Bilkent University

**CS319 Object-Oriented Software Engineering**

**Project Analysis Report Iteration 2**

**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

**11.12.2022**

**CONTENTS**

[**1. Introduction**](#_81d9zq2keefo) **2**

[**2. Domain Knowledge**](#_8eexpf1zlqqi) **3**

[**3. Proposed System**](#_wklzdfyw1o21) **4**

[3.1. Overview](#_1xkb2e3lfdmc) 4

[3.2. Functional Requirements](#_k191alvljp99) 4

[3.2.1. Login](#_odw8if4ulx9w) 5

[3.2.2. Score Table Integration](#_1nk1p1rvwrd7) 5

[3.2.3. Automated Placement](#_ndknr1wubxq6) 5

[3.2.4. Waiting Bin Algorithm](#_g4j53udx2v0r) 6

[3.2.5. Wish List Creation/Approval](#_ctnyot5jo92a) 6

[3.2.6. Pre Approval Form Creation / Approval](#_3t953qieuxyc) 6

[3.2.7. Automated Pre Approval Form - Transcript Matching](#_kctwol9upq85) 7

[3.2.8. To Do List](#_rqqqgmalukny) 7

[3.2.9. Message / Notifications / Announcements](#_gxv8ko50zvkl) 7

[3.3. Non-Functional Requirements](#_fam7pbm0b4cv) 8

[3.3.1. User-Friendly Interface](#_b1g12a4jot0k) 8

[3.3.2. Safety](#_mhglsl8gwatm) 8

[3.3.3. Maintainability](#_2e9h77990927) 9

[3.3.4. Performance](#_kusf3lz8bx06) 9

[3.4. Pseudo Requirements](#_7334hjtf83rw) 9

[3.5. System Models](#_h17hzmf1kl35) 11

[3.5.1. Use Case Model](#_9l03nrm05at8) 11

[3.5.2. Class Diagram](#_b2gn8jtjp91p) 34

[3.5.3. State Diagram](#_9lmn3fum6vpp) 40

[3.5.4. Activity Diagram](#_b2ad3dnuvdfd) 44

[3.5.5. Sequence Diagram](#_9bmytq9agyoc) 47

[3.5.6. User Interface](#_929et21bnn8g) 48

[**4. Improvement Summary**](#_8ifvcz63x1nr) **59**

[4.1. Formatting](#_odl1wk15f2b) 59

[4.2. Introduction](#_bnr8n3kbo2mj) 59

[4.3. Proposed System](#_h5zc1o6tzop9) 59

[4.4. Use Case Model](#_you160eyylmm) 59

[4.4.1. Changes in Use Case Diagram](#_lmhabpy5v07u) 59

[4.4.2. Changes in Textual Use Cases](#_mjtkbb1yw5kv) 60

[4.5. Class Diagram](#_mj65gr54gxfd) 60

[4.6. State Diagrams](#_j3y822ge53xn) 60

[4.6.1. State Diagram of Erasmus Process for Student](#_iv29r1kkxur2) 60

[4.6.2. State Diagram of Course Transfer Process](#_b7auyfqulvo5) 60

[4.7. Activity Diagram](#_5cylujxm3uxe) 60

[4.8. Sequence Diagram](#_9pq3frk41id8) 61

[4.9. User Interface Mock-Ups](#_z484gond565b) 61

[**5. References**](#_lwqpivylgm34) **62**

# **1. Introduction**

In our project, we created a web extension for the Erasmus app called Erasmus Application Manager. Erasmus Application Manager is a complementary system of the current Erasmus App of Bilkent. The web extension has actors: Students, Department Erasmus Coordinators, Administrative Erasmus Coordinators, Course Coordinators, Faculty Administration Committee, International Students Office, and BCC. These actors can use the web extension to fulfill their parts in the Erasmus process. The main aim of this web extension is to decrease the usage of mail and paper during the Erasmus process and make the process smooth for all users. The extension does not contain the application to host universities part of the Erasmus program; instead, our extension uses the “Sample Score Table” given by the International Students Office to place students in their wanted host universities and carry out the following procedures as Pre-Approval and Course Transfer in the following parts of the Erasmus application process.

Our project aims to combine the different parts of the Erasmus procedure and ease the whole process for all users with a semi-automated system. Each student's process will be much smoother by making it easier to track and manage their Erasmus operations for the coordinators. The web extension provides an algorithm to place students to host organizations according to the “Sample Score Table” received by the International Students Office, digitizing a process that was originally done manually. Furthermore, the web extension provides students a wishlist feature to select the courses they want to take abroad and collect the permissions required from course coordinators and department Erasmus coordinators. After the completion of the wishlist with all required permissions, students will be able to prepare their pre-approval forms without errors, again in a digitized platform. Throughout the process, students will be able to track their approval processes, from creating a wishlist for their potential pre-approval forms to the course transfer process. Additionally, the web extension provides a variety of features for coordinators to track and manually interact with the Erasmus process of onboarding students, such as approving their wishlists and pre-approval forms and giving permission to courses that want to be taken instead of elective courses by students. The digitized placement procedure of Erasmus applicants has the following criteria and selection process: students are placed in their preferred universities according to the total points they have. If the students cannot not be placed in their preferred universities, they will be placed in the waiting bin and will be offered a placement to a host university in case of an available quota due to cancellation by placed students.

# **2. Domain Knowledge**

***Department Erasmus Coordinator:*** We attended an info session related to the term project by our customer (Can Alkan) on October 11 Tuesday from 12:30 to 13:20 in Mithat Coruh. He explained the stages of the Erasmus Process and told us what he wants from us. He said that the current Erasmus Process contains lots of paperwork. One of our aims is to reduce those paper works and automate the process as much as possible. Our program tries to make Can Alkan’s and other coordinators’ lives easier.

One important keyword in the domain area is the **ECTS Credits.** ECTS Credits is the abbreviation of the European Credit Transfer and Accumulation System. Each course has corresponding ECTS credits to compare them. Credit scores are correlated with their associated workload. ECTS credits are used to facilitate transfer of courses. 60 ECTS credits are the equivalent of a full year of study or work. For example, CS 319 Object-Oriented Software Engineering course has 6.5 ECTS Credits and ENG 401 Technical Report Writing and Presentation has 5 ECTS Credits. According to Can Alkan, in Pre Approval, if the students want the financial support, total credits of the student must be at least 30 ECTS and cannot exceed by much (32-33 ECTS OK). Some universities have less load, i.e., 1-3 ECTS per course. In that case we might “merge” two or more courses into one. Our program will calculate the total credits and will check such kinds of constraints [1], [2].

***Student:*** One of our project members will go to the *University of L'Aquila* by using the Erasmus program. He is currently in the process of choosing the courses and uploading the Learning Agreement stage. He stated that the most challenging part of the application process is finding the relevant courses to transfer to the host university. Contents of courses in Bilkent and Erasmus universities may differ and it makes finding courses harder. Our program aims to solve this problem by implementing a system that when a student chooses a specific course our system will demonstrate the previously accepted courses in the host university and students can choose those courses easily.

# **3. Proposed System**

## **3.1. Overview**

Our system will have eight user types and personalized user interface and specialized functionalities for each user type. All of the features serve one main purpose which is to simplify the Erasmus process for all of these user types by automating as many parts as a system can. Except the representatives of the host universities, all main stakeholders can have a profile and use Erasmus Application Manager. Even though there are still a few manually done operations during the process, with this system, the use of paper is minimized, if not nullified. No user is required to print a document unless it is mandatory to provide legally issued paper such as the officially signed Pre Approval Form etc. Most of the documents are created by the system by using the information that the user gives by using the user interface. The website welcomes each user with the specialized Todo List which reminds the user to do whatever is required to be done and directs them to the relevant section. Overall, the range of capabilities of the system aims the most optimal way to minimize the drudgery in a secure way.

## **3.2. Functional Requirements**

The system has 8 types of users:

1. Student
2. Department Erasmus Coordinator
3. Administrative Erasmus Coordinator
4. Course Coordinator
5. Department Chair
6. International Student Office
7. Faculty Administration Committee
8. Dean
9. BCC (Admin)

**Features and Functionalities:**

* Home page for Student
* Home page for Course Coordinator/ Department Erasmus Coordinator/International Student Office/Administrative Erasmus Coordinator/Department Chair/Faculty Administration Committee/Dean
* Placement algorithm and WaitingBin algorithm (For placement)
* Ability to prepare WishList and Pre Approval forms (For students)
* Ability to track and interact with the Erasmus process (For coordinators)
* To-do list ( For all types of users)
* Classification and automation of information on courses, credits of courses, and their previous approval status.
* Sending direct messages to other users

### **3.2.1. Login**

Our app ErasmusManager will have a login functionality synchronized with the already-existing Erasmus App app.erasmus.bilkent.edu.tr. To login into our app, the user must be logged in to the Erasmus App and must click on the ErasmusManager link to land on our app. Authentication is done after clicking this link and the user privilege (Department Erasmus Coordinator, Course Coordinator, etc.) will be the same as in the Erasmus App.

### 

### **3.2.2. Score Table Integration**

One of the most important functions of our program is the automated score table integration into the system. Microsoft Excel file of the Score Table is uploaded to our system. The International Students Office user lands on the “Score Table” section to upload the required file. This file contains the data related to the placement of Erasmus. Our system uses this information for placement.

### **3.2.3. Automated Placement**

Having the relevant information about a student in terms of his/her total points and name, our system implements an automated placement system and places every student regarding their university preferences and their total points. After the placements are done, the remaining students who were not placed are put in a Waiting Bin. If an already placed student wants to cancel his/her placement he/she may do so by clicking a Cancel button, his/her Department Erasmus Coordinator is notified.

### **3.2.4. Waiting Bin Algorithm**

In the case of a common problem which is when a placed student cancels his/her mobility, the students waiting in the Waiting Bin must be able to replace the canceling student. Our system holds the students in the Waiting Bin sorted by their total points gathered and when an already placed student cancels his/her mobility, the student on the top of the Waiting Bin is asked to replace the student and is given a deadline to accept or decline this offer. If the answer is positive, he/she is put to the Placement Table and removed from the Waiting Bin and if the answer is negative or the deadline passes, the student is only removed from the Waiting Bin.

### **3.2.5. Wish List Creation/Approval**

In this functionality, students can create their wishlists and submit them to our system. In the home screen of the “student” user, in the Wish Lists section, students can see the tables that represent the courses in the host university and the courses at Bilkent University. Students can add and delete new courses to their wishlists. When the creation of the Wish List is completed, students can submit the Wish List to our system.

After this process, our system checks the validity of the Wish List. In this verification process, our system checks the constraints of the choices. Total credits of the courses must be at least 30 ECTS and cannot exceed by much (32-33 ECTS OK). Another constraint is that students should satisfy each prerequisite of all courses. Students cannot take the courses that they have already taken in Bilkent. If the Wish Lists of students satisfy each condition, they are approved by our system. Students who have approval for their Wish Lists are ready to create their Pre Approval forms.

### **3.2.6. Pre Approval Form Creation / Approval**

This functionality is very similar to the previous Wish List Creation/Approval. This functionality can be accessed by clicking the Pre Approval Form icon located on the home screen of the students. Students can create their Pre Approval forms and submit them to our system. The creation of Pre Approval happens on the Pre Approval page. This process is similar to the creation of the Wish List process. Students can not start the creation of the Pre Approval form process if they did not finish the creation of the Wish List process. Approval of this form is done by the Faculty Administration Committee. The Faculty Administration Committee decides the approval or rejection of the Pre Approval form. After this process, users got a notification of approval status.

### **3.2.7. Automated Pre Approval Form - Transcript Matching**

After the Erasmus semester ends and students return, members of the International Students Office (ISO) must enter each returning student’s transcript into our system in terms of the courses taken, the equivalent courses in Bilkent, and the passing grade for further computation to be done in our system. Since the courses that the student wished to take have been saved in our system and the ISO entered the relevant information, our program is able to automate the process of matching the Pre Approval Form and the Transcript of a returning student. In the case that a student’s transcript does not match to the courses in his/her Pre Approval Form, the courses that are not matched will not be transferred to the student transcript. After the matching is done, the Course Transfer Exemption is ready to be formed and signed by the Department Erasmus Coordinator, the Department Chair, and the Dean.

### **3.2.8. To Do List**

To-do list functionality is present for all types of users. In the “to-do list” section, there are tasks that should be completed by the user. In this functionality, deadlines for each task are given and users can see their tasks with deadlines. These tasks consist of Erasmus application procedures such as creating Wish List and creating Pre Approval. When the user completes the specific task, the corresponding part is marked as done by the system and the user can be sure that that part is completed successfully. This functionality is helpful for especially students because in the Erasmus application process there are a lot of steps that should be completed during a certain time. For instance, the creation of the Wish List and Pre Approval form should be completed during the semester before the mobility starts. Completed tasks are displayed with different styles to demonstrate to the user.

### **3.2.9. Message / Notifications / Announcements**

In our program, all users can send private messages to each other. This functionality allows users to have easier communication. Users can send messages to each other instead of sending emails or if they wish they can send both email and message from our system to have a faster communication opportunity. Notification functionality is not accessible to students, this functionality is created for other kinds of users such as coordinators, the Dean, and the Faculty Administration Committee. When they create announcements for other types of users (especially for students) notification is created and users who should receive that announcement can see its notification in the home page. This functionality is another helping mechanism for the student.

## **3.3. Non-Functional Requirements**

### **3.3.1. User-Friendly Interface**

The user-interface of the web extension must work as intended in all contemporary web browsers that are up-to-date which are the following: Google Chrome version 107, Mozilla Firefox 106, Safari version 16.1, Microsoft Edge version 107 and Opera version 92. Users must see all the images that are specified in the source code while viewing the user interface from their respective browsers. Furthermore, all links must be available and all buttons must perform their respective functionalities when they are clicked. There must be no disabled text fields in a scenario where they must be available. All buttons, texts must be aligned and in the proper place just as they are specified in the mock-up templates.The user interface must be responsive, meaning that neither images nor content must disappear or overlap when resizing is done by the user. When the user shrinks the page of the web extension, all of the content must shrink corresponding to the shrinking rate and must stay inside the frame if it was inside the frame as well before shrinking. On the other hand, all of the content must enlarge when the user enlarges the frame of the web extension and elements outside the frame must not enter thet frame after the enlargement operation. The web extension must not contain any typographical errors. Finally, images inside the web extension must be 400-600 pixels for a large image and 100-200 pixels for a thumbnail image. The resolution of the images must be 72 DPI and the preferred file format of the images must be JPEG. The approximated file size of the images must be between 20-200 kB.

### **3.3.2. Safety**

All users can access only their private profiles and it prevents unauthorized access to private information such as sex, nationality, and ID of the student. We also want to integrate our project into the current Erasmus App of Bilkent so that our system will be as safe as the current Erasmus App. Before some users login to our system, they must login to the STARS system and it increases the overall safety of our program.

### **3.3.3. Maintainability**

In our project, we have used a lot of classes and as a result of this, modularity is increased. So, adding a feature, removing a feature, or changing a feature is easy. When we want to change some detail of the implementation we can simply change the necessary classes without causing any other problems in other classes. Ease of maintainability and handling changes is provided by the practices of object-oriented programming. To measure and ensure that the application exceeds the desired maintainability level, developers record the time taken to implement or change a functionality or a concept in the application and check whether the time is under 4 hours. Furthermore, software architects of the application will develop a class diagram with low coupling and high cohesion and developers will verify that object-oriented programming, SOLID and DRY principles are followed during the implementation stage to ensure the desired maintainability. Developers will avoid code smells and technical debts while developing the software which are the following: lack of documented code, not using up-to-date tools, excessively long functions, duplicated code, long parameter lists and inconvenient names. All code written will be documented by the developers and tools that will be used will be up-to-date. No functions will be longer than 100 lines of code to avoid having excessively long functions. Number of parameters in a function will not exceed 6 parameters to avoid excessively long parameter lists. Finally, intuitive names will be given to fields and operations within the code and camel case convention will be followed throughout the code to ensure code consistency.

### **3.3.4. Performance**

Users must login to the application within at most 5 seconds. Each page must load within at most 2 seconds. The placement algorithm must finish within at most 1 hour so placement data is available 1 hour later the score Table is uploaded. The website should be compatible with Google Chrome, and Mozilla Firefox.

## **3.4. Pseudo Requirements**

* The programming language must support Object Oriented Programming (OOP).

## 

## **3.5. System Models**

### **3.5.1. Use Case Model**

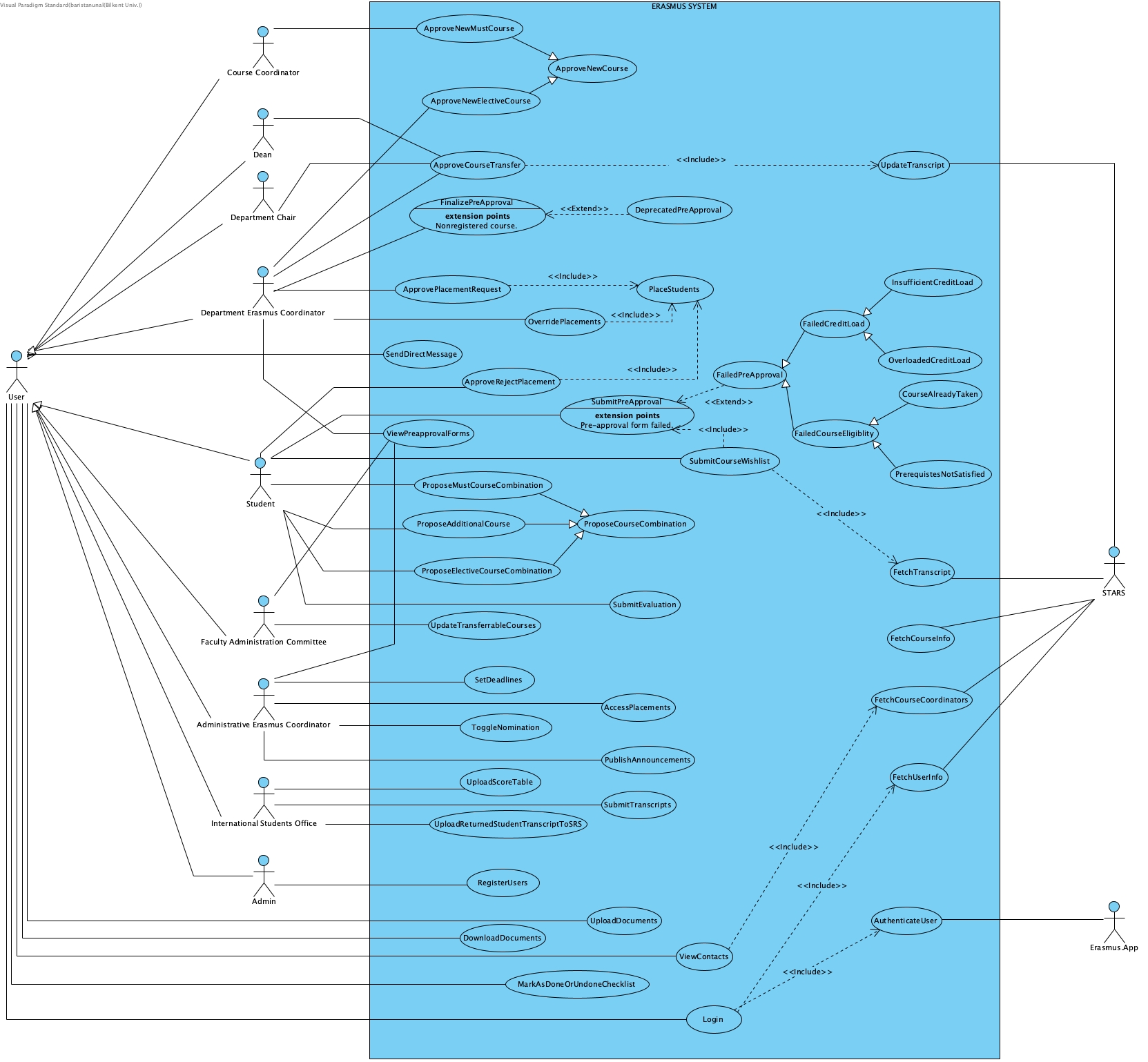


Fig. 1. Use Case Diagram

**Use Case Name:** Authenticate User

**Participating Actors:** Erasmus App, User

**Entry Conditions:**

* The user clicks the link to the system located on the headline of the Erasmus App.

