Bilkent University
Department of Computer Engineering

CS319 Object-Oriented Software Engineering

Internship Management System

Project Analysis Report

Ahmet Alperen Yılmazyıldız 22002712
Ahmet Berke Gökmen 22002105
Erdem Eren Çağlar 22002359
Mahmut Mert Gençtürk 22003506
Cem Apaydın 21802270

Instructor: Eray Tüzün
Teaching Assistant(s): Yahya Elnouby, Muhammad Umair Ahmed and Tolga Özgün

First Iteration

March 30, 2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the CS319

Contents

Project Analysis Report

Internship Management System

1 Introduction

Our project constitutes an internship management system that will allow the students, secretaries, teaching assistants, and evaluators to conduct their necessary operations more easily than the currently used system. We, as the team of this application, want to provide an application that the users will spend less time and effort on throughout this process. We endeavor to expedite the current system by eliminating any unnecessary by-hand operations.

The students will be able to track their submission’s status and submit their (revisioned) reports and comments about what they have changed via the application. The students won’t need any more agent or effort to find out their submission’s situation since the status check we offer will inform the student about any revisions and their submission’s progress until completion. The evaluators and the teaching assistants will be able to track the forms that are assigned to them. The teaching assistants will be able to check the reports before the evaluators and request changes. The evaluators will have more authority over these operations compared to the assistants. The evaluators will be able to grade the students’ reports, request revisions, add comments to the reports, and assign and change submission deadlines. The admins or secretaries will have the most authority over the application. They will be able to track each operation managed through this application and manipulate the process by adjusting the system's management, such as the deadlines, assigned evaluators and teaching assistants, semester initialization, and deactivation.

As detailed in the following section, the current system requires many redundant exertions that each individual taking part in this process has been going through for many years. This internship management system will enable the existing mechanism to actuate more efficiently and spare all individuals from this burden, especially the secretaries.

2 Current System

The stakeholder of this project, Selim Aksoy, presented how the internship reports are handled in the current system. In the current system, the internship reports are managed in a decentralized way. There is no inclusive platform that contains all of the internship management features. The current internship management is done in 3 different platforms: Moodle, Google Drive, and lastly, e-mail. The main steps of internship management are listed as this:

1. Students are required to upload their internship reports to Moodle until an established deadline.

2. Summer training evaluation forms filled by companies collected by the faculty secretary.

3. After this, the internship reports are pre-reviewed by the teaching assistants and according to their feedback, students may need to submit their reports again to Moodle.

4. Faculty secretary collects these reports from Moodle and uploads them to Google Drive for faculty members' evaluation alongside the company evaluation forms of the students.

5. Faculty members are assigned to the reports as evaluators by the faculty secretary. Assignments of faculty members are done according to the ranking of the student's name and surname, as Selim Aksoy indicated in the presentation. Also, the faculty secretary can arrange the assignment of evaluators according to how many students a faculty member will evaluate. Assigned faculty members find which student's internship report to evaluate from the Google Drive folders that the faculty secretary creates in the 4th step.

6. After the evaluation, faculty members may require revision from the student by indicating which changes they should make before proceeding to the report's final version. Faculty members may require multiple revisions until they find the report satisfactory for final assessment. Evaluators send an email explaining what changes students should make, or students can view the feedback in Google Drive. Google Drive links are made available to the students on a website specifically designed to announce the grades. The website contains a simple table with student IDs and corresponding letter grades. If the faculty member requires revision, they put the Google Drive link in the grade section. Since this website is public, students can see not only their grades or report status but also other students' grades and report status. Moreover, they can reach other students' feedback by clicking the Google Drive link, which is causing privacy issues on a large scale. Also, it is impractical for the students to find their internship report status in the pile of student IDs.

7. Students re-submit their report with an explanation of the changes they made by sending an e-mail to their evaluators by the established deadline.

8. Faculty members assess the final version of the reports by filling out a form about the quality of the internship report. By combining the grade of the form with the company evaluation form, the evaluator grades the overall internship report as satisfactory or unsatisfactory.

9. The grades are announced on a website, which is mentioned in the 6th step. As mentioned before, in the 6th step, students need to find their student ID and grades from the table, which is not a practical way of

learning grades. Moreover, students can also see other students' grades which violates privacy of the students.

As can be seen, these processes are made on independent platforms. Since there is no built-in connection between these platforms, keeping track of the progress may take a lot of work for all parties. For example, faculty members cannot track how many internship reports are left to evaluate, and the faculty secretary needs to send the summer internship reports by hand. Students cannot see the progress of their internship reports properly. Also, revision requirements made by faculty members and re-submission of the students were made manually through email. These issues cause unnecessary delays in the process, making internship management difficult for all parties.

3 Proposed System

3.1 Overview

The current system uses manual methods, which needs to be more practical and results in excessive time loss for students and academic staff. Manual paperwork should take little to no time. Therefore we propose to create a browser-based system that will benefit all parties involved in the internship report evaluation cycle.

Our system will automatize the processes currently done by hand (see section 2). Initially, the users of the application will be able to log in to the system by using the credentials that are automatically created by the super admin upon each semester start. Then, depending on the user's role, they will be shown a dashboard to access their related work. For instance, a student will be able to see their applications, track their reports' statuses, and resubmit their revised report if they receive a change request from either assistants or evaluators. Evaluators will see what reports they need to evaluate, will be able to request changes by explaining what is missing or needs to be changed, or approve the report and assign grades to each evaluation criterion. Assistants, just like evaluators, will be able to request changes by providing what needs to be changed on the reports.

Most importantly, the system will automate the process of semester and class tracking; that is, the secretaries (admins) will no longer have to deal with each student separately. They will get an excel sheet exported from STARS containing who is taking a specific internship course this semester. Then, upon uploading the excel file to the system, all the users' accounts will be automatically created (if they have not taken any course before), and they will get a notification email saying that "Your account has been registered to X semester, Y course. You can view your applications on your dashboard". Moreover, the admins will be able to add individuals to each semester course, such as assistants who joined later or evaluators. Furthermore, the secretaries will no longer have to direct company evaluation forms to evaluators by hand. The system will send an automatic email to the student's supervisor with a

one-time link, and the supervisor will be able to provide feedback using the system. However, if they want, secretaries can still get the grades in an enclosed letter. Then, they can upload the evaluation report to the system so that it can be viewed by faculty members while grading the work. Lastly, they will be able to set deadlines, send email notifications to students, and change which evaluator and assistant is assigned to which report if required. At the end of each semester, the admins will be able to deactivate a semester course and export all students' grades alongside their required information as an excel sheet.

In summary, our web-based application provides a new automated way of dealing with internship reports and combines everything into one place. It offers solutions for all parties and aims to reduce the time they spend during the internship report evaluation process.

3.2 Actors

As a result of the requirement analysis, we have identified four types of users that will be using the internship management system.

- **User**: User is the most basic type of user in the system. It encapsulates all other user roles and provides basic access to the system. Such as logging in & out, resetting passwords, requesting password reset link and getting notifications about new updates.

- **Student**: Student users are created by admins (secretaries) upon semester initialization. They cannot register to the system on their own; their accounts are automatically created when the admin starts the semester with a specific course (e.g., 2022-2023-spring CS299). Students in the system can view their registered semesters and internship report status. Later, they can create submissions in which they provide the necessary information about their internship, such as company, supervisor email, and start and end dates. They can upload their reports upon submission, and according to the assistant’s and evaluator’s response to the report, they upload a revised version of their report. Finally, students can choose whether to hand in the company evaluation letter, which can then be submitted to the system by admins. Alternatively, they can send a one-time link containing the online evaluation form for the student.

- **Assistant**: Assistants are responsible for pre-checks (grammar, spelling, Turnitin) on reports before evaluators can view them. They can request changes in the report and approve them.

- **Evaluator**: Evaluators are responsible for evaluating work and reports in a submission. They can request changes in the report and approve them. Furthermore, they can assign specific points to each field in work and report evaluation tables.

- **Admin**: Admins are also known as secretaries. Admin users' capabilities are broad. They can initialize a semester with a course and create accounts for each user. After initialization, if required, they can add additional people (TAs, evaluators, other students) to an existing semester so that they can start working on reports. Moreover, they can register courses (such as CS299 and CS399), allowing for different courses to be dynamically recognized by the system. Furthermore, they have the required role of viewing and editing all the reports, submissions, and semesters in the system. If needed, they can search for users by using their names to get all the information about them. At the end of each semester, admins can deactivate that specific semester and get an excel report of students and corresponding grades.

3.3 Non-functional Requirements

Usability

As the project is designed to ease the report evaluation procedure, it has to provide a clear, unique, and easy-to-interact interface. It should provide easy access to all the frequently used features, such as showing reports and their statuses for students, reports waiting to be graded to assistants and evaluators, and lastly, initializing a semester with a group of students should be as easy as possible.

- Uploading, downloading, and evaluating a report will be as simple as clicking a button.
- The colors of the reports will match their status. Red if it is rejected and requires changes, orange if it's waiting for review, and green if approved.
- Responsive design approaches will be used; the system should easily be used via any device. The website should be accessible and easily operated through a mobile phone with a small screen.

Reliability

Regarding reliability, the system should ensure that it will not lose any data and no reports will get stuck; that is, they can neither be re-uploaded nor evaluated. The system's functionalities should be solid as a rock and function as expected. All the end user expectations from the system should be satisfied.

- Since all of the components of the system (Database, backend, frontend, file storage, etc.) will be deployed to a cloud provider such as AWS, it will be ensured that the system will be up and running 99% of the time.

- Since the system must not lose any historical data for legal purposes, automatic database archiving will be performed monthly.

Security / Safety

Since the system will store sensitive user information (email, name, internship data, etc.), it should be protected against database attacks such as No-SQL injections. Also, the system should ensure that no student can see another's reports, status, or progress. Moreover, some people should only be able to access resources or view pages if they have the required roles.

- To protect privacy, only limited information about the students will be visible to other parties (students, assistants, evaluators).
- The passwords of all the users will not be saved as plain text in the database. They will be hashed using Bcrypt, and the hashed version will be saved into the database.
- There will be no register option; only admins can create student profiles upon the semester beginning and use the related excel file exported from STARS. This ensures that no unnecessary user is registered into the system.
- One student can only have one account, which will be created and tied to their Bilkent id and email.

Performance

The system should ensure that no operation takes too much time. All stakeholders should be pleased with the system and not get frustrated.

- Navigation between pages will be almost instant since the system will use dynamic page generation.
- Some frequently accessed resources (such as announcements) will be cached using an in-memory database for faster response times.

Maintainability

With the design patterns and state-of-the-art techniques used in the system, it will be easily extendable and improved.

- Since the system will be built on top of OOP (Object Oriented Programming) principles, it will be easy for developers to utilize features like encapsulation and polymorphism.
- All backend endpoints' documentation will be available through the Swagger API documentation tool to ease testing and frontend development & maintenance.

3.4 Pseudo Requirements

A numbered list is below:

1. The project must be a web based application.
2. The project must be implemented in an Object-Oriented programming language and use an Object-Oriented design pattern.

3.5 System Models

3.5.1 Use-Case Model

High resolution version of the use case diagram can be seen here.

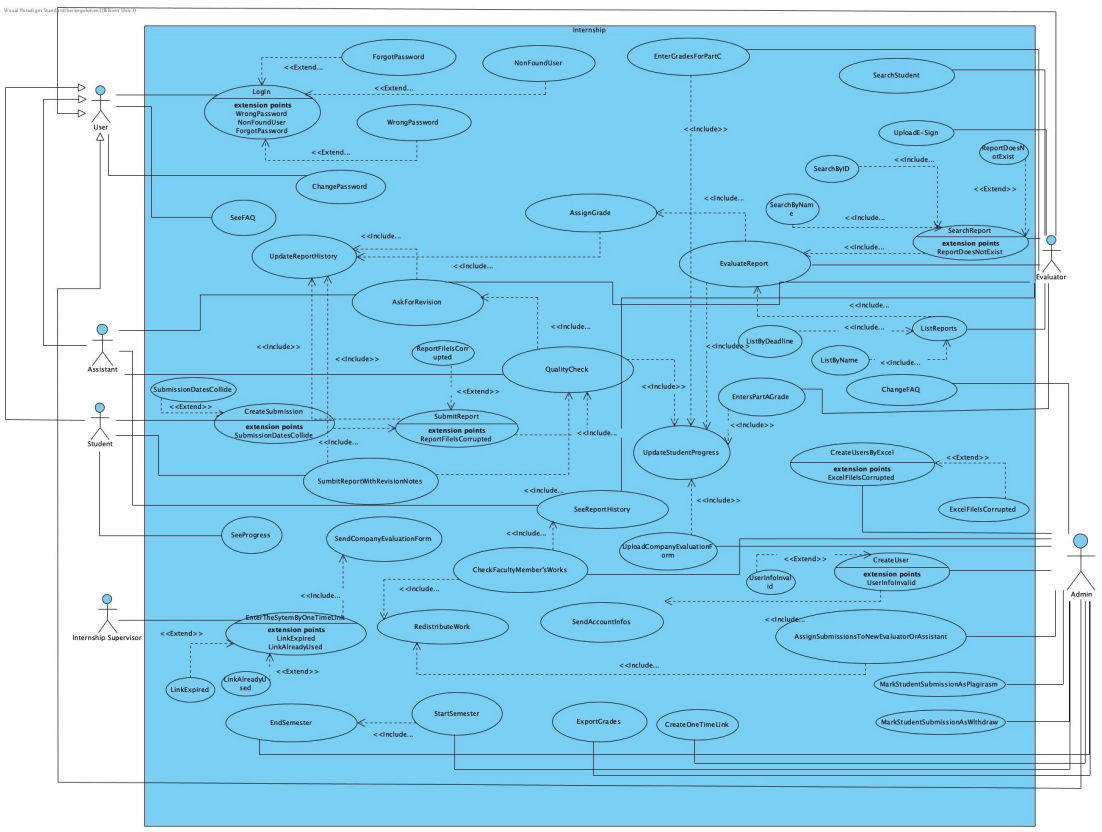

Figure 1: Use Case Diagram for the Internship Management System

---

Use case name: LogIn

Participating actors: Initiated by User

Flow of events:

1. The User enters their account information (Bilkent ID, password) to the login form of the system.

2. The User activates the login procedure by clicking the sign in button below the form.

3. IMS takes the information from the form to evaluate the login credentials.

Entry condition: The user does not have a valid token.

