IHSAN DOGRAMACI BILKENT UNIVERSITY

CS319 Object-Oriented Software Engineering

Internship Management System

Project Analysis Report

Section 1

Group BOSGII

Göktuğ Yılmaz (21903048)

Berkay İnceişçi (21802088)

Oğuz Kuyucu (21902683)

Safa Eren Kuday (21902416)

İlayda Zehra Yılmaz (22001769)

İsmail Emre Deniz (21901913)

Contents

1. Introduction

In our project, we will create a website about the internship evaluation process. This project is designed for Bilkent Engineering Faculty students and internship reports graders, and its aim is to automate some manual processes in the current system.

Our project will implement mainly the following pages and features:

- The login page
- Admin's page to initialize department secretaries
- Department secretary's Student initialization page
- Department secretary's Instructor assignment page
- Student's main page to keep track of his report/s status
- Student's report upload page
- Instructor's report evaluation page

2. Current System

Our team participated in Mr. Selim Aksoy's presentation about the internship management system. In the current system, a combination of Moodle, Google Docs, and email is used, and the system is a bit inefficient because some works are not automated.

In the current system [1], students upload their reports to Moodle. Moodle does Turnitin check. After the Turnitin check, a TA checks the report quality and format. After format checking of TA, those reports are uploaded to the system in two folders: CS 299/CS 399. Those reports are shared with instructors by the faculty secretary manually. After this assignment, the faculty secretary mails the instructor manually. After the instructor uploads the feedback, he/she shares the report and comments with the faculty secretary manually via Google Docs. Those comments are shared with students via mail manually by the faculty secretary. After a student revises his/her report, the faculty secretary uploads the report and assigns it to an instructor, and lets the instructor know about this manually. After the evaluation process, evaluation forms are sent to the faculty secretary manually. Those evaluation forms are archived manually for ABET's requirements.

3. Proposed System

We propose a system where all participants of the current system, the department secretary, instructors/graders, and students, have a smoother experience than the current system. Our main objective is to either automate or ease some of the tedious work done by the department secretary and provide both graders and students with a platform where they can handle their part of the process without worrying about the organization or being uninformed.

3.1 Nonfunctional requirements

3.1.1 Security

The system has an admin who has complete control over all users, including the secretary, student and grader. The system also ensures security for students taking CS299/399 courses. The secretary initializes students in the system, and the system sends them randomly generated private passwords for access.

3.1.2 Usability

To ensure a positive user experience, our web application should prioritize usability. This means designing the web page with a clear and intuitive layout, consistent design elements, and minimal as possible while being functional interface. Users should be able to quickly find what they're looking for without confusion or frustration. The web page should also be optimized for various devices and screen sizes. By prioritizing usability, our web application can provide a relaxed and easy experience for all users.

3.1.3 Maintainability

Our web application should be maintainable in order to function effectively. Like many web applications, ours will be a complex system that can face several changes throughout the time of its lifecycle, such as new features, bug fixes, functionality updates, and performance enhancements. We can ensure maintainability with these attributes: well-organized code, clear documentation, and standardized development practices.

3.2 Functional Requirements

3.2.1 Student Functionalities

- The students have their own accounts where they can keep track of their status

The students can log in to the system using their individual credentials. Their account holds every bit of information regarding their summer internship and the process of grading their report.

- The students can submit their initial reports to the system

The students can submit the first draft of their internship reports before the general deadline.

- The students can receive and view feedback on their submission

After the grader submits their feedback, the student can see the feedback given via our system.

- The students can submit the revised version of their report

After answering all the bullet points in their feedback, the students can submit their revised version to the system. This process can be repeated until the grader comes to a final decision.

- The students can see the progress of their CS299/399 course

The students can see where they are on the progress bar beginning from their evaluation report incoming from their mentor to the first draft and the revisions of their internship report and to their final grade on the course.

3.2.2 Grader Functionalities

- The graders have their own accounts to keep track of their part

The graders can log in to the system using their individual credentials. Their account holds information regarding the students assigned to them and their progress on their internship reports.

- The graders can view and download the student’s reports

The graders have access to the reports of their assigned students, and they can view and download the reports for inspection.

- The graders can give feedback on reports

After inspecting the reports, the graders can submit their feedback using our system. The feedback consists of bullet points for students to individually respond to. The fixed deadline interval of two weeks applies automatically for the student to submit their revised version. This process can be repeated until the grader comes to a final decision.

- The graders can grade the student reports

The graders can give satisfactory or unsatisfactory grades to the reports of their students.,

- The graders can fill out and generate the final evaluation form

The required form for the final evaluation can be filled out and generated via our system. After filling out the necessary text boxes corresponding to the questions in the evaluation form, the graders can give their consent to generate a signed document of the final evaluation.

- The graders can drop some/all students in their assigned list

The graders can drop selected students in their list for the department secretary to give other graders in case of need. The graders must fill out a text box to express the reason behind this action.

3.2.3 Department Secretary Functionalities

- The department secretaries have their own accounts

The department secretaries can log in to the system using their individual credentials. Their account holds all information about the graders and the students and their individual progress on the internship report submission.

- The department secretaries can initialize student information

The department secretaries can upload a .csv or .xlsx file with relevant student information and initialize a student list. The file may contain information on the company the students did their internship in.

- The department secretaries can initialize grader information

The department secretaries can initialize graders using the application interface providing relevant information on the graders.

- The department secretaries can assign students to the graders

The department secretaries can distribute students to each grader with regard to the maximum number of students each grader can be assigned to.

- The department secretaries can see the course progress.

- The department secretaries have access to see the course progress of students and graders. The department secretaries can see graders' activity on the application.

3.2.4 Admin Functionalities

- Admins can view the list of the secretaries. They can add new secretaries by specifying their e-mail addresses. Also, they can delete existing secretaries.

3.3 Pseudo Requirements

- Object oriented design paradigms must be used.
- The project has to be usable as a web based application.

3.4 System models

3.4.1 Use Case Textual descriptions

3.4.1.1 Student Use Cases

Use Case 1

Name: Upload Internship Report

Participating Actor: Student

Entry Condition: The student takes CS299/399 & the student has done their training in an approved company

Exit Condition: The Internship report is uploaded

The Flow of Events:

1. The student has done their training and written their report
2. The student uploads their report to the system

Special Requirements: The report must be uploaded before the deadline

Use Case 2

Name: View the Status of Feedback Report

Participating Actor: Student

Entry Condition: The student is logged in

Exit Condition: The student has viewed the status

The Flow of Events:

1. The student views the report status

Special Requirements: -

Use Case 3

Name: View The Report Feedback

Participating Actor: Student

Entry Condition: The student has submitted their report & Grader has given feedback

Exit Condition: The student has viewed the feedback

The Flow of Events:

1. The student views the feedback

Special Requirements: -

Use Case 4

Name: Upload the Revised Version of the Internship Report

Participating Actor: Student

Entry Condition: The student has viewed the report feedback

Exit Condition: The Revised report is uploaded

The Flow of Events:

1. The student views the feedback
2. The student fills in the corresponding textboxes to the grader's feedback with the revised information
3. The student uploads the revised report

Special Requirements: The report must be uploaded before the deadline & Every bullet point in the feedback must be responded to.

Use Case 5

Name: Notify the grader of the submitted report

Participating Actor: Student

Entry Condition: The student has uploaded the revised report

Exit Condition: The student has notified the grader

The Flow of Events:

1. The student clicks to notify grader button
2. The system sends an automatically generated email to the grader notifying their report status

Special Requirements: The submitted report is the revised version & the email information of the grader is available

Use Case 6

Name: See the Grade of CS299/399

Participating Actor: Student

Entry Condition: The grader has graded the student

Exit Condition: The student has seen the grade

The Flow of Events:

1. The student sees the letter grade

Special Requirements: -

3.4.1.2 Grader/Instructor Use Cases

Use Case 1

Name: View the Assigned Student List

Participating Actor: Grader

Entry Condition: The grader has assigned a student list

Exit Condition: The grader has viewed the student list

The Flow of Events:

1. The grader views the student list

Special Requirements: -

Use Case 2

Name: Download the Student's Internship Report

Participating Actor: Grader

Entry Condition: The student is on the grader's student list & the student has submitted their report.

Exit Condition: The grader has downloaded the report

The Flow of Events:

1. The grader views the student list
2. The grader downloads the selected student's report

Special Requirements: -

Use Case 3

Name: Give Feedback to Student's Report

Participating Actor: Grader

Entry Condition: The grader has downloaded the student's report

Exit Condition: The grader has submitted their feedback

The Flow of Events:

1. The grader selects the student to give feedback
2. The grader enters some bullet points of feedback to be answered
3. The grader submits their feedback

Special Requirements: -

Use Case 4

Name: Notify the Student of their status

Participating Actor: Grader

Entry Condition: The grader has submitted their feedback to the most recent version of the student's report

Exit Condition: The grader has notified the student

The Flow of Events:

1. The grader submits their feedback
2. The grader clicks to notify student button
3. The system sends an automatically generated email to the student notifying their report status

Special Requirements: The email information of the student must be available

Use Case 5

Name: Grade the Student's report

Participating Actor: Grader

Entry Condition: The grader has downloaded the most recent version of the student's report

Exit Condition: The grader has graded the report

The Flow of Events:

1. The grader grades the report

Special Requirements: If the report is a revised version, the feedback boxes must be filled

Use Case 6

Name: Generate the Final PDF

Participating Actor: Grader

Entry Condition: The grader has graded the student's report

Exit Condition: The grader has generated the final pdf

The Flow of Events:

1. The grader fills in the necessary sections in the final pdf with the relevant information
2. The grader gives their consent to sign the report
3. The grader generates the pdf

Special Requirements: The signature of the grader must be available

Use Case 7

Name: Dropout Selected Students from Student List

Participating Actor: Grader

Entry Condition: The grader has been assigned to a student list

Exit Condition: The grader has dropped out selected students from their list

The Flow of Events:

1. The grader views the student list
2. The grader selects some/all students
3. The grader clicks to drop out the students
4. The grader fills a textbox with the reason behind this action

Special Requirements: Selected students must be ungraded

3.4.1.3 Secretary Use Cases

Use Case 1

Name: Upload company evaluation form

Participating Actor: Secretary

Entry Condition: The student is taking xx299/399

Exit Condition: The company evaluation report is uploaded

The Flow of Events:

1. The department secretary selects the pdf which is e-mailed to him/her by the company.
2. The department secretary uploads the company evaluation report.

