CS 319
Object Oriented Software Engineering

Analysis Report
(Version 2.0)
Mar 31, 2023

Instructor: Eray Tüzün
Project: Student Internship System

Group Name: Quaso
Members:
Zeynep Doğa Dellal 22002572
Yağız Alkılıç 22003281
Muhammad Ali Waris 22001037
Yağız Berk Uyar 21902318
Mustafa Hamit Dölek 21703136

Table of Contents

1-Introduction

Quaso Internship Program is a desktop program intended to be used by Bilkent University engineering students. The program aims to reduce the difficulties that the currently implemented multi-platform system can cause by lowering the control, evaluation, and approval of reports mechanism of students' internship reports to be simpler for each user by reducing the system to a single platform.

In the currently used system, report approval, evaluation, revision request if necessary, and control of companies are carried out through 4 different platforms: Mfstaj, Moodle, Google Drive, and email. This multi-platform system makes the use of both evaluators and students complex, and sometimes it can confuse terms of controlling and tracking documents. Considering all this, the primary purpose of our program is to create a more practical, effective, and efficient system for each user by minimizing these complications.

Four types of users are targeted to use the program: Assistants, students, super-admins, and evaluators. Each user aims to log in to the program using the specially designed login page and work on the assigned job through a functional and aesthetic interface in which the user is in control, and consistent, with a low cognitive load.

Thanks to this program being developed, students will be able to upload and follow their internship reports more easily, view the acceptance status of the company they are doing their internship with, view their feedback and check their deadlines. At the same time, assistants and evaluators will be able to follow up with the students assigned to them more easily, view all the information of the uploaded and pre-loaded internship reports on a single page, check the forms submitted by the companies and, if necessary, upload their feedback via the application and request revisions.

2. Current System

The process for adding a new company to the mfstaj system database for summer internship consists of the following steps:

2.1. Create a new entry in the mfstaj database: You must create a new entry in the mfstaj database and add all the required information about the company. Ensure that there is no duplicate entry. If the company is approved for another department, you do not need to make a new entry.

2.2. Submit information to the student coordinators: You must submit the information to the student coordinators via email(staj[at]cs.bilkent.edu.tr). The coordinators will review the information and decide to approve or disapprove the proposed mfstaj entry. They may require additional information.

2.3. Required information: The following information must be submitted to the student coordinators:

- Company name, phone number, postal address, and web address
- Is the company doing computer engineering work? Explain.
- Will your supervisor be a person with a computer engineering degree? Explain.
- Will the tasks assigned to you satisfy the Performance Criteria listed in the guidelines document? Explain

. 2.4. Provide a detailed explanation: The word “Explain” in any question above means that you are required to explain. For example, simply writing “The tasks will satisfy the Performance Criteria” will not be accepted as an answer. You need to talk to the company and get information about how the internship they are planning will satisfy these criteria.

2.5. Approval process: Once the company is approved in the system, the rest of the process will be handled by the Dean’s Office. Therefore, if you have any questions about the remaining procedure, you should contact the Dean’s Office directly.

2.6. Deadlines: Note that the approval for the company and the application to the Dean’s office must be done before you start any internship. There are no exceptions for internships that were arranged for past dates. Ensure that you are aware of the deadlines announced by the Dean’s Office.

3. Proposed System

3.1 Non-Functional Requirements

3.1.1. Performance

The application must be able to handle a large number of users simultaneously without any lag or delay in response time. The response time of the application should be less than 3 seconds on average.

3.1.2. Security

The application should ensure the security and privacy of user data, including personal information and documents. The system should encrypt all sensitive data and ensure that only authorized users can access it. The applicant should also have password protection.

3.1.3. Compatibility

The application should be compatible with all popular web browsers(Chrome, Firefox, Safari, and Edge) and operating systems(Windows, MacOS, and Linux).

3.1.4. Scalability

The application should scale up easily as the number of users and data increases. It should be able to handle a growing amount of data without any issues and should be easily expandable in terms of infrastructure.

3.1.5. Usability

The application should be user-friendly and easy to use for all types of users, including students, instructors, coordinators, and company owners. The application should have clear instructions and provide useful feedback to users

.

3.1.6. Maintainability

The application should be designed with maintainability in mind, ensuring that it is easy to modify or update in the future. The code should be well-structured, modular, and easily understandable by other developers

3.2. System models

3.2.1. Use Case Model

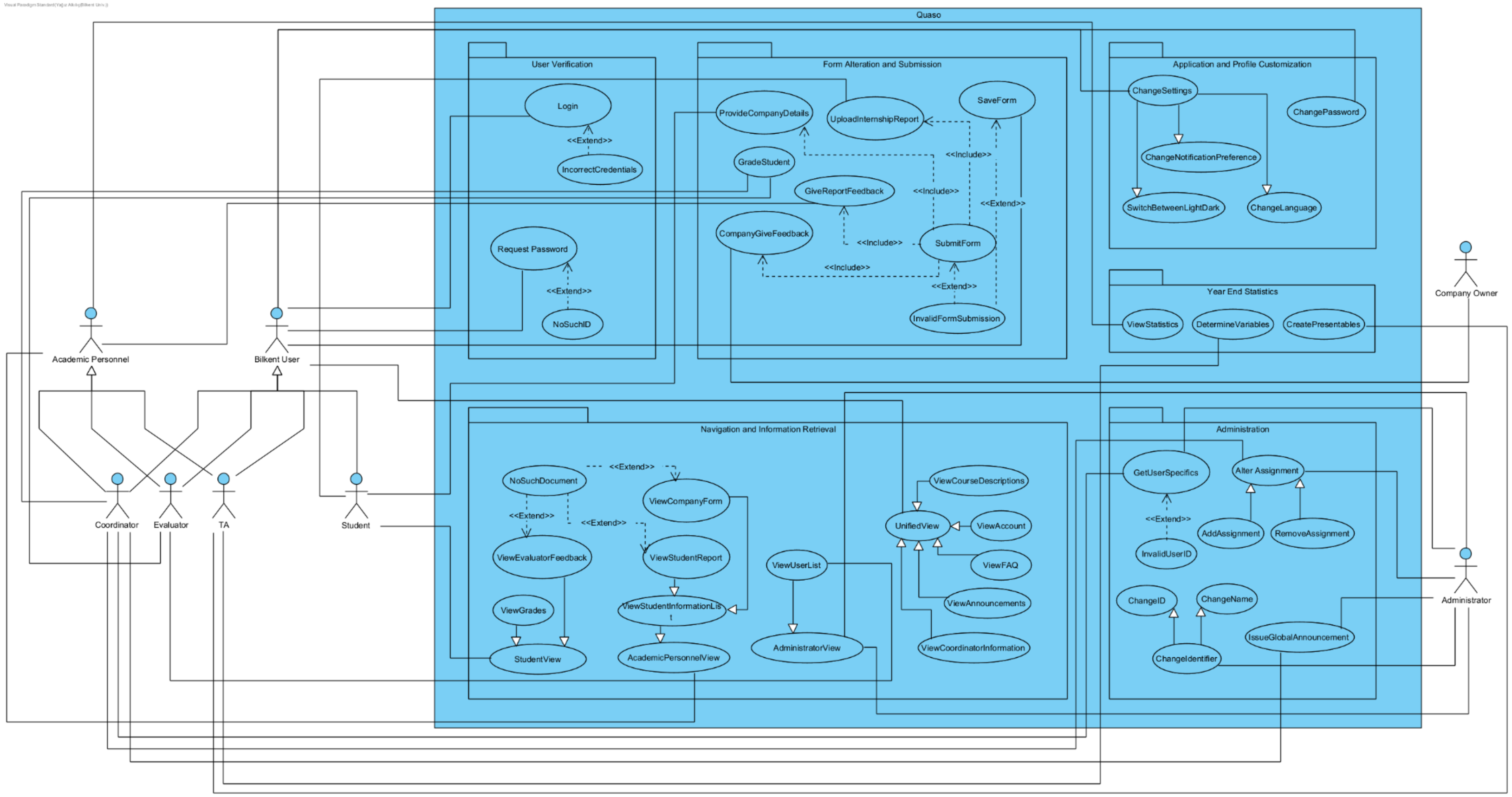


Figure 1: Use Case Model (High-quality version: https://github.com/zedyjy/CS319-Project/blob/main/Diagrams/Use%20Case.png)

3.2.2. Use Case Model Textual Descriptions

3.2.2.1. User Verification Package

Use case name: Login

Participating actors: Initiated by a student, an evaluator, a coordinator, a TA, or an administrator

The flow of events:

1. The user enters an id and password.
2. The user clicks the login button.
3. If login is successful, the user is directed to the student main menu; otherwise, an error message pops up.

Exit conditions:

- The user successfully logins, OR
- The login process is terminated.

Use case name: IncorrectCredentials

Participating actors: Communicates with a student, an evaluator, a coordinator, a TA, or an administrator

The flow of events:

1. The user enters incorrect information on the login page.
2. The system issues a warning message.

Entry conditions:

- This use case **extends** Login. The system initiates this case when a user fails to log in.

Exit conditions:

- The user closes the error message tab.

Use case name: RequestPassword

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user clicks on the “forgot your password” button.
2. The user enters their email address to the form and submits the information.

Exit conditions:

- The user submits the form, OR
- The user cancels the form subscription.

Use case name: NoSuchID

Participating actors: Communicates with a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user enters an email address that does not exist in the system on the request password page.
2. The system issues a warning message.

Entry conditions:

- This use case **extends** RequestPassword. The system initiates this case when a user fails to provide an appropriate email address.

Exit conditions:

- The user closes the error message tab.

3.2.2.2. Navigation and Information Retrieval Package

3.2.2.2.1 Navigation Initialization

Use case name: StudentView

Participating actors: Initiated by the student

The flow of events:

1. The homepage consists of an introductory list of options, after an option is selected, the user is directed to that page. After this initialization, every page contains a navigation bar that is located on the left side of the page. It can be used to visit other pages. The only instances in which the navigation bar is inaccessible are:
   a) An error message is displayed and requires dismissal.
   b) The user is yet to log in.
2. A list of buttons that leads to information relating to the student's progression throughout the course is displayed in the navigation bar. These buttons lead to pages that include reports, grades, and general tabs.
3. The student clicks on the relevant buttons to be directed to their desired operation.

Entry conditions:

- The student is logged in to the system, OR
- The user clicked on any option on the introductory page.

Exit conditions:

- The student logs out of the system via the logout button, AND
- The student proceeds to the operations listed on the menu.

Use case name: AcademicPersonnelView

Participating actors: Initiated by an evaluator, a coordinator, or a TA

The flow of events:

1. The homepage consists of an introductory list of options, after an option is selected, the user is directed to that page. After this initialization, every page contains a navigation bar that is located on the left side of the page. It can be used to visit other pages. The only instances in which the navigation bar is inaccessible are:
    a) An error message is displayed and requires dismissal.
    b) The user is yet to log in.
2. A list of buttons that leads to information relating to the course is displayed in the evaluator navigation bar. These buttons lead to pages that include students and reports, and general tabs.
3. The evaluator clicks on the relevant buttons to be directed to their desired operation.

Entry conditions:

- The evaluator is logged in to the system, AND
- The user clicked on any option on the introductory page.

Exit conditions:

- The evaluator logs out of the system via the logout button, OR
- The evaluator proceeds to the operations listed in the menu.

Use case name: UnifiedView

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The users have access to two navigation bars. One is at the top of each page, and the other one is at the left of each page. The top bar is identical for any user; it can be used to access FAQs, options, announcements, and contacts. The left bar changes but there are identical options for all users. These options are announcements, coordinator information, and course description.

Entry conditions:

- The user is logged in to the system, AND
- The user clicked on any option on the introductory page.

Exit conditions:

- The user logs out of the system via the logout button, OR
- The evaluator proceeds to the operations listed in the menu.

Use case name: AdministratorView

Participating actors: Initiated by an administrator

The flow of events:

1. Administrators do not have a left navigation bar. They have a main page and a top navigation bar.
2. The top bar consists of FAQs, options, announcements, and contacts.

Entry conditions:

- The administrator is logged in to the system.

Exit conditions:

- The administrator logs out of the system via the logout button, OR
- The administrator proceeds to manage a user.

3.2.2.2.2. Non-specific Navigation and Information Retrieval

Use case name: ViewCourseDescriptions

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user is shown a list of documents relating to the course.
2. The user can click on the document and view its contents.

Entry conditions:

- Conditions inherited from UnifiedView.

Exit conditions:

- The user navigates to another page or logs out.

Use case name: ViewAnnouncements

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user is shown a list of announcements together with their dates.

Entry conditions:

- Conditions inherited from UnifiedView.

Exit conditions:

- The user navigates to another page or logs out.

Use case name: ViewCoordinatorInformation

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user is shown a list of coordinators with their contact information.

Entry conditions:

- Conditions inherited from UnifiedView.

Exit conditions:

- The user navigates to another page or logs out.

Use case name: View Account

Participating actors: Initiated by a student, an evaluator, a coordinator, or a TA

The flow of events:

1. The user is shown identifying information such as id, name, and email address.
2. The user is shown an option to change the password.

Entry conditions:

- Conditions inherited from UnifiedView.

Exit conditions:

- The user navigates to another page or logs out.

Use case name: ViewFAQ

Participating actors: Initiated by a student, an evaluator, or a coordinator

The flow of events:

1. The user is shown a list of frequently asked questions and their answers.

Entry conditions:

- Conditions inherited from UnifiedView.

Exit conditions:

- The user navigates to another page or logs out.

3.2.2.2.3. User-specific Navigation and Information Retrieval

Use case name: ViewEvaluatorFeedback

Participating actors: Initiated by the student

The flow of events:

1. The student navigates to the “reports” page.
2. The student clicks on the “view evaluator review” button.
3. The student is shown the evaluator's feedback.
4. The student closes the document.

Entry conditions:

- Conditions inherited from StudentView.

Exit conditions:

- The student logs out, OR
- The student clicks on another tab in the main menu.

Use case name: ViewStudentInformationList

Participating actors: Initiated by an evaluator, a TA, or a coordinator

The flow of events:

1. The evaluator clicks on the “students and reports” tab.
2. A list of students is displayed to the user. The list identifiers consist of:
    a) Student information includes ID, name, report submission date, report submission status, grading status, company form, and student report form.
    b) Evaluator tasks including grade students, and requesting revision or giving feedback. These tasks are not displayed to the coordinator.
3. The user can choose to interact with the page to view information or complete tasks.

