Bilkent University 

**CS319 Object-Oriented Software Engineering**

**Project Final Report**

**Section: 2**

**Team: ABKUY**

**Group 2A:**

* **Ahmet Şahin (21902702)**
* **Barış Tan Ünal (22003617)**
* **Kaan Berk Kabadayı (22002580)**
* **Uğur Can Altun (22002701)**
* **Yusuf Şenyüz (21903105)**

**Instructor:** Eray Tüzün

**Teaching Assistants:** Emre Sülün, İdil Hanhan, Mert Kara,   
Muhammad Umair Ahmed, Metehan Saçakçı

**18.12.2022**

**TABLE OF CONTENTS**

[**1. Introduction**](#_e470i576xfli) **2**

[**2. Lessons Learned**](#_68dmwfgl7wh) **3**

[**3. User’s Guide**](#_v9c17m7v23g2) **4**

[3.1. Login Page](#_t4ef7k8mzzob) 4

[3.2. To-Do List Page](#_vb6u8ya0qsjc) 5

[3.3. Checklist Page](#_3i4nxf7hsdub) 6

[3.4. Deadlines Page](#_cssa90gbwam) 7

[3.5. Score Table Upload Page](#_v1iyk5ebpgft) 8

[3.6. Placements Page](#_t47i1rn9qy37) 9

[3.7. Wish List Page](#_m1q2pucogu1x) 9

[3.8. Course Proposals](#_nz6vwamt2l8h) 11

[3.9. Pre-Approval Page](#_hayw3t8pxg2a) 11

[3.10. Application Page](#_q08idirbu9d7) 12

[3.11. Learning Agreement Page](#_k973cwxznvou) 13

[3.12. Submit Transcript Page](#_vmwl1rtwcy82) 14

[3.13. Course Transfer Page](#_o22zhgsflzzs) 15

[3.14. Announcements Page](#_olffknhp0elz) 16

[3.15. Evaluations Page](#_sazioppq8mr) 16

[3.16. Messaging Screen](#_a8h5pltzrzfc) 18

[3.17. Forum Page](#_vndcjglrzqv7) 18

[3.18. Contacts Page](#_qnwtze7ei1ay) 19

[**Build Instructions**](#_p14hxad03laz) **20**

[**Work Allocation**](#_vykz55otgxor) **21**

[5.1. Ahmet Şahin](#_prxqvvwl66uo) 21

[5.2. Barış Tan Ünal](#_3dl2wz2lsr3t) 22

[5.3. Kaan Berk Kadabayı](#_au58o8afy34x) 22

[5.4. Uğur Can Altun](#_3ssxjfauxuhs) 23

[5.5. Yusuf Şenyüz](#_wl6mmc76d2sb) 24

[**What We Did And Did Not**](#_7vuem11aipy6) **25**

# Introduction

The implementation deadline for the Erasmus Application Manager has arrived as of 18 December 2022, and many core functionalities of the project are implemented and ready to use. Only a few core functionality and additional functionalities we have intended to implement are not implemented due to time limitations. Other than implementing most of the core functionalities, implemented core functionalities are tested thoroughly, and necessary precautions are taken for possible edge cases. In the project's current state, all user types can see their to-do lists and keep track of their tasks. Users can manually convert their tasks to completed or not completed on the to-do list page. Users can also delete their tasks manually to introduce flexibility to the to-do functionality. Students can prepare a wish list form with all the necessary properties and submit them for the coordinators' approval, and coordinators can view their students' pre-approval and approve or decline them. On the other hand, the International Students Office can submit incoming students' transcripts for course transfer and upload score tables for placements. Students can download and upload their learning agreement forms and keep track of their progress to Erasmus via the checklist page. Although visual layout and functionalities in the frontend part of announcements and evaluations pages are implemented, their connection to the backend is not implemented due to time limitations. The other additional features that are not implemented due to time limitations are as follows: forum screen, messaging, and notifications.

# **L**essons **L**earned

We started the analysis process as early as possible at the beginning. We made very frequent meetings and we felt that we have achieved a great understanding of the whole Erasmus and Exchange process. We took notes from Can Alkan’s presentation and analysed it. We held a meeting with the Administrator Erasmus Coordinator of the Engineering Faculty, Yelda İrem Ateş and asked questions about their role and the whole Erasmus process. We were more than content with our requirements elicitation. In the later stages, our effort on requirements elicitation paid off since there were no ambiguities about the Erasmus and Exchange process.

While we were writing the reports, more questions about the design arose and we did our best to find the proper tools that could tackle the problems we face. A couple of days after we started implementation, we have changed the version of Maven framework that we are using because we were unable to add the authorization feature that we want with the newest version. That slowed us down a little bit but we have learned the importance of choosing the right tools and versions.

In the last week of implementation, we met with each other every single day and did most of our work in that period. We made critical decisions about which features to include and give up. While taking those decisions, we focused on the main functionalities and gave importance to the features that we can implement in a shorter time but those that have greater effect on the overall experience of the application. We never lost hope about what we can do but we were also aware of the time constraints. The quote of Michael Jordan represents our feelings the best: “I never lost a game, I only ran out of time”.

# **U**ser**’**sGuide

## **3.1. Login Page**

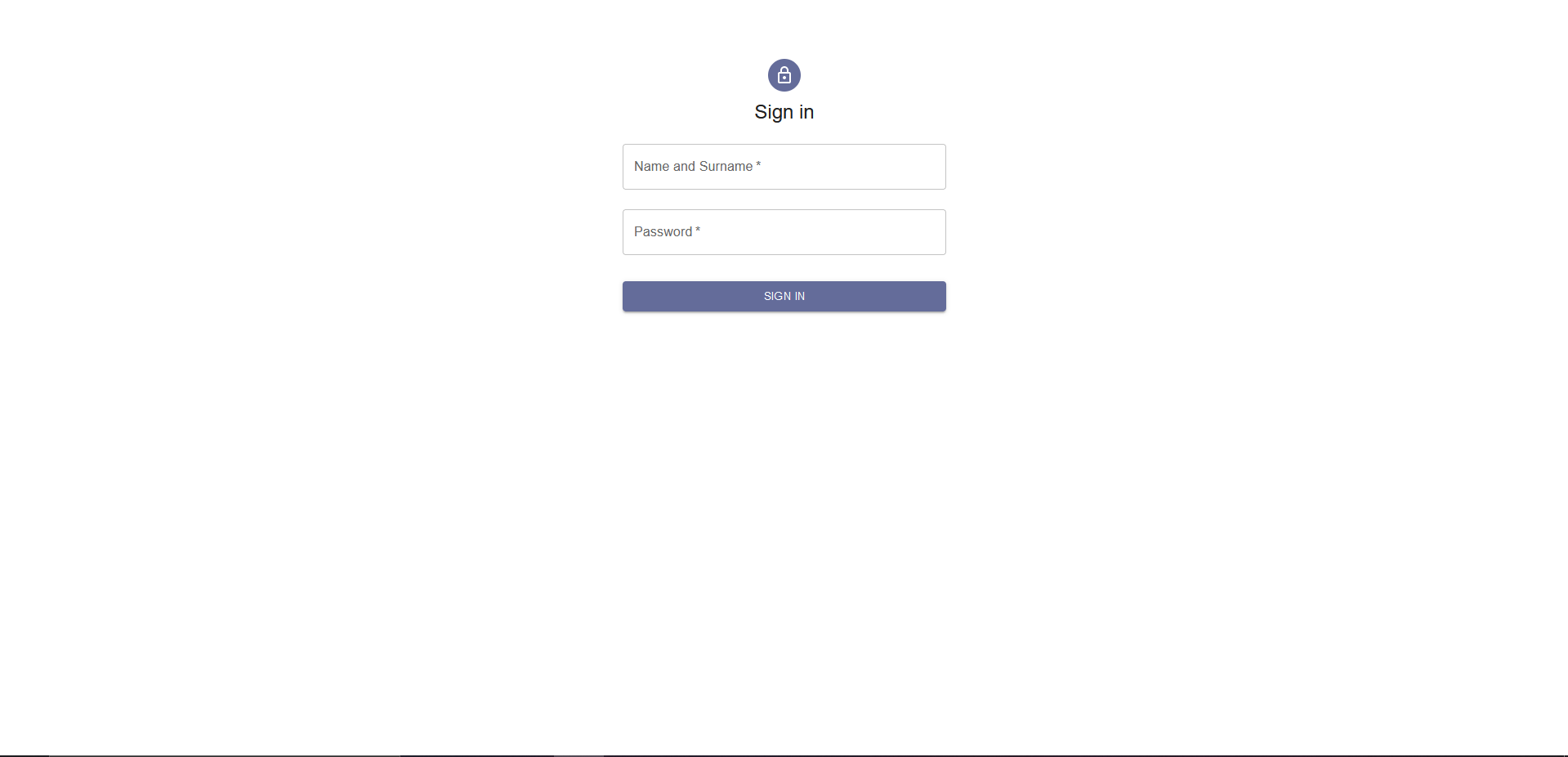


Fig. 1. Login Screen.

Login screen enables the allowed users to access the application. There is no registration because all users such as students, course coordinators, Erasmus coordinators are already registered to the STARS system. Users need to enter their Bilkent ID and password. This information is the same with the SRS system. Hence, users just should remember its STARS information to use our application.

## **3.2. To-Do List Page**

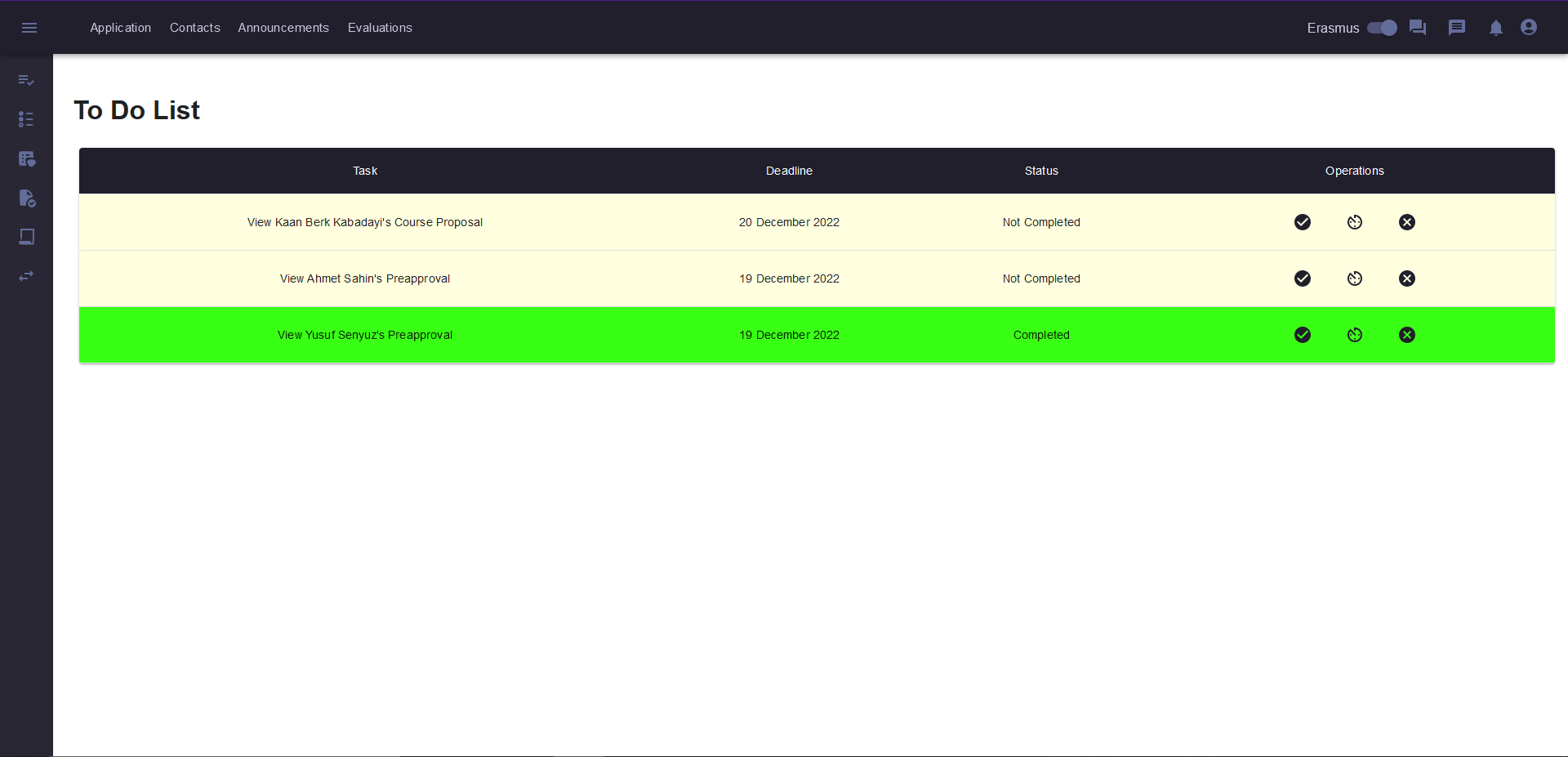


Fig. 2. To-Do List Screen.

