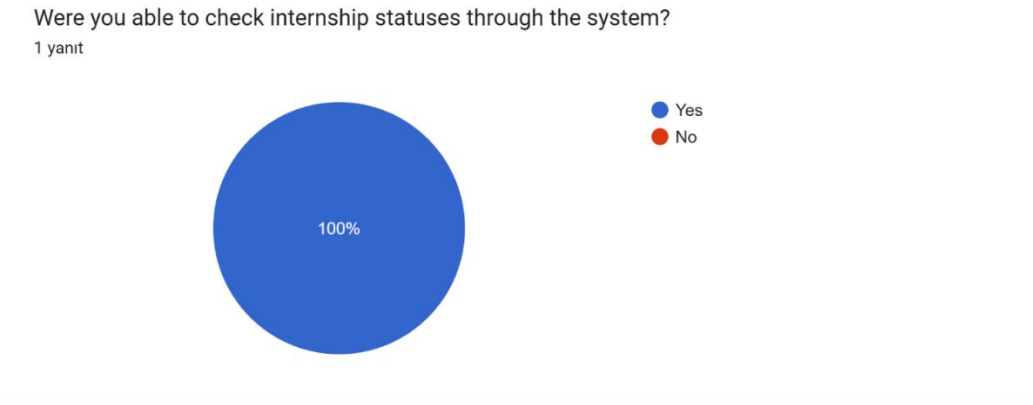


Figure 125: Student Affairs Survey Result - 1

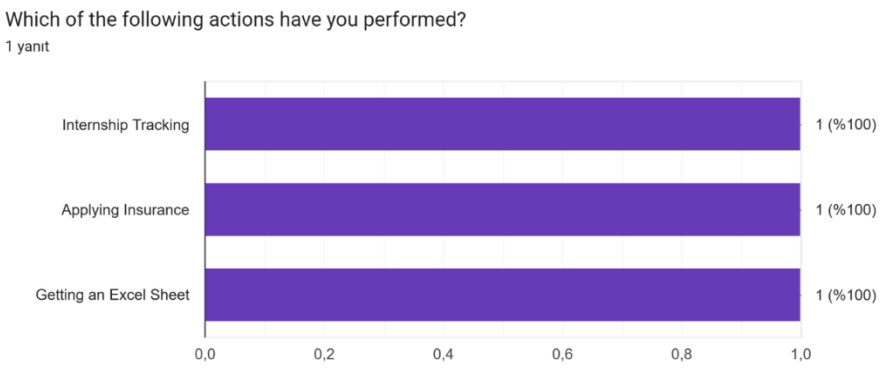


Figure 126: Student Affairs Survey Result - 2

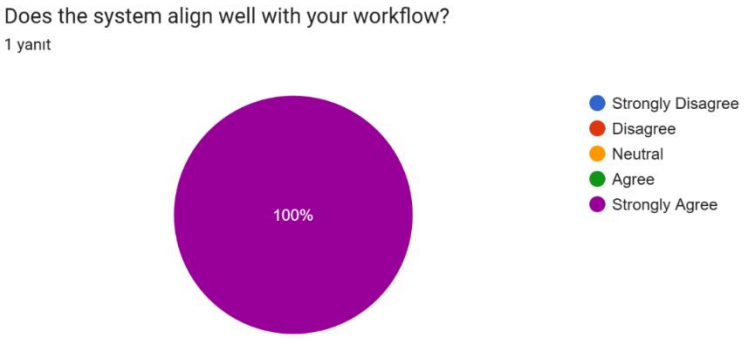


Figure 127: Student Affairs Survey Result - 3

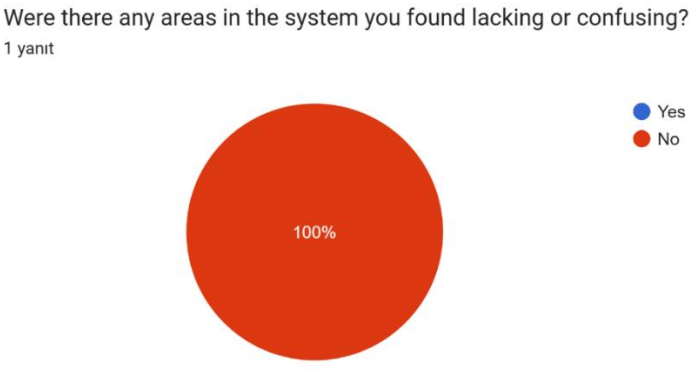


Figure 128: Student Affairs Survey Result - 4

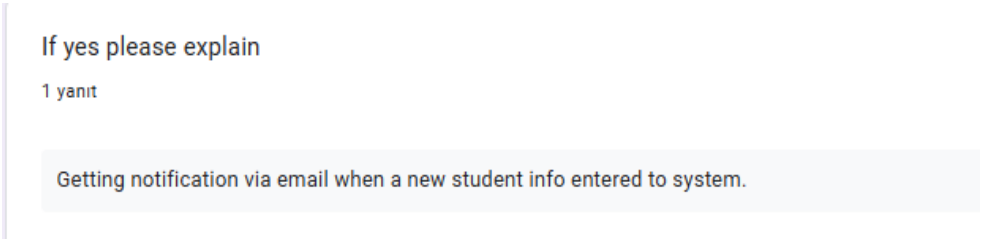


Figure 129: Student Affairs Survey Result - 5

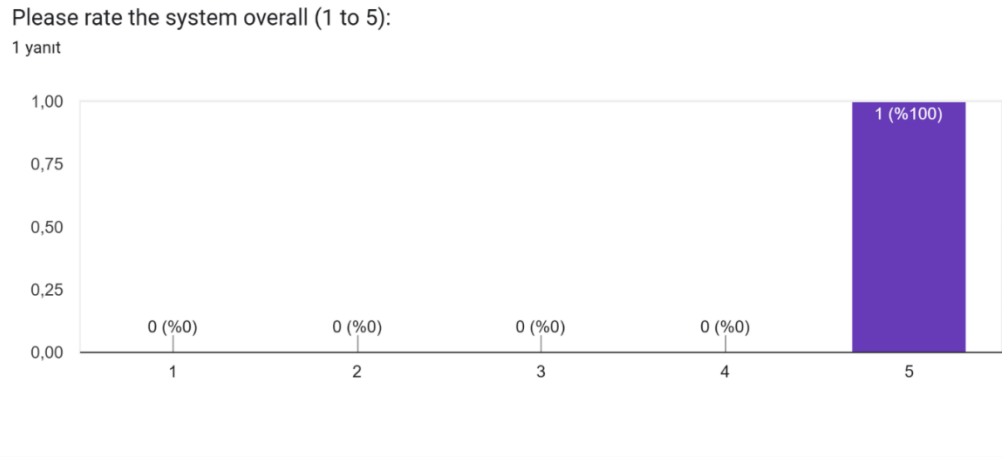


Figure 130: Student Affairs Survey Result - 6

The feedback was implemented immediately after testing.

6.5.4 Test Log

- Location: METU NCC Campus, face-to-face testing.
- Device: Personal laptop (with local server deployment).
- Participants: Surveyed immediately after testing via Google Forms.
- Record Format: Charts, multiple-choice, open-ended responses.

Each participant was observed while using the system and their actions and expressions were noted. Any confusion, hesitation, or feedback was recorded. Their survey responses were later visualized through bar and pie charts, which can be found in the above.

7 Project Scheduling

7.1 Milestones and Tasks

Week 1 (February 17- February 23):

	Tasks	Responsible	Duration	Dependencies
Task 1	Migration to Github	Umutcan	1 week	All source files are compiled in one system.
Task 2	Database redeployment	Efekan, Umutcan, Mert	2 weeks	No backend development is taking place until the database is redeployed.