Exit condition: **.** The User granted with an valid token; **OR**

. The process was canceled because of an error.

Use case name: ForgotPassword

Participating actors: Initiated by User

Flow of events:

1. The User sends a request for the change his/her password by clicking forgotPassword button and entering their email.

2. IMS sends verification code to the requested account’s email to verify the request's validation.

3. User clicks the link generated by the system that was sent to his/her email, to the system.

4. IMS verifies the code and generates a form for the new password of the user.

5. The User enters the new password which will be used as their new password in the system. Then confirm it and activate the changePassword function by clicking the submit button.

Entry condition: . Entered id by the user must be present in the system and link with an active account, OR

. The code in the link should be valid UUID and should not be expired or used.

Exit condition: . The User’s password successfully changed, **OR**

. Process exterminated because of an error at the process of code verification. The code may already be used or expired.

Use case name: WrongPassword

Participating actors: Initiated by User

Flow of events:

1. User enters his/her account information wrong five times in a row.

2. IMS applies a lock on the IP address so that the User cannot submit the login form again and has to wait one minute.

Entry condition: . User's id was found in the system however the entered password was not correct.

Exit condition: . User waits one minute.

Use case name: NonFounduser

Participating actors: Initiated by User

Flow of events:

1. User enters its account information to the login form

2. IMS could not find the entered id in the system database five times in a row.

3. IMS applies a lock on the IP address so that the User cannot submit the login form again and has to wait one minute.

Entry condition: . Enter the wrong id to the system three times in a row by the User.

Exit condition: . Successful login or lock on the IP address expires.

Use case name: ChangePassword

Participating actors: Initiated by User

Flow of events:

1. User clicks the change password button to generate the passwordChange function on her/his terminal.

2. User enters a new password. After that User clicks the submit button.

3. IMS takes the password information and changes it with the old one in the database after the request.

Entry condition: . User should successfully login to the system and has a valid token.

. Users should enter the account Settings page (Change password page).

Exit condition: . User successfully changes their password.

Use case name: Create Submission

Participating actors: Initiated by Student

Flow of events:

1. Student enters his/her internship information (start date, end date, etc.) and internship supervisor's information (email) to the form alongside their report and initiate the createSubmission function by clicking the submit button.

2. IMS checks the end date and start date correlation of the internship and supervisor's email eligibility. Also checks if the new submission dates collide with any existing submission.

3. IMS creates new submission for the internship report to the Student's account.

Entry condition: . Student should successfully login to the system and have a valid token throughout the process AND

. Students should not have any ongoing or done submission on the system registered to the same lecture in current or previous semesters.

Exit condition: . Student successfully register his/her submission to the system **OR**

. Process canceled out because of an error.

Use case name: Submit Report

Participating actors: Communicates with Student

Flow of events:

1. Student upload his/her internship report to the system

2. IMS checks for the file size limit then activates the submit button.

3. Student clicks the submit button and activates the uploadReport function on his/her terminal.

4. IMS saves the uploaded report to the database and links it to Student's submission then activates updateReportHistory use case.

Entry condition: . This use case **includes** the createSubmission use case. It is initiated by the system and requires a registered submission in the system from that Student.

Exit condition: . Student successfully uploaded his/her report to the system **OR**

. The process was canceled because the uploaded file exceeded the limit.

Use case name: SubmitReportWithRevisionNotes

Participating actors: Communicates with Student

Flow of events:

1. Student uploads revised internship report to the corresponding field.

2. Student uploads revision notes to the corresponding field

3. IMS checks for the file size limit then activates the submit button.

4. Student clicks the submit button and activates the uploadReportWithRevision function on his/her terminal.

5. IMS saves the uploaded revised report and revision notes to the database and links it to Student's submission then activates updateReportHistory use case.

Entry condition: . This use case **includes** the askForRevision use case. It is initiated by the system and requires a registered submission and report in the system from that Student, AND

. The previous version of the report should not get approval from the Evaluator or Assistant.

Exit condition: . Student successfully uploaded his/her report and revision notes to the system **OR**

. The process was canceled because the uploaded files exceeded the limit.

Use case name: SeeProgress

Participating actors: Communicates with Student

Flow of events:

1. Student navigates to the dashboard and activates the progress function of his/her submission on his/her terminal.

2. IMS shows the current state of the submission, state history of the submission with dates, and next state submission should be achieved to get approval.

Entry condition: . Student should have successfully login to the system and have a valid token throughout the process, **AND**

. Student should have a submission on the system.

Exit condition: . Student sees his/her submission's progress.

Use case name: SeeFAQ

Participating actors: Communicates with User

Flow of events:

1. Student clicks the seeFAQ button on his/her dashboard and activates the FAQ function on his/her terminal.

2. IMS shows the current FAQ and their answers written by the Admin.

3. Student propose new questions to the system by writing it to the empty label.

4. IMS takes the proposed question and sends it to the Admin to get approval to show it on this page.

Entry condition: . Student should have successfully logged in to the system and have a valid token throughout the process.

Exit condition: . Student successfully see the FAQ registered and approved to system, **OR**

. Student successfully proposed new questions for the system.

Use case name: QualityCheck

Participating actors: Communicates with Assistant

Flow of events:

1. Assistant downloads the assigned Student's report by the system.

2. IMS activates the updateReportHistory function to change the report's state.

3. IMS creates a form for evaluating the report.

4. In the case of an unsuccessful report Assistant activates askForRevision use case by requesting changes.

5. IMS activates the updateReportHistory use case to change the report's state.

Entry condition: . This use case **includes** the submitReport and submitReportWithRevisionNotes use cases. Therefore, this use case requires a submitted report from the Student with a state of "unchecked". This use case is initiated by the system.

Exit condition: . Assistant successfully evaluates the Student's report and changes its state in the system.

Use case name: AskForRevision

Participating actors: Communicates with Evaluator

Flow of events:

1. Assistant or Evaluator upload feedback notes to the system to indicate what should change in the revised version of the Student's internship report.

2. IMS checks the size limit of the uploaded file and activates the submit report.

3. Evaluator or Assistant decides a new deadline for the revised report.

4. Assistant or Evaluator clicks submit button and activates giveFeedback function on his/her terminal.

5. IMS saves the feedback notes to the system and links it to the Student's report.

6. IMS activates UpdateReportHistory use case to change the report's state.

Entry condition: . This use case **includes** the qualityCheck and evaluateReport use cases. Therefore, it requires a submitted report from the Student and initiated by the system.

Exit condition: . Assistant or Evaluator successfully gives feedback to the Student's report and change its state to "askForRevision", **OR**

. The process was canceled because the uploaded file exceeded the size limit.

Use case name: EnterTheSystemByOneTimeLink

Participating actors: Initiated by InternshipSupervisor

Flow of events:

1. InternsipSupervisor enters the IMS with the one time link sent by admin.

2. IMS shows the linked Student's information with the sent link to the InternshipSupervisor and generates a label to get confirmation from she/he that has acknowledged him/her internship.

3. InternshipSupervisor fills the confirm label to confirm that he/she is associated with linked Student's internship and click next button.

Entry condition: . InternshipSupervisor should have the one time link to enter the system, **AND**

. The link associated with Part A form should be generated and confirmed by the Admin before it was sent.

Exit condition: . InternshipSupervisor successfully pass on to Part A grade form page, **OR**

. The process was canceled because the InternshipSupervisor did not give confirmation about the associated Students internship.

Use case name: EnterCompanyEvaluationForm

Participating actors: Communicates with InternshipSupervisor

Flow of events:

1. InternshipSupervisor clicks the enterGradeForInternship button to generate company evaluation form function on his/her terminal.

2. IMS generates a company evaluation form on the page and a notification popup to inform InternsipSupervisor about Bilkent's passing grade policy.

3. InternshipSupervisor enters the grade of the associated Students internship.

4. After all the labels are filled, IMS activates the submit button.

5. InternshipSupervisor clicks the submit button.

6. IMS generates reports from entered grades in PDF format and sends it to Admin.

Entry condition: . This use case is activated after the EnterTheSystemByOneTimeLink use case. This use case is initiated by the system.

Exit condition: . InternshipSupervisor successfully filled the company evaluation form for the associated student.

Use case name: SearchStudent

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator searches a Student from the search label by his name from that list.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator sees the Student's information.

Use case name: EnterGradesForPartC

Participating actors: Communicates with Evaluator

Flow of events:

1. Evaluator sees the part B grade of the selected Students report assigned to him/her.

2. Evaluator enters the grades for the part C to the form.

3. IMS checks if any field remains blank and activates the submit button.

4. Evaluator clicks to submit button to generate passingNotification function on his/her terminal.

5. IMS generates a notification box to inform the Evaluator about bilkent passing grades and activates the "I Approve" button for the entered grades.

6. Evaluator clicks the "I Approve" button to generate the enterPartC function on the system.

7. IMS saves the entered grades to the associated Students submission and generates updateStudentProgress to change the submission's state.

Entry condition: . This use case extends from the SearchStudent use case. This use case is initiated by the system.

Exit condition: . Evaluator successfully entered part C grades of the associated Student, **OR**

. The process is canceled and filled form saves as an "ongoing evaluation" but does not update the Students process because Evaluator's exit the system without clicking the "I Approve" button.

Use case name: UploadE-Sign

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator clicks the "Upload E-Sign" button on his dashboard to generate eSignUpdate on his/her terminal.

2. IMS redirect Evaluator to a page which she/he could upload his/her e-sign in a pdf format.

3. Evaluator uploads his/her e-sign to the system.

4. IMS checks uploaded files not exceeding the size limit.

5. IMS saves the uploaded e-sign to the system and links it to the associated evaluator.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully uploads the e-sign of his/her to the IMS, **OR**

. The process was canceled because the uploaded file exceeded the size limit.

Use case name: SearchByID

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator sees the list of students assigned to he/she who uploaded a report which gets approval from the Assistant about spelling and grammar.

2. Evaluator selects the id search function from the searchOptions radio button and enters an id.

3. IMS narrows the list by taking out the Students report from the list whose id does not contain entered number combination.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully finds the target Student's report on the list.

Use case name: SearchByName

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator sees the list of students assigned to he/she who uploaded a report which gets approval from the Assistant about spelling and grammar.

2. Evaluator selects the name search function from the searchOptions radio button and enters a name or surname.

3. IMS narrows the list by taking out the Students report from the list whose name and surname does not contain entered char combination.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully finds the target Student's report on the list.

Use case name: SearchReport

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator clicks the PartB button to generate a listReports function on his/her terminal.

2. IMS lists all the reports which get approval from the Assistant about grammar and spelling, of Students who are assigned to that Evaluator.

3. Evaluator can generate SearchByID or SearchByName use cases by clicking the search label.

4. IMS shows the report of the Student according to the selected search key.

5. Evaluator selects a report's clicks the evaluate button to generate partBEvaluate function on his/her terminal.

6. IMS redirects the Evaluator to the evaluation page of the selected report.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully enters the target Student report's evaluation page.

Use case name: EvaluateReport

Participating actors: Communicates with Evaluator

Flow of events:

1. Evaluator can download the Student's internship report and if it is second or more iteration, revision notes and the report history.

2. IMS shows the criteria of the evaluation after the downloading process ends.

3. After the Evaluator finishes his evaluation process she/he can click the assignGrade button or askRevision button to generate assignGrade use case or askForRevision use case.

4. According to the decision of the Evaluator IMS redirect the Evaluator to askForRevision or PartBGradeForm page and generate updateStudentProgress use case to change the Student's submission history.

Entry condition: . This use case extends from the SearchReport and ListReport use cases. This use case is initiated by the system.

Exit condition: . Evaluator successfully evaluates the selected Student report's and redirects to another page according to his/her decision about the report, **OR**

. The process was canceled because the Evaluator exit from the system without making a decision about the Student's internship report and Student' process remains unchanged.

Use case name: AssignGrade

Participating actors: Communicates with Evaluator

Flow of events:

1. Evaluator enters the grades according to the part b criteria to the form generated by the system.

2. IMS checks no label remains blank and generates the submit button.

3. Evaluator clicks the submit button and generates enterPartBGrade function on his/her terminal.

4. IMS creates a notification box about Bilkent's grade policy and generates an "I Approve" button to get confirmation from the Evaluator.

5. Evaluator clicks to the "I Approve" button.

6. If the entered grades are below the benchmark IMS creates a notification box to inform the Evaluator that Student will get "U" grade from the registered lesson and generates a "I Approve" button**.**

7. Evaluator clicks the "I Approve" button.

8. IMS saves the entered grades to the associated Student's report and generates updateReportHistory and updateStudentProgress use cases and redirect Evaluator to the dashboard page.

Entry condition: . This use case extends from the EvaluateReport use case. This use case is initiated by the system.

Exit condition: . Evaluator successfully entered part B grade for the selected Student's report, **OR**

. The process is canceled because Evaluator terminates the process without clicking the "I Approve" button and gives confirmation about grade policy.

Use case name: ListReport

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator navigates to the pending reports page.

2. IMS lists all the reports which get approval from the Assistant about grammar and spelling, of Students who are assigned to that Evaluator.

3. Evaluator can generate ListByDeadline or ListByName use cases by clicking the list label.

4. IMS shows the reports of the Student according to the selected list key.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully sees all the reports assigned to him and that are pending.

Use case name: ListByDeadline

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator sees the list of students assigned to he/she who uploaded a report which gets approval from the Assistant about spelling and grammar.

2. Evaluator selects the deadline list function from the listOptions radio button.

3. IMS reorder the list by the line up the Students report from which the deadline is closer to the current time.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully sort the list of reports as his/her selection

---

Use case name: ListByName

Participating actors: Initiated by Evaluator

Flow of events:

1. Evaluator sees the list of students assigned to he/she who uploaded a report which gets approval from the Assistant about spelling and grammar.

2. Evaluator selects the name list function from the listOptions radio button.

3. IMS reorder the list of the Students report from which the owner's name closer to start according to the English alphabet.

Entry condition: . Evaluator should successfully login to the system and have a valid token throughout the process.