After the login users see the home screen of the application. Each type of user sees the corresponding type of home screen. In the home screen navbar, sidebar, and to-do list are present. Navbar is located at the top of the screen and sidebar is located at the left side of the screen. Elements of the navbar and sidebar change depending on the required functionalities of the users. Each user sees their To-Do List, depending on their tasks. To-Do List is a dynamic list where users can see the tasks they should do. As the user completes their tasks, the system automatically updates the To-Do List. This way, every user can easily see what they should do whenever they login to the application. In the navbar, the user can switch between Erasmus and Exchange programs.

## **3.3. Checklist Page**

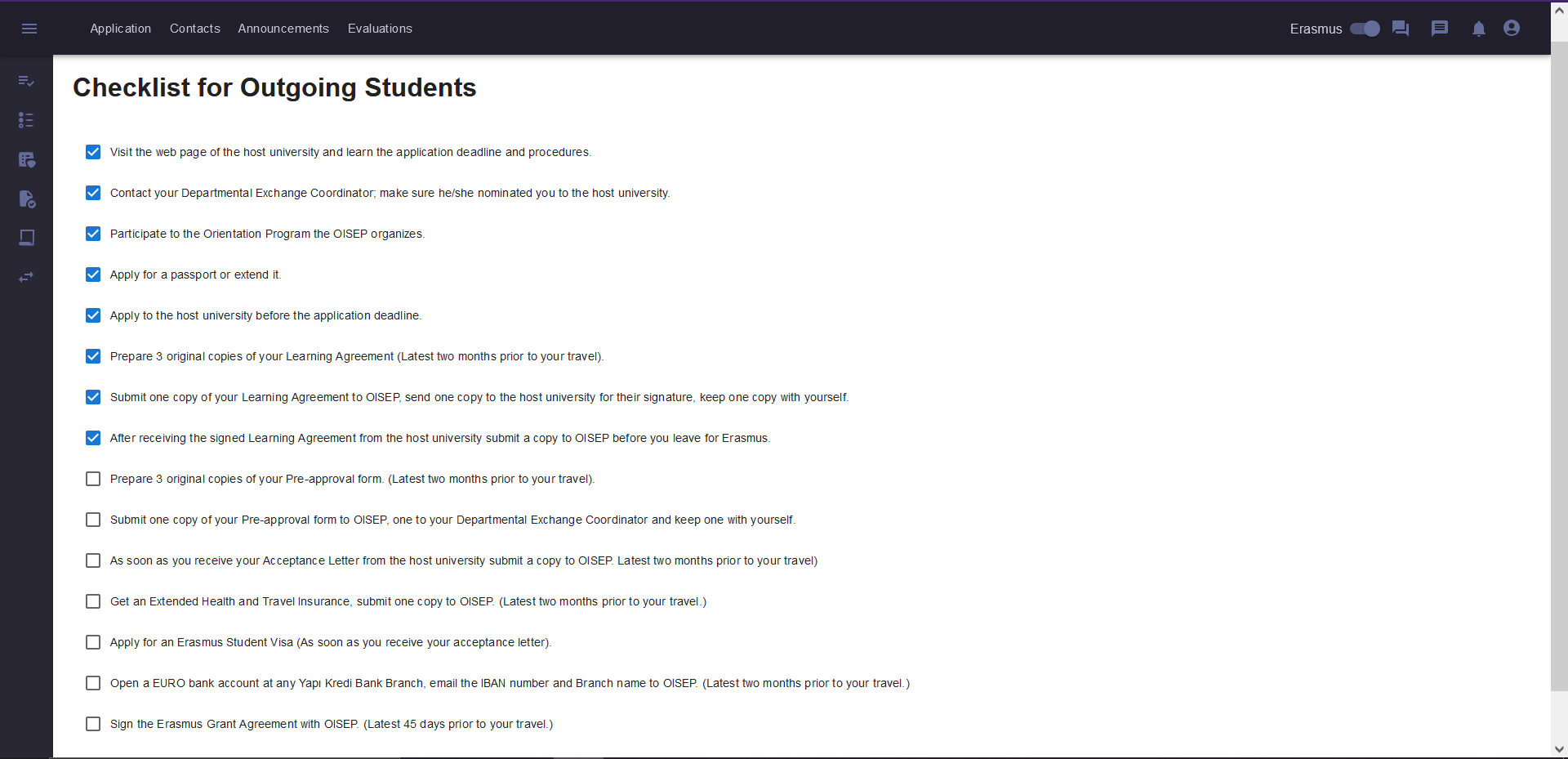


Fig. 3. Checklist Screen for Students.

Students can see the actions they should complete with intractable bullet points before mobility of the student starts. This list is a static list and is only accessible by students. This means students should manually mark the things they have done by clicking on the boxes. This is only for keeping track and does not have any connection to the other functionalities.

## **3.4. Deadlines Page**

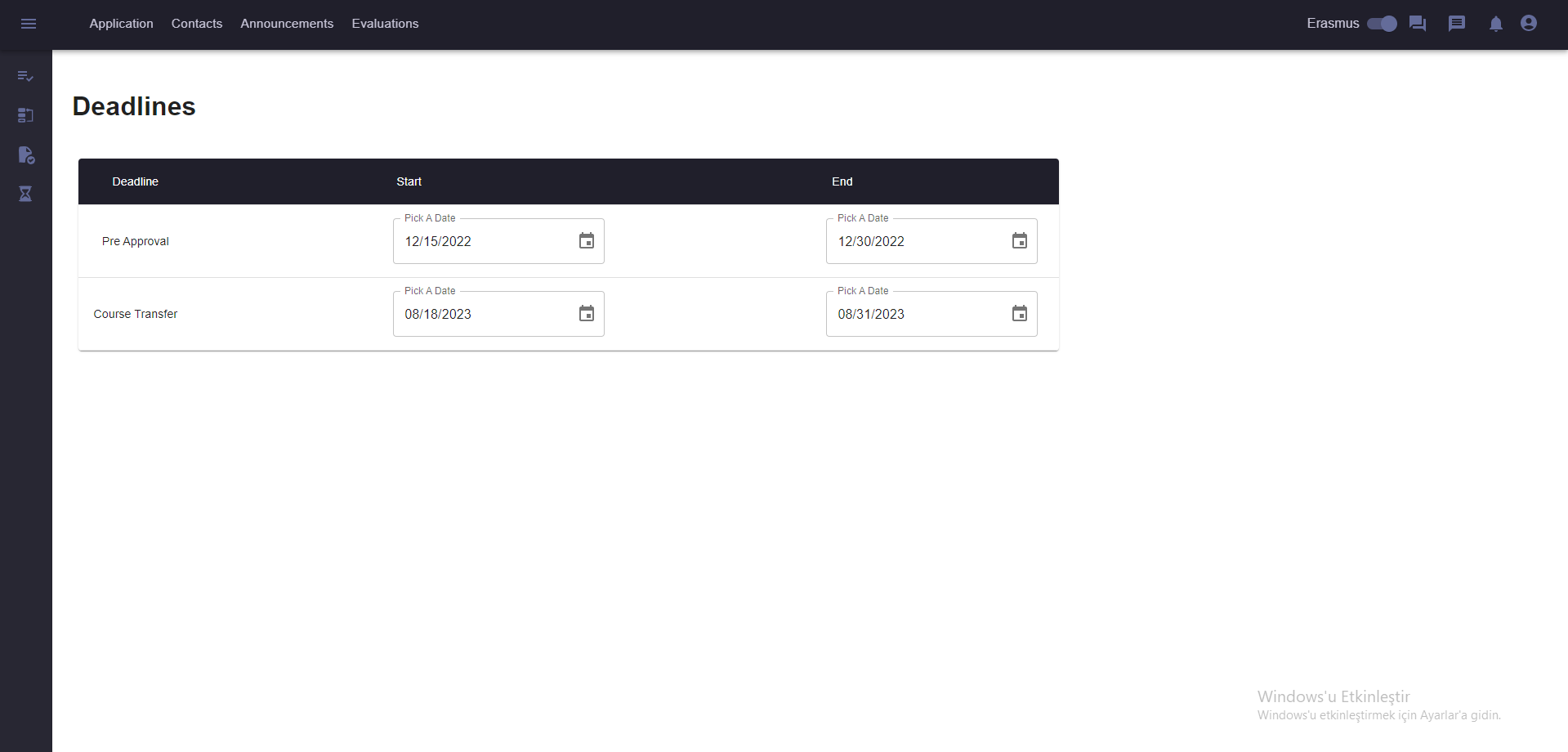
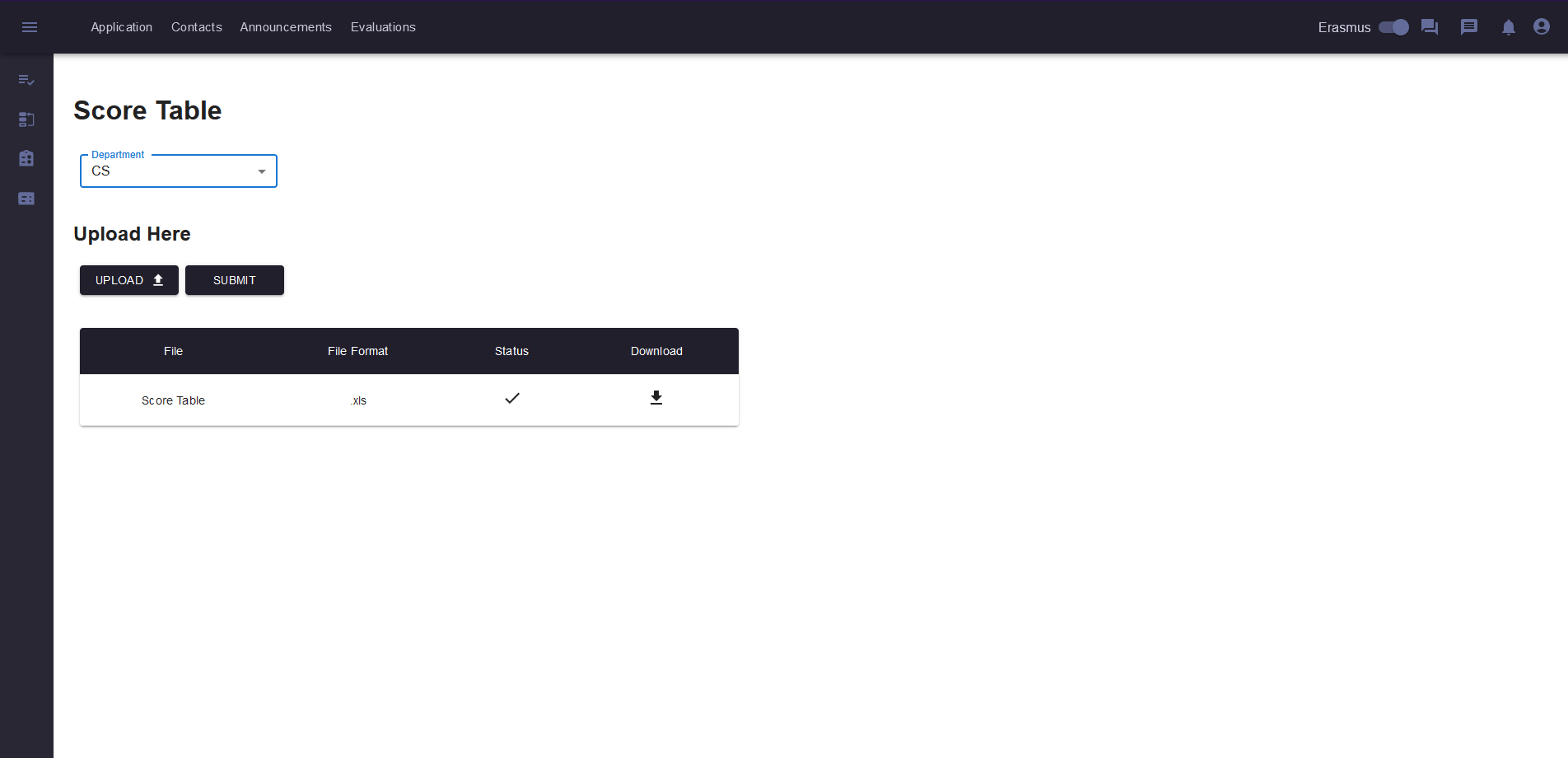


Fig. 4. Deadline Screen for Administrative Erasmus Coordinator.

The Administrative Erasmus Coordinator can set the deadlines for important activities such as Pre-Approval creation, Course Transfer period deadline etc. The students and other users need to obey these deadlines for them to proceed with their Erasmus/Exchange programs.

## **3.5. Score Table Upload Page**

Fig. 5. Score Table Upload Screen for International Students Office.

The whole Erasmus and Exchange process in our application starts with the upload of the score table which is done by the International Students Office. Each department has their own score table, therefore the process for each department starts independently. To start the process, the International Students Office can navigate to the “Upload Score Table” from the sidebar. On that page, there is an “upload file” button. The International Students Office should upload a valid Excel file there with extension “.xlsx”. Excel files above version 2007 with proper column orders and contents should work just fine.