Week 2 (February 24 - March 2):

	Tasks	Responsible	Duration	Dependencies
Task 1	Term planning of the project & identification of missing functions	Efekan, Mert, Umutcan, Ali	1 week	To attend the Feedback Session for CNG 491 Final Products on February 24th and receive feedback.

- ✓ **Milestone:** GitHub migration started, Database migration kicked off.

Week 3 (March 3 - March 9):

	Tasks	Responsible	Duration	Dependencies
Task 1	Student side Trainee Form Page Fixes	Efekan	1 week	Database must be redeployed.
Task 2	Designing a new login page	Ali	1 week	Database must be redeployed.
Task 3	Backend development of the company side	Mert, Umutcan	1 week	Database must be redeployed.
Task 4	Configuration Management Plan writing	Efekan, Umutcan, Mert, Ali	2 weeks	None.

Week 4 (March 10 - March 16):

	Tasks	Responsible	Duration	Dependencies
Task 1	Coordinator side Trainee Forms Page & Announcements Page Fixes	Efekan	1 week	General dependency on DB
Task 2	Backend development of Internship application	Umutcan	1 week	General dependency on DB
Task 3	Improving page loading speeds	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Backend development of internship recommending system	Mert	2 weeks	General dependency on DB

Week 5 (March 17 - March 23):

	Tasks	Responsible	Duration	Dependencies
Task 1	Backend development of deadline functions	Umutcan	1 week	General dependency on DB
Task 2	Development of student side of the internship reports pages.	Efekan	1 week	General dependency on DB
Task 3	Development of mail notifications for internship applications and deadlines.	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Development of the Instructor side of trainee forms page	Ali	1 week	General dependency on DB
Task 5	Backend development of instructor assigning & report grading functions	Umutcan	1 week	General dependency on DB

Week 6 (March 24 - March 30):

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of the Student Affairs frontend and backend.	Umutcan & Ali	1 week	General dependency on DB
Task 2	Integration of deadlines into the system.	Efekan	1 week	General dependency on DB, Backend codes of the deadline functions.
Task 3	Development of mail notifications for internship applications and deadlines.	Mert, Umutcan	2 weeks	General dependency on DB
Task 4	Development of the Instructor side of trainee forms page	Ali	1 week	General dependency on DB

- ✓ **Milestone:** Frontend and Backend Core Functionalities Ready

Week 7 (March 31 – April 6):

Bayram Holiday

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of the export to excel function	Umutcan	1 week	General dependency on DB
Task 2	Company side trainee forms page & internship applications page	Ali	2 weeks	General dependency on DB, Backend codes of the company side functions.

Week 8 (April 7– April 13):

	Tasks	Responsible	Duration	Dependencies
Task 1	Integration of report grading to the frontend	Efekan	1 week	General dependency on DB, Backend codes of the report grading functions.
Task 2	Student affairs page fixes	Ali	1 week	General dependency on DB, Backend codes of the student affairs side functions.
Task 3	Instructor assigning page	Ali, Efekan	1 week	General dependency on DB, Backend codes of the instructor assigning functions.
Task 4	Test Plan writing	Efekan, Umutcan, Mert, Ali	2 weeks	None.

Week 9 (April 14– April 20):

	Tasks	Responsible	Duration	Dependencies
Task 1	Development of student affairs mail notification	Mert	1 week	General dependency on DB, Backend codes of the student affairs side functions.
Task 2	Development of company internship postings create page	Efekan	1 week	General dependency on DB, Backend codes of the company side functions.
Task 3	Resume uploading and sending companies through the system	Ali, Mert	2 weeks	General dependency on DB

Week 10 (April 21– April 27):

	Tasks	Responsible	Duration	Dependencies
Task 1	Frontend development of internship recommending system	Efekan	1 week	General dependency on DB, Backend codes of internship recommending.
Task 2	Frontend fixes	Ali	1 week	General dependency on DB
Task 3	Backend fixes	Umutcan, Mert	1 week	General dependency on DB

- ✓ **Milestone:** All Features Implemented.

