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 Internship initialization page
- Department secretary’s Instructor initialization page
- Student’s main page to keep track of his report/s status
- Match instructors with students
- 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 in order to fully meet the expectations that come with an entirely new system. This means designing the web page with a clear and intuitive layout, consistent design elements, and minimal as possible while being a functional interface. To enhance user experience, we will ensure that all functionalities in our web application can be accessed with a maximum of 6 clicks. Also, the average active time it takes for users to complete simple tasks and find specific information within the app should not take more than 10 minutes in average. For more complex tasks like answering feedback comments average time should be less than 20. For readability, font sizes of headings should be more than 18 and for other lines it should be more than 12. Moreover, only the actions that are specific to the user type will be displayed, further reducing any unnecessary elements on the interface. For example, the students are only allowed to see their internship and report information, the evaluators are only allowed to see their list of students and their work, etc. This approach will simplify the user experience, enabling users to easily find and access the functionalities they need without encountering any difficulties.

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 Pseudo Requirements

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

3.3 System models

3.3.1 Use Case Textual descriptions

3.3.1.1 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 a 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 submits their report
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: -

3.3.1.2 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 Grade Form PDF

Participating Actor: Grader

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

Exit Condition: The grader has generated the final grade form 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

3.3.1.3 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 all student to their graders

Participating Actor: Secretary

Entry Condition: The student list and the grader list are initialized

Exit Condition: The graders are assigned to their students

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 && the total of the max number of assignable students of graders must be greater or equal to the total internships.

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: -

3.3.1.4 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: Delete Department Secretaries’ accounts

Participating Actor: Admin

Entry Condition: The user should have “admin” role

Exit Condition: Department Secretary’s account is deleted

The flow of events:

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

Special Requirements: An account should exist to be deleted

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: -

3.3.1.5 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:-

3.3.1.6 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: -

3.3.2. Use Case Model

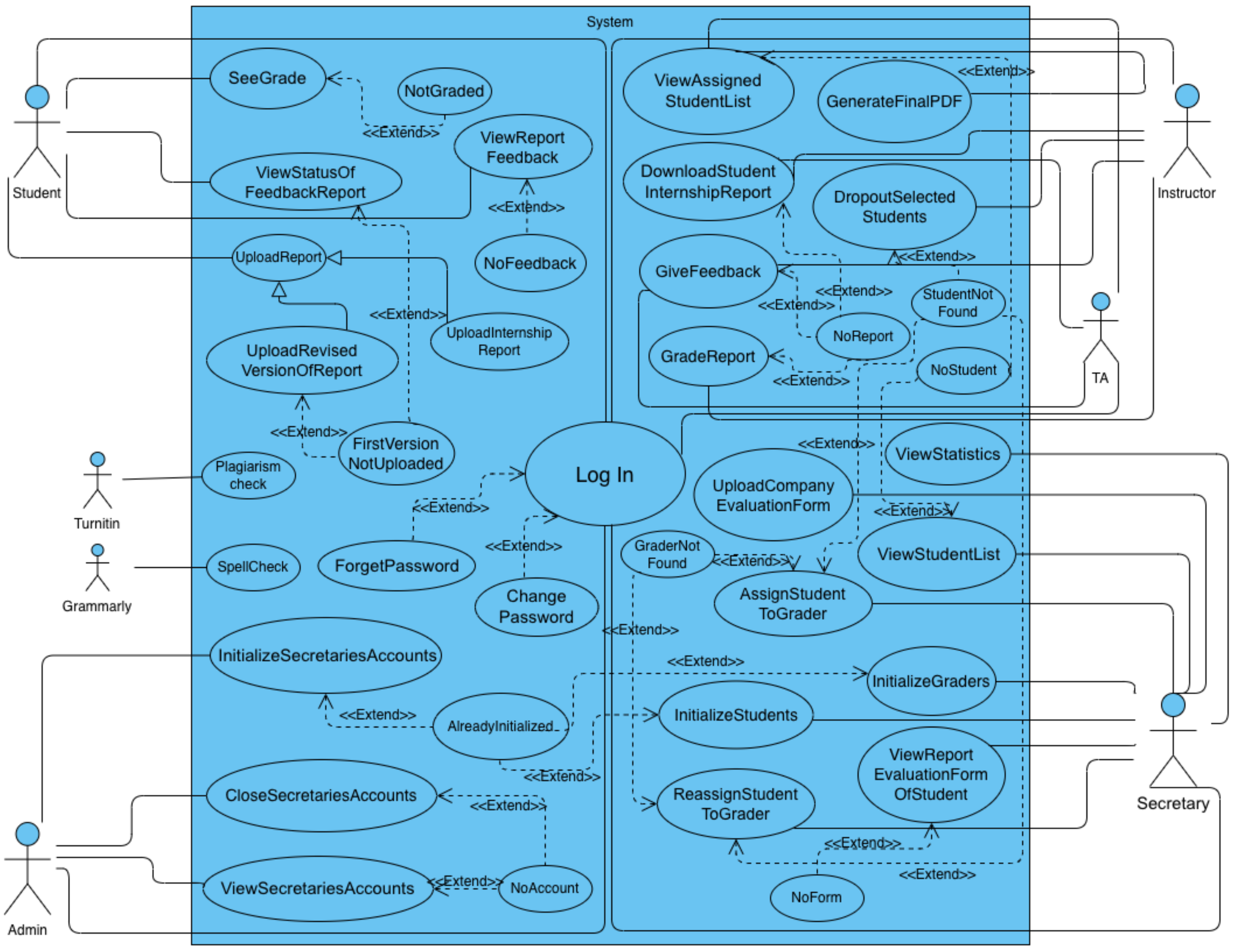


Figure 1: Use case diagram

3.3.3. Object and Class Model

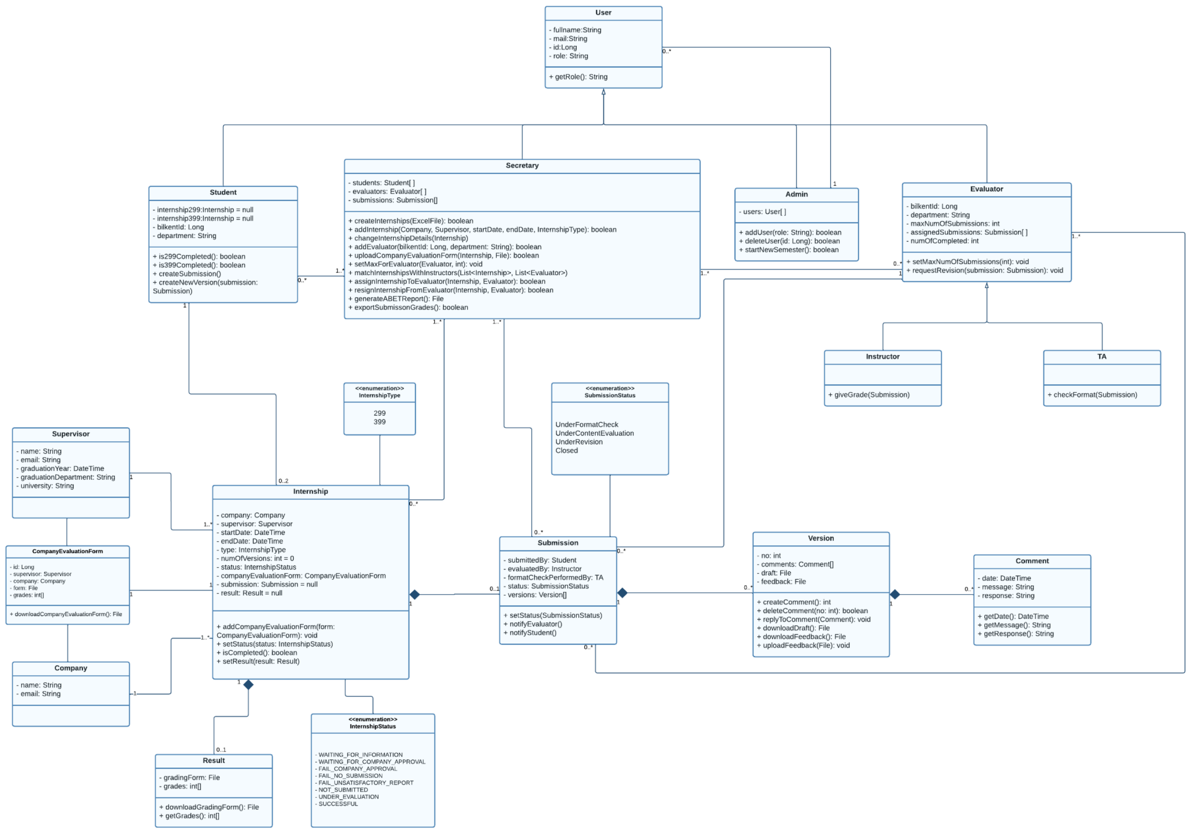

User
- fullname:String
- mail:String
- id:Long
- role: String
+ getRole(): String
Student
- internship299:Internship = null
- internship399:Internship = null
- bilkentId: Long
- department: String
+ is299Completed(): boolean
+ is399Completed(): boolean
+ createSubmission()
+ createNewVersion(submission: Submission)
Secretary
- students: Student[ ]
- evaluators: Evaluator[ ]
- submissions: Submission[]
+ createInternships(ExcelFile): boolean
+ addInternship(Company, Supervisor, startDate, endDate, InternshipType): boolean
+ changeInternshipDetails(Internship)
+ addEvaluator(bilkentId: Long, department: String): boolean
+ uploadCompanyEvaluationForm(Internship, File): boolean
+ setMaxForEvaluator(Evaluator, int): void
+ matchInternshipsWithInstructors(List<Internship>, List<Evaluator>)
+ assignInternshipToEvaluator(Internship, Evaluator): boolean
+ resignInternshipFromEvaluator(Internship, Evaluator): boolean
+ generateABETReport(): File
+ exportSubmissonGrades(): boolean
Admin
- users: User[ ]
+ addUser(role: String): boolean
+ deleteUser(id: Long): boolean
+ startNewSemester(): boolean
Evaluator
- bilkentId: Long
- department: String
- maxNumOfSubmissions: int
- assignedSubmissions: Submission[ ]
- numOfCompleted: int
+ setMaxNumOfSubmissions(int): void
+ requestRevision(submission: Submission): void
Instructor
+ giveGrade(Submission)
TA
+ checkFormat(Submission)
<<enumeration>> InternshipType
299
399
<<enumeration>> SubmissionStatus
UnderFormatCheck
UnderContentEvaluation
UnderRevision
Closed
Supervisor
- name: String
- email: String
- graduationYear: DateTime
- graduationDepartment: String
- university: String
CompanyEvaluationForm
- id: Long
- supervisor: Supervisor
- company: Company
- form: File
- grades: int[]
+ downloadCompanyEvaluationForm(): File
Company
- name: String
- email: String
Internship
- company: Company
- supervisor: Supervisor
- startDate: DateTime
- endDate: DateTime
- type: InternshipType
- numOfVersions: int = 0
- status: InternshipStatus
- companyEvaluationForm: CompanyEvaluationForm
- submission: Submission = null
- result: Result = null
+ addCompanyEvaluationForm(form: CompanyEvaluationForm): void
+ setStatus(status: InternshipStatus)
+ isCompleted(): boolean
+ setResult(result: Result)
Submission
- submittedBy: Student
- evaluatedBy: Instructor
- formatCheckPerformedBy: TA
- status: SubmissionStatus
- versions: Version[]
+ setStatus(SubmissionStatus)
+ notifyEvaluator()
+ notifyStudent()
Version
- no: int
- comments: Comment[]
- draft: File
- feedback: File
+ createComment(): int
+ deleteComment(no: int): boolean
+ replyToComment(Comment): void
+ downloadDraft(): File
+ downloadFeedback(): File
+ uploadFeedback(File): void
Comment
- date: DateTime
- message: String
- response: String
+ getDate(): DateTime
+ getMessage(): String
+ getResponse(): String
Result
- gradingForm: File
- grades: int[]
+ downloadGradingForm(): File
+ getGrades(): int[]
<<enumeration>> InternshipStatus
- WAITING_FOR_INFORMATION
- WAITING_FOR_COMPANY_APPROVAL
- FAIL_COMPANY_APPROVAL
- FAIL_NO_SUBMISSION
- FAIL_UNSATISFACTORY_REPORT
- NOT_SUBMITTED
- UNDER_EVALUATION
- SUCCESSFUL
0..*
1
1..*
0..2
0..1

Figure 2: Class diagram

(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 internships and graders, which is performed with the help of an excel sheet in the current system. This ability is represented as operations ‘createInternships()’ and ‘createEvaluators()’ in the system. Moreover, a secretary should be able to assign/resign internships to/from evaluators, and this behavior is demonstrated in the model by ‘assingInternshipToEvaluator()’ and ‘resignInternshipFromEvaluator()’ 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 Internships. 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 request a revision for a submission, which is indicated in the model by the operation ‘requestRevision(submission)’. Evaluators also have an attribute ‘numOfCompleted’ which helps secretary to track status of the evaluators (whether an instructor is lagging).

Instructor: Instructor is an Evaluator with capability of giving a grade to a report.

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. 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: At the beginning of the semester, internships are initialized by a secretary using an excel file. Then a company evaluation form is uploaded, which may either be obtained through a letter or email. A company evaluation form (indicated in the diagram as CompanyEvaluationForm) also contains a grades array, which represents the grades given by a supervisor. An internship has a status enumeration indicating the process, and a Submission that contains every drafts, comments, etc. Each internship can have at most one Submission (each Submission may have multiple versions, but we refer to versions with another name). Each Internship has an association with a Result class, which contains a grading form filled by an instructor.

Submission: A submission is generated when a student submits his/her first draft (to moodle in the current system). Each submission can have multiple versions, indicated by the versions array in the attribute list. 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().

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.

Version: A version object is constructed whenever an Evaluator issues requestRevision() operation and as a result new report upload is performed by a student. It should be storing the draft, 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.

3.3.4. Dynamic Models

3.3.4.1 State Diagrams

Internship Report diagram

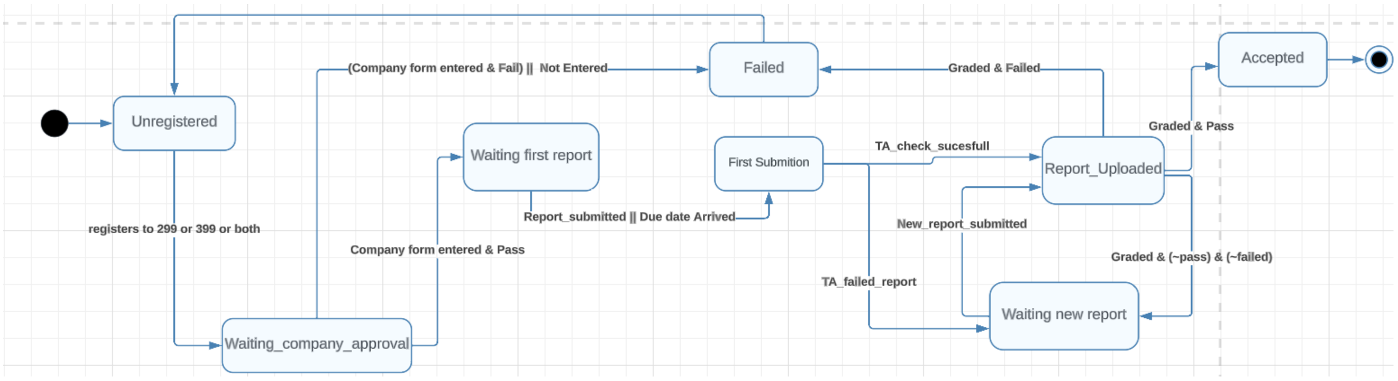


Figure 3: Internship Report Diagram

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 they exist. After students are added to the system for the current semester, they are in the “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 to the next state depending on if the grade is a pass or fail. If it is a pass, students proceed to the “Waiting_first_report” state, where 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.

Taking 299/399 and Prerequisite Rules Diagram

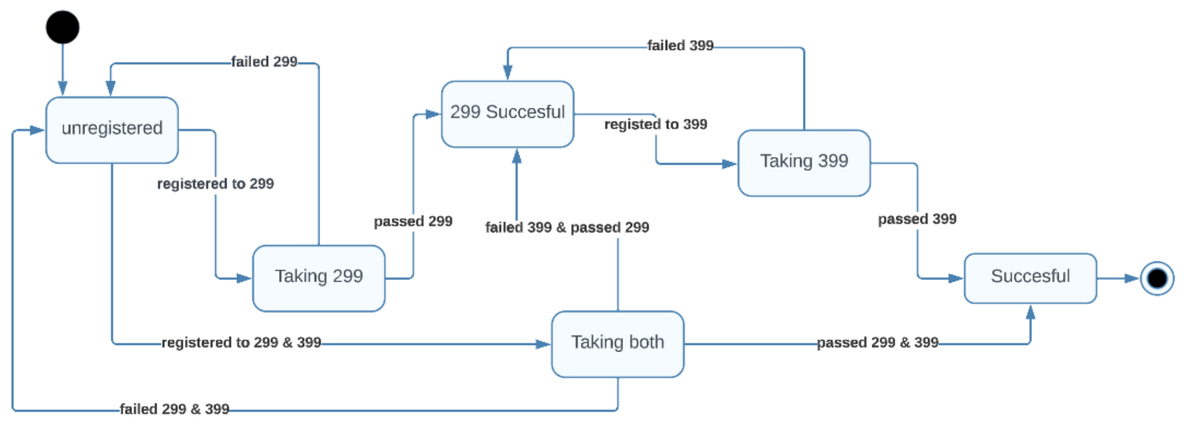


Figure 4: Taking 299/399 and Prerequisite Rules Diagram

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" in which both internships can be evaluated and viewed.

State Diagram: Secretaries' Management of An Internship Process

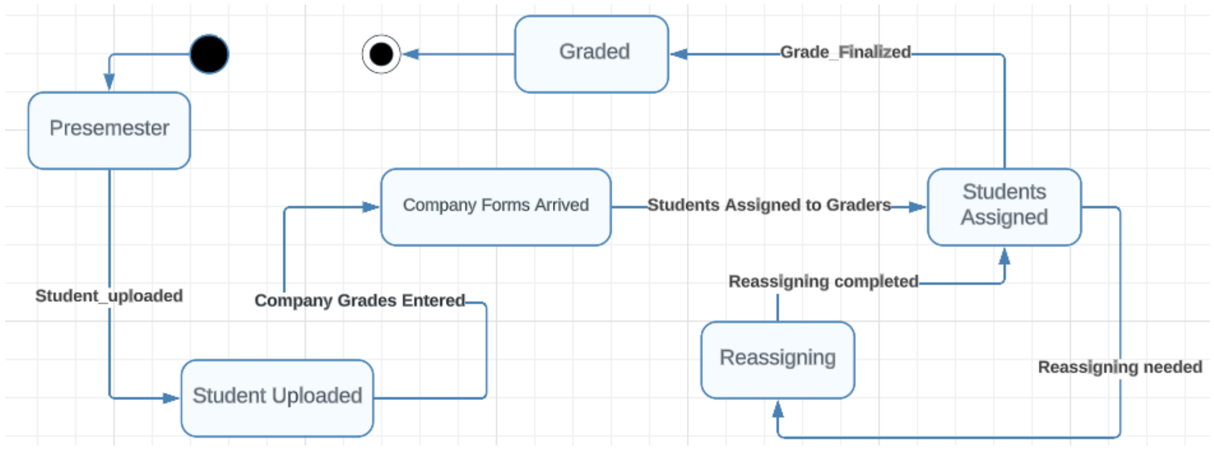


Figure 5: Secretaries' Management of An Internship Process

This diagram shows the process of a student's internship 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.