Exit condition: . Evaluator successfully sort the list of reports as his/her selection

---

Use case name: ChangeFAQ

Participating actors: Initiated by Admin

Flow of events:

1. Admin clicks the FAQ button from his dashboard and generates a toFAQ function on his/her terminal.

2. IMS redirect the Admin to FAQOfInternship page.

3. Admin sees the current FAQ for the internship report process for the Students and clicks the changeFAQ button.

4. IMS generates a box to show the admin a label to enter new questions and answers.

5. Admin enters the new question and its answer to the system.

6. IMS generates the saveChanges button.

7. Admin clicks the saveChanges button and generates saveNewFAQ function on his/her terminal.

8. IMS saves the new uploaded FAQ to the system.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: . Admin successfully uploaded the new question and its answer to the system

---

Use case name: CreateUser

Participating actors: Initiated by Admin

Flow of events:

1. Admin clicks the createUser button in her/his dashboard to generate the createUser function on his/her terminal.

2. IMS redirect Admin to a page that has fields representing necessary information for creating a user like email address, Bilkent Id, name and type and another "uploadExcel" button to activate the CreateUsersByExcel use case.

3. Admin enters User's information to the generated form.

4. IMS checks if the no form field remains blank and activates the submit button.

5. Admin clicks the "Submit" button to activate saveUser function on his/her terminal.

6. IMS creates a new User into the system by the given information.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process, **AND**

. At the time the system should have an ongoing registered semester.

Exit condition: . Admin successfully creates the User into the system.

Use case name: CreateUsersByExcel

Participating actors: Communicates with Admin

Flow of events:

1. Admin uploads his/her file in Excel format which contains necessary columns about the User, to the corresponding field.

2. IMS checks the Excel file for the necessary information columns and if the file contains all the prerequisite information for all the users, it generates a "Submit" button for the Admin.

3. Admin clicks the "Submit" button.

4. IMS creates a new User according to type information in the file for each row of the uploaded file and saves them into the database.

Entry condition: . This use case **extends** from the CreateUser use case. It is initiated by the system.

Exit condition: . Admin creates User accounts according the informations at the uploaded Excel file and saves them into the system, **OR**

. IMS stops the process because the uploaded Excel file does not contain one or more necessary information columns.

Use case name: AssignSubmissionToNewEvaluatorOrAssistant

Participating actors: Initiated by Admin

Flow of events:

1.Admin navigates to assign submission page.

2. IMS lists all of the submissions made for that semester.

3.Admin chooses the submission/s and selects which type of user will be assigned to that submission. After that, Admin will enter the id of the user that will be assigned.

4. IMS will save the response and add the submission to the submission to be evaluated list of the evaluator or the assistant.

Entry condition: Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: Admin assigns submission to the evaluator or assistant successfully.

Use case name: UploadCompanyEvaluationForm

Participating actors: Initiated by Admin

Flow of events:

1. Admin navigates to Part A file page of the selected Student.

2. IMS redirects Admin to the new page with a label which Admin could upload the company evaluation form file to the system in pdf format.

3. Admin uploads the company evaluation form to the corresponding field.

4. IMS checks the uploaded file size if it does not exceed the size limit and activates the "submit" report.

5. Admin clicks the "submit" button to generate the "enterCompanyEvaluationForm" function on his/her terminal.

6. IMS saves the uploaded file to the system and links it to the relative Student's submission.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: . Admin successfully uploads the company evaluation file to the system.

Use case name: CreateOneTimeLink

Participating actors: Initiated by Admin

Flow of events:

1. Upon request, Admin requests the system to generate a one time link for a Student's company evaluation form.

2. IMS generates a code and sends the link to the supervisor in an email.

3. The Internship Supervisor fills out the form and submits it.

4. IMS saves the responses of Supervisor.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process, **AND**

. The link of the Supervisor should not be expired.

Exit condition: . The Internship Supervisor fills out the form.

Use case name: ExportGrades

Participating actors: Initiated by Admin

Flow of events:

1. Admin clicks the "exportGrades" button from the selected semester's page and generates the "transformGradesToExcel" function on his/her terminal.

2. IMS creates a file to be downloaded by the Admin which contains registered Student's ID, name, and their current grade and semester information.

3. IMS generates an "download" button for the Admin to download the generated file.

4. Admin clicks the "download" button and downloads the generated file.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process, **AND**

. At the time the system should have an ongoing registered semester.

Exit condition: . Admin successfully downloads the "Grades" pdf.

Use case name: StartSemester

Participating actors: Initiated by Admin

Flow of events:

1. Admin clicks the "StartSemester" button from his/her dashboard to generate the "StartSemester" function on his/her terminal.

2. IMS redirects the Admin to a new page which contains input fields for semester name, course name, and a file upload field for users' information sheet (Excel file).

3. Admin fills the fields and uploads the file.

4. IMS activates the "Start" button after it checks that no field remains blank and the file is valid.

5. Admin clicks the "Start" button to generate a new semester object in the system.

6. IMS creates a new semester and saves it to the system as a semester. Then, the system creates users who are not previously enrolled into the system

according to the information in the uploaded sheet, sends semester registration mails to all users and sends account information mails to the new users.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process and have a valid Excel file that contains needed information for all users.

Exit condition: . Admin successfully ends a new semester and saves it to the system.

Use case name: EndSemester

Participating actors: Communicates with Admin

Flow of events:

1. Admin clicks the "EndSemester" button from the current semester's page and generates the "endSemester" function on his/her terminal.

2. IMS generates a confirmation dialog to get confirmation from the Admin to end the semester and activates the "Deactivate" button.

3. Admin clicks the deactivate button.

4. IMS marks the semester as deactivated and makes it available for exporting grades.

Entry condition: . This use case extends from the StartSemester use case. It is initiated by the system.

Exit condition: . Admin successfully ends a semester of the system.

Use case name: CheckFacultyMember'sWork

Participating actors: Communicates with Admin

Flow of events:

1. Admin can see the number of reports assigned to each Evaluator and Assistant and may change the distribution of the submissions as he/she desires.

2. IMS will show a list of the evaluators' and assistants' assigned submissions and their statuses.

Entry condition: Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: Admin successfully sees or changes the assignments.

Use case name: RedistributeWork

Participating actors: Communicates with Admin

Flow of events:

1. Admin can see which reports are assigned to who and change their assigned evaluators and assistants.

2.IMS will show the list of reports and information of these reports' evaluators and a button for each report to change the evaluator.

3. Admin clicks change buttons besides the reports and types in the Bilkent ID of the requested User.

4.IMS will assign the selected report to the selected evaluator.

Entry condition: Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: Admin successfully sees or changes the assignments.

Use case name: SeeReportHistory

Participating actors: Communicates with Admin

Communicates with Evaluator

Communicates with Assistant

Flow of events:

1. Admin, Evaluator or Assistant clicks the "see ReportHistory" button of the selected Student.

2. IMS generates a label for every report uploaded by the Student' submission, and every revision request by the Assistant and Evaluator.

3. Admin, Assistant or Evaluator can see and download every uploaded report and revision requests.

Entry condition: . This use case extends from the CheckFacultyMemeber'sWork use case. This use case is initiated by the system.

Exit condition: . Admin, Assistant or Evaluator successfully sees the report history of the selected Student.

Use case name: MarkStudentSubmissionAsPlagirasm

Participating actors: Communicates with Admin

Flow of events:

1. Admin navigates to mark submission as plagiarism or withdrawn page.

2. Admin types in Student's id and selects which submission to handle.

3. Admin clicks the Plagiarised button.

4. IMS marks the report as plagiarized and stops the evaluation process.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: . Admin, marks the submission as Plagiarized.

Use case name: MarkStudentSubmissionAsWithdraw

Participating actors: Communicates with Admin

Flow of events:

1. Admin navigates to mark submission as plagiarism or withdrawn page.

2. Admin types in Student's id and selects which submission to handle.

3. Admin clicks the Withdrawn button.

4. IMS marks the report as withdrawn and stops the evaluation process.

Entry condition: . Admin should successfully login to the system and have a valid token throughout the process.

Exit condition: . Admin, marks the submission as Withdrawn.

Use case name: ExcelFileIsCorrupted

Participating actors: Communicates with Admin

Flow of events:

1. Admin navigates to initialize a semester page.

2. Admin uploads the Users file and submits the form.

3. IMS rejects the file due to one of the several reasons. File is not an excel file, file does not contain required fields or file is empty.

Entry condition: . Uploaded file was corrupted.

Exit condition: . IMS rejects the file and tells the Admin to upload a valid file via a pop up.

Use case name: ReportFileIsCorrupted

Participating actors: Communicates with Student

Flow of events:

1. Student navigates to create a submission page.

2. Student uploads the report file and submits the form.

3. IMS rejects the file due to one of the several reasons. File is not a pdf file, or the file is not readable.

Entry condition: . Uploaded file was corrupted.

Exit condition: . IMS rejects the file and tells the Student to upload a valid file via a pop up.

Use case name: SubmissionDatesCollide

Participating actors: Communicates with Student

Flow of events:

1. Student navigates to create a submission page.

2. Student enters the internship dates and submits the form.

3. IMS rejects the submission if the Student has another internship record (for instance, if student is taking CS299 and CS399 at the same time) that overlaps with the given dates.

Entry condition: . Submission dates are not valid and overlaps with another internship record.

Exit condition: . IMS rejects the submission and tells the Student to provide valid dates.

Use case name: LinkExpired

Participating actors: Communicates with Internship Supervisor

Flow of events:

1. The Internship Supervisor clicks on the link in the email that was sent by the IMS.

2. IMS does not open the page and tell the Supervisor the link has been expired.

Entry condition: . Clicked link has an expired code associated with it.

Exit condition: . IMS does not open the page and creates a pop-up saying the link is expired.

Use case name: LinkAlreadyUsed

Participating actors: Communicates with Internship Supervisor

Flow of events:

1. The Internship Supervisor clicks again on the link in the email that was sent by the IMS.

2. IMS does not open the page and tell the Supervisor the link has already been used.

Entry condition: . Clicked link has been used before.

Exit condition: . IMS does not open the page and creates a pop-up saying the link is used before.

Use case name: UserInfoInvalid

Participating actors: Communicates with Admin

Flow of events:

1. Admin navigates to create a User page.

2. Admin enters the information of the User (ID, email, role, department, etc.)

3. Admin submits the form.

4. IMS rejects the creation of the user since the given information is invalid.

Entry condition: . Given User info is invalid.

Exit condition: . IMS does not create the user and creates a pop saying which fields are invalid and need to change.

Use case name: ReportDoesNotExist

Participating actors: Communicates with Evaluator

Flow of events:

1. Evaluator tries to check the report of the student by searching the student by his/her ID.

2. IMS does not forward the report since it does not exist.

Entry condition: Student has not uploaded any report.

Exit condition: IMS does not forward the report and creates a pop saying that the report does not exist.

3.5.2 Object and Class Model

High resolution version of the use case diagram can be seen here.

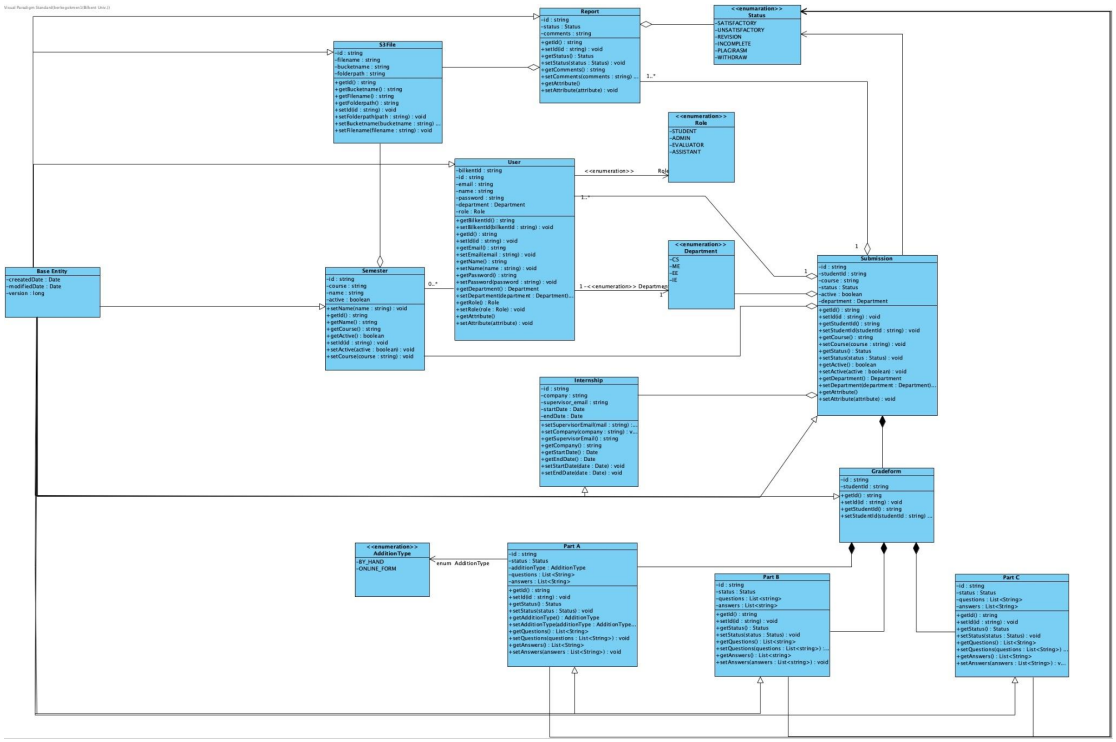

Figure 2: Object and Class models for the Internship Management System

Base Entity: A base supermodel that has creation date, modification date and version attributes. All the other models extend it.

S3File: A file model that is used to store the data of all kinds of files. Main purpose of this additional file class is to store a reference to the files in Amazon Simple Storage Service (S3). Since storing files in our main database would cost too much, we decided to use S3 and store related file id instead of files.

Semester: Semester model is used to store the course and semester information together.

User: User model stores the data of all kinds of users with related Role enumeration. We decided to use only one user model to decrease complexity and workload for the project. While the User model holds typical data for users like id, name and password, it also stores the semesters that the user has access to. For instance, if a student or evaluator has 2022-2023-spring/CS299 semester in their semester list, they will be able to do operations regarding that semester. Students will be able to upload reports, whereas the evaluator will be able to see assigned reports to them coming from that semester. Also, inactive semesters that the user had been assigned to can be seen for statistical reasons. Same logic applies for Assistant and Student types of users too. Users with admin roles however, won't have any semester assigned semesters to them.

