Bilkent University

Department of Computer Engineering

Internship Report Management System

Project short-name: Bilport

Analysis Report

Arda İynem, Atika Zeynep Evmez, Ege Çenberci, Yağız Özkarahan, Zeynep Naz Sevim

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

Analysis Report
March 26, 2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object-Oriented Software Engineering course CS319

Contents

1. Introduction

This analysis report was created for the CS-319 course taught in Bilkent, in the Spring 2023 semester. The project aims to create an easy and reliable internship management system to handle the internship processes of Engineering Faculty undergraduate students in Bilkent University. It is the purpose of this project to automate the entire processes of the Engineering Faculty's summer training courses, which will save the valuable time of instructors, students and other participating parties. The details of the current system employed to manage summer training courses and the details of the system that is intended to replace it will be discussed in the following sections.

2. Current System

The current system employed in Bilkent University has some major drawbacks. After doing their internship, students are expected to upload their internship report on the Moodle page of the course they are taking. After the deadline of the report submissions have passed, all the reports submitted by students are downloaded and uploaded to various Google Drive folders. Since which of the courses CS 299 or CS 399 are taken by the student, and which department member is grading the students matter; a huge number of folders and sub folders are created. The folders are then shared by the evaluators selected for the course. If there are some teacher's assistants assigned to the course, the reports are shared with them for preliminary quality checking. Afterwards, the evaluators check the students' reports and give feedback. The feedback given by evaluators is either sent through the registrar's office or via other channels. Then, the revised reports are sent to the registrar's office, where they are uploaded to the corresponding Drive folders. Then, the feedback process is repeated until the report is deemed satisfactory by the evaluator, and the final grade of the student is posted in the SRS system.

There are multiple problems that are easy to spot in the current system. Although the Moodle system is used for initial report uploads, it is not utilized in the process any further than that. The most vital parts of the report evaluation, the evaluator feedback and revision, are not conducted on the Moodle system. This inconsistency is highly inefficient, as all the reports submitted have to be first downloaded from Moodle, grouped, and then uploaded to Google Drive. The process reports undergo before they are ready for evaluation is a complete waste of valuable man-hours.

The back and forth procedure of report revision is also sub-optimal. As mentioned before, the feedback is given to the students through the Registrar's office or via other channels, such as email. The feedback going through the Registrar’s office is, again, a waste of valuable time and is susceptible to human error. A simple mistake in communication, such as sending the wrong revised report or the wrong feedback, could have gone unnoticed because of the unnecessary actors in the system. Sharing feedback via email etc. is a slightly better method as it is more direct than to send everything through the Registrar’s office. However, using systems that are not specifically designed for teaching purposes hinders the fluidity of the process.

The final step of the report evaluation, the grade assignment, is also made on a completely different system. The evaluators often track the students they are responsible for in Excel sheets, and transfer each student’s grade from one document/system to another numerous times. The process of tracking grades manually is prone to error, and could be handled in a more efficient manner.

Considering the drawbacks of the current system, it would be ideal to have the students upload their reports to a single system, upon which the evaluators can give direct feedback. Upon this system, each iteration of the internship report would be saved with its feedback, making vital information easily accessible. The students would be graded through the system, with the option to export the relevant information for grading and documentation purposes.

3. Proposed System

3.1 Non-functional Requirements

3.1.1 Usability

- The main purpose of this project is to make the convoluted internship report system simple, easy and practical to use. With this in mind, the resulting website should be easily usable for all actors.
- The user interface will be simple and compact, but still modern and stylish. All users should be able to navigate the website easily, without getting lost in the interface or needing a manual.
- High priority key functionalities such as submitting reports, viewing report status, giving feedbacks, viewing feedbacks etc. will not be hidden between menus. They will be easily available on the screen, in a way which will make the process as easy as possible.
- The default light color scheme of the website won’t be too bright in a way that would make the user uncomfortable while browsing. Soft color tones will be used.
- In addition to the default light mode, the website is also planned to include a dark mode, to make it more comfortable to use for all users.

3.1.2 Compatibility

- The general goal is to make our website compatible for all kinds of hardware, operating systems and browsers, so the website can offer the same experience for all users.
- The layout of the website will be compatible for monitors with different sizes and resolutions without any drawbacks or loss of functionality.
- The website will run on all modern internet browsers without any loss of functionality.
- Since the resulting application will be a website and not a separate software, the system performance shouldn’t be impacted any more than a regular

website. Therefore, the website will not use a significant portion of RAM, CPU, GPU or any other system part (although the portion can change depending on the browser, it should not be more than %25 higher than any ordinary web page displayed on the same browser).

3.1.3 Performance

- We want our website to be readily available and running smoothly for all users. It is not desired for the users to experience tedious waiting screens and slow loading times, so the application will be optimized accordingly.
- The main constraint for the website performance should be the internet connection speed of the user, in which we have no control of. As long as the user has a stable and sufficiently fast internet connection (at least 3 Mbps), the website will also be sufficiently fast to use with no constraints.
- Key actions such as downloading or uploading reports and viewing or publishing comments and feedbacks will be handled in no longer than 3 to 5 seconds (given a stable internet connection, as previously mentioned).
- There might be time periods where many users rush to enter the website at the same time, such as when the feedbacks are given or when all grades are announced. Considering this possibility, the website is planned to be able to run without freezing or crashing even when all the registered students are using the website simultaneously (therefore the upper limit is around the number of typically enrolled students).

3.1.3 Security

- It is of utmost importance for all user information to be invisible to all unauthorized parties.
- The user passwords will be encrypted in the database, to be protected in the undesired possibility of a security breach.
- The student reports will be protected, as they are private between the students and the course staff (such as their assigned evaluator and TA).

Students won’t be able to see or download each other’s reports, as they will not be authorized to do so.

- Similarly, students will not be able to view each other’s information through the website. Only the course staff will be authorized to view the students and their information.
- When a semester ends, all existing student accounts of the semester will be deleted from the system. This is done to prevent any complications or unwanted scenarios from happening in the future.

3.2 Pseudo Requirements

- The project will be developed entirely as a stand-alone web application. It is not planned or desired for the project to rely on other websites; it should be self-sufficient for all necessities of the university’s internship report management systems.
- The project will be coded in an object oriented manner, in line with the requirements of the course.
- To achieve object orientation with the implementation of backend, Java programming language with Spring framework and MongoDB database are planned to be used. For the frontend, JavaScript programming language with React library will be used. Bootstrap library is also planned to be used to make HTML/CSS more efficient. That being said, these are not requirements or constraints and it is possible for these technologies to change during the implementation. We may leave some of them out or employ new technologies.
- All stages of implementation will be available to be tracked from GitHub by the teaching assistants and course coordinators.

3.3 System Models

3.3.1 Scenarios (Functional Requirements)

User types:

- Super Admin
- Admin
- Student
- TA
- Evaluator
- Supervisor

a. Sign Up/Enrollment

None of the users can enroll to the system by themselves, they must be added to the system by an authorized person, except for super admin. So, there is not any ''sign up” feature in this application. Super admin and admins will enroll all of the other users.

b. Login

BilPort will have a login page, separate from the STARS system but the login credentials are the same with its credentials, it will only require an ID and a password. So if the department decides to shift this system to the STARS, it will be possible.

After their enrollment process has completed, all users will be able to login to the system by entering their IDs and passwords. They can also reset their password if they forget it.

c. Functionalities about Student Users

A student will see their reports and reports’ information in this application. They will also upload and download reports to the system and will be able to download the TA’s or evaluator’s feedback. They will also see their information overall status. They can find a specific report in the list of reports. In the list of reports they can track each report’s details.

They will also enter their supervisor’s information, such as email address. So that supervisor can log into the system as well.

d. Functionalities about Evaluator Users

An evaluator can see the students assigned to them by super admin or admins as a list which can be queried customarily and see these students’ information and report submissions. They can give feedback to these reports, they can approve them or decline them, as well. Also they can see the internship evaluation form which is filled by the student's supervisor.

An evaluator can also see the list of all students, they can see the information about each student but they cannot see all stıdents’ reports. They also can see their supervisee’s profiles and report submissions.

An evaluator can see all TAs as a list, and can see each TAs profile and information.

e. Functionalities about TA(s)

The main task of the TA(s) in the system is to check first submissions of students by means of quality. They will give feedback accordingly but this feedback has not any grade, it will be more like a suggestion.

They can see all students and their information such as status, etc. They can download the student’s report and upload a suggestion feedback.

Another task that we plan to assign to the TA(s) is to check whether the supervisor's signature is the same as the signature in the internship acceptance letter.

f. Functionalities about Admins

In this system, an admin can be an evaluator or a non-academic staff member. Admin can enroll or delete students, TAs, and other admins. They can list all of the students, evaluators, TAs, admins, supervisors at separate pages. All of these lists can be customarily queried and a specific user can be found by search bar (as the all other lists in the application).

Admins can assign students to evaluators and export information from the system too. They can assign supervisees to their supervisors. At the end of the semester they can export the grade information of all the students.

g. Functionalities about Super Admins

In this system, there will be only one super admin, which may or may not be an academic staff member. The main task of super admin is to initiate the system by enrolling all of the users by several excel files. They can also assign students to evaluators. At the end of the semester, they will close the system.

h. Functionalities about Supervisors

Supervisors are mentors in the company assigned to the student in the internship period. They can be enrolled in the system by students. the student should enter their supervisor’s information, name, mail address, and etc. Supervisor’s ID and password are first determined by the system but evaluators can change their password. After supervisors log in, they will be directed to the “Summer Training Evaluation Form”, they can fill and save this form. And they can also make changes on these forms after they first fill it.

Log In & Change Password Package

Use case name: Log In

Participating Actor(s): User

Flow of events:

1. User opens the login page.
2. User enters their id and password.
3. If the user enters correct information,
   3.1. Logging into the system is successful. (Log In Successful)
4. If the user enters the wrong password or id,
   4.1. “Wrong password or id” message will be displayed.

Entry condition(s): Opening the login page.

Exit condition(s): Successful login or failure in authentication process.

Use case name: Forgot Password

Participating Actor(s): User

Flow of events:

1. User clicks on the “Forgot password?” button.
2. User writes the e-mail address registered in the database.
3. A recovery mail with a link is sent to the address.
4. User changes their password using the link in the recovery mail.

Entry condition(s): Clicking on the “Forgot password?” button

Exit condition(s): Resetting password

Use case name: Change Password

Participating Actor(s): User

Flow of events:

1. User clicks on the “Change Password” button.
2. User enters their old password.
3. User enters their new password twice, one for verification.
4. User clicks on the button “Change”.

Entry condition(s): Clicking on the “Change Password” button

Exit condition(s): Changing password

<u>Report Package</u>

Use case name: Pre-Evaluate Report

Participating Actor(s): TA

Flow of events:

1. TA chooses a student.
2. TA downloads the report of the chosen student.
3. TA sends comments on the report OR
4. TA browses a pdf or doc file from the computer and uploads that file as feedback.

Entry condition(s): Opening the report to evaluate

Exit condition(s): Finishing evaluation

Use case name: View Report

Participating Actor(s): Evaluator, Student, TA

Flow of events:

1. User clicks on the “Report” button.
2. The report downloads to the computer of the user as a file to be viewed.

Entry condition(s): Clicking on the “Report” button

Exit condition(s): Viewing the report

Use case name: Evaluate Report

Participating Actor(s): Evaluator

Flow of events:

1. Evaluator opens the evaluation page of the student.
2. Evaluator evaluates part A.
3. Evaluator evaluates part B and gives feedback.
4. Evaluator evaluates part C.

Entry condition(s): Opening the evaluation page of the student

Exit condition(s): Being done with evaluation until it is satisfactory

---

Use case name: Evaluate Part A

Participating Actor(s): Evaluator

Flow of events:

1. Evaluator clicks on “Part A”.
2. Evaluator enters the average of the grades on the summer training evaluation form.
3. Evaluator makes a choice regarding whether the work done was related to computer engineering.
4. Evaluator makes a choice regarding whether the supervisor was a computer engineer or had similar background.
5. Evaluator clicks on the “Submit” button.

Entry condition(s): Clicking on “Part A”

Exit condition(s): Clicking on the “Submit” button

Use case name: Evaluate Part B

Participating Actor(s): Evaluator

Flow of events:

1. Evaluator clicks on “Part B”.
2. Evaluator choses whether the report is satisfactory or requires revision.
3. If revision is required,
    3.1. evaluator gives feedback and
    3.2. evaluator sets a date for resubmission.
4. Evaluator clicks on the “Submit” button.

Entry condition(s): Clicking on “Part B”

Exit condition(s): Report being satisfactory or unsatisfactory

Use case name: Evaluate Part C

Participating Actor(s): Evaluator

Flow of events:

1. Evaluator clicks on “Part C”.
2. Evaluator enters assessment/quality score of evaluation of the work - item(1).
3. Evaluator enters the sum of the assessment/quality score of evaluation of the work-item(2)-(7).
4. Evaluator enters assessment/quality score of the report.
5. Evaluator clicks on the “Submit” button.