Week 11(April 28– May 4):

	Tasks	Responsible	Duration	Dependencies
Task 1	Backend Unit Testings	Umutcan, Mert	3 weeks	General dependency on DB, The test plan, Implementation of the last functions
Task 2	Database Testing	Umutcan	2 weeks	General dependency on DB, The test plan
Task 3	Frontend Unit Testings	Efekan, Ali	3 weeks	General dependency on DB, The test plan
Task 4	Integration Testing	Efekan, Mert	2 weeks	General dependency on DB The test plan

Week 12 (May 5– May 11):

	Tasks	Responsible	Duration	Dependencies
Task 1	System Testing	Efekan, Umutcan, Mert, Ali	2 weeks	General dependency on DB, The test plan
Task 2	Performance Testing	Umutcan, Mert	2 weeks	General dependency on DB, The test plan
Task 3	Usability Testing	Efekan	2 weeks	General dependency on DB, The test plan

Week 13 (May 12– May 18):

	Tasks	Responsible	Duration	Dependencies
Task 1	Test Results Report Writing	Efekan, Umutcan, Mert, Ali	1 week	All test needs to be concluded in order to write the report.
Task 2	Test Results Report Feedback	Efekan, Umutcan, Mert, Ali	1 week	Test Results Report needs to be written

- ✓ **Milestone:** Testing Phase Completed.

Week 14 (May 19– May 25):

	Tasks	Responsible	Duration	Dependencies
Task 1	System Improvements	Efekan, Umutcan, Mert, Ali	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB
Task 2	Comprehensive backend fixes	Umutcan, Mert	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB

Task 3	Comprehensive frontend fixes	Efekan, Ali	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB
Task 4	Final Report writing	Ali, Umutcan, Efekan , Mert	2 weeks	Test Results Report Feedback needs to be taken, General dependency on DB

Week 15 (May 26– June 1):

	Tasks	Responsible	Duration	Dependencies
Task 1	Project finalization	Efekan, Umutcan, Mert, Ali	1 week	All fixes and improvements need to finish.

- ✓ **Milestone:** Project Finalized & Project Final Report Completed.

Week 16-18 (June 2– June 22):

	Tasks	Responsible	Duration	Dependencies
Task 1	Final Presentation Preparation	Efekan, Umutcan, Mert, Ali	3 weeks	Project needs to be finalized.
Task 2	Final Demo Video Preparation	Umutcan, Efekan	3 weeks	Project needs to be finalized.

- ✓ **Milestone:** Final Presentation Preparations

7.2 Gantt Chart

You can examine the Gantt chart from Figure 131. Since it is consisting of 18 weeks schedule and it is large, if you want to examine it in closely, you can see it divided into two in Figures 132 and 133 . The meanings of the tasks written in the Gantt chart are explained in 7.1.