Special Requirements: -

Use Case 2

Name: Assign a student to a grader

Participating Actor: Secretary

Entry Condition: The student is taking xx299/399 and the grader is available

Exit Condition: A grader assigned to a student

The Flow of Events:

1. The department secretary selects a grader
2. The department secretary selects a student from the student list
3. Department Secretary sets deadline for grader

4. The department secretary assigns the selected student to the selected grader.

Special Requirements: Selected students must be ungraded, an instructor cannot be assigned more than max number of assignable students.

Use Case 3

Name: Initialize students

Participating Actor: Secretary

Entry Condition: the students should be eligible to take xx299/399

Exit Condition: All necessary students have accounts

The flow of events:

1. Secretary chooses the excel file which contains students' id numbers, mails and courses(xx299/399 or both)
2. The program creates an account for every student in the excel sheet
3. The program emails username and password to students

Special Requirements: -

Use Case 4

Name: Re-assign student to a grader

Participating Actor: Secretary

Entry Condition: the student's report should be under evaluation and the grader needs to be changed for whatever reason

Exit Condition: the student is assigned to the available grader

The flow of the events:

1. The secretary selects a student from the list
2. The secretary selects an available grader
3. The secretary assigns the student to the grader

Special Requirements: If the student is taking 299/399 at the same time, reports may be assigned to different graders

Use Case 5

Name: View student list

Participating Actor: Secretary

Entry Condition:-

Exit Condition:Secretary selects something else from the website or closes the website

The flow of events:

1. The secretary selects view the students option.
2. Secretary views a table with students' names, ids, their reports’ status, and the deadline of their report if it is still under evaluation.

Special Requirements: -

Use Case 6

Name: View the report evaluation form of the student

Participating Actor: Secretary

Entry Condition: Report grade form should be available

Exit Condition: Secretary views report evaluation form of student

The flow of events:

1. The secretary selects a student from the student list.
2. Secretary downloads report evaluation form of the student

Special Requirements: If a student is taking 299/399 at the same time, the department secretary can see both of the report evaluation forms.

Use Case 7

Name: Initialize graders

Participating Actor: Secretary

Entry condition:-

Exit condition: Graders’ accounts are initialized

The flow of events:

1. The department secretary uploads the excel file which contains the graders’ emails and names.
2. The program creates an account for every grader
3. The program emails graders their usernames and passwords.

Special Requirements: -

Use Case 8

Name: Set the maximum number of assignable students to one instructor

Participating Actor: Secretary

Entry condition: There should be no instructors who are assigned more students than that number

Exit condition: Graders' accounts are initialized

The flow of events:

1. The department secretary sets the maximum number of students for one grader.

Special Requirements: -

3.4.1.4 Admin Use Cases

Use Case 1

Name: Initialize Department Secretaries' accounts

Participating Actor: Admin

Entry Condition: The user should have "admin" role

Exit Condition: Department Secretaries have accounts

The flow of events:

1. Admin selects initialize account for department secretary option
2. Admin types department secretaries' mail and selects his/her department
3. Program emails username and password to department secretaries

Special Requirements: In order to initialize an account for a department secretary, that department shouldn't have a department secretary account

Use Case 2

Name: View secretaries' accounts

Participating Actor: Admin

Entry Condition: The user should have "admin" role

Exit Condition: Admin views secretaries account

The flow of events:

1. The admin selects view department secretaries
2. The admin views department secretaries' accounts

Special Requirements:

Use Case 3

Name: Seize Department Secretaries' accounts

Participating Actor: Admin

Entry Condition: The user should have "admin" role

Exit Condition: Department Secretary's account is seized

The flow of events:

1. Admin selects department secretary from the list
2. Admin clicks seize account from the list

Special Requirements: An account should exist to be seized

Use Case 4

Name: Log in

Participating Actor: Student, Grader, Coordinator, Secretary, admin

Entry Condition: The user should have a valid account

Exit Condition: The user entered the system

The flow of events:

1. The user types his username and password
2. The user is logged in

Special Requirements: -

Use Case 5

Name: Change password

Participating Actor: Student, Grader, Coordinator, Secretary, admin

Entry Condition: The user should be logged in

Exit Condition: The user changed his/her password

The flow of events:

1. The user selects change password
2. The user types his/her old password and new password
3. The user's password is changed

Special Requirements: -

Use Case 6

Name: Forget password

Participating Actor: Student, Grader, Coordinator, Secretary

Entry Condition: -

Exit Condition: User changed his/her password

The flow of events:

1. The user selects forget password
2. The user is sent a mail which includes a link
3. The user clicks a link and he/she changes his/her password

Special Requirements: -

Use Case 7

Name:Logout

Participating Actor: Student, Grader, Coordinator, Secretary

Entry Condition: -

Exit Condition: User logs out

The flow of events:

1. The user clicks logout button.
2. The user’s session is seized.

Special Requirements: -

3.4.1.5 Grammarly Use Cases

Name: Spell check

Participating Actor: Grammarly

Entry Condition: Grammarly is ordered to check spelling

Exit Condition: The spelling is checked

The flow of events:

1. Grammarly ordered to check spell
2. Grammarly creates a report
3. Grammarly returns spell check report

Special Requirements:-

3.4.1.6 Turnitin Use Cases

Name: Plagiarism check

Participating Actor: Turnitin

Entry Condition: Turnitin is ordered to check the similarity

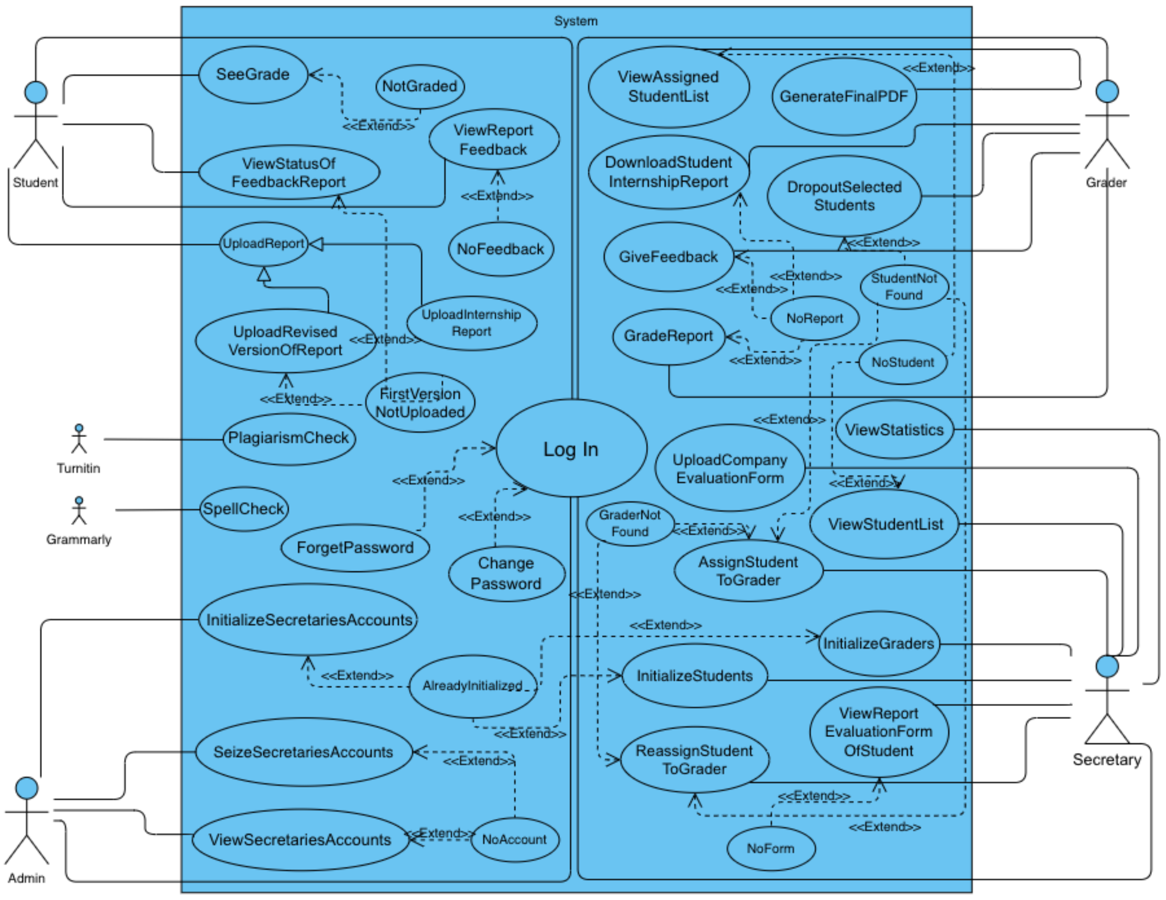
Exit Condition: The similarity is checked

The flow of events:

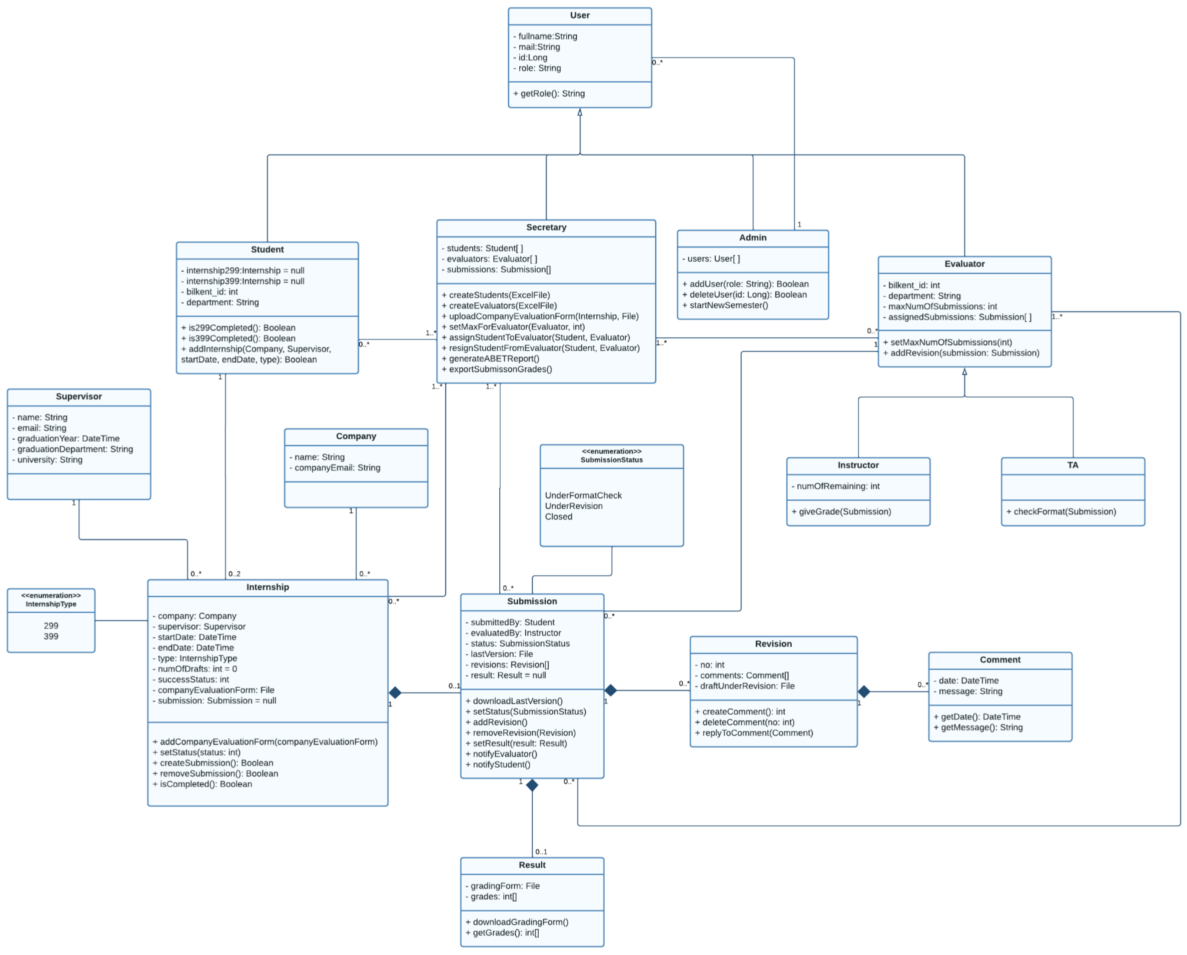
1. Turnitin is ordered to check the similarity
2. Turnitin creates a report
3. Turnitin returns the similarity report

Special Requirements: -

3.4.2. Use Case Model



3.4.3. Object and Class Model


User
- fullname:String
- mail:String
- id:Long
- role: String
+ getRole(): String
Student
- internship299:Internship = null
- internship399:Internship = null
- bilkent_id: int
- department: String
+ is299Completed(): Boolean
+ is399Completed(): Boolean
+ addInternship(Company, Supervisor, startDate, endDate, type): Boolean
Secretary
- students: Student[ ]
- evaluators: Evaluator[ ]
- submissions: Submission[]
+ createStudents(ExcelFile)
+ createEvaluators(ExcelFile)
+ uploadCompanyEvaluationForm(Internship, File)
+ setMaxForEvaluator(Evaluator, int)
+ assignStudentToEvaluator(Student, Evaluator)
+ resignStudentFromEvaluator(Student, Evaluator)
+ generateABETReport()
+ exportSubmissonGrades()
Admin
- users: User[ ]
+ addUser(role: String): Boolean
+ deleteUser(id: Long): Boolean
+ startNewSemester()
Evaluator
- bilkent_id: int
- department: String
- maxNumOfSubmissions: int
- assignedSubmissions: Submission[ ]
+ setMaxNumOfSubmissions(int)
+ addRevision(submission: Submission)
Supervisor
- name: String
- email: String
- graduationYear: DateTime
- graduationDepartment: String
- university: String
Company
- name: String
- companyEmail: String
<<enumeration>> SubmissionStatus
UnderFormatCheck
UnderRevision
Closed
Instructor
- numOfRemaining: int
+ giveGrade(Submission)
TA
+ checkFormat(Submission)
<<enumeration>> InternshipType
299
399
Internship
- company: Company
- supervisor: Supervisor
- startDate: DateTime
- endDate: DateTime
- type: InternshipType
- numOfDrafts: int = 0
- successStatus: int
- companyEvaluationForm: File
- submission: Submission = null
+ addCompanyEvaluationForm(companyEvaluationForm)
+ setStatus(status: int)
+ createSubmission(): Boolean
+ removeSubmission(): Boolean
+ isCompleted(): Boolean
Submission
- submittedBy: Student
- evaluatedBy: Instructor
- status: SubmissionStatus
- lastVersion: File
- revisions: Revision[]
- result: Result = null
+ downloadLastVersion()
+ setStatus(SubmissionStatus)
+ addRevision()
+ removeRevision(Revision)
+ setResult(result: Result)
+ notifyEvaluator()
+ notifyStudent()
Revision
- no: int
- comments: Comment[]
- draftUnderRevision: File
+ createComment(): int
+ deleteComment(no: int)
+ replyToComment(Comment)
Comment
- date: DateTime
- message: String
+ getDate(): DateTime
+ getMessage(): String
Result
- gradingForm: File
- grades: int[]
+ downloadGradingForm()
+ getGrades(): int[]
0..*
1
0..*
1..*
1..*
0..*
1
1..*
1
1..*
1..*
1
1
0..*
0..2
0..*
0..*
0..*
0..*
0..1
1
0..*
1
0..*
1
1
0..*
0..1

(For simplicity, getter and setter operations are removed from the diagram.)

* In the diagram, all of the associations are shown as bidirectional (navigation arrows are not provided). According to the course book (p.52), navigation is usually omitted during the analysis phase; such navigational decisions are made during object design [2].

User: It is the base class for all the user types in the application domain. A typical user in the current system should have a name, mail and a unique identifier. This base class is designed to be the parent of four user types in the application domain: Student, Secretary, Admin, Evaluator.

Secretary: Secretary is a User that coordinates the internship evaluation process. Therefore, the Secretary should have a reference to all of the students and evaluators. He/she should be able to initialize accounts for the students and graders, which is performed with the help of an excel sheet in the current system. This ability is represented as operations ‘createStudents()’ and ‘createEvaluators()’ in the system. Moreover, a secretary should be able to assign/resign students to/from evaluators, and this behavior is demonstrated in the model by ‘assignStudentToEvaluator()’ and ‘resignStudentFromEvaluator()’ operations. At the end of the semester, the Secretary should be able to create an ABET report from the submissions and their corresponding grades through generateABETReport() operation and export the grading forms as PDF using exportSubmissonGrades(). To create such reports, a Secretary should be associated with all of the submissions, indicated by the ‘submissions: Submission[]’ attribute.

Admin: Admin can be considered as the superuser of the system. He/she should be able to add or remove users from the system. In the model, Admin is provided two basic operations to do so. addUser(role) operation lets an Admin create an account for a user based on the role passed as the argument. Ideally, an Admin should only create an account for a Secretary, and the creation of the other accounts should be performed by the secretary. However, since an admin is a superuser, it would not be harmful to also allow him/her to create an account for an arbitrary user. deleteUser(id) operation lets an Admin to delete an account corresponding to the unique id passed as the argument. Deletion of a user account can only be accomplished by an admin in the system.

Evaluator: Evaluator is simply a user that reviews the student internship reports. However, in the current system, the review of the reports is divided into two main steps: format review and content review. There are subtle differences between format and content review. For example, format review is conducted by a TA while content review is conducted by an instructor from the department. Moreover, a content review definitely results in a grade (either pass or fail), but a report cannot be given a grade considering its format. Therefore, TA’s should not have a right to give grades to reports; they can only perform a format check and issue comments related to the format. On the other hand, department instructors should be able to conclude a report by giving it a grade (more specifically, by filling a form where grades for each evaluation criteria are given). To address such a distinction in the model, class Evaluator is divided further into two classes: Instructor and TA.

An Evaluator is responsible for a number of Submissions. Class ‘Submission’ will be explained later, but it basically refers to a report submission made by a Student. The maximum number of submissions to be assigned to an evaluator is indicated by the attribute ‘maxNumOfSubmissions’. Evaluators should also be able to add Revisions for a submission, which is indicated in the model by the operation ‘addRevision(submission)’.

Instructor: Instructor is an Evaluator with capability of giving a grade to a report. Instructors also have an attribute ‘numOfRemaining’ which helps secretary to track status of the instructors (whether an instructor is lagging).

TA: TA is an Evaluator that cannot give grades to the reports, but can issue a revision if the format of a report is problematic. Therefore, the TA class is provided with a checkFormat() operation in the model, which (for now) is the only functionality he/she can perform.

Student: A student performs two mandatory internships, each of which has its own Submission. An internship is defined through Company, Supervisor, startDate, endDate and type (299 or 399), all of which are passed as arguments to the addInternship() operation. Since a student can have at most two internships, instead of using an array of Internships, two separate attributes are used in the model (internship299 and internship399). A Student is associated with at most two Internships, and an Internship is associated with one and only one Student.