Entry conditions:

- Conditions inherited from AcademicPersonnelView.

Exit conditions:

- The user logs out, OR
- The user clicks on another tab in the main menu.

Use case name: ViewCompanyForm

Participating actors: Initiated by an evaluator, a TA, or a coordinator

The flow of events:

1. The user clicks on the “view” button of the corresponding student in the “company form” column.
2. The company form that was uploaded by the student is displayed to the user.

Entry conditions:

- Conditions inherited from ViewStudentInformationList.

Exit conditions:

- The user closes the document.

Use case name: ViewStudentReport

Participating actors: Initiated by an evaluator, a TA, or a user

The flow of events:

1. The user clicks on the “view” button of the corresponding student in the “student report form” column.
2. The report that was uploaded by the student is displayed to the user.

Entry conditions:

- Conditions inherited from ViewStudentInformationList.

Exit conditions:

- The user closes the document.

Use case name: NoSuchDocument

Participating actors: Communicates with the user

The flow of events:

1. A user requests to view a document. An example document can be a report review a student wants to access or company information an evaluator wants to review.
2. The requested document does not exist in the system.
3. The system issues a warning message.

Entry conditions:

- This use case **extends** any case where a document is requested from the server. The system initiates this case when a requested document cannot be provided.

Exit conditions:

- The evaluator closes the error message tab.

3.2.2.3. Form Alteration and Submission Package

Use case name: ProvideCompanyDetails

Participating actors: Initiated by the student

The flow of events:

1. The student fills out and submits a form to provide company information.
2. The system saves the information.
3. The system sends an approval request to the assigned evaluator.

Entry conditions:

- The student is logged in to the system, AND
- The student clicks on the “company info” button in the student's main menu.

Exit conditions:

- The student chooses to cancel the submission, OR
- The student provides the information and submits the form.

Quality Requirements:

- This case can include the company availability checker use case that compares the name of the company with approved companies in the database. Then the user can be given a warning regarding the situation and be asked if they would like to continue with the submission.

Use case name: UploadInternshipReport

Participating actors: Initiated by the student

The flow of events:

1. The student clicks on the “upload internship report” or “update internship report” button depending on their progression.
2. The student enters the requested information.
    a. If a document is requested, the student pushes the “upload document button” and selects the related document on their device.
    b. The uploaded document is shown to the student.
3. The student clicks the “submit” button.

Entry conditions:

- The student is logged in to the system, AND
- The student clicks on the “reports” button in the student main menu.

Exit conditions:

- The student chooses to cancel the submission, OR
- The student clicks the “upload” button.

Use case name: GradeStudent

Participating actors: Initiated by an evaluator, or a coordinator

The flow of events:

1. The evaluator clicks on the “grade student” button of the corresponding student.
2. The evaluator is shown a form. The form consists of:
    a) Buttons that can be clicked to view information such as student reports, or company forms. The interactions with these buttons are identical to the use cases ViewCompanyForm and ViewStudentReport.
    b) Text fields that correspond to grades. These grades correspond to different parts of the course and they can be entered separately.
    c) Buttons that can be clicked to submit the grades, or to save the current state for later.
3. The evaluator enters the grades and submits the form, or the evaluator enters grades and saves the form for later use, or the evaluator clicks the “cancel” button to terminate the operation.

Entry conditions:

- Conditions inherited from ViewStudentInformationList.

Exit conditions:

- The evaluator submits grades, saves the form, cancels the submission, clicks on another tab, or logs out.

Use case name: CompanyGiveFeedback

Participating actors: Initiated by the company owner

The flow of events:

1. The company owner fills in the form that is shown to them.
2. The company owner submits the form.
3. The company owner can choose to save the form to complete it later.

Entry conditions:

- The company owner entered the application by using a specified URL that was provided to them.

Exit conditions:

- The company owner saves the form, submits the form, or closes the application.

Use case name: GiveReportFeedback

Participating actors: Initiated by an evaluator, a TA, or a coordinator

The flow of events:

1. The evaluator clicks on the “request revision and gives feedback” button of the corresponding student.
2. The evaluator is shown a form. The form consists of:
    a) A button that can be clicked to view previously submitted student reports. The interaction with this button is identical to the use case ViewStudentReport.
    b) A text field that can be filled in by the evaluator to specify the request.
3. The evaluator fills in the text field and clicks the “send feedback and request revision” button, or terminates the operation by clicking the “cancel” button.

Entry conditions:

- Conditions inherited from ViewStudentInformationList.

Exit conditions:

- The evaluator submits the request, cancels the submission, clicks on a different tab, or logs out.

Use case name: InvalidFormSubmission

Participating actors: Communicates with a student, a company owner, or an evaluator

The flow of events:

1. A condition of an invalid form submission is detected. These conditions can be listed as:
    a) Entry in the text field consists of invalid characters.
    b) A number entry is not in the valid interval if an interval is given.
    c) A required field is empty (This condition only applies if the form is trying to be submitted. Users can leave fields empty when they want to save).
    d) The submission deadline has passed.
2. Submit or Save form button is clicked.
3. The system issues a warning message according to the rule violation.

Entry conditions:

- This use case **extends** any use case that is listed in the form alteration and submission package. The system initiates this case when an invalid entry is made to the submission page.

Exit conditions:

- The user closes the error message tab.

3.2.2.4. Administration Package

Use case name: GetUserSpecifics

Participating actors: Initiated by the administrator

The flow of events:

1. For any user two types of information are listed in a user specifics menu:
    a) Identifiers: ID, name, and course enrollment status of the user.
    b) Assignments: Every user is in relation to another user in discrete ways. For example, a student is assigned to an evaluator to be graded, or an evaluator is assigned to a student to grade them. These relations can be altered by manipulating assignments.
2. An administrator can choose to change this information.
3. An administrator can choose to navigate to a related user. If a student is assigned to an evaluator, for example, the administrator can navigate to that evaluator's profile via the student's profile.

Entry conditions:

- The administrator is logged in to the system, AND
- The administrator requested a user profile.

Exit conditions:

- The administrator logs out of the system via the logout button, OR
- The administrator proceeds to the administrative homepage.

Use case name: ChangeIdentifier

Participating actors: Initiated by the administrator

The flow of events:

1. An administrator can choose to edit a user's name or id. To do this they press the "edit" button near the attribute that they wish to change.
2. After the button is pressed a text field pops up.
3. The administrator enters the new information into the text field.
4. The administrator presses the "submit" button.
5. The administrator can choose to abort this process by closing the text field.

Entry conditions:

- Conditions inherited from GetUserSpecifics.

Exit conditions:

- The administrator logs out of the system via the logout button, OR
- The administrator proceeds to the administrative homepage.

Use case name: AddAssignment

Participating actors: Initiated by the administrator

The flow of events:

1. The administrator fills the text field with the id of the user that they would like to assign.
2. The administrator clicks on the “display profile” button.
3. The administrator is shown the specifics of the given user.
4. The administrator clicks the “assign” button to assign the user.

Entry conditions:

- Conditions inherited from GetUserSpecifics.

Exit conditions:

- The administrator logs out of the system via the logout button, OR
- The administrator proceeds to the administrative homepage.

Use case name: RemoveAssignment

Participating actors: Initiated by the administrator

The flow of events:

1. The administrator clicks on the “remove” button near the assignment that they would like to remove from the user.
2. The system issues a success message.

Entry conditions:

- Conditions inherited from GetUserSpecifics.

Exit conditions:

- The administrator logs out of the system via the logout button, OR
- The administrator proceeds to the administrative homepage.

Use case name: InvalidUserID

Participating actors: Communicates with the administrator

The flow of events:

1. The administrator enters an unaccepted character, the user id does not exist in the system, or the user id does not correspond to the correct user type.
2. The system issues a warning message.

Entry conditions:

- This use case **extends** ViewAdministrativeHomepage and GetUserSpecifics. The system initiates this case when an administrator enters incorrect id information.

Exit conditions:

- The administrator closes the error message tab.

Use case name: IssueGlobalAnnouncement

Participating actors: Initiated by a coordinator, or an administrator

The flow of events:

1. The user clicks on the “issue notification” button.
2. The user fills in the text field to specify the message.
3. The user enters the issue date to specify a time slot for notification.
4. The notification is sent to all users.

Entry conditions:

- The user is logged in to the system.
- The user navigated to the announcements page.

Exit conditions:

- The user logs out of the system via the logout button, OR
- The user navigates to a different page.

3.2.2.5. Application and Profile Customization Package

Use case name: ChangeSettings

Participating actors: Initiated by any

The flow of events:

1. The user views the application settings.

Entry conditions:

- This user is logged in to the system, AND
- The user clicked on the “settings” button on the top bar.

Exit conditions:

- The user logs out, OR
- The user navigates to a different page.

Use case name: ChangeLanguage

Participating actors: Initiated by any user

The flow of events:

1. The user clicks on the dropdown list labeled as the current language of the app.
2. The user selects the desired language.
3. The user clicks on the “apply” button.
4. Application language changes to the desired language.

Entry conditions:

- Conditions inherited from ChangeSettings.

Exit conditions:

- The user clicked on apply, OR
- The user canceled the operation.

Use case name: ChangeNotificationPreference

Participating actors: Initiated by any user

The flow of events:

1. The user clicks on and approves notification types to change the type of notifications that they would like to receive. The types can be “system-issued reminders”, “coordinator notifications”, and “administrator notifications”.
2. The user enters an email address into the text field to receive notifications outside of the app.
3. The user clicks on the “apply” button.

Entry conditions:

- Conditions inherited from ChangeSettings.

Exit conditions:

- The user clicked on apply, OR
- The user canceled the operation.

Use case name: SwitchBetweenLightDark

Participating actors: Initiated by any user

The flow of events:

1. The user clicks on the “change to dark mode”, or “change to light mode” button depending on the current settings.
2. The user clicks on the “apply” button.
3. The color palette of the app changes.

Entry conditions:

- Conditions inherited from ChangeSettings.

Exit conditions:

- The user clicked on apply, OR
- The user canceled the operation.

Use case name: ChangePassword

Participating actors: Initiated by a student, an evaluator, or a coordinator

The flow of events:

1. The user clicks on the “change password” button.
2. The user enters a new password in the text field.
3. The user clicks on the “save password” button.

Entry conditions:

- Conditions inherited from ViewAccount.

Exit conditions:

- Conditions inherited from ViewAccount.