Entry condition(s): Clicking on “Part C”

Exit condition(s): Clicking on the “Submit” button

Use case name: Submit Report

Participating Actor(s): Student

Flow of events:

1. Student opens the report submission page.
2. Student browses the report as a pdf or doc file.
3. Student will enter their name and surname.
4. Student will enter the name of the company that they have done their internship with.
5. Student will enter their supervisor's id.
6. Student will choose the course code.
7. Student clicks on the "Submit" button.
8. Student may update their report by submitting a new report. (Update Report)

Entry condition(s): Opening report submission page

Exit condition(s): Submitting report

Use case name: Enter Supervisor Information

Participating Actor(s): Student

Flow of events:

1. Student clicks on the "Enter Supervisor Information" button.
2. Student enters the name and the surname of the supervisor.
3. Student enters the company mail address to the system.

Entry condition(s): Clicking on the "Enter Supervisor Information" button

Exit condition(s): Completion of entering information

Use case name: Submit Training Evaluation Form

Participating Actor(s): Supervisor

Flow of events:

1. Supervisor opens the page where submission will be done.
2. Supervisor browses the form from the computer.
3. Supervisor clicks on the "Submit" button.

Entry condition(s): Opening the submission page

Exit condition(s): Clicking on the “Submit” button

System Initialization Package

Use case name: Assign Students to Evaluators

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the “Student Assignment - Evaluator” page.
2. User chooses the evaluator name.
3. User distributes students alphabetically.
4. User adds students to the student student list of the evaluator or user deletes students from the student student list of the evaluator.

Entry condition(s): Opening the “Student Assignment - Evaluator” page

Exit condition(s): Assigning students to evaluators

Use case name: Import Information

Participating Actor(s): Super Admin

Flow of events:

1. Super admin opens the management page.
2. Super admin chooses a title which is evaluator, TA, or student.
3. Super admin goes to the management page of the chosen title.
4. Super admin uploads an excel document.
5. Super admin clicks on the “Import” button.

Entry condition(s): Opening management page

Exit condition(s): Importing information

Use case name: Export Information

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the management page.
2. User chooses a title which is evaluator, TA, or student.
3. User goes to the management page of the chosen title.
4. Super admin clicks on the “Export ” button.

Entry condition(s): Opening management page

Exit condition(s): Exporting information

Use case name: Initialize Website

Participating Actor(s): Super Admin

Flow of events:

1. Super admin opens the “System” page.
2. Super admin enters the semester information.
3. Super admin adjusts system preferences.
4. Super admin initializes the system by clicking the “Initialize” button.
5. Super admin may choose to terminate or pause the system later.

Entry condition(s): Opening the “System” page.

Exit condition(s): Initializing the system

Use case name: Enroll Evaluators

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the “Evaluator Management” page.
2. If the user wants to add new evaluator,
    - 2.1. user enters the name, surname, and id of the evaluator and
    - 2.2. user enters the quota indicating the number of the students that evaluator will have and
    - 2.3. user clicks on the “Submit” button.
3. If the user wants to edit information,

    3.1. user changes the name or surname of the evaluator or
    3.2. user changes the quota and
    3.3. user clicks on the “Edit” button.
4. If the user wants to delete an evaluator,
    4.1. user clicks on the “Delete” button.

Entry condition(s): Opening the “Evaluator Management” page

Exit condition(s): Adding new evaluator or editing information or deleting evaluator

---

Use case name: Enroll TA

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the “TA Management” page.
2. If the user wants to add new TA,
    2.1. user enters the name, surname, and id of the TA and
    2.2. user enters the quota indicating the number of the students that TA will have and
    2.3. user clicks on the “Submit” button.
3. If the user wants to edit information,
    3.1. user changes the name or surname of the TA or
    3.2. user changes the quota and
    3.3. user clicks on the “Edit” button.
4. If the user wants to delete a TA,
    4.1. user clicks on the “Delete” button.

Entry condition(s): Opening the “TA Management” page

Exit condition(s): Adding new TA or editing quota or deleting TA

Use case name: Enroll Students

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the “Student Management” page.
2. If the user wants to add new student,
    2.1. user enters the name, surname, and id of the student and
    2.2. user clicks on the “Submit” button.
3. If the user wants to edit information,
    3.1. user changes the name or surname of the student and
    3.2. user clicks on the “Edit” button.
4. If the user wants to delete a student,
    4.1. user clicks on the “Delete” button.

Entry condition(s): Opening the “Student Management” page

Exit condition(s): Adding new student or editing quota or deleting student

---

Use case name: Enroll Admins

Participating Actor(s): Administrator, Super Admin

Flow of events:

1. User opens the “Admin Management” page.
2. If the user wants to add new admin,
    2.1. user enters the name, surname, and id of the admin and
    2.2. user clicks on the “Submit” button.
3. If the user wants to edit information,
    3.1. user changes the name or surname of the admin and
    3.2. user clicks on the “Edit” button.
4. If the user wants to delete an admin,
    4.1. user clicks on the “Delete” button.

Entry condition(s): Opening the “Admin Management” page

Exit condition(s): Adding new admin or editing quota or deleting admin

---

View Package

Use case name: View list of all students

Participating Actor(s): Administrator, Evaluator, TA

Flow of events:

1. User opens the page where students are displayed as a list.
2. User sees the report status of the students. (View Students' Statuses)
3. When user clicks on a student from the list,
    3.1. User views the profile of the student. (View Student's Profile)

Entry condition(s): Opening the page where students are displayed as a list

Exit condition(s): Viewing list of all students

Use case name: View Their Own Student List

Participating Actor(s): Evaluator

Flow of events:

1. Evaluator opens the page where their own students are displayed as a list.
2. User sees the report status of the student. (View Students' Statuses)
3. When user clicks on a student from the list,
    3.1. User views the profile of the student. (View Student's Profile)

Entry condition(s): Opening the page where their own students are displayed as a list

Exit condition(s): Viewing list of their own students

Use case name: Search Student

Participating Actor(s): Administrator, Evaluator, TA

Flow of events:

1. User searches the student by name, surname or id.
2. User clicks on the found student.
3. User views the profile of the student. (View Student’s Profile)

Entry condition(s): Searching a student from the search

Exit condition(s): Finding the searched student

Use case name: View Student’s Profile

Participating Actor(s): Administrator, Evaluator, Student, TA

Flow of events:

1. User opens the profile.
2. User views the submitted report’s lists. (View Student’s Report List)
3. User views the personal information of the student like name, surname, or id. (View Student’s Information)
4. User views the status of the student. (View Status)

Entry condition(s): Opening the profile

Exit condition(s): Being done with the student’s profile

Use case name: View TA(s) as a list

Participating Actor(s): Administrator, Evaluator, TA

Flow of events:

1. User opens the page where TA(s) is displayed as a list.
2. User sees the profile of the chosen TA. (View TA profile)
3. User sees the list of students of the TA(s). (View Their Student List)

Entry condition(s): Opening the page where TA(s) is displayed as a list

Exit condition(s): Viewing TA(s) as a list

Use case name: View Evaluators as a list

Participating Actor(s): Administrator, Evaluator, TA

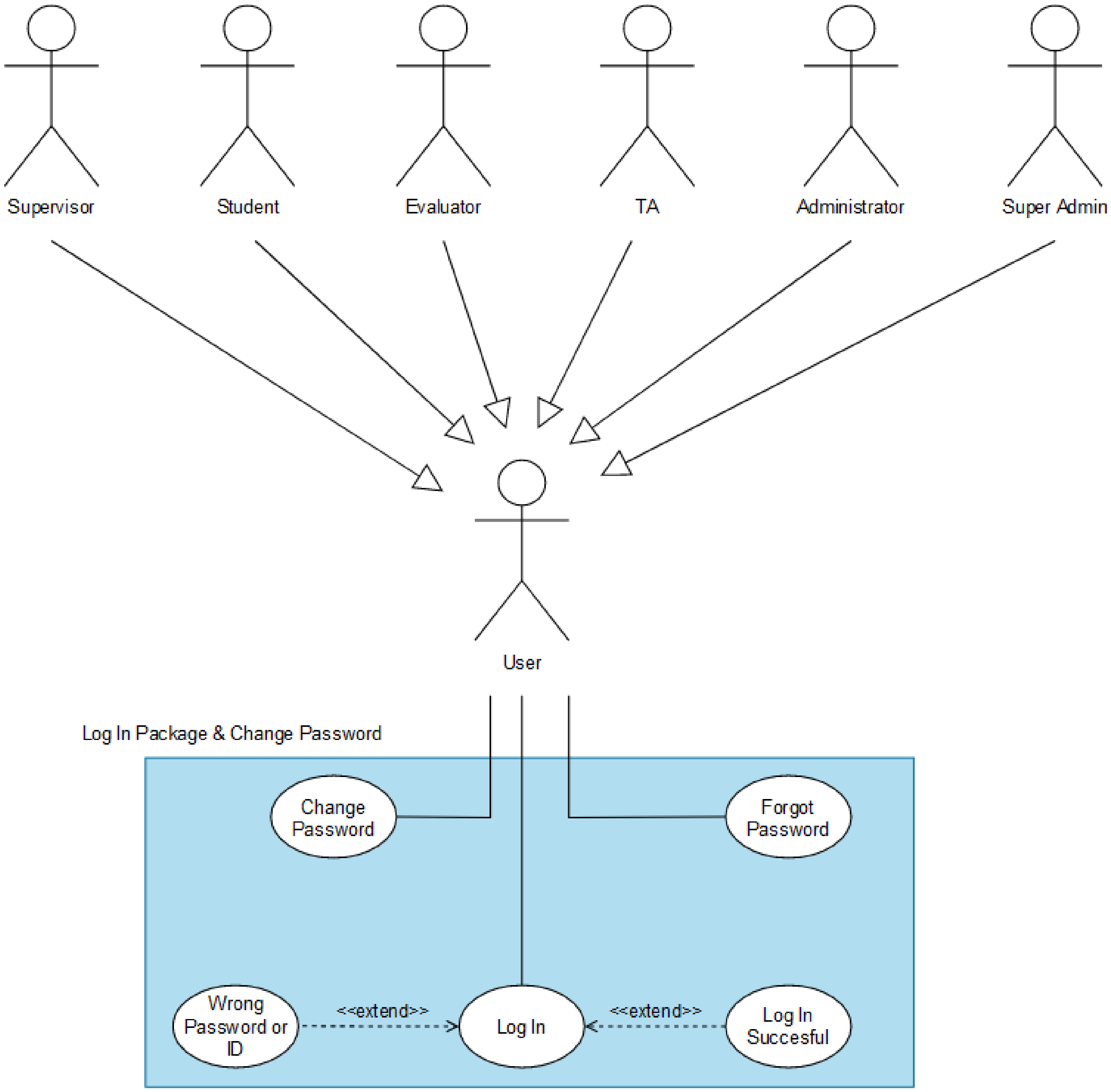
Flow of events:

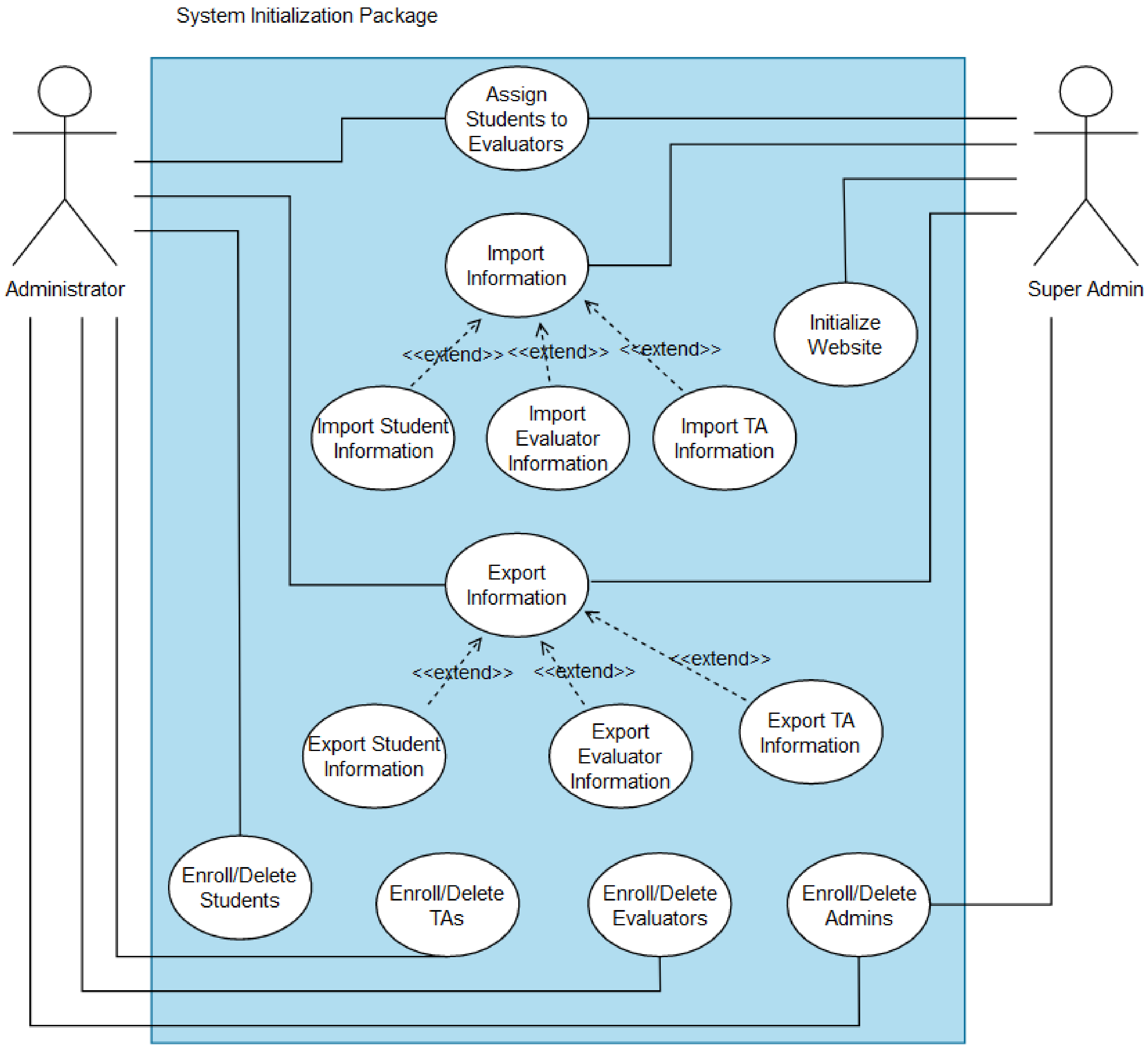
1. User opens the page where evaluators are displayed as a list.
2. User sees the profile of the chosen evaluator. (View Evaluator profile)
3. User sees the list of students of the evaluator. (View Their Student List)

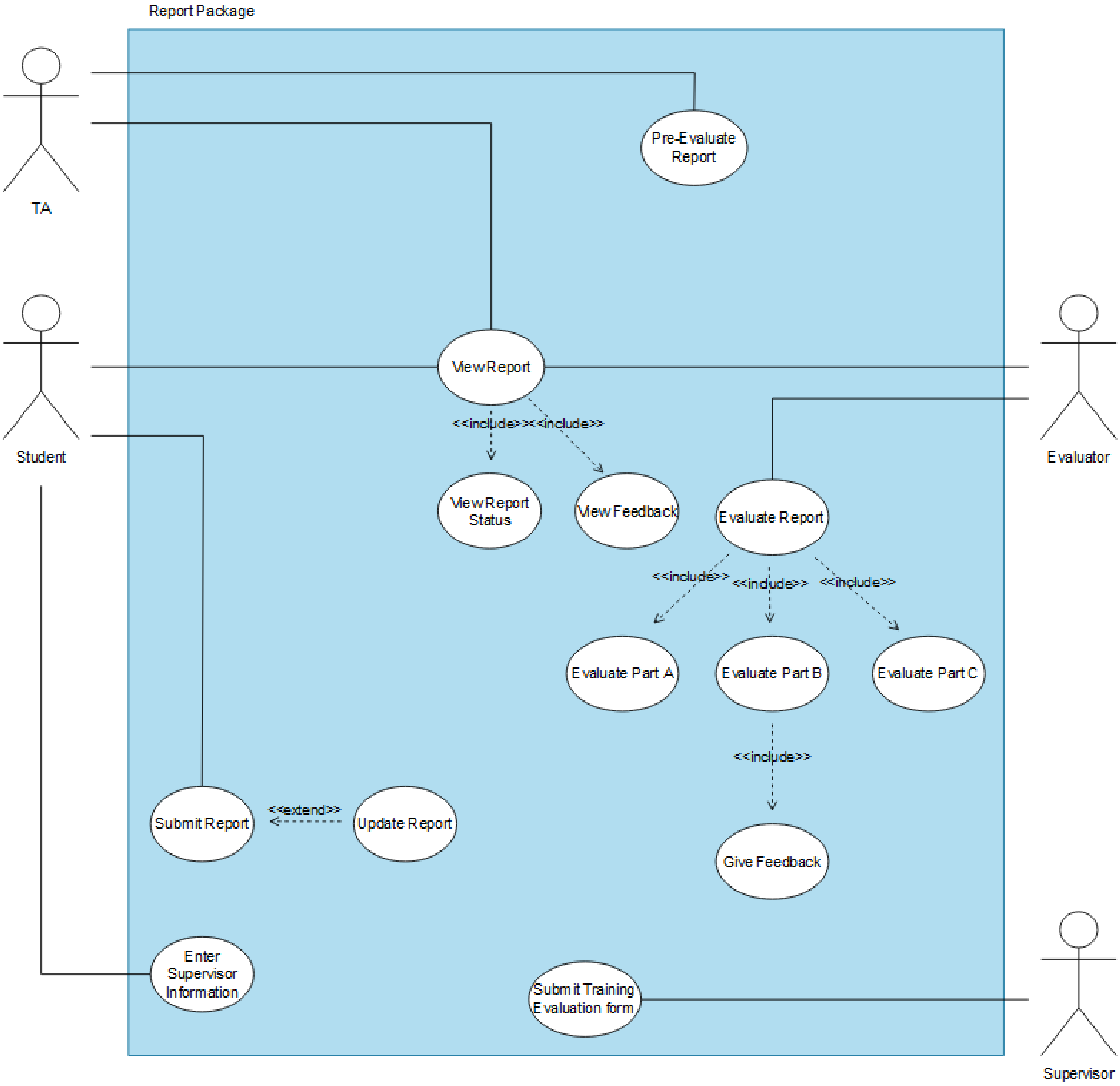
Entry condition(s): Opening the page where evaluators are displayed as a list

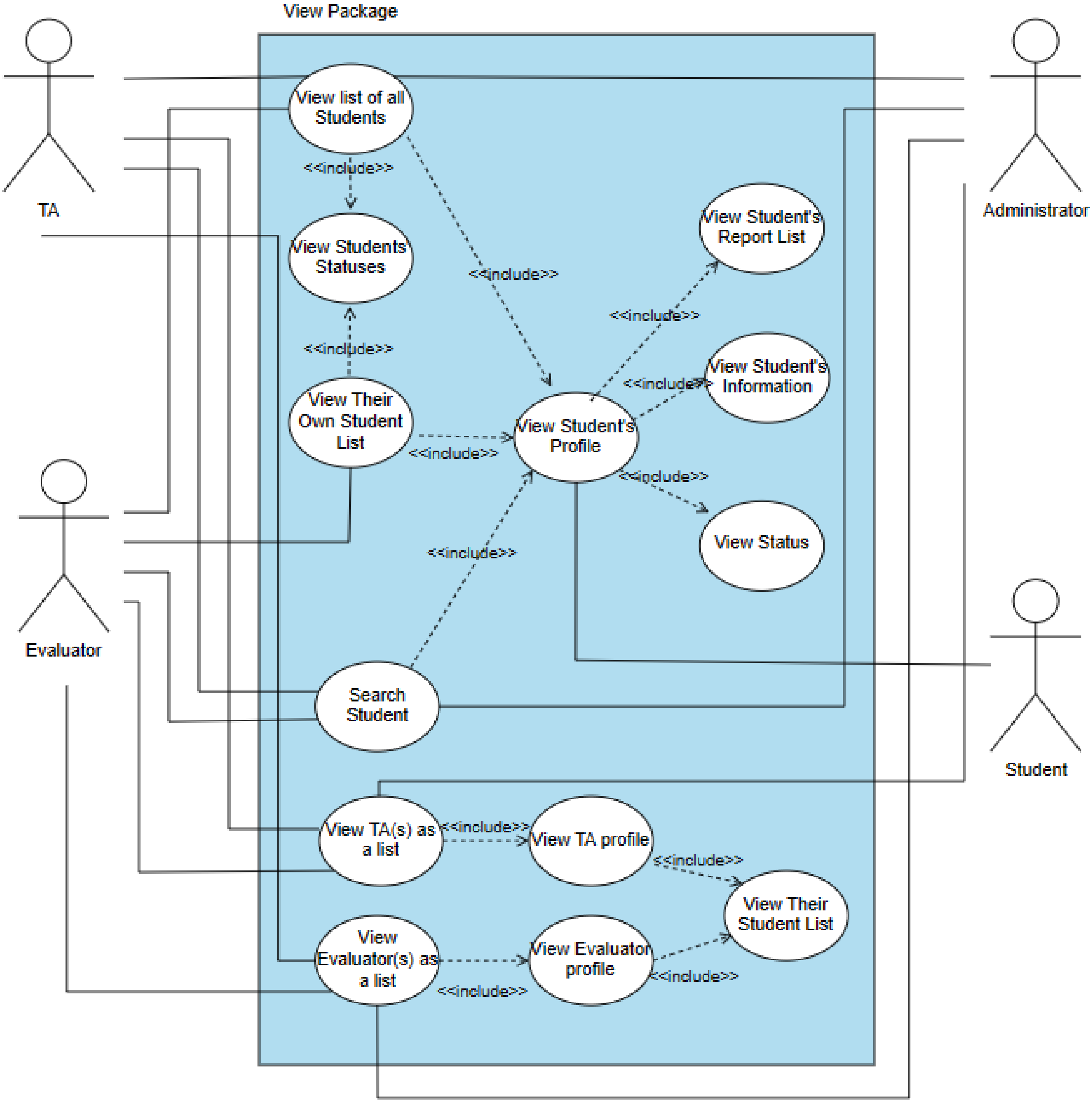
Exit condition(s): Viewing evaluators as a list

3.3.2 Use-Case Models

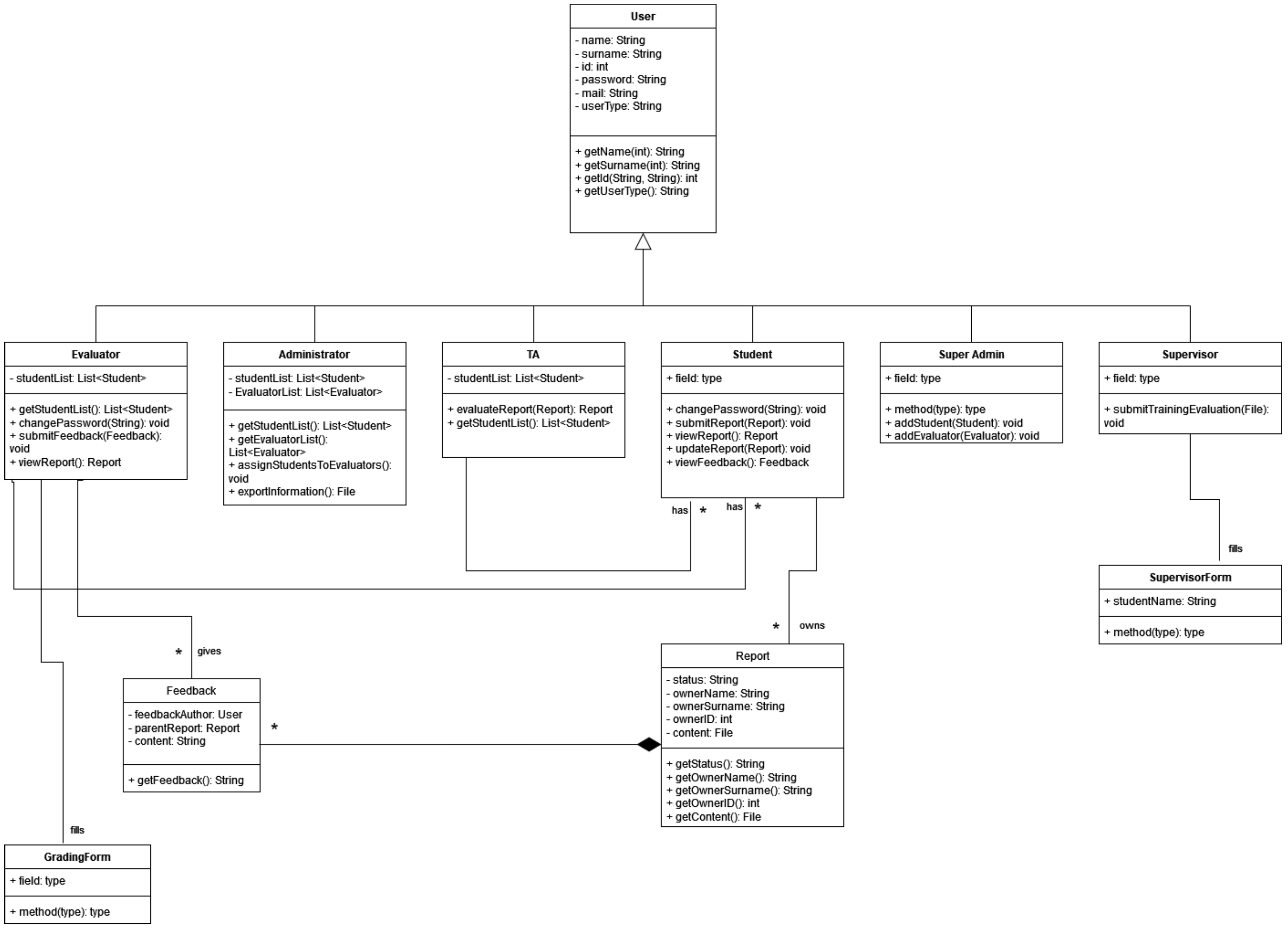



System Initialization Package
Administrator
Super Admin
Assign Students to Evaluators
Import Information
Initialize Website
<<extend>>
<<extend>>
<<extend>>
Import Student Information
Import Evaluator Information
Import TA Information
Export Information
<<extend>>
<<extend>>
<<extend>>
Export Student Information
Export Evaluator Information
Export TA Information
Enroll/Delete Students
Enroll/Delete TAs
Enroll/Delete Evaluators
Enroll/Delete Admins