Report: Report model represents the single report file with its comments and status. Every file upload correlates to a report object.

Internship: Internship model is used to store necessary data of a student's internship information like company name, supervisor email, and date range of internship. This model will be used for automating the company evaluation process and checking whether two internships collide in terms of start and end dates.

Submission: Submission model represents the student’s submission to the course. It stores the student’s data, all the uploaded reports (before and after the reviews), the internship information, the course and semester information, and related evaluation form of their submission. It also shows the general status of students’ reports. It is a general model for the evaluation process of the internship report. Moreover, it holds the assistant and evaluator information that is assigned to each submission.

Gradeform: Gradeform model represents the general evaluation of the student. It has multiple parts that are ordered and do not hold any other data than the student's id.

PartA: PartA model represents the first part of the evaluation form where a company evaluation grade is needed. It has an additional type of enumeration for keeping track of how company evaluation grades are entered to the system; by hand (Admin (Secretary) entered the grade) or automated (company supervisor filled the electronic form that comes with a one-time link). Other than that, the model only holds the question and answers for related parts of the report.

PartB - PartC: PartB and PartC represent the other parts of the evaluation form where part B needs to be passed for part C to be opened to the evaluator. These models only hold the data of questions in the forms and given answers.3.5.3 Dynamic Models

3.5.3 Dynamic Models

3.5.3.1 Activity Diagrams

3.5.3.1.1 Activity Diagram for Assigning Evaluators

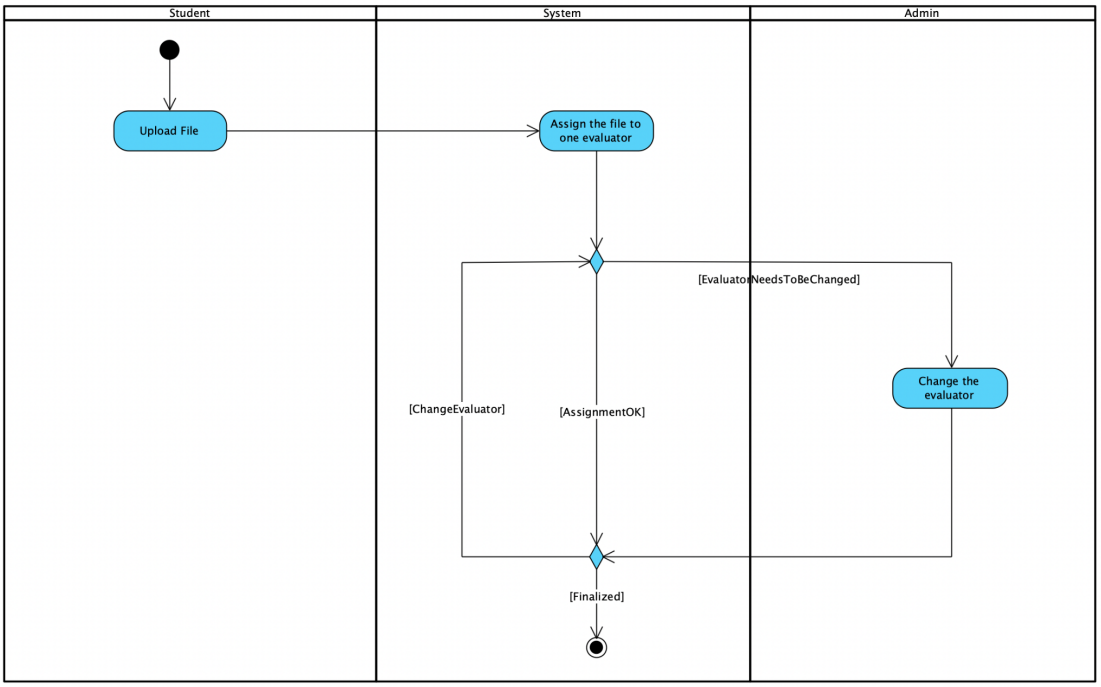

Figure 3: Changing Evaluator (Faculty Member) Activity Diagram for the Internship Management System

In the activity diagram for assigning teaching assistants, the student will upload his/her file. Then, the system will assign the file to a teaching assistant. If the assignment is convenient, there will be no changes to the assignment. If the teaching assistant needs to be changed, the admin will change the assigned teaching assistant. If any re-assignments need to be made after, the admin will be able to make this adjustment.

3.5.3.1.2 Activity Diagram for Assigning Assistants

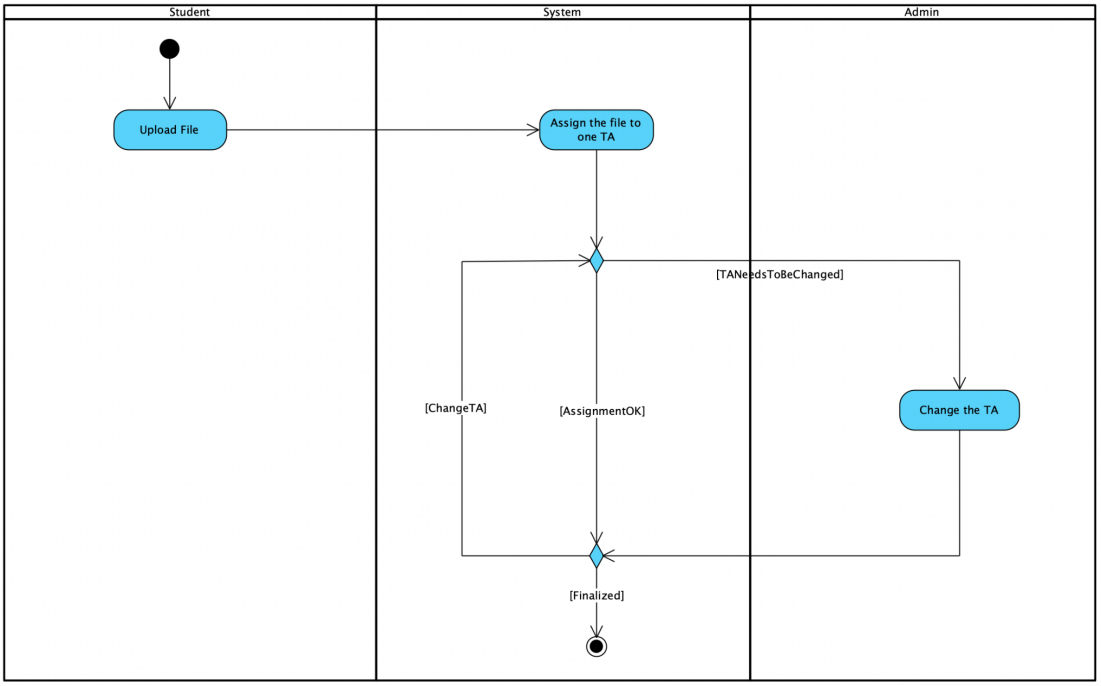

Figure 4: Activity Diagram of Assistant (TA) Assignment for the Internship Management System

In the activity diagram for assigning evaluators, the student will upload his/her file. Then, the system will assign the file to an evaluator. If the assignment is convenient, there will be no changes to the assignment. If the evaluator needs to be changed, the admin will change the assigned evaluator. If any re-assignments need to be made after, the admin will be able to make this adjustment.

3.5.3.1.3 Activity Diagram for Part A Evaluation

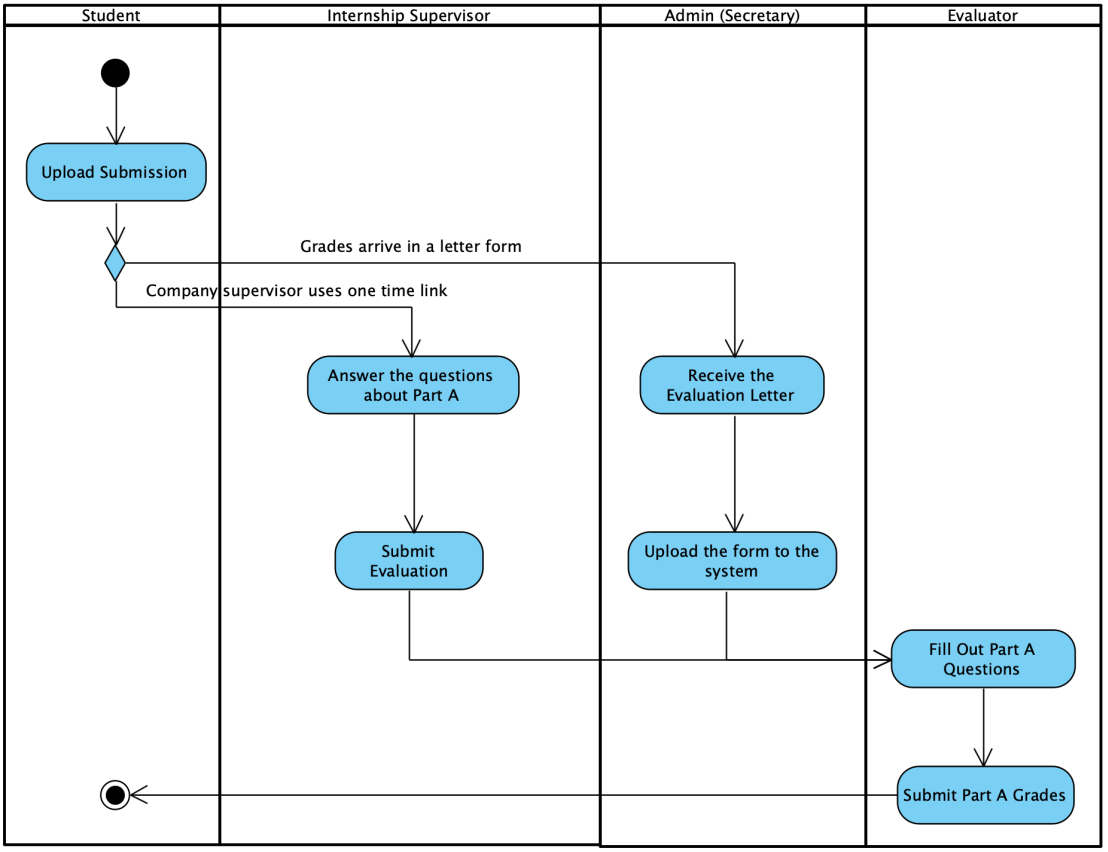

Figure 5: Part A Evaluation Activity Diagram for the Internship Management System

This diagram explains how the evaluation of part A is done. First, after the student uploads their submission, they choose how the company evaluation form will be delivered. If it'll be online via the link created by the system, it's only done once by the supervisor and the rest is handled by the system. If it'll arrive in letter form, then one of the department secretaries (admin) needs to upload that document to the system. Lastly, the faculty member (evaluator) looks at the grades and submits the overall evaluation for part A.

3.5.3.1.4 Activity Diagram for Part B Evaluation

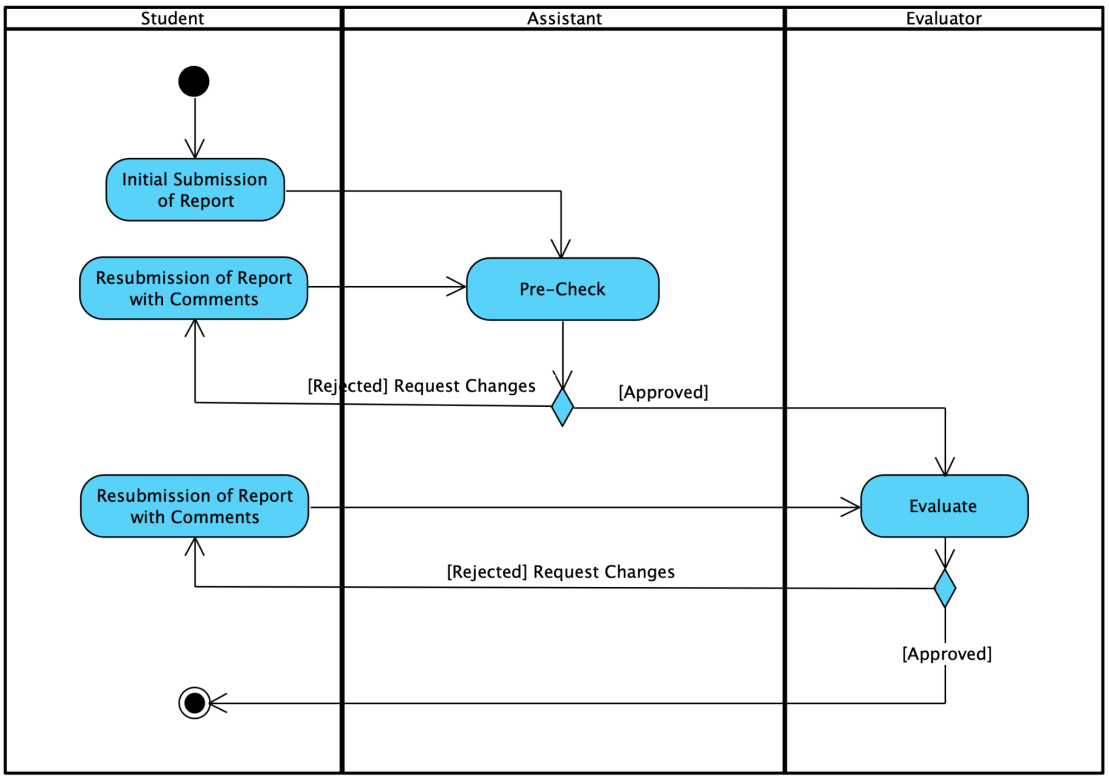

Figure 6: Part B Evaluation Activity Diagram for the Internship Management System

This diagram explains how the evaluation of part B is done. First, after the student submits their initial report, it's pre-checked by an assistant and re-uploaded by the student if requested. Then, it goes through the same steps with the evaluator, that is, it gets re-uploaded upon the evaluator's requests as many times as needed. Lastly, when the report no longer requires re-upload, the evaluator marks it as satisfactory.

3.5.3.2 State Diagrams

3.5.3.2.1 State Diagram for Semester Initialization & Deactivation

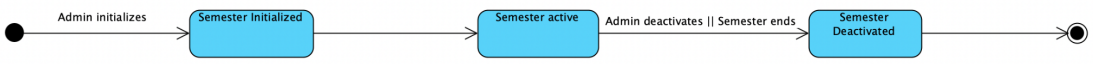

Figure 7: State Diagram for Semester Initialization and Deactivation

In the state diagram for semester initialization and deactivation, the admin of the internship management system will initialize the semester. The semester will be deactivated when the admin deactivates or the semester ends.

