



Bilkent University

Department of Computer Science

CS 319: Object-Oriented Software Engineering

INTERNSHIP MANAGEMENT SYSTEM

Final Report

29 May 2023

RIGEL

Aytekın İsmail, 22003988

Ece Ateş, 22002908

İzgi Nur Tamcı, 22002682

Ömer Asım Doğan, 21903042

Zeynep Begüm Kara, 22003880

1. Introduction.....	3
2. Project Experience.....	3
a. Requirement Analysis.....	3
b. Design of the System.....	4
c. Implementation.....	5
d. What We Did, Did Not, and Why.....	5
i. What we did.....	5
ii. What we did not:.....	7
e. Lessons Learned.....	8
3. Build Instructions.....	8
4. User's Guide.....	8
5. Work Allocation.....	8
a. Aytekin İsmail.....	8
b. Ece Ateş.....	9
c. İzgi Nur Tamcı.....	10
d. Ömer Asım Doğan.....	10
e. Zeynep Begüm Kara.....	10
6. References.....	11

1. Introduction

Bilkent University has a requirement for second and third-year engineering students to participate in a summer training program lasting twenty workdays. The objective of the program is to enhance students' academic and professional development. The process includes tasks such as registering with internship companies and completing specific courses within the department, such as xx299 or xx399 [1]. During the fall semester, around 1300 students undertake these courses. However, managing such a large number of students and adhering to university policies presents a significant organizational challenge for department secretaries.

As part of the program, students must submit a report that highlights what they have learnt during the summer training. This report ensures their active involvement in the professional experience and helps the evaluation of the credibility of the participating companies for future candidates. Instructors assess these internship reports based on ABET criteria. Students whose reports do not meet all the criteria may need to revise their work multiple times under the guidance of their assigned instructor. Ultimately, students receive a satisfactory or unsatisfactory status based on their performance in the summer training and the XX299/XX399 courses.

To address the organizational burden on department secretaries and improve efficiency, the Rigel Internship Evaluation System is being developed as a web-based application. This system aims to integrate all aspects of the internship process into a single platform, benefiting department secretaries, evaluator instructors, and registered students.

This report introduces the Rigel software, which manages internships for the Bilkent Engineering Department. This final report of the software highlights the process we as a group went through during this project and explains the reasons behind the certain design and implementation choices we've made. This report also covers the build instructions and user guide needed to use our project locally. It also includes the lessons we've learnt about group projects while creating this website and the work allocation of our program.

2. Project Experience

a. Requirement Analysis

We understood the importance of requirement analysis in our project and made an enormous effort to identify clients' needs and how they problematized the internship management process. Before the presentation of Selim Aksoy about the current system and what is the general expectation of the department, we did a general research about the internship system to learn the application domain and understand better his expectation. The presentation defined the problem comprehensively and, what we are going to build become more clear. Then, we identified the possible stakeholders and came up with use cases based on their roles and needs. During that phase, we saw that only the presentation, which mostly reflects the department chair aspect, is not enough to cover the whole project in terms of the different needs of each actor and detail. Hence, we have talked to Begüm Çınar,

CS department secretary, Eray Tüzün, instructor, and several Bilkent students who have completed at least one summer training course. Telling them what we have understood, what we are planning to build, and what are our assumptions about some specific situations, we wanted to ensure that our work aligned with their specific needs. For example, talking to Begüm Çınar we have concluded that there is a lot of paperwork to be done which can be automated. Hence, we add an automatic pdf generation functionality. Talking to Eray Tüzün, we concluded that evaluating internship reports is a time-consuming task and we tried to minimize it by using online annotations through Google Drive that skips downloading the student report, annotating the pdf on their computer, and uploading it again. Also, we have added criteria mode functionality that reduces time spent by changing windows by displaying both the internship report uploaded and criteria questions side by side on a single page. Updating the use cases based on their statements, we also observed conflicting cases. For example, even though the department secretary indicated that criteria questions are only answered once after the report revisions were completed, some instructors indicated that they ask for revisions based on that criteria. Hence, we came up with the save/submit criteria report functionality and changed some prerequisites, such as the criteria report can only be filled after it is certain that no revision is asked so that this conflict is solved. State and activity diagrams helped us to think of some corner cases and the flow of events. Also, having a unified model brought all the group members on the same page.