Report Package
TA
Pre-Evaluate Report
Student
View Report
Evaluator
<<include>>
<<include>>
View Report Status
View Feedback
Evaluate Report
<<include>>
<<include>>
<<include>>
Evaluate Part A
Evaluate Part B
Evaluate Part C
<<include>>
Submit Report
<<extend>>
Update Report
Give Feedback
Supervisor
Enter Supervisor Information
Submit Training Evaluation form

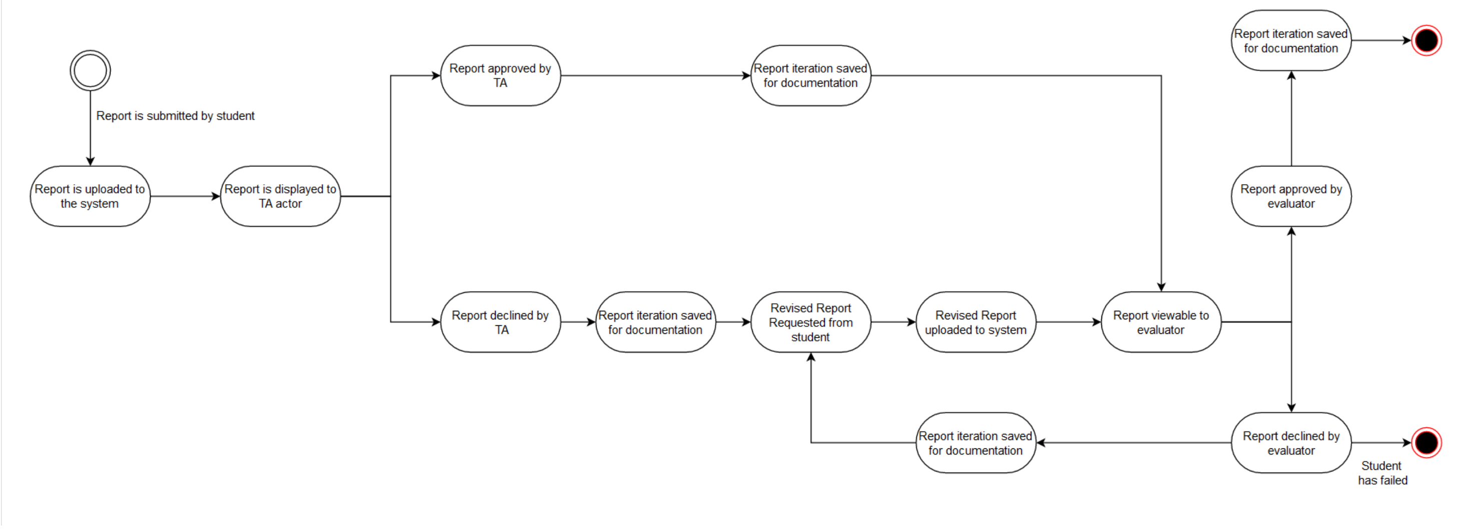

View Package
TA
Administrator
Evaluator
Student
View list of all Students
<<include>>
View Students Statuses
<<include>>
View Their Own Student List
<<include>>
View Student's Report List
<<include>>
View Student's Information
<<include>>
View Student's Profile
<<include>>
View Status
<<include>>
<<include>>
Search Student
View TA(s) as a list
<<include>>
View TA profile
<<include>>
View Their Student List
View Evaluator(s) as a list
<<include>>
View Evaluator profile
<<include>>

3.3.3 Object and Class Model

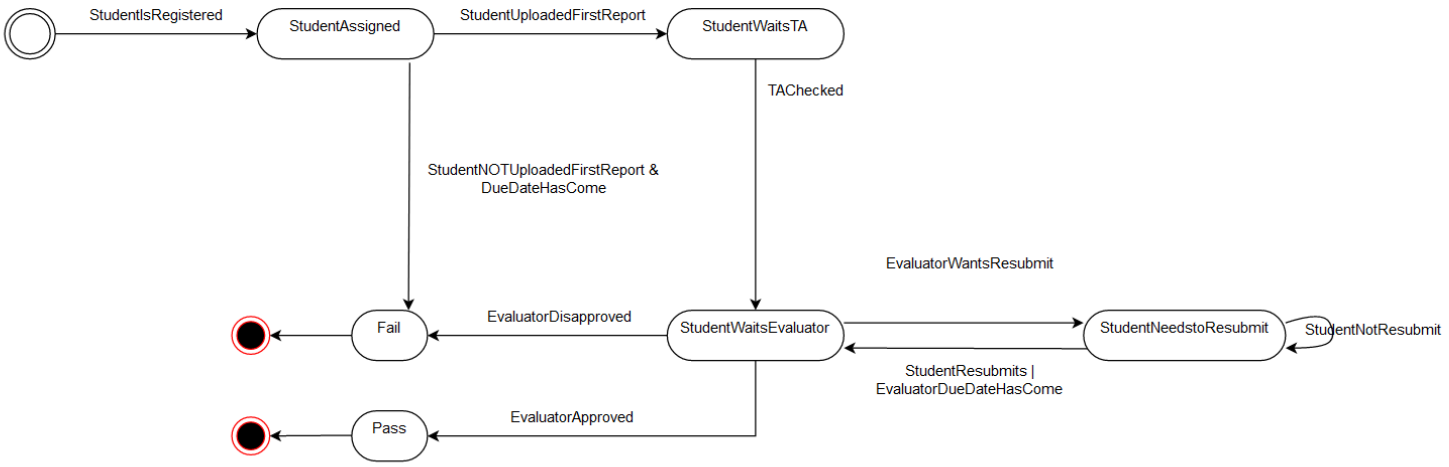

User
- name: String
- surname: String
- id: int
- password: String
- mail: String
- userType: String
+ getName(int): String
+ getSurname(int): String
+ getId(String, String): int
+ getUserType(): String
Evaluator
- studentList: List<Student>
+ getStudentList(): List<Student>
+ changePassword(String): void
+ submitFeedback(Feedback): void
+ viewReport(): Report
Administrator
- studentList: List<Student>
- EvaluatorList: List<Evaluator>
+ getStudentList(): List<Student>
+ getEvaluatorList(): List<Evaluator>
+ assignStudentsToEvaluators(): void
+ exportInformation(): File
TA
- studentList: List<Student>
+ evaluateReport(Report): Report
+ getStudentList(): List<Student>
Student
+ field: type
+ changePassword(String): void
+ submitReport(Report): void
+ viewReport(): Report
+ updateReport(Report): void
+ viewFeedback(): Feedback
Super Admin
+ field: type
+ method(type): type
+ addStudent(Student): void
+ addEvaluator(Evaluator): void
Supervisor
+ field: type
+ submitTrainingEvaluation(File): void
has
*
has
*
fills
SupervisorForm
+ studentName: String
+ method(type): type
*
owns
*
gives
Feedback
- feedbackAuthor: User
- parentReport: Report
- content: String
+ getFeedback(): String
*
Report
- status: String
- ownerName: String
- ownerSurname: String
- ownerID: int
- content: File
+ getStatus(): String
+ getOwnerName(): String
+ getOwnerSurname(): String
+ getOwnerID(): int
+ getContent(): File
fills
GradingForm
+ field: type
+ method(type): type

3.3.4 State Diagrams

REPORT STATE DIAGRAM

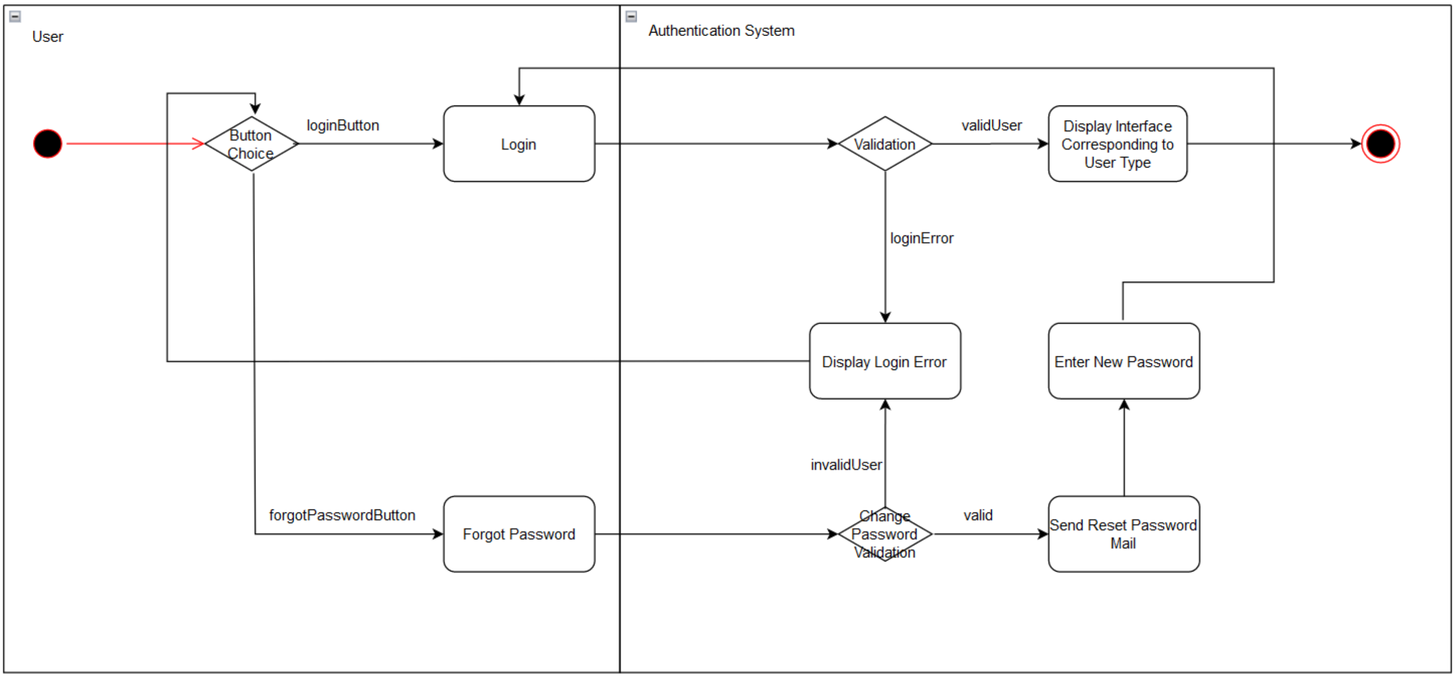

Report is submitted by student
Report is uploaded to the system
Report is displayed to TA actor
Report approved by TA
Report iteration saved for documentation
Report declined by TA
Report iteration saved for documentation
Revised Report Requested from student
Revised Report uploaded to system
Report viewable to evaluator
Report iteration saved for documentation
Report approved by evaluator
Report declined by evaluator
Report iteration saved for documentation
Student has failed