**Exit Conditions:**

* The user is successfully logged into the system.

**Flow of Events:**

1. The user is logged into the already existing Erasmus App of Bilkent University.
2. The user wishes to log into the extension system.

**Quality Requirements:**

* The user should be logged in to the app with the same user privileges.

**Use Case Name:** Upload Score Table

**Participating Actors:** International Students Office

**Entry Conditions:**

* The application period for Erasmus is past due (not directly related to our system).

**Exit Conditions:**

* The user successfully uploads the table and the system successfully parses it.

*OR*

* The user uploads an invalid document and receives an error message.

**Flow of Events:**

1. The user from the International Students Office clicks the “Score Table” section from the sidebar.
2. The system asks for a single Microsoft Excel (.xlsx) file as input.
3. The user uploads a Microsoft Excel file that consists of all the applicants and their relevant information.
4. The system parses the Excel file and writes the relevant information to the database for further use.
5. The system displays a message that indicates whether the file is successfully uploaded and accepted or rejected with its cause of error such as incompatible file extension or invalid table type etc.

**Quality Requirements:**

* The upload time should not take longer than 30 seconds.

**Use Case Name:** Place Students

**Participating Actors:** N/A

**Entry Conditions:**

* The International Students Office has uploaded a valid Score Table.

**Exit Conditions:**

* The algorithm places students in universities and places the rest in the waiting box.

**Flow of Events:**

1. The system gets the placement data (student name, student ID, score, etc.) of each student from the database and sorts them according to their Erasmus score.
2. Starting from the first-placed student, the system places the students to their most preferred university with available quota until there are no students left on the list or all of the universities are full.
3. The system creates a “waiting bin” which consists of students who cannot be placed due to quota limitations from highest-scored student to the lowest.

**Quality Requirements:**

* The algorithm should function properly and be finished in 15 seconds without any errors.

**Use Case Name:** Approve/Reject Placement

**Participating Actors:** Student

**Entry Conditions:**

**{**

* The student is placed in a certain university after the algorithm of Place Student selects it and assigns it to the host university according to the Score Table.

*OR*

* The student who was not placed to a certain university after the Place Students algorithm is selected from the waiting bin when another student who had a placement cancels his Erasmus placement.

}

AND

* Deadline for placements has not passed.

**Exit Conditions:**

* The student approves (accepts) its placement.

*OR*

* The student cancels (rejects) its placement.

OR

* Department Erasmus Coordinator failed to approve/reject the placement before deadline ends

**Flow of Events:**

1. The system sends the information of approval to the Student.
2. The student makes his decision of approval or rejection by clicking the corresponding button.

**Quality Requirements:**

* N/A

**Use Case Name:** Approve Placement Request

**Participating Actors:** Department Erasmus Coordinator

**Entry Conditions:**

* A student withdraws their Erasmus application.

*AND*

* Another student in the waiting bin accepts to take the vacancy.

AND

* Deadline for vacancy approval is not due

**Exit Conditions:**

* The student either takes the previous student’s place or is kept in the waiting bin, depending on the response of the Department Erasmus Coordinator.

*OR*

* The vacancy period (the number of days each student is given to accept/decline placement) is past due before the Department Erasmus Coordinator responds to the request and the request is assumed to be approved to prevent any grievance of the student who is willing to take the vacancy.

*OR*

* The placement period is past due before the Department Erasmus Coordinator responds to the request and the request is assumed to be approved to prevent any grievance of the student who is willing to take the vacancy.

**Flow of Events:**

1. The Department Erasmus Coordinator is asked whether they approve the student’s request of filling a vacancy of another student who canceled their Erasmus placement.
2. The Department Erasmus Coordinator either approves or declines the student’s request.
3. When the vacancy/mobility period is past due, the Department Erasmus Coordinator is assumed to approve the placement request.

**Quality Requirements:**

* N/A

**Use Case Name:** Override Placements

**Participating Actors:** Department Erasmus Coordinator

**Entry Conditions:**

* There is a situation that requires the placement to be overridden such as transfer of the student to another university or the death of the student.

**Exit Conditions:**

* A placement of a student is canceled and that student is kicked to the waiting bin and a vacancy occurs.

*OR*

* A placement of a student is canceled and another student is chosen manually by the Department Erasmus Coordinator to fill the vacancy.

*OR*

* No override was made.

**Flow of Events:**

1. The Department Erasmus Coordinator is notified of the situation that requires the placement to be overridden and The Department Erasmus Coordinator views it.
2. The Department Erasmus Coordinator overrides (modifies) placement results by canceling or altering a placement.
3. The Department Erasmus Coordinator can place new students in case of requirement of invalidation of some placements (For example, if the placed student transfers to another university, the Department Erasmus Coordinator can override the placement).
4. The Department Erasmus Coordinator finishes making changes and submits the altered placement results.
5. Students are notified of their placement result.

**Quality Requirements:**

* The Department Erasmus Coordinator should finalize its initial overrides before the deadline which Administrative Erasmus Coordinator sets beforehand.
* No changes can be made after the placement period is past due.

**Use Case Name:** Submit Course Wishlist

**Participating Actors:** Student

**Entry Conditions:**

* The placement period is completed and the student is placed at any host university.

**Exit Conditions:**

* The status of each course in the Wishlist is “ready to submit” which means the wishlist is valid.

*AND*

* The total ECTS credits of the accepted courses are above the bare minimum limit at the end of the Wishlist period or when the student submits their Wishlist, whichever is first.

**Flow of Events:**

1. The student opens the Wishlist section.
2. The student clicks the “add new course” button and adds the courses as they wish as it is stated in the “Propose New Course” use case.
3. The student submits their Wishlist.

**Quality Requirements:**

* This operation can only be done during the Wishlist preparation period which is after the placement period and before the Pre Approval period.
* If the student fails to submit a valid Wishlist, the Erasmus application of the student gets automatically canceled.

**Use Case Name:** Propose Course Combination

**Participating Actors:** Student

**Entry Conditions:**

* The student is placed at a university at the end of the placement period.

*AND*

* The student has not exceeded the maximum credit load limit.

*AND*

* The wishlist period is not past due.

**Exit Conditions:**

* The student has typed a valid (alphanumeric characters without spaces) course code and ECTS credits (float number).

*AND*

* The course is successfully added to the Wish List.

**Flow of Events:**

1. The student selects the type of the new course which can be taken as either a mandatory course, an elective course, or an additional course.
2. The student fills the blanks with appropriate information (which are explained in detail in type-specific use cases below).
3. The student submits their request.
4. The request is added to the Wishlist page of the student with its status which can be one of previously accepted, new request accepted, no approval needed, pending, or rejected.

**Quality Requirements:**

* N/A

**Use Case Name:** Propose Must Course Combination

**Participating Actors:** Student

**Entry Conditions:**

* There is a mandatory course left in the curriculum of the student.

*AND*

* The student has selected the “Must Course” option from the “Add Course” dialog.

**Exit Conditions:**

* The course combination is ready to be submitted to the Wish List.

**Flow of Events:**

1. The student selects whether they would like to use a previously   
 accepted combination of a host university course - Bilkent course or   
 propose a new equivalent course from their host university to a Bilkent   
 course.

2. The student selects whether to use a previously accepted combination or a new   
 combination.

2.1 If the student selects to use a previously accepted combination,

2.1.1. The student selects the host university and Bilkent course codes from   
 the limited number of choices.

2.2. If the student selects to propose a new combination,

2.2.1 The student types the details which consist of the course code, course   
 name, and ECTS credits of the host university course that they want to   
 take and transfer.

2.2.2. The student selects a Bilkent course from their curriculum that is   
 equivalent to the host university course.

2.2.3. The student uploads the syllabus of the host university course.

2.2.4. The student uploads their intent to take that course.

**Quality Requirements:**

* N/A

**Use Case Name:** Propose Elective Course Combination

**Participating Actors:** Student

**Entry Conditions:**

* The student has selected the “Elective Course” option from the “Add Course” dialog.

*AND*

* There is an elective course (of any type) left in the curriculum of the student.

**Exit Conditions:**

* The course combination is ready to be submitted to the Wish List.

**Flow of Events:**

1. The student selects whether they would like to use a previously   
 accepted combination of a host university course - Bilkent course or   
 propose a new equivalent course from their host university to a Bilkent course.

2. Student selects whether to use a previously accepted combination or a new   
 combination.

2.1 If the student selects to use a previously accepted combination,

2.2.1. The student selects the host university and Bilkent course codes from   
 the limited number of choices depending on the elective type that they   
 selected (Social Science Elective, Technical Elective, Breadth Elective   
 etc.) which also depends on their department.

2.2. If the student has selected to propose a new combination,

2.2.1. The student types the details which consist of the course code, course   
 name, and ECTS credits of the host university course that they want to  
 take and transfer.

2.2.2. The student selects the elective type from their Bilkent curriculum.

2.2.3. The student uploads the syllabus of the host university course.

**Quality Requirements:**

* N/A

**Use Case Name:** Propose Additional Course

**Participating Actors:** Student

**Entry Conditions:**

* The student has selected the “Additional Course” option from the “Add Course” dialog.

**Exit Conditions:**

* The additional course is ready to be submitted to the Wish List.

**Flow of Events:**

1. The student specifies only the host university course code and credit load.

**Quality Requirements:**

* N/A

**Use Case Name:** Update Transferable Courses

**Participating Actors:** Faculty Administration Committee

**Entry Conditions:**

* There is at least one transferable course for deletion.
* At least one Department Erasmus Coordinator should have accepted a host university course as equivalent to a Bilkent course for accepting a pending request.

**Exit Conditions:**

* A new course appears on the transferable courses list.

*OR*

* A course was deleted from the transferable courses list.

**Flow of Events:**

1. The user from the Faculty Administration Committee views the transferable courses from its section.
2. All requests approved by the Department Erasmus Coordinators are placed on the pending requests list. The user from the Faculty Administration Committee can approve or reject that request. If they approve, that becomes visible in the transferable courses list.
3. The user from the Faculty Administration Committee may remove a previously accepted course combination from the list.
4. The user from the Faculty Administration Committee may manually add a new course combination.

**Quality Requirements:**

* N/A

**Use Case Name:** Toggle Nomination

**Participating Actors:** Administrative Erasmus Coordinator

**Entry Conditions:**

* A student cancels their Erasmus placement after the nominations are made.

*AND*

* Another student sends a request to fill the vacancy.

*AND*

* It is still possible to nominate another student.

**Exit Conditions:**

* A new student gets nominated to the host university if possible.

*OR*

* A quota for a university remains unfilled, therefore one less student is nominated.

**Flow of Events:**

1. The Administrative Erasmus Coordinator adds the student who requested to take the other one’s place manually from the waiting bin.
2. The student, who is placed manually to a university, gets notified.

**Quality Requirements:**

* N/A

**Use Case Name:** Submit Pre Approval

**Participating Actors:** Student

**Entry Conditions:**

* The Student’s Course Wish List is a valid Wish List (the ECTS credit load is neither underloaded nor overloaded).

*AND*

* The Student’s Course Wish List only consists of courses that are ‘approved’.

**Exit Conditions:**

* The Student successfully submits the Pre Approval Form which contains a valid Course Wish List.

**Flow of Events:**

1. The student checks and submits their Pre Approval Form which is created automatically by the system once the conditions hold.

**Quality Requirements:**

* The Pre Approval Form should be downloadable in DOCX and PDF format.

**Use Case Name:** Finalize Pre Approval

**Participating Actors:** Department Erasmus Coordinator

**Entry Conditions:**

* There is at least one Student whose Pre Approval has not been approved/finalized by the Department Erasmus Coordinator

**Exit Conditions:**

* The Department Erasmus Coordinator successfully finalizes the Student’s Pre Approval Form and this form is saved to the system for later use.

**Flow of Events:**

1. The Student is placed into his/her university of preference.
2. The Student successfully creates a valid Course Wish List.
3. The Student successfully converts the valid Course Wish List into a valid Pre Approval Form

**Quality Requirements:**

* Only a valid Pre Approval Form should be viewed by the Department Erasmus Coordinator.

**Use Case Name:** Send Direct Message

**Participating Actors:** Any User

**Entry Conditions:**

* The user clicks the Messages tab.

*AND*

* The user has a message that can be sent. (No empty message)

**Exit Conditions:**

* The user completes the message.  
  *AND*
* The user clicks Send.

**Flow of Events:**

1. The user wants to reach any user in the program.
2. The user goes into his/her Direct Messages tab.
3. The user finds the other User he/she wishes to reach and types his/her message.

**Quality Requirements:**

* Direct messages sent to a user should appear on the Direct Messages tab to the received user.

**Use Case Name:** Fetch Transcript

**Participating Actors:** STARS

**Entry Conditions:**

* Placements are done.

*AND*

* The student is ready to form a Course Wish List.

**Exit Conditions:**

* The taken and remaining courses of each student are obtained which will be used in the preparation of the Wish Lists by the students.

**Flow of Events:**

1. The system sends a request to the STARS database.
2. The courses taken by the student and the courses that student can take in the future are fetched successfully from the curriculum page of the SRS.

**Quality Requirements:**

* Fetch operation should not take more than 30 seconds.

**Use Case Name:** Submit Transcript

**Participating Actors:** STARS

**Entry Conditions:**

* The transferable courses of a Student after he/she returns are present in the system.

**Exit Conditions:**

* The courses which are present in the Pre Approval Form and have a grade of at least C are transferred to the Student’s transcript.

**Flow of Events:**

1. A Student ends his Erasmus Programme and returns to his/her university.
2. The Student’s Course Transfer Exemption Form is formed by the system.
3. The list of transferable courses is determined by the system.

**Quality Requirements:**

* The courses which are transferred should be visible in the Student’s transcript tab in STARS.

**Use Case Name:** Mark Task As Done

**Participating Actors:** Any User

**Flow of Events:**

1. The user completes the task that was assigned to them such as the creation of a Wish List or the creation of a Pre Approval Form.
2. The system detects the completion of a particular task.
3. The system marks the task in the to-do list as done and corresponding user can see that task as marked/completed in their Todo List.

**Entry Conditions:**

* Task that can be completed is created.

*AND*

* Tasks should be placed in the to do list of the user(s).

**Exit Conditions:**

* Task is marked as done

**Quality Requirements:**

* Tasks that will be marked as done should be marked in 1 second (at most) due to avoid any potential problems related to tasks.

**Use Case Name:** Download Documents

**Participating Actors:** Any User

**Entry Conditions:**

* An Executive Coordinator (Department Erasmus Coordinator, International Students Office, Administrative Erasmus Coordinator, Faculty Administration Committee, Dean, Department Chair) needs to download a document such as the Pre Approval Form or the Course Transfer Exemption Form to their computer for formality purposes.

*OR*

* A student needs to download the Learning Agreement Form to fill it out.

**Exit Conditions:**

* The specified document is downloaded to the User’s computer

**Flow of Events:**

1. A user needs to download a document
2. The user specifies a file type for the document
3. Clicks the download link

**Quality Requirements:**

* The document should be downloaded as the specified file type: PDF or DOCX.

**Use Case Name:** Publish Announcements

**Participating Actors:** Administrative Erasmus Coordinator

**Entry Conditions:**

* The Administrative Erasmus Coordinator is on the Publish Announcements tab.

**Exit Conditions:**

* An announcement is successfully made to the desired user types.

**Flow of Events:**

1. The Administrative Erasmus Coordinator needs to make an announcement for the students or the other users such as incoming deadlines or changing deadlines.
2. The Administrative Erasmus Coordinator lands into the Publish Announcements tab in the system.
3. The Administrative Erasmus Coordinator types the desired announcement to be published.
4. The Administrative Erasmus Coordinator checks the desired user types who should see the announcement.

**Quality Requirements:**

* N/A

**Use Case Name:** Submit Evaluation

**Participating Actors:** Student

**Entry Conditions:**

* A student clicks the Evaluation tab

*AND*

* A student types the desired evaluation.

**Exit Conditions:**

* A student completes the evaluation.

*AND*

* A Student clicks the submit button.

**Flow of Events:**

1. Student completes his/her Erasmus process.
2. After the return of the student, the Submit Evaluation section becomes active in the system.
3. The student goes into his/her Submit Evaluation tab.
4. The student writes his/her evaluation of the host university and submits it.

**Quality Requirements:**

* Evaluations submitted by students should appear on the Evaluations tab on the system.

**Use Case Name:** Access Placements

**Participating Actors:** Administrative Erasmus Coordinator

**Entry Conditions:**

* The Administrative Erasmus Coordinator lands on the Placements tab on the sidebar.

**Exit Conditions:**

* The Administrative Erasmus Coordinator views the placements.

*OR*

* The Administrative Erasmus Coordinator successfully overrides the placements.

**Flow of Events:**

1. The International Students Office creates the Score Table and uploads it to the system.
2. The system makes the placements according to each Student’s preference and total points.
3. The Administrative Erasmus Coordinator overrides the placements if necessary and submits them.

**Quality Requirements:**

* The Administrative Erasmus Coordinator should be able to override the placements until the deadline.

**Use Case Name:** Submit Transcripts

**Participating Actors:** International Student Office

**Entry Conditions:**

* Students complete the Erasmus Programme

*AND*

* The International Students Office user receives the returned Student’s Transcript via email.

**Exit Conditions:**

* The student's data contained in the Transcript is serialized to be used later for creating the Course Transfer Exemption Form.

**Flow of Events:**

1. A Student completes the Erasmus Programme and returns to his/her university
2. A representative from the host university sends the Student’s transcript for the semester to the International Student Office via email.
3. The International Student Office user needs to submit the returned Student’s Transcript into the system and land on the tab.

**Quality Requirements:**

* The data written to the system should be in a standardized format for optimized serialization.