## **3.6. Placements Page**

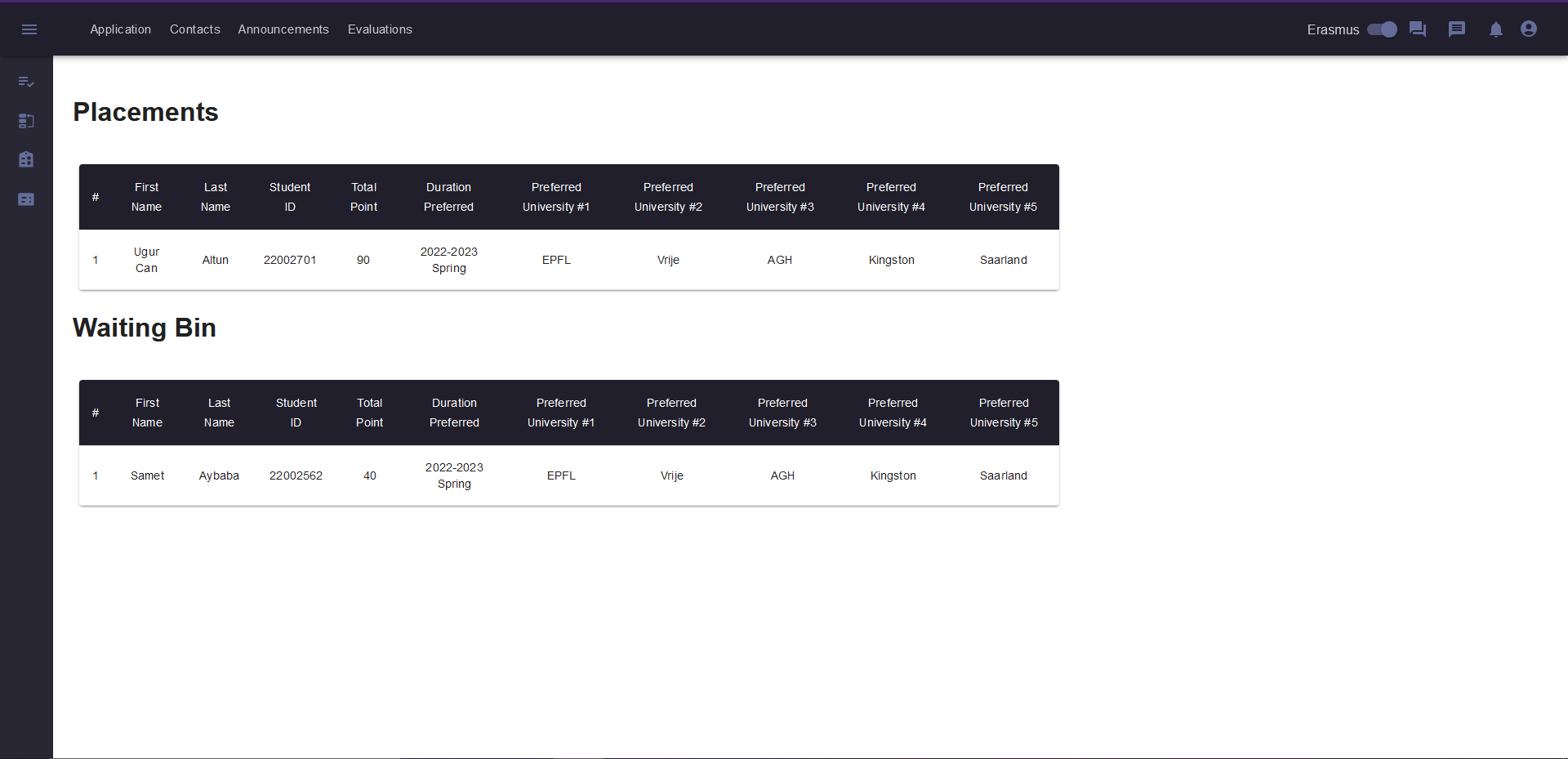


Fig. 6. Placements Page for Department Erasmus Coordinator.

Placements screen displays the placed students and students in the waiting bin and their related information such as their total points, duration preferred and their preferred universities.

## **3.7. Wish List Page**

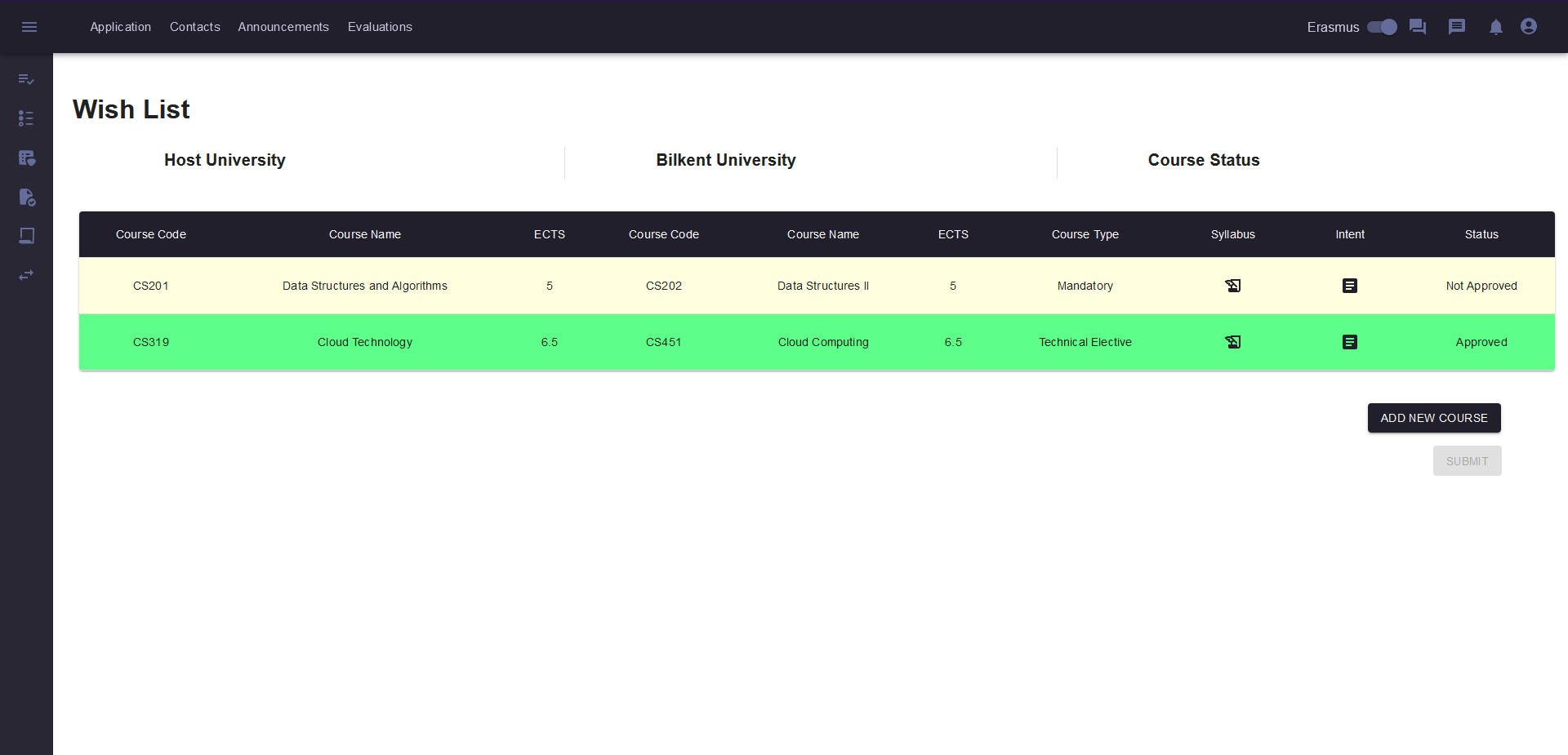


Fig. 7. Wish List Screen for Students.

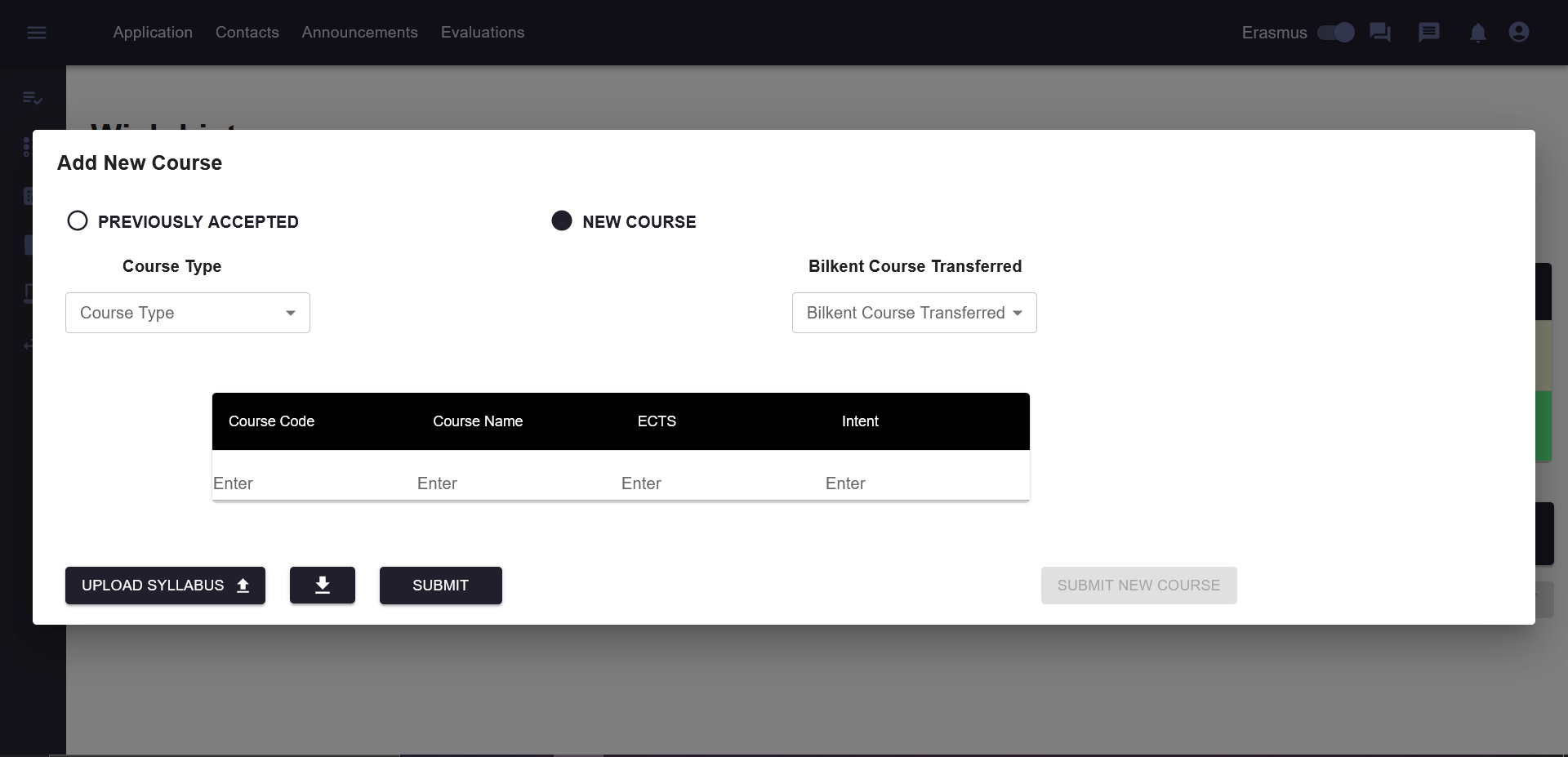


Fig. 8. Add New Course Dialog for New Course.