Internship: An internship has a status value indicating whether an internship (or related course) in general was successful, and a Submission that contains every drafts, comments, etc. The status of the internship is set after the evaluation of the Submission is performed. Since an internship can have at most one Submission (each Submission may have multiple revisions, but we refer to revision with another name) createSubmission() and removeSubmission() operations do not take any arguments, they simply set the value of the attribute ‘submission’.

Submission: A submission is generated when a student submits his/her first draft (to moodle in the current system). Each submission can have multiple revisions, indicated by the revisions array in the attribute list. A student's last report upload is held in an attribute called lastVersion, and this last version can be downloaded by

Students, TA's, Instructors, who have access to it (who may access a submission and download the last version is indicated by the association lines in the model). We provided a separation between Instructors and Evaluators and modeled their communication through Submissions. Whenever a student creates a submission, the corresponding evaluator is notified through notifyEvaluator(). Similarly, whenever a revision is requested by an evaluator, the corresponding student is notified through notifyStudent(). Each Internship has an association with a Result class.

A Submission must be associated with at least one Evaluator. One may question if that means allowing a submission to be graded by multiple instructors. This is not a usual case in the current situation, except the possibility that an instructor might not be able to grade a student due to an emergency. In this case, the Submission should be redirected to another Instructor. To support such a situation, a Submission should be associated with one or many Instructors.

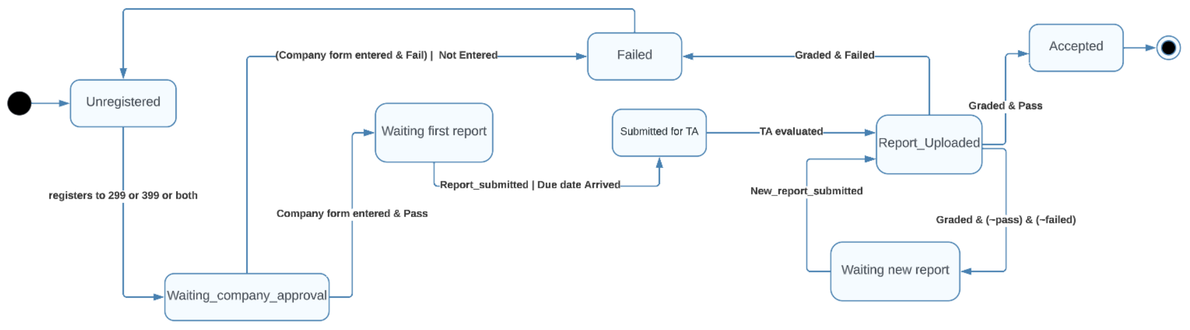
Result: A Result object contains a grading form which is filled by the instructor at the end, as well as an array of integers holding the grades for each section. That array of integers makes it easy to compute statistics of the internship results. The grading form can be downloaded using downloadGradingForm() operation.

Revision: A Revision object is constructed whenever an Evaluator issues addRevision() operation of the Submission. It should be storing the draft under the revision, and comments created by an evaluator. A comment can be created/deleted by an evaluator using createComment()/deleteComment() operations. A comment can be replied by a Student using replyToComment() operation. If the updated draft (the last version kept in submission) is not found satisfactory by the evaluator, new Revision can be created by addRevision() operation which is provided by the Submission class.

3.4.4. Dynamic Models

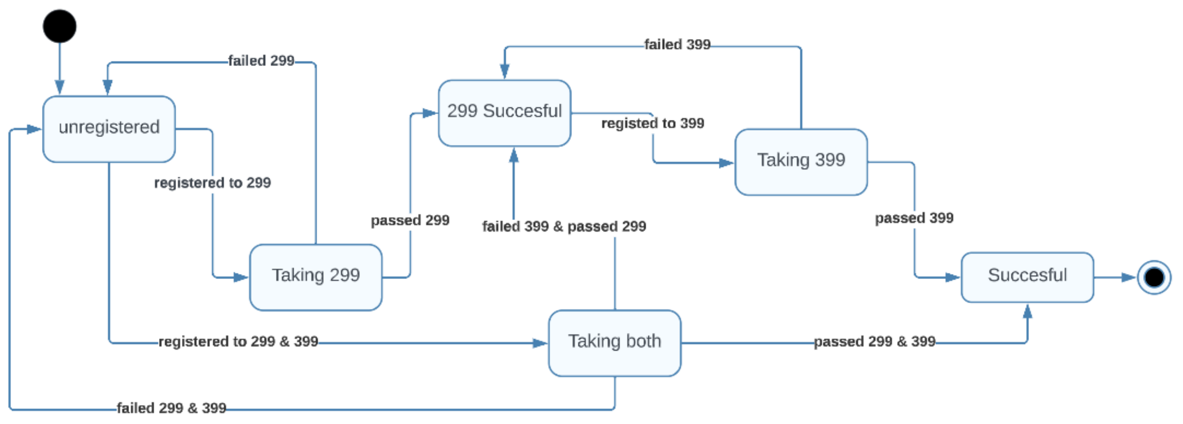
3.4.4.1 State Diagrams

Internship Report diagram



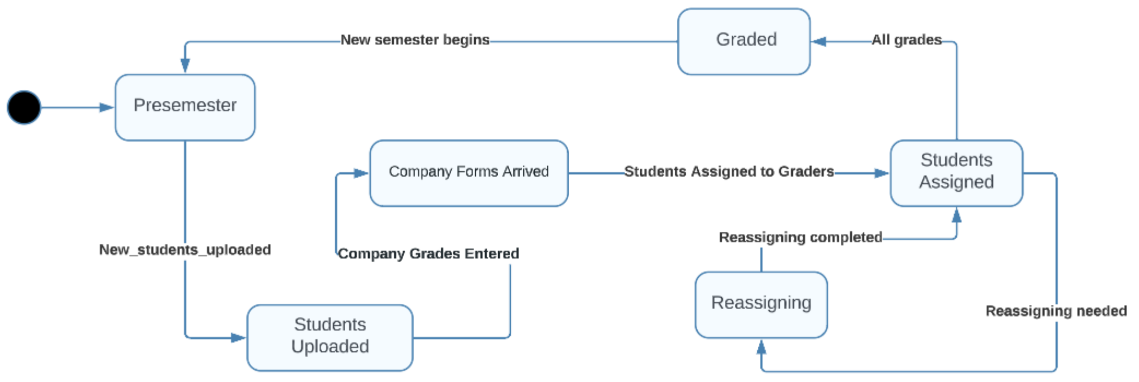
This diagram illustrates the process of internship report evaluation. Until students are added to the system by secretaries, they are at the state “Unregistered”. In this state they are not allowed to use systems’ functionalities except viewing their drafts of past internship reports if exists. After students are added to the system for the current semester, they are on the state “Waiting_company_approval” state. In this state they are able to view information about their internship and the progress bar. When a secretary enters student’s grades given by companies, report evaluation proceeds the next state depending on if grade is a pass or fail. If it is a pass, students proceed to “Waiting_first_report” state, which they can upload their first draft. Depending on the instructor's evaluation, students arrive at the state “Waiting__new_report” where they can view feedback and upload a new report accordingly. In the state “Report Uploaded”, students can no longer upload anything and they wait for instructors evaluation.

299/399 diagram



This diagram demonstrates how internships are classified as 299 and 399 courses. Since 299 is a prerequisite for 399, 299 have to be completed first. Also, students are allowed to take both courses in certain situations, that’s why we have a state called “Taking both” which both internships can be evaluated and viewed.

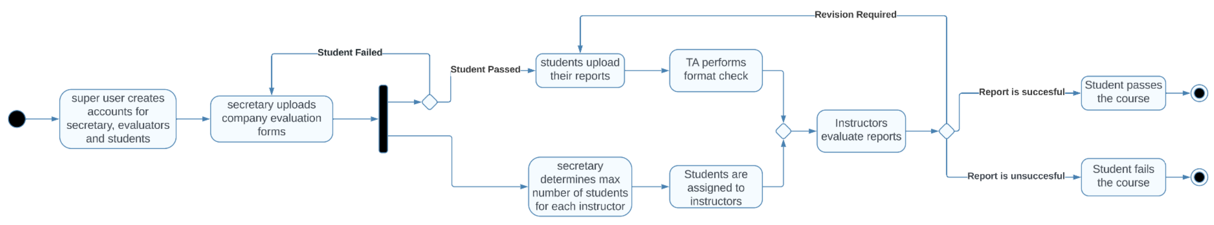
Secretary diagram



This diagram shows the process of a student managed by a secretary. Secretary first adds the student to the system, then enters company grades, then assigns him/her to an instructor. If the instructor is not able to complete the evaluation for some reason, the secretary reassigns an instructor to the student. After the completion of evaluation, the secretary collects the data. At the end of the semester admin can start the new semester.

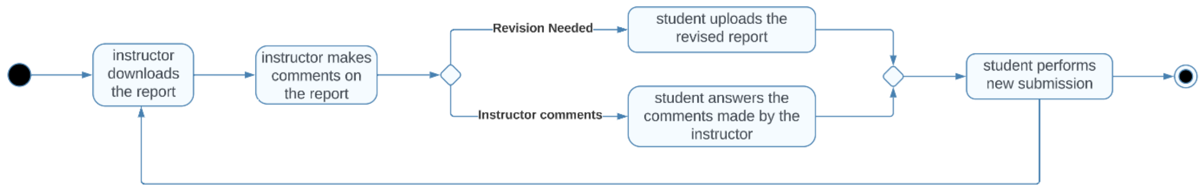
3.4.4.2 Activity Diagrams

General Activity Diagram


super user creates accounts for secretary, evaluators and students
secretary uploads company evaluation forms
Student Failed
Student Passed
students upload their reports
TA performs format check
Revision Required
secretary determines max number of students for each instructor
Students are assigned to instructors
Instructors evaluate reports
Report is succesful
Report is unsuccesful
Student passes the course
Student fails the course

General Activity of the system starts with the super user creating accounts for the secretary, evaluators, and students. After the super user creates accounts, the secretary uploads company evaluation forms of each student before passing to the new activity. After forms are successfully uploaded, two things happen. Secretary determines the maximum number of students for each instructor. After this action is done students are assigned to instructors. At the same time, if students had a passing grade they continue to upload their internship reports. If they have failed they need to return to the beginning and bring another company evaluation form. In the case, the student had a passing grade, the report is sent to the TA who does a format check. After TA completes the format check and students are assigned to the instructors, instructors evaluate students' reports. After this, a decision must be made from 3 different cases. If the report is successful, the student successfully completes the course. If the report requires revision, students return to the activity where they upload a revised version of the report. If the report is unsuccessful, the student fails the course.