3.5.3.2.2 State Diagram for Evaluation of the Part A

High resolution version of the diagram can be seen here.

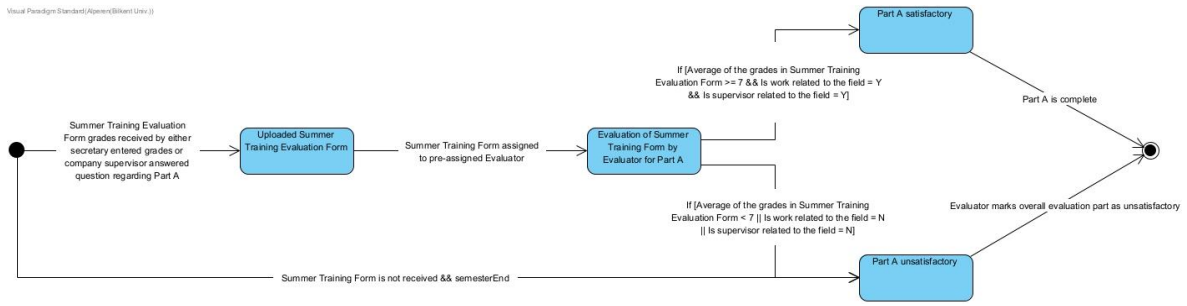

Fig. 8: Evaluation of the Part A State Diagram for the Internship Management System

This diagram shows how Part A of the internship evaluation form is evaluated.

3.5.3.2.3 State Diagram for Evaluation of the Part B and C

High resolution version of the diagram can be seen here.

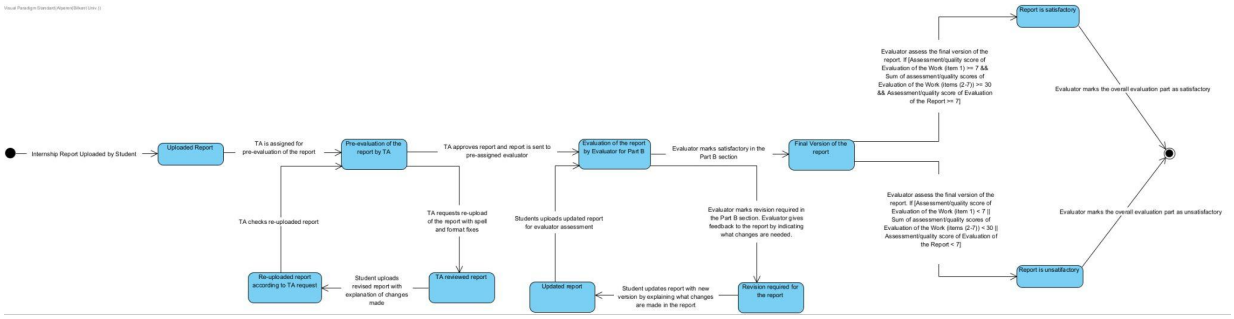

Fig. 9: Evaluation of Part B and C State Diagram for the Internship Management System

This diagram shows how the Part B and C of the internship evaluation form is evaluated.

3.5.3.2.4 State Diagram for Final Course Grade

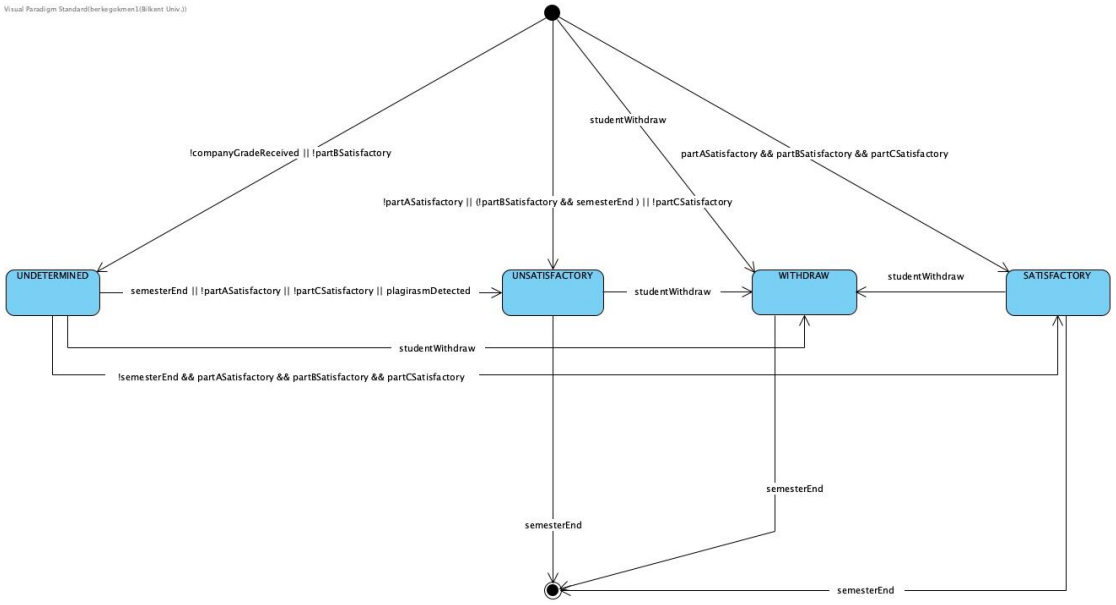

Figure 8: Course Grade State Diagram for the Internship Management System

This diagram explains how the overall grade of a student is determined. All the conditions depend on individual parts of the evaluation being completed. The state diagram of part A can be seen in *3.5.3.2.1*, part B and C in *3.5.3.2.*.