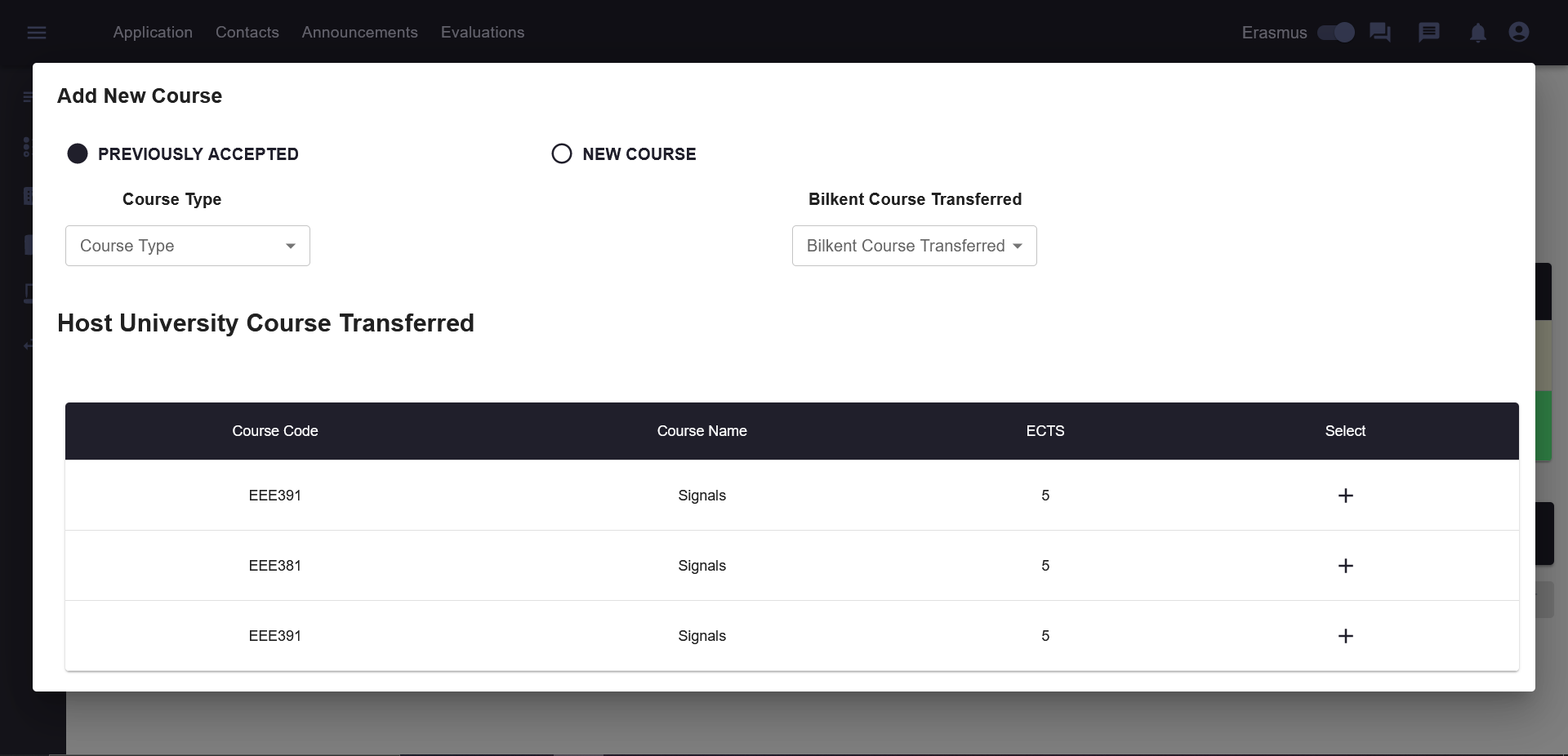


Fig. 9. Add New Course Dialog for Previously Accepted Courses.

After placements are finalised and the students are placed to the universities, students can create their course Wish List from the Wish List page. In that screen they should add courses by clicking the “add new course” button. When they add all the courses they want to take during Erasmus, they can submit their Wish List so that they are visible to the Department Erasmus Coordinator. The student can either choose from previously accepted courses or propose a new course that is not previously accepted.

## **3.8. Course Proposals**

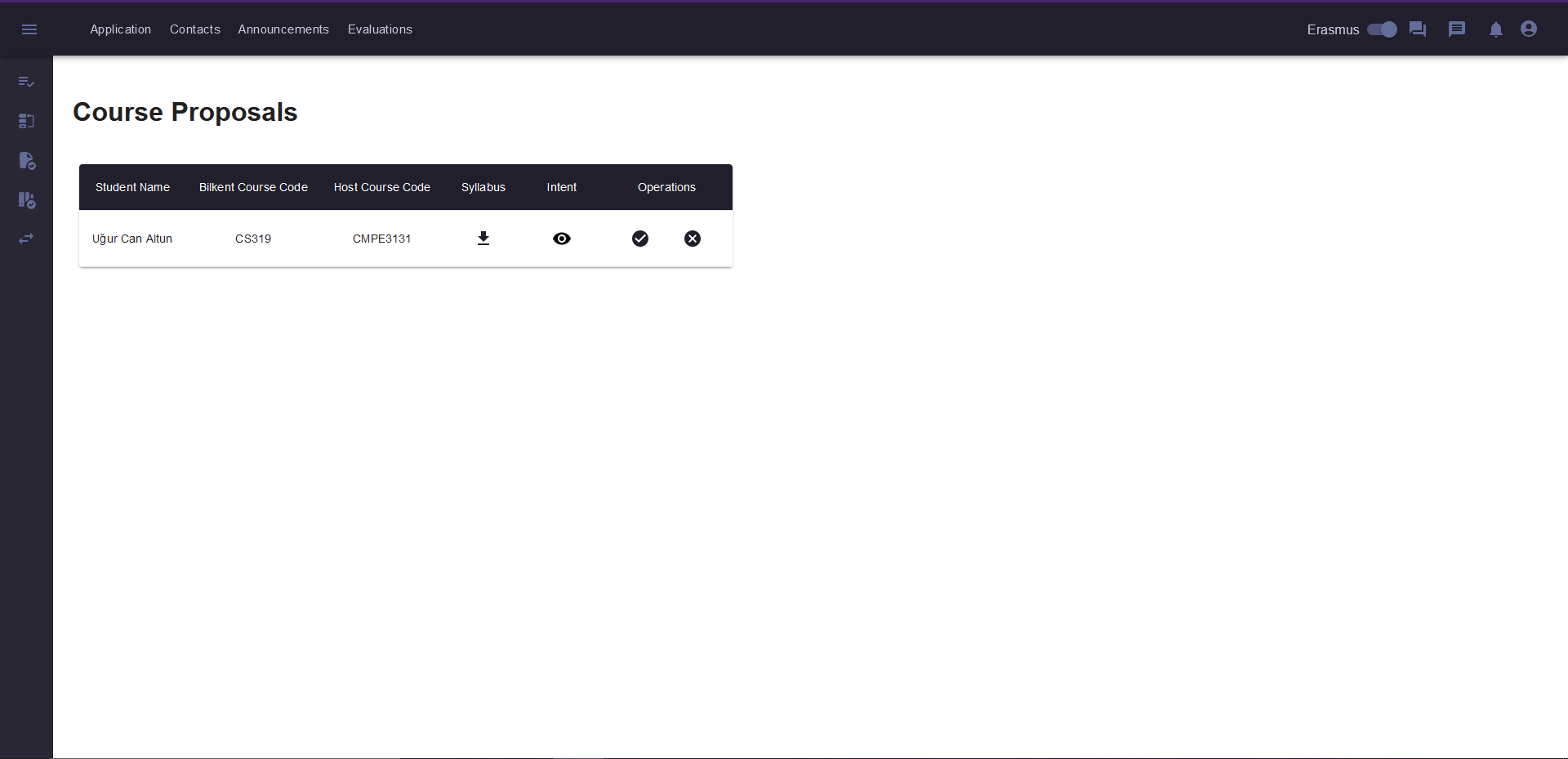


Fig. 10. Course Proposal Screen for Course Coordinator.

Course coordinators can view the related course proposals of the students in the course proposals page. Course coordinators can download the syllabus of the course proposal, view the intent and approve or decline the given course proposal of the student.

## **3.9. Pre-Approval Page**

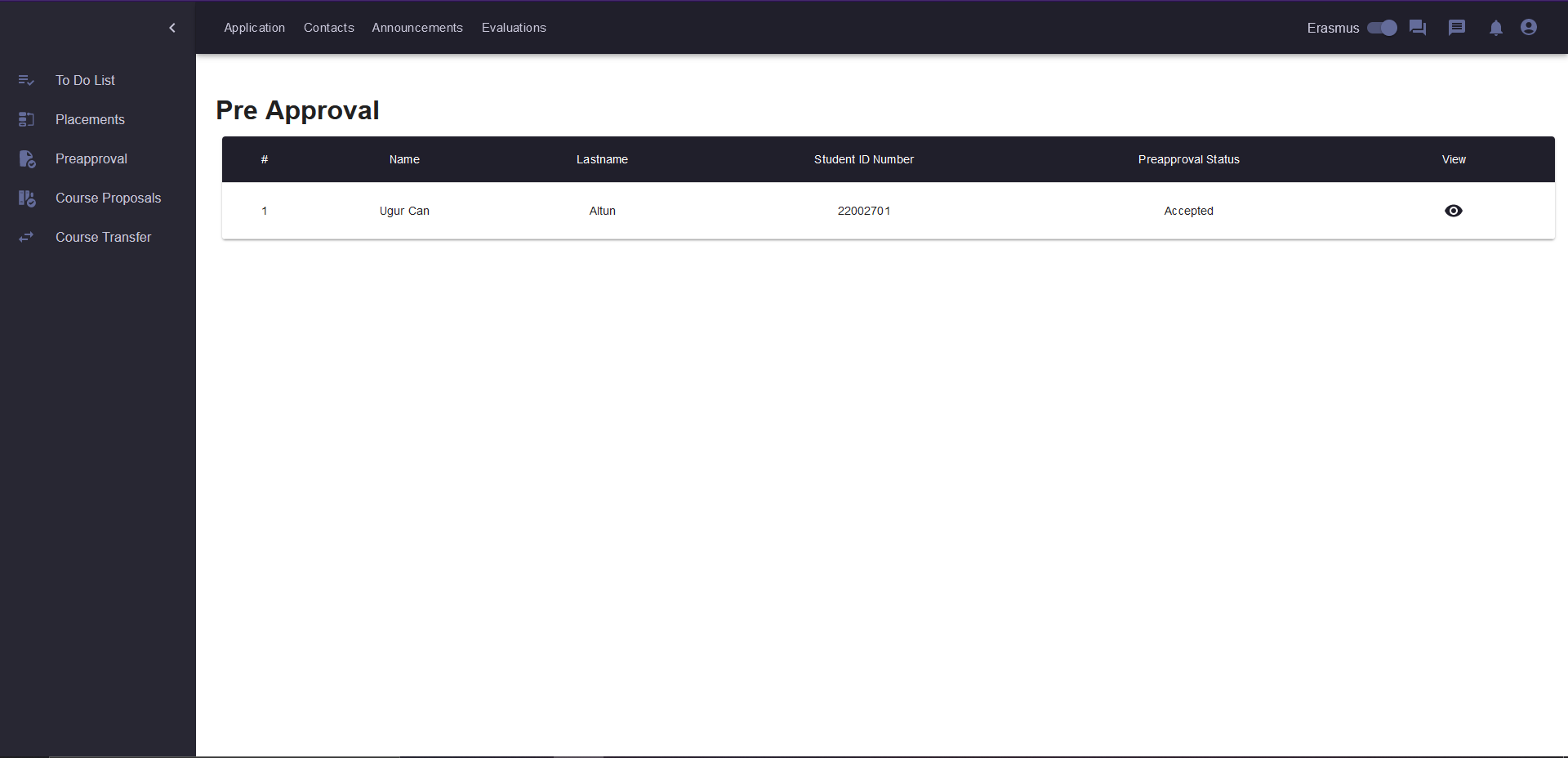


Fig. 11. Pre-Approval Screen for Department Erasmus Coordinator.