Revision Activity Diagram

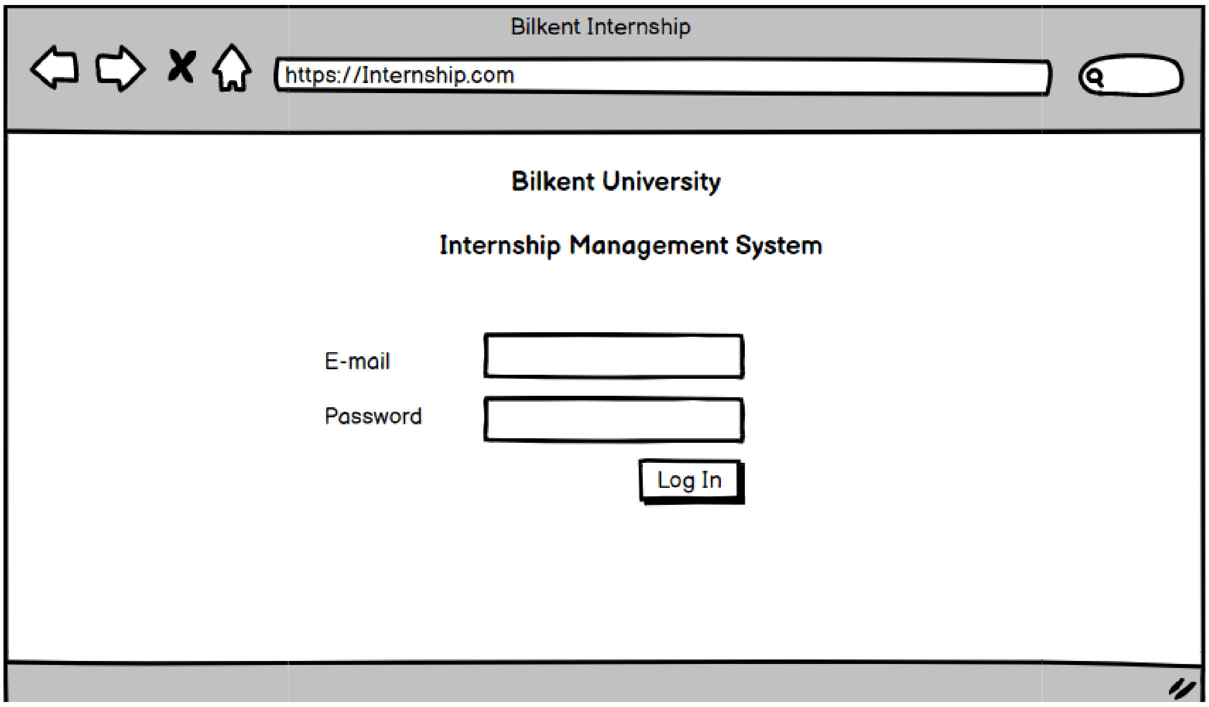


instructor downloads the report
instructor makes comments on the report
Revision Needed
student uploads the revised report
Instructor comments
student answers the comments made by the instructor
student performs new submission

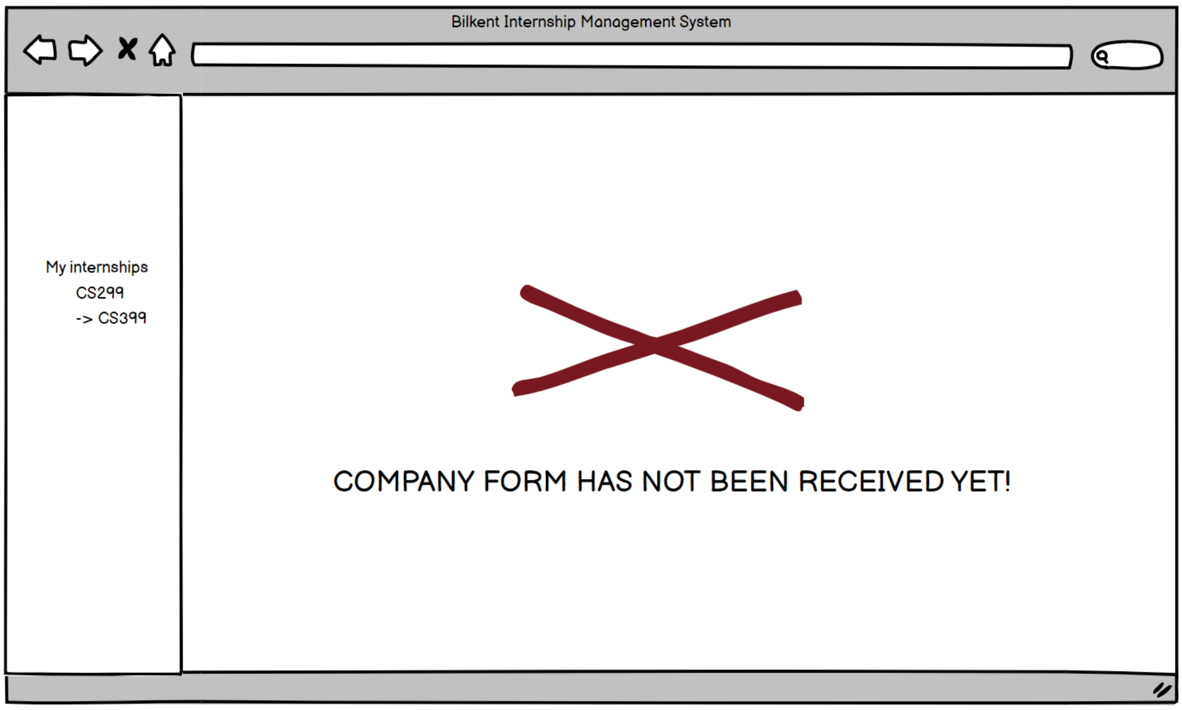
Once the revised report is uploaded, the instructor needs to download the report to make comments on the report. If the instructor comments that a revision is needed, the student uploads the revised report and performs a new submission. If the instructor's comments do not require a revision, the student answers the comments and performs new submission. In both cases, if the new submission needs to be evaluated, the instructor makes new comments, if the report is successful this time the student passes to the next phase and ends this activity.

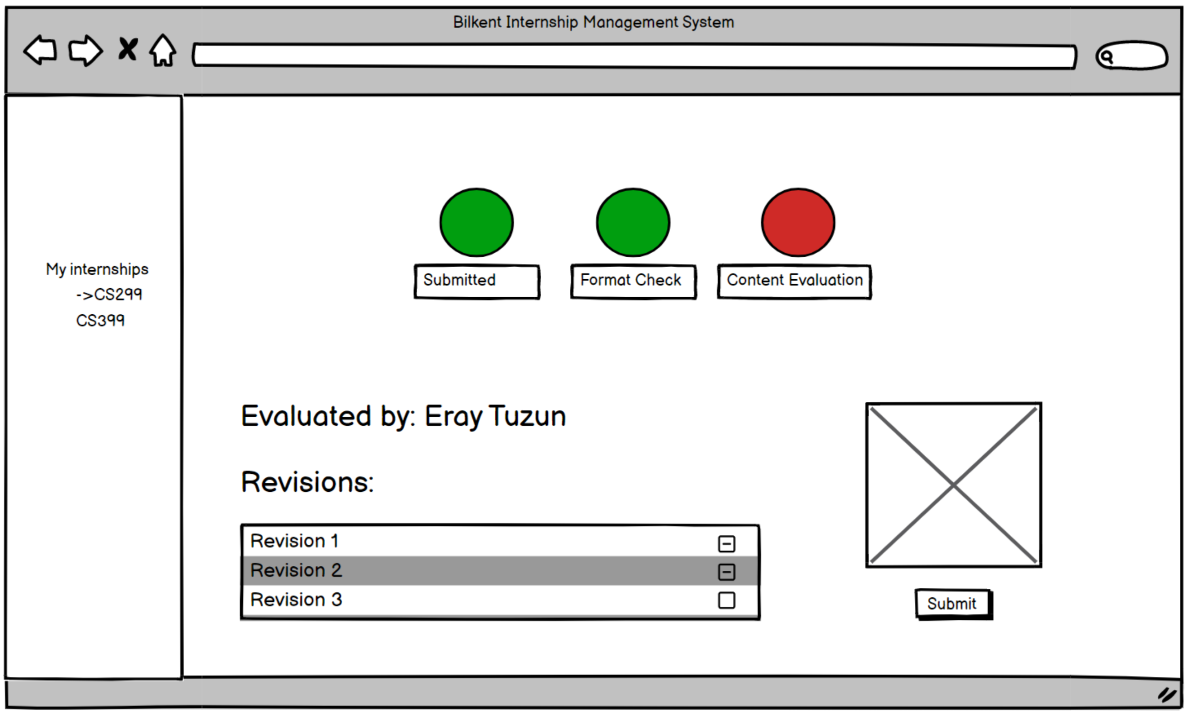
3.4.5 User interface - navigational paths and screen mock-ups