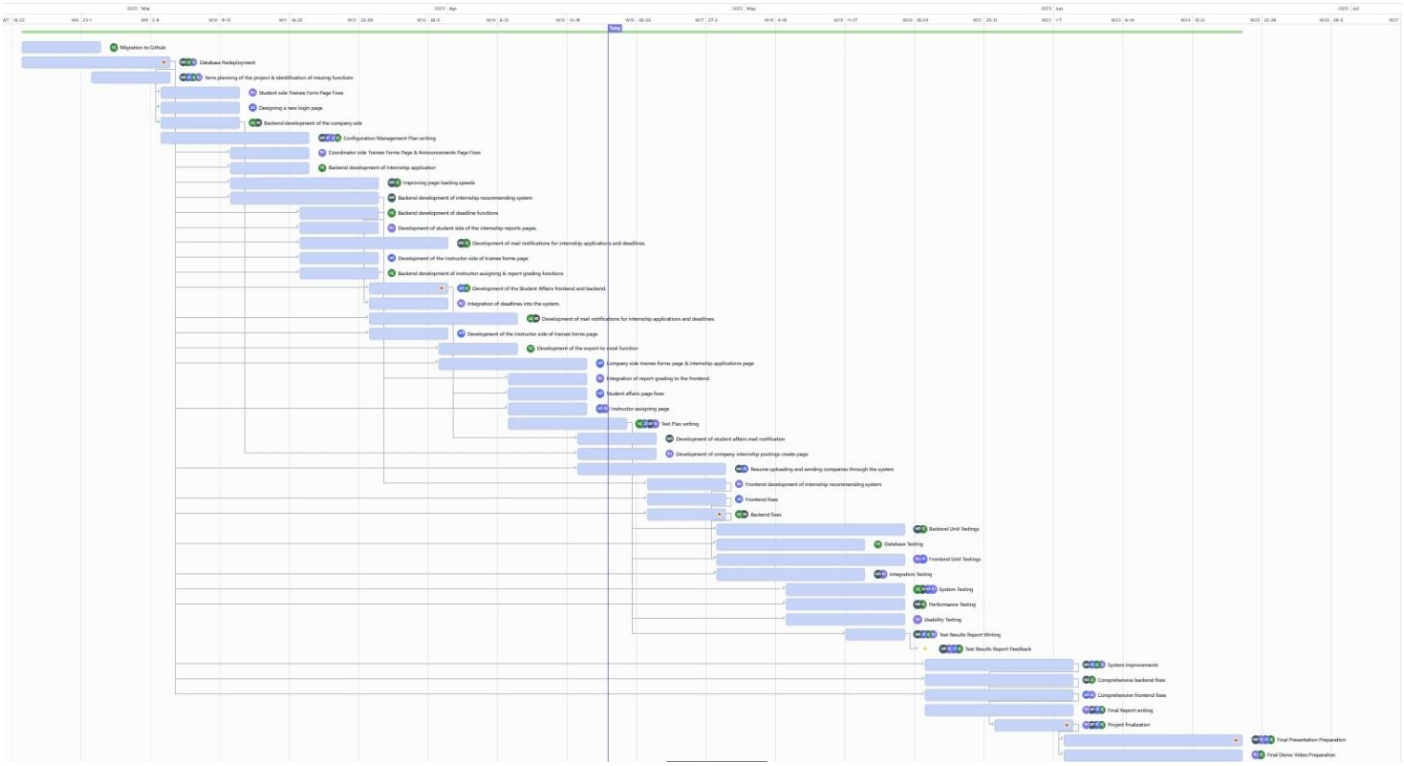
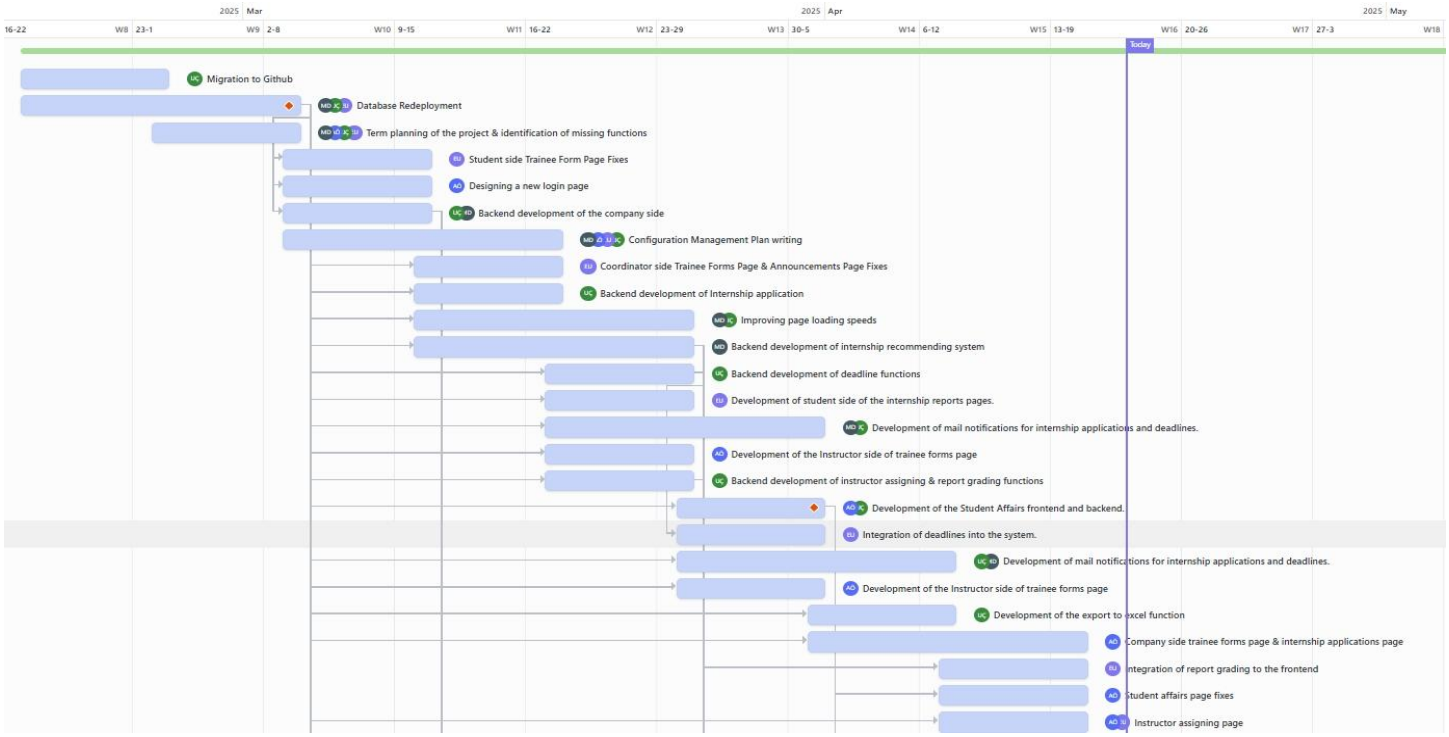