**Use Case Name:** Update Transcript

**Participating Actors:** STARS

**Flow of Events:**

1. A Student completes the Erasmus Programme and returns to his/her university.
2. The International Student Office user submits the returned Student’s Transcript.
3. The system serializes the Transcript for Transcript/Pre Approval Form matching.
4. A list of transferable courses is obtained.

**Entry Conditions:**

* The Student has at least one course that is transferable to his/her Transcript in SRS.

**Exit Conditions:**

* The courses which have a passing grade and are matching with what is written on the Pre Approval Form are successfully transferred to the Student’s Transcript.

**Quality Requirements:**

* The courses deemed to be transferable should immediately be seen in the Student’s SRS system.

**Use Case Name:** Register Users

**Participating Actors:** Admin

**Flow of Events:**

1. Admin user views the list of users.
2. Admin user adds a new user by specifying mail, password and role.

**Entry Conditions:**

* N/A

**Exit Conditions:**

* A new user is added.

**Quality Requirements:**

* N/A

**Use Case Name:** Approve New Course

**Participating Actors:** Course Coordinator, Department Erasmus Coordinator

**Entry Conditions:**

* Student proposed a new course which is a must or elective course.

**Exit Conditions:**

* The offer is approved by the relevant coordinator(s).

*OR*

* The offer is rejected by the relevant coordinator(s).

**Flow of Events:**

1. The appropriate coordinator (depending on the type of the course) downloads the syllabus and the intent.
2. The appropriate coordinators approve or reject the proposal.
3. The student gets notified of the decision.

**Quality Requirements:**

* The approval process should be completed before the deadline - which is determined by the Administrative Erasmus Coordinator - of the approval process.

**Use Case Name:** Approve New Must Course

**Participating Actors:** Course Coordinator

**Entry Conditions:**

* The student proposed a new must course.

**Exit Conditions:**

* The offer of a student is accepted by both the Course Coordinator and Department Erasmus Coordinator.

*OR*

* The offer is rejected by the Course Coordinator.

*OR*

* The offer is approved by the Course Coordinator but gets rejected by the Department Erasmus Coordinator.

**Flow of Events:**

1. The Course Coordinator of that course downloads the syllabus and intent which are uploaded by the student.
2. The Course Coordinator evaluates the course pair and determines whether it is valid or not valid.
3. The Course Coordinator approves the proposal or rejects it.

**Quality Requirements:**

* The approval process should be completed before the deadline - which is determined by the Administrative Erasmus Coordinator - of the approval process.

**Use Case Name:** Approve New Elective Course

**Participating Actors:** Department Erasmus Coordinator

**Entry Conditions:**

* Student proposed a new elective course.

**Exit Conditions:**

* The offer of a student is accepted by the Department Erasmus Coordinator.

*OR*

* The offer of a student is rejected by the Department Erasmus Coordinator

**Flow of Events:**

1. The Department Erasmus Coordinator downloads the syllabus and intent which are uploaded by the student.
2. The Department Erasmus Coordinator evaluates the course pair and determines whether it is valid or not valid.
3. The Department Erasmus Coordinator approves the proposal or rejects it.

**Quality Requirements:**

* The approval process should be completed before the deadline - which is determined by the Administrative Erasmus Coordinator - of the approval process.

**Use Case Name:** View Pre Approval Forms

**Participating Actors:** Faculty Administration Committee, Administrative Erasmus Coordinator

**Entry Conditions:**

* Pre Approval of students is created by the students (Pre Approval Exist).

**Exit Conditions:**

* The Faculty Administration Committee or Administrative Erasmus Coordinator successfully accesses the Pre Approval and views it.

*OR*

* The Faculty Administration Committee or Administrative Erasmus Coordinator can not access the Pre Approval due to an error such as the student did not create the Pre Approval form yet.

**Flow of Events:**

1. Student creates Pre Approval form
2. That Pre Approval form is stored in the system
3. Faculty Administration Committee or Administrative Erasmus Coordinator can view the Pre Approval Form when needed such as comparing them with the official transcript sent by the host university

**Quality Requirements:**

* The viewing of the Pre Approval form should occur in 2 seconds (at most) by the system to avoid any problem with the course transfer.

**Use Case Name:** Set Deadlines

**Participating Actors:** Administrative Erasmus Coordinator

**Entry Conditions:**

* The Administrative Erasmus Coordinator goes to the Set Deadline Tab.

**Exit Conditions:**

* The Administrative Erasmus Coordinator selects the valid time and deadline changes.

*OR*

* The Administrative Erasmus Coordinator does not specify the valid time and an error occurs.

**Flow of Events:**

1. The Administrative Erasmus Coordinator goes to the “Set Deadlines Tab”.
   1. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Placement Period”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for the placement period and confirms it.
   2. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Wishlist Period”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for the wishlist period and confirms it.
   3. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Pre Approval Period”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for Pre Approval period and confirms it.
   4. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Time to Accept to Take Vacancy for Student”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for time to accept to take a vacancy for students and confirms it.
   5. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Time to Approve Nomination for Student”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for time to Approve Nomination for a student and confirms it.
   6. If the Administrative Erasmus Coordinator selects the option “Set Deadline for Time to Approve New Course Combination Request”.
      1. The Administrative Erasmus Coordinator specifies the specific deadline for time to approve a new course combination request and confirms it.
2. The system writes the deadline information that is submitted by the Administrative Erasmus Coordinator to the database for further use.

**Quality Requirements:**

* The Administrative Erasmus Coordinator should do at least one deadline change.

**Use Case Name:** Approve Course Transfer

**Participating Actors:** Dean, Department Chair

**Entry Conditions:**

* The host university of the student sends an official transcript to the International Student Office

*AND*

* The Department Course Coordinator creates a course transfer form.

**Exit Conditions:**

* Course transfer is rejected because the Dean does not approve it.

*OR*

* Course transfer is rejected because the Department Chair does not approve it.

*OR*

* Course transfer is approved because both the Department Chair and the Dean approve it.

**Flow of Events:**

1. Student completes his/her Erasmus process.
2. Host university of student sends an official transcript to the International Student Office
3. The Department Course Coordinator compares the Pre Approval form with the official transcript.
4. The Department Course Coordinator includes the matching courses in the course transfer form.
5. This transfer form becomes visible to the Dean and Department Chair and they finalize the course transfer by approving it.

**Quality Requirements:**

* The decision of The Department Chair and the Dean should be detected in 10 seconds (at most) by the system to avoid any problem with the course transfer.

### **3.5.2. Class Diagram**

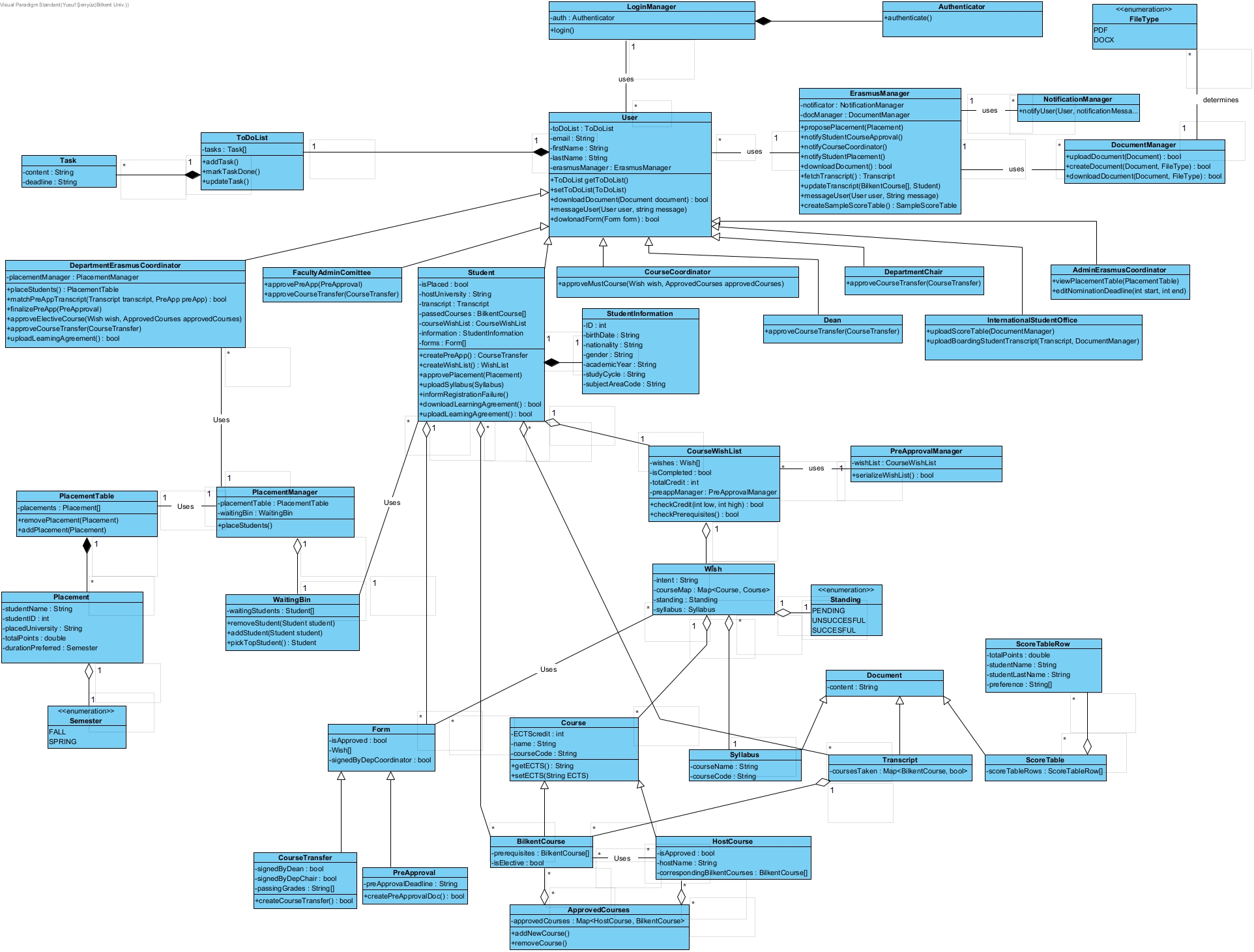


Fig. 2. Class Diagram

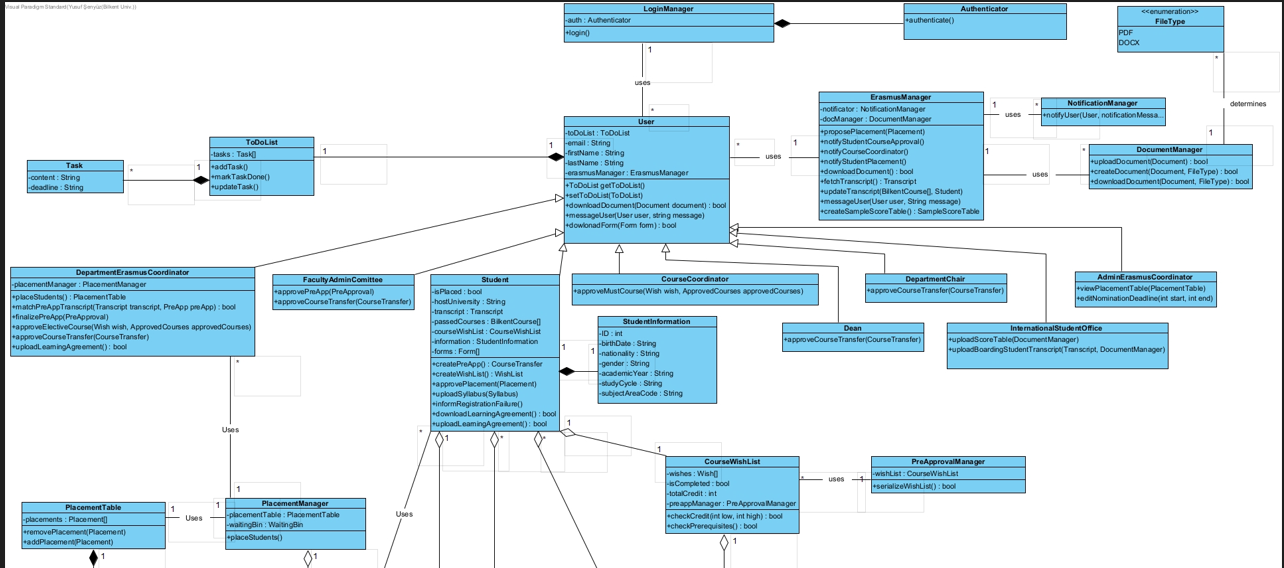


Fig. 2.1. Class Diagram Part 1

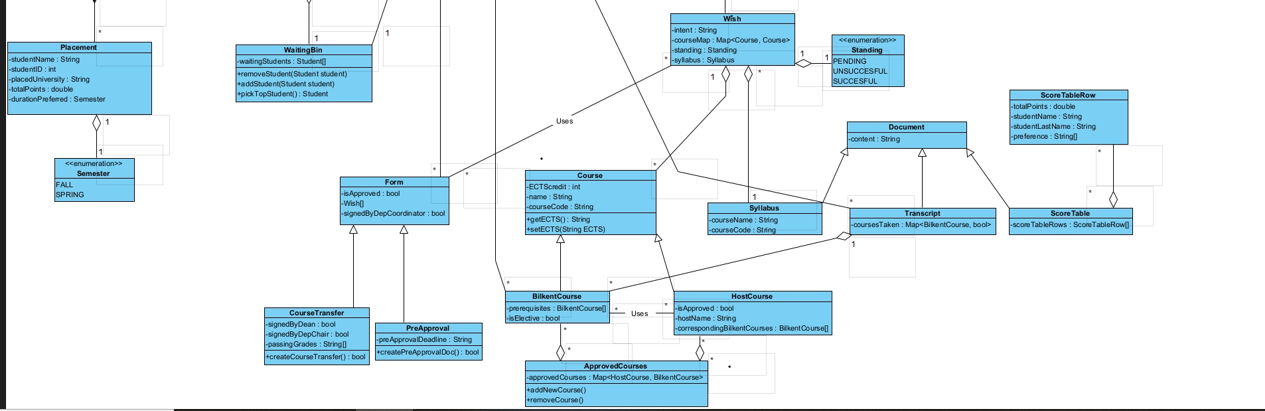


Fig. 2.2. Class Diagram Part 2

**User:** User is an abstract class that captures the common behavior of the stakeholders of our software. Every user has a to-do list on which they can add tasks, remove tasks, every user may download forms (Pre Approval Form, Course Transfer Exemption Form), every user may download documents (Transcript, Score Table, etc.), and these functionalities are mostly performed with the help of the ErasmusManager class.

**Department Erasmus Coordinator:** DepartmentErasmusCoordinator is a child class of the User class and is one of the major stakeholders in the program. They are the department-specific coordinators and in the case of the CS department, they are Ayşegül Dündar and Can Alkan. This class is responsible for many form approvals, transcript/Pre Approval matching for returning students as well as placing outgoing students to the host universities. They can override the student placement. This class uses the PlacementManager class to handle complex placement algorithms.

**FacultyAdminCommittee:** FacultyAdminCommittee is a child class of the User class and is one of the minor stakeholders. They can approve Pre Approval Forms, and Course Transfer Exemption Forms and on a monthly basis, they can check out the approval of the new courses.

**Student:** The student is a child class of the User class and is one of the major stakeholders in this program. This class contains critical information for the forms about a student such as a name, host university, academic year, and so on. Students are required to submit a wishlist of classes they want to take once they are placed at a university. Their wish list contains the classes they want to take in the upcoming semester, the student's intent of taking this course, and the corresponding course given at Bilkent University. If the course hasn’t been approved yet, the student is required to upload the syllabi of those classes. If the registration after the mobility is somehow failed, the student may inform the coordinator.

**CourseCoordinator:** The CourseCoordinator is one of the stakeholders in the program which is important for approving new must courses. When a student wants to enroll in a course that is not in the approved courses database, the corresponding Course Coordinator is notified to approve or disapprove the course by looking at the syllabus and intent of a given student.

**Dean:** Dean is a child class of the User class and is one of the minor stakeholders in the program. They are only relevant for approving (signing) the Course Transfer Exemption Form in the process.

**DepartmentChair:** DepartmentChair is a child class of the User class and is one of the minor stakeholders in the program. They are only relevant for approving (signing) the Course Transfer Exemption Form in the process.

**AdminErasmusCoordinator:** AdminErasmusCoordinator is a child class of the User class and is a stakeholder in the program. They can view or override the student placements and they are responsible for handling most deadlines in the process.

**InternationalStudentsOffice:** InternationalStudentsOffice or the ISO is a child class of the User class and is one of the stakeholders in the program. They are responsible for uploading the Score Table which contains a table of the students who applied to Erasmus and their respective scores. The ISO is also responsible to upload each returning student’s transcript to the system for Pre-Approval/Transcript matching.

**LoginManager:** LoginManager is a container class to make the authentication process abstract and usable by the User class without any problems. The class is composed of the Authenticator class and has only the function of logging in.

**Authenticator:** Authenticator class is used by the LoginManager class and is useful for handling the authentication process when the link to our program is clicked. The Authenticator class makes use of different web services and user-specific data such as cookies and sessions to confirm and authenticate every user.

**ErasmusManager:** This class is used by the User class and encapsulates the required external functionalities such as notifying users, messaging between users, downloading documents, and all other similar behaviors that should be abstracted out from the other classes.

**NotificationManager:** This class is used by the ErasmusManager class and is used solely for the purpose of sending notifications to a user from a user. It serves as a layer of abstraction for the notification-sending logic to increase maintainability.

**FileType:** This class is an enumerable class and is useful for handling the problem of downloading documents in different formats. These different formats are PDF and DOCX format.

**Form:** Form class is an abstract class to capture the similar behavior of the Pre Approval Forms and the Course Transfer Exemption Forms. These mutual behaviors are whether the Form is approved to be processed, whether it is signed by the Department Erasmus Coordinator, (both of the forms must be signed by the Department Erasmus Coordinator), and the Wishes representing the rows in these forms.

**PreApproval:** PreApproval class is a child class of the Form class. It contains the relevant information of a Form as well as the deadline for the Pre Approval Form creation. It has an operation and it is downloading the Pre Approval Form in document format.

**CouseTransfer:** CourseTransfer class is a child class of the Form class. It contains the relevant information found in Forms as well as the information on whether the form has been signed by the dean and the department chair as well as the passing grades. It has an operation and it is downloading the Course Transfer Exemption Form in document format.

**ToDoList:** ToDoList class aggregates the Task class and contains a list of Tasks. This list can be modified by adding or marking tasks done. Every user contains the tasks in the To Do section.

**Task:** A task contains the content of the task and the deadline to complete the task. Every User class can see these tasks listed on his/her ToDoList. These tasks are added automatically to the ToDoList and when the User completes the task they are removed from the list.

**CourseWishList:** CourseWishList aggregates the Wish class and contains a list of Wishes. A valid Wish List must be submitted to create a Pre Approval Form. To submit a Wish List, some validations such as checking prerequisites, and checking if the ECTS credit is enough or it is overloaded. To do these comparisons, CourseWishList uses the PreAppManager class.