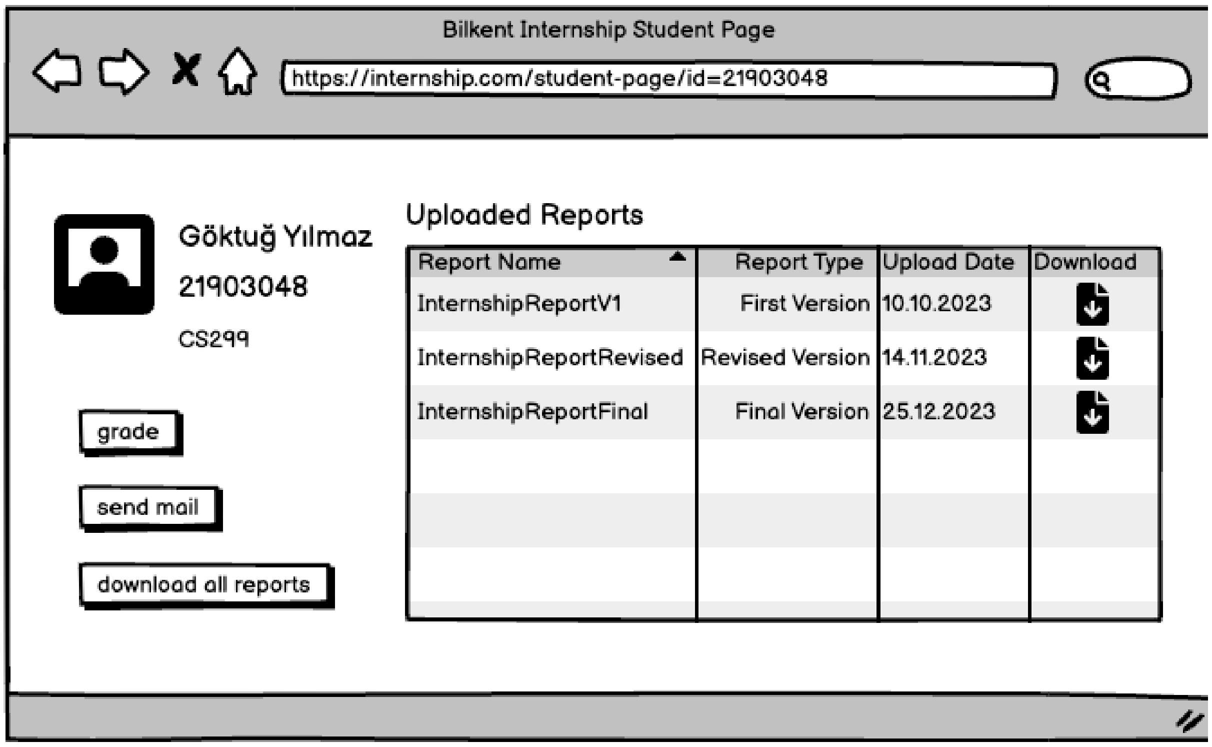
Login Screen



Student Screen


Bilkent Internship Management System
My internships
CS299
-> CS399
COMPANY FORM HAS NOT BEEN RECEIVED YET!





Bilkent Internship Student Page

https://internship.com/student-page/id=21903048

Göktuğ Yılmaz

21903048

CS299

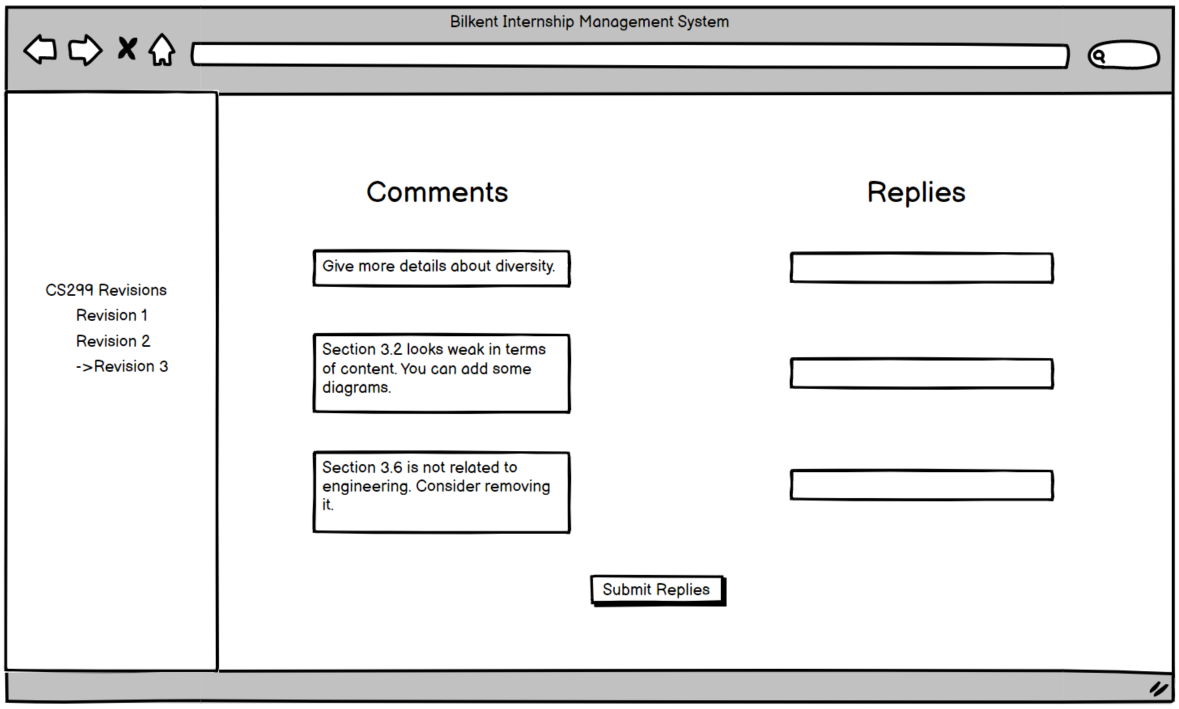
grade

send mail

download all reports

Uploaded Reports

Report Name	Report Type	Upload Date	Download
InternshipReportV1	First Version	10.10.2023	⬇
InternshipReportRevised	Revised Version	14.11.2023	⬇
InternshipReportFinal	Final Version	25.12.2023	⬇



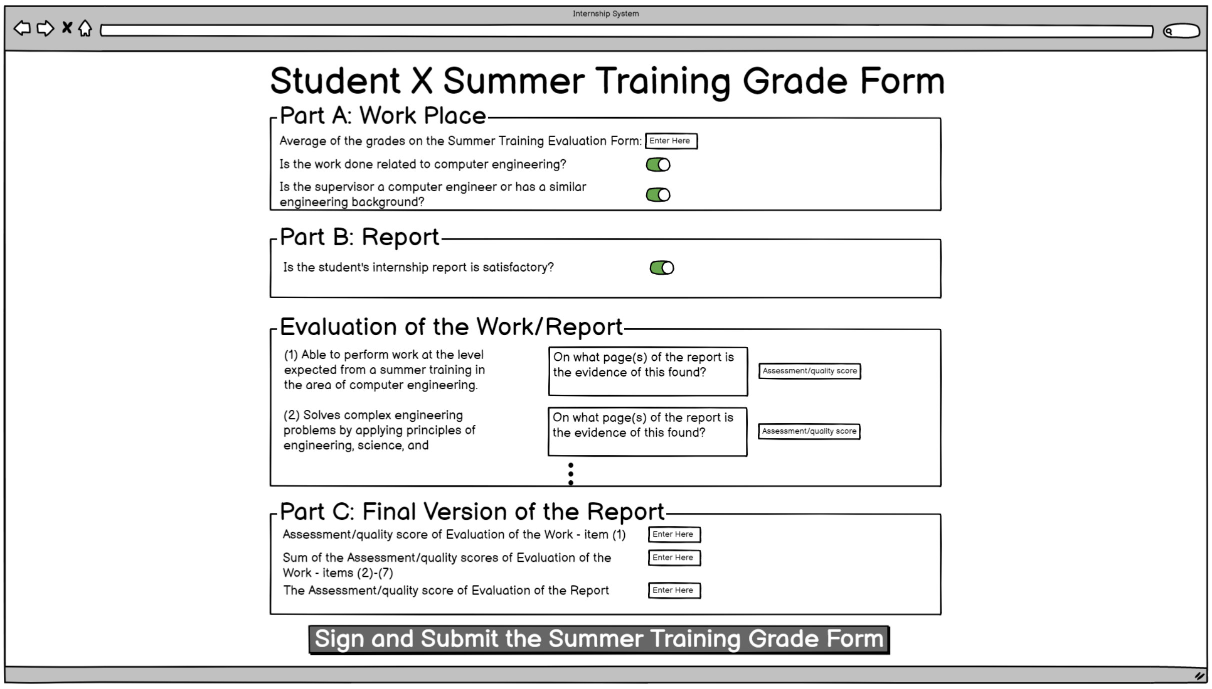
Bilkent Internship Management System

CS299 Revisions
Revision 1
Revision 2
->Revision 3

Comments	Replies
Give more details about diversity.	
Section 3.2 looks weak in terms of content. You can add some diagrams.	
Section 3.6 is not related to engineering. Consider removing it.	

Submit Replies

Instructor Screen


Internship System

Student X Summer Training Grade Form

Part A: Work Place

Average of the grades on the Summer Training Evaluation Form: Enter Here

Is the work done related to computer engineering?

Is the supervisor a computer engineer or has a similar engineering background?

Part B: Report

Is the student's internship report is satisfactory?