STUDENT STATE DIAGRAM

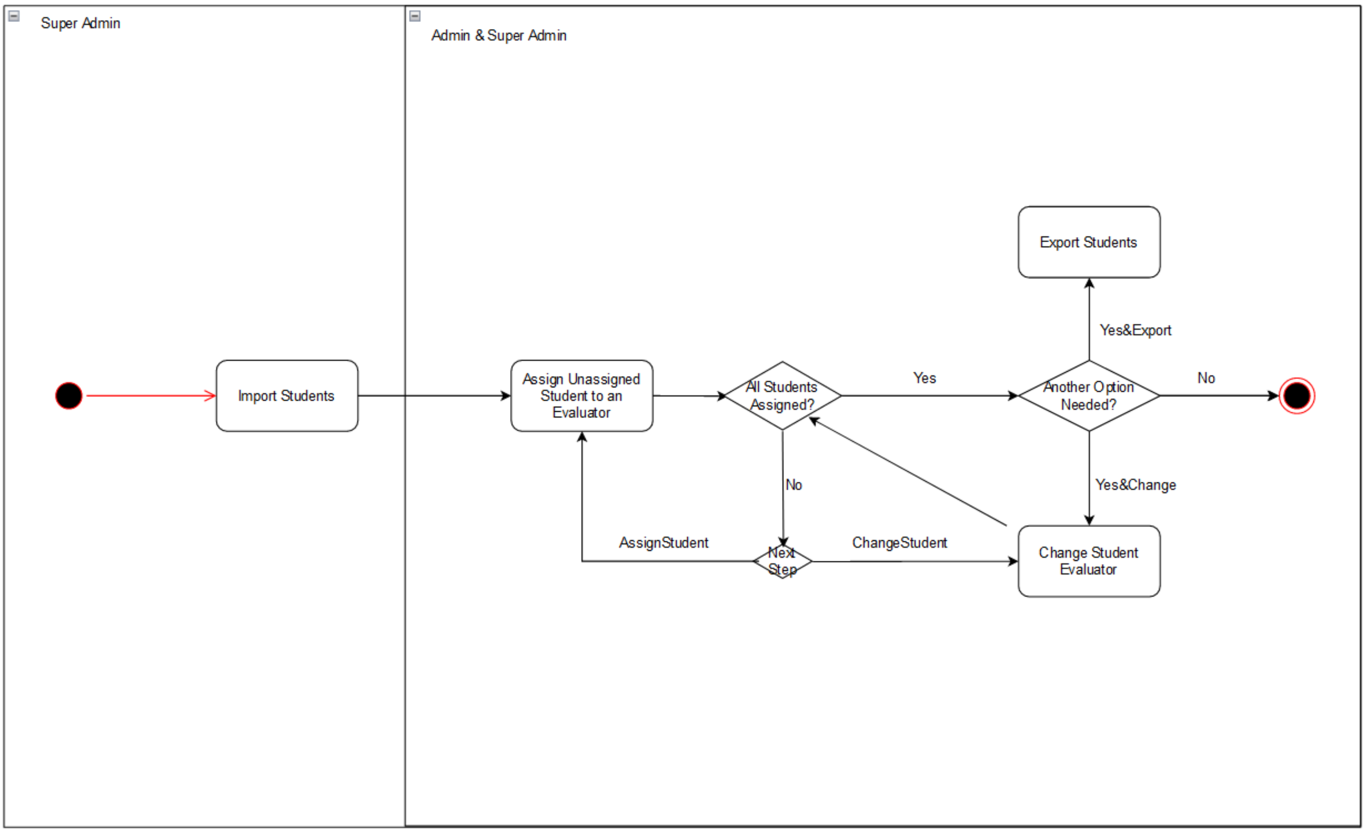


3.3.5 Activity Diagrams

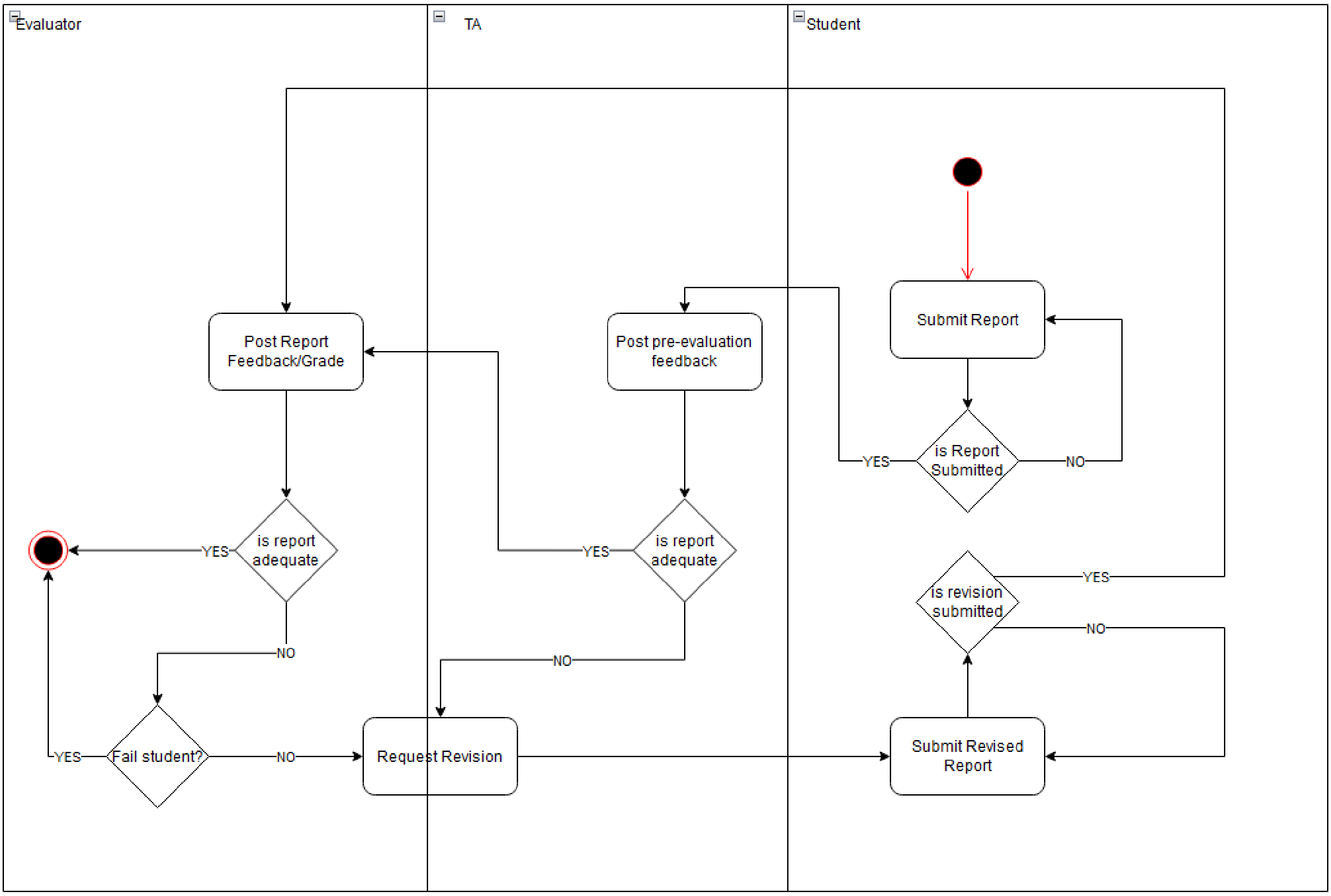
Log in



Initializing System (Enrolling all the students)