3.2.3. Object and Class model

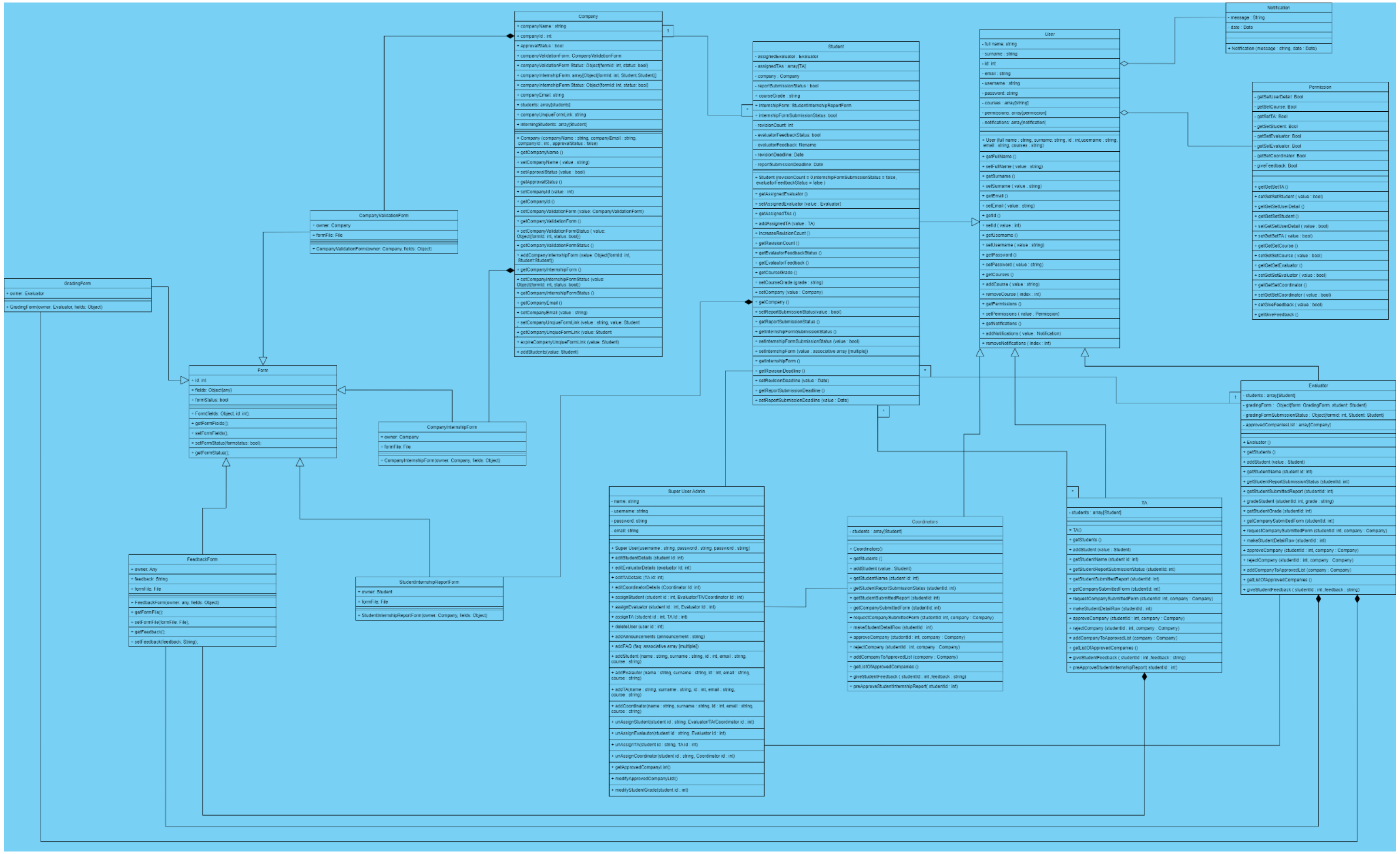

Company
+ companyName : string
+ companyId : int
+ approvalStatus : bool
+ companyValidationForm: CompanyValidationForm
+ companyValidationForm Status: Object{formId: int, status: bool}
+ companyInternshipForm: array[Object{formId: int, Student:Student}]
+ companyInternshipForm Status: Object{formId: int, status: bool}
+ companyEmail: string
+ students: array[students]
+ companyUniqueFormLink: string
+ interningStudents: array[Student]
+ Company (companyName : string, companyEmail : string, companyId : int , approvalStatus : false)
+ getCompanyName ()
+ setCompanyName (value : string)
+ setApprovalStatus (value : bool)
+ getApprovalStatus ()
+ setCompanyId (value : int)
+ getCompanyId ()
+ setCompanyValidationForm (value: CompanyValidationForm)
+ getCompanyValidationForm ()
+ setCompanyValidationFormStatus (value: Object{formId: int, status: bool})
+ getCompanyValidationFormStatus ()
+ addCompanyInternshipForm (value: Object{formId: int, Student:Student})
+ getCompanyInternshipForm ()
+ setCompanyInternshipFormStatus (value: Object{formId: int, status: bool})
+ getCompanyInternshipFormStatus ()
+ getCompanyEmail ()
+ setCompanyEmail (value : string)
+ setCompanyUniqueFormLink (value : string, value: Student
+ getCompanyUniqueFormLink (value: Student
+ expireCompanyUniqueFormLink (value: Student)
+ addStudents(value: Student)
1
*
Student
- assignedEvaluator : Evaluator
- assignedTAs : array[TA]
- company : Company
- reportSubmissionStatus : bool
+ courseGrade : string
+ internshipForm: StudentInternshipReportForm
+ internshipFormSubmissionStatus: bool
- revisionCount: int
- evaluatorFeedbackStatus: bool
- evaluatorFeedback: filename
- revisionDeadline: Date
- reportSubmissionDeadline: Date
+ Student (revisionCount = 0,internshipFormSubmissionStatus = false, evaluatorFeedbackStatus = false)
+ getAssignedEvaluator ()
+ setAssignedEvaluator (value : Evaluator)
+ getAssignedTAs ()
+ addAssignedTA (value : TA)
+ IncreaseRevisionCount ()
+ getRevisionCount ()
+ getEvalautorFeedbackStatus ()
+ getEvalautorFeedback ()
+ getCourseGrade ()
+ setCourseGrade (grade : string)
+ setCompany (value : Company)
+ getCompany ()
+ setReportSubmissionStatus(value : bool)
+ getReportSubmissionStatus ()
+ getInternshipFormSubmissionStatus ()
+ setInternshipFormSubmissionStatus (value : bool)
+ setInternshipForm (value : associative array [multiple])
+ getInternshipForm ()
+ getRevisionDeadline ()
+ setRevisionDeadline (value : Date)
+ getReportSubmissionDeadline ()
+ setReportSubmissionDeadline (value : Date)
*
*
User
- full name: string
- surname : string
- id: int
- email : string
- username : string
- password: string
- courses : array[string]
- permissions: array[permission]
- notifications: array[notification]
+ User (full name : string, surname: string, id : int,username : string, email : string, courses : string)
+ getFullName ()
+ setFullName (value : string)
+ getSurname ()
+ setSurname (value : string)
+ getEmail ()
+ setEmail (value : string)
+ getId ()
+ setId (value : int)
+ getUsername ()
+ setUsername (value : string)
+ getPassword ()
+ setPassword (value : string)
+ getCourses ()
+ addCourse (value : string)
+ removeCourse (index : int)
+ getPermissions ()
+ setPermissions (value : Permission)
+ getNotifications ()
+ addNotifications (value : Notification)
+ removeNotifications (index : int)
Notification
- message : String
- date : Date
+ Notification (message : string, date : Date)
Permission
- getSetUserDetail: Bool
- getSetCourse: Bool
- getSetTA: Bool
- getSetStudent: Bool
- getSetEvaluator: Bool
- getSetEvaluator: Bool
- getSetCoordinator: Bool
- giveFeedback: Bool
+ getGetSetTA ()
+ setGetSetStudent (value : bool)
+ getGetSetUserDetail ()
+ getGetSetStudent ()
+ setGetSetUserDetail (value : bool)
+ setGetSetTA (value : bool)
+ getGetSetCourse ()
+ setGetSetCourse (value : bool)
+ getGetSetEvaluator ()
+ setGetSetEvaluator (value : bool)
+ getGetSetCoordinator ()
+ setGetSetCoordinator (value : bool)
+ setGiveFeedback (value : bool)
+ getGiveFeedback ()
CompanyValidationForm
+ owner: Company
+ formFile: File
+ CompanyValidationForm(owner: Company, fields: Object)
GradingForm
+ owner: Evaluator
+ GradingForm(owner: Evaluator, fields: Object)
Form
+ id: int
+ fields: Object{any}
+ formStatus: bool
+ Form(fields: Object, id: int();
+ getFormFields();
+ setFormFields();
+ setFormStatus(formstatus: bool);
+ getFormStatus();
CompanyInternshipForm
+ owner: Company
+ formFile: File
+ CompanyInternshipForm(owner: Company, fields: Object)
FeedbackForm
+ owner: Any
+ feedback: String
+ formFile: File
+ FeedbackForm(owner: any, fields: Object)
+ getFormFile();
+ setFormFile(formFile: File);
+ getFeedback();
+ setFeedback(feedback: String);
StudentInternshipReportForm
+ owner: Student
+ formFile: File
+ StudentInternshipReportForm(owner: Company, fields: Object)
Super User Admin
- name: string
- username: string
- password: string
- email: string
+ Super User(username : string, password : string, password : string)
+ editStudentDetails (student Id: int)
+ editEvaluatorDetails (evaluator Id: int)
+ editTADetails (TA Id: int)
+ editCoordinatorDetails (Coordinator Id: int)
+ assignStudent (student id : int, Evaluator/TA/Coordinator Id : int)
+ assignEvaluator (student id : int, Evaluator Id : int)
+ assignTA (student id : int, TA Id : int)
+ deleteUser (user id : int)
+ addAnnouncements (announcement : string)
+ addFAQ (faq: associative array [multiple])
+ addStudent (name : string, surname : string, id : int, email : string, course : string)
+ addEvalautor (name : string, surname : string, id : int, email : string, course : string)
+ addTA(name : string, surname : string, id : int, email : string, course : string)
+ addCoordinator(name : string, surname : string, id : int, email : string, course : string)
+ unAssignStudent(student id : string, Evaluator/TA/Coordinator id : int)
+ unAssignEvalautor(student id : string, Evaluator id : int)
+ unAssignTA(student id : string, TA id : int)
+ unAssignCoordinator(student id : string, Coordinator id : int)
+ getApprovedCompanyList()
+ modifyApprovedCompanyList()
+ modifyStudentGrade(student id : int)
Coordinators
- students : array[Student]
+ Coordinators()
+ getStudents ()
+ addStudent (value : Student)
+ getStudentName (student id: int)
+ getStudentReportSubmissionStatus (studentId: int)
+ getStudentSubmittedReport (studentId: int)
+ getCompanySubmittedForm (studentId: int)
+ requestCompanySubmittedForm (studentId: int, company : Company)
+ makeStudentDetailRow (studentId : int)
+ approveCompany (studentId : int, company : Company)
+ rejectCompany (studentId : int, company : Company)
+ addCompanyToApprovedList (company : Company)
+ getListOfApprovedCompanies ()
+ giveStudentFeedback (studentId : int ,feedback : string)
+ preApproveStudentInternshipReport(studentId : int)
*
TA
- students : array[Student]
+ TA()
+ getStudents ()
+ addStudent (value : Student)
+ getStudentName (student id: int)
+ getStudentReportSubmissionStatus (studentId: int)
+ getStudentSubmittedReport (studentId: int)
+ getCompanySubmittedForm (studentId: int)
+ requestCompanySubmittedForm (studentId: int, company : Company)
+ makeStudentDetailRow (studentId : int)
+ approveCompany (studentId : int, company : Company)
+ rejectCompany (studentId : int, company : Company)
+ addCompanyToApprovedList (company : Company)
+ getListOfApprovedCompanies ()
+ giveStudentFeedback (studentId : int ,feedback : string)
+ preApproveStudentInternshipReport(studentId : int)
1
Evaluator
- students : array[Student]
- gradingForm : Object{form: GradingForm, student: Student}
- gradingFormSubmissionStatus : Object{formId: int, Student: Student}
- approvedCompaniesList : array[Company]
+ Evaluator ()
+ getStudents ()
+ addStudent (value : Student)
+ getStudentName (student Id: int)
+ getStudentReportSubmissionStatus (studentId: int)
+ getStudentSubmittedReport (studentId: int)
+ gradeStudent (studentId: int, grade : string)
+ getStudentGrade (studentId: int)
+ getCompanySubmittedForm (studentId: int)
+ requestCompanySubmittedForm (studentId: int, company : Company)
+ makeStudentDetailRow (studentId : int)
+ approveCompany (studentId : int, company : Company)
+ rejectCompany (studentId : int, company : Company)
+ addCompanyToApprovedList (company : Company)
+ getListOfApprovedCompanies ()
+ giveStudentFeedback (studentId : int ,feedback : string)

Figure 2: UML Class Diagram of Internship System (High-quality version: https://github.com/zedyjy/CS319-Project/blob/main/Diagrams/anlysis%20report%20object%20class%20diagram.png)

3.2.4 Dynamic Models

3.2.4.1 Activity Diagrams

3.2.4.1.1 Student Report Processing

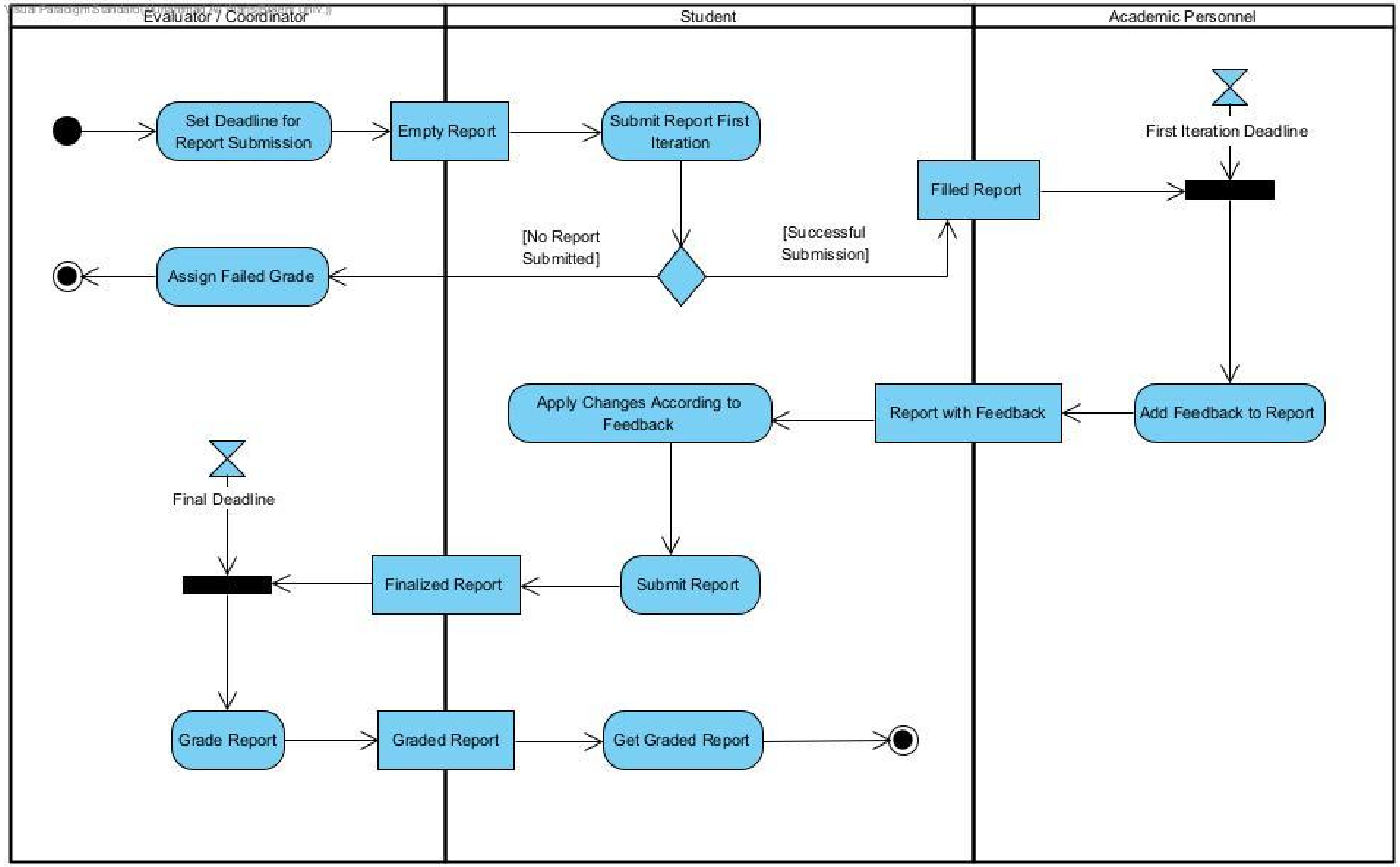


Figure 3: UML Activity Diagram of the Student Report Processing Process

The evaluator/coordinator sets the deadline for the current semester's internship report submissions. This creates instances of empty reports for all students enrolled in the course. The student submits the report as a first iteration. If by the end of the semester, the student has not submitted any report, they are assigned a Failing Grade. If the report was submitted by the student, this updates the report instance to be 'Filled', which is then sent to the responsible academic personnel. The academic personnel either adds feedback to the report if there is time for the final submission or Grades the student if the first iteration is done without leaving time for feedback. The report with the feedback is also accessible to the student. The student can make changes, update the new report, and submit it as a finalized one. If the final deadline has approached, the report is graded or more feedback is given. Otherwise, the report is then updated for the student with the Grade given.