Evaluation of the Work/Report

(1) Able to perform work at the level expected from a summer training in the area of computer engineering. — On what page(s) of the report is the evidence of this found? — Assessment/quality score

(2) Solves complex engineering problems by applying principles of engineering, science, and — On what page(s) of the report is the evidence of this found? — Assessment/quality score

⋮

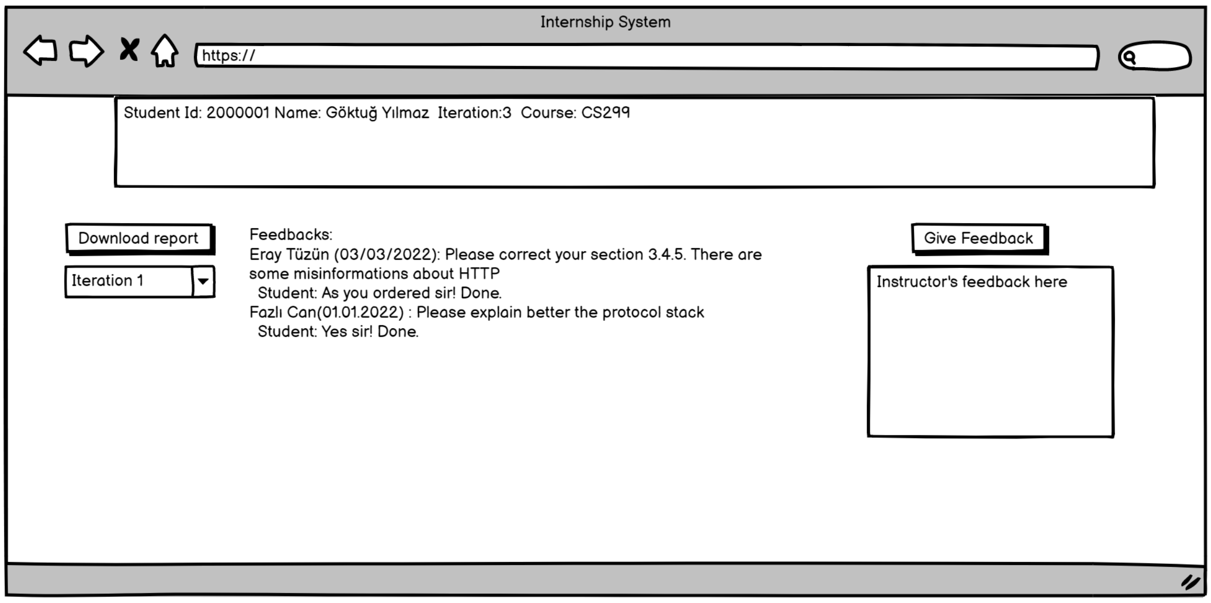
Part C: Final Version of the Report

Assessment/quality score of Evaluation of the Work - item (1) — Enter Here

Sum of the Assessment/quality scores of Evaluation of the Work - items (2)-(7) — Enter Here

The Assessment/quality score of Evaluation of the Report — Enter Here

Sign and Submit the Summer Training Grade Form


Internship System

https://

Student Id: 2000001 Name: Göktuğ Yılmaz Iteration:3 Course: CS299

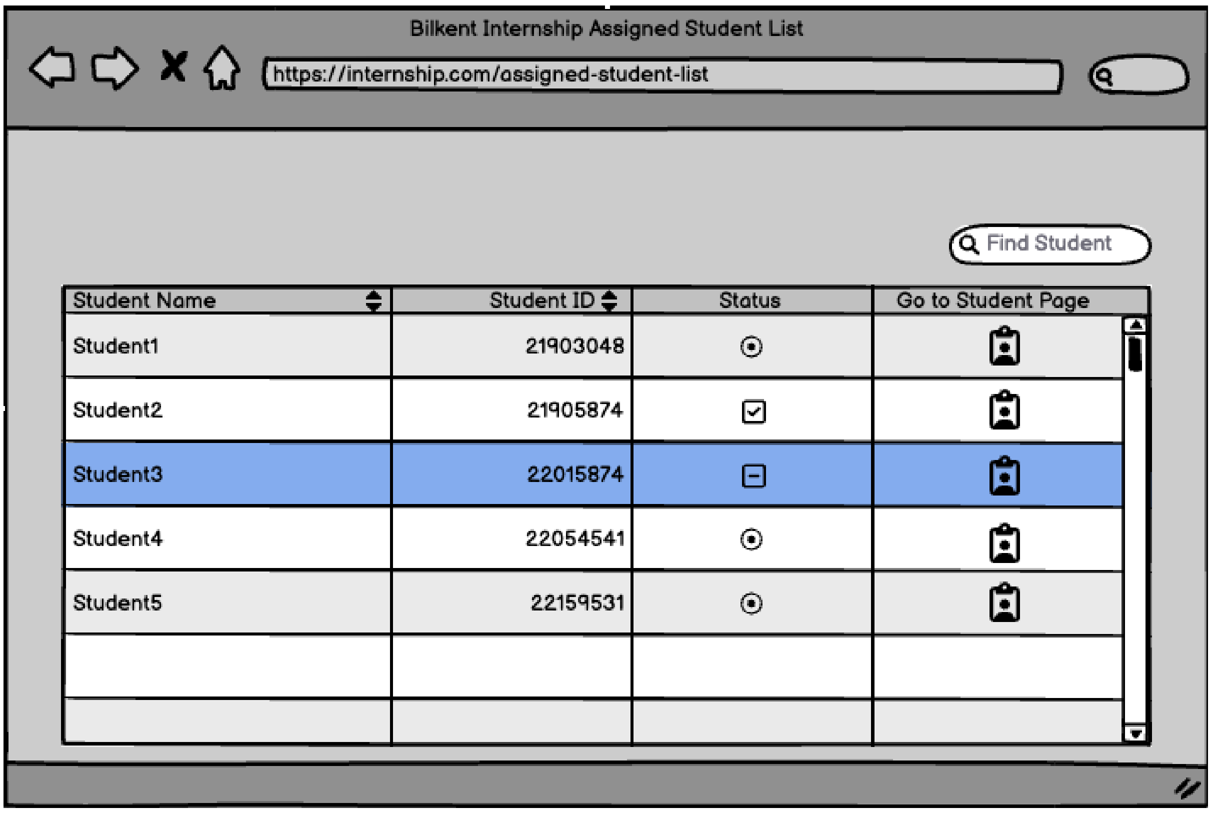
Download report

Iteration 1

Feedbacks:
Eray Tüzün (03/03/2022): Please correct your section 3.4.5. There are some misinformations about HTTP
Student: As you ordered sir! Done.
Fazlı Can(01.01.2022) : Please explain better the protocol stack
Student: Yes sir! Done.

Give Feedback

Instructor's feedback here



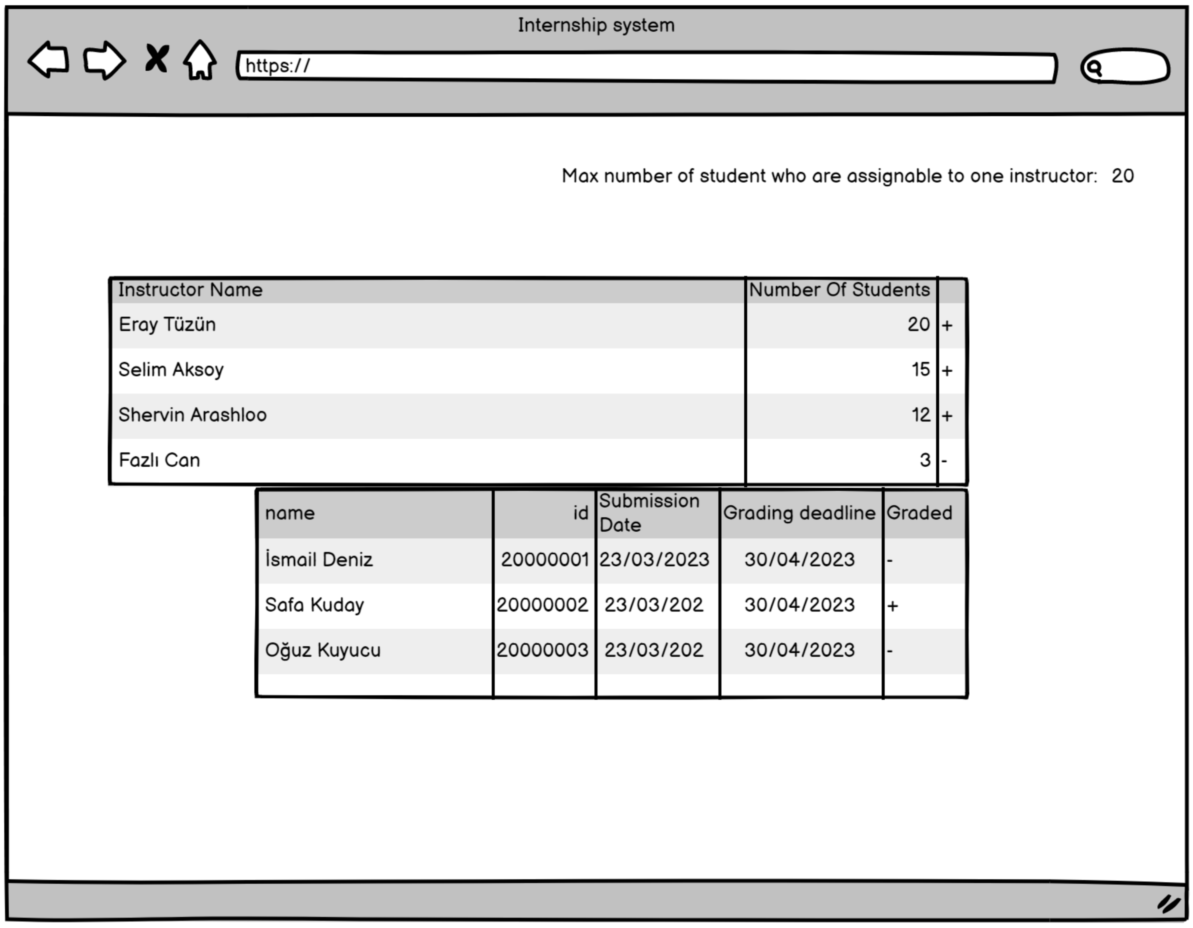
Bilkent Internship Assigned Student List

https://internship.com/assigned-student-list

Find Student

Student Name	Student ID	Status	Go to Student Page
Student1	21903048	⊙	
Student2	21905874	☑	
Student3	22015874	⊟	
Student4	22054541	⊙	
Student5	22159531	⊙	