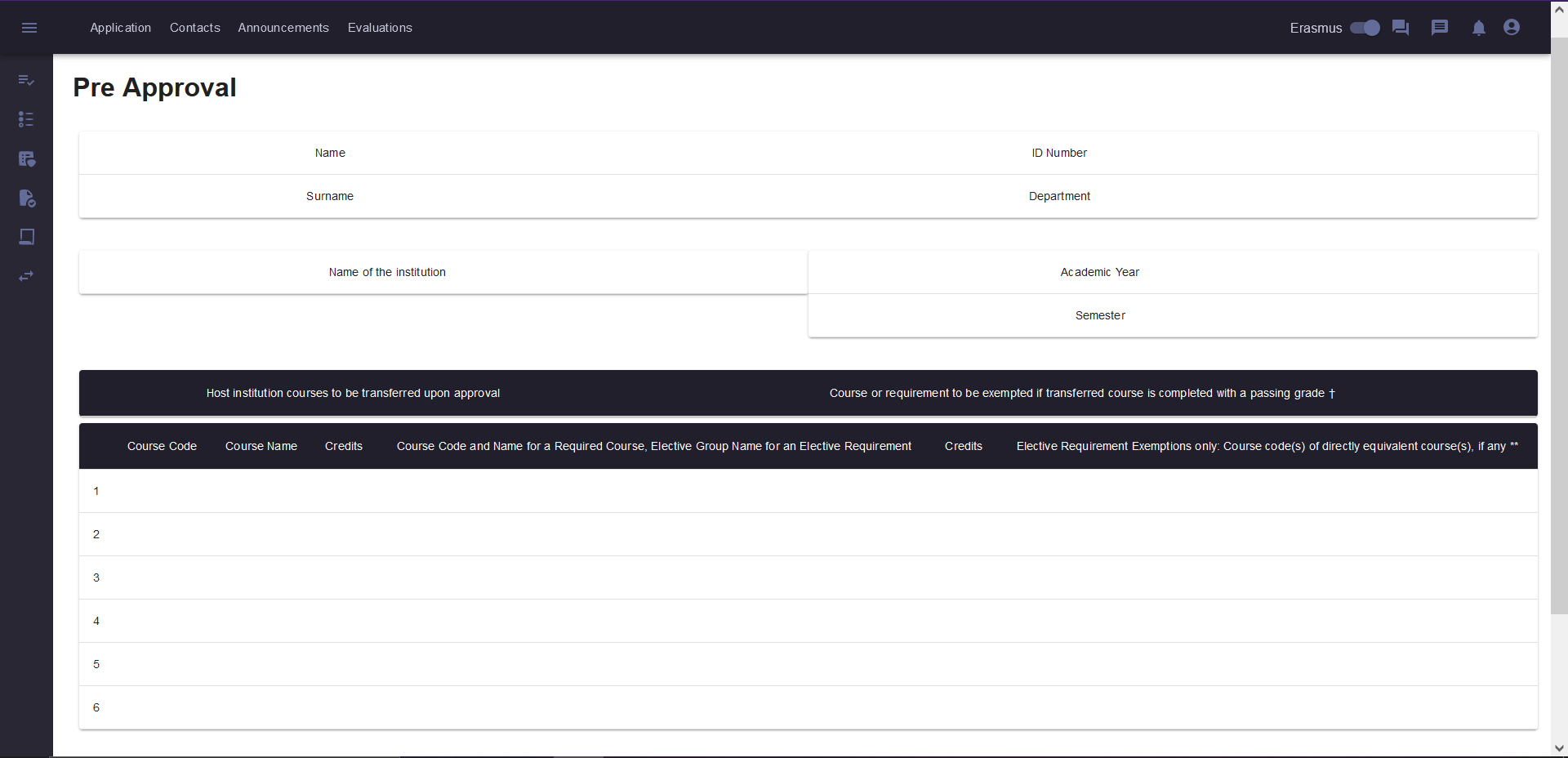
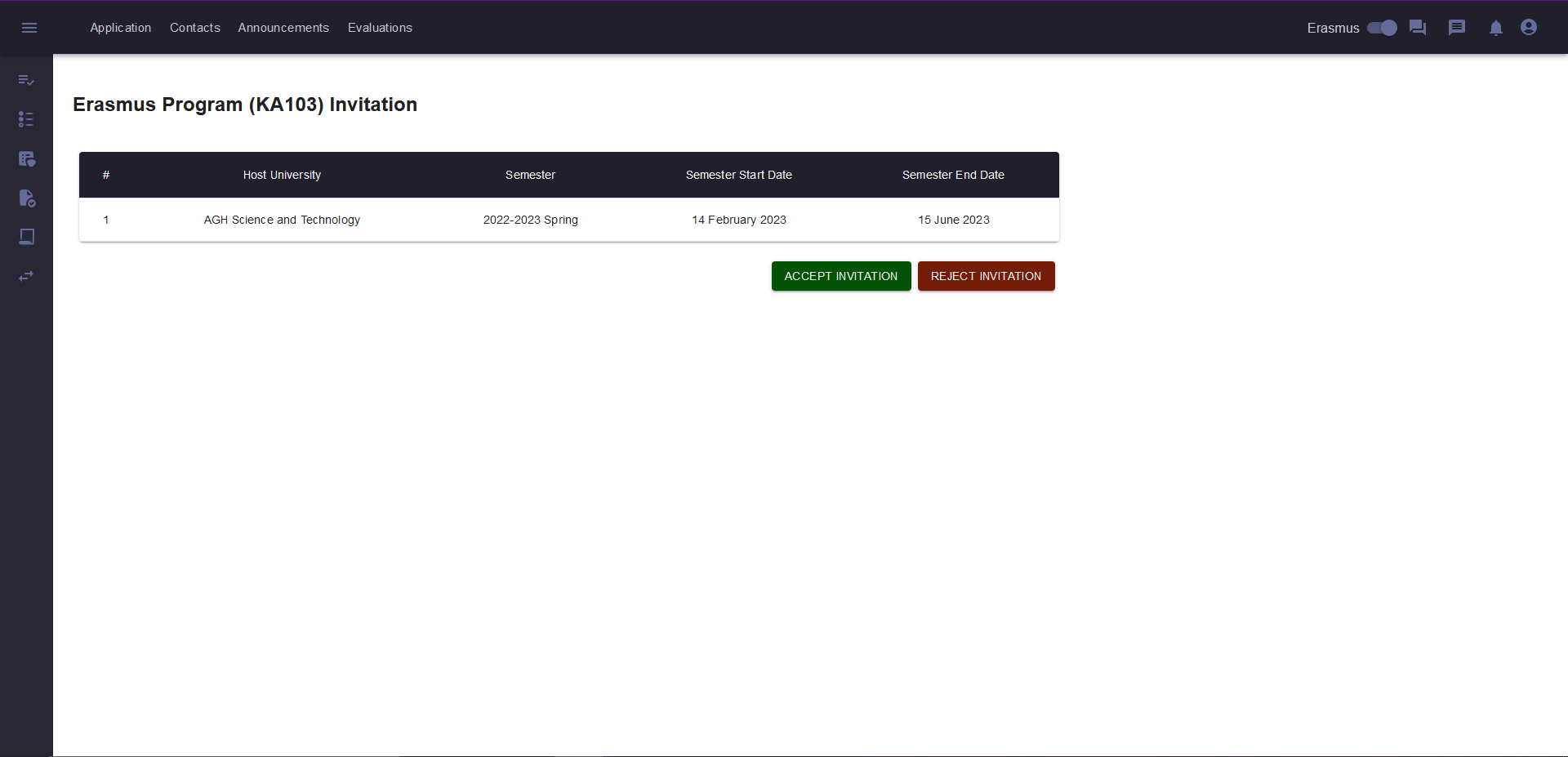


Fig. 12. Pre-Approval Screen for Students.

Students can see their Pre-Approval Forms in this Pre Approval page, once their wish list is approved by the system and contains courses that are approved by their respective coordinators or they are from the previously accepted. The students can download their Pre Approval forms in pdf format. On the other hand, the Department Erasmus Coordinator, can approve or decline these Pre Approvals by clicking accept or decline.

## **3.10. Application Page**

Fig. 13 Application Screen for Students.

Students can see their applications on the Application page by clicking it on the Navbar. Applications can contain Erasmus applications or Exchange applications or both. In this screen, applications and their relevant information are displayed. If the student is in the waiting bin and a request to be placed arrives, he/she can accept or decline the request. If the student is already placed, he/she only sees a button to cancel his/her application.

## **3.11. Learning Agreement Page**

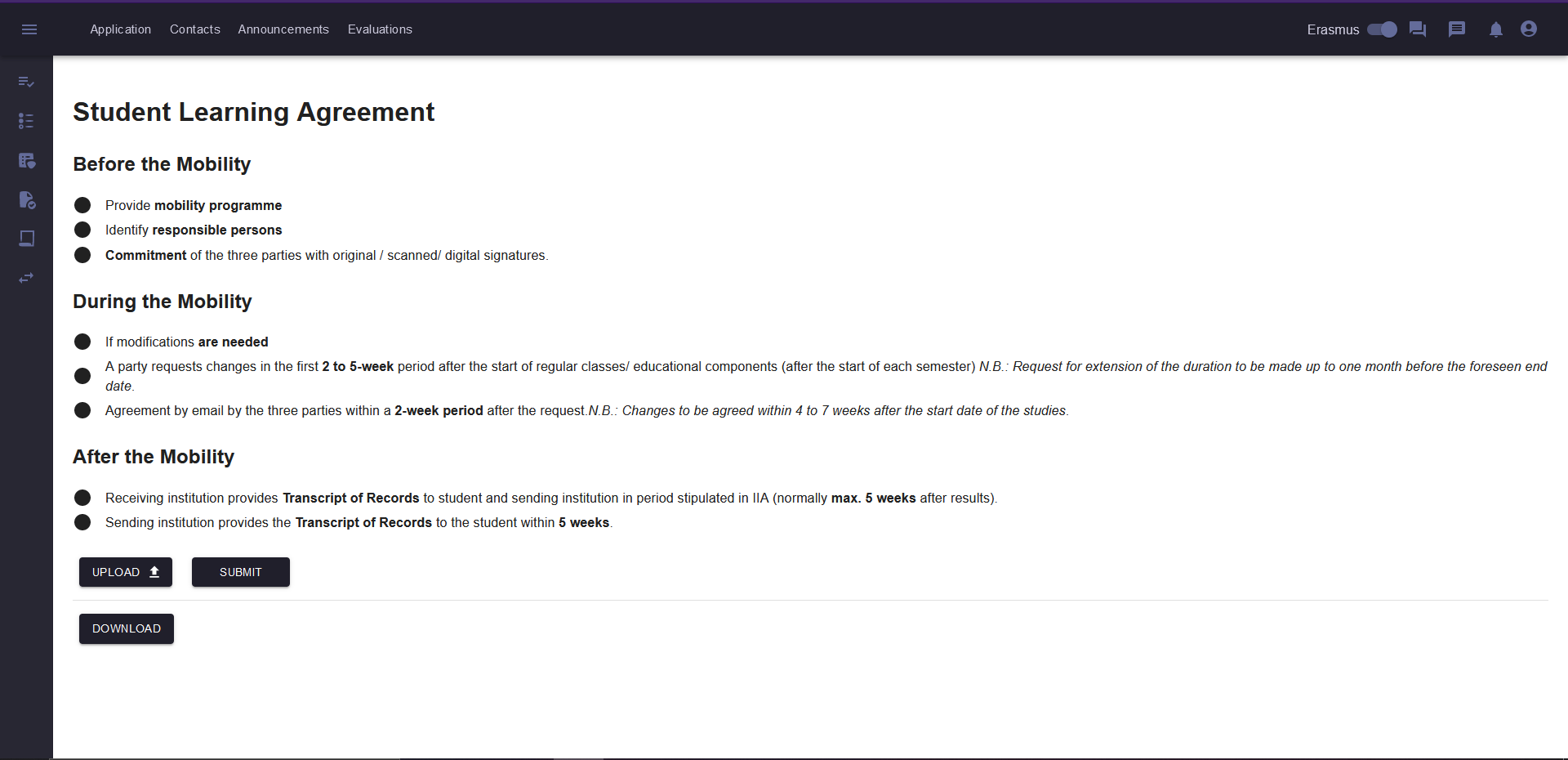


Fig. 14. Student Learning Agreement Screen.

In this page, the student must submit his/her Learning Agreement Form before starting the mobility. Students should first download the empty Learning Agreement document then fill the required cells in the form and upload to the system. Once the file is uploaded, they can download their filled versions of their forms.

## **3.12. Submit Transcript Page**

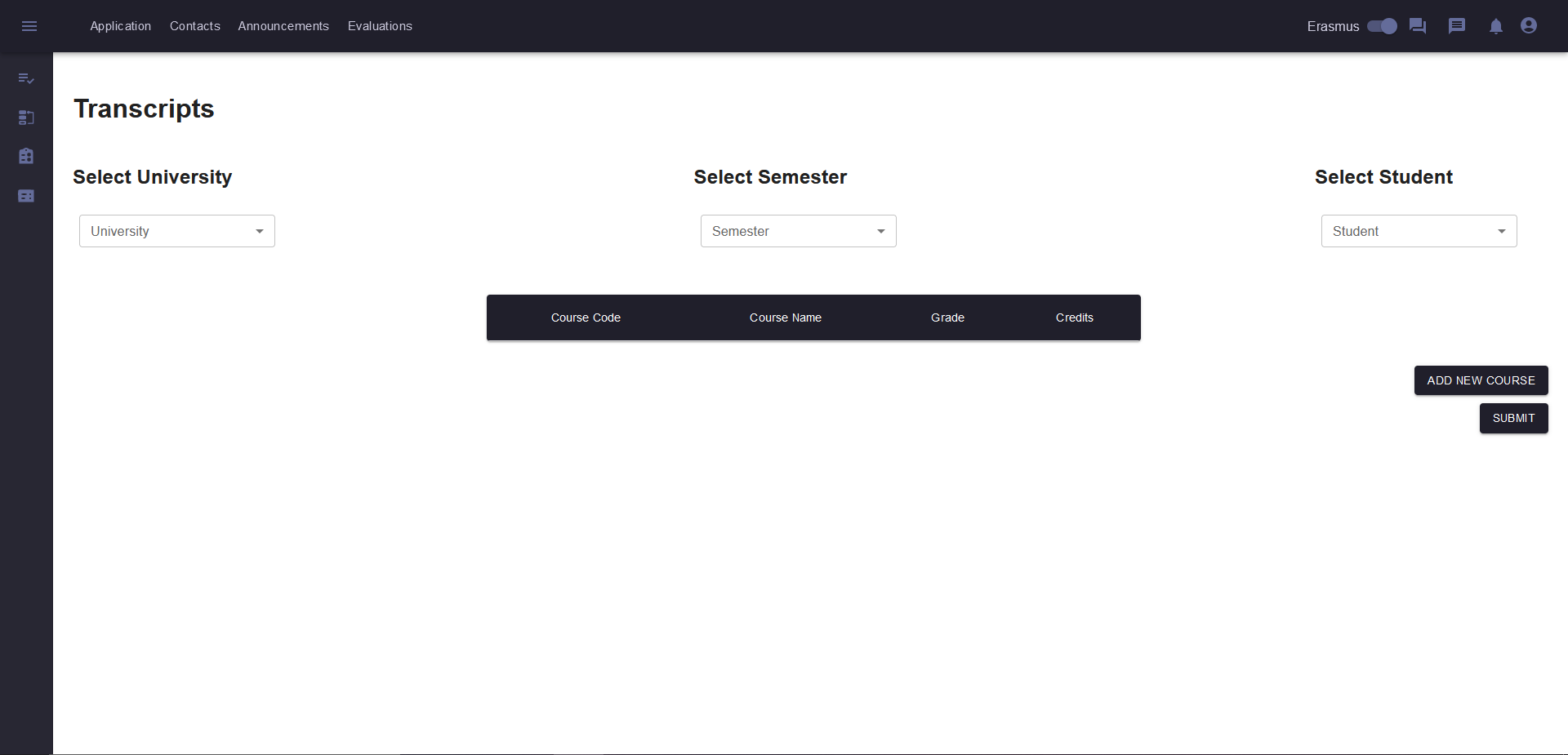


Fig. 15. Submit Transcript Screen for Admin Erasmus Coordinator.