3.3.4.2 Activity Diagrams

General Activity Diagram

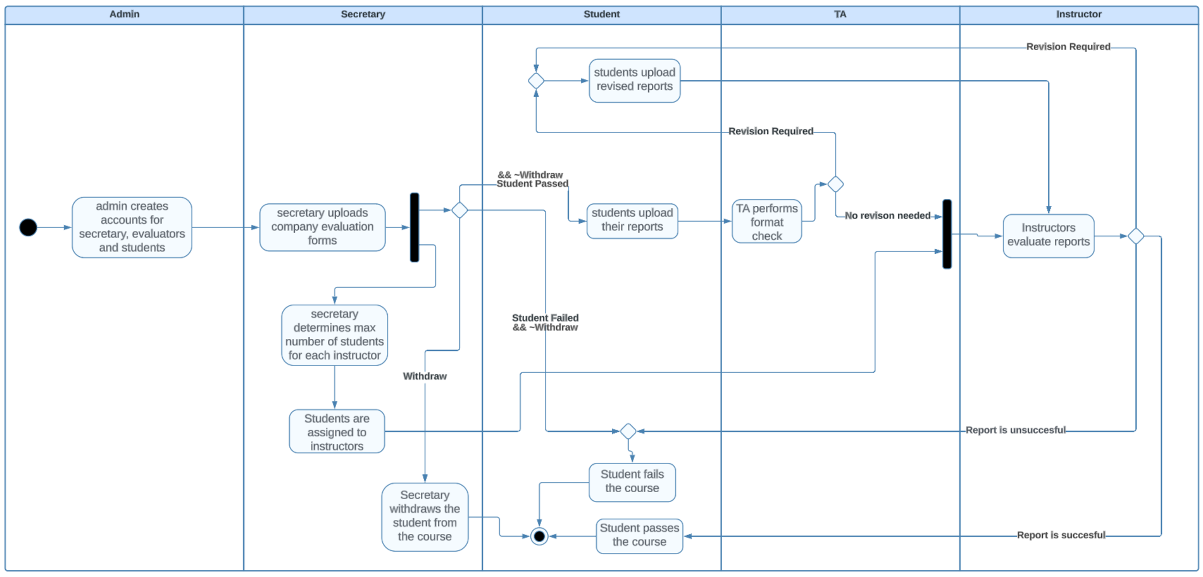

Admin
Secretary
Student
TA
Instructor
admin creates accounts for secretary, evaluators and students
secretary uploads company evaluation forms
secretary determines max number of students for each instructor
Students are assigned to instructors
Withdraw
Secretary withdraws the student from the course
&& ~Withdraw Student Passed
Student Failed && ~Withdraw
students upload revised reports
students upload their reports
Revision Required
Student fails the course
Student passes the course
TA performs format check
No revison needed
Instructors evaluate reports
Revision Required
Report is unsuccesful
Report is succesful

Figure 6: General Activity Diagram

The general Activity of the system starts with the admin creating accounts for the secretary, evaluators, and students. After the admin creates accounts, the secretary uploads company evaluation forms of each student before passing to the new activity. After forms are successfully uploaded, the control flow diverges into two parallel flows. At one flow secretary determines the maximum number of students for each instructor. After this action is done, students are assigned to instructors, and this parallel flow is joined and synchronized with the other. At the same time, at the other flow, a decision is made: If students had a passing grade and didn't withdraw from the course, they continue to upload their internship reports. If they had a failing grade, then they failed the course. If the student withdraws from the course secretary will make the withdrawal. In the case the student has a passing grade, the report is sent to the TA, who does a format check. After TA completes the format check, students are either required to upload a new version of the report (which will later be evaluated by the instructor) or don't need to revise their report and continue. After this event, this parallel flow is also synched and joins the other flow. Then 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. After withdrawal, passing, or failing, general activity reaches a finish.

Revision Activity Diagram

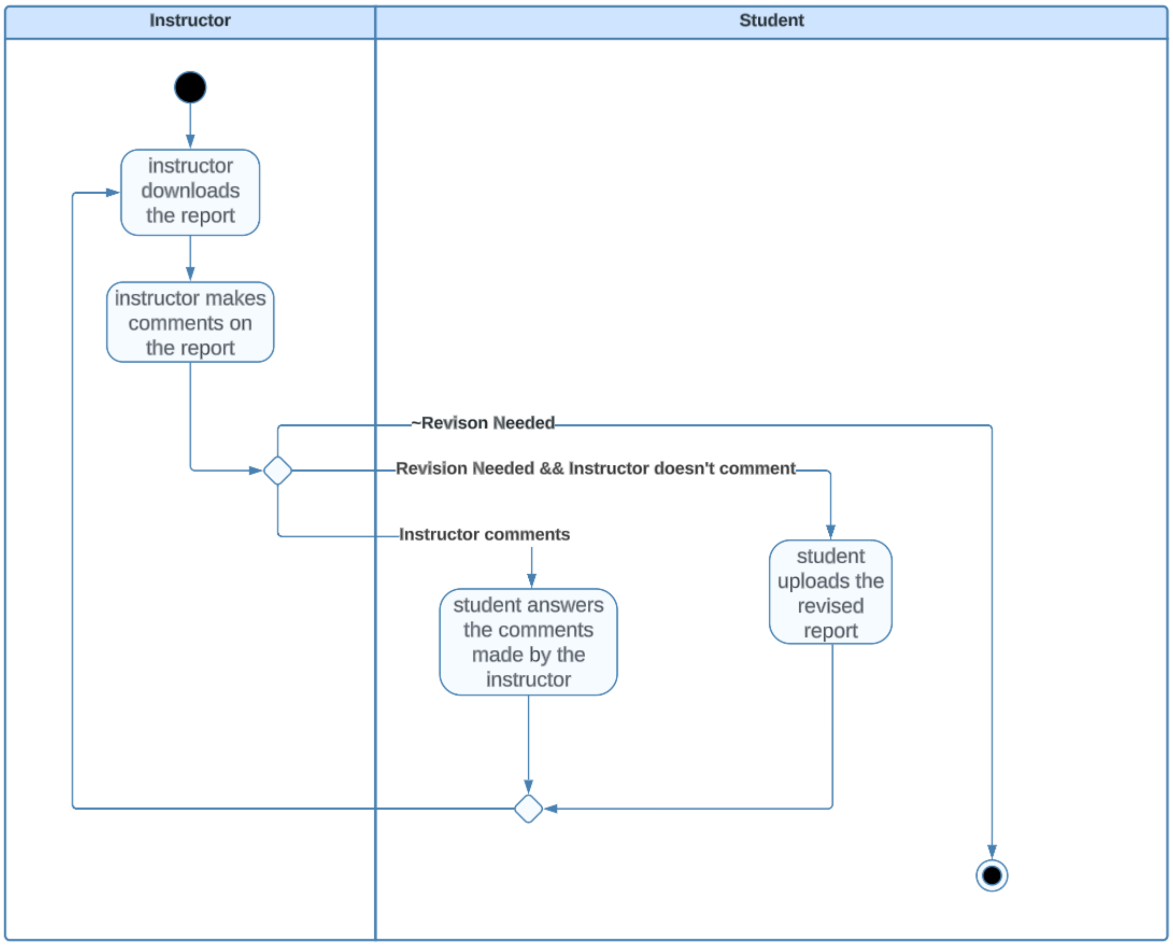

Instructor
Student
instructor downloads the report
instructor makes comments on the report
~Revison Needed
Revision Needed && Instructor doesn't comment
Instructor comments
student answers the comments made by the instructor
student uploads the revised report

Figure 7: Revision Activity Diagram

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 and the instructor doesn't comment, the student uploads the revised report and performs a new submission. If the instructor's comments and does not require a revision, the student answers the comments and performs new submission. In both cases, if the new submission needs to be downloaded and evaluated again and activity starts again. If the revision is not needed then the revision activities are over and the report is not revised again.