b. Design of the System

For the design of the system, we have tried to find the best methods for the functionality we were aiming to implement. Hence, we have decided to use Spring Boot for the backend infrastructure, React.js for the frontend infrastructure, and Google Drive API for the report management. We have also used Spring Boot to communicate with our database, and React.js to communicate with our backend structure. We have chosen to use Google Drive API due to its relatively easy integration, many functionalities, and Google's great maintenance. However, it didn't turn out exactly the way we wanted. To ease the process of giving feedback, we wanted to integrate the internship report of the student which will be evaluated into the page to easily read and give feedback through the built-in annotation functionality of Google Drive. However, even though Google Drive API allows displaying the internship report integrated into the page, contrary to our expectations, during the implementation phase we learned that Google API doesn't support annotation on integrated pdfs. Similarly, although our system was designed to allow users to view and navigate through folders and their subfolders in an embedded view, it turns out that the Drive API does not provide native support for this functionality. However, we have managed to handle these problems and integrate Google Drive API as closer to our design as we could. This means we had to limit the capability to only read the internship report within an embedded view. For annotation purposes and navigating through subfolders, users are required to use a separate window instead of embedding them. Still, allowing online annotation and having a concrete folder structure that can be archived easily, even if it is in a different window, increases the overall usability of our system.

While trying to design our system, we were faced with certain trade-offs. We had to choose between Usability and Security, in order for our program to be easily usable for new users but these easy-to-use functionalities led to a less secure program (for example, we had

decided to remove the two-factor authentication and “forgot password” features). Moreover, we had decided to prioritize security over automation since several of the documents we handle are confidential (for example, Criteria Reports and Grade Forms). So, we’ve implemented the project so that only authorized people in Bilkent, like the department secretary, can view confidential reports. In addition, all the new users must be signed up by either the department secretary or admin to the system. We assume that this manual sign-up ensures that users will be from the Bilkent Engineering Department, which again decreases the automation of our system. Lastly, We have given priority to usability over maintenance. For example, the system uses the Google Drive API to handle PDF uploading and feedback operations. By leveraging the Drive API, the usability of our system is greatly enhanced, as it offers a plethora of functionalities that are easy to use. However, this also means that our system heavily depends on another system, and if any changes occur in the Google Drive API, our system will also require changes. We are also dependent on Spring and React implementation in a similar manner, which significantly decreases our program’s maintenance, even though it increases our website’s usability. While determining our subsystem decomposition, we have noticed certain aspects that are used by a myriad of users. We have identified those general functionalities and turned them into subsystems for easy implementation and maintenance. Grouping related functionalities to identify, we have broken down the complex system into manageable subsystems. Identifying these subsystems also helped us to divide the project’s workload amongst ourselves more or less equally. Finally, we have used the State Design Pattern in our “StudentCourse” class to handle its changing internal statuses. It has seven states which are: “waiting Summer Training Evaluation From Company”, “waiting Instructor Evaluation”, “upload Revision”, “waiting Final Confirmation”, “grade Satisfactory”, “grade Unsatisfactory”, and “withdrawn”, each of which represents an unique state that StudentCourse objects can be in.

c. Implementation

During the implementation part, we actually believe that we had started at a reasonable time. When we started the implementation properly, there were about 1.5 months until the code deadline. However, it turned out that we needed more time to implement all of our functionality. It was mainly due to unforeseen code errors and the need to re-implement and debug considerable parts of both the program code and the reports. However, we have managed to reach a conclusion. Our team has frontend and backend groups that communicate between them, especially in the meetings. We have discussed and implemented functionalities like Google Drive API, PDF generation and statistics generation. These functionalities brought out a lot of data endpoints and frontend-backend integration. These were partly because of particular compilation and execution order of the React library.

d. What We Did, Did Not, and Why

i. What we did

+ Automatch:

We have added automatch functionality to ease the job of assigning students to instructors for Department Secretaries. When a secretary presses the “Start

Semester” button on their main page, the course matching is done automatically in the backend for the existing students and instructors of the Secretary's department.

+ Create user:

Creating users is a functionality reserved for admins and department secretaries. This functionality allows admins and secretaries to create a single user by supplying the necessary information on their main pages.

+ Create Users from File:

This functionality is created to ease the job of creating users (students, instructors and other secretaries) in bulk. This way, a secretary or an admin can create a lot of users with a single .txt upload that contains the required user data.