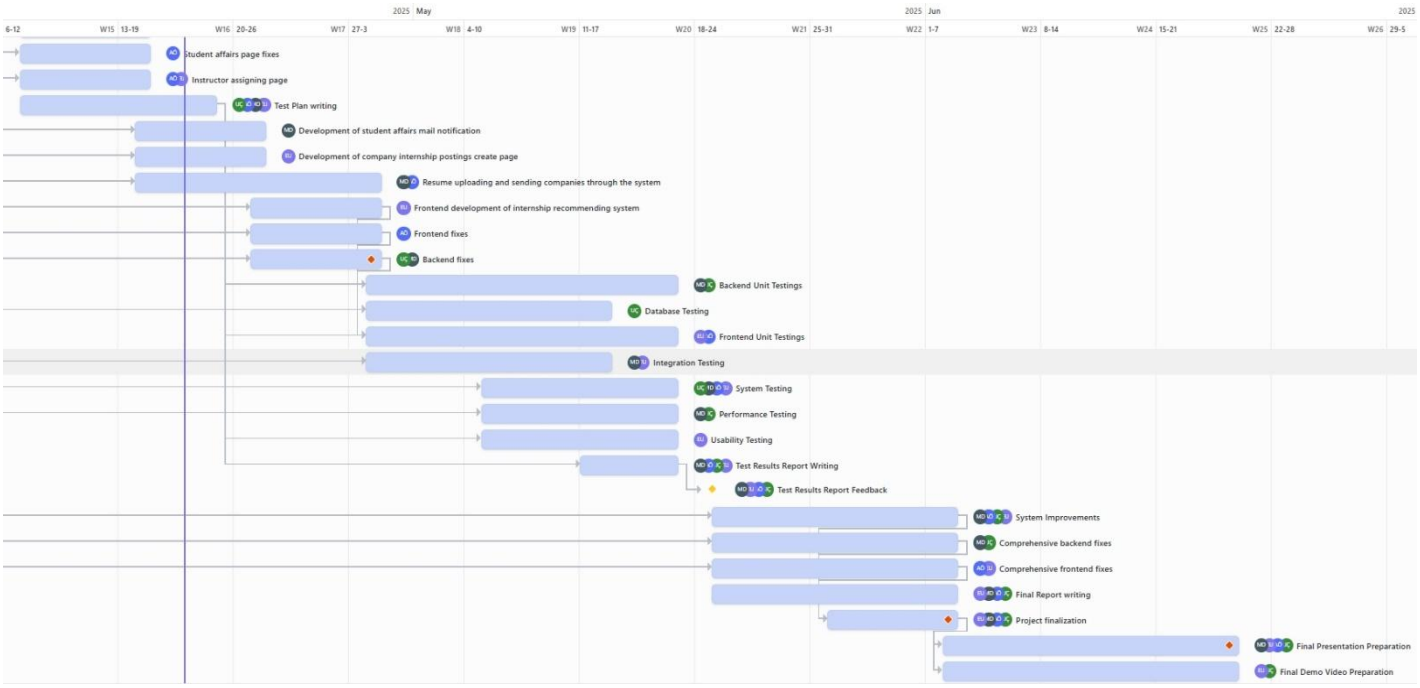