Secretary Screen

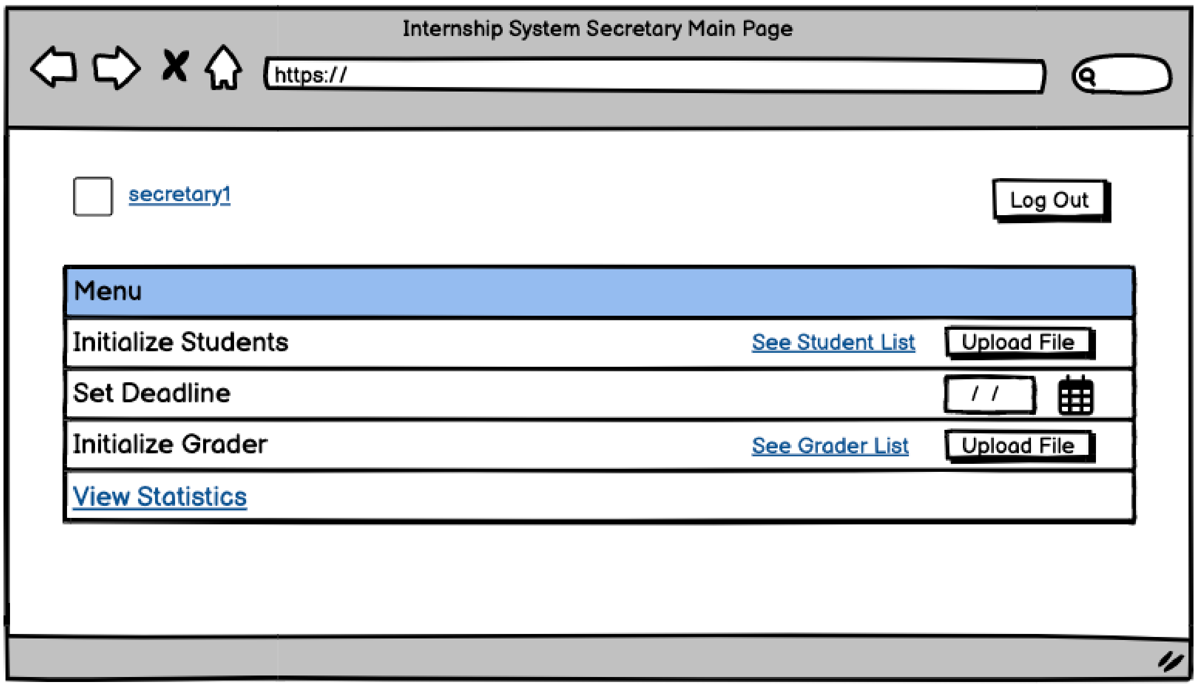

Internship system

https://

Max number of student who are assignable to one instructor: 20

Instructor Name	Number Of Students	
Eray Tüzün	20	+
Selim Aksoy	15	+
Shervin Arashloo	12	+
Fazlı Can	3	-

name	id	Submission Date	Grading deadline	Graded
İsmail Deniz	20000001	23/03/2023	30/04/2023	-
Safa Kuday	20000002	23/03/202	30/04/2023	+
Oğuz Kuyucu	20000003	23/03/202	30/04/2023	-

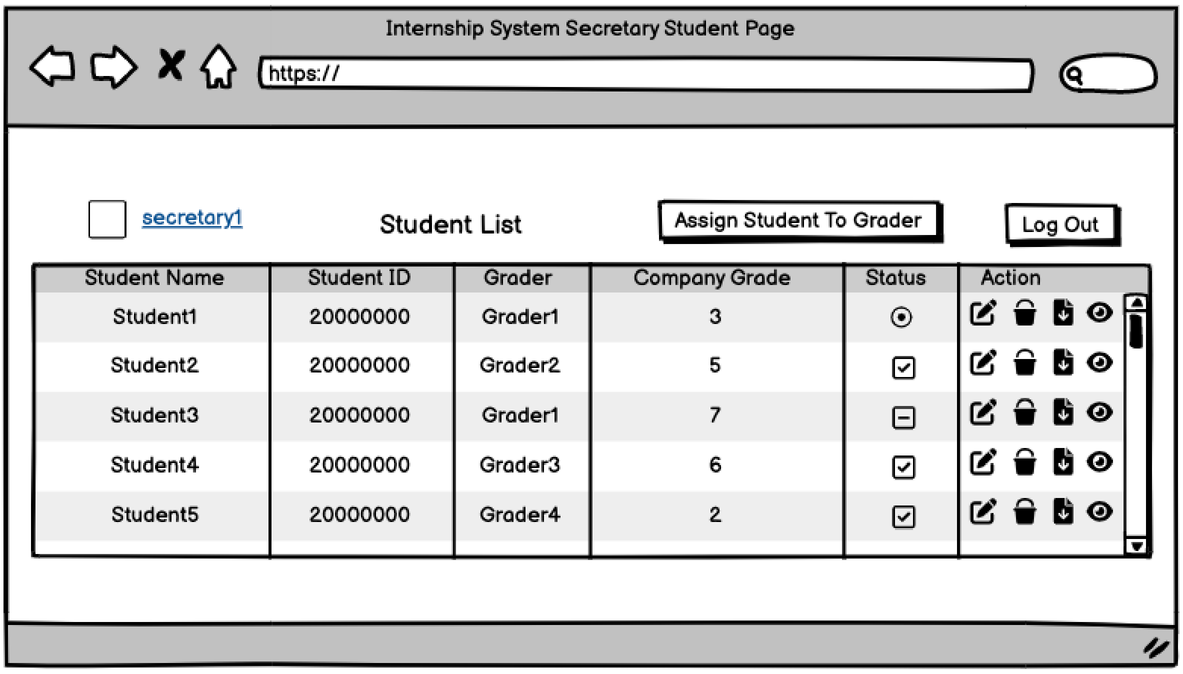

Internship System Secretary Main Page

https://

secretary1

Log Out

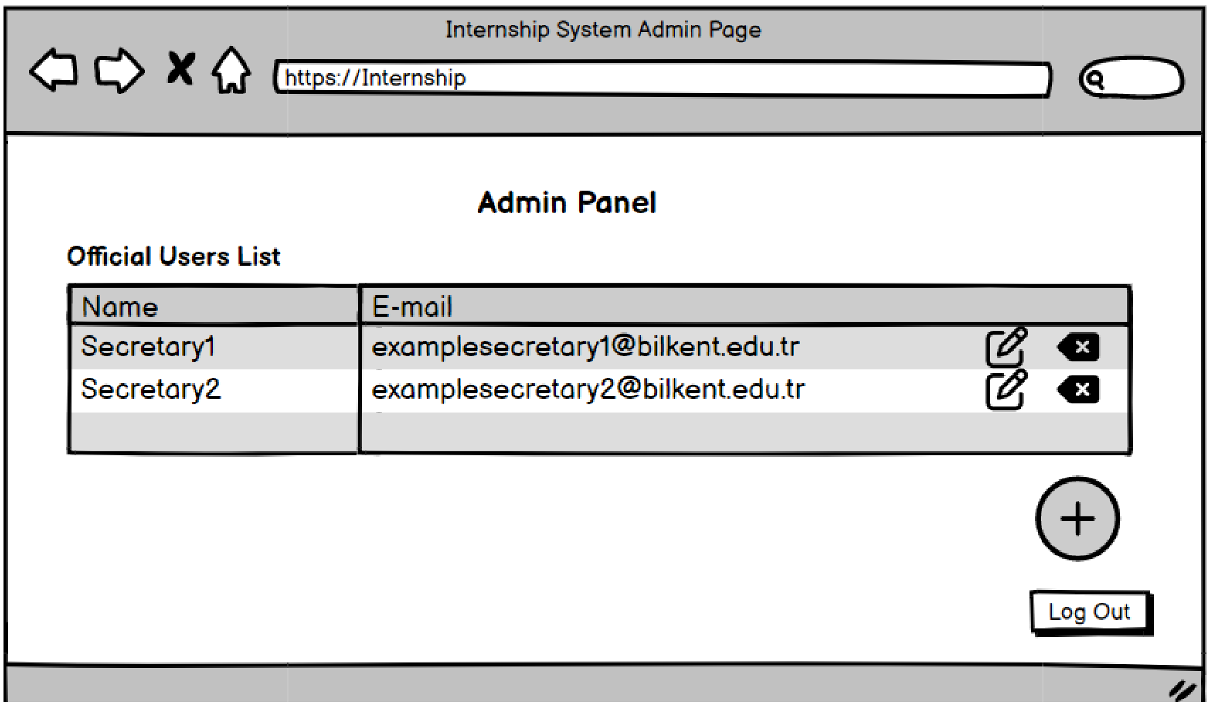
Menu		
Initialize Students	See Student List	Upload File
Set Deadline	/ /	
Initialize Grader	See Grader List	Upload File
View Statistics		



Student List

Student Name	Student ID	Grader	Company Grade	Status	Action
Student1	20000000	Grader1	3	◉	
Student2	20000000	Grader2	5	☑	
Student3	20000000	Grader1	7	⊟	
Student4	20000000	Grader3	6	☑	
Student5	20000000	Grader4	2	☑	

Admin Screen



Admin Panel

Official Users List

Name	E-mail
Secretary1	examplesecretary1@bilkent.edu.tr
Secretary2	examplesecretary2@bilkent.edu.tr

4. References

[1] “Summer training,” *Bilkent University Faculty of Engineering*. [Online]. Available: http://mf.bilkent.edu.tr/?page_id=844. [Accessed: 31-Mar-2023].

[2] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering, Using UML, Patterns, and Java*, 3rd ed. Pearson, 2010.