3.3.5 User interface - navigational paths and screen mock-ups

Login Screen

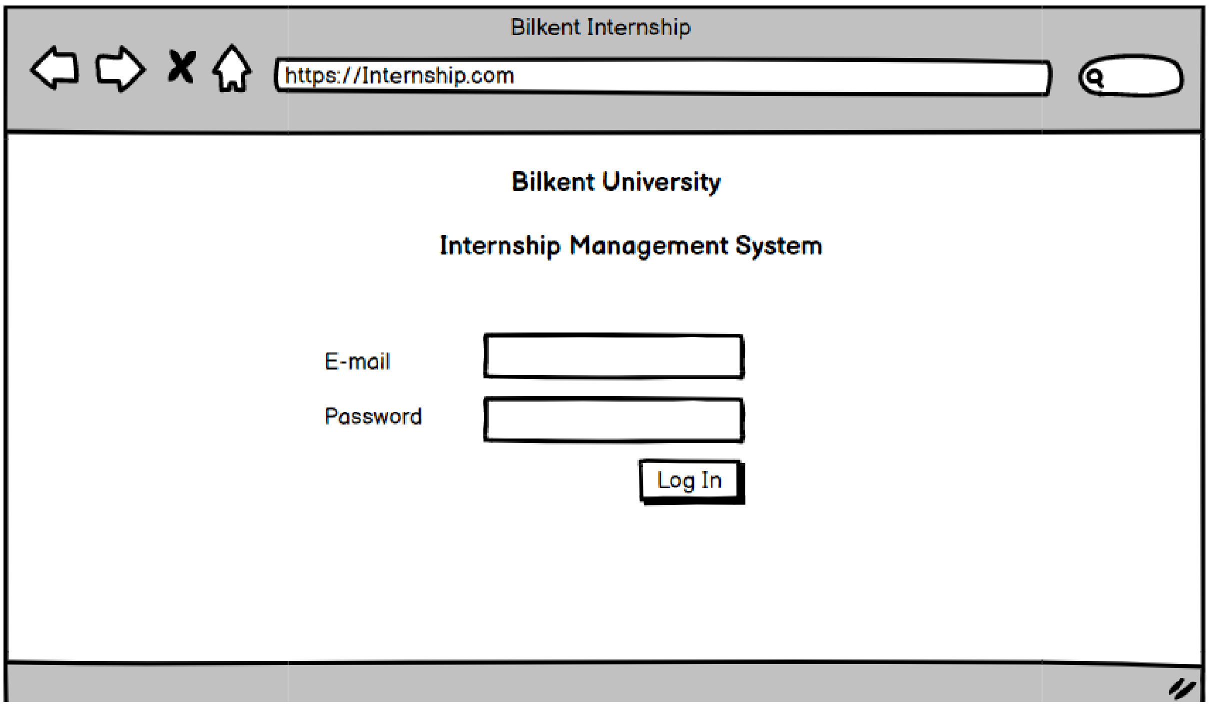
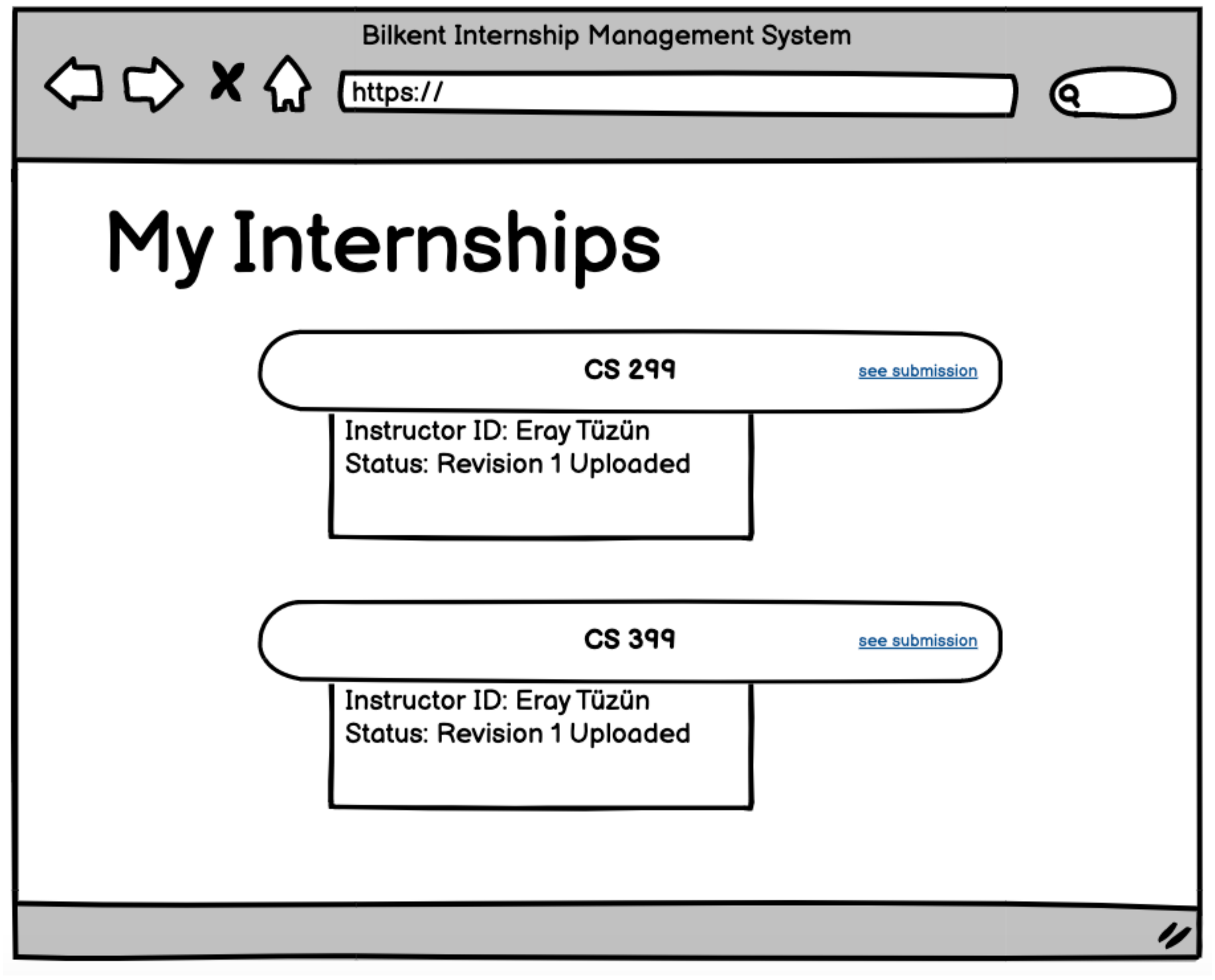


Figure 8: Login screen

Student Screen


Bilkent Internship Management System

https://

My Internships

CS 299 see submission

Instructor ID: Eray Tüzün
Status: Revision 1 Uploaded

CS 399 see submission

Instructor ID: Eray Tüzün
Status: Revision 1 Uploaded

Figure 9: Student Homepage Screen

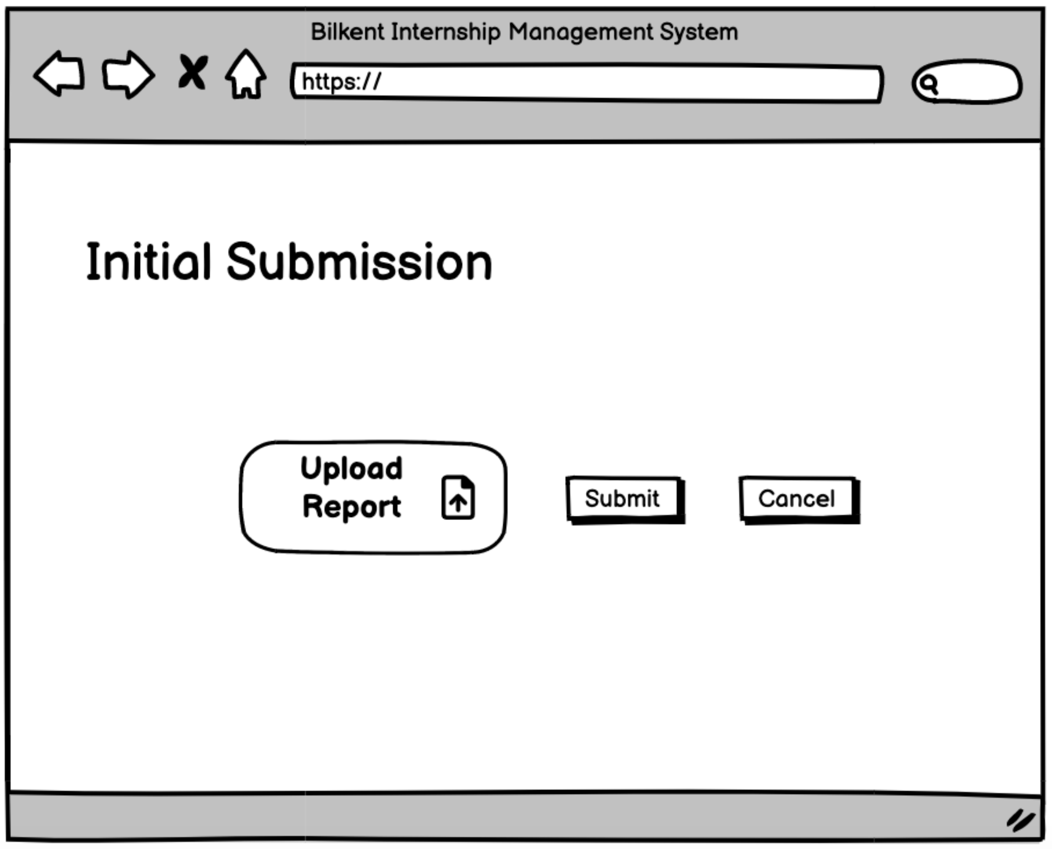
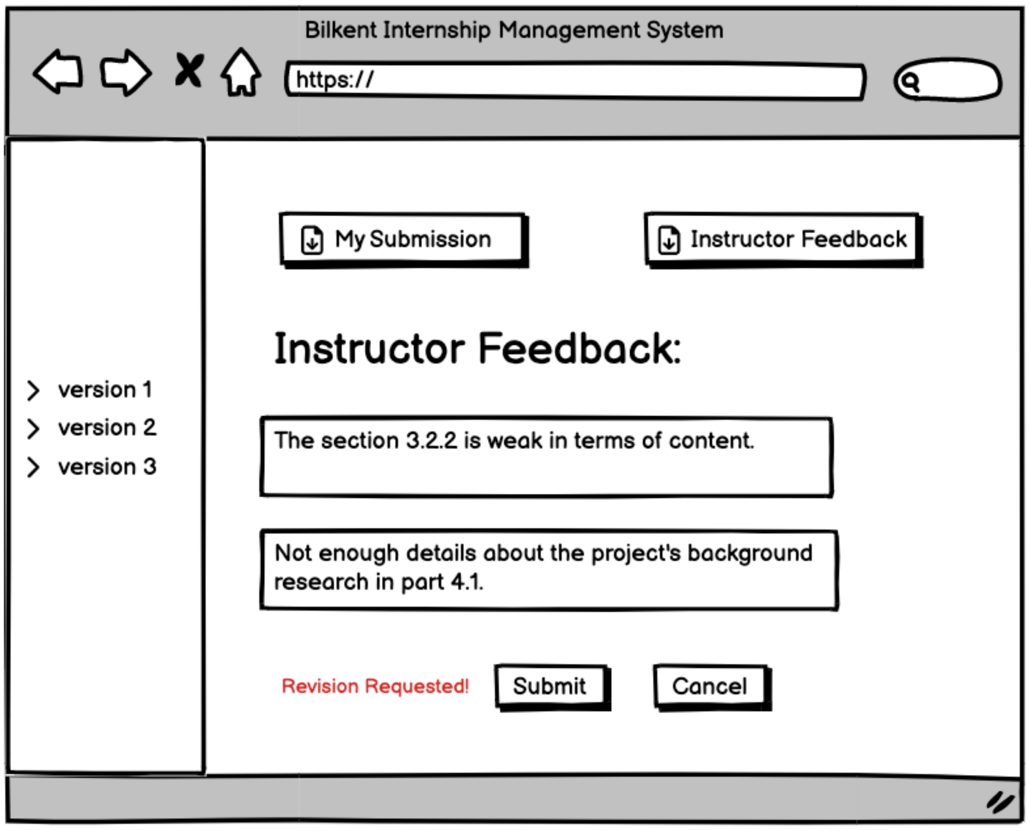