+ Auto-generated filled PDFs:

This functionality is actually handled by the Secretary in the current system. However, creating a file is an easily automated functionality. So, we have decided to generate the PDF in our backend structure and store it in a Drive Folder where the Department Secretary and Instructor of the StudentCourse can view and download it.

+ Create Semester:

Creating a semester is a functionality of Admin. Here, Admin creates the essential users for a semester which are mainly Department Secretaries. In order to create and officially start a semester, each department must have at least one Secretary, otherwise Semester does not start.

+ Start semester:

Starting the semester is a functionality of Department Secretaries. Here, the Secretary creates all the instructors and students related to XX299 and XX399. They can either manually enter a single user, or they can upload a .txt file that contains the necessary information to create users in bulk.

+ Drive API integration:

We have used Drive API to ease the internship report management process for all users. Secretaries can easily view the files related to their department and download them. Instructors can easily see the internship reports of their students and give feedback to those reports, fill out Criteria Reports and Grade Forms of their students, all without leaving the website. Students can see their older internship reports and view them.

+ File upload:

We have implemented a “file upload” feature for students to be able to upload to our system, and therefore, related Google Drive API. Initially, the admin creates a semester folder that will start the all related documents of the internship management system of all departments.

ii. What we did not:

- **Notifications:**

Our initial idea for the notifications was to notify users about the event that concerns them in some way. For example, we'd planned for Instructors to be notified when one of their students uploaded a report. However, due to misconception on our part we had realized too late that we need to implement a lot of functionality to be able to properly notify someone (like inboxes and read statuses). Due to our limited time limit, we couldn't manage to implement notifications.

- **Announcements:**

Our initial for announcements was to allow admin and secretaries to create entries in a page. Those entries were meant to be about news and updates related to the whole school or whole departments. However, similar to announcements, we couldn't manage our time well enough to properly create a functional announcements page. Therefore, we had to remove it.

- **Edit student-instructor matchings:**

Our initial idea for editing student-instructor matches was allowing Department Secretaries to change student-instructor pairings when necessary. We were going to implement this functionality mainly to cover the corner case where an instructor is not able to give XX299 and XX399 courses. However, again, due to our limited time and certain hardships in other more crucial functionalities! implementations (like generate PDFs) were not able to create a functioning "edit student-instructor matchings" even though the backend implementation is working.

- **MongoDB:**

We were actually pretty adamant about using MongoDB. However, we had tried for about a week to integrate it into our system without any sort of improvement. Since the errors we got also affected the H2 database (which is what we'd used as a database), we sadly had to remove it and continued to work on the H2 database.

- **Profile Pages:**

Initially we decided to create a profile page for every user in our system where they can see their personal and other important information. However, we'd realized during implementation that there isn't actually enough data to fill out an entire page. So, we'd decided to move the critical information to the user's main page where they can be easily reachable.

- **Deadline extensions and extension request:**

We'd initially decided to implement a functionality to allow students to ask for deadline extension for their internship reports and instructors to change their students deadlines if they saw it fit to do so. However, after talking to several stakeholders of our project we saw that there was no need to implement such a

functionality. Due to this fact and our limited implementation time, we'd decided to remove this functionality from our project.

- **Administrator and TA classes:**

During the requirements stage we identified TA and Administrator as actors of this program. However, during the implementation stage, we'd realized that their functionality is contained in other classes (Instructor and Secretary), so we'd decided to remove them for our system due to our limited time to implement this project.

e. Lessons Learned

Even though we think that we had started early enough, our mock-up designs were noticeably late, and we had to redesign them multiple times. It turned out that we needed more time to completely decide upon UI design than we initially had thought. We learnt that we should decide how each part of the project will be implemented. In our case, occasionally, different members had different opinions about the same functionality. This caused both inconsistency in UI design and cost us a lot of additional time to finally decide upon one implementation.

Also, we had learnt that we needed to specify the workload of each user more clearly. Some features were left without getting assigned to anyone and therefore they got implemented the last. This caused additional hardships since those "left alone" features were sometimes required by other already implemented functionality. This made the testing process noticeably hard. For example, the frontend implementation for the "start semester" functionality of Department Secretaries was not done. Although the backend configurations for starting semester (which includes automatching of students and instructors) were completed and seemingly functional, it took us a lot of time to finally see it integrated to the frontend.