3.5.4 User Interface

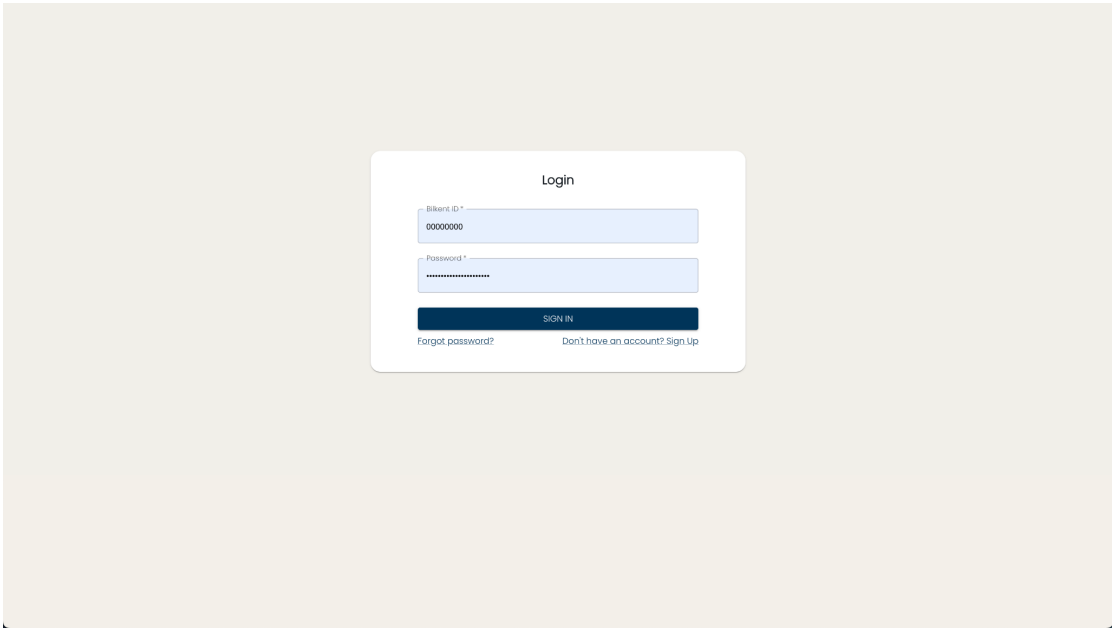

Figure 9: Login Page

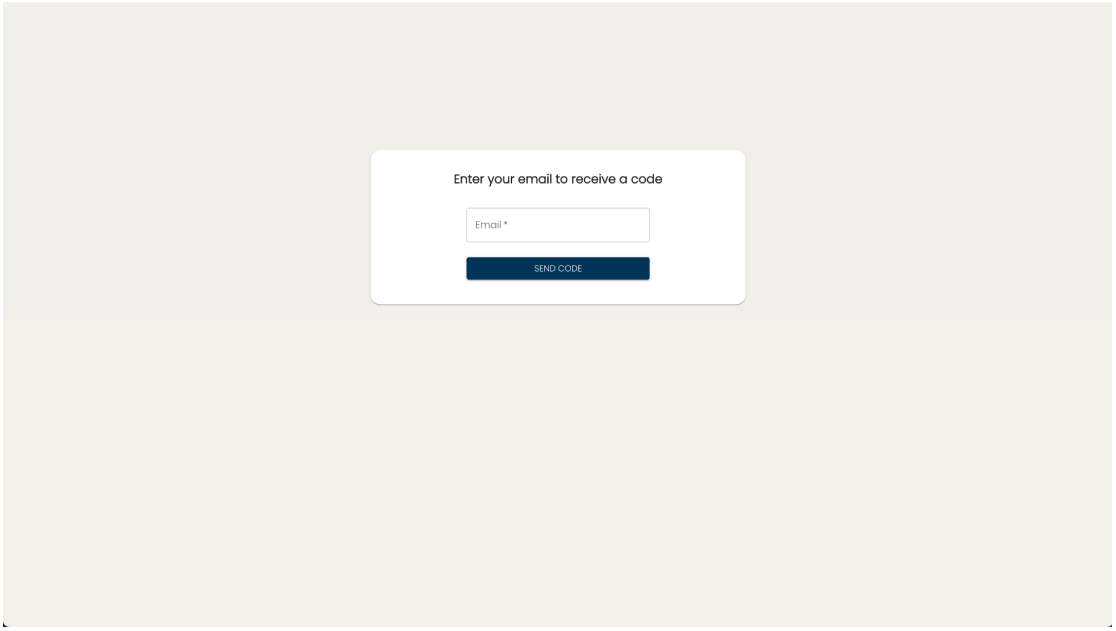

Figure 10: Request Password Reset Email Page

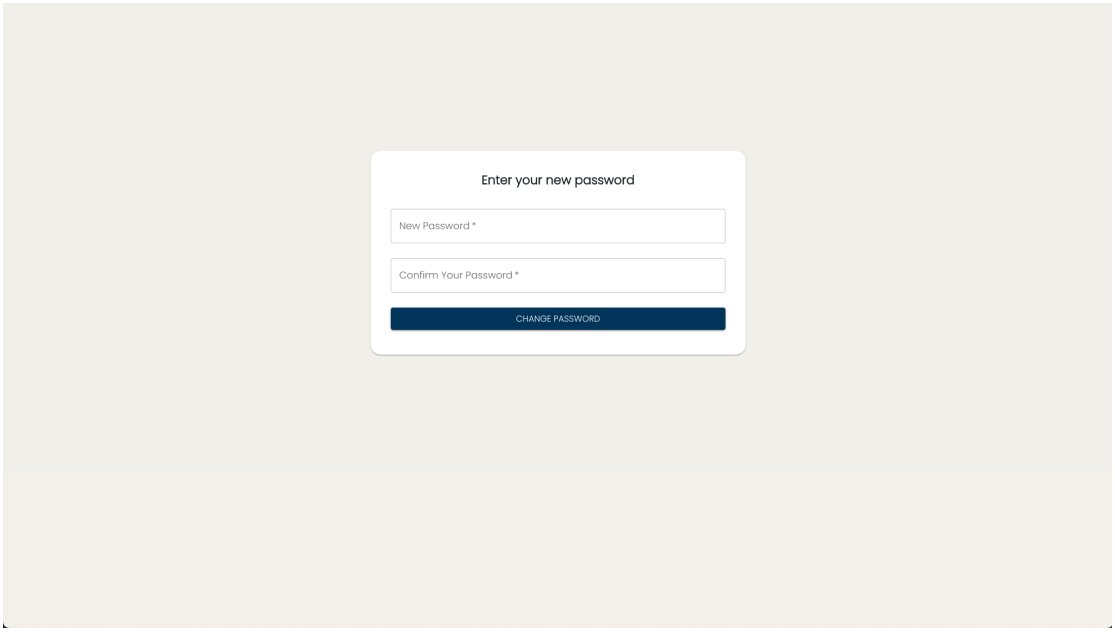

Figure 11: Change Password With Given Link Page

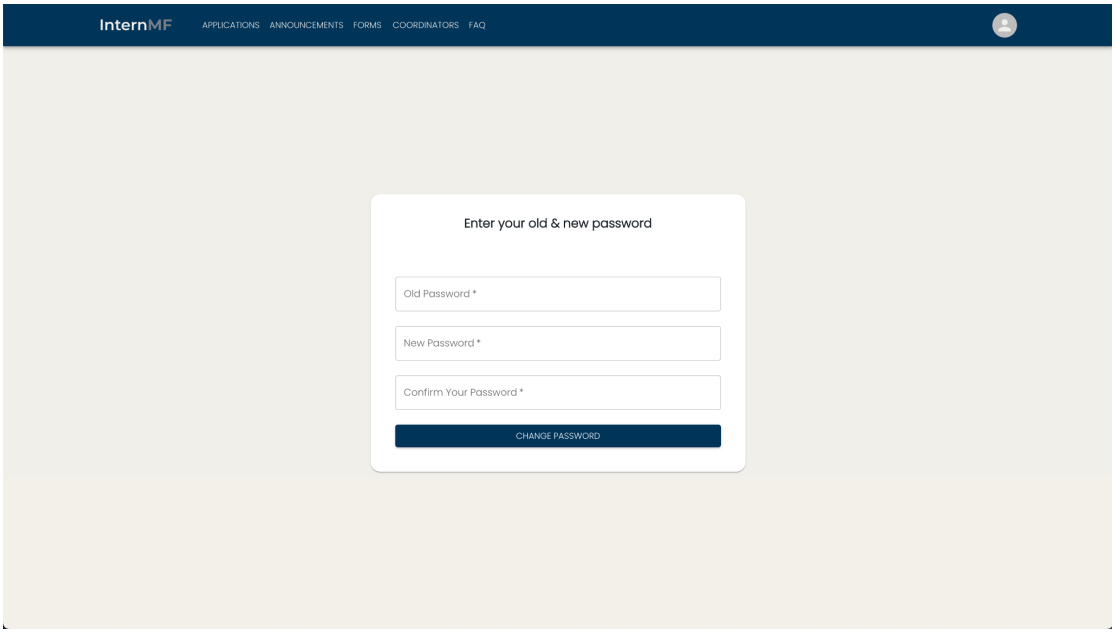

Figure 12: Change Password Using Existing Password Page

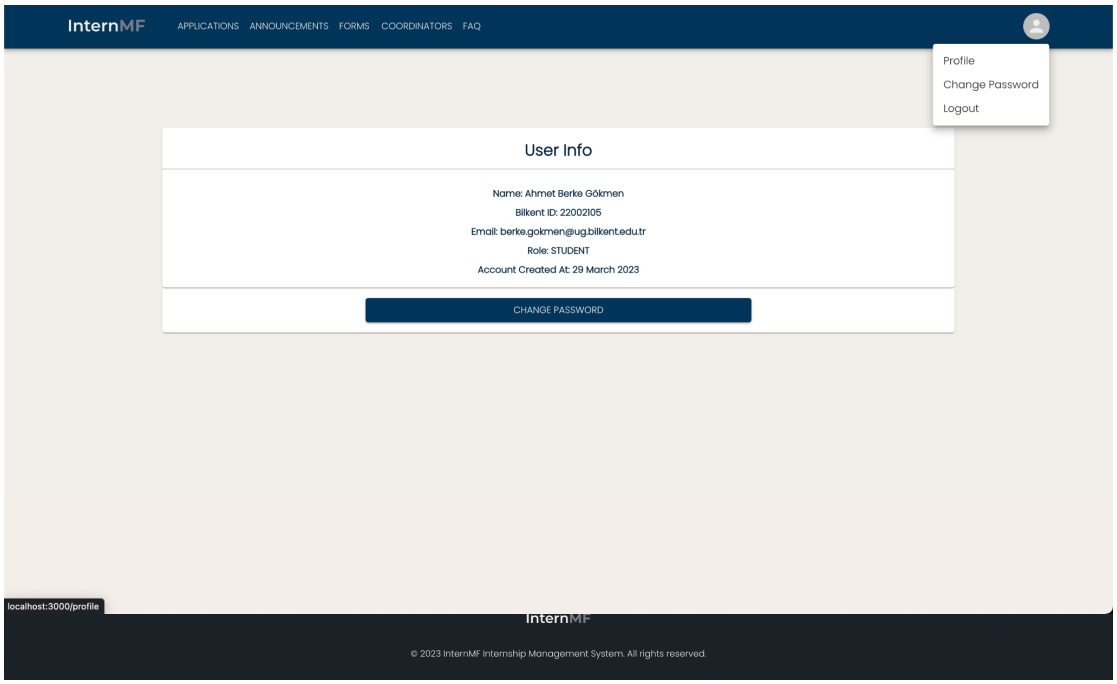

Figure 13: User Profile Page

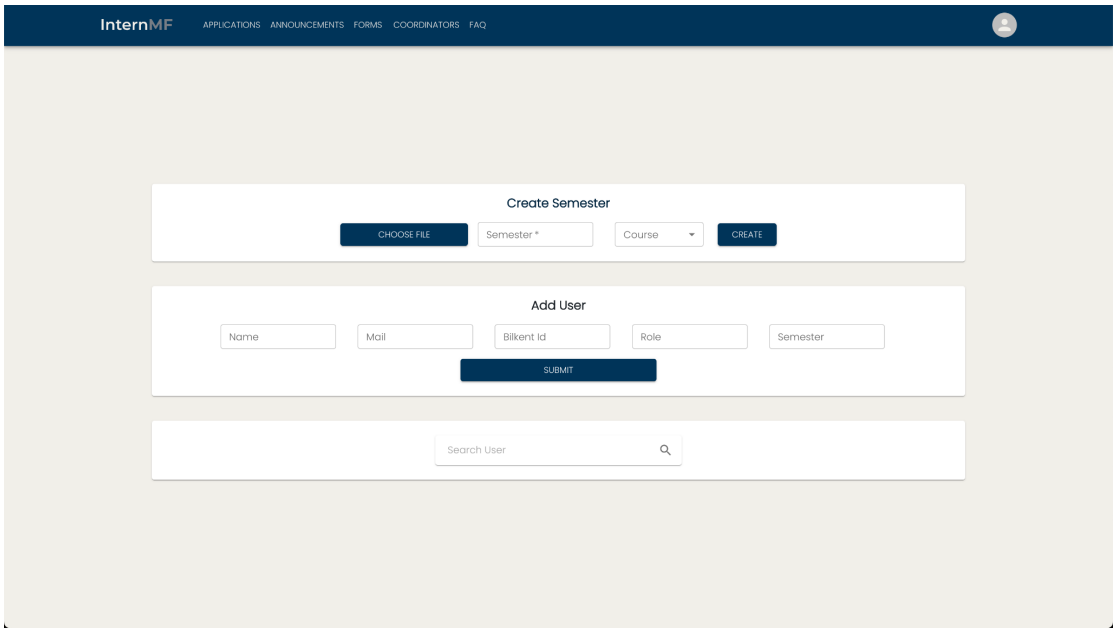

Figure 14: Initialize Semester and Add User Page

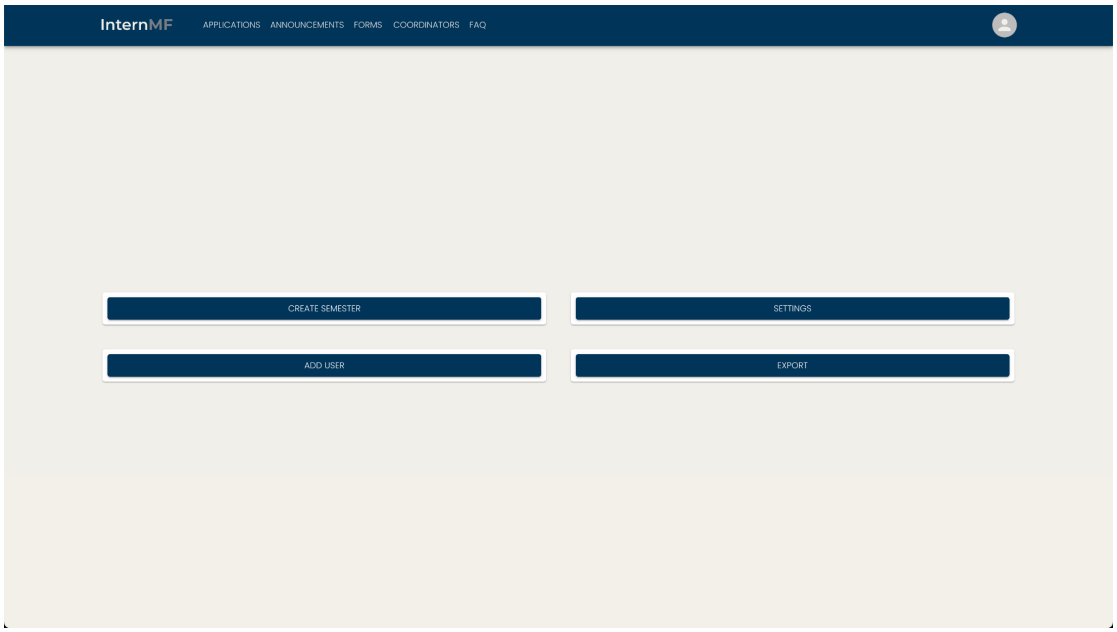

Figure 15: Admin Dashboard Page

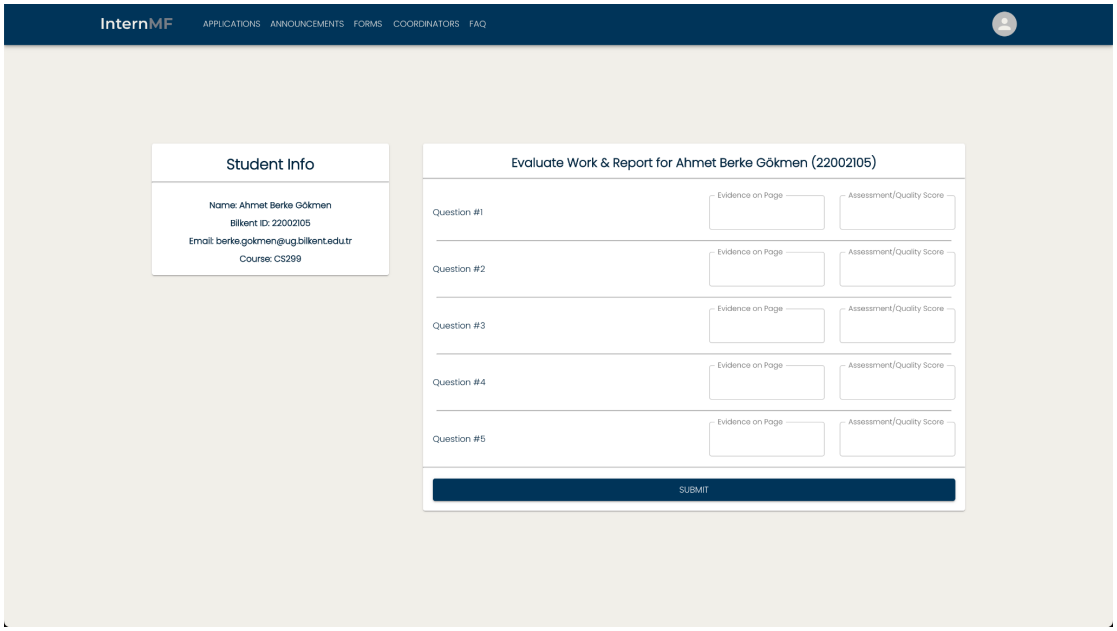

Figure 16: Evaluation of Work & Report Page

