Bilkent University

Department of Computer Engineering

CS319 Term Project

Fall 2022

Section 1

Group 1B

Analysis Report

Group members:

Ayşe Kelleci 21902532

Yusuf Doğan 21702970

Zeynep Hanife Akgül 22003356

Kardelen Ceren 22003017

Melisa Tanrıkulu 21703437

Instructor:

Eray Tüzün

Contents

1. Introduction

Currently, Bilkent students leaving for a foreign university through the Erasmus program need to email their coordinators frequently for the course selection and approval process. Instead, we propose a new website in order to decrease the workload of coordinators, make it easier for both students and coordinators to track their files, and increase communication and information sharing without emails. Through this website, the placement of students to their preferred universities is done automatically, the students may choose their courses, the coordinators may approve or reject them, the relevant documents may be generated, and students and coordinators may message each other or create posts in the forum that all users can see and reply to. Overall, the goal is to create a user-friendly website where outgoing students and coordinators can access and handle all their Erasmus-related tasks.

2. Proposed System

2.1. Functional Requirements

2.1.1. User Types

There will be three user types on the site. “Student” users refer to students who applied for the Erasmus program. “Coordinator” users refer to Erasmus department coordinators in Bilkent who are responsible for evaluating the courses the students choose during their mobility period (i.e. the semester(s) they go to the foreign university). “Board member” users are the faculty board members who approve the course transfers along with the coordinator. Board members are very restricted in what they can do or see on the site in an effort to decrease the process’ complexity. In our system there is no user type for lecture instructors, since we interact with them only at merging courses and we deal with that issue by e-mail communication by our system.

2.1.2. Login

All students applying for Erasmus have accounts that are created by the system administrator. Students will be given an initial password paired with their Bilkent emails, which they will be advised to change after they log in. Coordinator and board member accounts will be created by the administrator in the same way. If a user forgets their password, they can change it through their email.

2.1.3. Profile

All users will have a profile page through which they can change their contact information and passwords. They can also choose whether to display their information on their public profile.

2.1.4. Forum

There will be a forum feature accessible through the main page. The forum will be used to create posts that all students and coordinators can see and reply to. For example, a student may post their questions about the visa process for a certain country and other students who will go or who have gone to the same country may help by replying.

The posts' subject lines will be listed on the forum page, as well as the person who created the post, the last person who replied to the post, and the number of replies to the post. After they click on the post, users can see the main text of the post and the replies it received. Through the same page, they can write a reply and by clicking on the name of the people who sent a reply, users can access their profiles.

2.1.5. FAQ

FAQ (frequently asked questions) page will be available on the site. While students will have no authority to change the FAQ, the coordinator will be able to add questions along with answers to the FAQ or remove questions from this part. FAQ is necessary to give a general understanding of the Erasmus process to students and answer their most common questions without emails.

2.1.6. Messages

Messaging is a way of direct communication between the coordinator and students. On the left side of the messaging page, there will be a list of people chronologically ordered from the most recent person to the first person that the user messaged. New messages will appear here as well. Users can access other users´ profiles from that list. To continue an existing chat, users can click one of those messages from the list. To start a new chat, the user will click the "new chat" button and select a person from a list of users, which they can search or filter. Coordinators and students can start messaging with

each other or between themselves but board members can only message with coordinators. Users in the messaging screen can see some info about whom they chat. Also, the users will be able to filter messages. This feature aims to reduce the amount of time spent on searching messages. Users can filter by name, the university applied for Erasmus, or the country and sort alphabetically or chronologically.

2.1.7. Choosing Courses

Students have to add courses that they will take in their Erasmus semester. The Erasmus Application system aims to enable students to conduct this process in an efficient way. Course-related operations can be accessed everywhere from the navigation bar through the “Courses” title. Students are able to see the number of approved courses that they are planning to take with course name, id (such as CS319), ECTS credit, and their equivalent courses in Bilkent. Since this data is provided to us. The total amount of ECTS credits will be at the bottom of the list. At the bottom of the page, there will be two buttons, one of which is “add previously approved courses”. Through this button, students will be able to view and choose courses that were previously accepted by Bilkent. The other button, “add a new course”, will be used if the student wants to take a course that was not previously approved by Bilkent. If the new course is the equivalent of an elective course in Bilkent, the coordinator will be notified to either approve or reject the course. However, if the new course will be taken instead of a mandatory course in Bilkent, the instructor of the said course will be sent an email by the system and the instructor’s response will be shown to the coordinator, which will then either approve or reject the course. If the coordinators reject a course, they may choose to include a reason. In addition, courses with low credit can be merged to be equivalent with courses that have high credit, which requires the coordinator´s approval as well. This operation is available under the selected course list.

2.1.8. To-Do List

Students and coordinators will have a to-do list on their homepage. The content of this for students is a list of actions to be done in the Erasmus process. Also, students can remove tasks from the list and add new tasks to their to-do list at any time. The coordinator´s to-do list will have a list of waiting for course approvals. Likewise, a coordinator can remove a task or add a new task to their list. When a task is done, it will be moved to the completed items list on the main page.

2.1.9. Notification

All users have a notification panel. Notifications can be accessed from the main page of the site. Students and coordinators can be notified about new messages, upcoming events, approvals/rejections, new answers to their posts in the forum, and successful document uploading/downloading.

2.1.10. File Handling

In the Erasmus application process, students deal with some documents such as the pre-approval form, the learning agreement, or the course transfer form. Our site will enable students and coordinators to generate such documents using their planned courses, stored on the site. In addition, they will be able to download these documents in .doc or .pdf formats and upload their own documents at any time. To sign, users will need to download the generated document first, add their signature images manually or by using an e-signature app, and upload the signed documents to the system. Moreover, if a change is made to a file, it will optionally be highlighted to make it easier to spot differences.