3.2.4.1.2 Company Approval

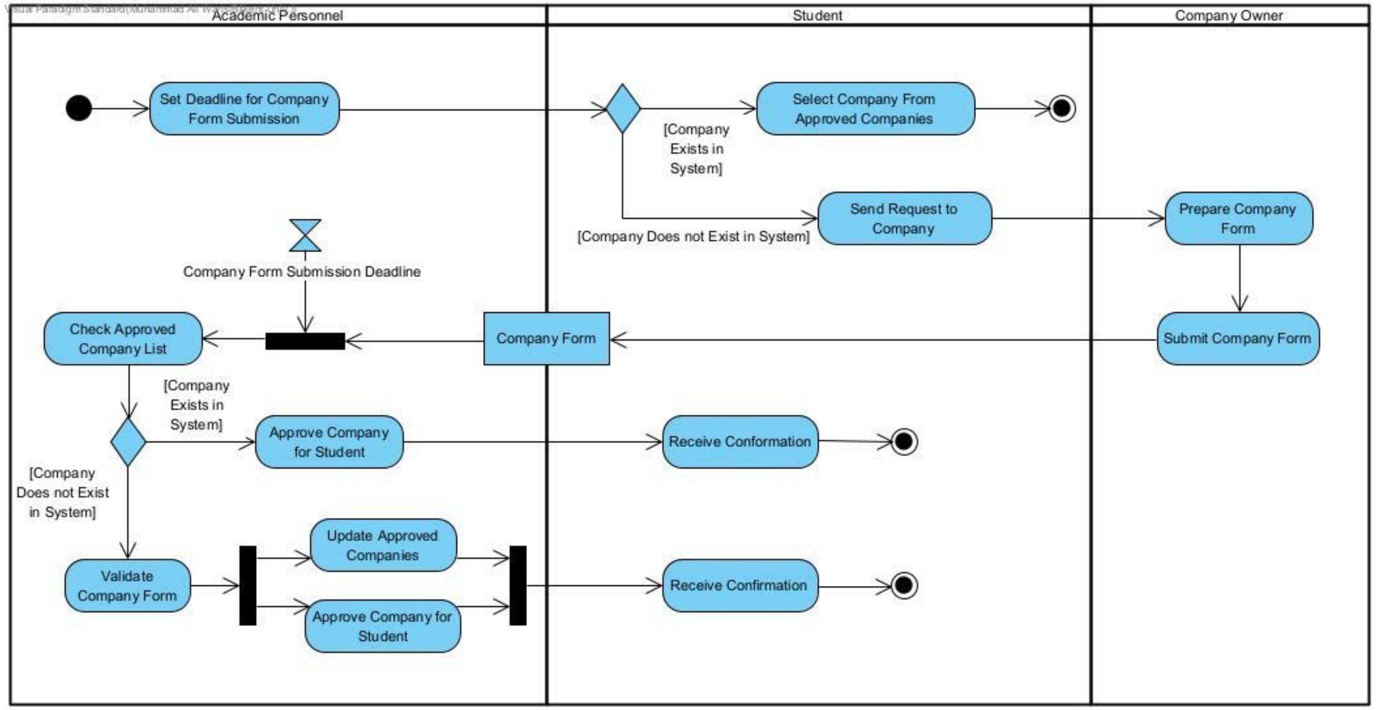

Academic Personnel
Student
Company Owner
Set Deadline for Company Form Submission
[Company Exists in System]
Select Company From Approved Companies
[Company Does not Exist in System]
Send Request to Company
Prepare Company Form
Company Form Submission Deadline
Check Approved Company List
Company Form
Submit Company Form
[Company Exists in System]
Approve Company for Student
Receive Conformation
[Company Does not Exist in System]
Update Approved Companies
Validate Company Form
Approve Company for Student
Receive Confirmation

Figure 4: UML Activity Diagram of the Company Approval Process

The evaluator/coordinator sets the deadline for the company form submission. If the company is already on the approved company list, it is selected as a company for the student for submitting internship work forms. If the company does not exist in the system, a request is sent to the company for approval, and/or more validation from the academic personnel is also done. When the company is selected by the student, and if it does not exist in the system before, the company validation form is prepared. The company can then submit the form before the deadline. This form is then sent to the Academic Personnel. If the company form submission deadline has not approached, a check is done to see if the company exists in the system again. If it exists, it is approved for the student. If the company does not exist in the system, the validation form is checked by the Academic Personnel and then this company is added to the approved company list, and approved for the student if the validation process was successful.

3.2.4.1.3 Company Work Report Processing

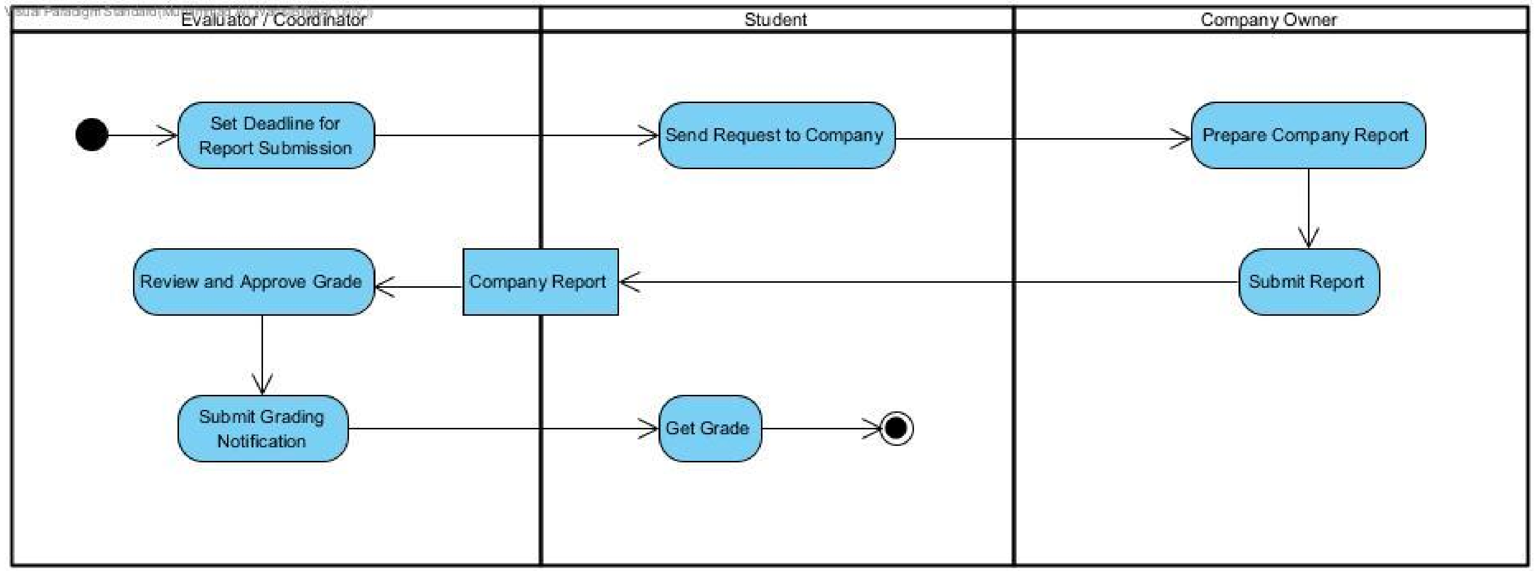


Figure 5: UML Activity Diagram of the Company Report Processing Process

The evaluator/coordinator sets the deadline for the report submission. A request is then sent to the company to their contact. The company then prepares the report for submission and submits it. The academic personnel then receive the report from the company and review/approve the report and Grade the form. This releases a notification for all stakeholders that a Grade was submitted, and the Grade is saved.

3.2.4.2 State Diagrams

3.2.4.2.1 Student State

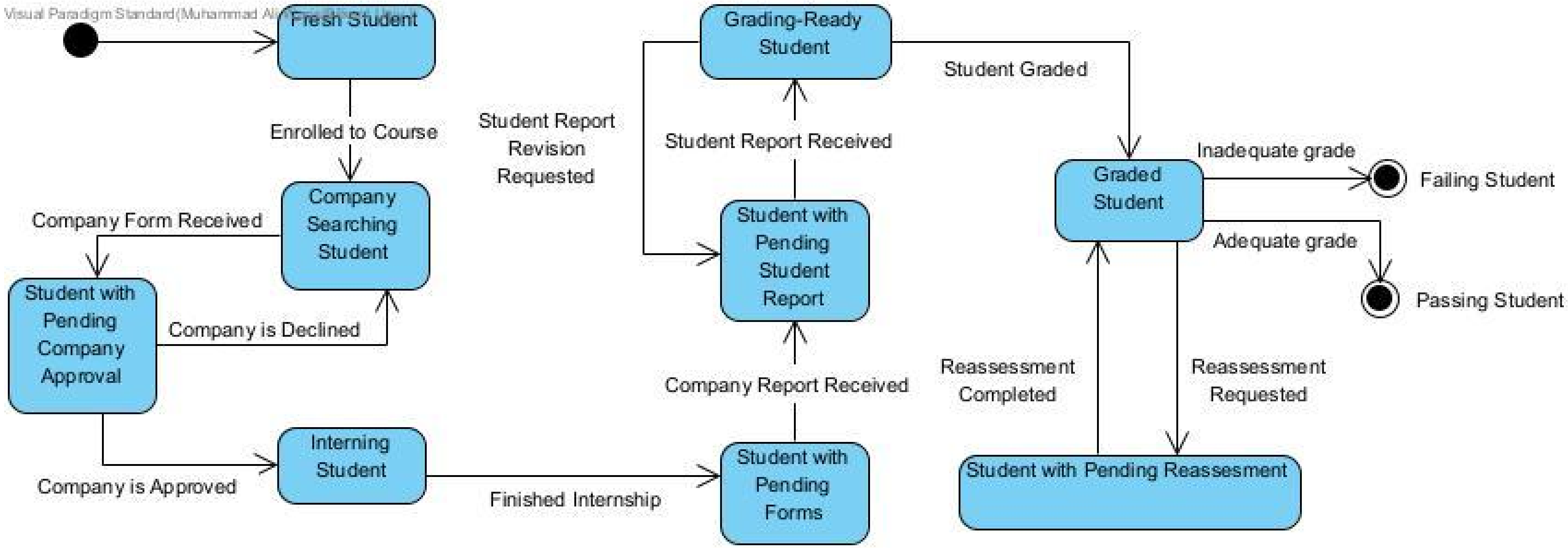


Figure 6: UML State Diagram for a Student

The student is registered as a Fresh Student. They are then a Company Searching for Students when they enroll in a course. When the company validation form is received they are updated to a Pending Company Approval State. If the form was declined, they go back to searching for a company. If the form was approved, they are now an interning student. When they finish the internship and submit reports, they are updated to a Student with Pending Company forms state. When the company work report is received, the students are now having pending Reports. If the report was approved they are ready for grading, if the report was not approved, they are again having pending reports. When the student is graded by the Academic Personnel, they are now a graded student. If they have a passing grade, they are passed, otherwise, they fail the course. if a reassessment was requested, they are now a student with Pending Reassessment. When the reassessment is complete they go back to being a Graded Student.

3.2.4.2.2 Company State

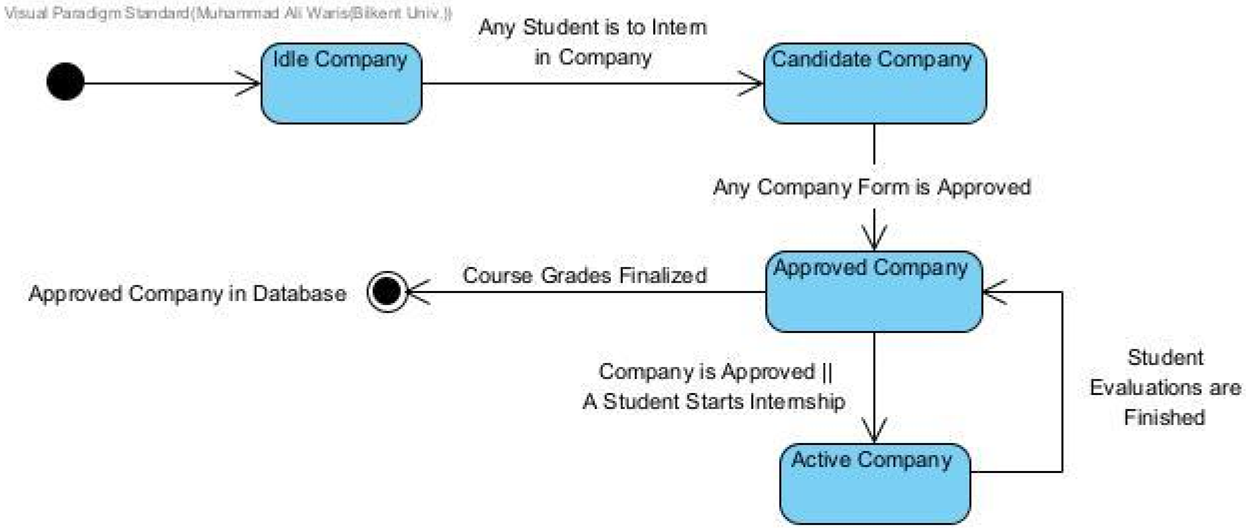

Visual Paradigm Standard(Muhammad Ali Waris(Bilkent Univ.))
Idle Company
Any Student is to Intern in Company
Candidate Company
Any Company Form is Approved
Approved Company
Course Grades Finalized
Approved Company in Database
Company is Approved || A Student Starts Internship
Active Company
Student Evaluations are Finished

Figure 7: UML State Diagram for a Company

A company starts as an ‘idle’ company, meaning they have not been assigned a student yet. When a student wants to intern in a company, it is now a Candidate Company waiting for approval. If it is approved, it is now an Approved Company. When a student starts their internship in this company, they are now an active company. The exit point is when the grades are finalized using other items and the Approved Company’s form.

3.2.5 User Interface

3.2.5.1 Navigational Paths

Users' capability to navigate between the pages is shown below. The relation between the pages is specified with the arrows.