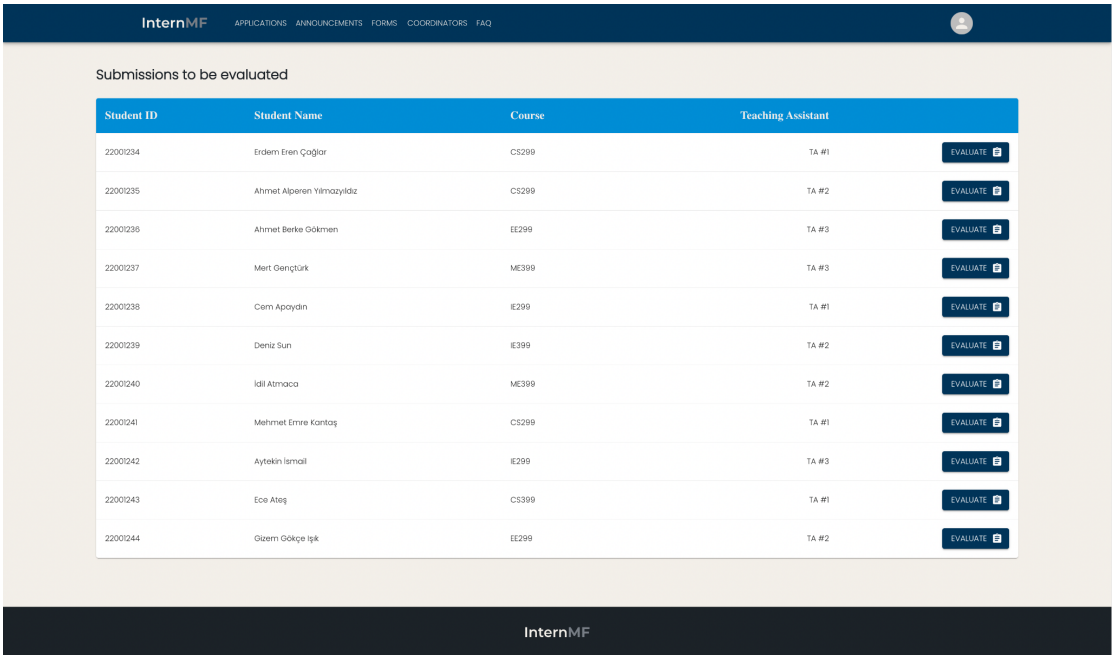

Figure 17: View Submissions to be Evaluated Page for Faculty Members

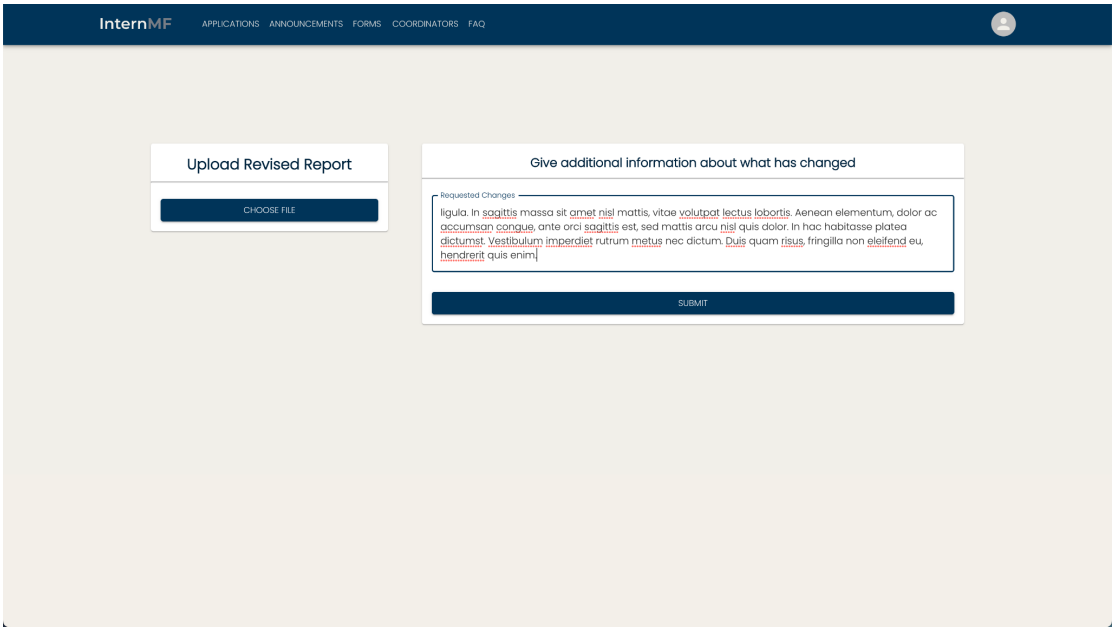

Figure 18: User Uploading Their Report With Changes Page

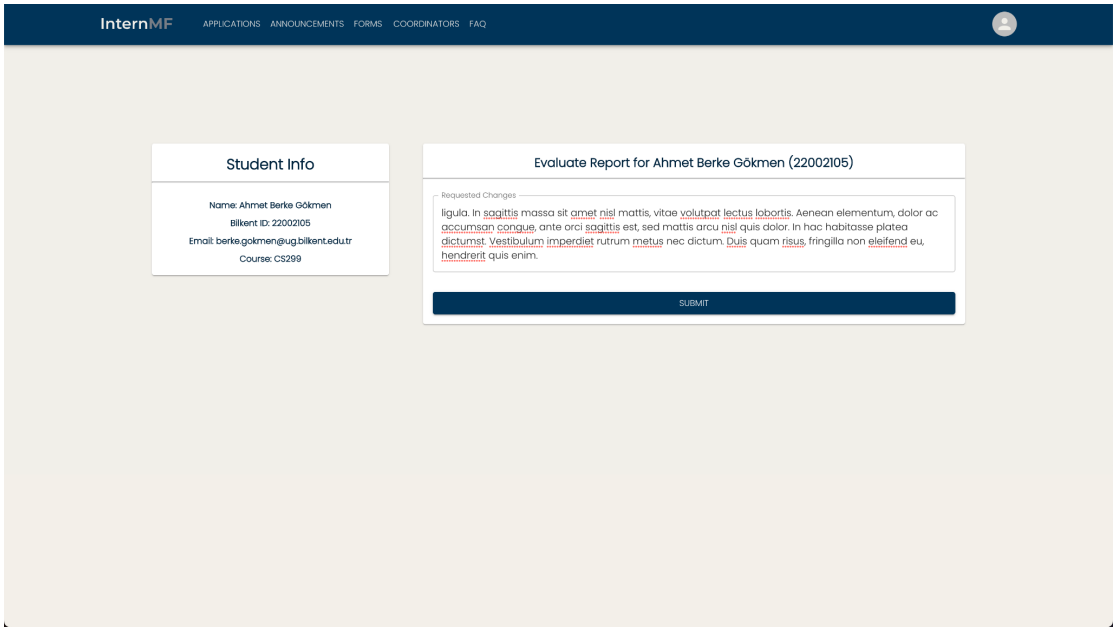

Figure 19: Faculty Member Request Changes in Report Page

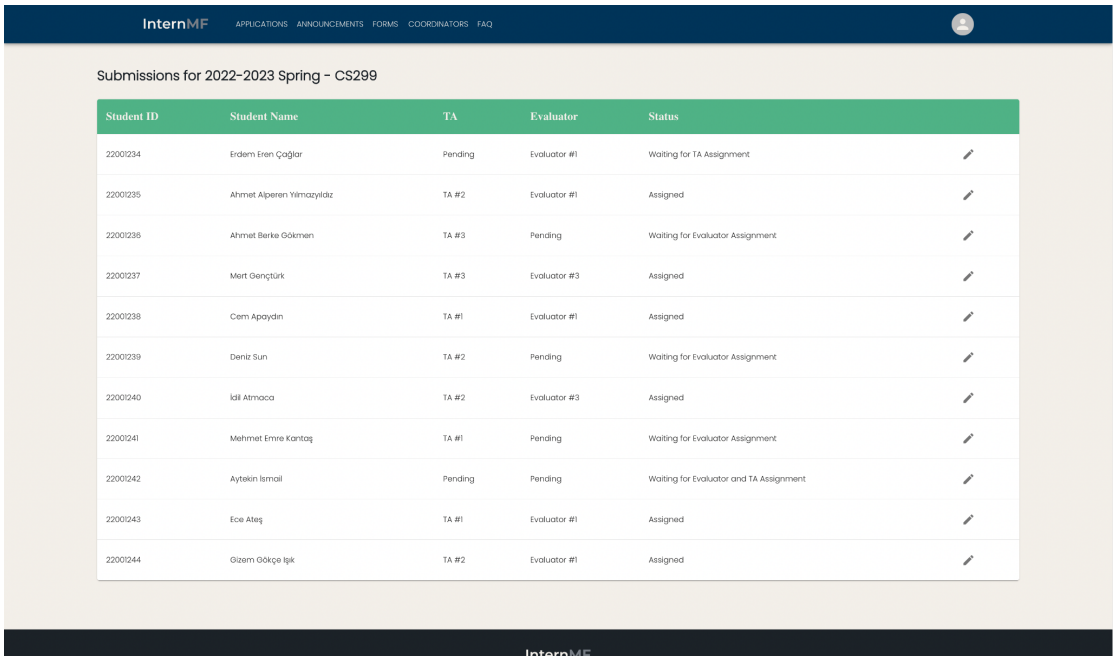

Figure 20: View Submissions and Change Assistant & Evaluator Page

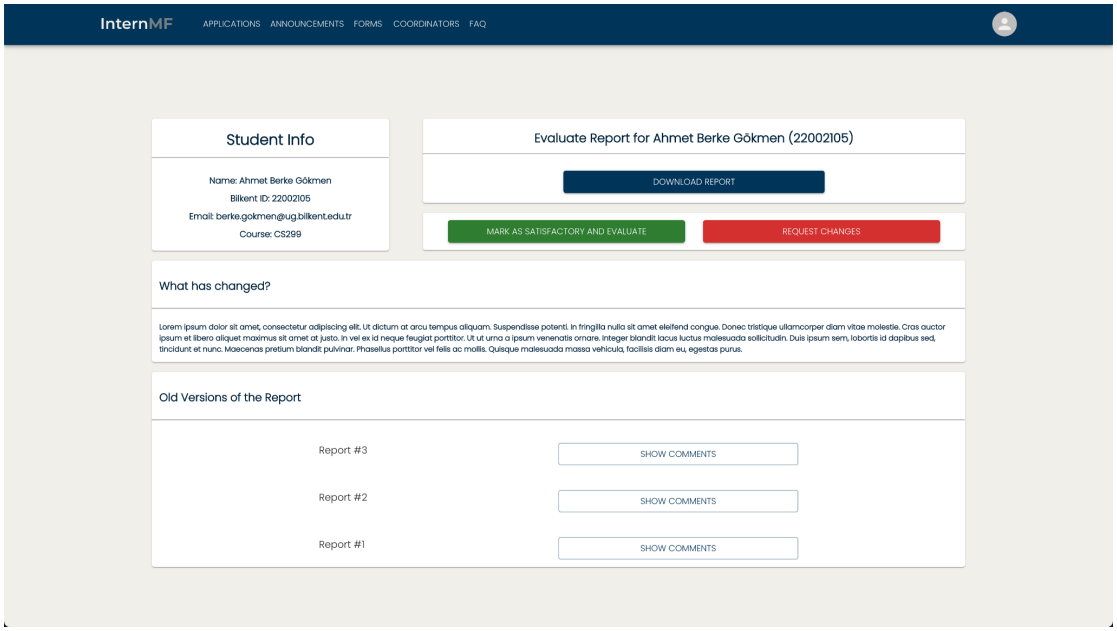

Figure 21: Evaluate Report Page for Faculty Member

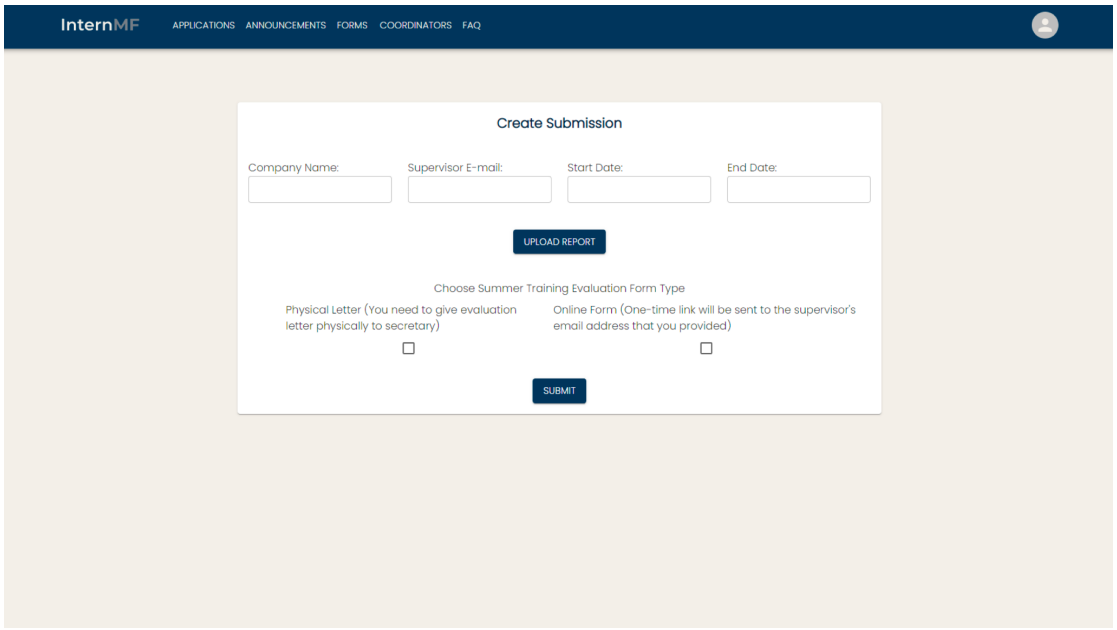

Figure 22: User Create Submission Page

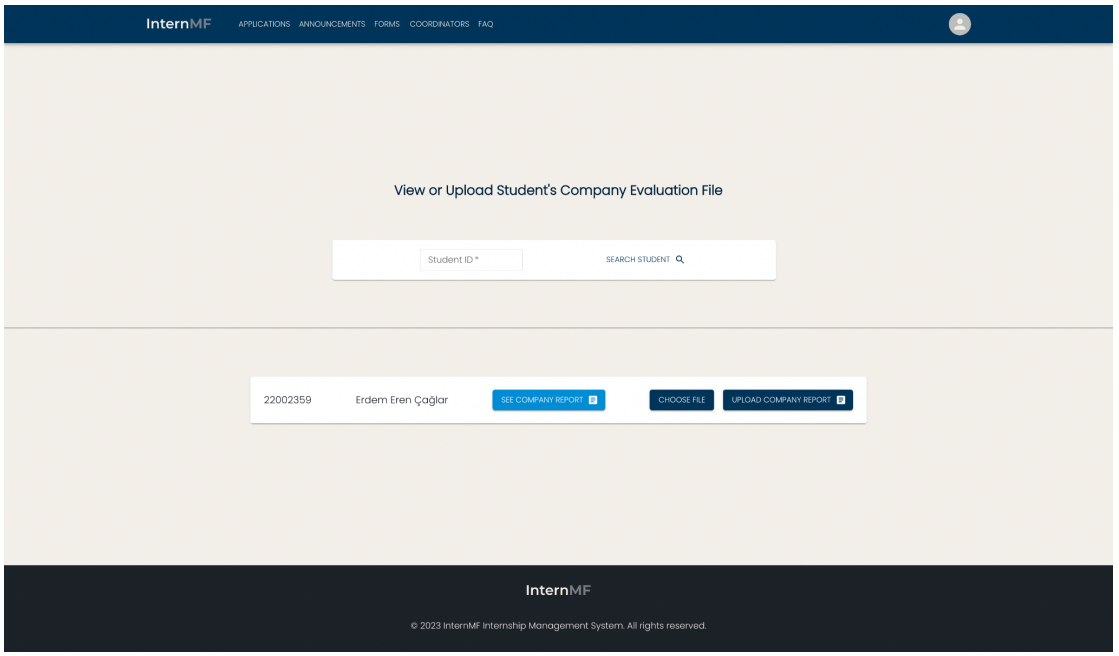

Figure 23: Upload Company Evaluation File for Students Page for Admins (Secretaries)