2.1.11. Student List

The coordinator will be able to access the list of all students. Students will be displayed with their information and clicking on their name will direct the page to their profile. There will be a button next to each student’s name

that will direct the coordinator to the message page for that student. The coordinator can see the responsible coordinator for a student on the list. Students may be filtered based on their applied university, mobility period, etc.

2.1.12. Status

On the student's main page, their current status will be shown. The status is the current step in the student's Erasmus process, such as waiting for approval from the coordinator, uploading a specific file, and so on. The status of students can also be seen from their profile by the coordinator.

2.1.13. Placement

Placement will be done after the coordinator uploads the student table containing students' scores and university preferences. The accounts are created at this stage. Students will then be placed in their preferred universities, starting with the top-ranking student to the bottom-ranking. If the student's first preference's quota is already reached, then their next preference is considered. After the initial placements are done, students are informed about their placement through an in-system notification. They then have the option to cancel their placement. In that case, there will be no reshuffling for the canceled placement; instead, the top-ranking student without a placement is offered the place. Depending on the student's answer, either the student is placed or the next top-ranking student is asked. After the cancellation deadline, all placements are final.

2.2. Non-functional Requirements [3]

2.2.1. User Friendliness

The site will have an intuitive user interface to make it easy for users to understand the fundamentals of the program since the site will keep interacting with users throughout its life. For this purpose, we decided to have a navigation-bar that is visible on all pages, which makes passing between different pages easy. We aim our target audience, university students and instructors, to be able to use our site without any guidance.

2.2.2. Extensibility

In this system, Object Oriented Model will be utilized and the architecture of the system that we designed all with models and diagrams will allow developers to extend and update it later on when required. In order to ensure extensibility, the code will be properly commented and documented. Likewise, our ui can be integrated with new features. We can add new features to navbar and make it as accessible as the other features.

2.2.3. Privacy

All personal information (password, GPA, messages etc.) about users will be kept secret. Students will also be given the option to share information about themselves to other students like phone number, email address, university, mobility period, and other information.

2.3. System Models

2.3.1. Use-Case Model

The use case diagram given above shows the general structure of our web-based Erasmus Application. There are 37 use cases. ERASMUS refers to the system behind our website.