**Wish:** The wish is one of the 5-6 (maybe more, depending on the ECTS credit load), wishes a student must submit before forming a valid Pre Approval Form. A wish contains the class student wishes to take in the host university, the corresponding Bilkent course to transfer it, the intent of taking this course, and the syllabus of the course in the host university.

**PreApprovalManager:** This class is used by the CourseWishList class and it is useful for many functionalities. The PreApprovalManager class is useful for checking whether the given CourseWishList is a valid wishlist in terms of the prerequisites and credit load. It performs as a layer of abstraction for the validation process.

**Course:** The course is an abstract class useful for capturing the behavior of HostCourse and BilkentCourse. This class contains the ECTS credit, course name, and course code.

**BilkentCourse:** This class is a child class of the Course class and contains the information about whether the course is to be taken as an elective course and the prerequisite BilkentCourses of this specific BilkentCourse.

**HostCourse:** This class is a child class of the Course class and contains the information about whether the course has been approved earlier, the name of the host university, and what are the BilkentCourses that a HostCourse can correspond to.

**ApprovedCourses:** This class acts as an interface between the Approved Courses Database and this program. It contains a map of HostCourse and the corresponding BilkentCourses it can be transferred to. The class contains the functions of adding or removing courses from the database.

**PlacementTable:** PlacementTable class is composed of Placements. Placements may be added to or removed from the PlacementTable to edit the Table.

**Placement:** A placement contains the relevant information of a student’s placement including the student’s name, id, name of the placed university as well as the duration preferred (fall/spring).

**Semester:** The Semester is an enumerable class and it’s either FALL or SPRING.

**PlacementManager:** This class handles the complex computations taking place in the placement process. It has a function to make the placements and creates a PlacementTable object and a WaitingBin object.

**WaitingBin:** WaitingBin is an object to denote the Waiting Bin which is essentially a sorted array of the Students who haven’t been placed yet. WaitingBin has the operations to add a student, remove a student, picking the top student which is critical for placement algorithms.

**Document:** Document is an abstract class for capturing the similar behavior of the classes Syllabus, Transcript, ScoreTable. It has the attribute content that is of type string.

**Syllabus:** Syllabus class is a child class of the Document class, it contains the course code and the name of the class. The actual syllabus is contained in the database and is accessed through the course code in the underlying server structure.

**Transcript:** Transcript class is a child class of the Document class, it contains a map that maps a Course object to a boolean value that denotes whether the course has been taken or not.

**ScoreTable:** ScoreTable class is a child class of the Document class, it is composed of a ScoreTableRow class and it captures the relevant information of a row in a score table uploaded by the ISO.

**ScoreTableRow:** ScoreTableRow contains the student’s name, last name, the host universities the student wishes to be placed, and the total points gathered before the application.

### **3.5.3. State Diagram**

### 

Fig. 3. State Diagram of Erasmus process for students

This is the main state diagram that shows the Erasmus process for students. After students apply for Erasmus, they wait for placement results. There are 2 different ways for a student to place host organizations. Firstly, the student can be placed in a host organization according to the initial score table. Secondly, the student can be placed in universities from the waiting bin. If the student is not placed in any host organization according to the initial score table, the student enters the waiting bin. If the student is at the top of the waiting bin and there is a cancellation of placement before the placement period ends, the student at the top of the waiting bin is placed in a vacancy. If the student at the top of the waiting box refuses to place, that student is removed from the list, and the top of the list changes. After the placement period ends, the placed student is ready to be nominated to the host organization by the Administrative Erasmus Coordinator. After the student is nominated, they need to prepare their wish list, which includes the courses that they want to take in Erasmus. If the Pre Approval period due passes, the student cannot submit a wish list. Also, the student cannot submit a wish list until all the courses in the wish list have been approved by the relevant coordinators. To send approval requests for counting courses in the host university to courses in Bilkent university to coordinators, the student must upload the syllabus for all disapproved courses. Except for the previously accepted host university course - Bilkent course combinations, the student must upload the syllabus for both MUST and elective courses and their intent to take that course in order to propose to take a course to be counted for the place of a Bilkent course. It is up to the relevant course coordinator to approve or reject that request. If the course coordinator rejects the request, the student needs to update his/her wish list. Some courses are approved before, so the student may take any of these courses. After all courses on the wish list are approved by related coordinators, students submit the wish list, and this wish list is transformed into a Pre Approval form automatically. After that, the student waits for mobility. After this student makes pre-registration in the host organization. However, some courses may not be loaded for the student due to schedule conflicts or prerequisites. When a student fails to register all courses in their Pre Approval form, they are said to have failed Pre Approval in the state diagram. Therefore they need to submit another Pre Approval form according to the courses they were able to register for. After the student finishes all courses at the host organization, if the official transcript matches the Pre Approval form then courses will be transferred, otherwise unmatched courses will not be transferred.

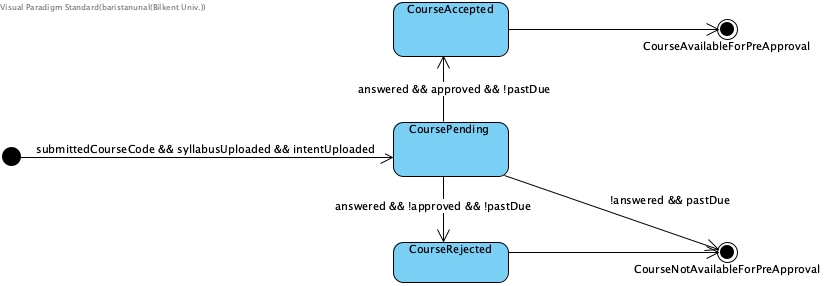


Fig. 4. State Diagram for course approval process

This is the state diagram for the course approval part after preparing the wish list. Students upload the syllabus for each unapproved course in the selected course. Students send approval requests in order to propose to take a course to be counted for the place of a Bilkent course to the Department Erasmus Coordinator for elective courses and to the related Course Coordinator for MUST courses. Coordinators approve or reject. If they do not answer before the Pre Approval period ends, then courses are not approved. Therefore, the student cannot take this course. In such a situation the student has to choose courses that are approved before.

### 

Fig. 5. State Diagram for course transfer process

This is the state diagram for the course transfer part after the student returns. According to a student who completed her Erasmus process, her host university sent an official transcript to her to send to ISO. In the first part of this diagram, we assumed that some host universities may not send official transcripts to the International Students Office. Therefore, the official transcript could be sent by the host university or the student herself/himself. After that, the Department Course Coordinator compares the Pre Approval form with the official transcript in the pending status of the course transfer form. If the courses in the transcript are the same as the courses in the Pre Approval form, all pre approved courses are transferred, else the course transfer form is updated and the courses that are taken in the host university but not in the Pre Approval form are excluded from the course transfer form. This cycle repeats until all remaining courses in the course transfer form are in the Pre Approval form submitted. When this condition is satisfied, pre approved courses that are taken in the host university are transferred and the faculty committee signs the course transfer form. Finally, the course transfer form is sent to the student services and the faculty committee is notified.

### **3.5.4. Activity Diagram**

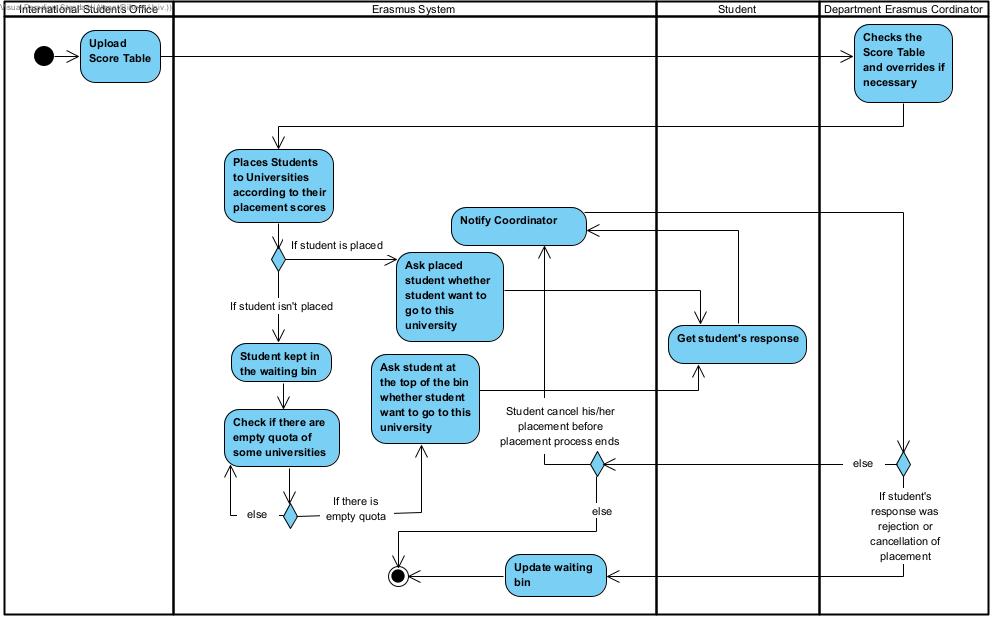
****

Fig. 6. Activity Diagram for placement process for students

This activity diagram shows the process of placement for students. After the Score Table is uploaded, the Department Erasmus Coordinator can override the Score Table if it is necessary. Either by initial sample scores or the waiting bin, the student is selected for placement. The application asks students automatically whether the student wants to go to the universities they were chosen for. The student may cancel his/her placement before the placement process ends. If they cancel, the Department Erasmus Coordinator is informed, and the application updates the waiting bin and picks the student at the top of the bin, and asks them whether they want to go or not.

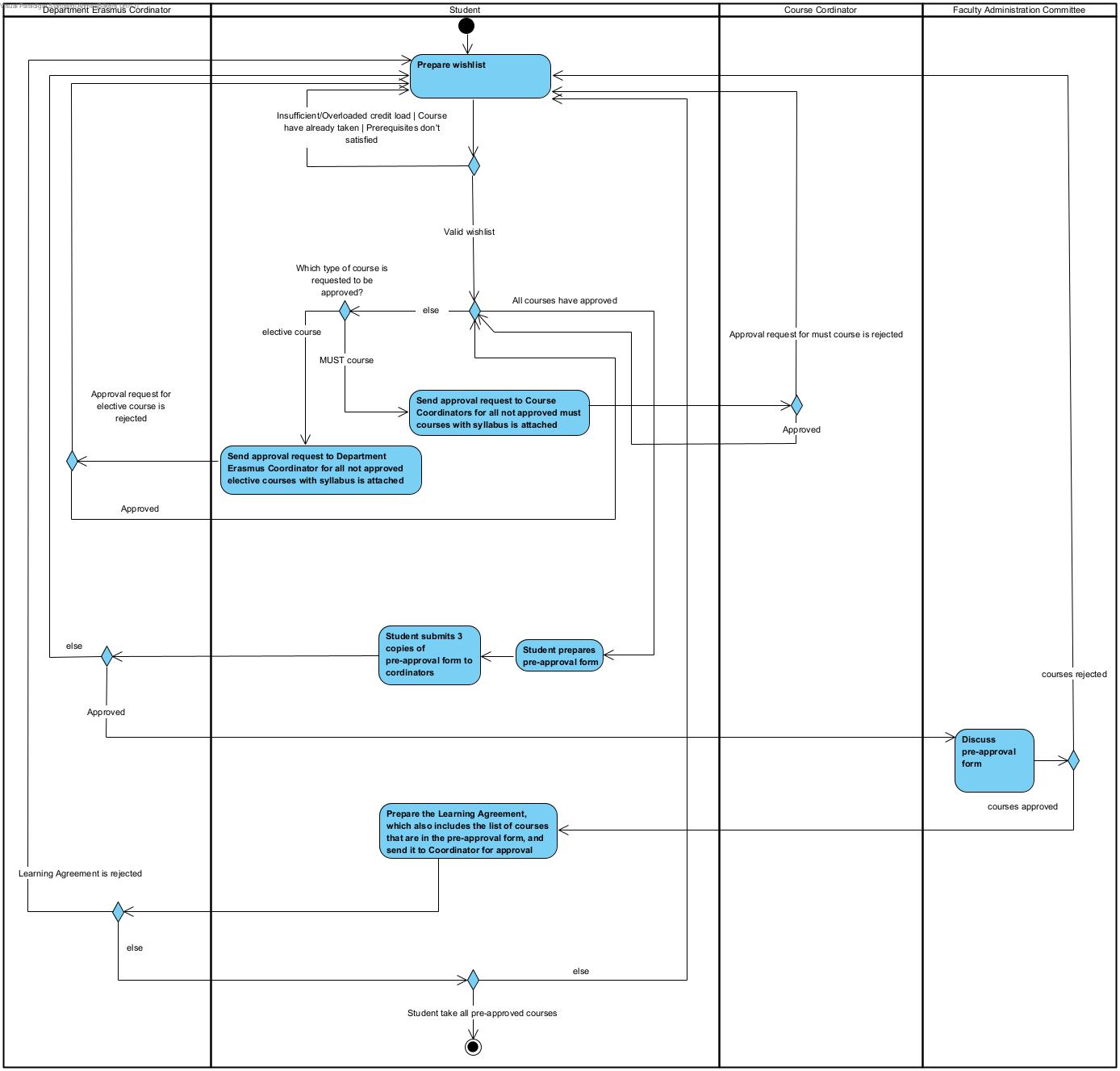


Fig. 7. Activity Diagram for Pre Approval process

This activity diagram shows the Pre Approval process for the placed students. Each student prepares a wish list. Each course in the wish list should be an approved course. student sends the syllabus for each unapproved selected course to the Coordinator(s). Students send approval requests in order to propose to take a course to be counted for the place of a Bilkent course to the Department Erasmus Coordinator for elective courses and to the related Course Coordinator for MUST courses. After all host university courses, that student wants to count for the Bilkent course, on the wish list are approved, the student prepares a Pre Approval form and sends copies to coordinators. Department Course Coordinator approves or disapproved Pre Approval form. If the Department Course Coordinator approves it, the approval request is also forwarded to The Faculty Administration Committee. The Faculty Administration Committee discusses and approves or disapproves each Pre Approval form. If the Pre Approval form is approved by the Committee, then the student downloads the Learning Agreement and fills it out. After that, the student sends the Learning Agreement to the Department Erasmus Coordinator for final approval. After that, the students will register for courses at their host university. If any student doesn’t successfully take all courses in the Pre Approval form, then the student needs to prepare a new wish list and repeat the process.

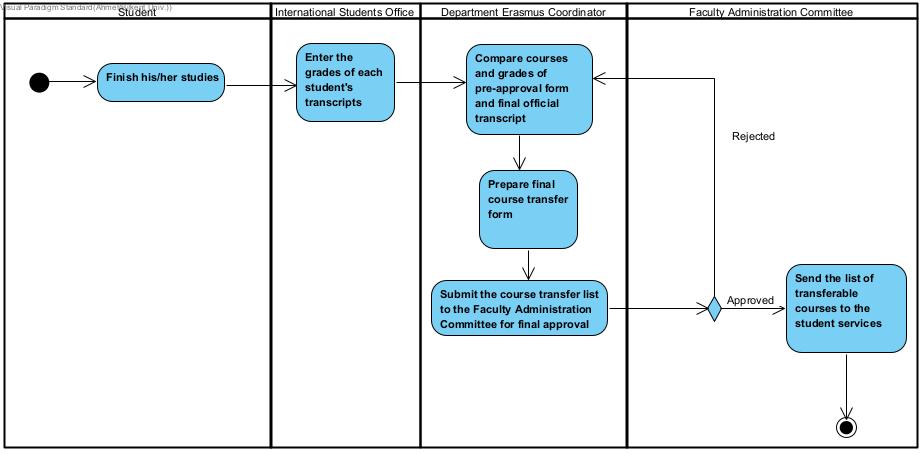


Fig. 8. Activity Diagram for course transfer process

This activity diagram shows the final approval process for transferable courses. The International Students Office (ISO) receives the transcript of a student either from the Host Organization or the student depending on the circumstances. Some Host Organizations don’t send transcripts to ISO directly according to some Bilkent students who completed their Erasmus process. Therefore, the student has to get their transcript and send it to ISO. ISO enters the grades of each student's transcripts to system. Department Erasmus Coordinator compares the official transcript with the Pre Approval form of the student and prepares the final course transfer form. Then send it to the Faculty Administration Committee. The Faculty Administration Committee approves or disapproves the given course transfer list. According to the result of this decision, courses will transfer for the student.

### **3.5.5. Sequence Diagram**

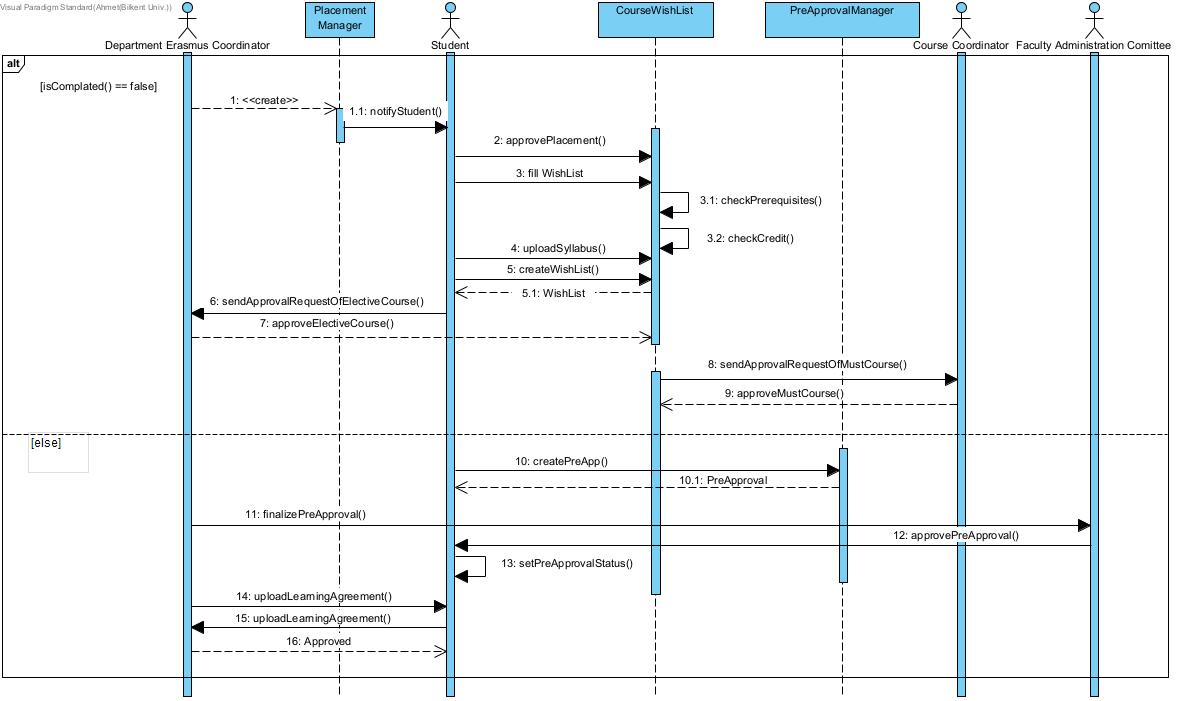


Fig. 9. Sequence Diagram for Pre Approval process

Students may or may not approve their placement once selected by the Placement Manager. After confirming their placement, the student clicks on the WishList Page and fills in the courses he/she wants to take. If selected courses satisfy prerequisites and credit limits, students click the button and create a wish list. If any of the selected MUST or elective courses are not approved before, the student uploads the syllabus to the allocated area near the selected course and submits these to the coordinators responsible for each specific course. If any of the selected courses are not approved by the Coordinators then the student has to change such courses and repeat the process above. After all selected courses are approved, the student creates a Pre Approval course and 3 copies of the Pre Approval form are automatically sent to the coordinator(s). The Department Course Coordinator approves or disapproves Pre Approval form. If the Department Course Coordinator approves it, approval request is also forwarded to The Faculty Administration Committee. The Faculty Administration Committee discusses and approves or disapproves them. Then the student downloads the Learning Agreement, which is uploaded by the Department Course Coordinator, and fills it out. Then students upload the filled Learning Agreement and send it to the Department Course Coordinator for final approval. If it is accepted, then students are ready to start the Erasmus process.