Finally, we learnt that we should start integration of frontend and backend very early on. We have started integration about one week before the code deadline and we still had a hard time properly connecting the frontend and backend. It took way more time than we had previously anticipated.

In the future, we will keep these points in mind in order to create and complete a project way more efficiently and with much less stress.

3. Build Instructions

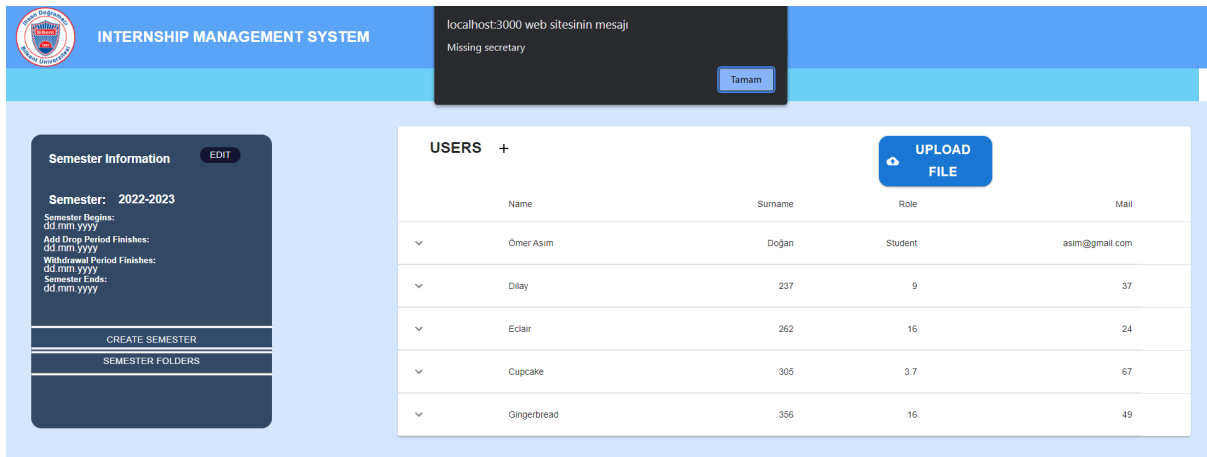
1. Make sure that you have an IDE compatible with the latest versions of Java and React.
2. Clone our repository: <https://github.com/EceAtes/RIGEL.git>
3. Go to the path /src/java/com/example/rigel_v1 from the cloned project in your IDE.
4. Wait for the Maven dependencies to finish.
5. Make sure that credentials.json file is located in main/resources, which contains the necessary tokens and keys required for the Google Drive connection.
6. Run the project, this action will initiate the backend of the project.
7. In terminal of your IDE, go to the directory of your frontend project folder.
8. On the terminal, run the command "npm install" to have all necessary React dependencies. Besides that, axios library should be installed with "npm install axios"

command, material-ui libraries with “npm install @mui/material @emotion/react @emotion/styled” command.

9. After installation is successfully completed, run the “npm start” command. Please note that npm commands are only available if Node.js is installed on your system (<https://nodejs.org/en>)
10. If everything works fine, you will end up with an introduction page to our system.
11. Initially, only admin is on the h2-database. One can initiate the system by clicking the “create semester” button using the admin credentials in BootstrapData.java.
12. We advise you to run front-end and back-end codes in separate workspaces to avoid any possible runtime errors.