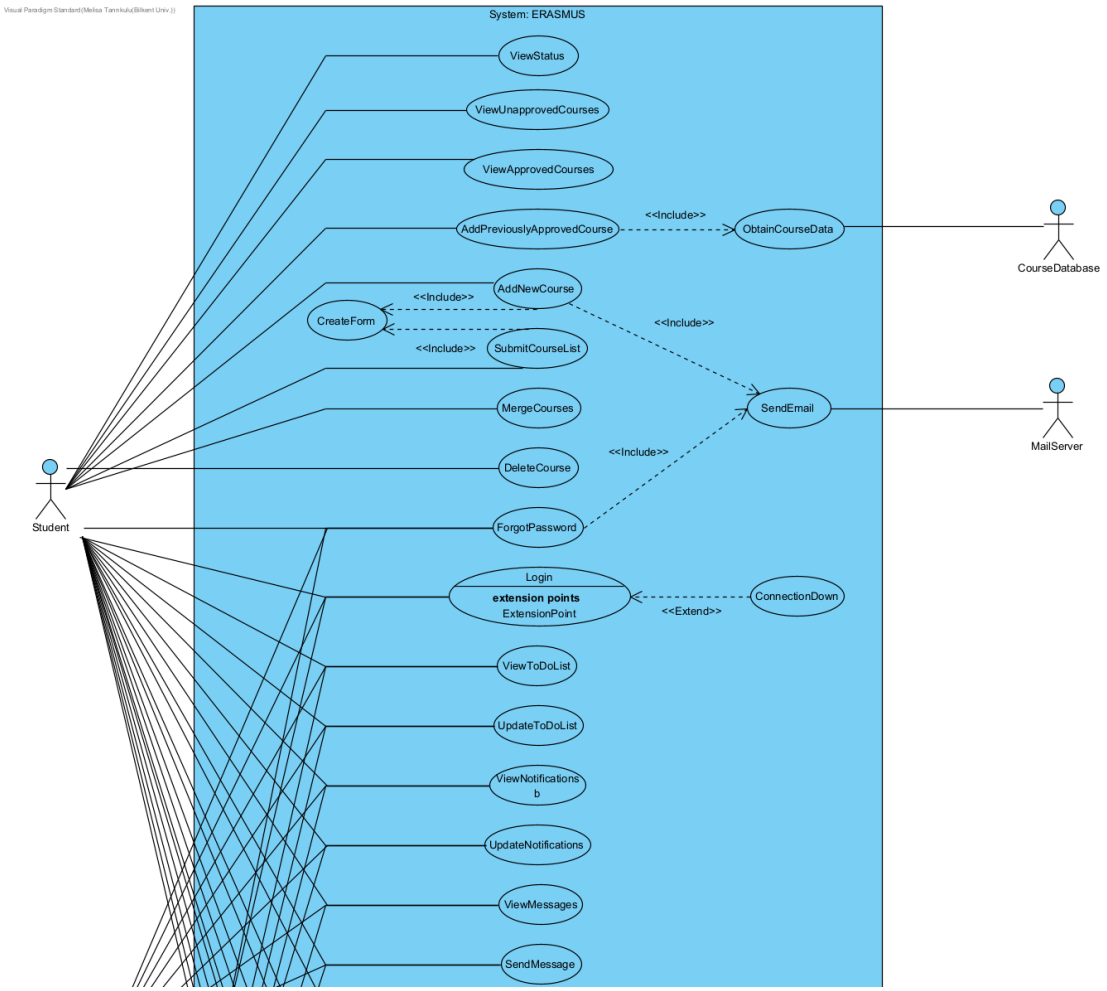


Fig. 2.3.1.1.: Use Case Diagram Part 1

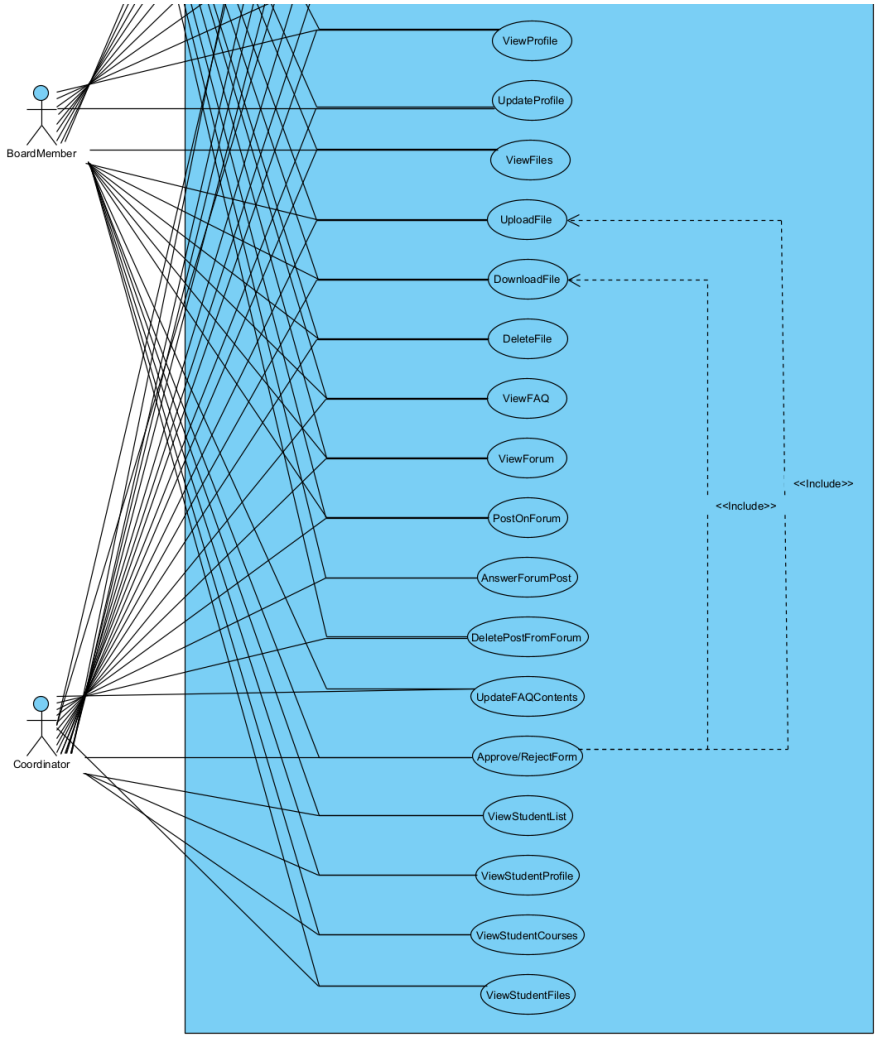


Fig. 2.3.1.2.: Use Case Diagram Part 2

Use Case # 1

---

1. **Name:** Login
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR has an existing account in the system.
4. **Exit condition:**
    - The ACTOR gains access to the system to use any functionality assigned to them.
5. **Flow of events:**
    1. The ACTOR activates the “Login” function of their terminal.
    2. ERASMUS responds by requesting the ACTOR’s Bilkent ID and password.
    3. The ACTOR enters their Bilkent ID and password.
    4. ERASMUS compares the given login information with the ones in the database. If they match, the ACTOR is authenticated. Otherwise, the terminal rejects the authentication attempt.
6. **Special/quality requirements:**
    - ERASMUS connects to the server within 30 seconds.

Use Case # 2

---

1. **Name:** ConnectionDown
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - This use case extends the Login use case. It is initiated by the system whenever the network connection between the ACTOR and the system is lost.

4. **Exit condition:**
    - The ACTOR could not log in to ERASMUS.
    - The ACTOR is informed about the status.
5. **Flow of events:**
    1. The ACTOR is specified as not able to log in, and is informed about the status.

Use Case # 3

---

1. **Name:** ForgotPassword
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR has an existing account in the system.
4. **Exit condition:**
    - The ACTOR's password changes.
    - The ACTOR is notified about the password change.
5. **Flow of events:**
    1. The ACTOR activates the "Forgot Password" function of their terminal.
    2. ERASMUS responds by requesting the ACTOR's Bilkent ID.
    3. The ACTOR enters their Bilkent ID.
    4. ERASMUS responds by sending a message including a password reset link to the email, which belongs to the given Bilkent ID account.
    5. The ACTOR clicks on the link and enters a new password as an input on the directed page and submits it.
    6. ERASMUS updates the information in the database and notifies the user.

6. **Special/quality requirements:**
    - This use case includes the SendEmail use case. The SendEmail use case is initiated when the ERASMUS invokes the send email function to enable the user to change their password.

Use Case # 4

---

1. **Name:** ViewStatus
2. **Participating actors:** Student
3. **Entry condition:**
    - STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - STUDENT is notified about their status.
5. **Flow of events:**
    1. STUDENT activates the “View Status” function of their terminal.
    2. ERASMUS responds by displaying the status of the student.

Use Case # 5

---

1. **Name:** SendEmail
2. **Participating actors:** MailServer
3. **Entry condition:**
    - This use case is included in the ForgotPassword and AddNewCourse use cases. In the ForgotPassword use case, it is initiated by the system when the user requests password change. In the AddNewCourse use case, it is initiated by the system when the user wants to add a new course to inform the Instructor of the course which is an equivalent of the new course.
4. **Exit condition:**
    - An email message is sent to the user’s email.