### **3.5.6. User Interface**

**Student Wish List Page**

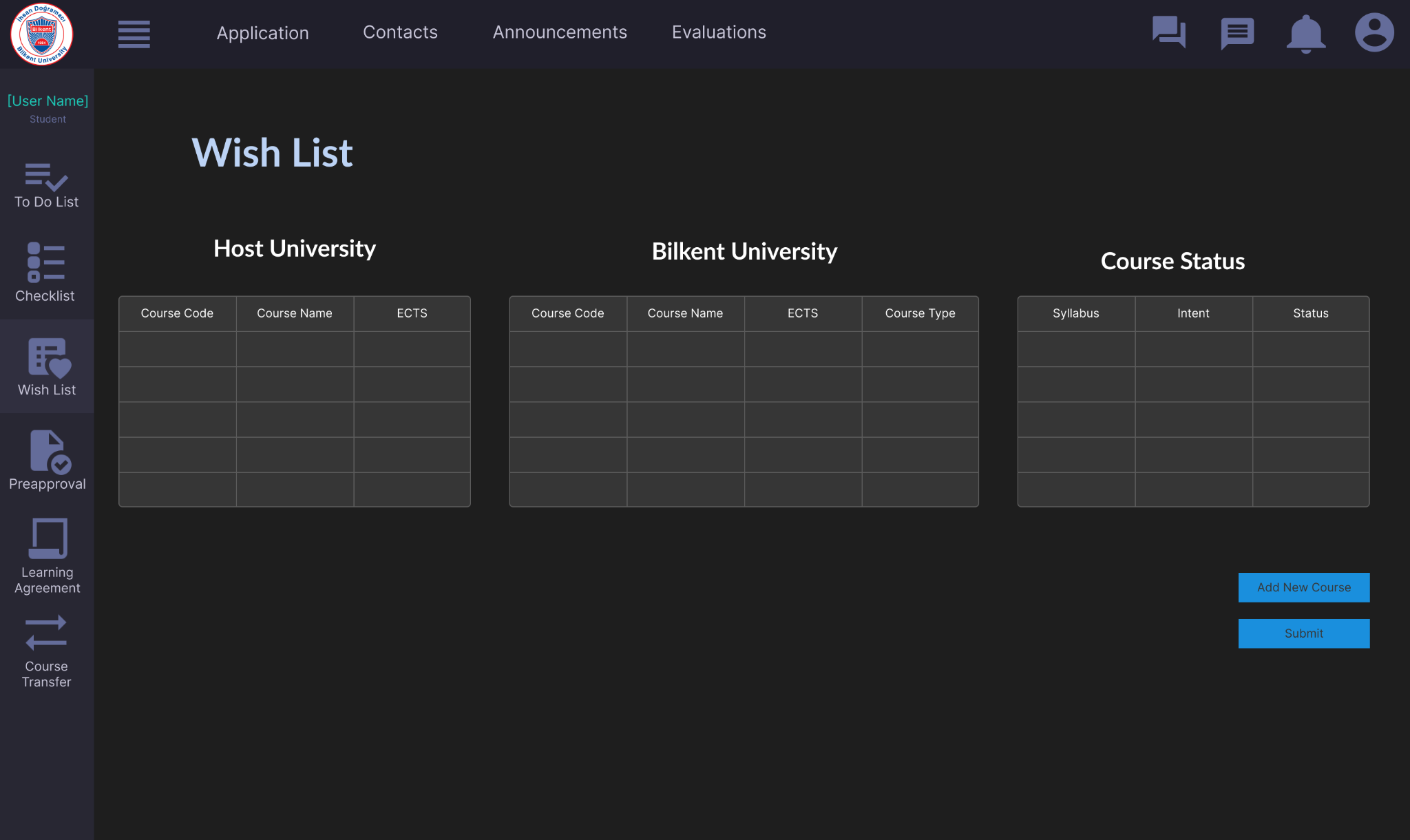


Fig. 10. UI for Student Wish List Page

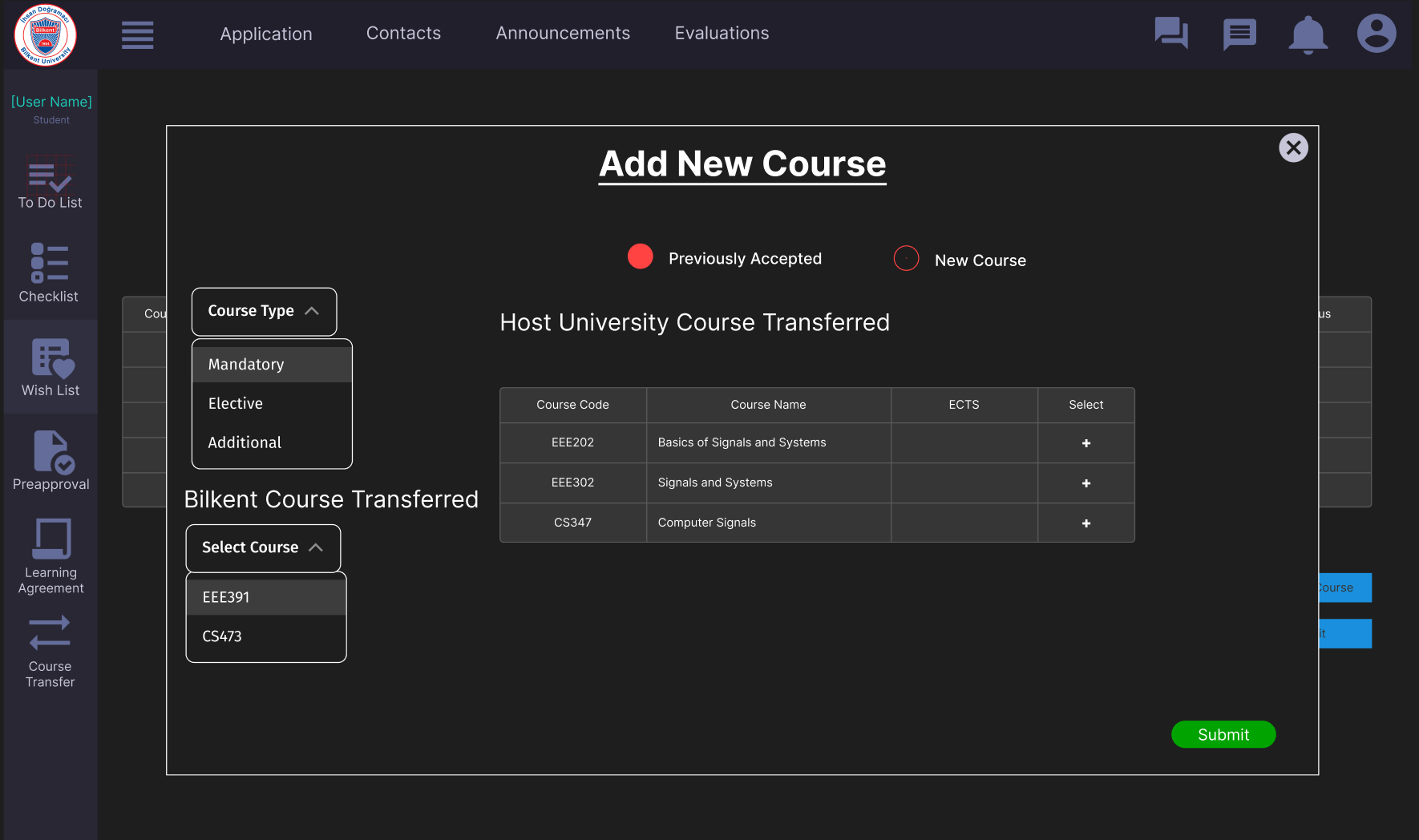
**Student Wish List - Add New Course Page**