After a student returns from mobility, the host university sends the transcript of that student to the ISO by email. After that ISO should go to the Submit Transcript Page. In this page, the International Students Office (ISO), selects the university, semester and student. Then, he/she submits the transcripts of returning students by adding new courses and once they are all added to the system, the ISO clicks the “submit” button to submit these data to the system for the course transfer logic.

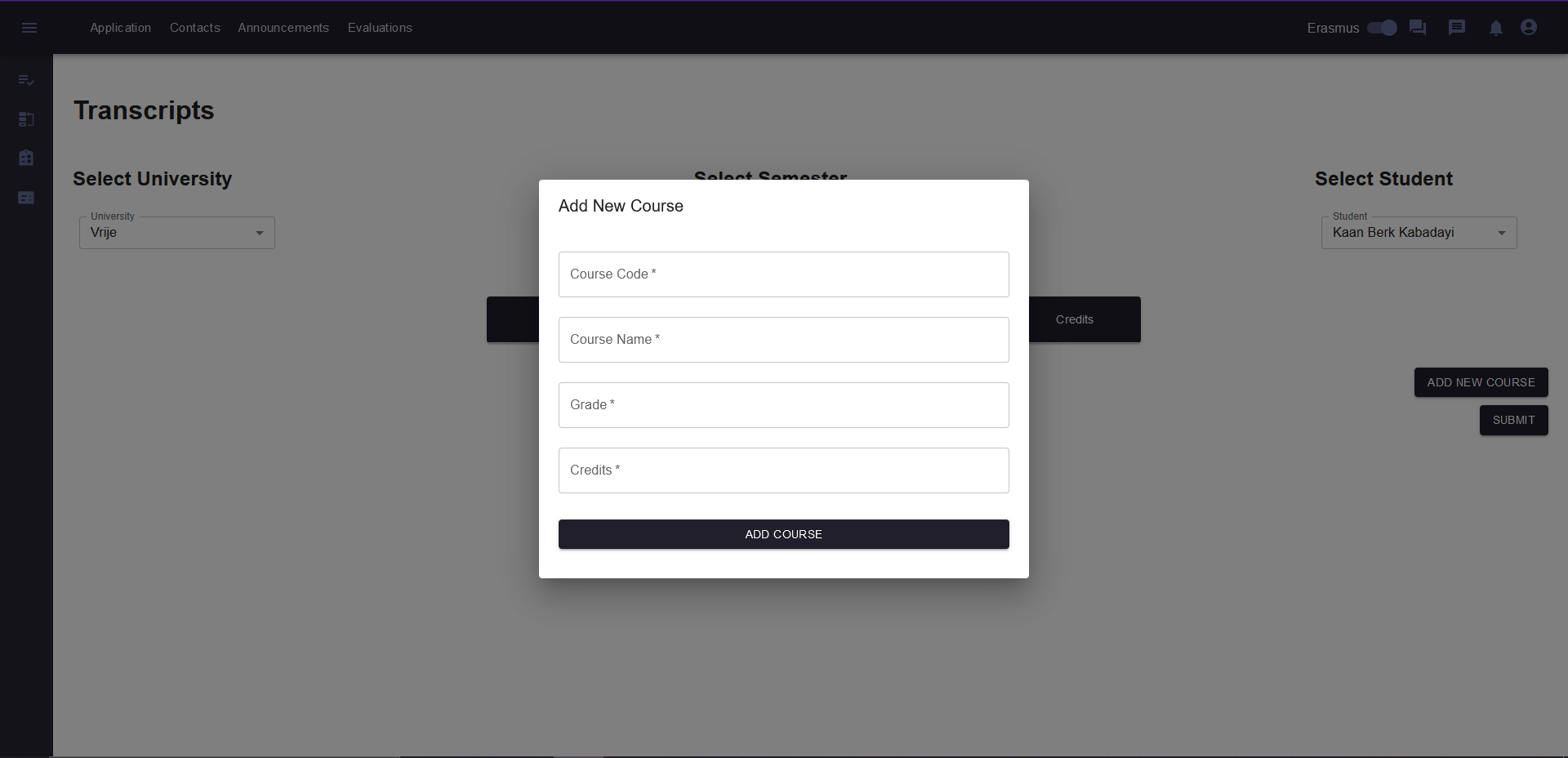


Fig. 16. Add New Course Dialog in Submit Transcript Screen.

Once the ISO clicks the “Add New Course” button, the ISO specifies the Course Code, Course Name, Grade and Credits of the course which are contained in the returning student’s transcript.

## **3.13. Course Transfer Page**

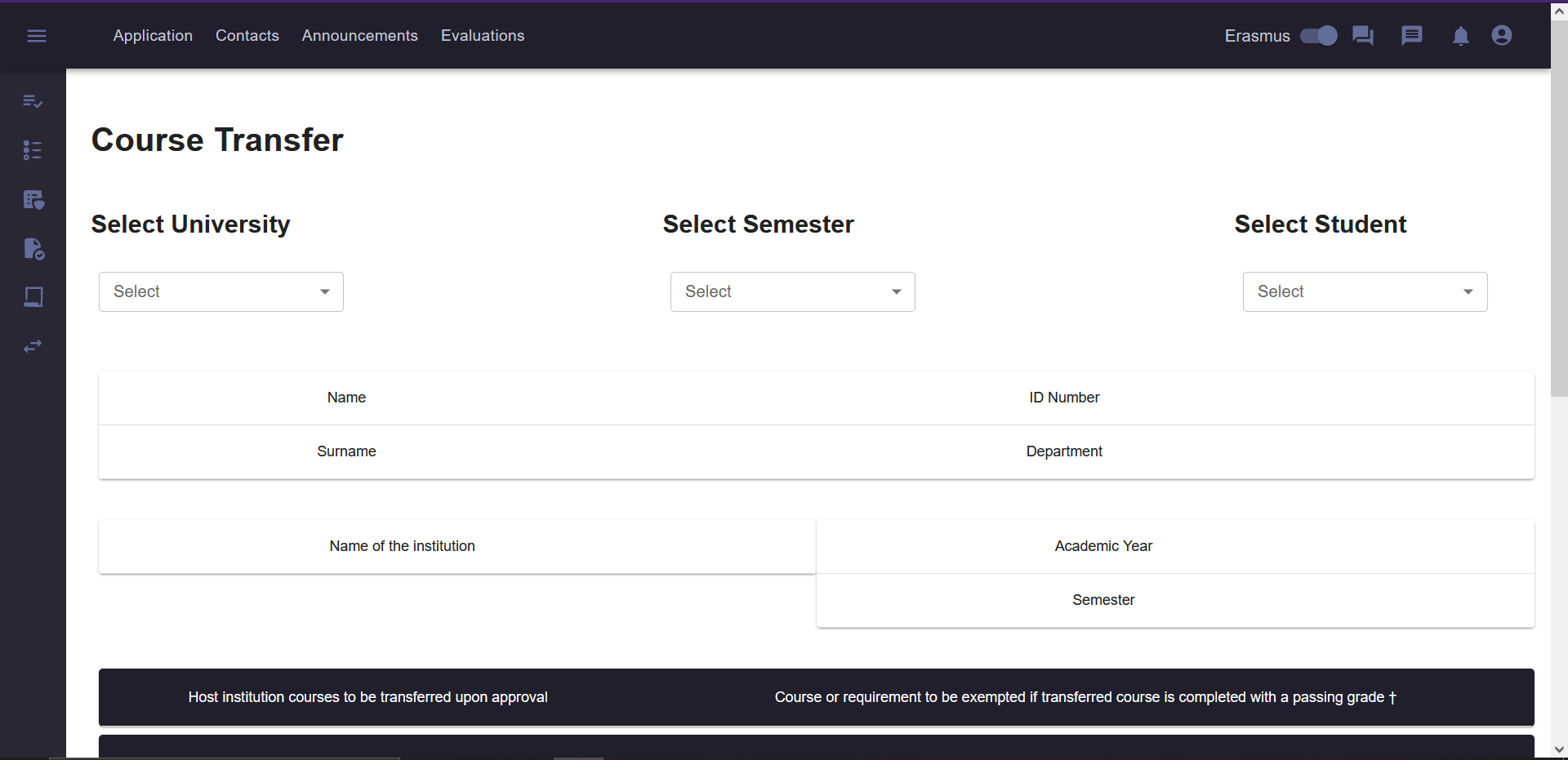


Fig. 17. Course Transfer Screen for Department Erasmus Coordinator.

The Department Chair, Dean and the Department Erasmus Coordinator can view this page by clicking the Course Transfer icon in the Sidebar. After selecting the university, semester and student, the Course Transfer page displays the selected students Course Transfer. Once the Dean/Department Chair/Department Erasmus Coordinator is satisfied with the request they can accept/decline the form.

## **3.14. Announcements Page**

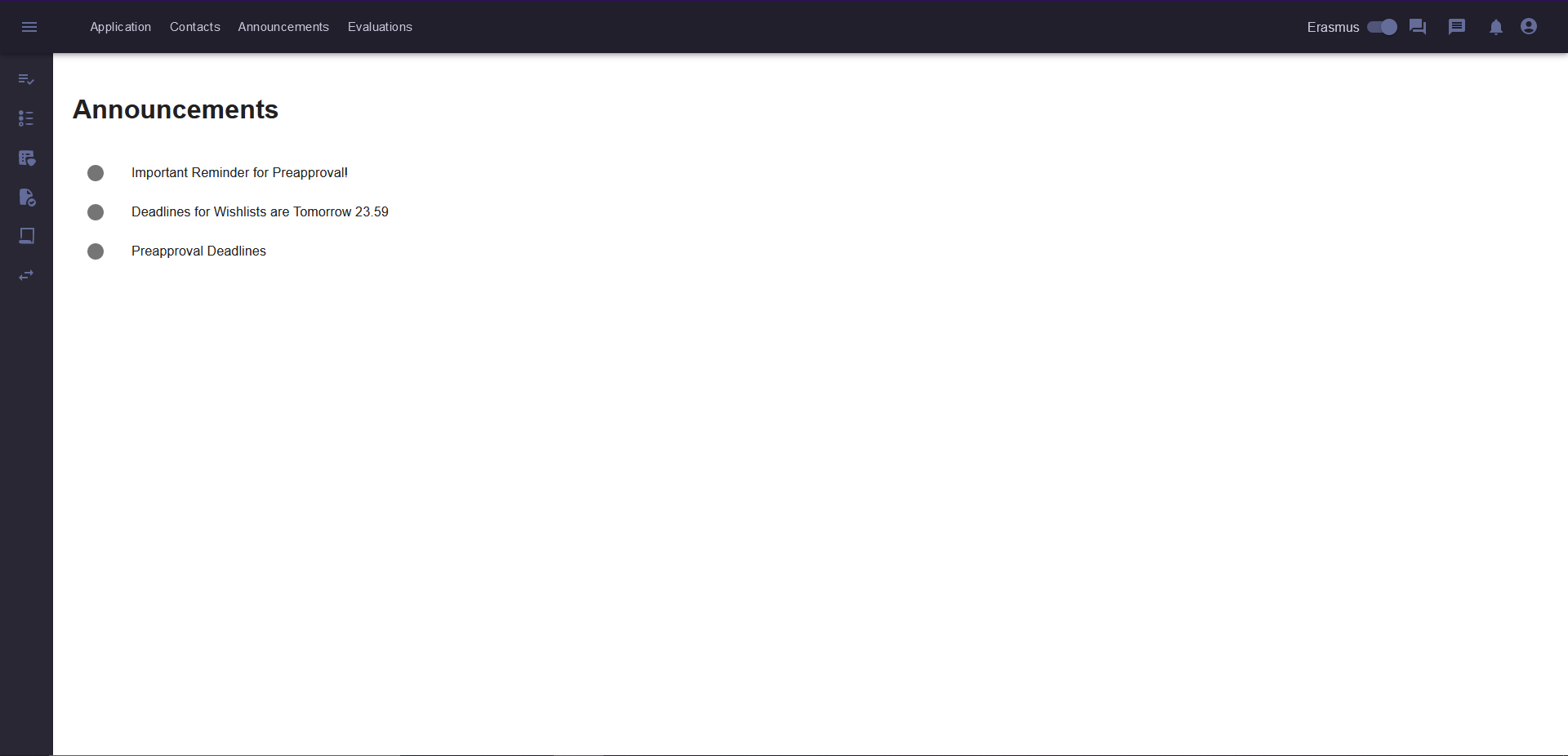


Fig. 17. Announcements Screen.