5. **Flow of events:**
    1. ERASMUS connects to the mail server and requests to send an email message to the user with the given content.
    2. The MAILSERVER receives the request and sends the email message.

Use Case # 6

---

1. **Name:** ViewToDoList
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR has an existing account in the system.
4. **Exit condition:**
    - The ACTOR views the To-Do and Done lists.
5. **Flow of events:**
    1. The ACTOR activates the “View To-Do List” function of their terminal.
    2. ERASMUS responds by presenting the to-do list and done list content of the ACTOR.

Use Case # 7

---

1. **Name:** UpdateToDoList
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR has an existing account in the system.
4. **Exit condition:**
    - The ACTOR updates the to-do list/done list.
    - The ACTOR is informed about the update.

5. **Main Flow of events:**
    1. The ACTOR activates the “Add New To-Do Item” function of their terminal.
    2. ERASMUS responds by presenting a pop-up to the ACTOR. The pop-up includes a form with input placeholders for the name of the to-do item, and a link.
    3. The ACTOR completes the form by specifying minimally the name of the to-do item and submits the form.
    4. ERASMUS receives the form, updates the database and displays the updated list.
6. **Alternative Flow of events:**
    1. The ACTOR wants a to-do list item to be marked as done.
        1. The ACTOR selects a to-do item from the list and specifies it as done.
        2. ERASMUS responds by moving the to-do item to the done list in the database, and displaying the updated lists.
    2. The ACTOR wants a to-do list item to be flagged.
        1. The ACTOR selects a to-do item from the list and specifies it as flagged.
        2. ERASMUS responds by marking the selected to-do item as flagged in the database and displays the updated list.
    3. The ACTOR wants a to-do list item to be deleted.
        1. The ACTOR selects a to-do item from the list and specifies it as to be deleted.
        2. ERASMUS responds by deleting the item from the to-do list in the database, and displays the updated list.

4. The ACTOR wants a done list item to be moved to the to-do list.
    1. The ACTOR selects a done list item and specifies it as to-do item.
    2. ERASMUS responds by moving the done list item to the to-do list in the database and displays the updated lists.

Use Case # 8

---

1. **Name:** ViewNotifications
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views the notifications.
5. **Flow of events:**
    1. The ACTOR activates the “View Notifications” function of their terminal.
    2. ERASMUS responds by presenting the content of the notifications list of the ACTOR.

Use Case # 9

---

1. **Name:** UpdateNotifications
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR updates the status of a notification.
    - The ACTOR is informed about the update.

5. **Main Flow of events:**
    1. The ACTOR selects a notification and specifies it as read.
    2. ERASMUS responds by marking the notification as read in the database and displays the updated list.
6. **Alternative Flow of events:**
    1. The ACTOR wants to delete a notification.
        1. The ACTOR selects a notification and specifies it as to be deleted.
        2. ERASMUS responds by deleting the notification from the database and displays the updated list.
    2. The ACTOR wants to flag a notification.
        1. The ACTOR selects a notification and specifies it as flagged.
        2. ERASMUS responds by marking the notification as flagged in the database and displays the updated list.
    3. The ACTOR wants to be directed to the page associated with the notification.
        1. The ACTOR selects the notification link.
        2. ERASMUS responds by directing the ACTOR to the link included within the notification.

Use Case # 10

---

1. **Name:** ViewMessages
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views their messages.

5. **Flow of events:**
    1. The ACTOR activates the “View Messages” function of their terminal.
    2. ERASMUS responds by presenting the list of contacts that the ACTOR has received/sent any message from/to.
    3. The ACTOR selects a contact from the list.
    4. ERASMUS responds by presenting the message content with the selected contact and the information (full name, phone number, e-mail, etc.) of that contact.
    5. The ACTOR reads the message(s) they received from that contact.
    6. ERASMUS specifies the messages as read if not already.

Use Case # 11

---

1. **Name:** SendMessage
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR sends a message to a contact.
    - The contact receives a notification about the message.
5. **Main Flow of events:**
    1. The ACTOR activates the “Send Message” function of their terminal.
    2. ERASMUS responds by presenting the list of contacts that the ACTOR has received/sent any message from/to.
    3. The ACTOR selects a contact from the list.
    4. ERASMUS responds by providing an input placeholder and displaying the information (full name, phone number, e-mail, etc.) of that contact and the messages with that contact.

5. The ACTOR writes their message on the input.
6. ERASMUS updates the database, sends a notification to the contact and displays the updated message content.

6. **Alternative Flow of events:**
    1. The ACTOR wants to create a new chat.
        1. The ACTOR activates the “New Chat” function of their terminal.
        2. ERASMUS responds by displaying the list of possible contacts to create a new chat.
        3. The ACTOR selects a contact from the list.
        4. ERASMUS responds by providing an input placeholder and displaying the information (full name, phone number, e-mail, etc.) of that contact.
        5. The ACTOR writes their message on the input.
        6. ERASMUS updates the database, sends a notification to the contact and displays the updated message content..

Use Case # 12

---

1. **Name:** ViewProfile
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views their profile.
5. **Flow of events:**
    1. The ACTOR activates the “View Profile” function of their terminal.
    2. ERASMUS responds by displaying the contents of the ACTOR’s profile (full name, email, ID, etc.).