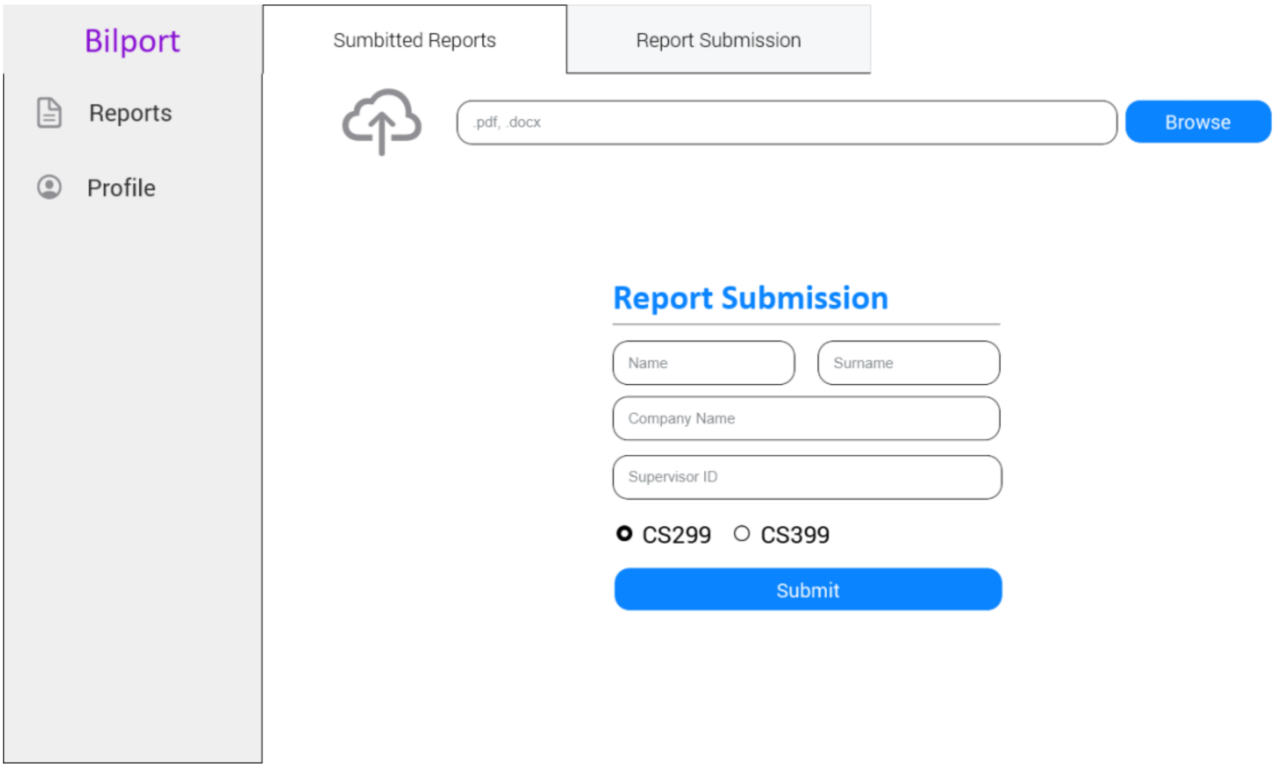


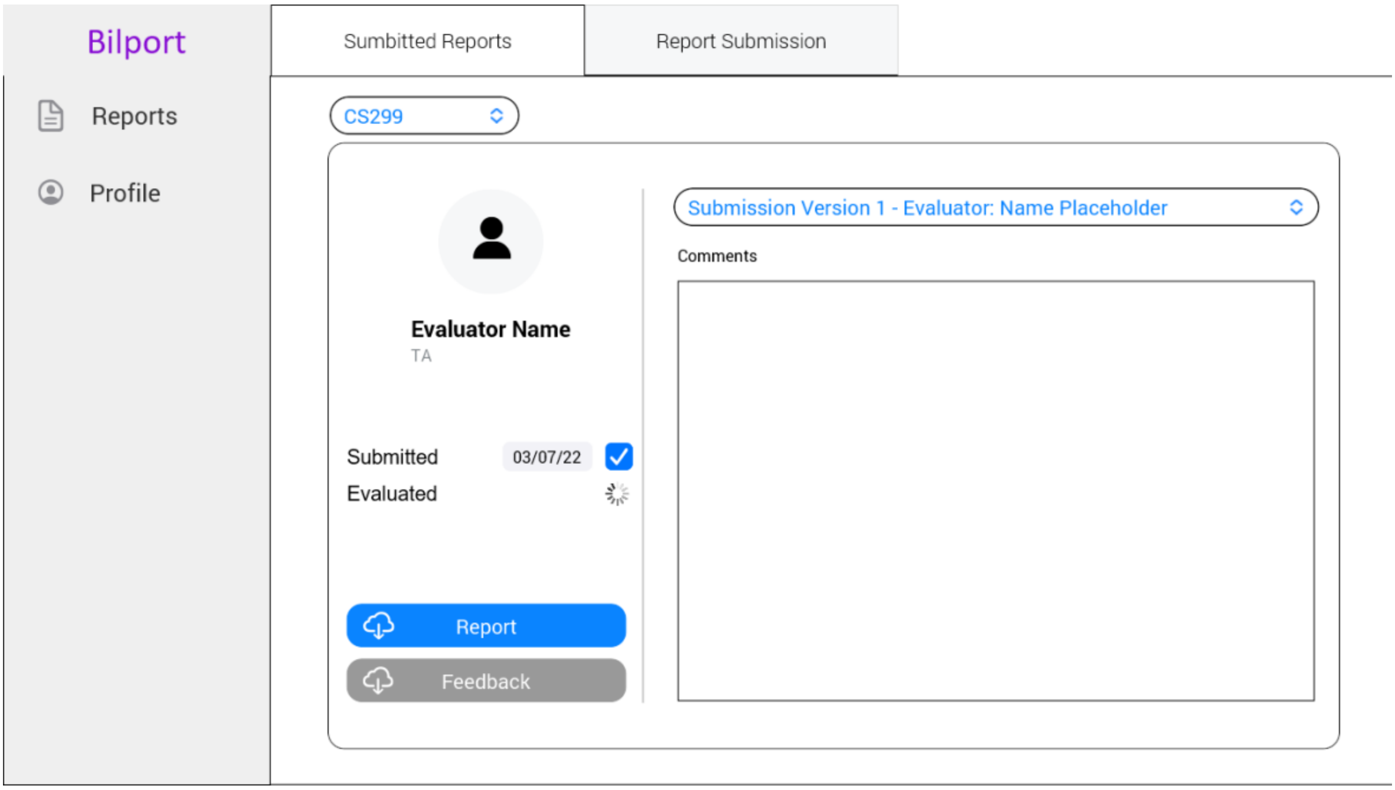
Report Evaluation



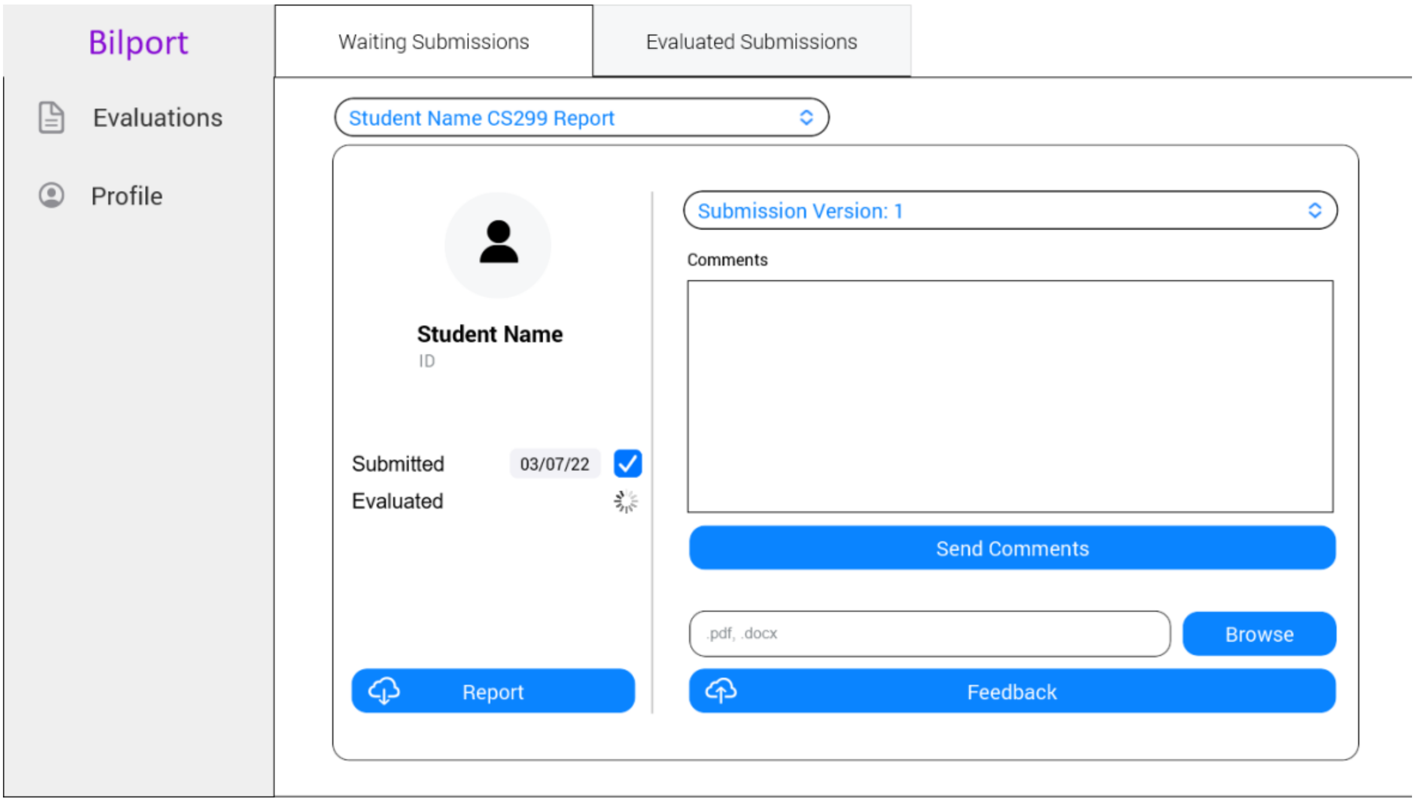
3.3.6 User Interface

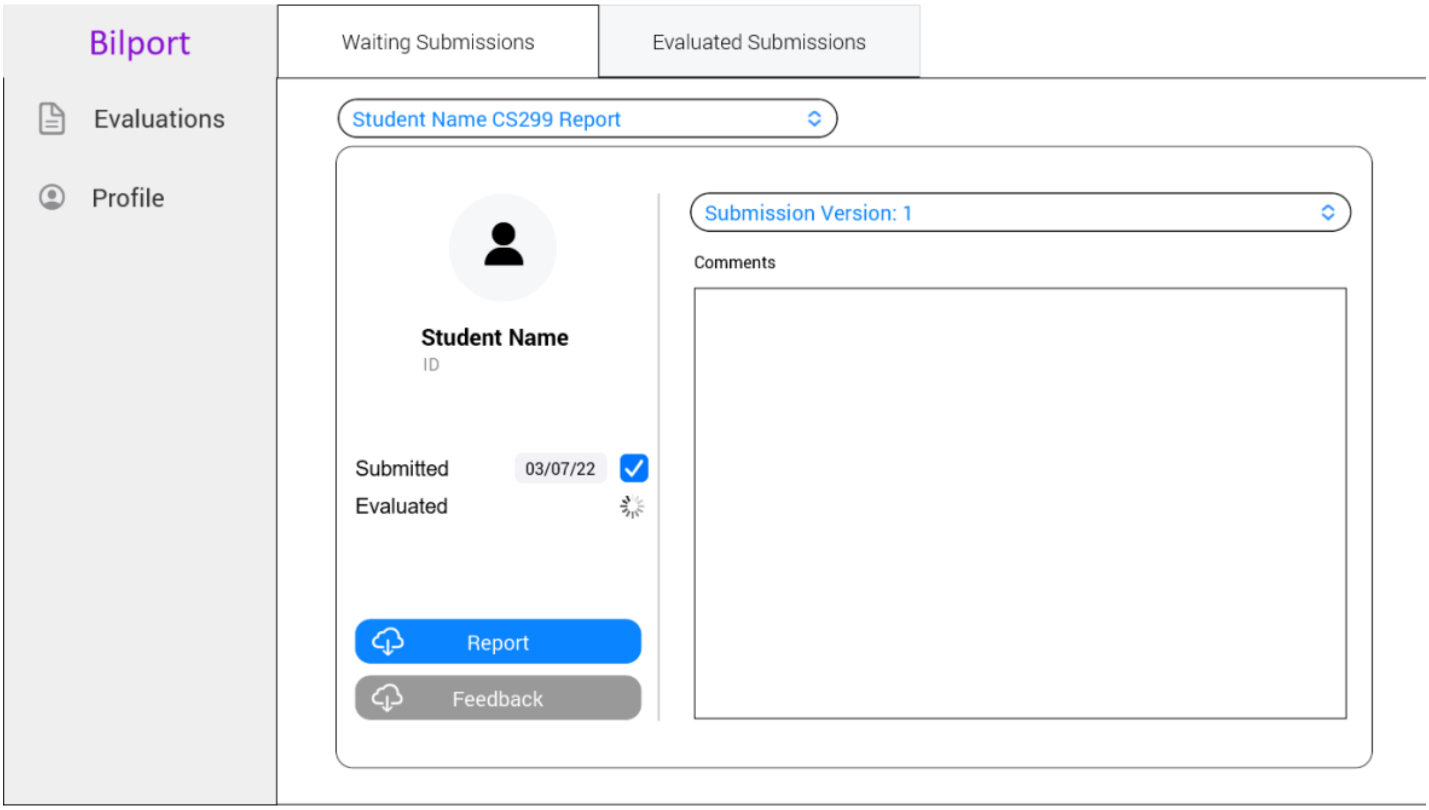
Student:



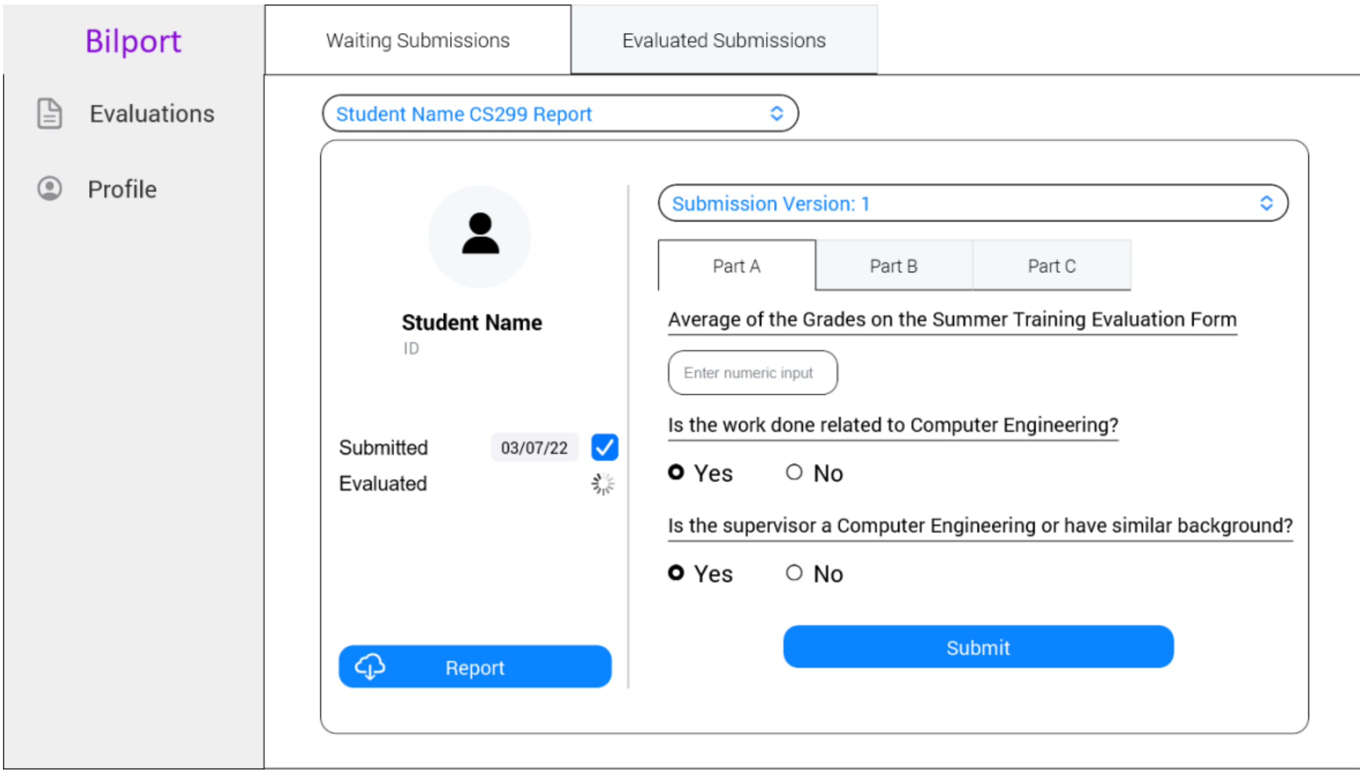


TA:


Bilport
Waiting Submissions
Evaluated Submissions
Evaluations
Profile
Student Name CS299 Report
Student Name
ID
Submitted
03/07/22
Evaluated
Report
Submission Version: 1
Comments
Send Comments
.pdf, .docx
Browse
Feedback


Bilport
Waiting Submissions
Evaluated Submissions
Evaluations
Profile
Student Name CS299 Report
Student Name
ID
Submitted
03/07/22
Evaluated
Report
Feedback
Submission Version: 1
Comments

Evaluator:



Submitted 03/07/22 ☑

Evaluated

Average of the Grades on the Summer Training Evaluation Form

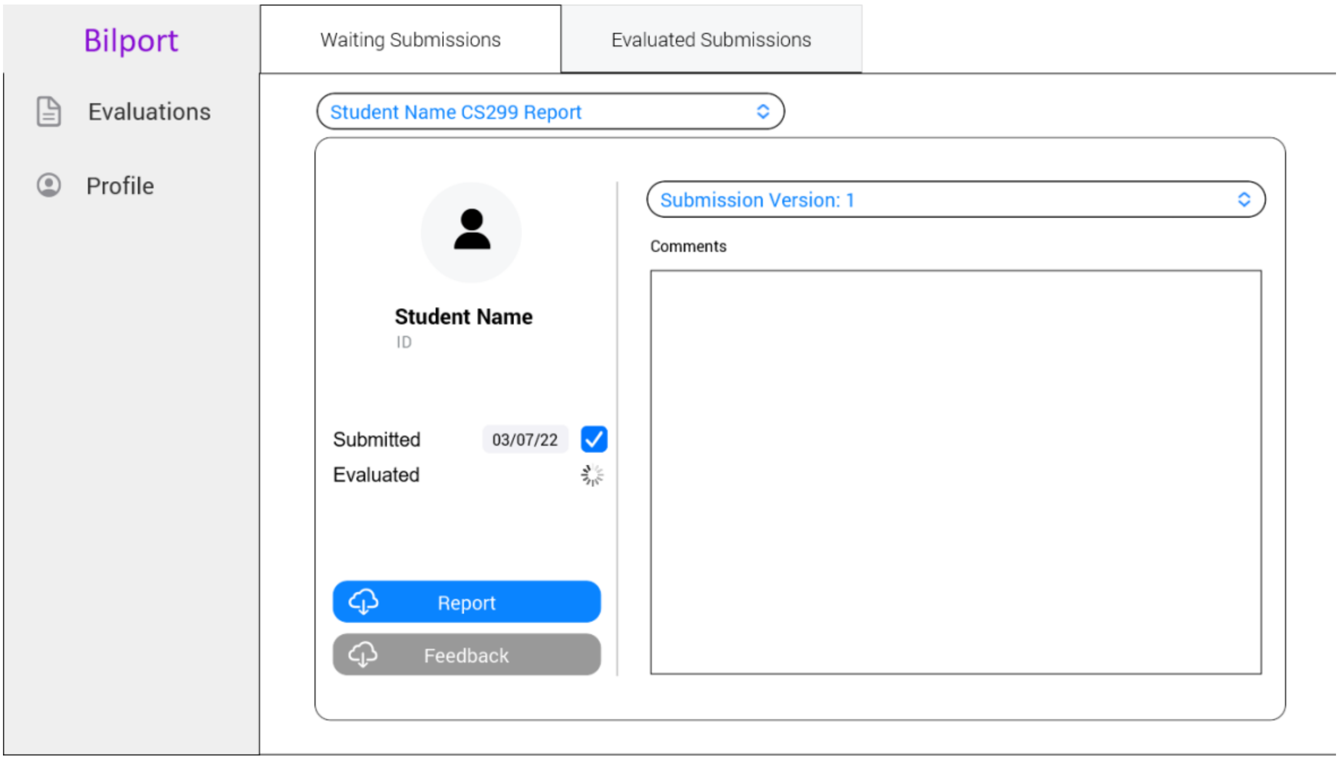
Enter numeric input

Is the work done related to Computer Engineering?

● Yes ○ No

Is the supervisor a Computer Engineering or have similar background?

● Yes ○ No

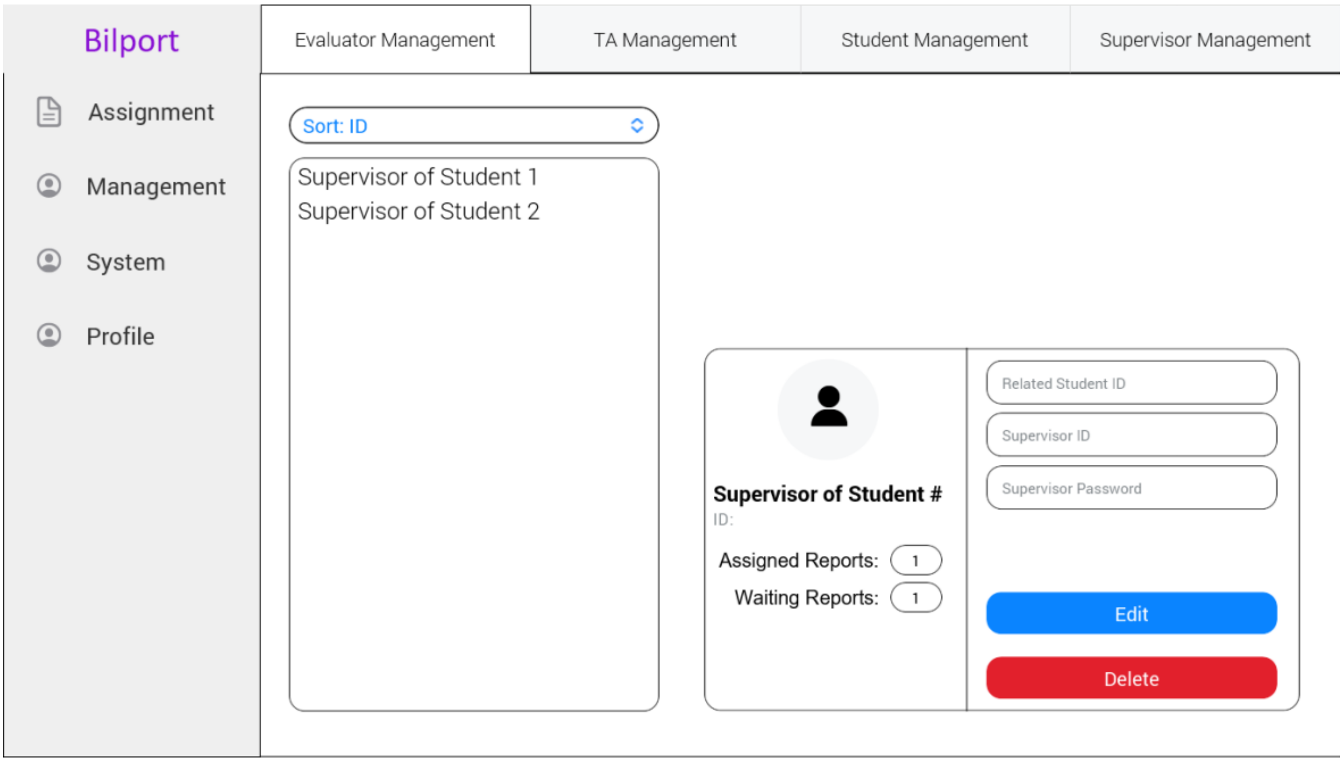


Submitted 03/07/22 ☑

Evaluated

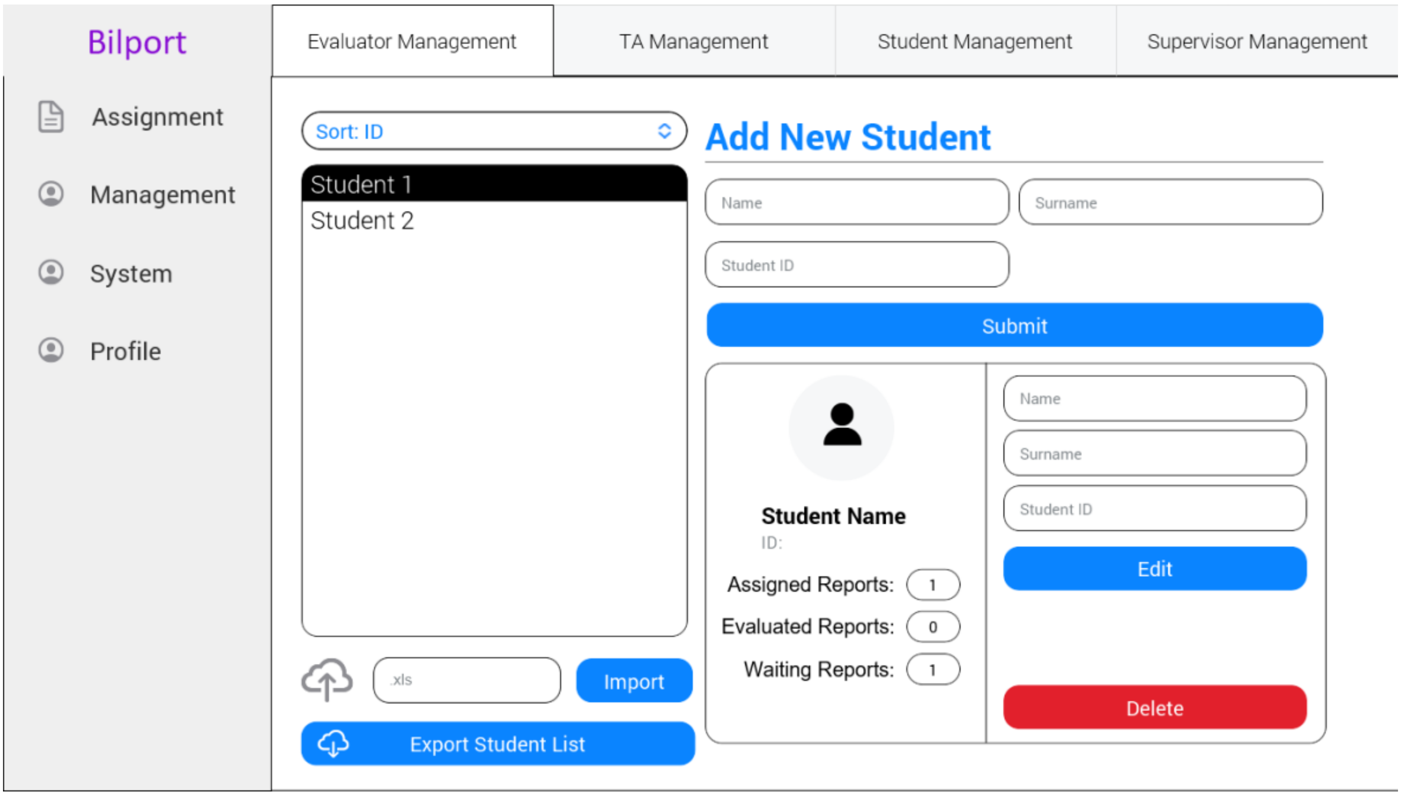
Comments

Super Admin:



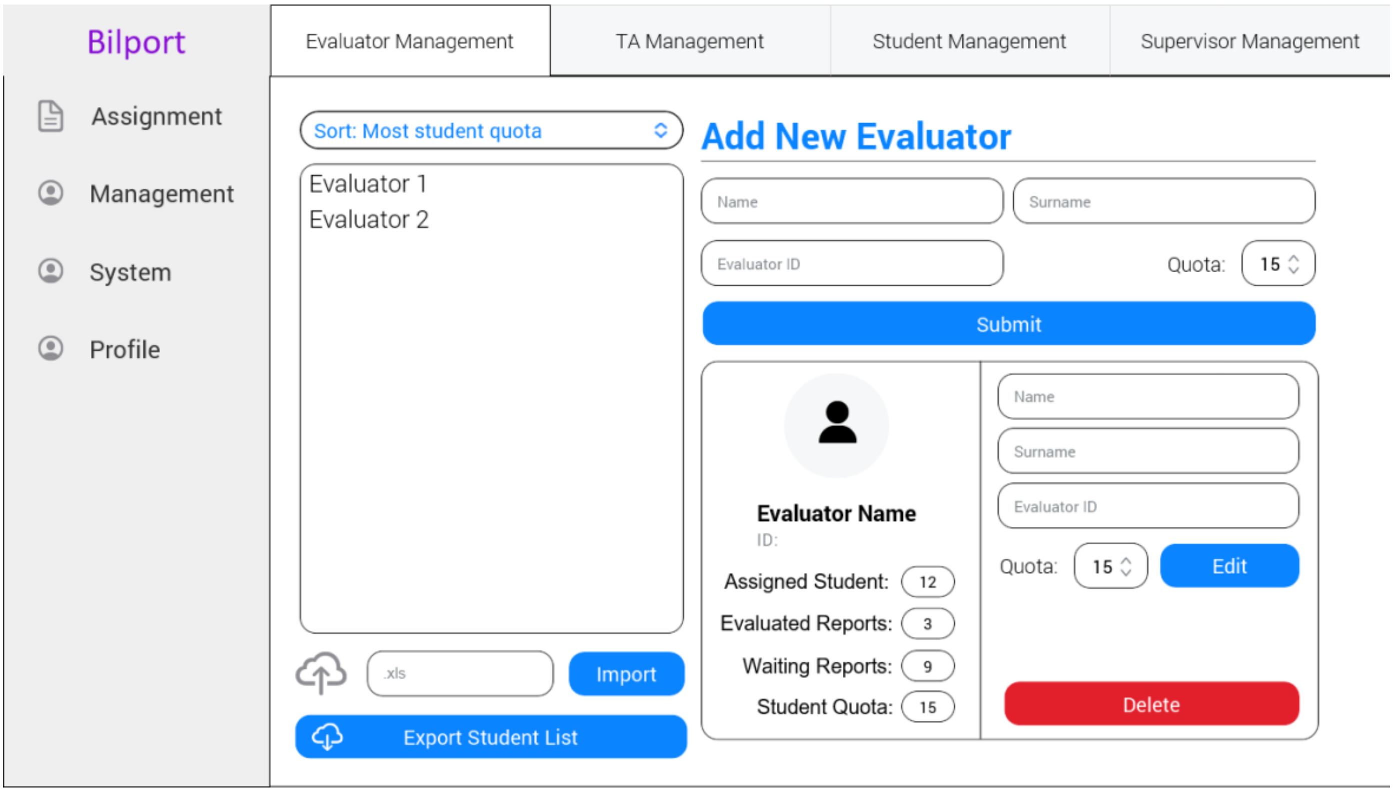
Supervisor of Student 1
Supervisor of Student 2