4. User's Guide

4.1 Admin's Guide

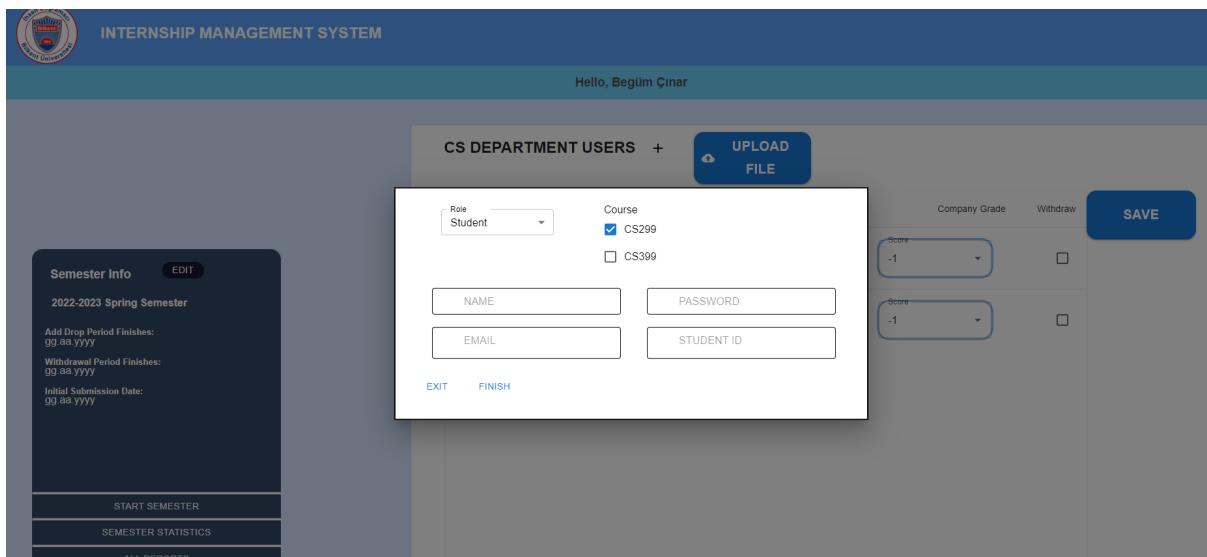
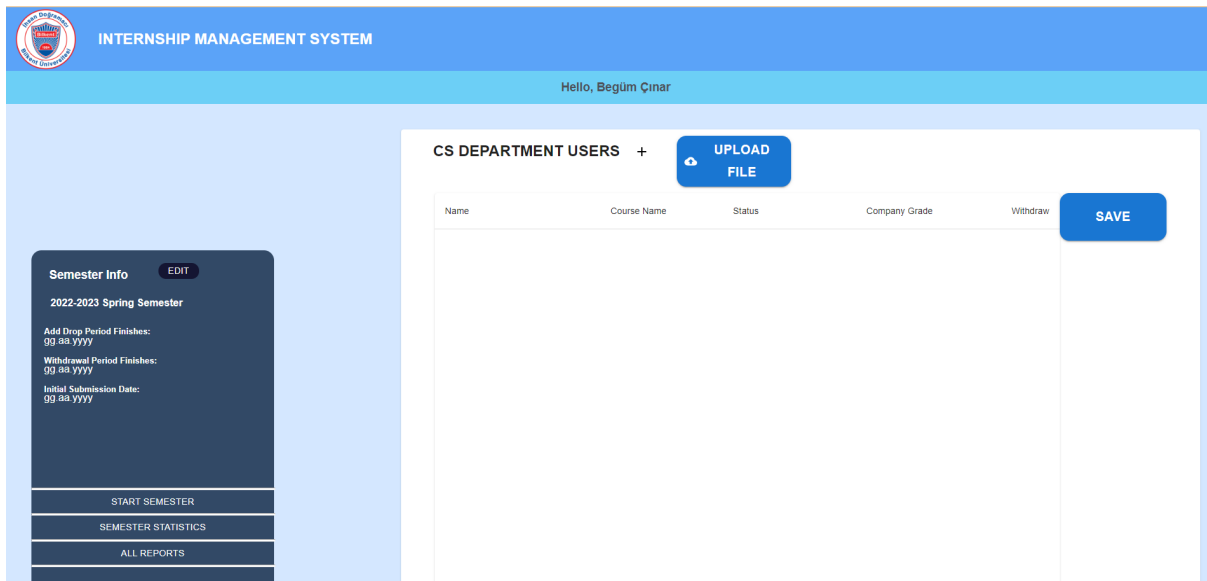


The screenshot shows the Admin Dashboard of the Internship Management System. The top header includes the system logo, the title "INTERNSHIP MANAGEMENT SYSTEM", and a notification area displaying "localhost:3000 web sitesinin mesaji" and "Missing secretary" with a "Tamam" button. The left sidebar contains a "Semester Information" panel with an "EDIT" button, showing details for the 2022-2023 semester, and buttons for "CREATE SEMESTER" and "SEMESTER FOLDERS". The main content area features a "USERS" section with a "+" button and an "UPLOAD FILE" button. Below these is a table listing users with columns for Name, Surname, Role, and Mail.