Fig. 11. UI for Student Add New Course Dialog

**Student Checklist Page**

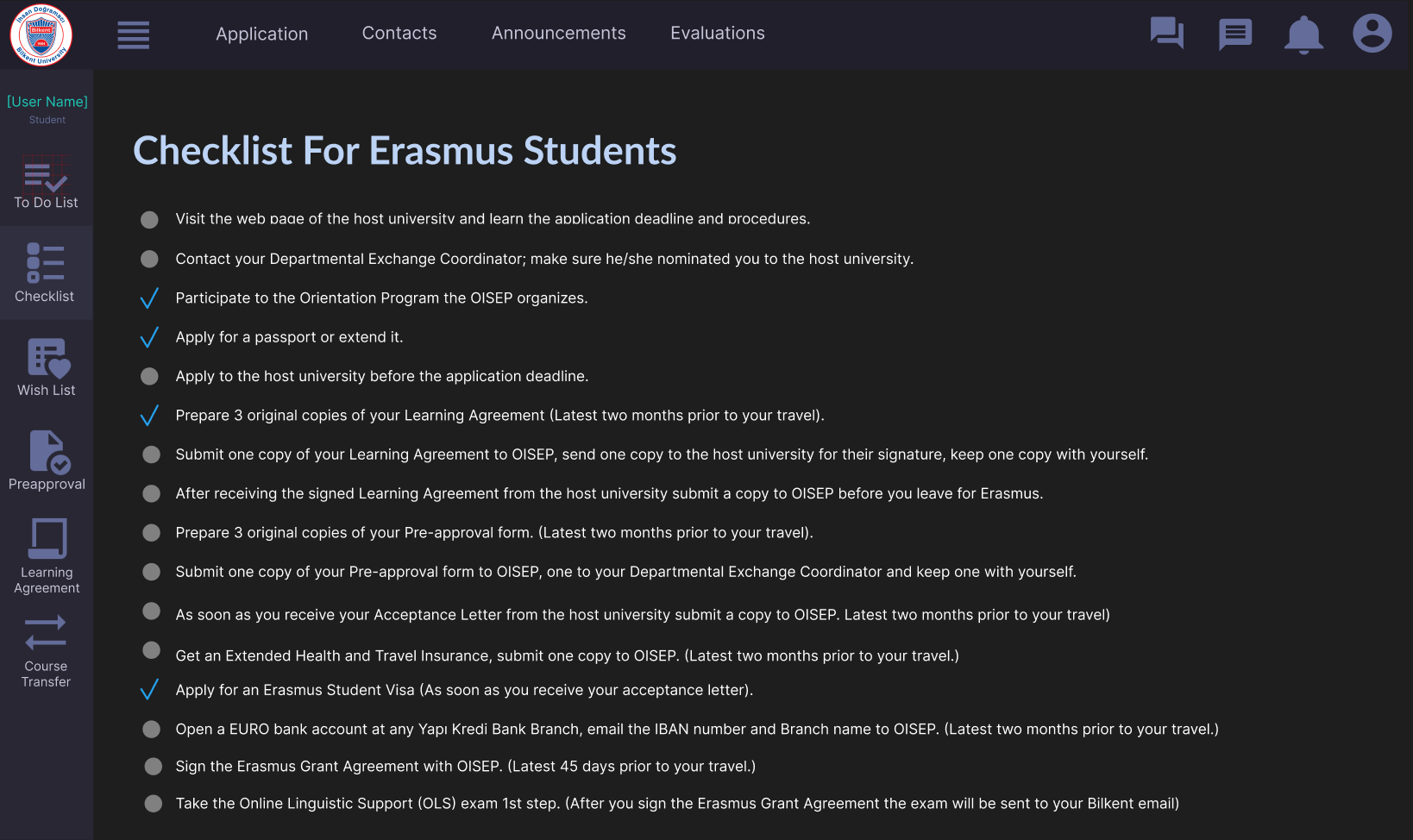
****

Fig. 12. UI for Student Checklist Page

**Student Pre Approval Page**

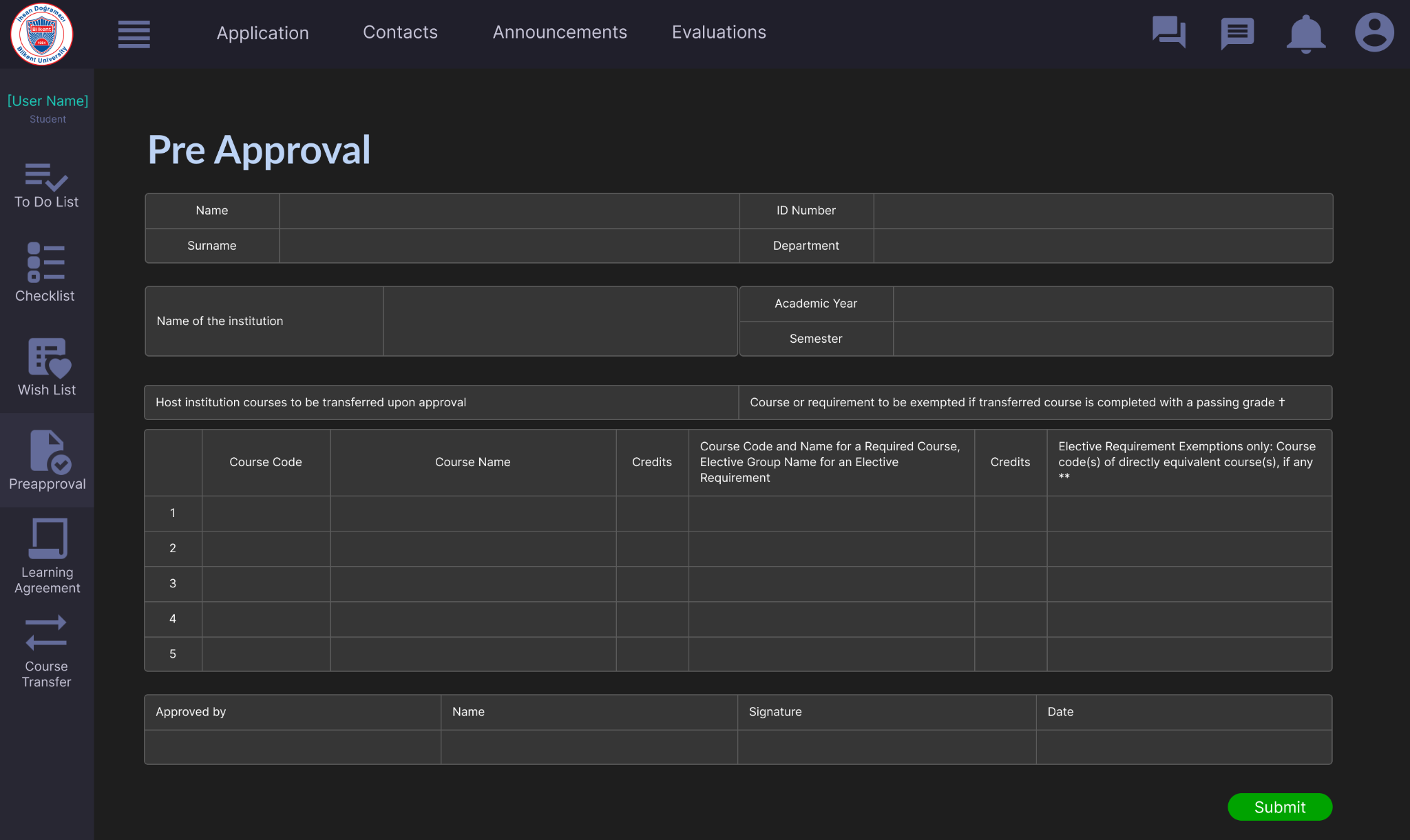
****

Fig. 13. UI for Student Pre Approval Page

**Student Learning Agreement Page**

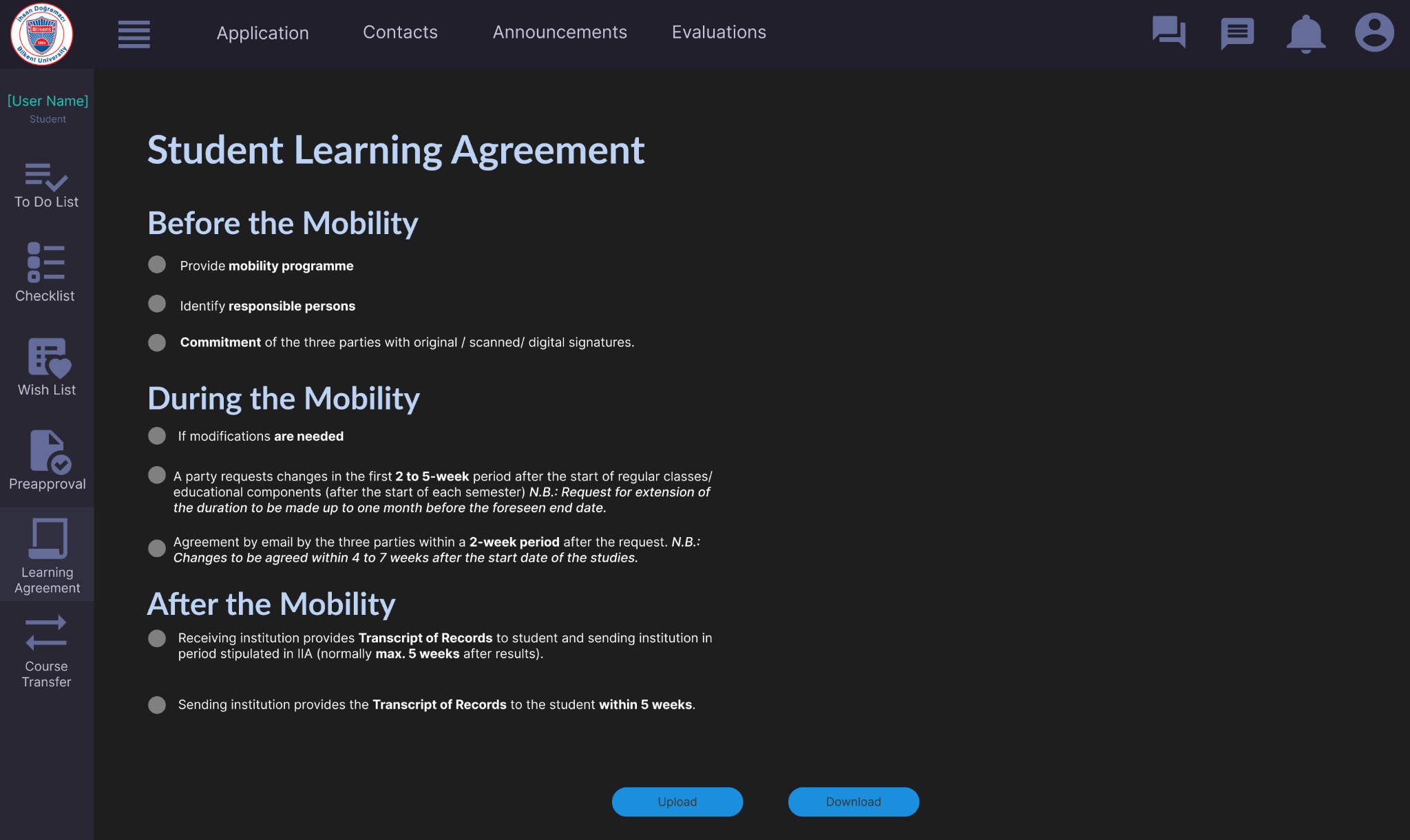
****

Fig. 14. UI for Student Learning Agreement Page

**Student Applications Page**

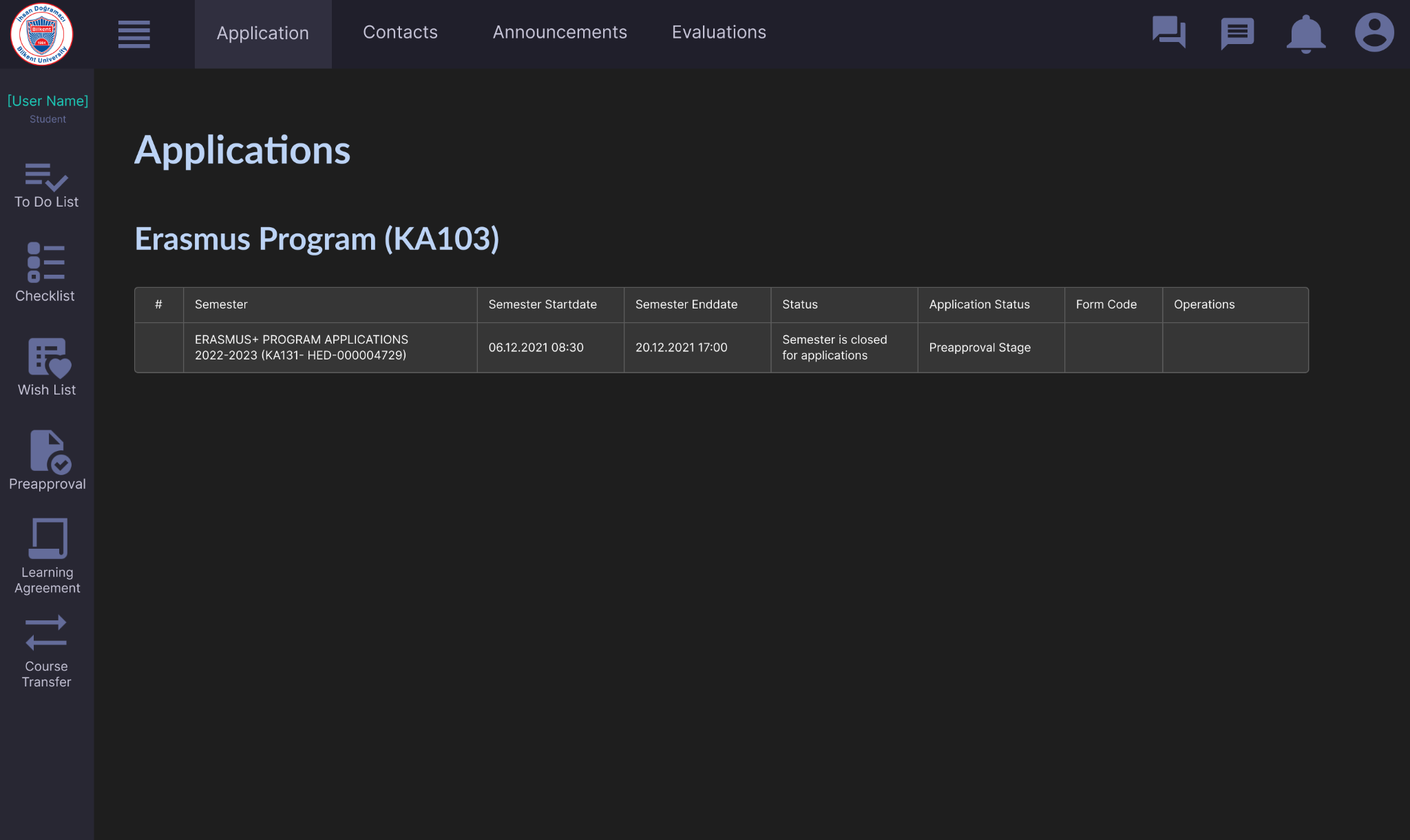
****

Fig. 15. UI for Student Applications Page

**Student Contacts Page**

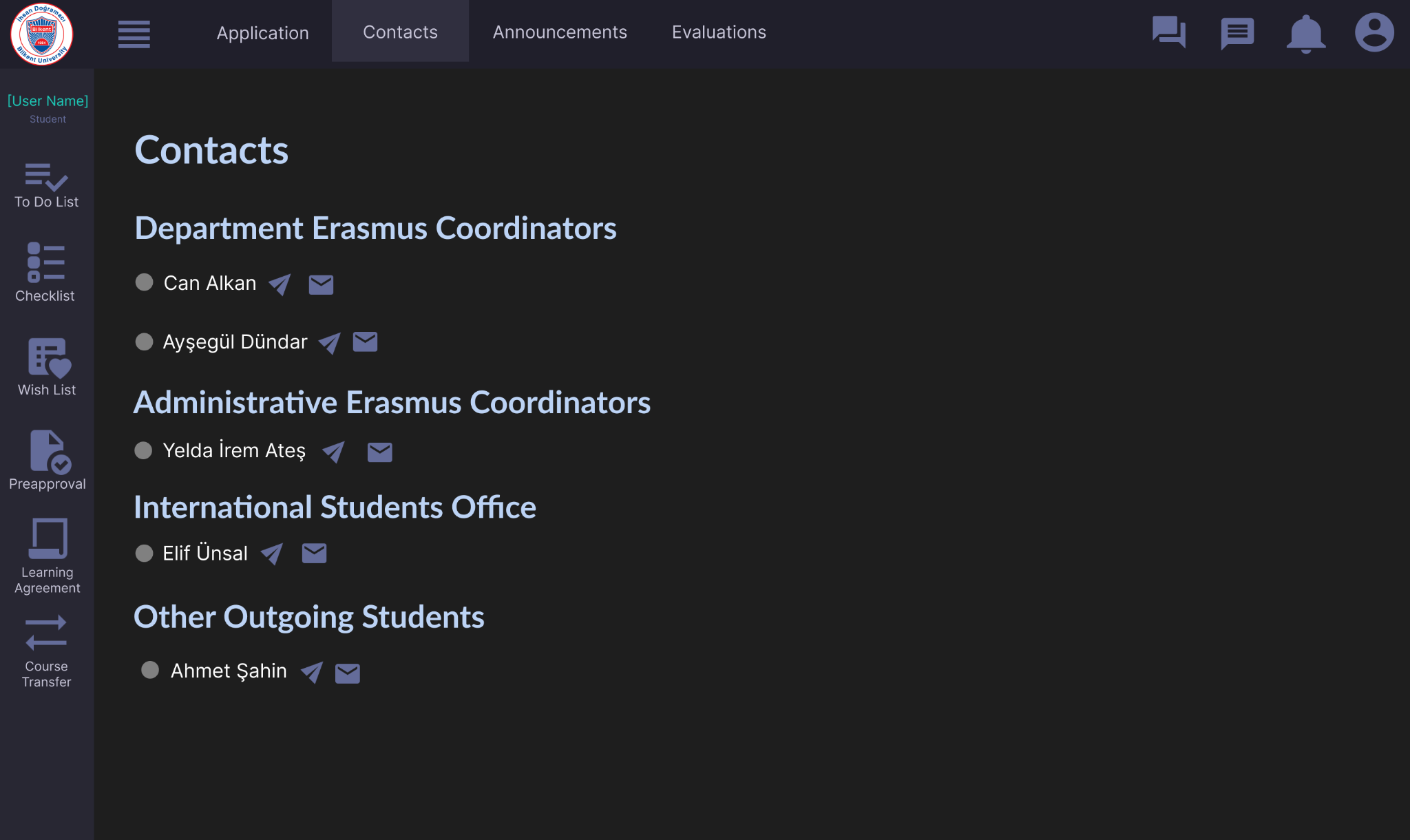
****

Fig. 16. UI for Student Contacts Page

**Student View Evaluations Page**

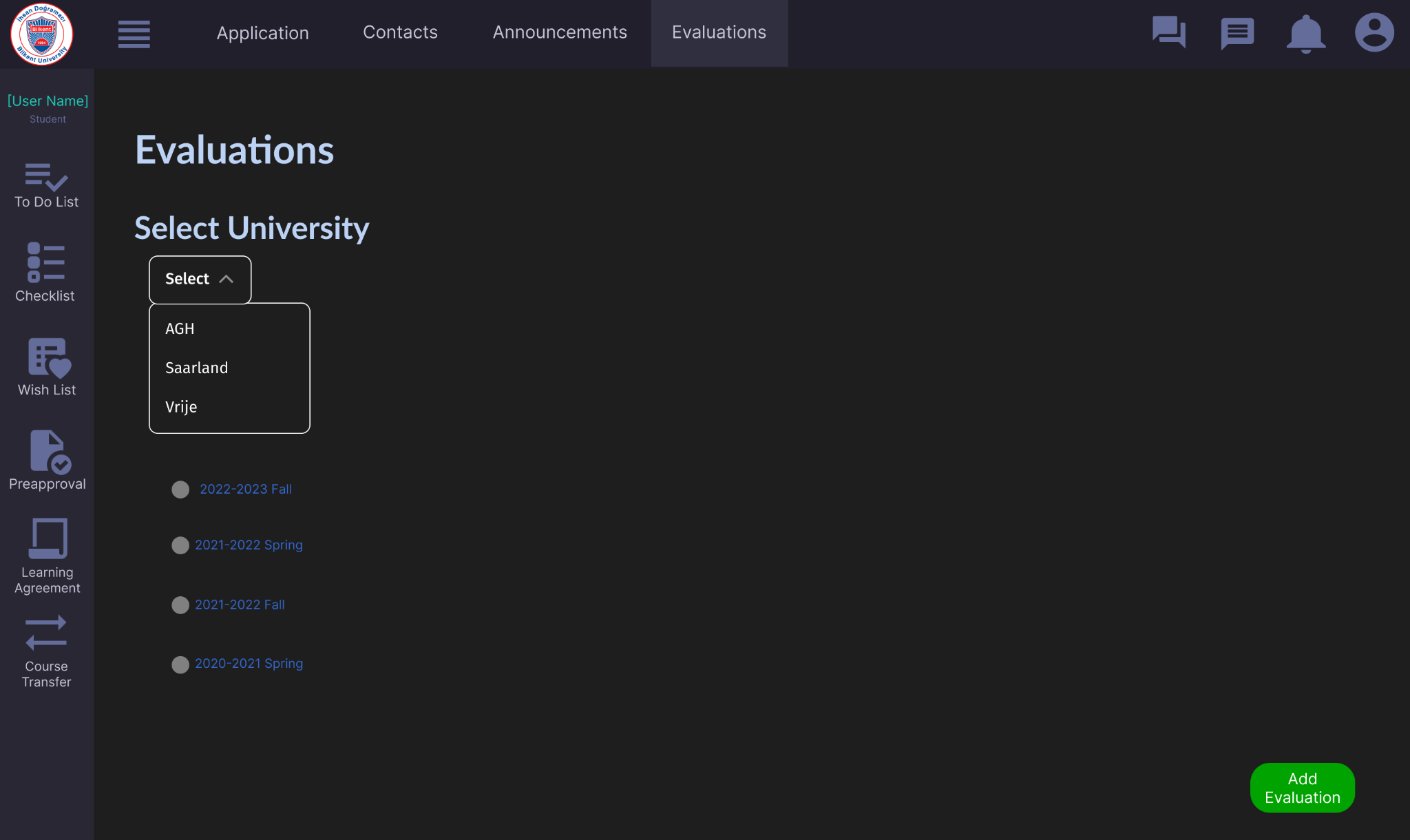
****

Fig. 17. UI for Student Evaluations Page

**Student Evaluate Host University Page**

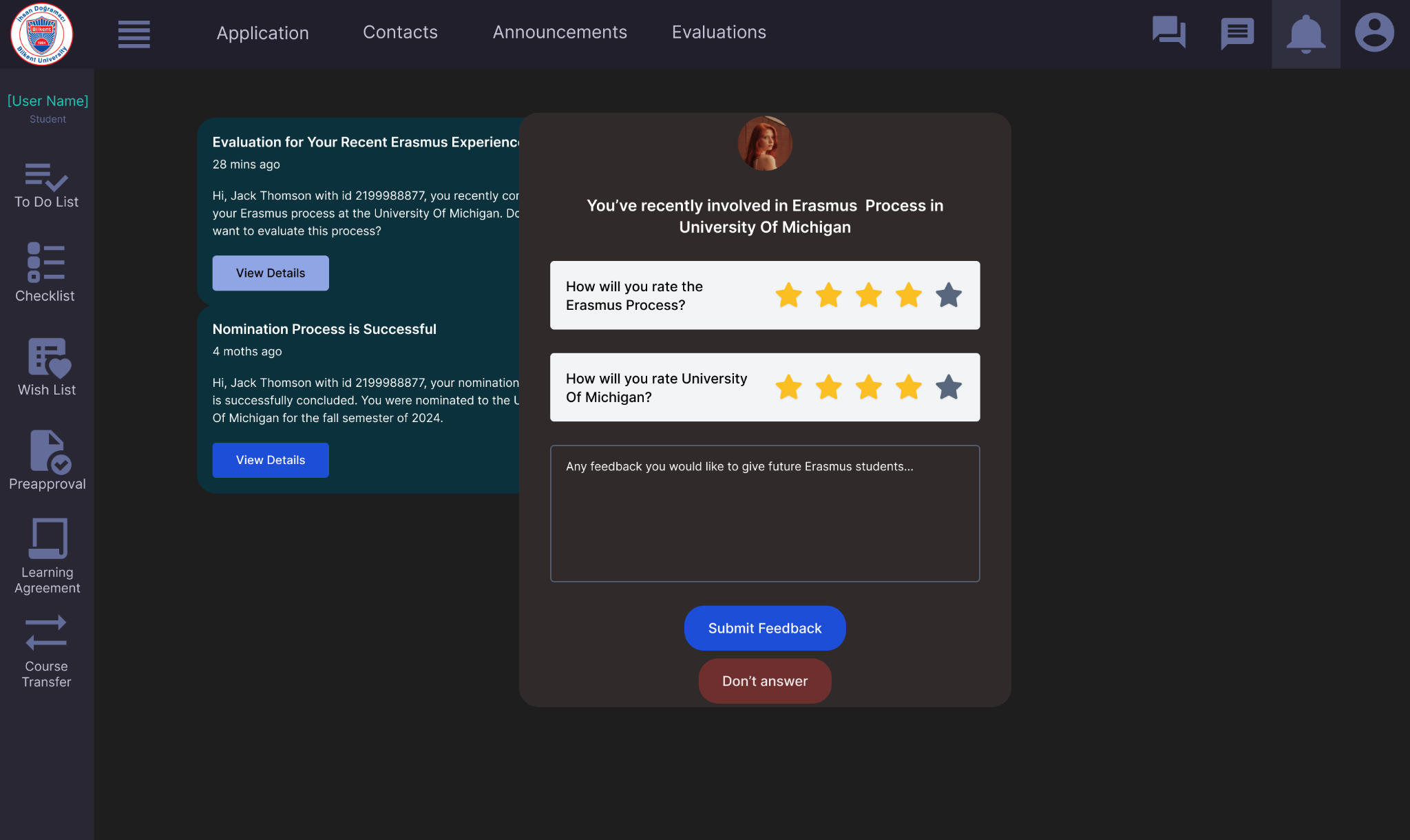