Supervisor of Student #
ID:
Assigned Reports: 1
Waiting Reports: 1



Student 1
Student 2

Student Name
ID:
Assigned Reports: 1
Evaluated Reports: 0
Waiting Reports: 1



Evaluator 1
Evaluator 2

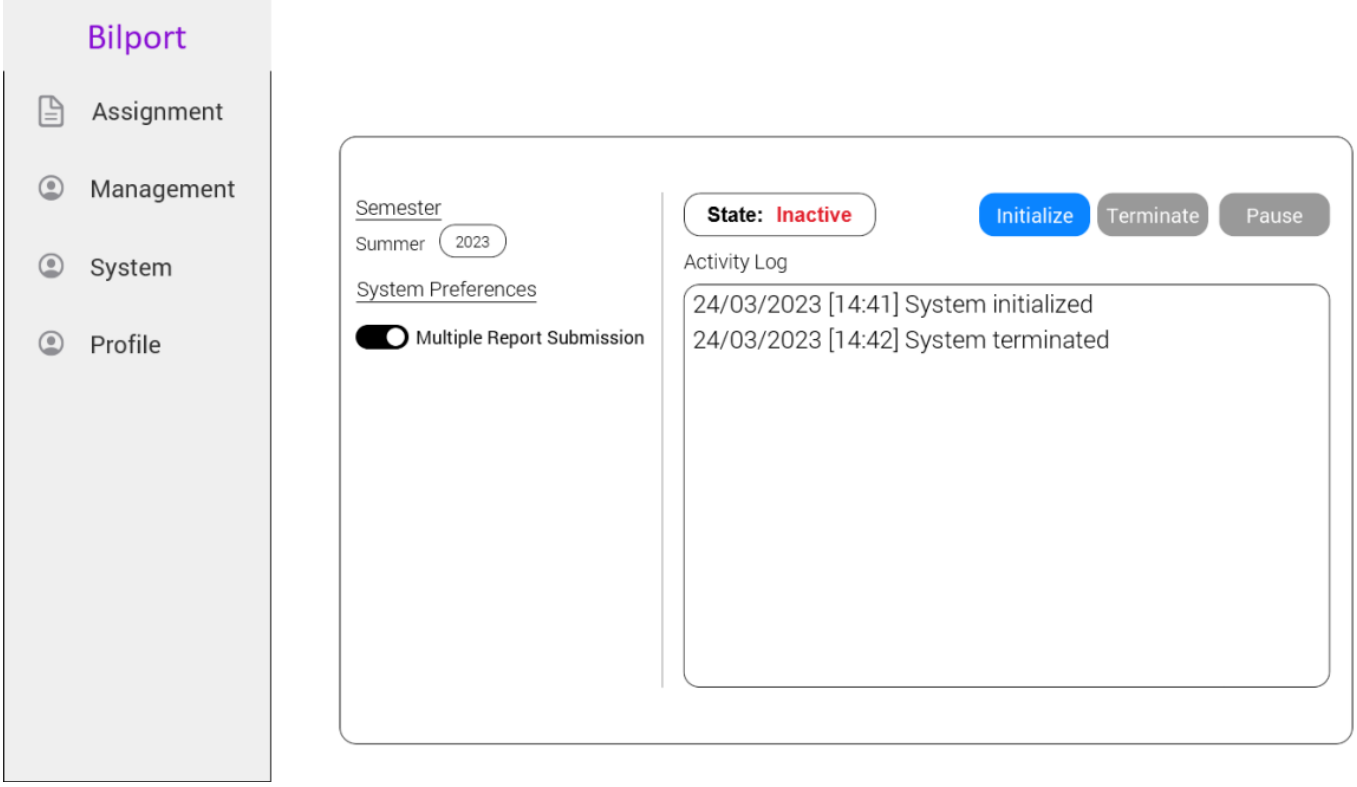
.xls

Quota: 15

Evaluator Name
ID:

Assigned Student:	12
Evaluated Reports:	3
Waiting Reports:	9
Student Quota:	15

Name
Surname
Evaluator ID
Quota: 15



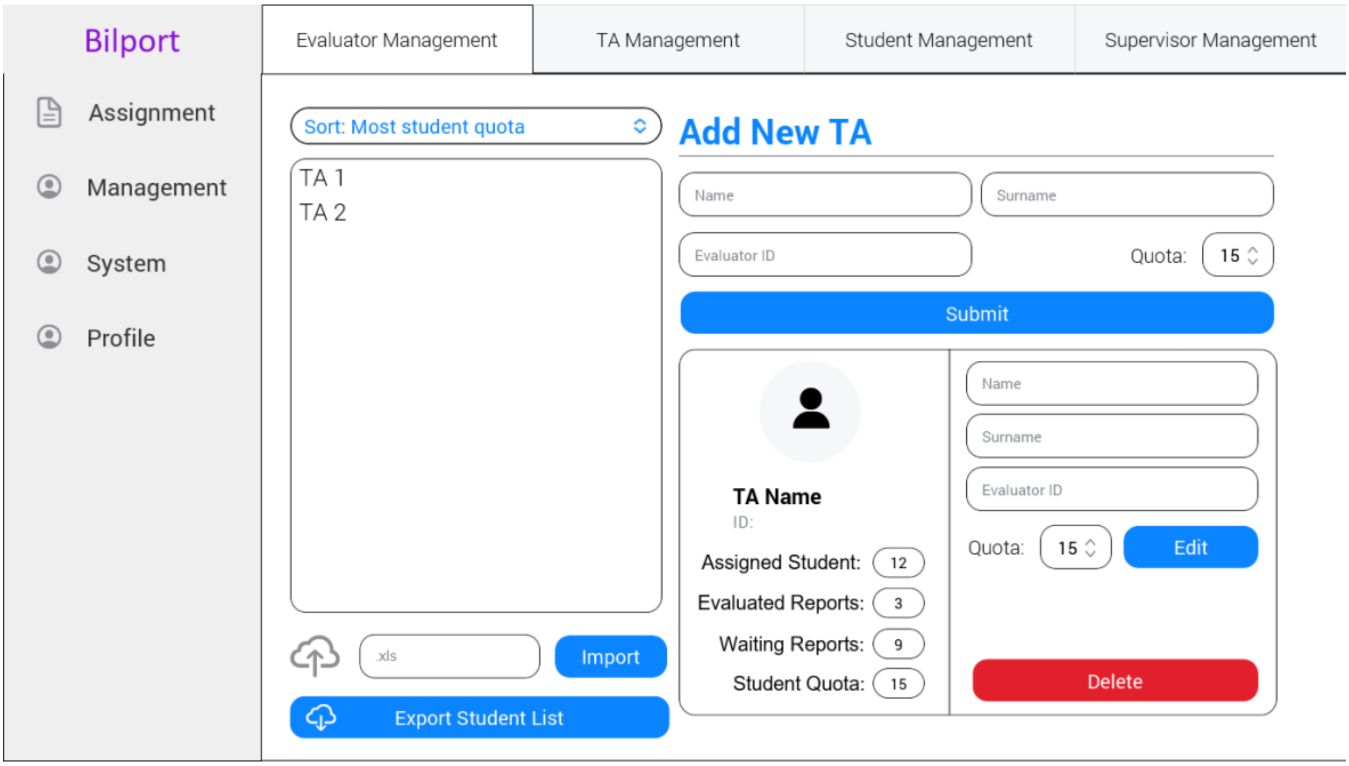
Semester
Summer 2023

System Preferences
Multiple Report Submission

State: Inactive

Activity Log

```
24/03/2023 [14:41] System initialized
24/03/2023 [14:42] System terminated
```



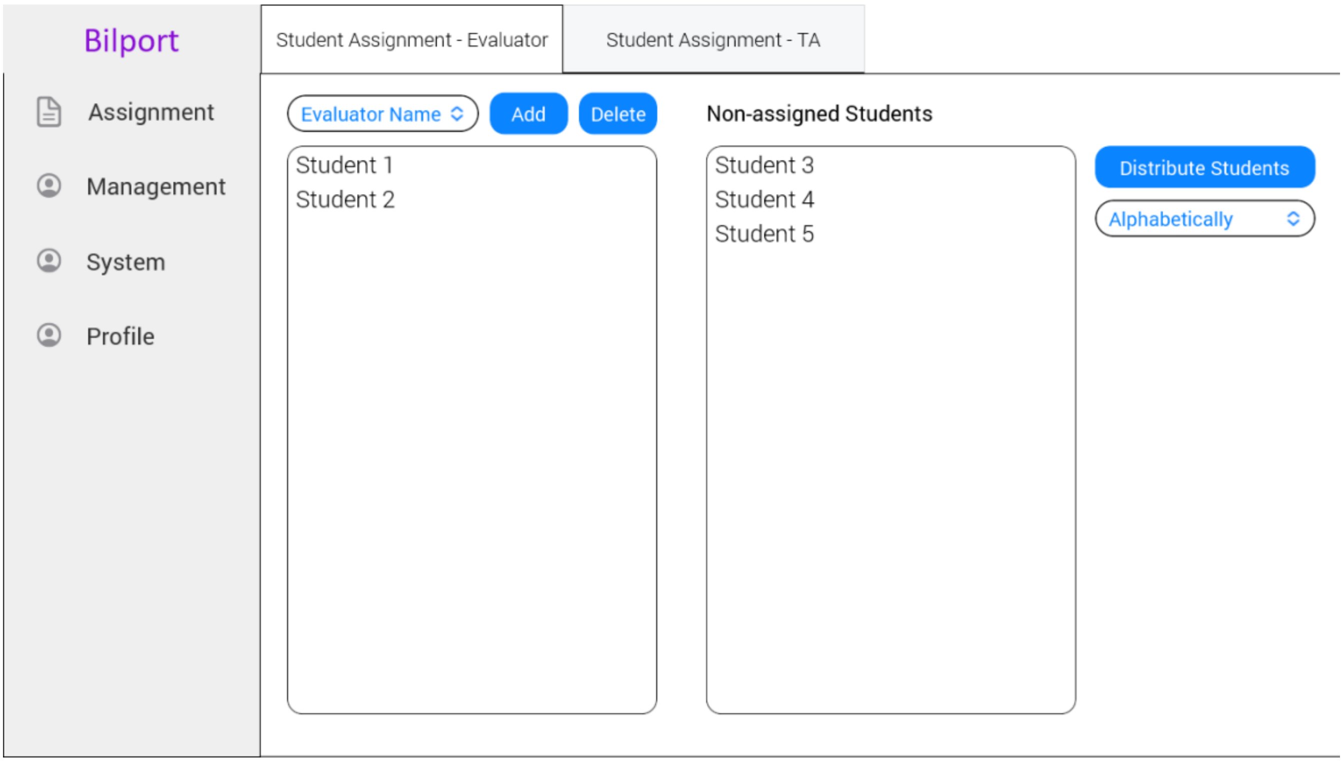
Sort: Most student quota

TA 1
TA 2

Quota: 15

Assigned Student: 12
Evaluated Reports: 3
Waiting Reports: 9
Student Quota: 15

Quota: 15

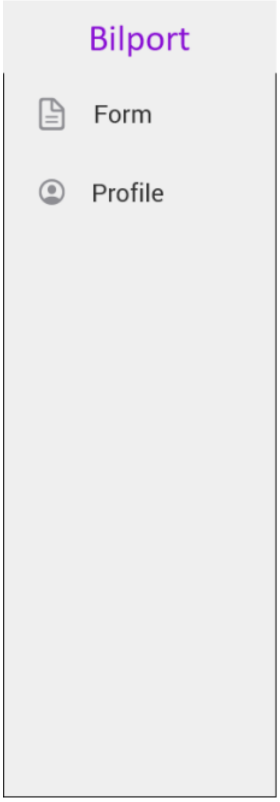


Student 1
Student 2

Student 3
Student 4
Student 5

Alphabetically

Supervisor:



4. References

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