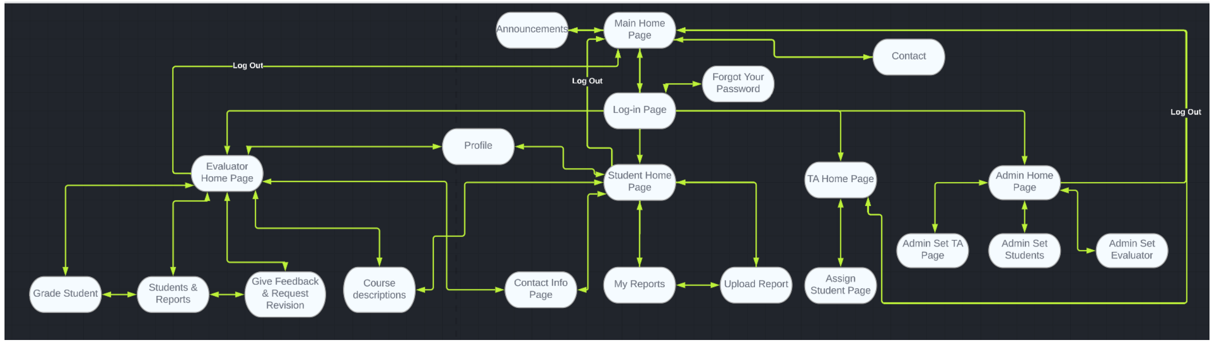


Figure 8: Navigational Paths

3.2.5.2 Screen Mock-ups

In this section, the main features of the system and how these features will appear in front of the user are explained. Please note that these interpretations are not from the final version of the system and do not represent the entire operation of the system.

3.2.5.2.1 Home Page

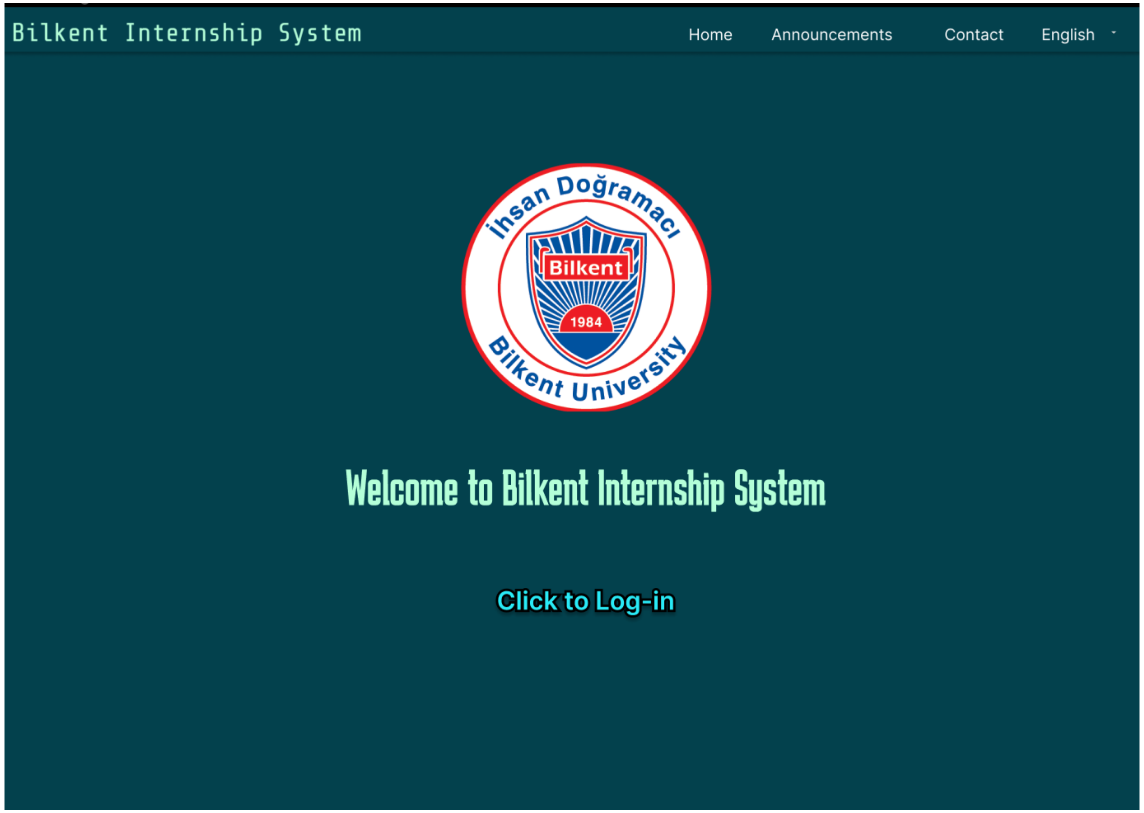

Bilkent Internship System
Home
Announcements
Contact
English
İhsan Doğramacı Bilkent University
Bilkent
1984
Welcome to Bilkent Internship System
Click to Log-in

Figure 9: Home Page Mock-up

The home page is the first starting page of the Bilkent Internship System, and this page consists of the main upper bar and login option.

At the same time, the user, via the upper bar, can access the language setting, announcements, and contact sections without logging in. And if they want to return to the main page again, they can return to the main login page by using the home button on the top bar.

3.2.5.2.2 Log-in Page and Forgot Password

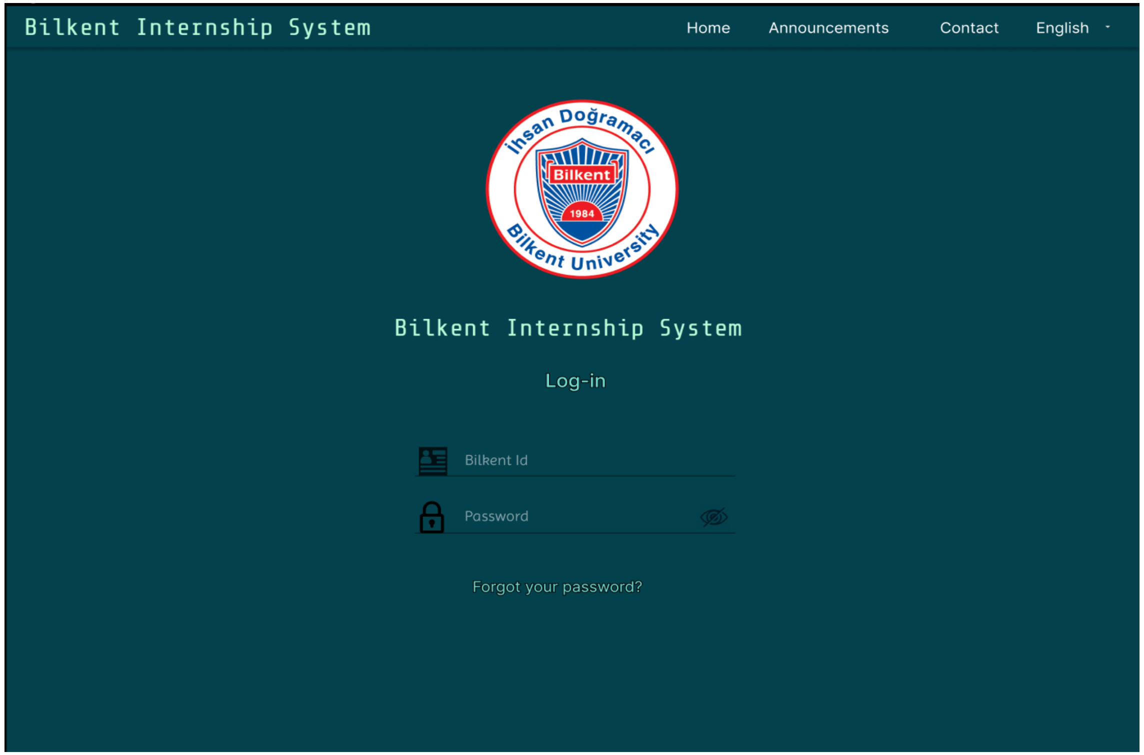
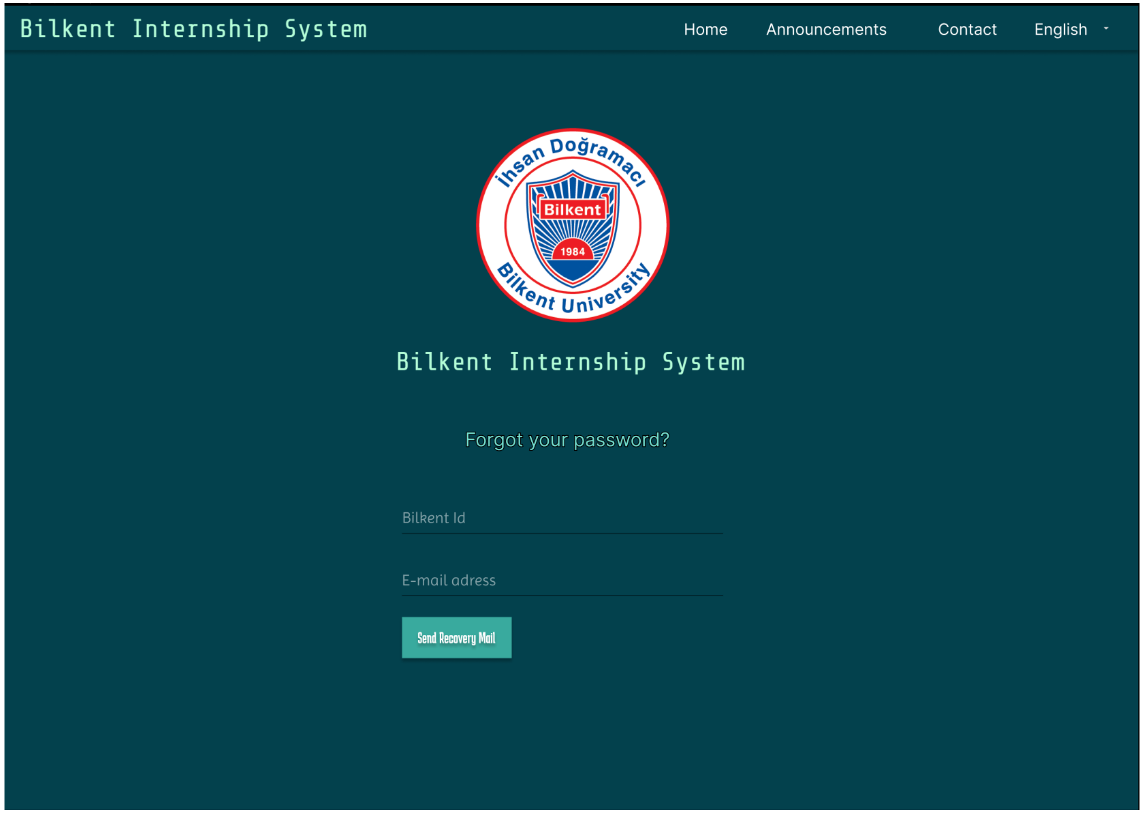


Figure 10: Login Page Mock-up

The login page accepts the username and password and creates a link to the home pages customized according to the user type. The Bilkent Internship System does not require registration because it works with each user type's currently used id and the Stars system's password. In addition, the "forgot password" button takes you to the "Forgot password page" where you can get a password recovery email.



Bilkent Internship System

Bilkent Internship System

Forgot your password?

Figure 11: Forgot Password Page Mock-up

With the help of this page, users who forget their passwords can receive password recovery mail by entering their Bilkent id and e-mail address registered in the system.