Use Case # 13

---

1. **Name:** UpdateProfile
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR updates their profile.
    - The ACTOR is informed about the update.
5. **Main Flow of events:**
    1. The ACTOR activates the “Change Password” function of their terminal.
    2. ERASMUS responds by presenting a form to the ACTOR. The form requires the ACTOR to enter their old password, and the new password.
    3. The ACTOR fills the form and submits it.
    4. ERASMUS updates the database and informs the ACTOR.
6. **Alternative Flow of events:**
    1. The ACTOR wants to change their email.
        1. The ACTOR activates the “Change Email” function of their terminal.
        2. ERASMUS responds by presenting a form to the ACTOR. The form requires the ACTOR to enter their old email, and their new email.
        3. The ACTOR fills the form and submits it.
        4. ERASMUS updates the database and informs the ACTOR.

2. The ACTOR wants to change their preferences.
    1. The ACTOR activates the “Change Preferences” function of their terminal.
    2. ERASMUS responds by presenting their preferences to the ACTOR.
    3. The ACTOR updates their preferences.
    4. ERASMUS updates the database, displays the updated preferences and informs the ACTOR.

· **Use Case # 14**

---

1. **Name:** ViewFiles
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views their file.
5. **Main Flow of events:**
    1. The ACTOR activates the “View Files” function of their terminal.
    2. ERASMUS responds by presenting the list of files the ACTOR has uploaded.

6. **Alternative Flow of events:**
    1. The ACTOR wants to see the contents of a file.
        1. The ACTOR activates the “See File” function of their terminal.
        2. ERASMUS responds by presenting the list of files the ACTOR has uploaded and an input placeholder for the user to enter a name of a file.
        3. The ACTOR enters the name of a file.
        4. ERASMUS responds by providing the visual content of the file.

· **Use Case # 15**

---

1. **Name:** UploadFile
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR uploads a new file.
    - The ACTOR is informed about the process.
5. **Flow of events:**
    1. The ACTOR activates the “Upload File” function of their terminal.
    2. ERASMUS responds by presenting an input placeholder for the ACTOR to enter the directory of their file.
    3. The ACTOR enters the directory of the file.
    4. ERASMUS updates the database, displays the updated file list and informs the user.

Use Case # 16

---

1. **Name:** DownloadFile
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR downloads a file.
    - The ACTOR is informed about the process.
5. **Flow of events:**
    1. The ACTOR activates the “Download File” function of their terminal.
    2. ERASMUS responds by presenting the list of files the ACTOR has uploaded and an input placeholder for the user to enter a name of a file.
    3. The ACTOR enters the name of a file to download.
    4. ERASMUS responds by enabling the download process and informs the user.

Use Case # 17

---

1. **Name:** DeleteFile
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR deletes a file from the uploaded file list.
    - The ACTOR is informed about the change.

5. **Flow of events:**
    1. The ACTOR activates the “Download File” function of their terminal.
    2. ERASMUS responds by presenting the list of files the ACTOR has uploaded and an input placeholder for the user to enter a name of a file.
    3. The ACTOR enters the name of a file to delete.
    4. ERASMUS responds by enabling the download process and informs the user.
    5. ERASMUS responds by deleting the file from the database, displays the updated file list and informs the user.

Use Case # 18

---

1. **Name:** ViewFAQ
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views the contents of the FAQ.
5. **Flow of events:**
    1. The ACTOR activates the “View FAQ” function of their terminal.
    2. ERASMUS responds by presenting the FAQ content.

Use Case # 19

---

1. **Name:** UpdateFAQContents
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.

4. **Exit condition:**
    - The ACTOR changes the contents of the FAQ.
    - The ACTOR is informed about the update.
5. **Main Flow of events:**
    1. The ACTOR activates the “Update FAQ Contents” function of their terminal.
    2. ERASMUS responds by presenting the FAQ content.
    3. The ACTOR selects an item and specifies it as to be deleted.
    4. ERASMUS responds by deleting the selected item from the database and displays the updated FAQ list.
6. **Alternative Flow of events:**
    1. The ACTOR wants to add a new item to the FAQ list.
        1. The ACTOR activates the “Post New FAQ” function of their terminal.
        2. ERASMUS responds by presenting a form to the ACTOR. The form includes the question and the corresponding answer.
        3. The ACTOR fills out the form and submits it.
        4. ERASMUS updates the database and displays the updated post list.

Use Case # 20

---

1. **Name:** ViewForum
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views the forum.

5. **Main Flow of events:**
    1. The ACTOR activates the “View Forum” function of their terminal.
    2. ERASMUS responds by presenting the Forum content.
6. **Alternative Flow of events:**
    1. The ACTOR wants to view a Forum post.
        1. The ACTOR activates the “View Forum Post” function of their terminal.
        2. ERASMUS responds by displaying the list of posts on the Forum and an input placeholder for the user to enter a post name.
        3. The ACTOR enters a post name.
        4. ERASMUS responds by displaying the contents of the Forum post.

Use Case # 21

---

1. **Name:** PostOnForum
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR creates a new post on the forum.
    - The ACTOR is informed about the update.

5. **Flow of events:**
    1. The ACTOR activates the “Create New Forum Post” function on their terminal.
    2. ERASMUS responds by presenting a form to the ACTOR. The form includes the subject and the content of the post.
    3. The ACTOR fills the form and submits it.
    4. ERASMUS receives the form, updates the database and displays the updated forum content.

Use Case # 22

---

1. **Name:** AnswerForumPost
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR sends a comment on a post on the forum.
    - The ACTOR is informed about the update.
5. **Flow of events:**
    1. The ACTOR activates the “Answer Forum Post” function of their terminal.
    2. ERASMUS responds by displaying the list of posts on the Forum and an input placeholder for the user to enter a post name.
    3. The ACTOR enters a post name.
    4. ERASMUS responds by displaying the post content and a form to the ACTOR. The form includes the content of the answer.
    5. The ACTOR fills the form and submits it.
    6. ERASMUS receives the form, updates the database and displays the updated post content.