Figure 10: Login screen



The section 3.2.2 is weak in terms of content.

Not enough details about the project's background research in part 4.1.

Figure 11: Student Versions Screen

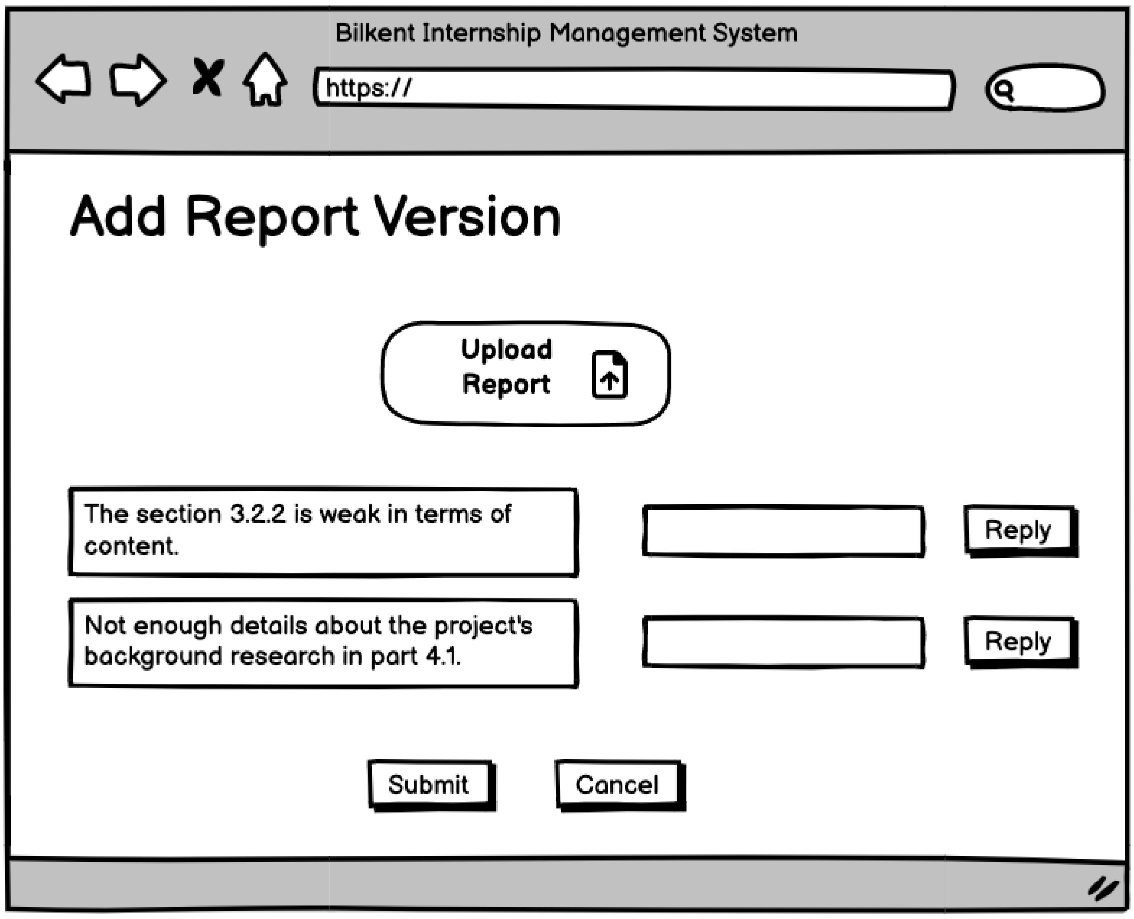
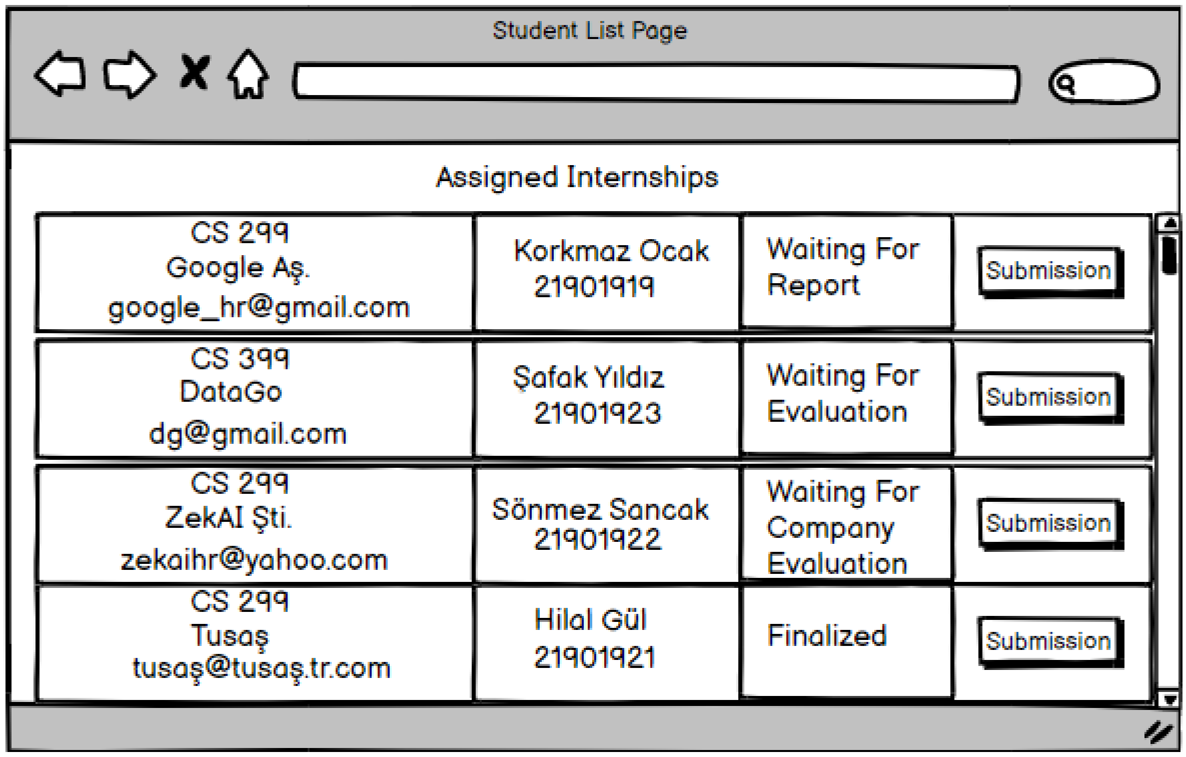

Bilkent Internship Management System
https://
Add Report Version
Upload Report
The section 3.2.2 is weak in terms of content.
Reply
Not enough details about the project's background research in part 4.1.
Reply
Submit
Cancel

Figure 12: Student Upload Version and Reply Feedback Screen

Instructor Screen



Student List Page

Assigned Internships

CS 299 Google Aş. google_hr@gmail.com	Korkmaz Ocak 21901919	Waiting For Report	Submission
CS 399 DataGo dg@gmail.com	Şafak Yıldız 21901923	Waiting For Evaluation	Submission
CS 299 ZekAI Şti. zekaihr@yahoo.com	Sönmez Sancak 21901922	Waiting For Company Evaluation	Submission
CS 299 Tusaş tusaş@tusaş.tr.com	Hilal Gül 21901921	Finalized	Submission