****

Fig. 18. UI for Student Contacts Page

**Student Messages Page**

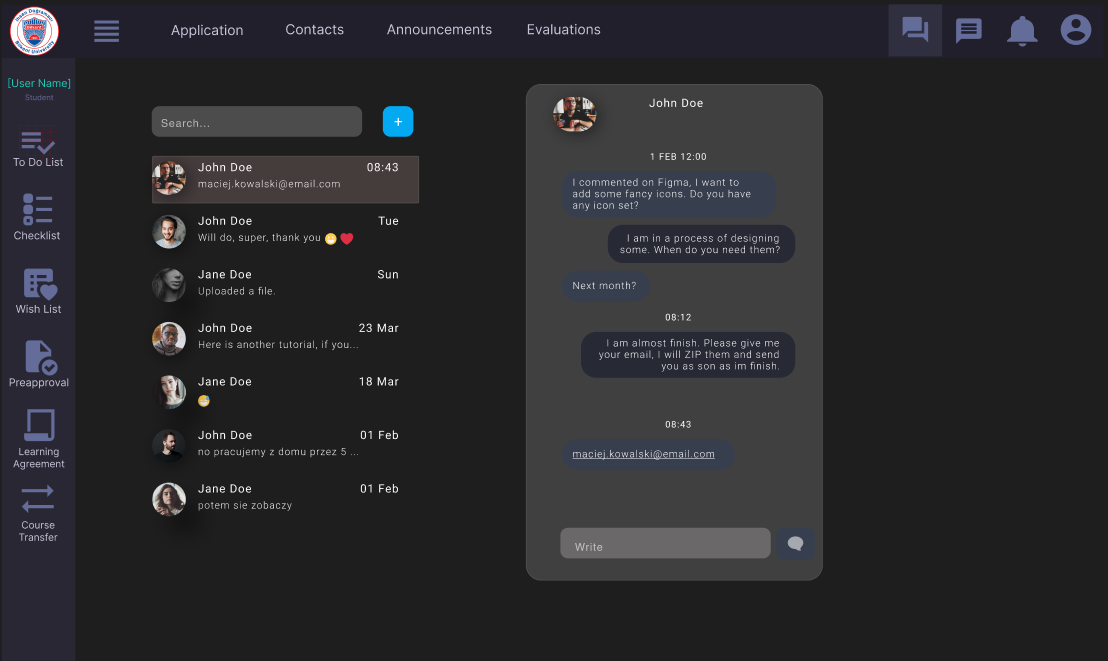
****

Fig. 19. UI for Student Messages Page

**Student Profile Page**

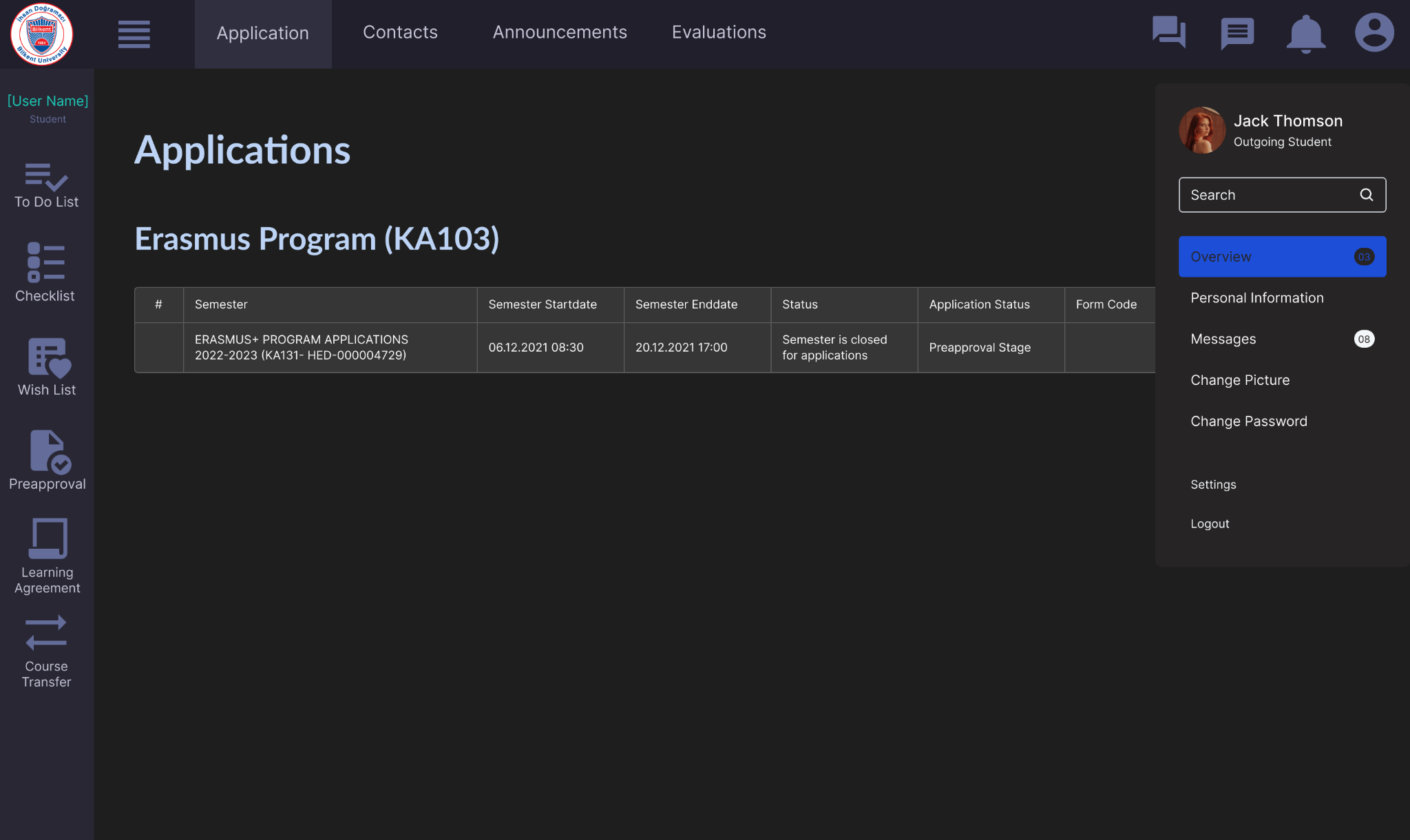
****

Fig. 20. UI for Student Profile Page

**Department Erasmus Coordinator Placements Page**

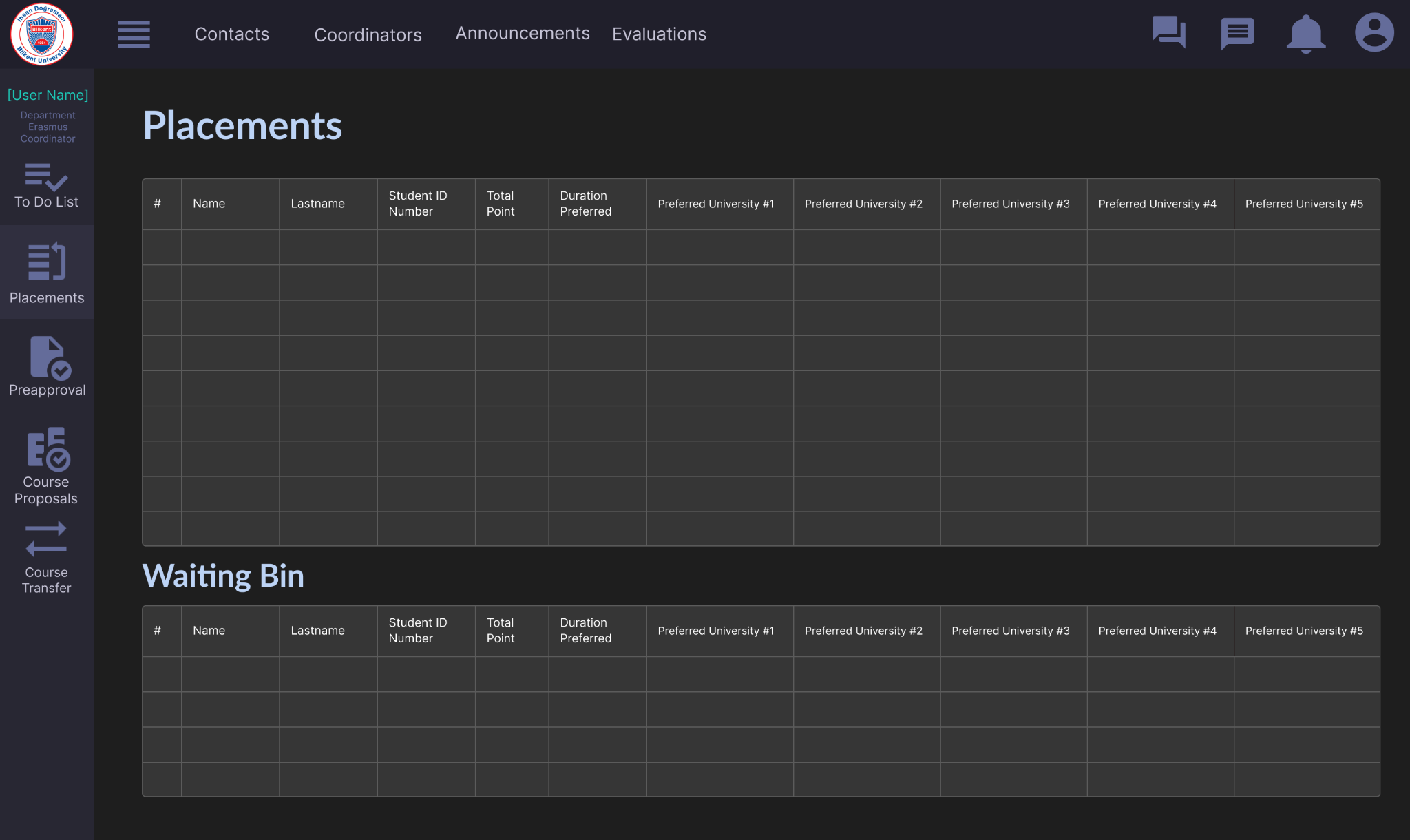
****

Fig. 21. UI for Department Erasmus Coordinator Placements Page

**Department Erasmus Coordinator Pre Approval View**

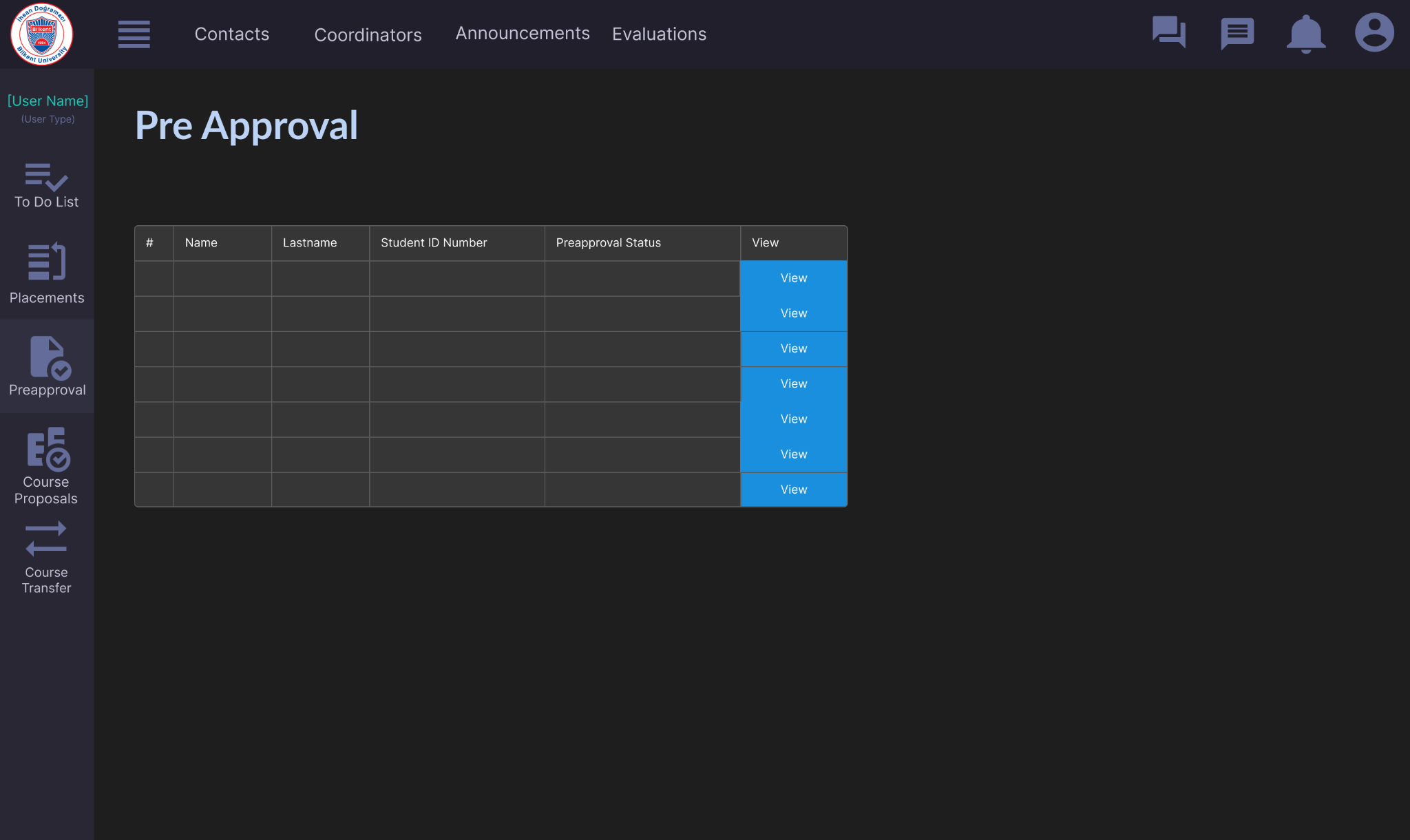
****

Fig. 22. UI for Department Erasmus Coordinator Pre Approval Page

**Department Erasmus Coordinator Pre Approval Approve**

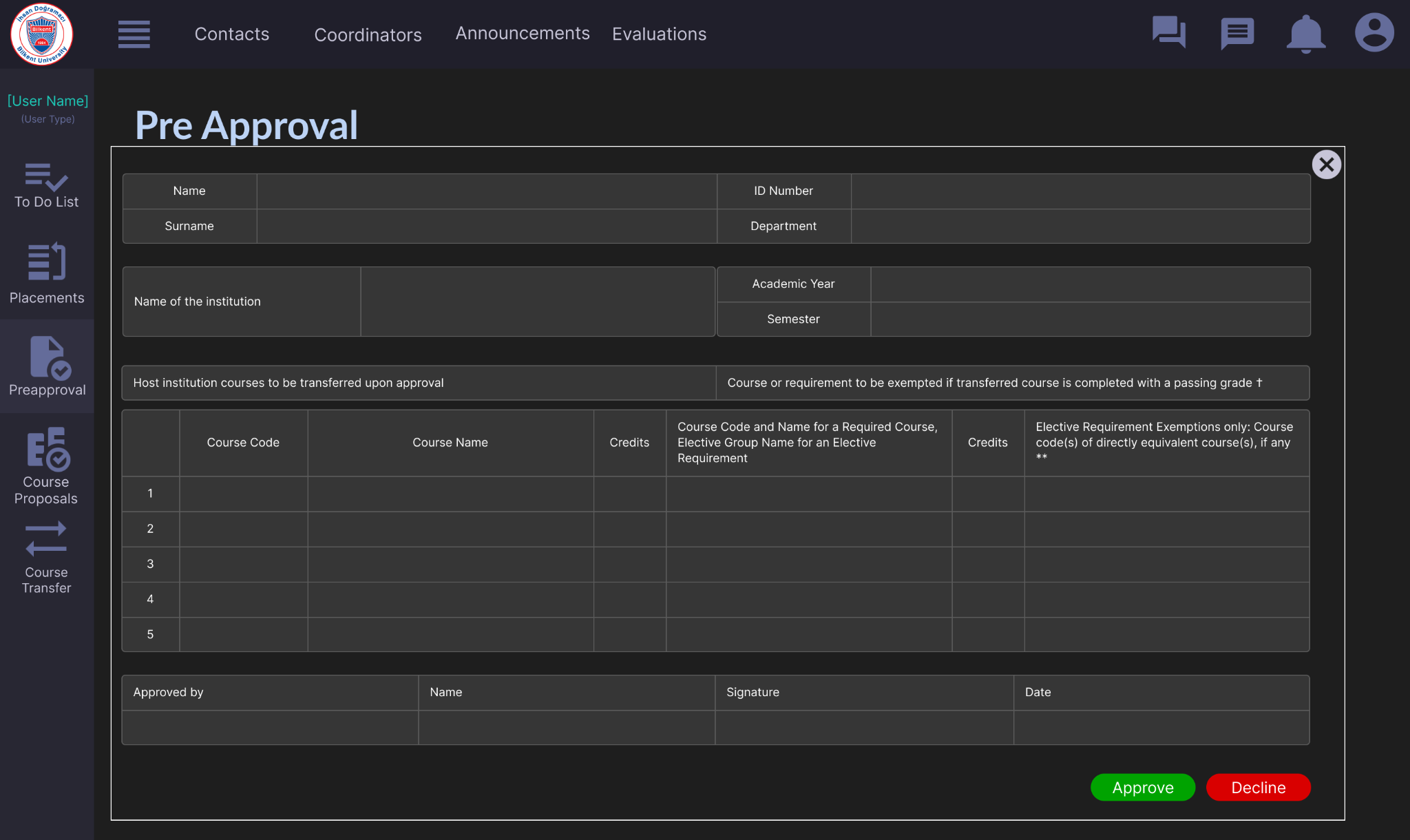
****

Fig. 23. UI for Department Erasmus Coordinator Approve Pre Approval Dialog

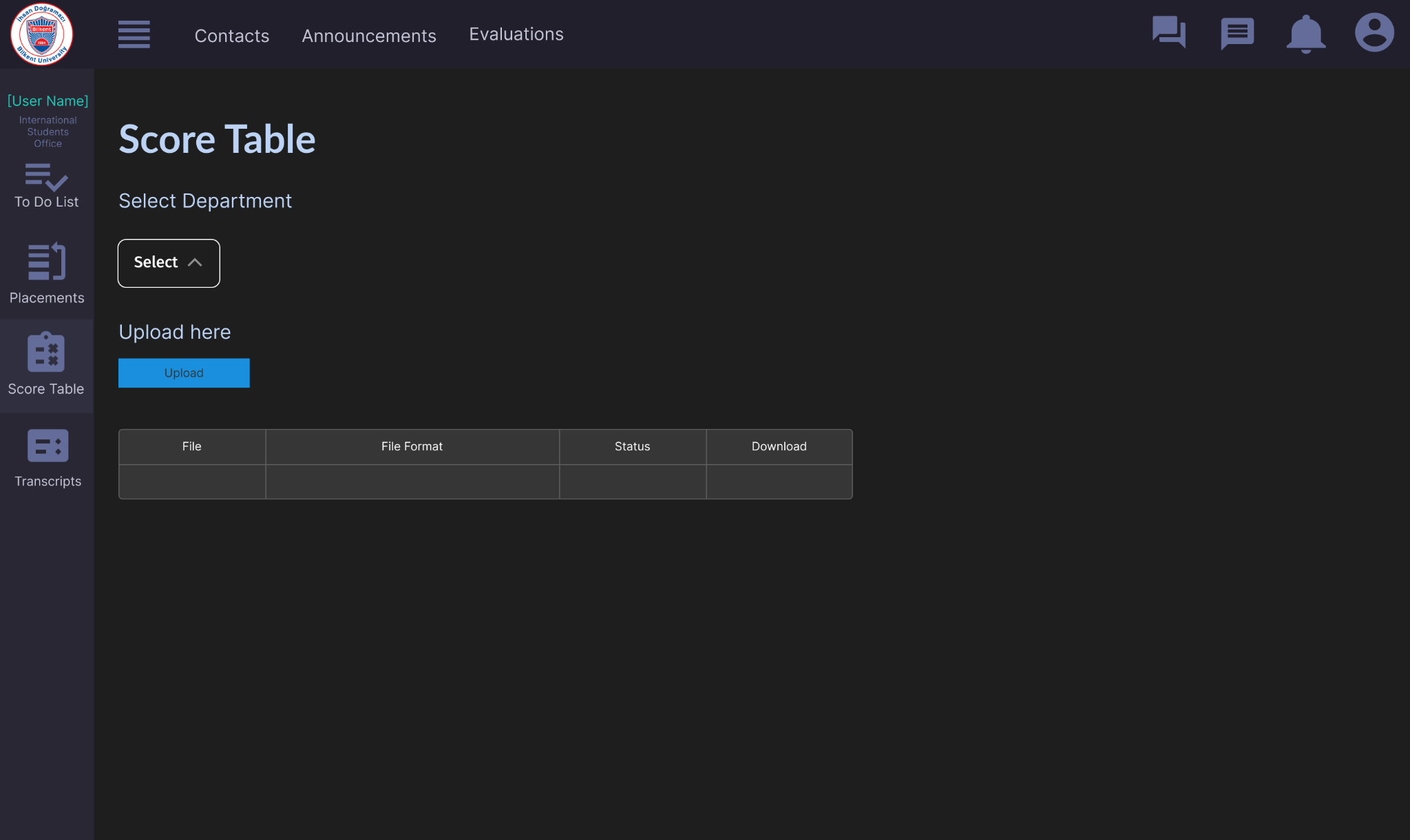
**International Students Office Upload Score Table Page**