Use Case # 23

---

1. **Name:** DeletePostFromForum
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR deletes a post from the forum.
    - The ACTOR is informed about the update.
5. **Flow of events:**
    1. The ACTOR activates the “Delete Forum Post” function of their terminal.
    2. ERASMUS responds by displaying the list of posts on the Forum that the user can delete and an input placeholder for the user to enter a post name.
    3. The ACTOR enters a post name.
    4. ERASMUS responds by deleting the post from the database and displays the updated list of posts.

Use Case # 24

---

1. **Name:** ViewUnapprovedCourses
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT views their currently added courses that haven't been approved.

5. **Flow of events:**
    1. STUDENT activates the “View Unapproved Courses” function of their terminal.
    2. ERASMUS responds by presenting the list of the courses the user added but haven't been approved and their information (course name, course ID, ECTS credits, the course equivalent to it according to Bilkent courses, ECTS total credits) which the STUDENT added.

Use Case # 25

---

1. **Name:** ViewApprovedCourses
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT views their added courses that have been approved.
5. **Flow of events:**
    1. STUDENT activates the “View Approved Courses” function of their terminal.
    2. ERASMUS responds by presenting the list of the courses the user added and have been approved, and their information (course name, course ID, ECTS credits, the course equivalent to it according to Bilkent courses, ECTS total credits) which the STUDENT added.

Use Case # 26

---

1. **Name:** AddPreviouslyApprovedCourse
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT adds a previously approved course.
    - The STUDENT is informed about the update.
5. **Flow of events:**
    1. STUDENT activates the “Add Previously Approved Course” function of their terminal.
    2. ERASMUS responds by presenting the list of previously approved courses.
    3. The STUDENT selects a course they want to add.
    4. ERASMUS responds by adding the selected course to the unapproved course list.

Use Case # 27

---

1. **Name:** ObtainCourseData
2. **Participating actors:** CourseDatabase
3. **Entry condition:**
    - This use case is included in the AddPreviouslyApprovedCourse use case. It is initiated by the system when the user requests to add a previously approved course.
4. **Exit condition:**
    - The course data is pulled from the course database.

5. **Flow of events:**
    1. ERASMUS connects to the course database and requests to pull the previously approved course list.
    2. The COURSEDATABASE receives the request and sends the database content.

Use Case # 28

---

1. **Name:** AddNewCourse
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT adds a new course.
    - The STUDENT is informed about the update.
5. **Main Flow of events:**
    1. STUDENT activates the “Add New Course” function of their terminal.
    2. ERASMUS responds by presenting a form to the ACTOR. The form includes the course name, course code, ECTS credit, the equivalent course and its instructor’s email address, and the link of the course syllabus.
    3. STUDENT fills the form and submits it.
    4. ERASMUS responds by creating a form with the course information the user provided and sends an email message to the email address the user has provided.
    5. ERASMUS receives the email sent from the instructor, updates the list of previously approved courses, adds the course to the user’s course list which haven't been approved and informs the user that the request was approved.

6. **Alternative Flow of events:**
    1. The course request is rejected.
        1. STUDENT activates the “Add New Course” function of their terminal.
        2. ERASMUS responds by presenting a form to the ACTOR. The form includes the course name, course code, ECTS credit, the equivalent course and its instructor’s email address, and the link of the course syllabus.
        3. STUDENT fills the form and submits it.
        4. ERASMUS responds by creating a form with the course information the user provided and sends an email message to the email address the user has provided.
        5. ERASMUS receives the email sent from the instructor and informs the user that the request was rejected.
7. **Special/quality requirements:**
    - This use case includes the SendEmail and CreateForm use cases. The SendEmail use case is initiated when the ERASMUS invokes the send email function to enable

Use Case # 29

---

1. **Name:** MergeCourses
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT merges selected courses as one.
    - The STUDENT is informed about the process.

5. **Main Flow of events:**
    1. STUDENT activates the “Merge Courses” function of their terminal.
    2. ERASMUS responds by presenting the list of added but unapproved courses which are appropriate to merge.
    3. STUDENT selects the courses to merge.
    4. ERASMUS responds by merging the course as one, updating the database and informing the user.

Use Case # 30

---

1. **Name:** DeleteCourse
2. **Participating actors:** Student
3. **Entry condition:**
    - The STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - The STUDENT deletes a course from the added but unapproved course list.
    - The STUDENT is informed about the change.
5. **Flow of events:**
    1. STUDENT activates the “Delete Course” function of their terminal.
    2. ERASMUS responds by presenting the list of added but unapproved courses.
    3. STUDENT selects courses to delete.
    4. ERASMUS responds by deleting the selected course from the list, and displaying the updated list.

Use Case # 31

---

1. **Name:** SubmitCourseList
2. **Participating actors:** Student
3. **Entry condition:**
    - STUDENT is logged into ERASMUS.
4. **Exit condition:**
    - STUDENT submits the course list.
    - COORDINATOR is informed about the Course Approval form.
5. **Flow of events:**
    1. STUDENT activates the “Submit Courses” function of their terminal.
    2. ERASMUS creates the Course Approval form with the given course list, adds the form to the Student’s Files as visible to the user, the COORDINATOR and informs the COORDINATOR.
6. **Special/quality requirements:**
    - This use case includes the CreateForm use cases. The CreateForm use case is initiated when the ERASMUS invokes create form function to create the Course Approval form.

Use Case # 32

---

1. **Name:** Approve/RejectForm
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
    - The STUDENT submitted the course list.
4. **Exit condition:**
    - The ACTOR approved/rejected the form.
    - STUDENT is informed about the status of the file.