Figure 13: Instructor Student List Page Screen

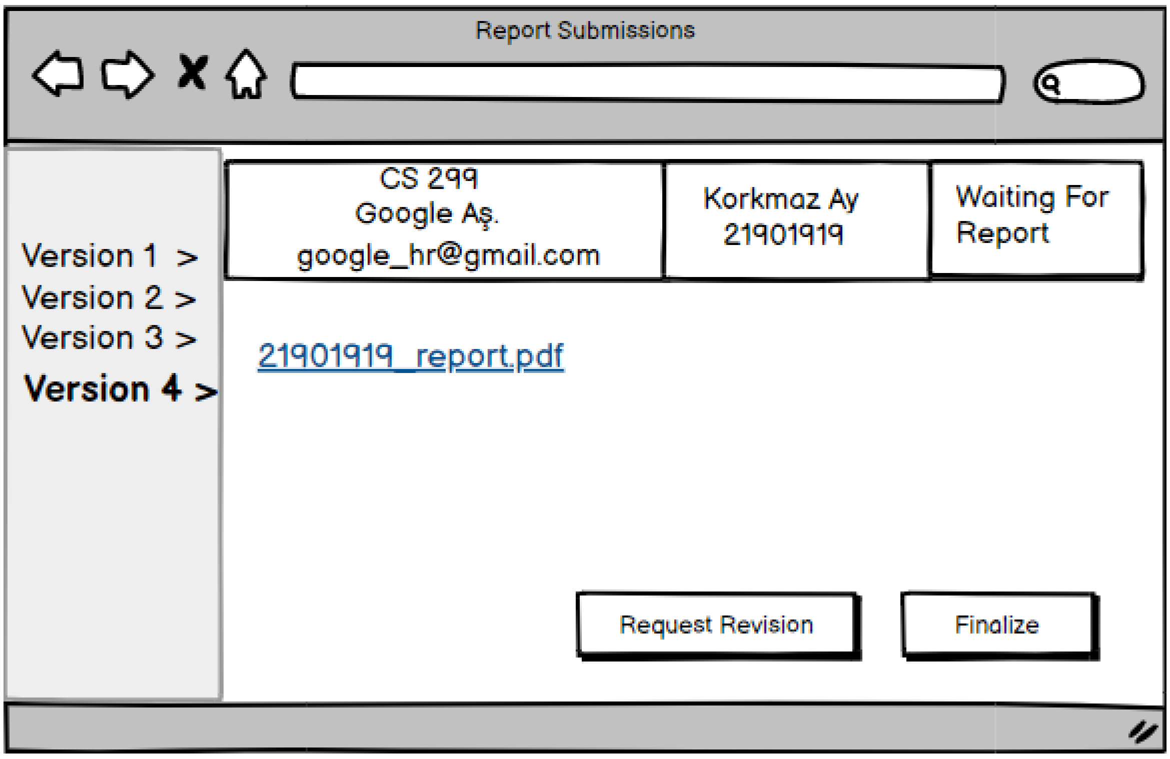
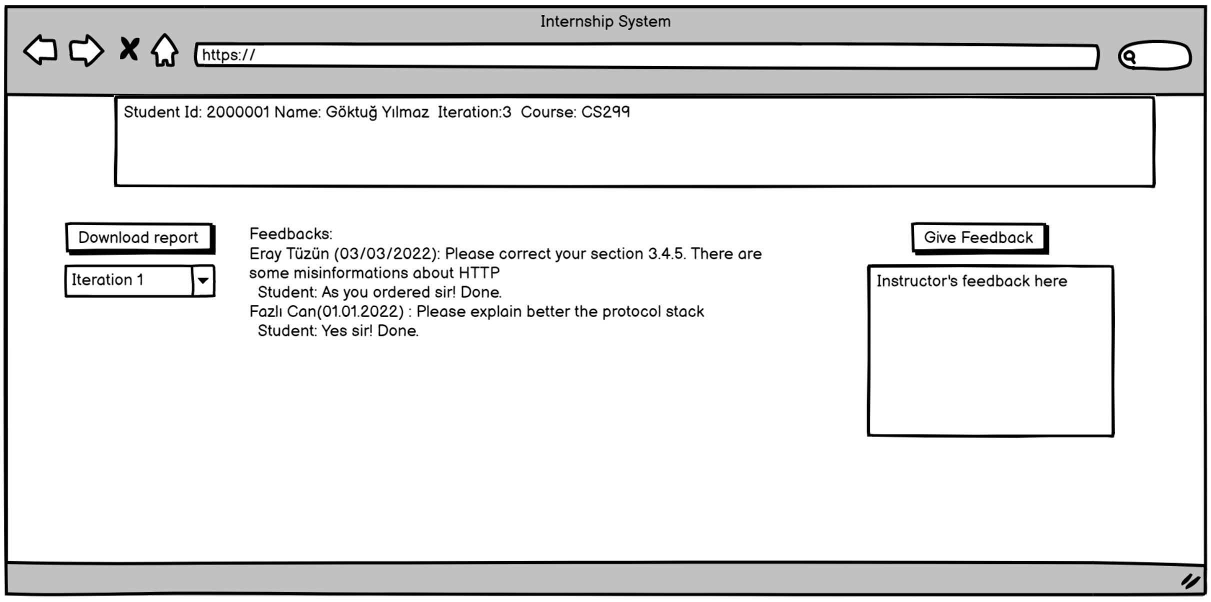


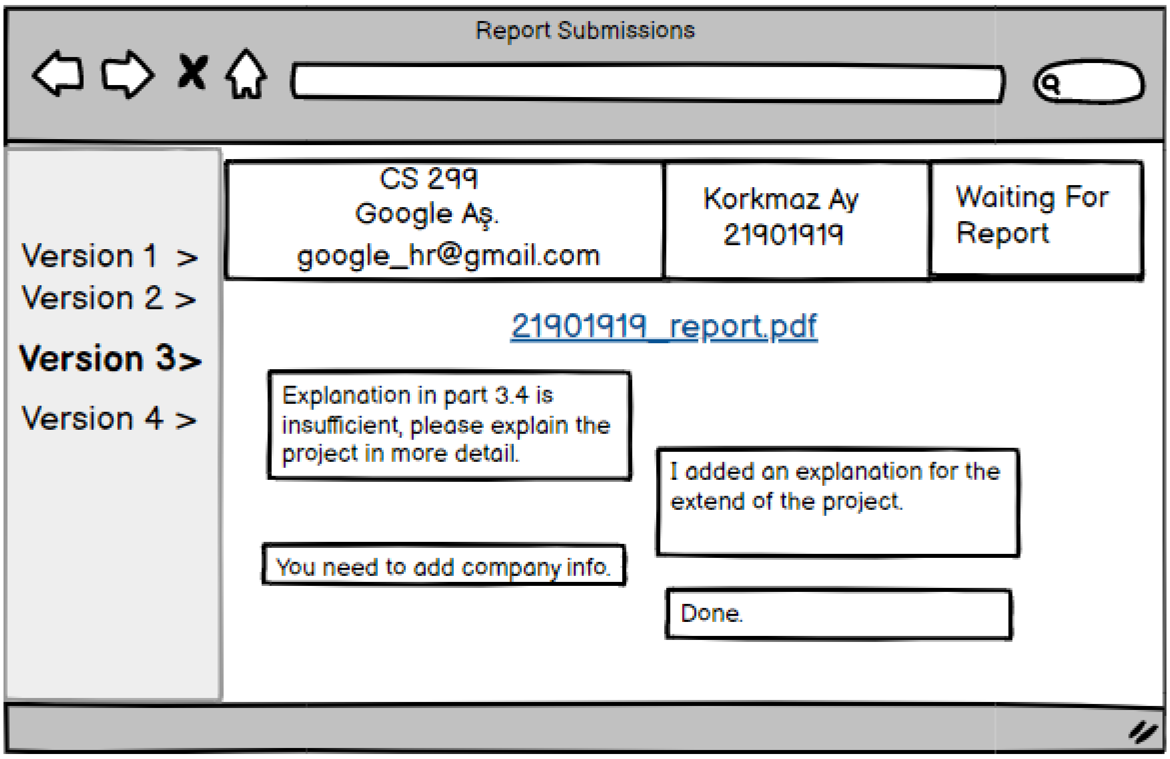
Figure 14: Instructor Student's Uploaded Versions Screen



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

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.

Figure 15: Instructor Give Feedback Screen


Report Submissions

Version 1 >
Version 2 >
Version 3>
Version 4 >

CS 299
Google Aş.
google_hr@gmail.com

Korkmaz Ay
21901919

Waiting For Report

21901919_report.pdf

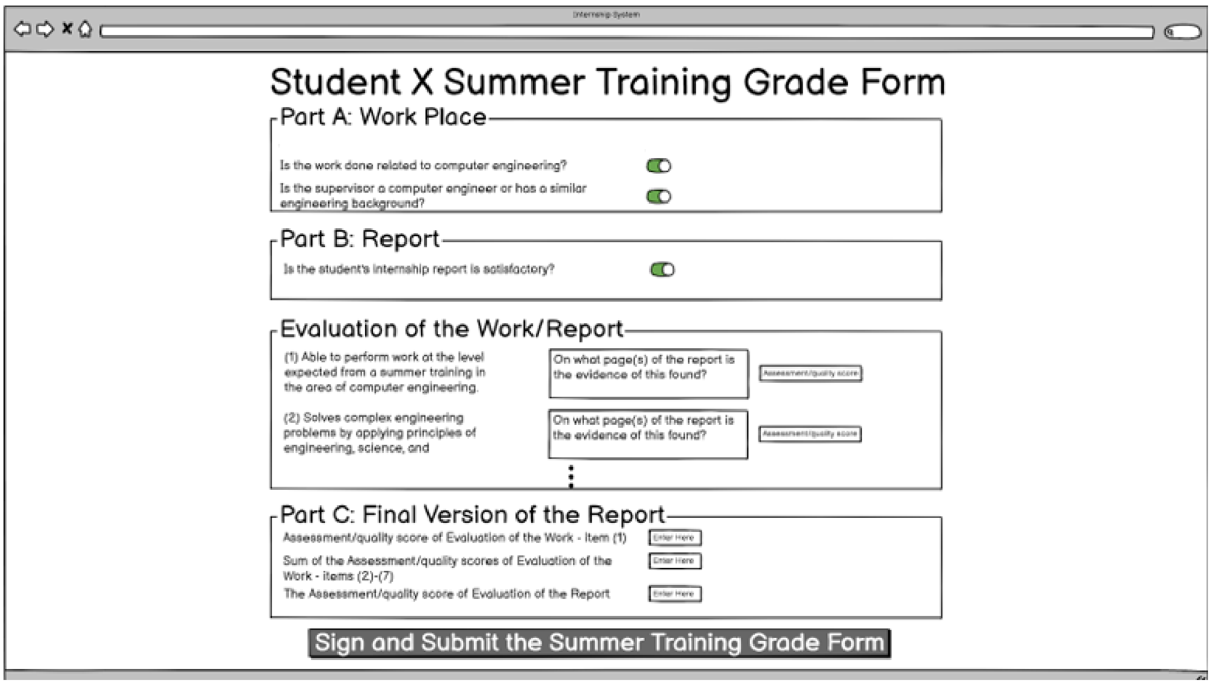
Explanation in part 3.4 is insufficient, please explain the project in more detail.