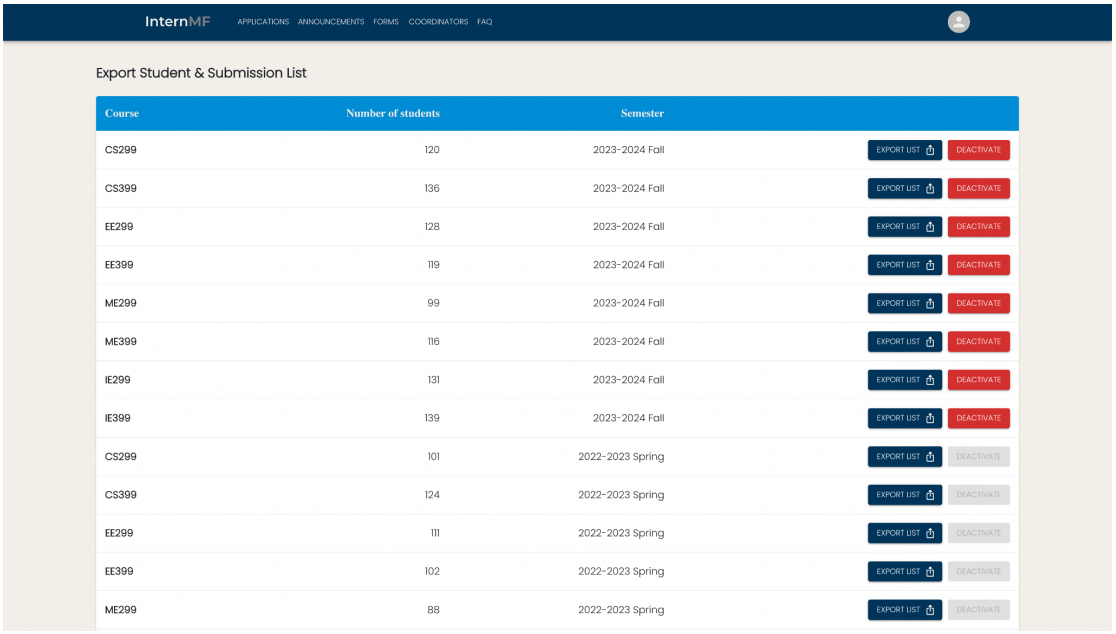

Figure 24: Deactivate Semester and Export Grades Page for Admins

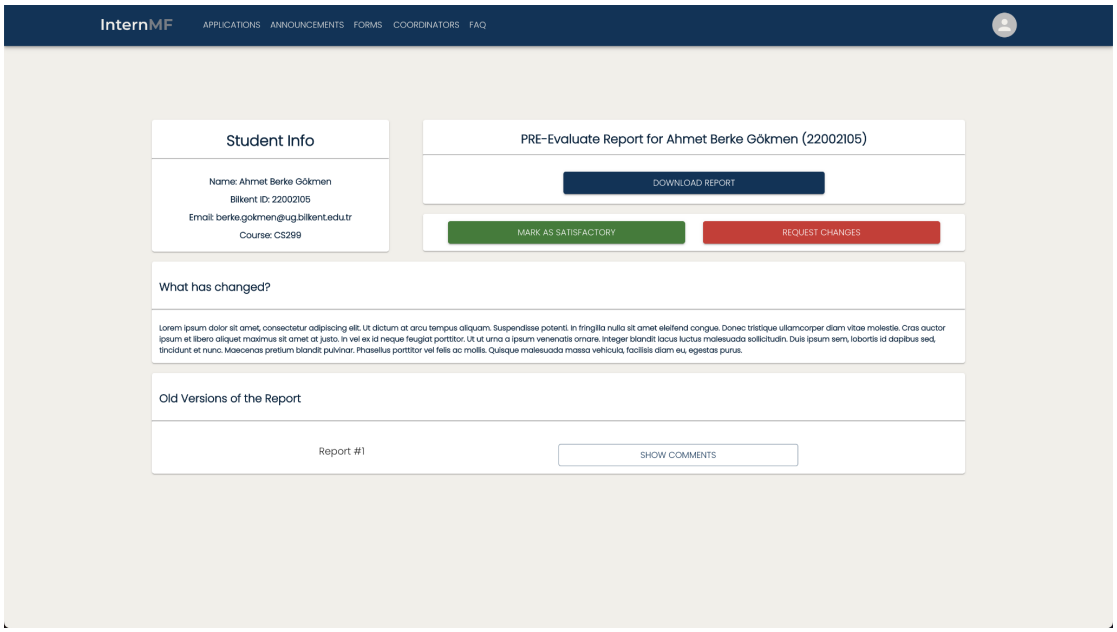

Figure 25: Evaluate Report Page for Assistant

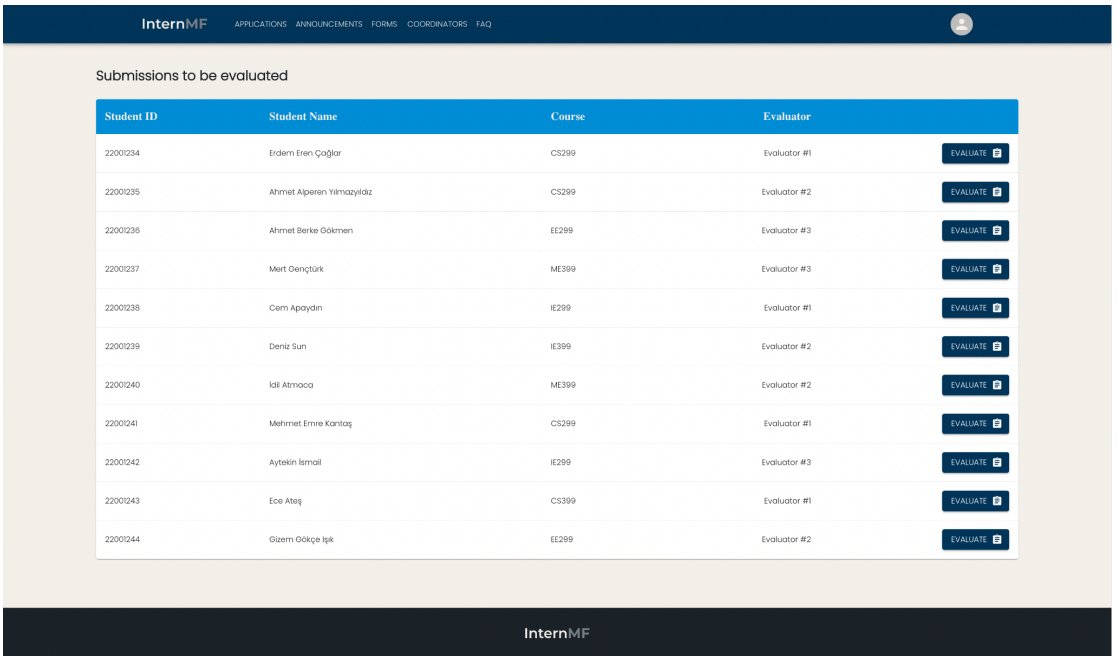

Figure 26: Evaluate Report Page for Assistant

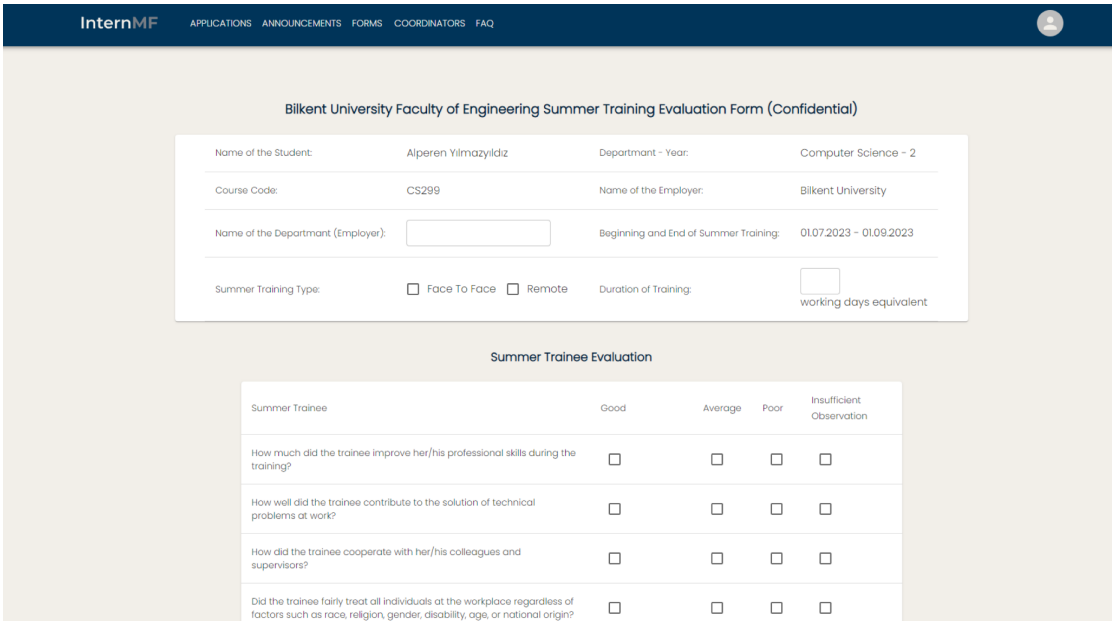

Figure 27: One-time Link Form for Internship Supervisor

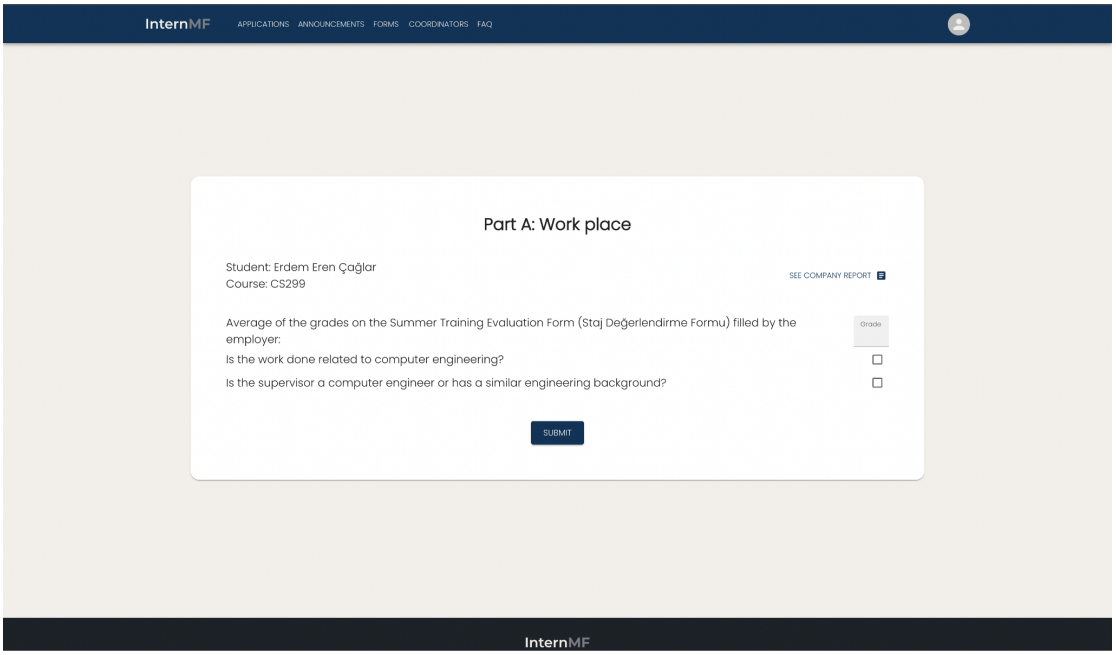

Figure 28: Part A Evaluation Page for Faculty Members

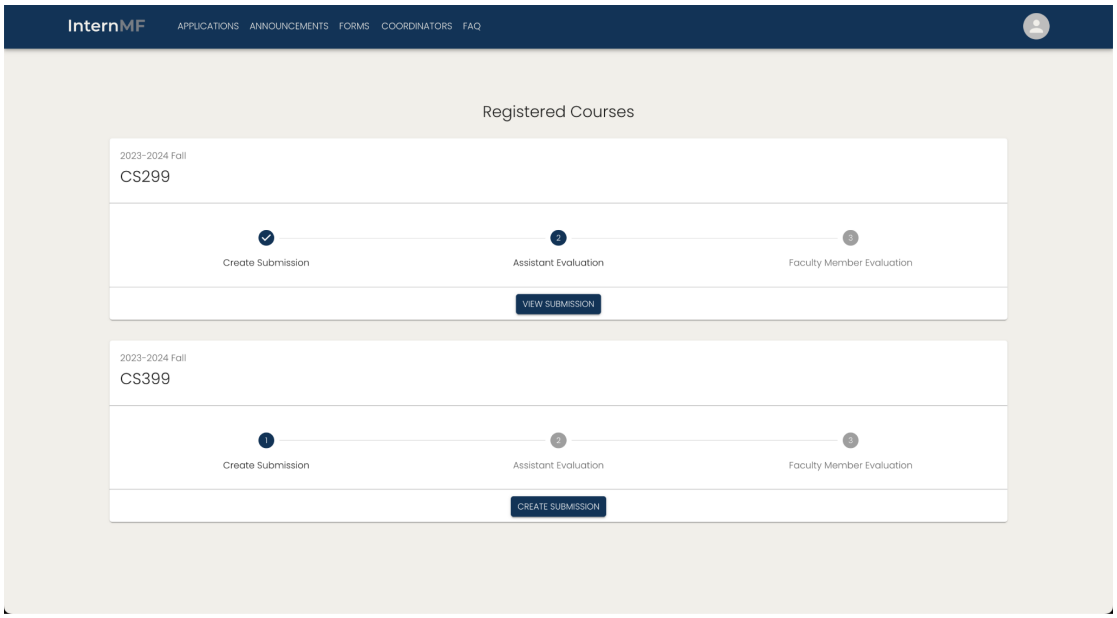

Figure 29: Student Dashboard Page

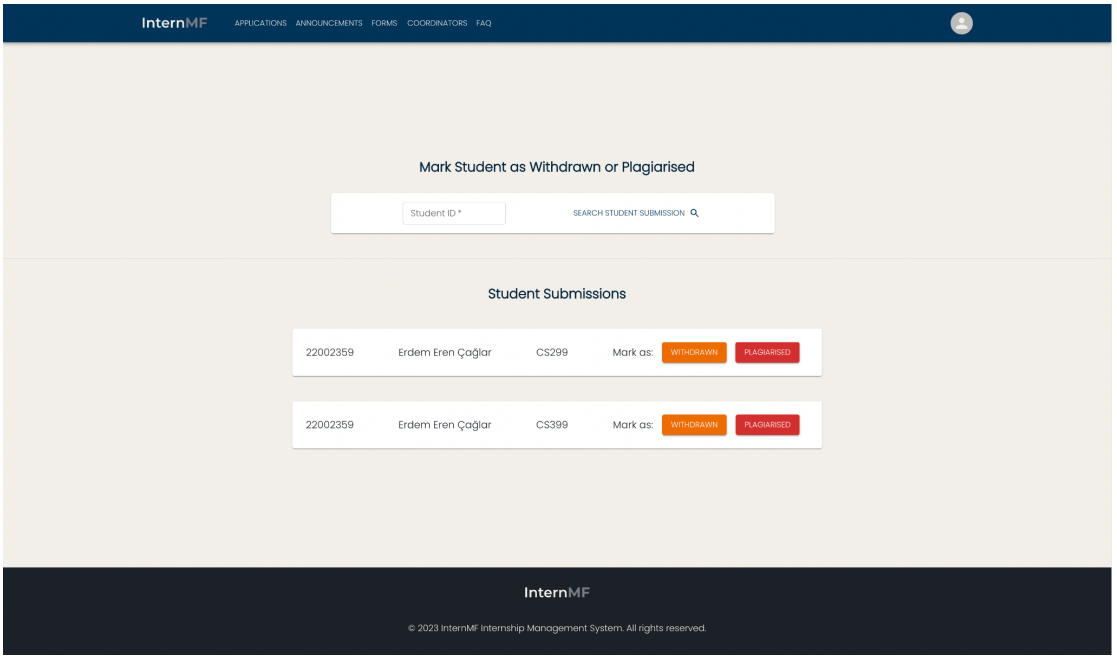

Figure 30: Marking Student Submission Withdrawn or Plagiarised Page for Admins

4 References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.