Users can access the Announcements page by clicking the Announcements button that is located on the navbar. Announcements page contains the announcements about the Erasmus applications and Erasmus procedure. Announcements are related to the deadline of Preapproval and similar procedures. Reminders are added to this page to inform users.

## **3.15. Evaluations Page**

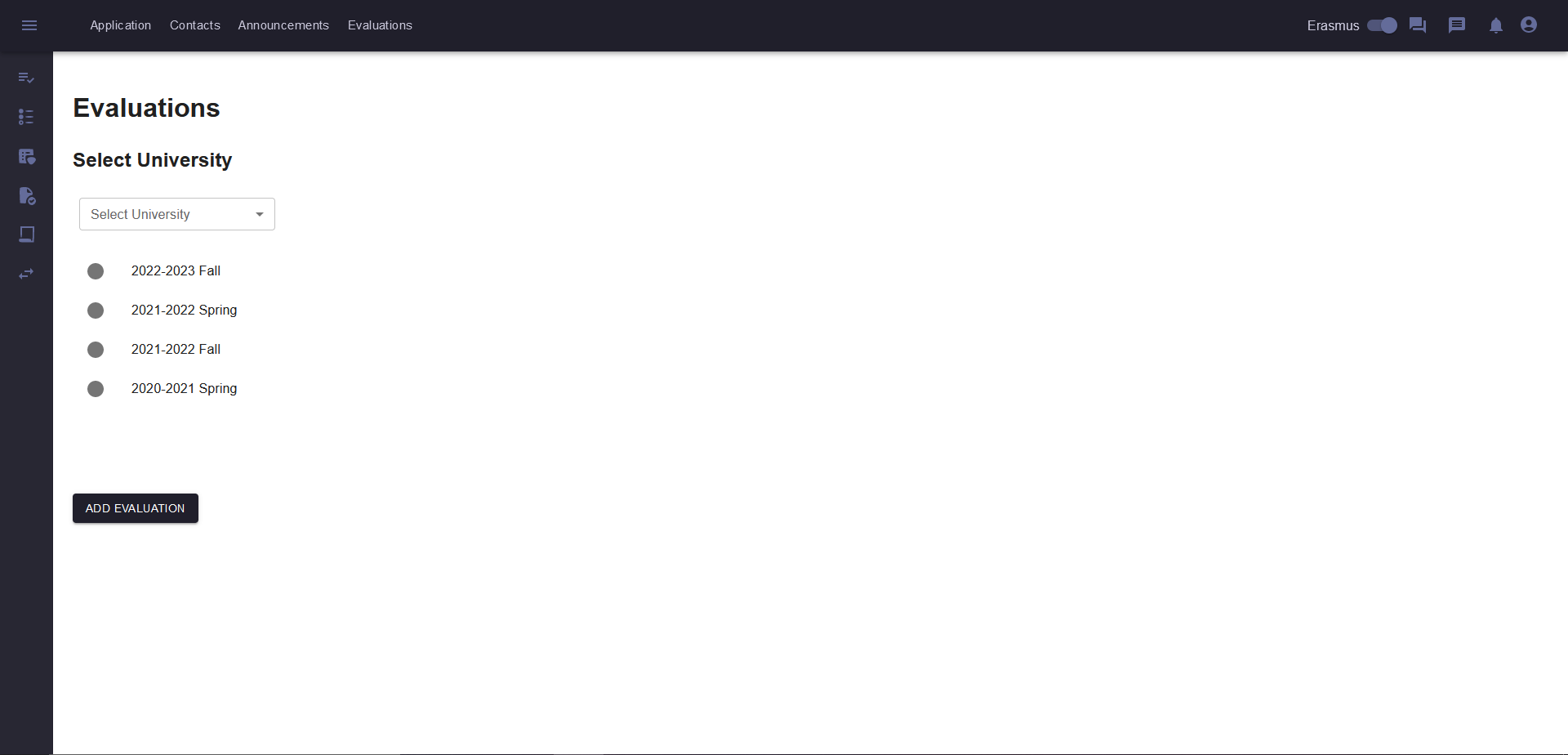


Fig. 18. Evaluations Screen.

In this page, students who have completed their erasmus program can evaluate their erasmus experience by adding evaluation via “add evaluation” button and students who are planning to go to Erasmus and other user types can view the existing evaluations by selecting a university and a semester.

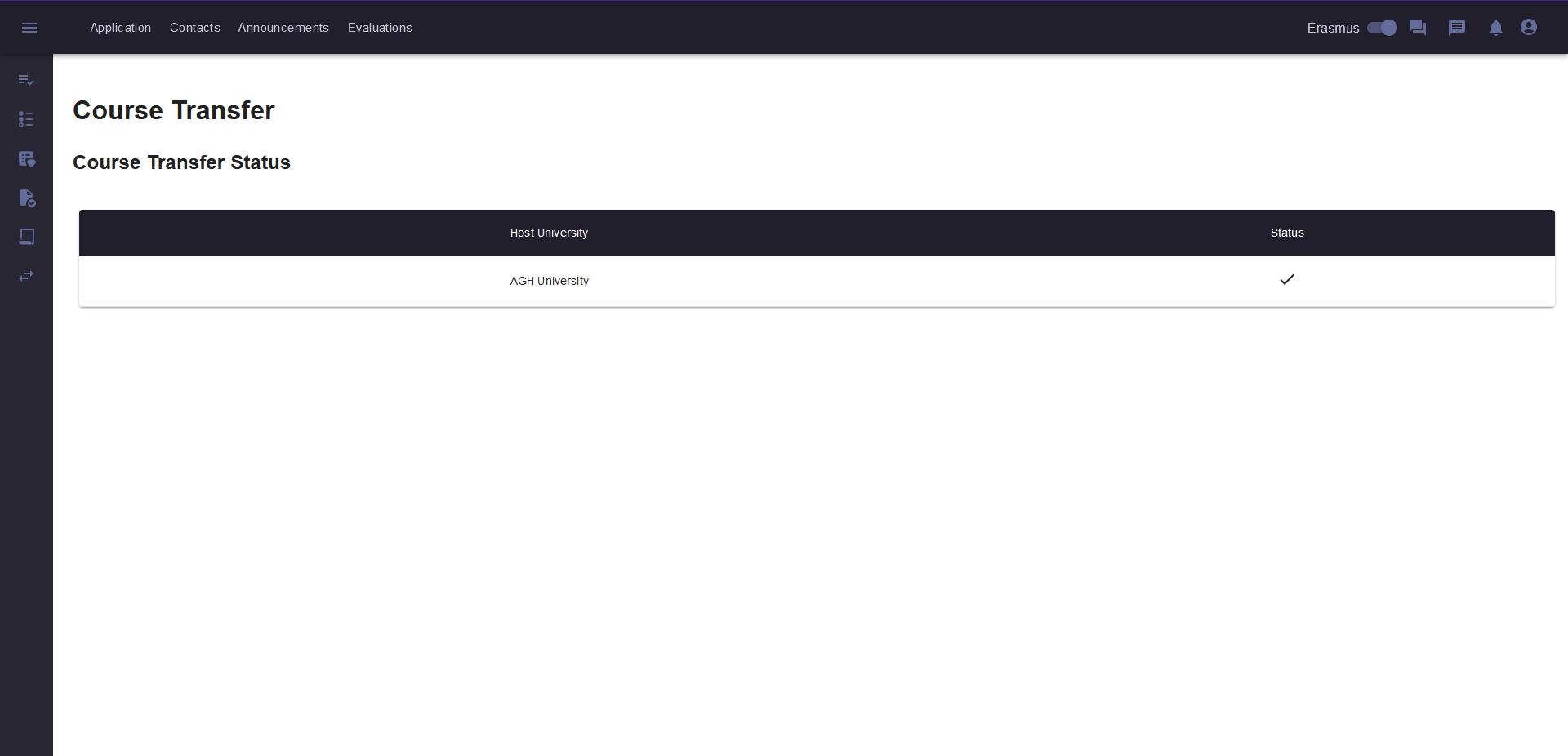


Fig. 19. Course Transfer Screen for Students.

Students can view their Course Transfer status in the Course Transfer tab. This page is read-only, as the student cannot do anything in their course transfer process. If all of the courses are transferred to STARS, the student sees their Course Transfer’s status as a tick.

## **3.16. Messaging Screen**

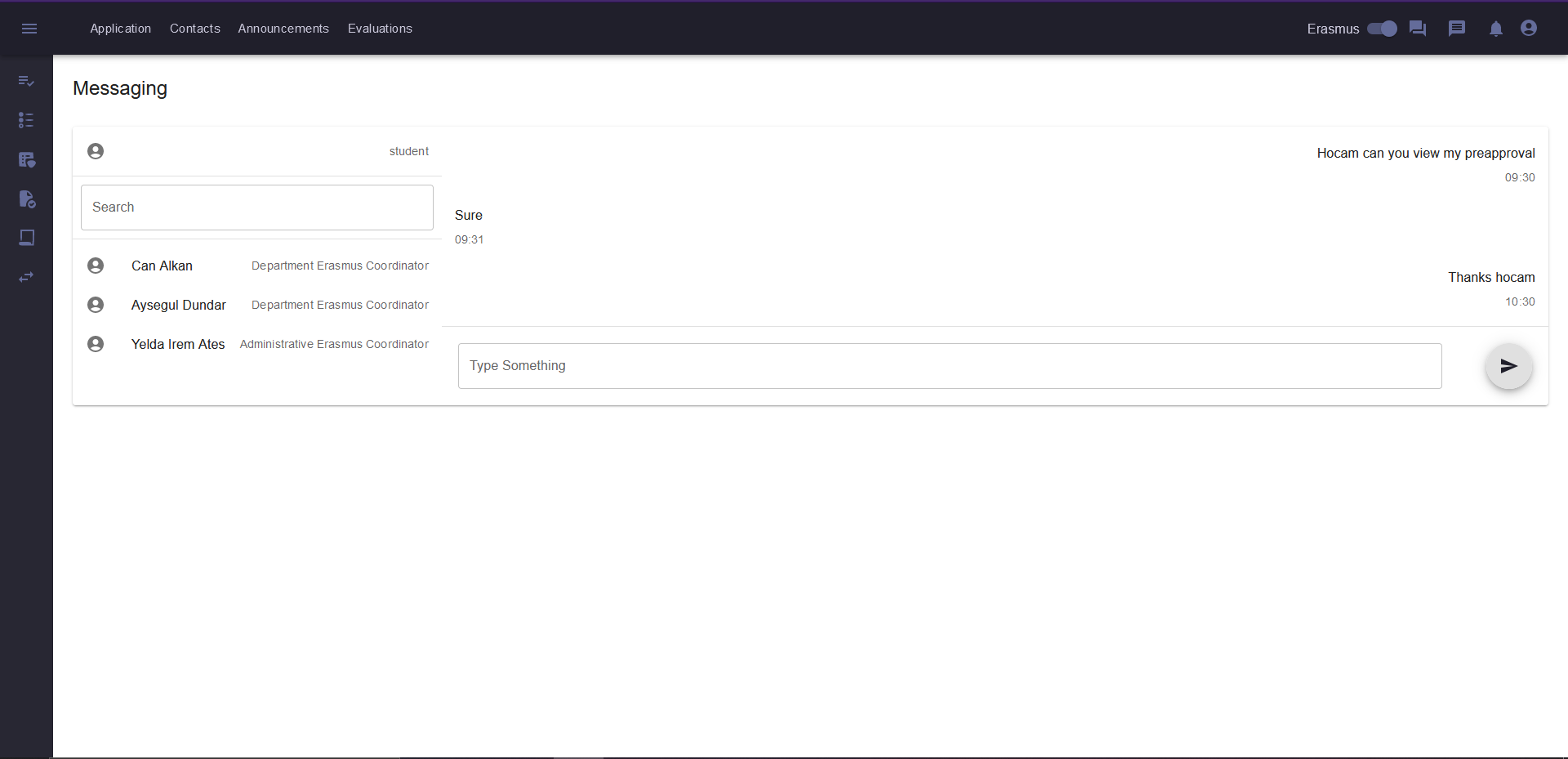


Fig. 20. Messaging Screen.

Messaging screen enables the students to contact department erasmus coordinator(s) and administrative erasmus coordinator directly via direct message.