I added an explanation for the extend of the project.

You need to add company info.

Done.

Figure 16: Instructor Student's Uploaded Versions Screen


Internship System

Student X Summer Training Grade Form

Part A: Work Place

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

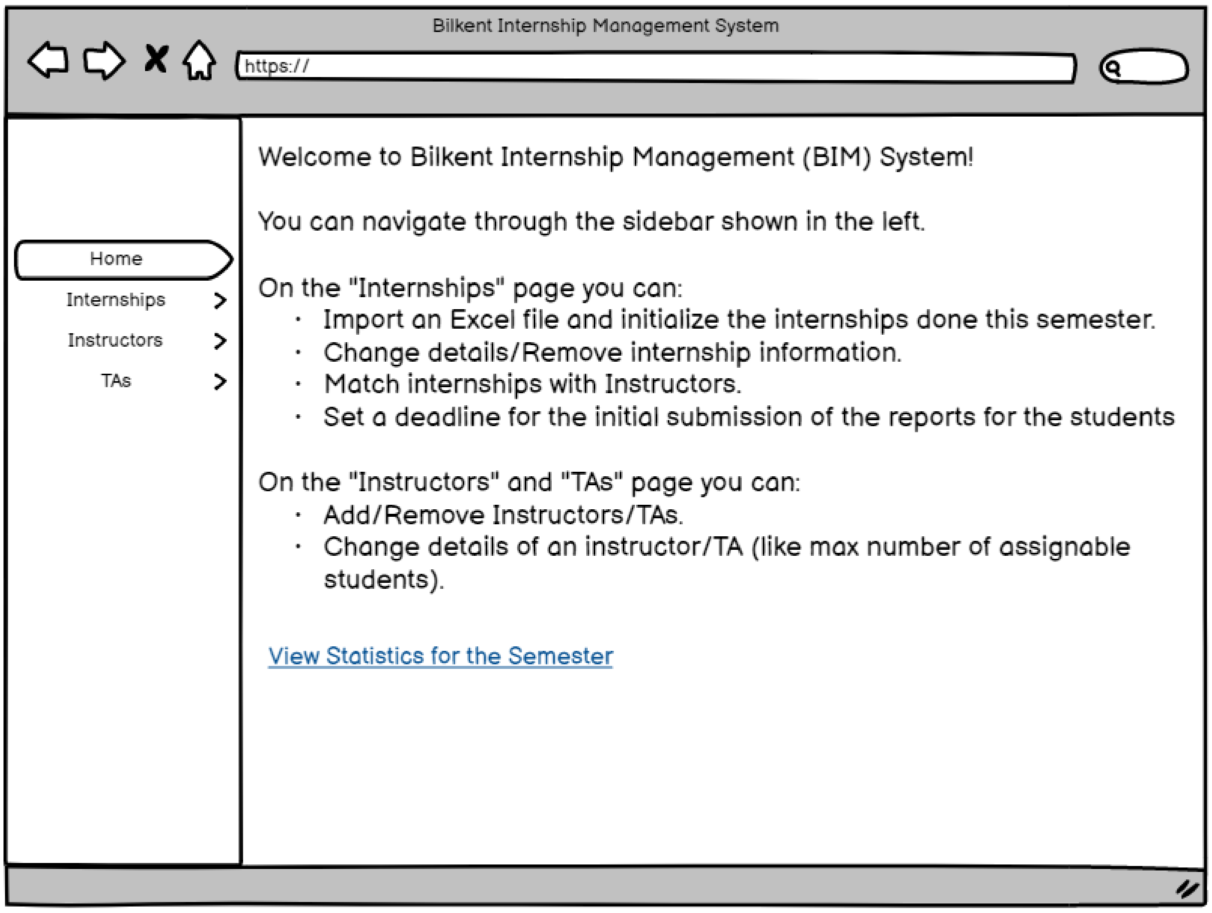
Sum of the Assessment/quality scores of Evaluation of the Work - items (2)-(7) Enter Here

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

Sign and Submit the Summer Training Grade Form

Figure 17: Instructor Grade Internship Of Student Screen

Secretary Screen



Welcome to Bilkent Internship Management (BIM) System!

You can navigate through the sidebar shown in the left.

On the "Internships" page you can:

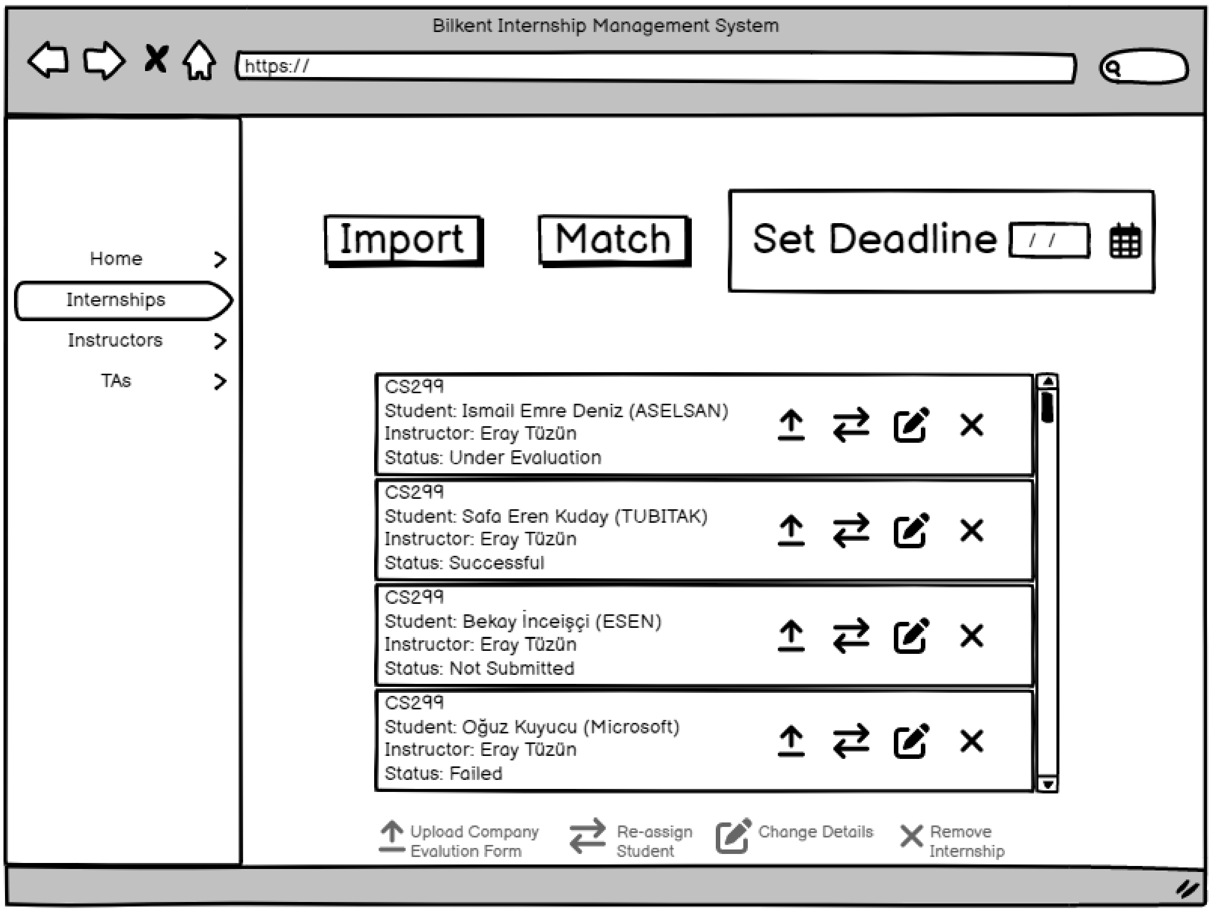
- Import an Excel file and initialize the internships done this semester.
- Change details/Remove internship information.
- Match internships with Instructors.
- Set a deadline for the initial submission of the reports for the students

On the "Instructors" and "TAs" page you can:

- Add/Remove Instructors/TAs.
- Change details of an instructor/TA (like max number of assignable students).