3.2.5.2.3 Evaluator Main Page

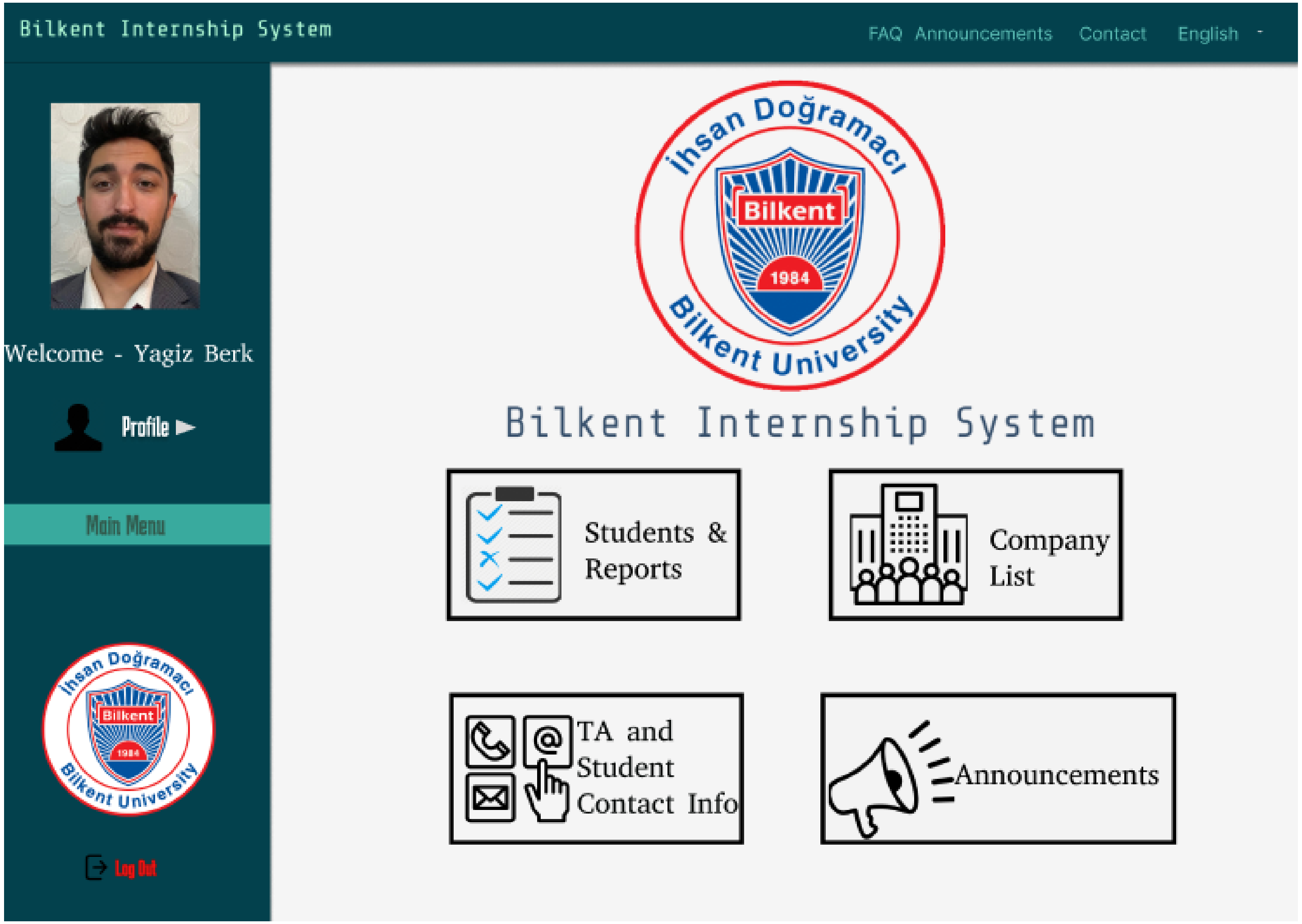
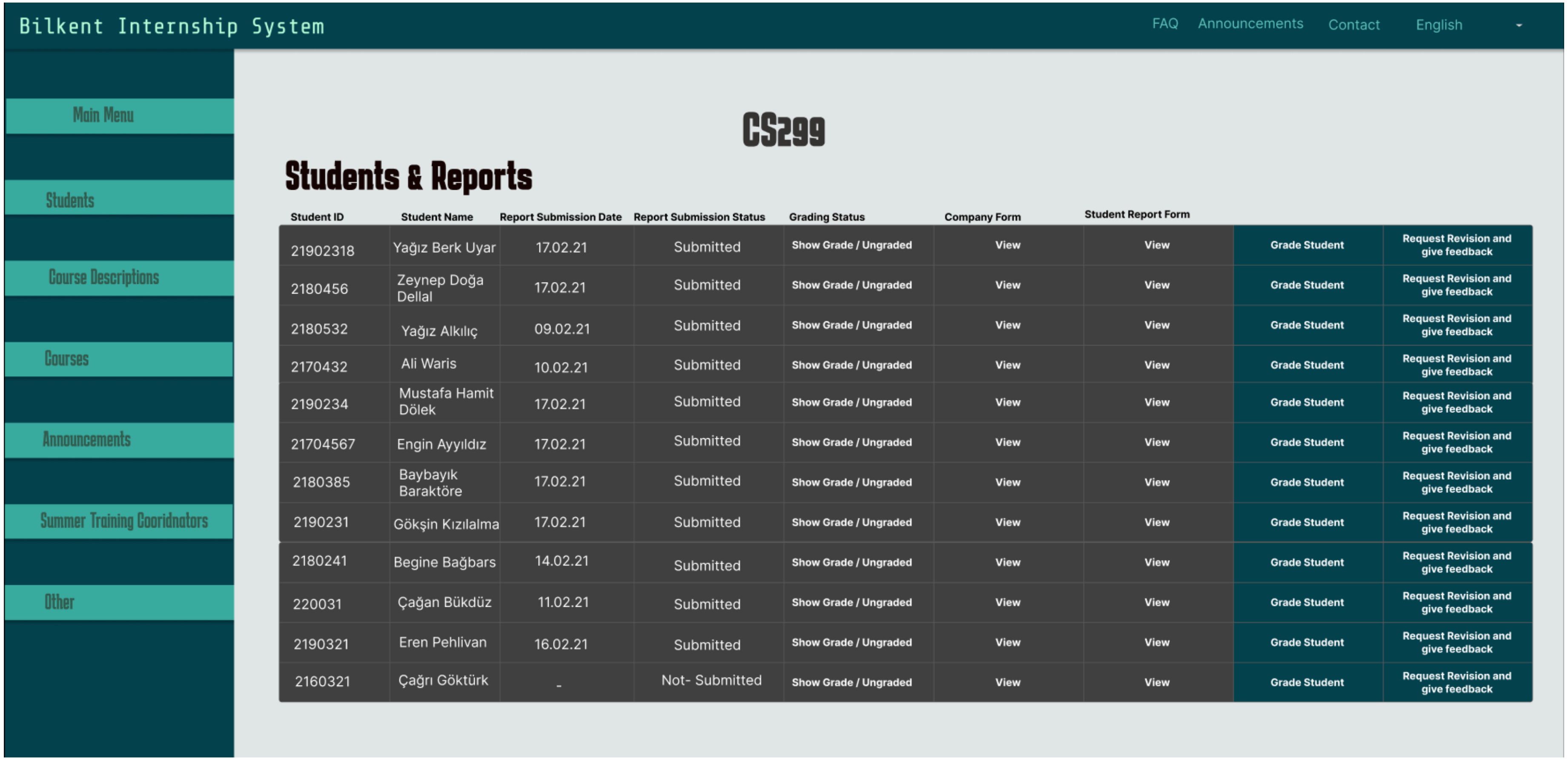


Figure 12: Evaluator Main Page Mock-up

After logging in as an evaluator, the system is directed to the home page specially designed for evaluators. On the left sidebar, there is a photo of the user from Bilkent's database, and below that, there is a welcome text with the user's name. The middle menu has four-page orientations; these are the Students & Reports page, Company List Page, TA-Student Contact Info page, and Announcements Page. Contact info and announcements page can also be accessed from the upper bar for further convenience on the other pages. The upper bar contains an additional FAQ button and the FAQ button leads to the FAQ page on the internship website of Bilkent

3.2.5.2.4 Evaluator Students & Reports Page



CS299

Students & Reports

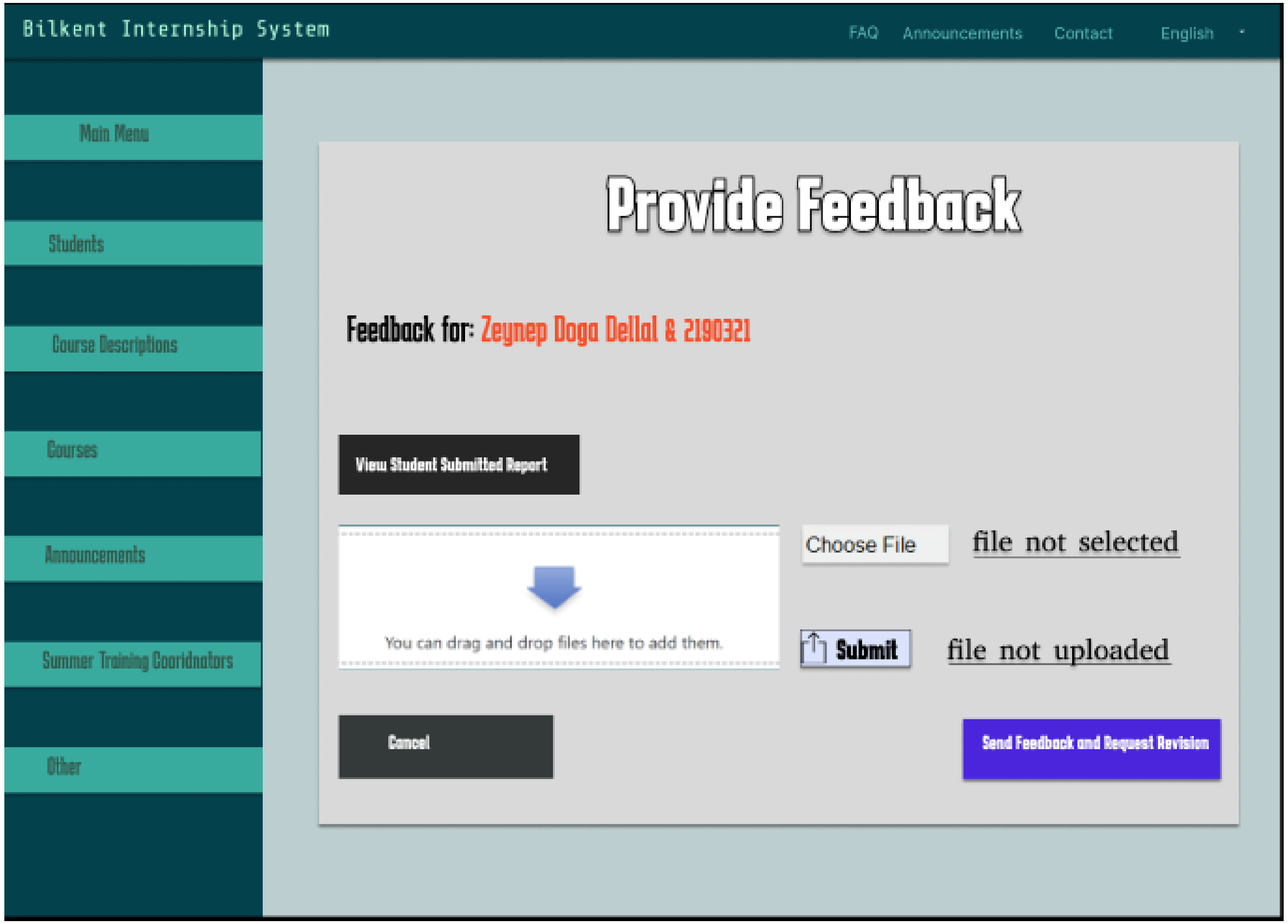
Student ID	Student Name	Report Submission Date	Report Submission Status	Grading Status	Company Form	Student Report Form		
21902318	Yağız Berk Uyar	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2180456	Zeynep Doğa Dellal	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2180532	Yağız Alkılıç	09.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2170432	Ali Waris	10.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2190234	Mustafa Hamit Dölek	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
21704567	Engin Ayyıldız	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2180385	Baybayık Baraktöre	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2190231	Gökşin Kızılalma	17.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2180241	Begine Bağbars	14.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
220031	Çağan Bükdüz	11.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2190321	Eren Pehlivan	16.02.21	Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback
2160321	Çağrı Göktürk	-	Not- Submitted	Show Grade / Ungraded	View	View	Grade Student	Request Revision and give feedback

Figure 13: Evaluator Students & Reports Page Mock-up

The Students & Reports page allows evaluators to see and track their assigned students and the information on the reports uploaded by these students. The table on the page contains all the necessary information about the student and their reports, such as the upload date of the uploaded reports, whether the report was uploaded, grading status, and company form. The page orientation where students will be graded is placed at the end of the student information. The evaluator determines the future of the student's report by clicking on the grading page or the request revision page from the end of the student’s row.

Also, the sidebar becomes one that contains the redirects of other pages to provide more practical use. Thanks to this sidebar, users can access the orientation of every page they can go to, including the home page.

3.2.5.2.5 Evaluator Request Revision and Give Feedback Page


Bilkent Internship System
FAQ
Announcements
Contact
English
Main Menu
Students
Course Descriptions
Courses
Announcements
Summer Training Cooridnators
Other
View Student Submitted Report
Choose File
Submit
Cancel
Send Feedback and Request Revision

Provide Feedback

Feedback for: Zeynep Doga Dellal & 2190321

You can drag and drop files here to add them.

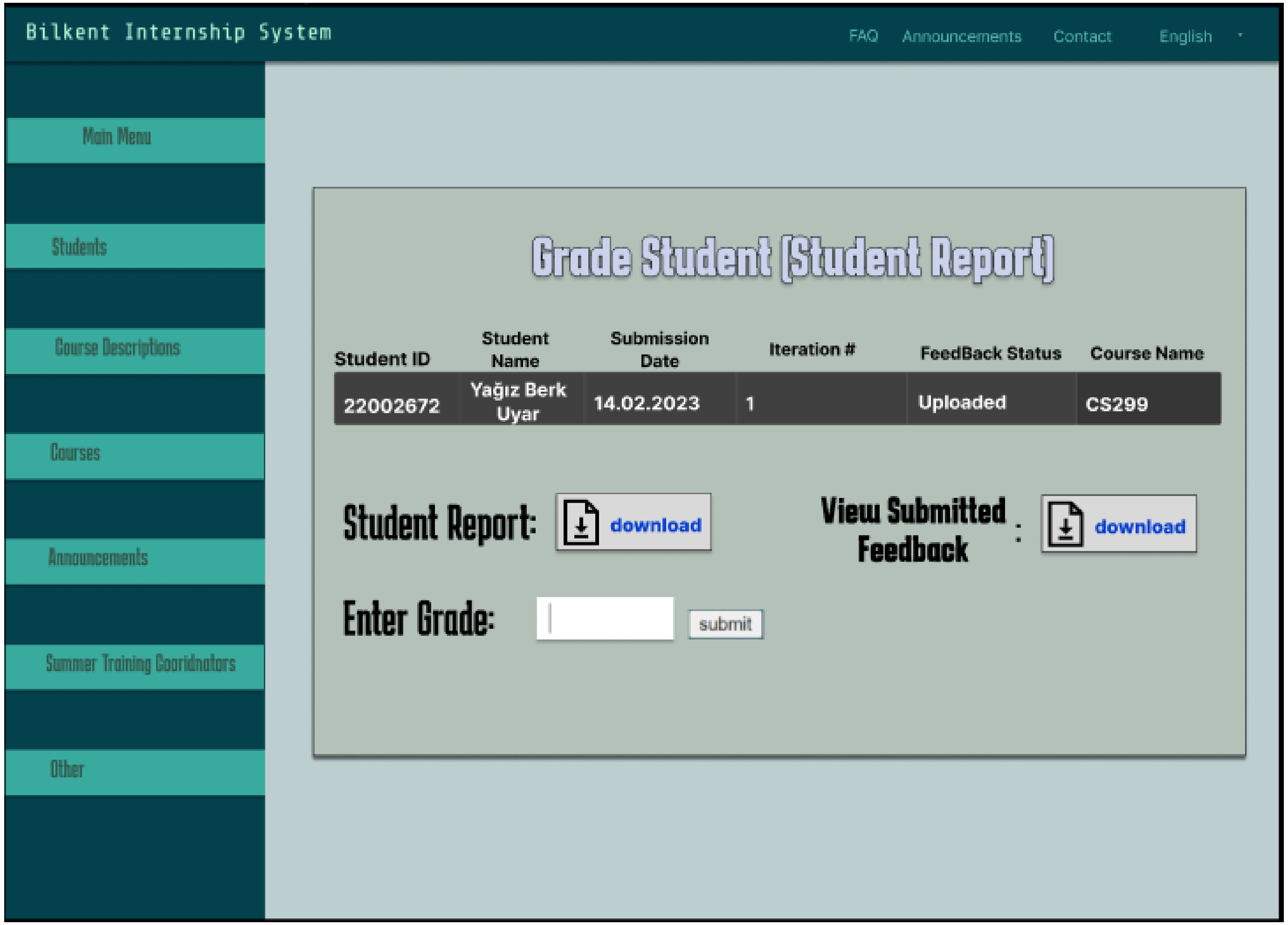
file not selected

file not uploaded

Figure 14: Evaluator Request Revision and Give Feedback Page Mock-up

On this page, Evaluators can give feedback to students they deem necessary. The submission box on the page will allow the evaluators to provide written feedback in text form and upload documents. The "View Students Submitted Report" button on the top left of the submission box will help the evaluator easily access the student report if necessary.