	Name	Surname	Role	Mail
✓	Omer Asim	Doğan	Student	asim@gmail.com
✓	Dilay	237	9	37
✓	Eclair	262	16	24
✓	Cupcake	305	3.7	67
✓	Gingerbread	356	16	49

- Admin logs into the system with their credentials provided in Bootstrap.
- Creates semester by entering appropriate dates for start, add-drop, withdrawal and end of semester. The conditions checked for dates are whether start comes earlier than ending semester as well as add-drop and withdrawal dates fall between these dates.
- He needs to have created at least 4 secretary users for each department that creates folders for each department. Then the semester folder that encapsulates all departments is created by creating the semester successfully.
- Admin can create these users manually or by uploading a .txt file with appropriate data for creation of users.

4.2 Secretary's Guide



- Logs in with the email and password generated in “User Creation” by admin or other secretaries.
- Starts the semester with the “start semester” button, this action also automatches the students with instructors. To start the semester, there should be at least one instructor and one student user.
- They can see their department’s students and instructors on the right table on their main page.
- They can generate users by either manually entering new user’s information to the pop-up which is generated by clicking on the plus button (next to “USERS” headline of the table) or uploading a .txt file that includes the necessary information about user creation.
- “All Reports” button redirects the Secretary user to the Drive Folder that contains all the reports that are related to their department.

4.3 Instructor’s Guide

- They can see all of their students and the status of those students’ courses.

- Since the buttons of each status are different, instructors can easily understand what they are supposed to do next.
- “Evaluate” button redirects the Instructor user to our Google Drive Integration. There they can decide whether or not that report needs feedback, and they can give general feedback related to the whole of the report (to give a general idea to students) on the right side of the page. If they click on the button located just top left of the internship report, they will be redirected to Google API’s annotate interface. There they can give extensive feedback to the internship report, which will be visible to the owner student.
- “See report” button redirects the Instructor user to the report they’ve added annotations. It is implemented to give the instructor a more plain environment to read the reports.
- “Criteria Mode” button redirects Instructor users to our Google Drive integration. Here, they can fill out required criteria report questions by supplying each question an answer and a score. They can either save or submit this report. If they choose to save, the criteria report will still be changeable by the instructor user. However, choosing the submit button will make the criteria unchangeable.
- “See all reports” button redirects the Instructor user to the Drive Folder where they can reach all of the reports that are related to them.

4.4 Student’s Guide

- After logging in with their credentials, it is assumed that they will be able to login to the system with their webmail address and STARS password.
- Student Main Page is the page where they can upload their reports by clicking on the indicated symbols located on the blocks that represent their courses located to the right side of the page. They can also see their courses’ status, access their feedback and their old reports from those blocks.

5. Work Allocation

a. Aytekin İsmail

Analysis Report

In the first iteration, I have written the introduction, current system sections, non-functional requirements and pseudo-requirements. I have also written 5-6 use cases in the report. In the mock ups, I have created the main pages of users, and submit report pages for students. I have created the logo of the group and the colors of the logo are then used in implementation as themes. In the second iteration, I have updated the use cases according to our current progress in the implementation and rewritten the non-functional requirements according to the feedback. I have made the revision activity diagram.

Design Report

In the first iteration, I have written the introduction, current system sections and design goals. I have written design trade-offs and design patterns. In the second

iteration, I have updated the design goals and trade-offs according to feedback. I have added an improvement summary to the report.

Implementation

I was responsible for the front end part of the project. I have designed the introduction page (the first page that users see), login page and instructor main page. I have implemented all the back end functionalities that Ece and Begüm have written of the corresponding pages to make it responsive. I added h2-console connection to the secretary and admin main pages so that they can create users within the system. I have combined all the front-end pages that İzgi has implemented(student functionalities) with my instructor functionalities and the system works as a whole. When the student uploads an internship report, this change will be reflected on both student and instructor, therefore, the instructor can start evaluating the report.

b. Ece Ateş

Analysis Report

For its first iteration I've worked on "Overview" and "Non-functional Requirements". I've also completed 2-3 use cases. I was also responsible for the "Object and Class Model" and its class descriptions. Finally, I was responsible for "Activity Diagram 3: Student Assignment Activity Diagram" and "Activity Diagram 6: Autogenerate Form Activity Diagram". For the second iteration, I've edited "Object and Class Model" and its description thoroughly, worked on the assignment format, and edited about 15 use cases. I also wrote the majority of the final report.