Fig. 24. UI for International Students Office Upload Score Table Page

**International Students Office Enter Returning Student’s Transcript Page**

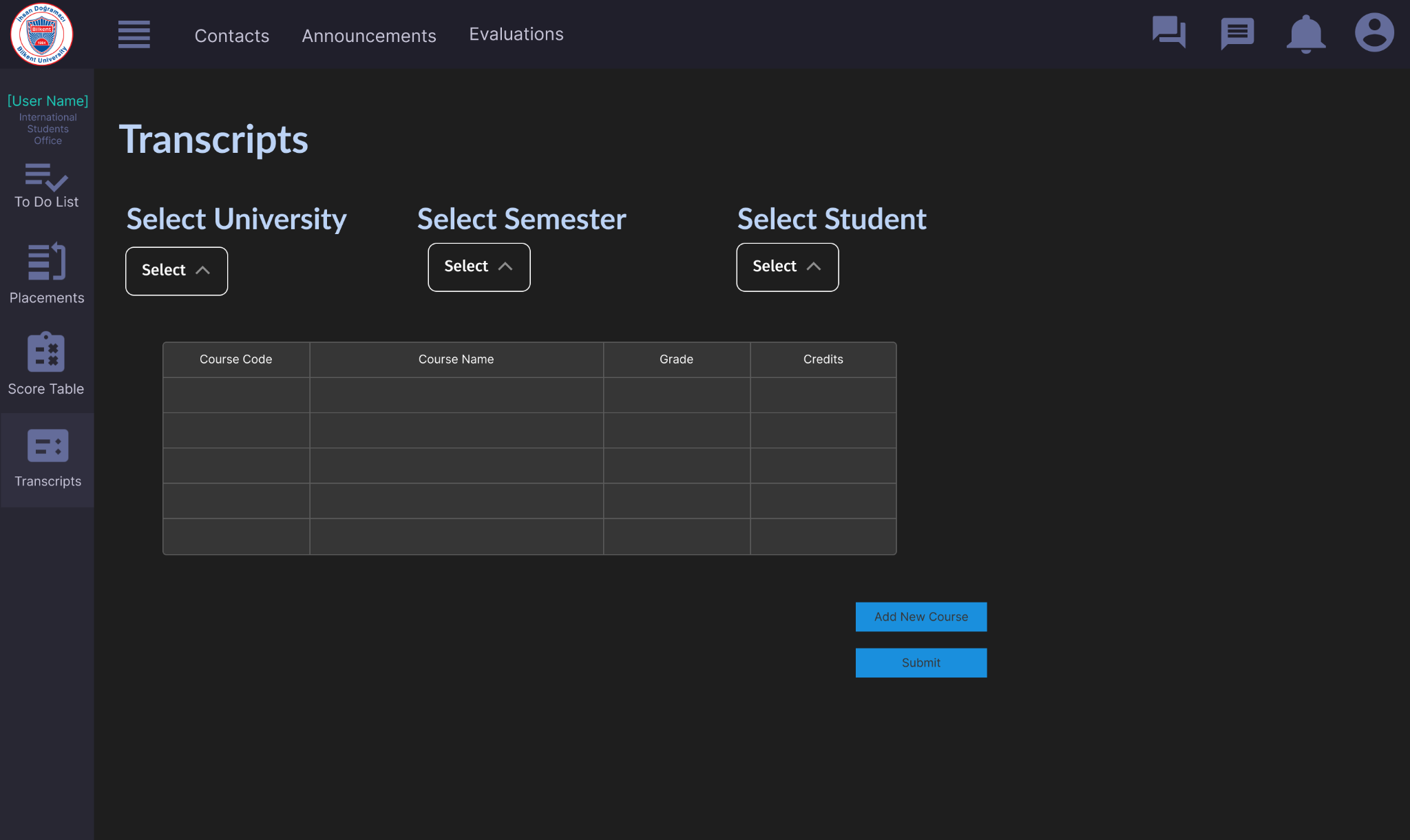
****

Fig. 25. UI for International Students Office Transcript Page

**Dean Course Transfer Exemption Approval Page**

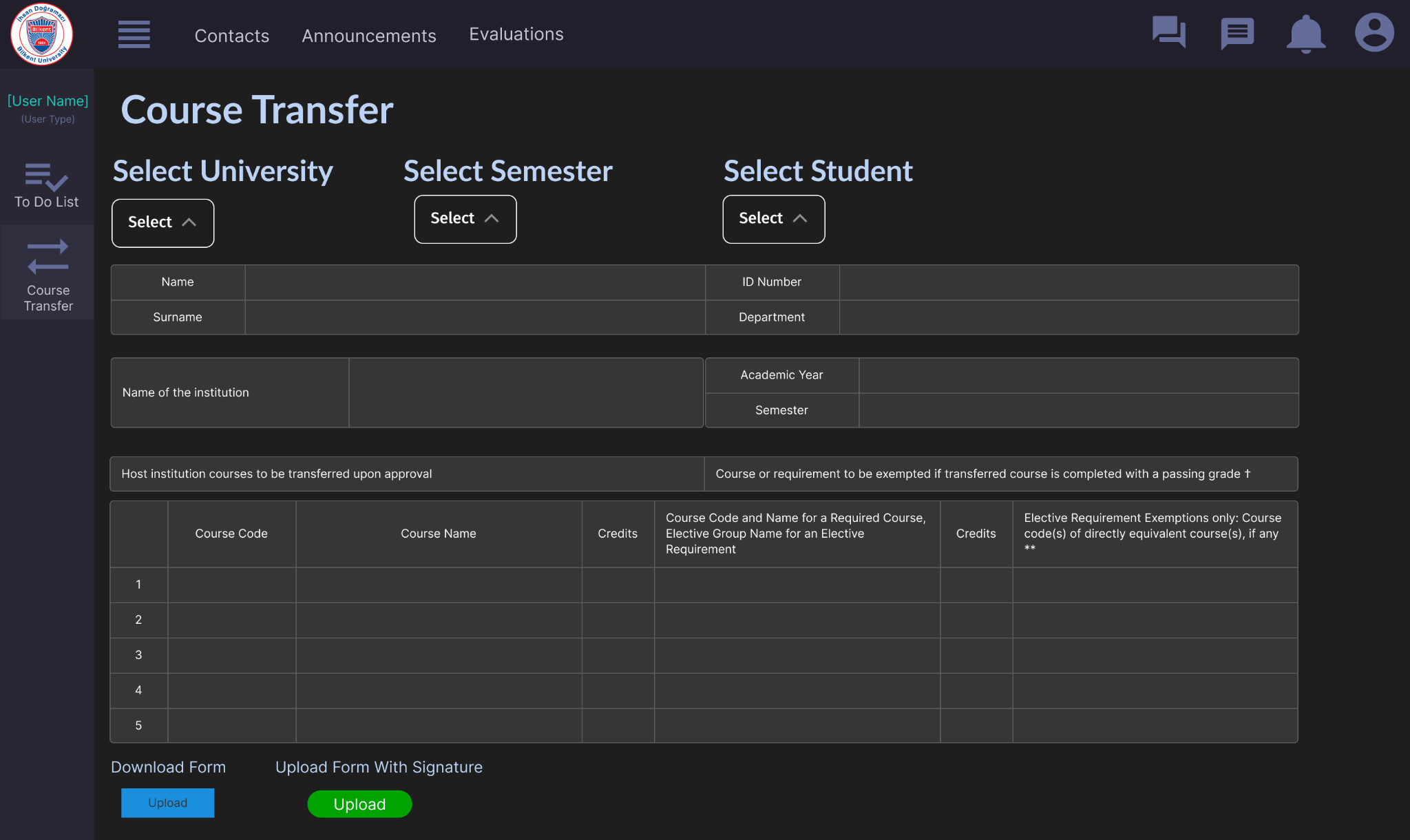
****

Fig. 26. UI for International Students Office Upload Score Table Page

**Faculty Administration Committee Transferable Courses Page**

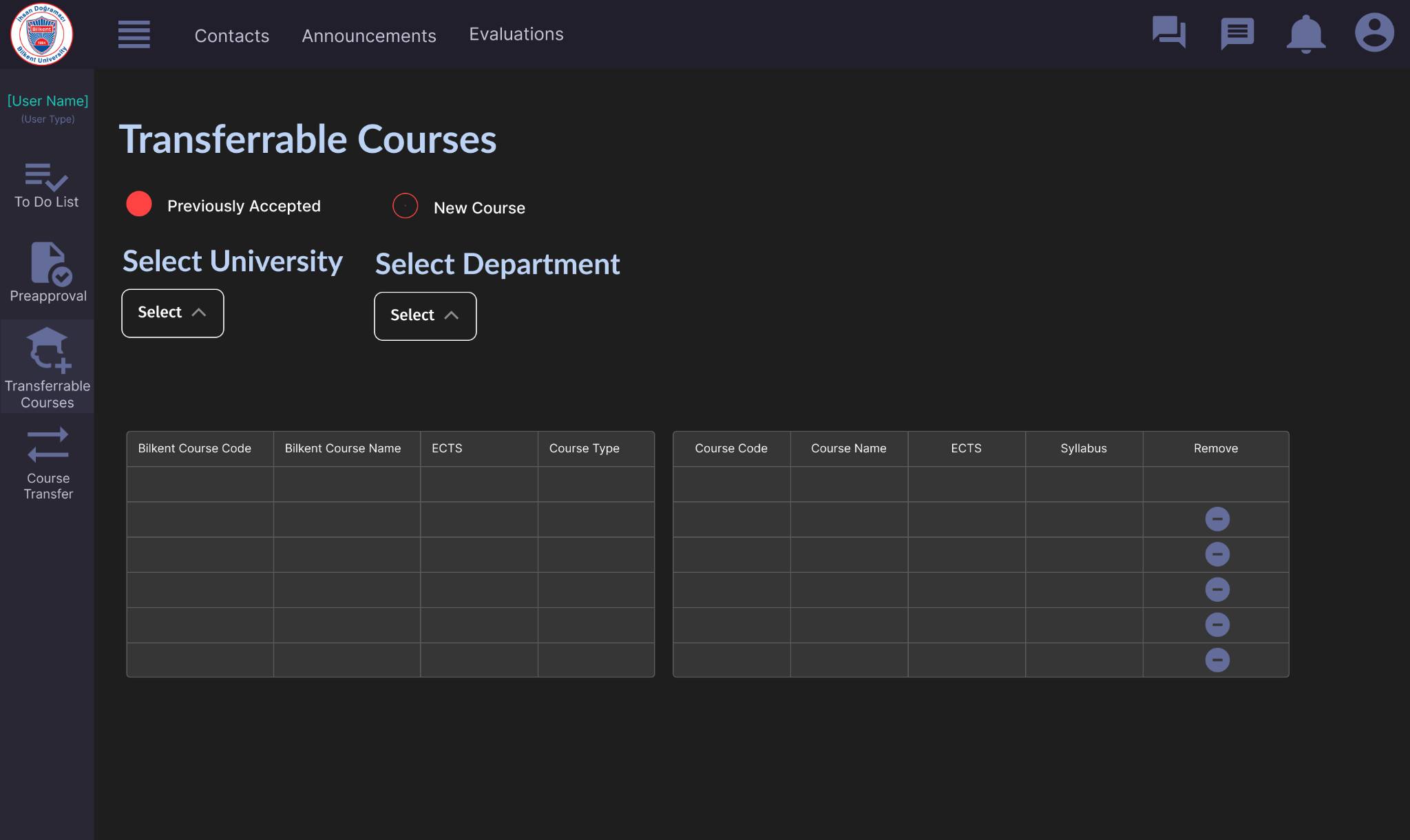
****

Fig. 27. UI for Faculty Administration Committee Transferable Courses Page

**Administrative Erasmus Coordinator Deadline Setting Page**

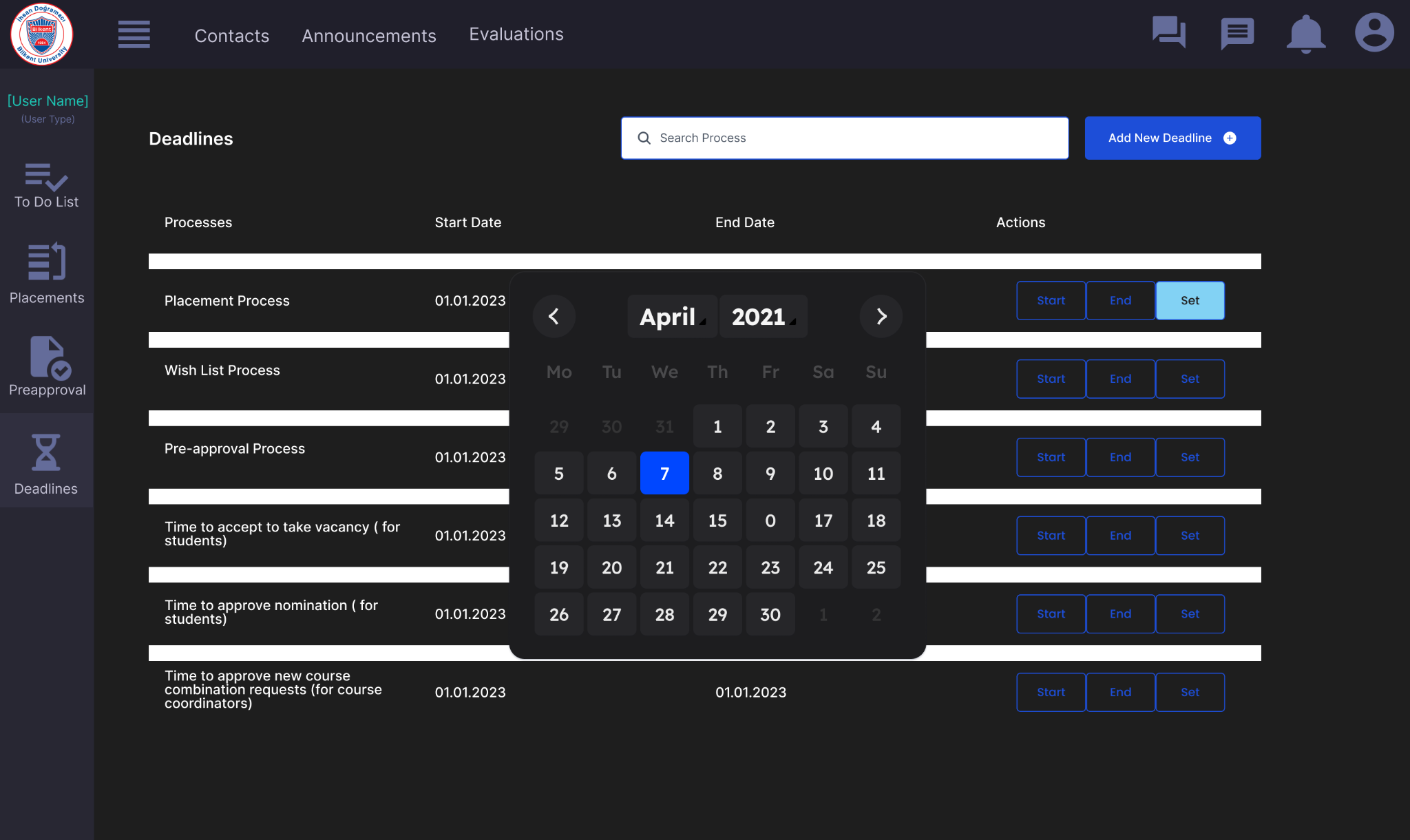


Fig. 28. UI for Administrative Erasmus Coordinator Deadline Page

# **4. Improvement Summary**

## **4.1. Formatting**

* TA and instructor names are added to the cover page.
* Class diagram is made bigger for readability.

## **4.2. Introduction**

* Confusing sentences about the subsystems and our program are changed.
* Parts that explain the algorithm are changed so that now they explain the criteria and process.
* Features and functionalities are placed in the part of the functional requirements.

## **4.3. Proposed System**

* In the overview part, examples are given for additional reasons that can be required to print a document.
* Details about the UI and implementations are removed.
* The name of the actor in one example is changed.
* NFRs changed so that now they are non-ambiguous and objectively testable.
* Pseudo-requirements that are not enforced are removed.

## **4.4. Use Case Model**

### **4.4.1. Changes in Use Case Diagram**

* The use cases “Login”, “UploadDocuments”, “ViewContacts”, “RegisterUsers”, “FetchCourseCoordinators”, “AddNewAdditionalCourse”, and “FetchCourseInfo” are added.
* The use cases “UploadIntent” and “UploadSyllabus” were removed since they are a part of “ProposeNewCourse”.
* Added appropriate connections to and from these newly added use cases.
* “Admin” user type is added.

### **4.4.2. Changes in Textual Use Cases**

* Entry and exit conditions are placed before the flow of events for understandability.
* More generic use case names are used for better description.
* Incorrect entry conditions are modified.
* Some phrases are further explained for understandability.
* Possible conflicts between events are modified.
* Repetitions are removed.
* The wrong participating actor is changed.

## **4.5. Class Diagram**

* Entry and exit conditions are placed before the flow of events for understandability.
* Aggregations of ToDoList-User, StudentInformation-Student are converted to compositions.
* Aggregations of Student-WaitingBin, Wish-Form, HostCourse-BilkentCourse are converted to just associations.
* Wish class is further explained.
* Syllabus class is further explained.

## **4.6. State Diagrams**

### **4.6.1. State Diagram of Erasmus Process for Student**

* The transitions between “WaitingPreApproval” and “StudentOnBoard” states were updated.

### **4.6.2. State Diagram of Course Transfer Process**

* A new state diagram for the course transfer process was created and the related description of the state diagram was written. The old state diagram for the course transfer process was replaced by the new one.

## **4.7. Activity Diagram**

* The start node was added to the Activity Diagram for the placement process for students.
* The start node was added to the Activity Diagram for Pre Approval process.
* The start node was added to the Activity Diagram for the course transfer process
* The flow of events was corrected and ambiguity is resolved.
* Involving actors were modified.

## **4.8. Sequence Diagram**

* The usage of <<create>> was corrected.
* The approvePlacement call was modified.
* The sequence of the events was corrected.
* Returning arrows are added.
* Involving actors are modified.

## **4.9. User Interface Mock-Ups**

* Removed the unrelated text bodies such as “Lorem Ipsum…” and added relevant content.
* Edited Student Wish List page mock-up and added the table where the previously accepted courses can be selected.

# 

# **5. References**

[1] P. Atack, “What is the European Credit Transfer System (ECTS)?,” Study.eu. [Online]. Available: <https://www.study.eu/article/what-is-the-ects-european-credit-transfer-and-accumulation-system>. [Accessed: Nov 04, 2022].

[2] “European Credit Transfer and Accumulation System (ECTS),” European Education Area. [Online]. Available: <https://education.ec.europa.eu/education-levels/higher-education/inclusive-and-connected-higher-education/european-credit-transfer-and-accumulation-system>. [Accessed: Nov 04, 2022].