## **3.17. Forum Page**

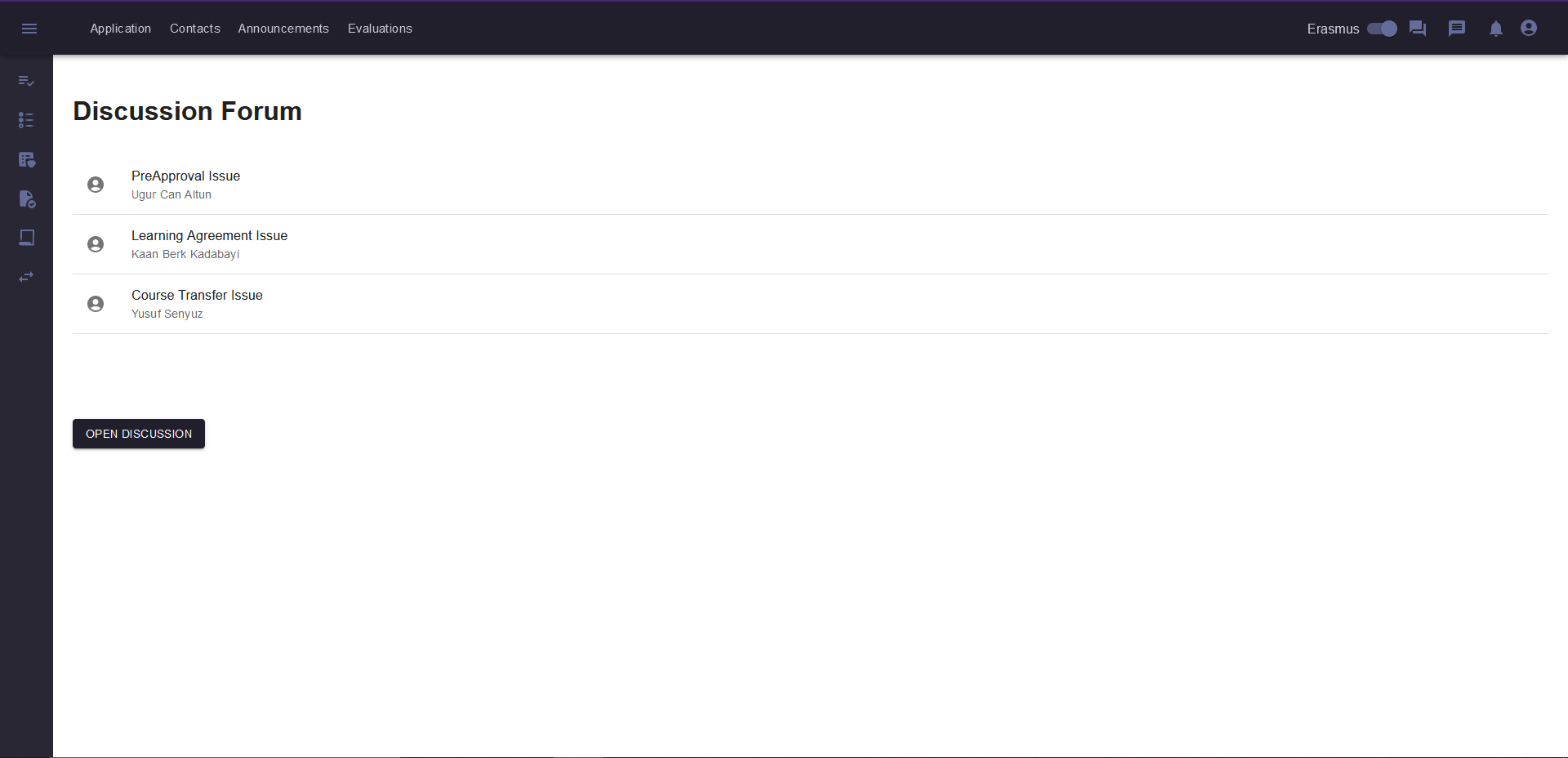


Fig. 21. Forum Screen.

Users can post discussions in the discussion forum to discuss asked questions. Other users can enter replies in the opened discussion threads and reply to each other.

## **3.18. Contacts Page**

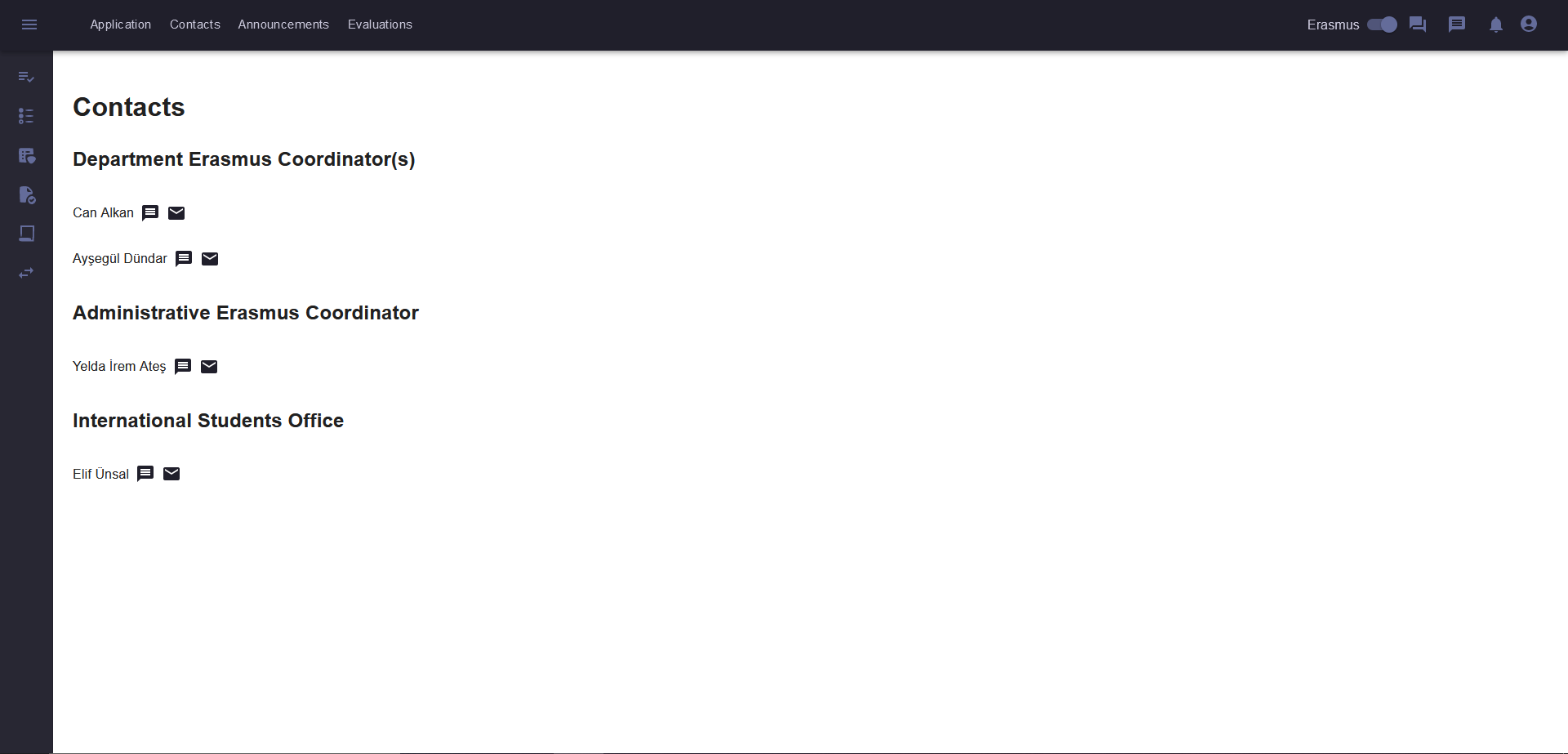


Fig. 22. Contacts Screen for Students.

Students can view and contact with their administrative and department erasmus coordinators in the Contacts page which can be accessed via navbar.

The pages are connected, so one can access every page from the application. However, some of the features do not have any functionality such as messaging since we were unable to connect the backend and frontend parts. Nevertheless, most of the core functionalities are ready to use.

# **B**uildInstructions

Our application is a Spring Boot application, and we used Intellij to run it. To store data we used PostgreSQL. For frontend we used React.

**To run backend:**

1. Download Intellij Ultimate edition.
2. Download PostgreSQL 15 from enterprisedb. Give password to your account ( which is asked to you to finish the building process of PostgreSQL)
3. Clone our code from Github to your computer.
4. After downloading PostgreSQL, access psql from the command line.
   1. If you are using windows write “psql -U postgres” to console and write the password that you gave.
   2. If you are using macOS, write “psql” to the console and write the password that you gave.
5. Then write “CREATE DATABASE erasmusappdb;” to the console. ( Be sure you logged on psql successfully)
6. Then write
   1. If you are using windows write ‘GRANT ALL PRIVILEGES ON DATABASE “erasmusappdb” TO postgres;’ to console.
   2. If you are using macOS, ‘GRANT ALL PRIVILEGES ON DATABASE “erasmusappdb” TO $DefaultUser;’ to console. ( $DefaultUser is the default use of your computer. After you wrote psql in console, psql shows your default user.)
7. Open Intellij, download all necessary files which are shown by Intellij IDEA. Then update the “/resources/application.properties” file.(write your username and password)
8. Click on the “Database” field on the right side of the Intellij idea. ( You have to use the Intellij Ultimate edition. Otherwise you have to follow different procedure which we did not use)
9. Run a program and open “[http://localhost:3000/](http://localhost:8080/)” from your browser.

**To run frontend:**

1. Clone code from github.
2. Write “npm install”.
3. Then write “npm start”.

After these steps the application is working.

# 5. **W**ork **A**llocation

## **5.1. Ahmet Şahin**

Requirements Report:

* Sequence Diagram
* Activity Diagram
* Explanation of Sequence and Activity Diagram
* Explanation of Some Functional Requirements

Design Report:

* Some Final Object Design Explanations
* Some Final Object Design Diagram
* Explanation of Persistent Data Management
* Some Corrections on Subsystem Decomposition
* Formatting

Implementation:

* Worked on Backend
* Database Creation
* Binding Relations Between Classes
* Debug of All Errors
* Rest of the Models, Services, Controllers
* Frontend-Backend Connection via Controllers.
* Security with JWT token and spring security
* Low Level Design
* Changes in Class diagram for Database

## **5.2. Barış Tan Ünal**

Requirements Report:

* Use Case Diagram
* Most of the Textual Use Case Explanations
* Student General State Diagram
* Formatting

Design Report:

* Subsystem Decomposition
* Deployment Diagram
* Web Server Management Layer
* Data Management Layer (Entity Class Diagram)
* Formatting

Implementation:

* Worked on Backend
* Excel Data Fetching Operations
* Placement Related Operations
* Some Entity Classes
* API Testing
* Connection Between Frontend-Backend
* Rest of the Models, Services, Controllers

## **5.3. Kaan Berk Kadabayı**

Requirements Report:

* Class Diagram and Explanations
* Use Case Diagram
* Some Textual Use Cases
* Functional Requirements

Design Report:

* UI Subsystem Decomposition
* UI Subsystem Explanations
* Top Design Goals
* Access Control Security
* Some Final Object Design
* Some Final Object Design Explanations

Implementation:

* Worked on Both Frontend & Backend
* API Connections
* API Testing
* Page Design

## **5.4. Uğur Can Altun**

Requirements Report:

* Non-Functional Requirements
* Pseudo Requirements
* State Diagram for Course Transfer Process and Its Explanation
* Mock-Up Screens for the User Interface

Design Report:

* Subsystem Decomposition’s Explanation
* Hardware/Software Mapping’s Explanation
* Persistent Data Management
* Access Control and Security
* User Permission Table
* Boundary Conditions State Diagram
* Boundary Conditions
* Database Subsystem Diagram

Implementation:

* Worked on Frontend
* API Connections
* API Testing
* Page Design

## **5.5. Yusuf Şenyüz**

Requirements Report:

* Domain Knowledge
* Functional Requirements
* Some Non-Functional Requirements
* Class Diagram
* Some Textual Use Cases
* Formatting

Design Report:

* Purpose of the System
* Design Goals
* Most of the Final Object Design Diagram
* Some Explanations of the Final Object Design Diagram
* Formatting

Implementation:

* Worked on Backend
* Upload-Download Functionality
* PDF Generation
* API Testing
* Rest of the Models, Services, Controllers

# 6. **W**hat **W**e **D**id **A**nd **D**id **N**ot

We Did:

* Login
* Password Encryption
* To-Do List For Tasks
* Checklist
* UI Screens for All User Types
* Score Table Upload
* Excel Fetching Functionality
* Student Placement Functionality
* Placement Management (Deleting & Adding & Waiting Bin Algorithm)
* PDF Generation for Forms
* Wishlist Operations
* Pre-Approval Operations
* Learning Agreement Operations

We Did Not:

* Chat
* Forum
* Evaluation
* Transcript Upload (No Connection Between Backend-Frontend)
* Course Transfer (No Connection Between Backend-Frontend)