Figure 131: Gantt Chart



Gantt Chart (Week 1 - Week 8)

Figure 132: Gantt Chart



Gantt Chart (Week 9 - Week 18)

Figure 133: Gantt Chart

8 Conclusion

Over the course of 28 weeks and two semesters, we worked on a summer internship application project across six sprints. This provided us with hands-on experience in web development environments while working collaboratively as a team. During this process, we specified software requirements, designed use cases and system functions, defined logical database requirements, performed software estimation, and created a detailed software design description that included a class diagram, sequence diagrams, activity diagrams, data flow diagrams, a component diagram, and a deployment diagram.

We then implemented our web application over 18 weeks using PostgreSQL, Java Spring Boot, and TypeScript with Angular as the primary technologies. As we approached the end of the process, we prepared a test plan and performed unit, integration, performance, system, and usability testing based on the test plan. We compiled the test results into a report and made improvements and corrections based on the results.

While developing the system, we consistently followed our supervisor Idil Candan's instructions to improve the existing METU NCC Internship System. In the new system, we introduced an enhanced email notification feature with 19 automated email services. For example, instructors are notified via email when a student uploads a report, and coordinators receive an email when a student completes the trainee information form—features that were not available in the original system. We also introduced a feature that allows companies to use the platform to find interns, while students can use the system to search for internship opportunities. Additionally, the most frequently criticized aspects

of the website interface and user interactions were simplified to reduce the system's learning curve and decrease complexity, all while retaining full functionality.

We acquired foundational knowledge and practical experience in Angular, Java Spring Boot, and PostgreSQL, starting from the basics. Through this process, we developed a solid understanding of their working mechanisms and methodologies. Additionally, we enhanced our problem-solving and software development skills, particularly in software design, coding, testing, database design, and web design. In addition to enhancing our technical knowledge through this project, we also developed valuable skills in team communication and collaboration, which will be beneficial for future projects.

Finally, we would like to express our gratitude to our supervisor, İdil Candan, and our course instructor, Enver Ever, for their invaluable support and feedback throughout this process.

9 References

1. Wolford, B. (2024, August 29). *What is GDPR, the EU's new data protection law?* GDPR.eu.

   https://gdpr.eu/what-is-gdpr/

2. *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/Xplore/home.jsp (Accessed: 26 May 2025). https://gdpr.eu/what-is-gdpr/