5. **Main Flow of events:**
    1. The ACTOR activates the “View Form” function of their terminal.
    2. ERASMUS responds by displaying the list of students which submitted their Course Approval form.
    3. The ACTOR responds by selecting a student from the list.
    4. ERASMUS responds by initiating the download process for the file.
    5. The ACTOR views the form, adds their digital sign to the file and initiates the “Approve Form” function of their terminal.
    6. ERASMUS responds by providing an input placeholder for the directory of the updated file.
    7. The ACTOR enters the directory of the updated file.
    8. ERASMUS responds by updating the files list, moving the unapproved added courses to the approved courses list and informing the STUDENT.

6. **Alternative Flow of events:**
    1. The course request is rejected.
        1. The ACTOR activates the “View Form” function of their terminal.
        2. ERASMUS responds by displaying the list of students which submitted their Course Approval form.
        3. The ACTOR responds by selecting a student from the list.
        4. ERASMUS responds by initiating the download process for the file.
        5. The ACTOR views the form and initiates the “Reject Form” function of their terminal.
        6. ERASMUS responds by providing an input placeholder for the rejection reason.
        7. The ACTOR enters the rejection reason.

8. ERASMUS responds by informing the STUDENT about the rejection.

7. **Special/quality requirements:**
    - This use case includes the DownloadFile and UploadFile use cases. The DownloadFile use case is initiated when the COORDINATOR/BOARDMEMBER wants to download the Course Approval form. The UploadFile is initiated when the COORDINATOR/BOARDMEMBER wants to upload the signed form.

Use Case # 33

---

1. **Name:** CreateForm
2. **Participating actors:** Student, Coordinator, BoardMember
3. **Entry condition:**
    - This use case is included in the AddNewCourse and SubmitCourseList use case. In the AddNewCourse use case, it is initiated by the system when the user requests to add a new course and a form is needed to be created to send to the instructor. In the SubmitCourseList use case, it is initiated by the system when a Course Approval form is needed to be created.
4. **Exit condition:**
    - A form with given information is created.
5. **Flow of events:**
    1. ERASMUS creates a form with the given content.

Use Case # 34

---

1. **Name:** ViewStudentList
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views the student list.
5. **Flow of events:**
    1. The ACTOR activates the “View Student List” function of their terminal.
    2. ERASMUS responds by presenting the student list to the actor.

Use Case # 35

---

1. **Name:** ViewStudentProfile
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views a student’s profile.
5. **Flow of events:**
    1. The ACTOR activates the “View Student Profile” function of their terminal.
    2. ERASMUS responds by presenting the student list to the actor.
    3. The ACTOR selects a student from the list.
    4. ERASMUS responds by displaying the student’s information (email, phone number, university/mobility period, status and ID).

Use Case # 36

---

1. **Name:** ViewStudentCourses
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views a student's added and approved courses.
5. **Flow of events:**
    1. The ACTOR activates the "View Student's Courses" function of their terminal.
    2. ERASMUS responds by presenting the student list to the actor.
    3. The ACTOR selects a student from the list.
    4. ERASMUS responds by displaying the list of courses the student has added and approved.

Use Case # 37

---

1. **Name:** ViewStudentFiles
2. **Participating actors:** Coordinator, BoardMember
3. **Entry condition:**
    - The ACTOR is logged into ERASMUS.
4. **Exit condition:**
    - The ACTOR views a student's uploaded files.
5. **Flow of events:**
    1. The ACTOR activates the "See Student's Files" function of their terminal.
    2. ERASMUS responds by presenting the student list to the actor.
    3. The ACTOR selects a student from the list.
    4. ERASMUS responds by displaying the list of files that belong to the student and is visible for the user to see.

2.3.2. Object and Class Models

The class diagram given above shows the general structure of our web-based Erasmus Application. There are 19 classes.