3.2.5.2.6 Evaluator Grade Student Page



Grade Student (Student Report)

Student ID	Student Name	Submission Date	Iteration #	FeedBack Status	Course Name
22002672	Yağız Berk Uyar	14.02.2023	1	Uploaded	CS299

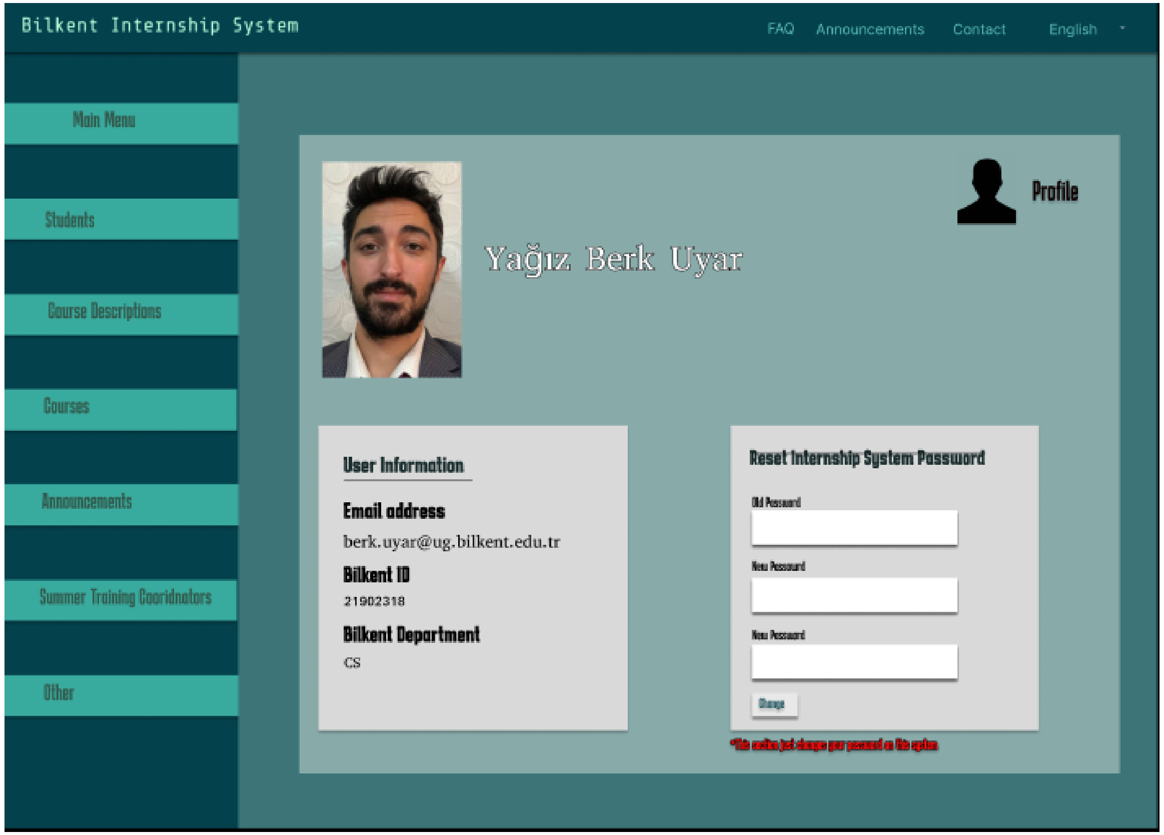
Student Report:

View Submitted Feedback :

Enter Grade:

Figure 15: Evaluator Grade Student Page Mock-up

On this page, evaluators can grade according to the sections of the reports of the student they choose from the reports page. At the same time, if evaluators do not want to give the grade at a time and continue the evaluation process, they can save the grade as a draft and finish the grading when they are completed later.3.2.5.2.7 Profile Page for Evaluators and Students



Yağız Berk Uyar

User Information

Email address
berk.uyar@ug.bilkent.edu.tr

Bilkent ID
21902318

Bilkent Department
CS

*This section just changes your password on this system

Figure 16: Profile Page for Evaluators and Students Mock-up

The profile page allows users to see and control their information in the system and to change only their passwords that are valid in the internship system.

3.2.5.2.8 Student Home Page

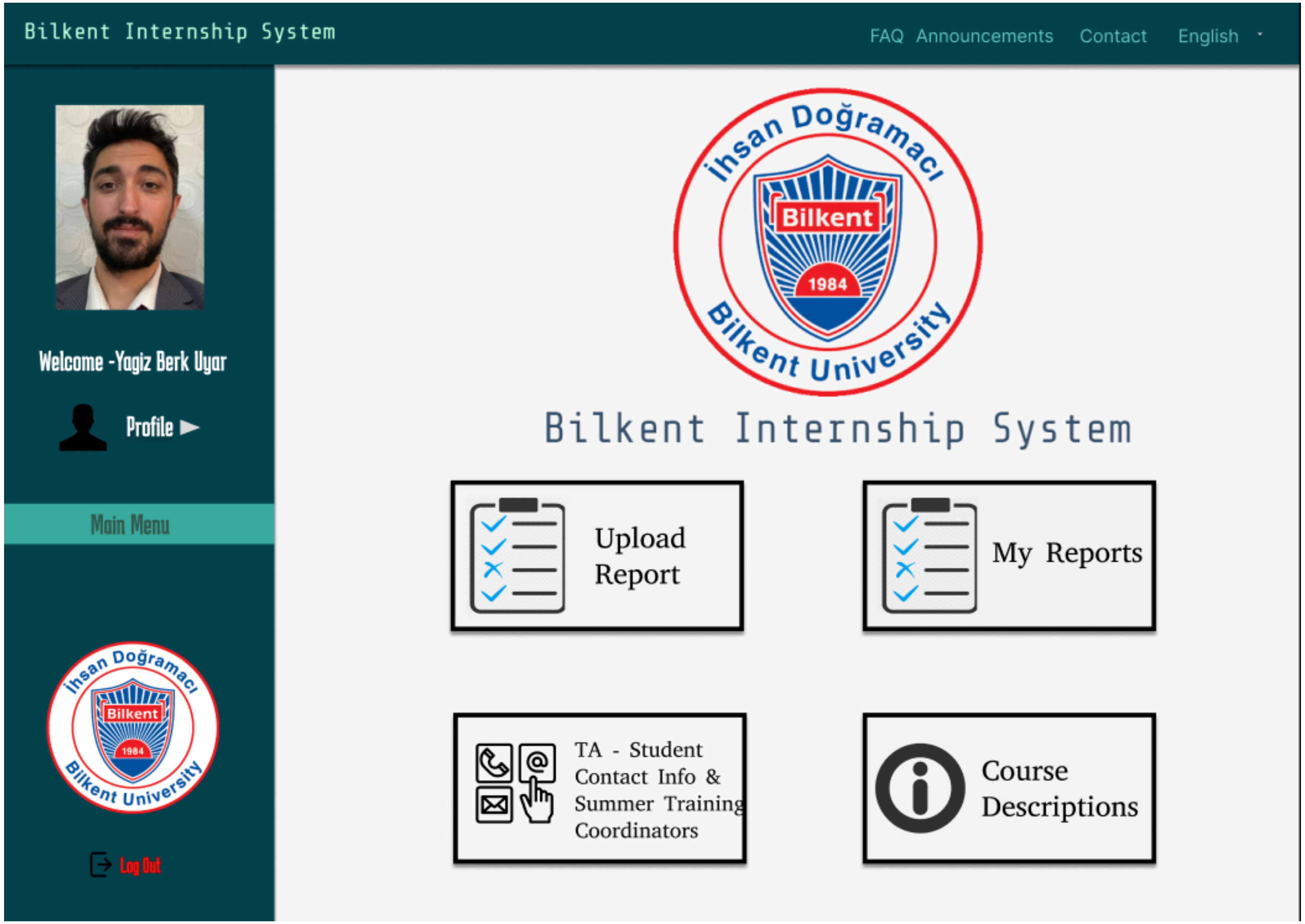
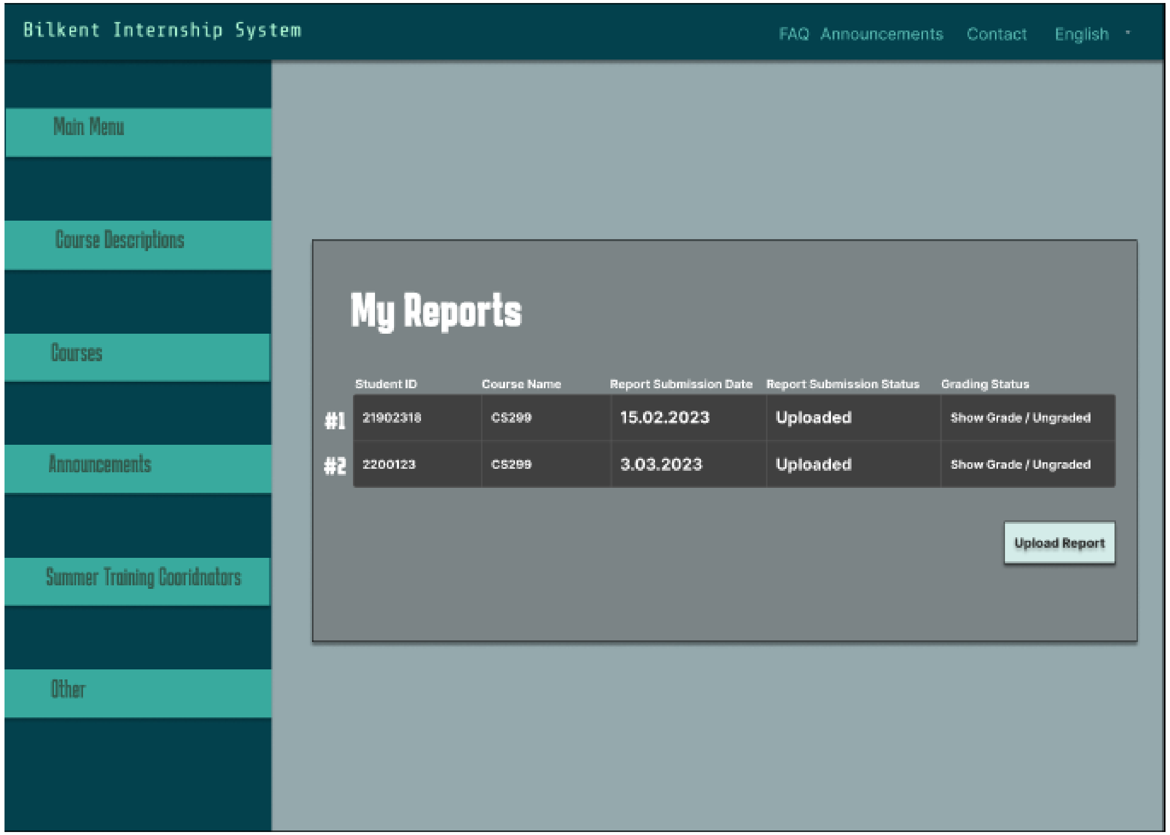


Figure 17: Student Home Page Mock-up

Student Home Page has a similar design to Evaluator Home Page. From this page, students can access the most necessary pages for them, such as Upload Report and My Reports, from the middle menu, and they will also be able to easily enter informational pages, such as announcements and frequently asked questions, with the help of the upper bar

3.2.5.2.9 Student My Reports Page



My Reports

	Student ID	Course Name	Report Submission Date	Report Submission Status	Grading Status
#1	21902318	CS299	15.02.2023	Uploaded	Show Grade / Ungraded
#2	2200123	CS299	3.03.2023	Uploaded	Show Grade / Ungraded

Figure 18: Student My Reports Page Mock-up

My Reports Page allows students to keep track of the reports they have previously uploaded, if any. Users who wish can continue to the report upload page by clicking the "Upload Report" button at the bottom of this page.