View Statistics for the Semester

Figure 18: Secretary Homepage Screen



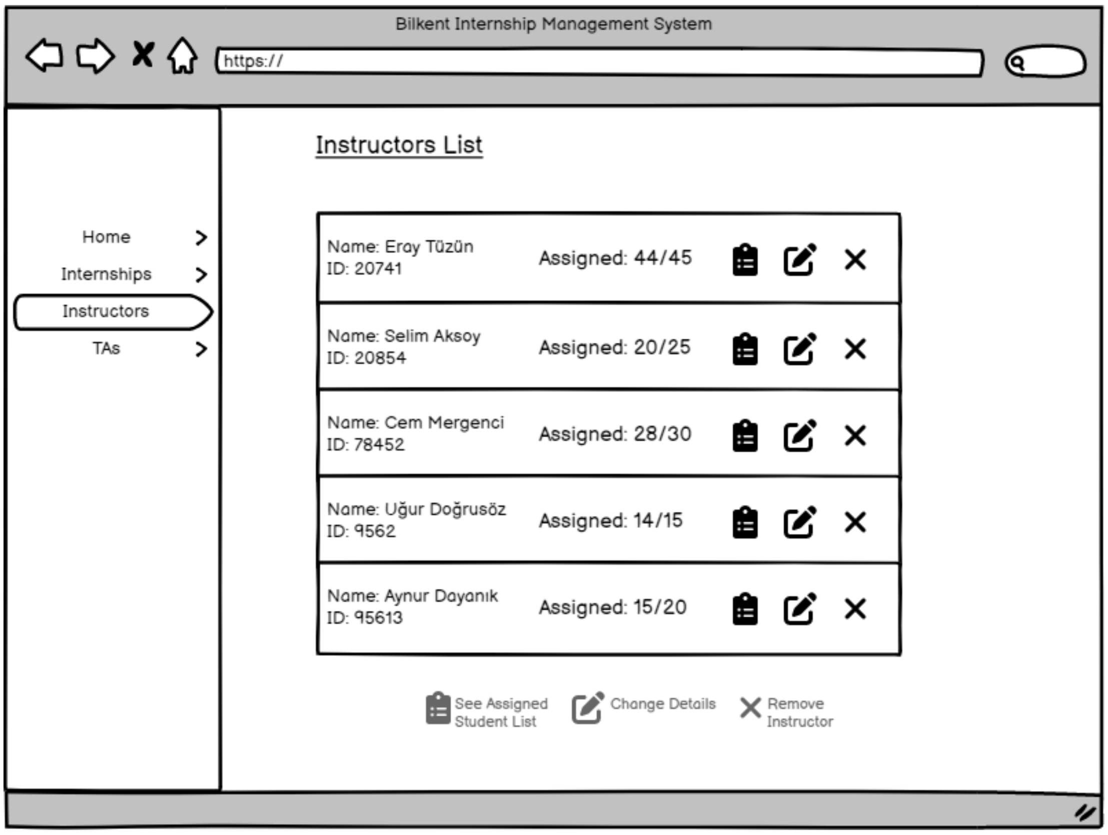
CS299
Student: Ismail Emre Deniz (ASELSAN)
Instructor: Eray Tüzün
Status: Under Evaluation

CS299
Student: Safa Eren Kuday (TUBITAK)
Instructor: Eray Tüzün
Status: Successful

CS299
Student: Bekay İnceişçi (ESEN)
Instructor: Eray Tüzün
Status: Not Submitted

CS299
Student: Oğuz Kuyucu (Microsoft)
Instructor: Eray Tüzün
Status: Failed

Figure 19: Secretary Internships Screen



Bilkent Internship Management System

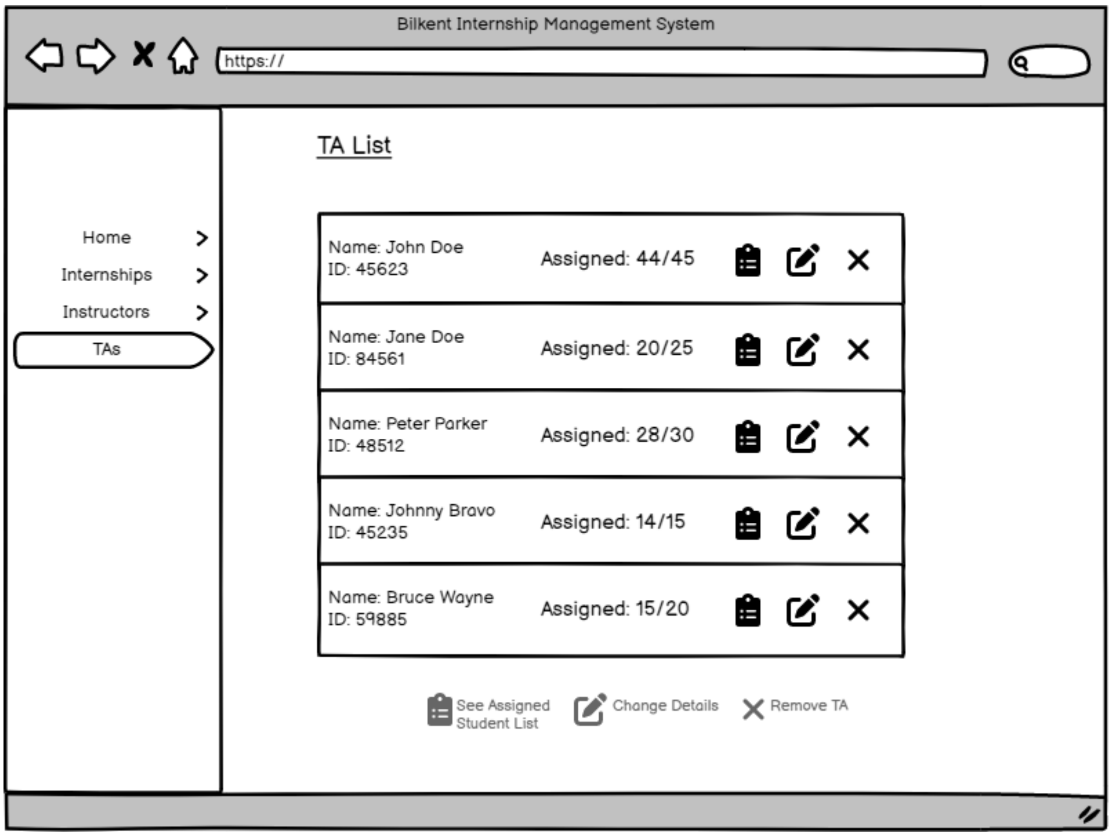
https://

- Home
- Internships
- Instructors
- TAs

Instructors List

Name: Eray Tüzün ID: 20741	Assigned: 44/45
Name: Selim Aksoy ID: 20854	Assigned: 20/25
Name: Cem Mergenci ID: 78452	Assigned: 28/30
Name: Uğur Doğrusöz ID: 9562	Assigned: 14/15
Name: Aynur Dayanık ID: 95613	Assigned: 15/20

See Assigned Student List · Change Details · Remove Instructor



Bilkent Internship Management System

https://

- Home
- Internships
- Instructors
- TAs

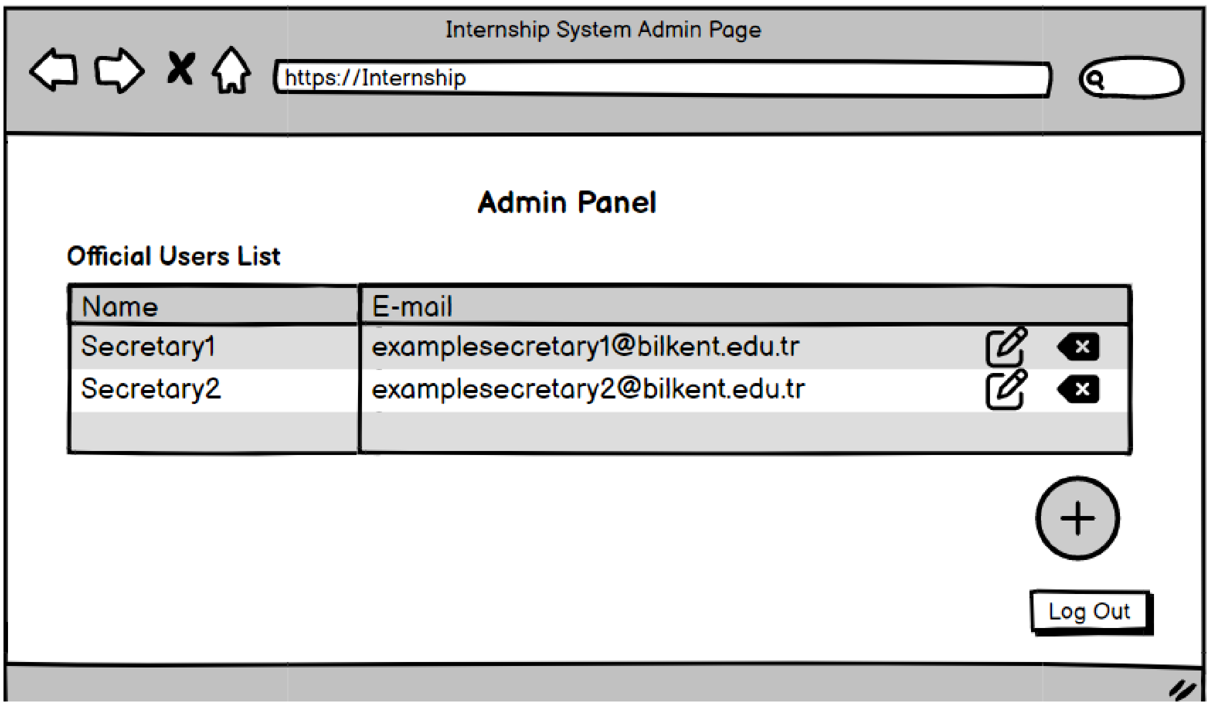
TA List

Name: John Doe ID: 45623	Assigned: 44/45
Name: Jane Doe ID: 84561	Assigned: 20/25
Name: Peter Parker ID: 48512	Assigned: 28/30
Name: Johnny Bravo ID: 45235	Assigned: 14/15
Name: Bruce Wayne ID: 59885	Assigned: 15/20

See Assigned Student List · Change Details · Remove TA

Figure 20: Secretary Instructors and TA's Screen

Admin Screen



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

Figure 21: Admin Panel

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.