Design Report

For the first iteration, I've worked on "Hardware / Software Mapping" and its diagram, "Persistent Data Management", and "Data Management Layer: Entity Diagram". I've also written the "Glossary". For the second iteration, I've updated the two diagrams I'd worked on and re-wrote the "Object Design Trade-offs" and "Design Patterns" parts.

Implementation

For the implementation, I've mainly worked on back-end structure. I was responsible for the JSON request handling. So, I've implemented a large part of the controllers (I have only written a small part of the PDF controller, and Begüm has worked on GoogleDriveAPI integration). I had implemented domain classes and the relations between them. I've implemented the secretary's "automatch", "rematch", and "createUserFromFile" functions. I've also worked on the integration of the frontend and backend of the project.

c. İzgi Nur Tamcı

Analysis Report

For its first iteration, I was responsible for drawing the Use Case diagram. We came up with the functionalities in our first few meetings. I edited the diagram and use case titles according to my teammates' opinions. Then we all wrote the

explanations for use cases. I took on two or three explanations. I also designed the instructor's main page and evaluation page in the first iteration mock-up. I was also responsible for the "Archive Activity Diagram".

Design Report

I was responsible for Boundary Conditions and Program Lifecycle Diagram. For this part, I did research on network, authentication and how a program and our program should initialize checking the boundary condition and terminate gracefully in case of an extreme situation.

Implementation

I have implemented Student Main Page, Feedback Report Page, Criteria Report Page, and these pages' functionalities. As functionalities imply, I also integrated these pages' front end with our H2 backend console.

d. Ömer Asım Doğan

Analysis Report

I have written some of the use cases for the mock-ups we prepared. I designed the opening page, login page, statistics page, some pages of the secretary, all the pop-ups, and some other pages all around the system. In the second iteration, I made sure all the parts that were given feedback are fixed and upgraded according to the feedback.

Design Report

The final object design and user interface management layer are designed by me. I wrote one part of the design trade-offs and created the use-case actor relation table. TA never gave feedback on any of my parts so I just cleared the use case diagram in the visual paradigm for the time being.

Implementation

I have written the code for some pages in our Analysis report but at the halftime of implementation, our system design is changed. Then I wrote the code for the secretary's main page and the admin's main page. I tried to connect with the backend but the last part is mainly completed by Aytekin. Also, I have written some pop-ups in the system.

e. Zeynep Begüm Kara

Analysis Report

In both two iterations, I worked on use case model improvements and a few use case descriptions. I drew seven state diagrams and two activity diagrams. I redesign the navigational paths and screen mock-ups design for the second iteration.

Design Report

I was responsible for subsystem decomposition for the design report. Hence, distributed each actor to each of my friends and with their help I concluded the subsystem diagram and explained each I also draw Service Management Layer with the help of Ece.

Implementation

For the implementation, I've mainly worked on back-end structure. I was responsible for the file management. Hence, I worked on Google Drive API to both upload and annotate files. I also managed folder hierarchy in Drive, which includes creating semester folders, department folders, and student folders. Also, I worked on the automatic PDF generation of the Summer Training Grade form with Ece. Additionally, I've also worked on the integration of the front-end and back-end of the project.

I was also leading the group meetings and summarizing the meeting decisions after each meeting to have more efficient meetings (see ToplantıNotları.txt in GitHub).

6. Glossary

Grade Form: Grade Form is the main form where each student's XX299 or XX399 grade is determined. This report is filled once when the student's grade is determined (as "satisfactory" or "unsatisfactory")

Internship Report: The internship report is the report students prepare about their internship experience. Students get feedback from TAs and their instructor for this report if it doesn't match the "satisfactory" grade requirements.

Criteria Report: Criteria Report is filled as a part of ABET Accreditation. Instructors fill this report to show evidence of the "satisfactory" status of the student's internship report.

Evaluation Form: The evaluation Form is based on the company where the student had completed their internship. The company's executive evaluates the student, which influences the student's grade. The company is also evaluated on certain criteria, for example, whether the student's supervisor was an engineer.

StudentCourse Class: StudentCourse class is where the student's internship-related reports are held. It also has the student's current status and deadlines. The system automatically assigns deadlines for each revision, but students can ask for extensions. It has multiple states related to the status of the student's progress.

7. References

[1] "Bilkent University." [Online]. Available: https://w3.bilkent.edu.tr/web/kalite_guvencesi/faaliyet_raporu_2016-2017.pdf. [Accessed: 29-March-2023].