3.2.5.2.10 Student Upload Report Page

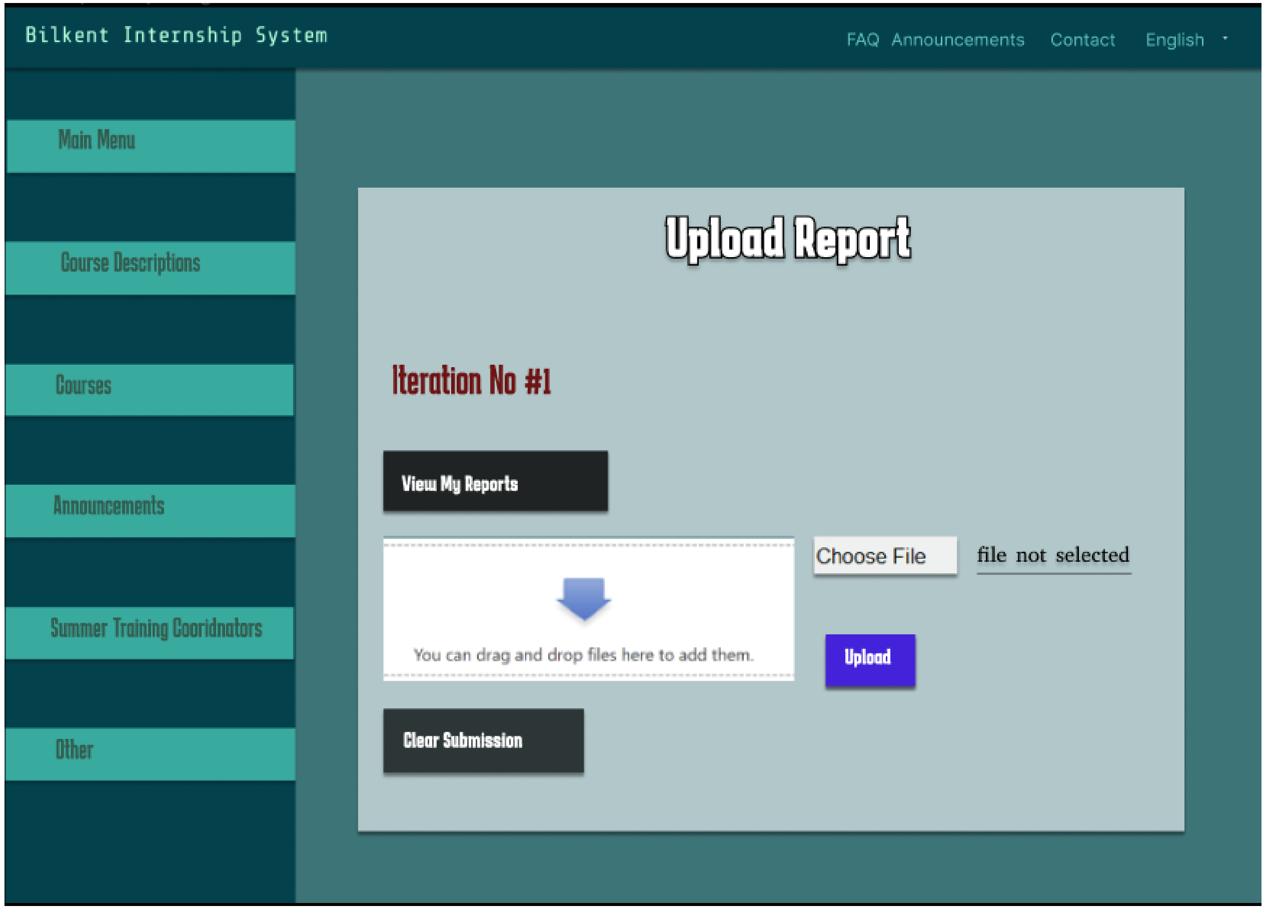
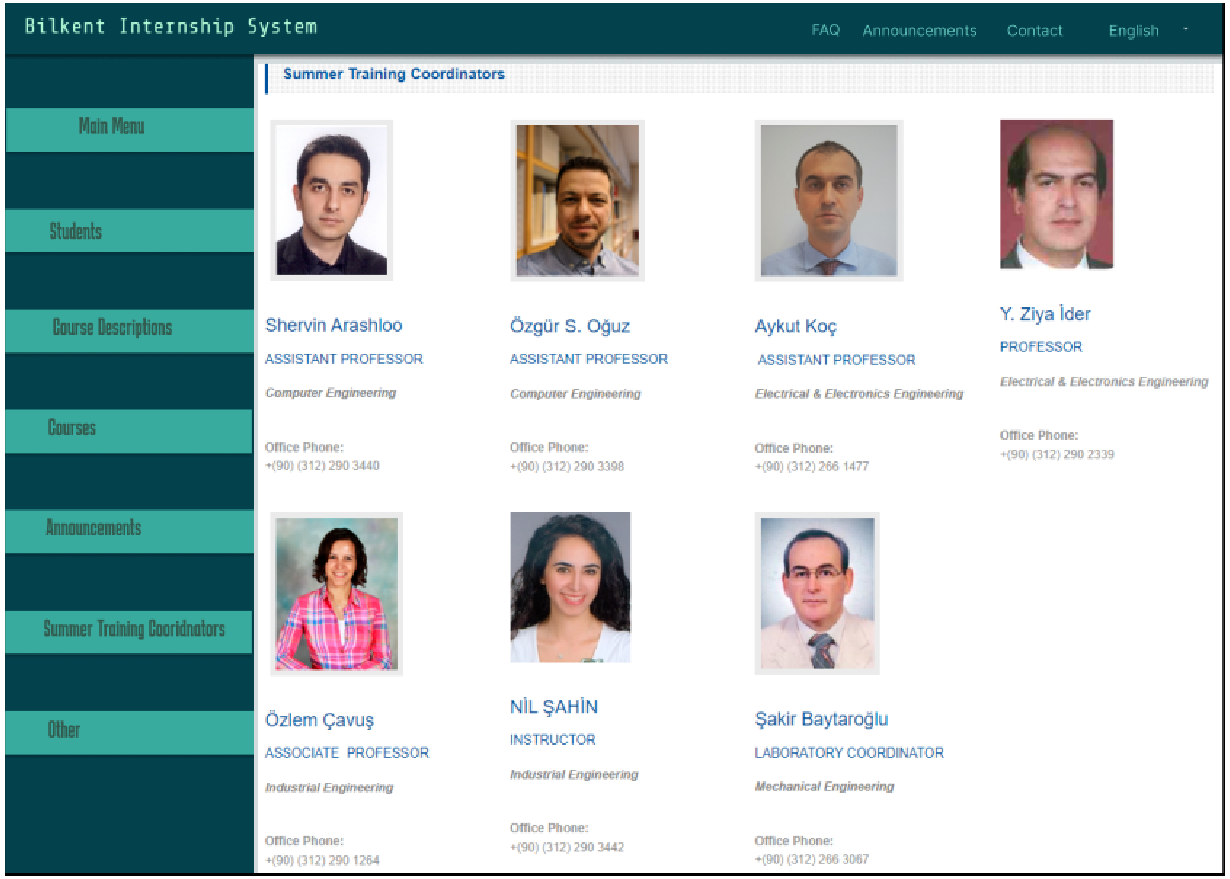


Figure 19 : Student Upload Report Page Mock-up

On this page, students can upload their Reports in the format specified for them. As on the My Reports page, this page also has a button that allows them to return to the My Reports page for user convenience.

3.2.5.2.11 Summer Training Coordinators Contact Info Page



Summer Training Coordinators

Shervin Arashloo
ASSISTANT PROFESSOR
Computer Engineering
Office Phone:
+(90) (312) 290 3440

Özgür S. Oğuz
ASSISTANT PROFESSOR
Computer Engineering
Office Phone:
+(90) (312) 290 3398

Aykut Koç
ASSISTANT PROFESSOR
Electrical & Electronics Engineering
Office Phone:
+(90) (312) 266 1477

Y. Ziya İder
PROFESSOR
Electrical & Electronics Engineering
Office Phone:
+(90) (312) 290 2339

Özlem Çavuş
ASSOCIATE PROFESSOR
Industrial Engineering
Office Phone:
+(90) (312) 290 1264

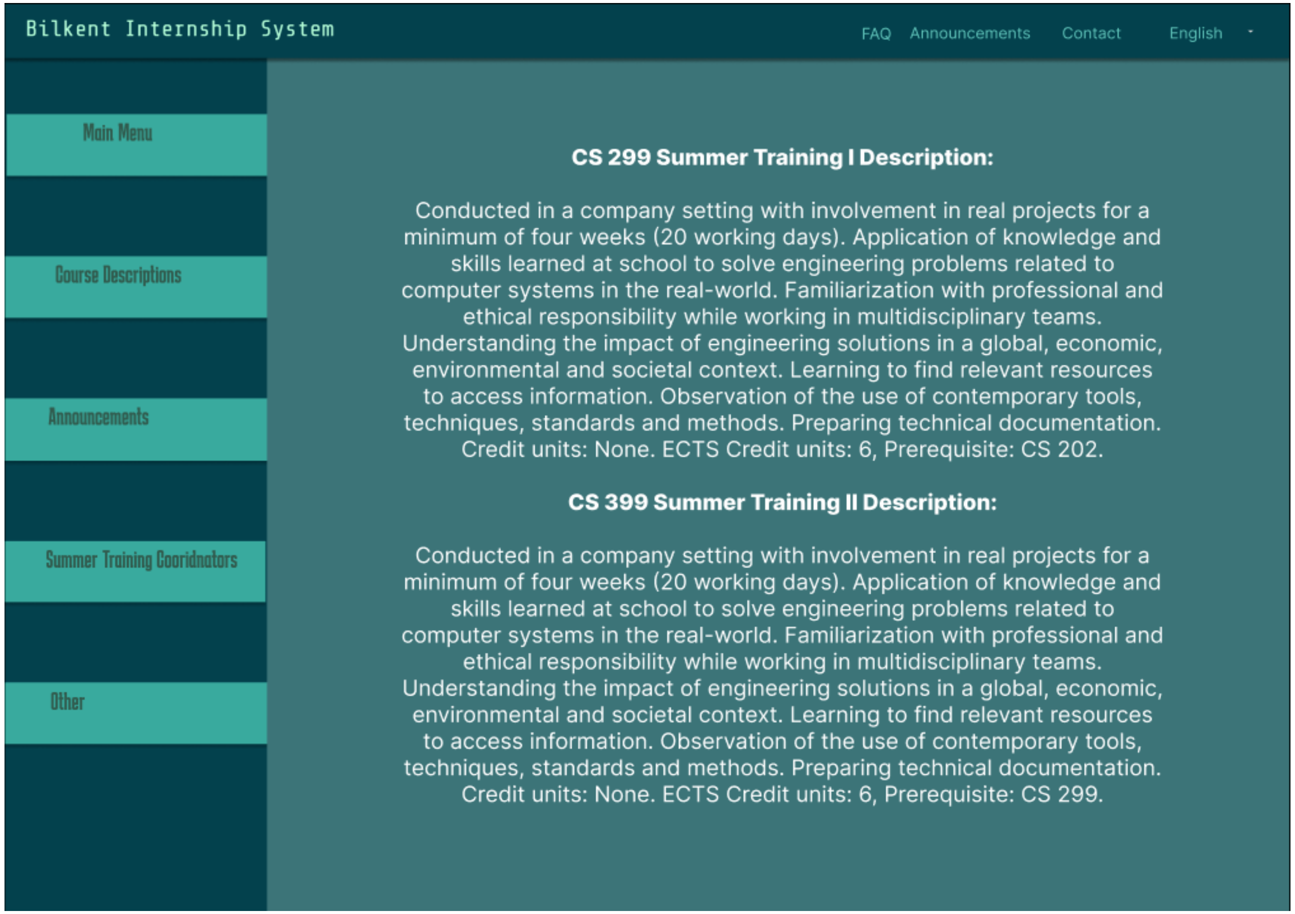
NİL ŞAHİN
INSTRUCTOR
Industrial Engineering
Office Phone:
+(90) (312) 290 3442

Şakir Baytaroğlu
LABORATORY COORDINATOR
Mechanical Engineering
Office Phone:
+(90) (312) 266 3067

Figure 20: Summer Training Coordinators Contact Info Page Mock-up

Users can access the contact details of the coordinators here.

3.2.5.2.12 Course Descriptions Page



CS 299 Summer Training I Description:

Conducted in a company setting with involvement in real projects for a minimum of four weeks (20 working days). Application of knowledge and skills learned at school to solve engineering problems related to computer systems in the real-world. Familiarization with professional and ethical responsibility while working in multidisciplinary teams. Understanding the impact of engineering solutions in a global, economic, environmental and societal context. Learning to find relevant resources to access information. Observation of the use of contemporary tools, techniques, standards and methods. Preparing technical documentation. Credit units: None. ECTS Credit units: 6, Prerequisite: CS 202.

CS 399 Summer Training II Description:

Conducted in a company setting with involvement in real projects for a minimum of four weeks (20 working days). Application of knowledge and skills learned at school to solve engineering problems related to computer systems in the real-world. Familiarization with professional and ethical responsibility while working in multidisciplinary teams. Understanding the impact of engineering solutions in a global, economic, environmental and societal context. Learning to find relevant resources to access information. Observation of the use of contemporary tools, techniques, standards and methods. Preparing technical documentation. Credit units: None. ECTS Credit units: 6, Prerequisite: CS 299.

Figure 21: Course Descriptions Page Mock-up

Students and evaluators can check the descriptions of the internship courses they are responsible for on this page

3.2.5.2.12 Announcements & Company List Pages

These pages do not have a user interface. Clicking on these pages redirects the user to the relevant http://mf.bilkent.edu.tr/ page

4. Conclusion

The Quaso Internship Program was created to not only make handling internship reports simpler but also to improve the user experience for all parties involved. Users can easily navigate the program and complete their duties quickly thanks to its aesthetically pleasing and functional interface, which has a low cognitive load. To keep all users updated on the state of their reports, deadlines, and feedback, the application also offers real-time updates and notifications. Additionally, the user-friendly layout of the program and the centralization of documents and information lessen the possibility of mistakes and confusion, ensuring the efficiency of the internship administration process.

The Quaso Internship Program's capacity to improve the caliber of internship reports is a major additional benefit. Students can concentrate more on the content of their reports thanks to the system's streamlined and streamlined process, which frees them from having to worry about the administrative duties involved in the process. The system's centralized platform also enables better report tracking and management, making it simpler for evaluators to offer thorough criticism and development suggestions. This feature makes sure that students get insightful criticism and direction, which results in internship reports of better quality.

In conclusion, engineering students and employees at Bilkent University can benefit greatly from the system. The system improves cooperation and communication between students, assistants, and evaluators, streamlines the management of internship reports, and raises the caliber of internship reports. A smooth and effective internship administration process is ensured by the program's user-friendly design, centralization of papers and information, and real-time updates. The Quaso Internship Program is anticipated to be an indispensable instrument for students and staff, simplifying their lives and raising the standard of the internship management process thanks to its many advantages.

5. Improvement Summary

The functional requirements section has been deleted, and figure captions have been added to enhance clarity. Sequence Diagrams have been removed from the project. The Object & Class Model has undergone modifications, introducing Form classes to improve form handling and adapting all stakeholder classes to align with the new Form Object structure. Activity Diagrams and State Diagrams have been included to provide a more comprehensive understanding of the system's behavior. More descriptive trade-offs have been incorporated, offering a deeper analysis of the project's choices. Links to high-quality images have been added, ensuring better visualization. Text and image formatting issues have been resolved for improved presentation. The Use Case diagram now includes statistics. Based on feedback and course descriptions, almost every user interface page has been updated, with the addition of coordinators contact info page and course descriptions page. Readability has been enhanced by updating some font types. Lastly, realistic data has been included in the User Interface designs to provide a more authentic representation.

6. References

Image sharing and image hosting service. *Imgur*. Retrieved 21 May 2023, from https://imgur.com/

The Collaborative Interface Design Tool. Figma. (n.d.). Retrieved May 5, 2023, from https://www.figma.com/

Lucid visual collaboration suite: Log in. Lucid visual collaboration suite. (n.d.). Retrieved May 5, 2023, from https://app.lucidchart.com/

Security-first diagramming for teams. draw.io. (n.d.). Retrieved May 21, 2023, from https://www.drawio.com/

Welcome - faculty of engineering summer training system. (n.d.). Retrieved May 21, 2023, from http://mfstaj.bilkent.edu.tr/