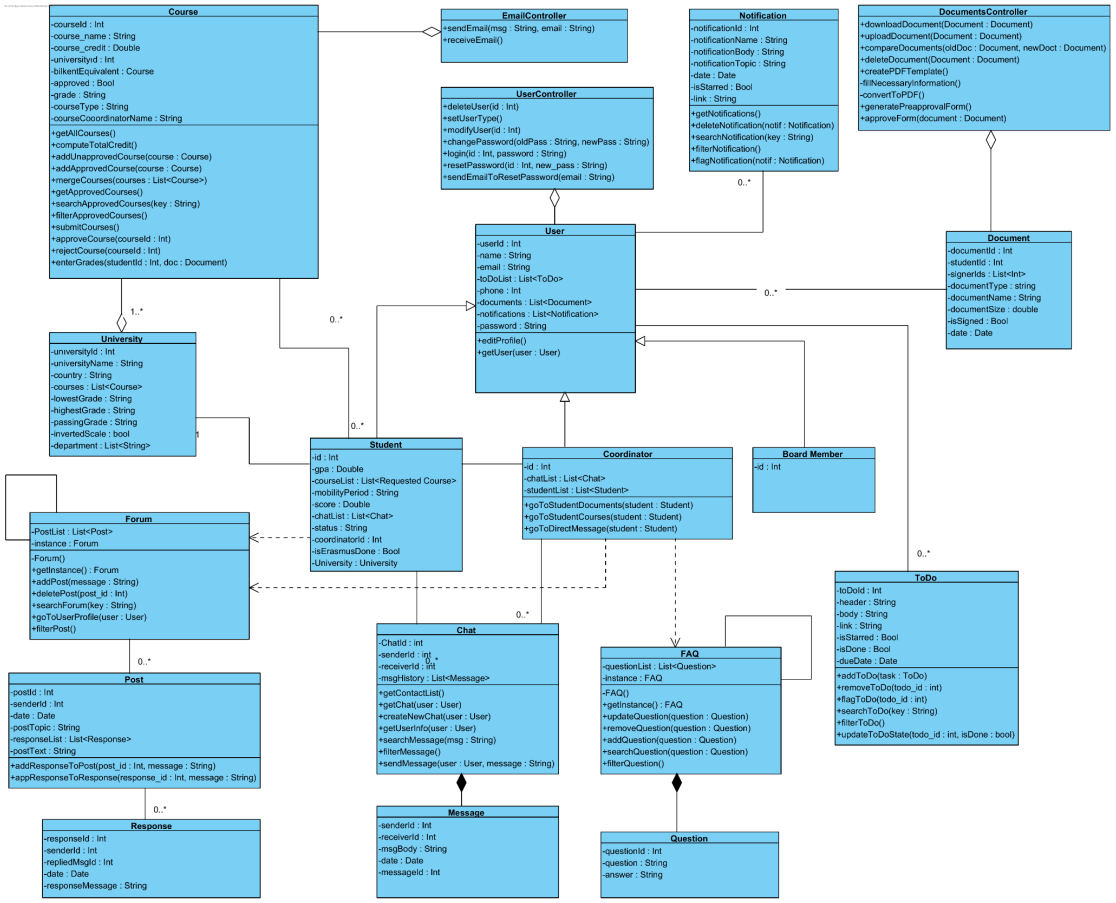


Fig. 2.3.2.1.: Object and Class Diagram

User Class: User class includes all user types and common aspects such as name, id, and email.

Student Class: The student class is a subclass of the user class. It includes students who applied to Erasmus. Students should be able to choose courses they will take in Erasmus and its equivalent at Bilkent University. If a preferred course is not on the course list, students can add them to the course list, and if their request is approved, they can add this course to their personal course list.

Coordinator Class: The coordinator class is a subclass of the user class. It includes Erasmus Coordinators. Coordinators are responsible for approving/rejecting courses and signing documents.

Board Member Class: The board member class is a subclass of the user class. The board member is responsible for course transfer.

Course Class: Course class carries aspects of the courses. It can be seen in the information of courses such as credit, course name, Bilkent equivalent course, and so on. Students are able to choose courses from Erasmus University and include them in their course list. Also, they can add new courses which have not been approved previously if a coordinator approves them. They can merge two or more courses if their credit is lower than required.

University Class: University class includes general information about universities and the student list for selected universities.

EmailSender Class: When adding new courses, it may be required to get approval from the coordinator of the equivalent course. In this circumstance, students can send an email to the course coordinator by using this class.

Document class: Document class is used for creating or uploading new documents.

Documents Controller: Documents controller includes necessary methods to handle documents. Users are able upload-delete documents. Also, coordinators can approve documents and can upload signed versions of documents.

Chat Class: Chat class is holding chats between users (coordinators and students.)

Message Class: Message class is for sending messages.

Notification class: Notification class is used to notify users about new updates, info, or requirements.

ToDo Class: ToDo class holds users' todos. Users can mark todos as done or add new ones.

FAQ Class: The FAQ class can be modified by coordinators. Coordinators add a new FAQ and their answers.

Question Class: The question class is a subclass of the FAQ.

User Controller: The User class is used for user management

Forum Class: Forum is a class for asking a question to the public (all Erasmus students and coordinators), and everybody can answer that.

Post Class: Post class is used for adding a new post to the forum.

Response Class: Response class is used for replying to posts or previous responses.

2.3.3. Dynamic Models

2.3.3.1. Activity Diagrams

Student placement activity diagram:

The diagram below shows the student placement process according to students' scores and preferences.

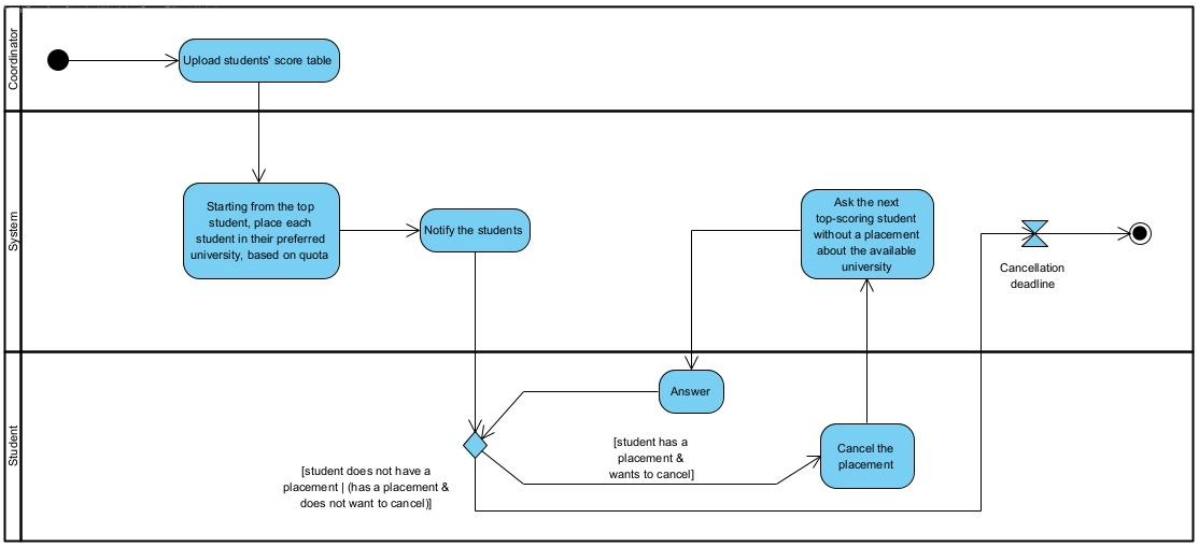


Fig. 2.3.3.1.1.: Student Placement Activity Diagram

Course choosing activity diagram:

The diagram below shows the process of students choosing courses for their mobility period, coordinators evaluating them, and signing the pre-approval and learning agreement documents that the system created.

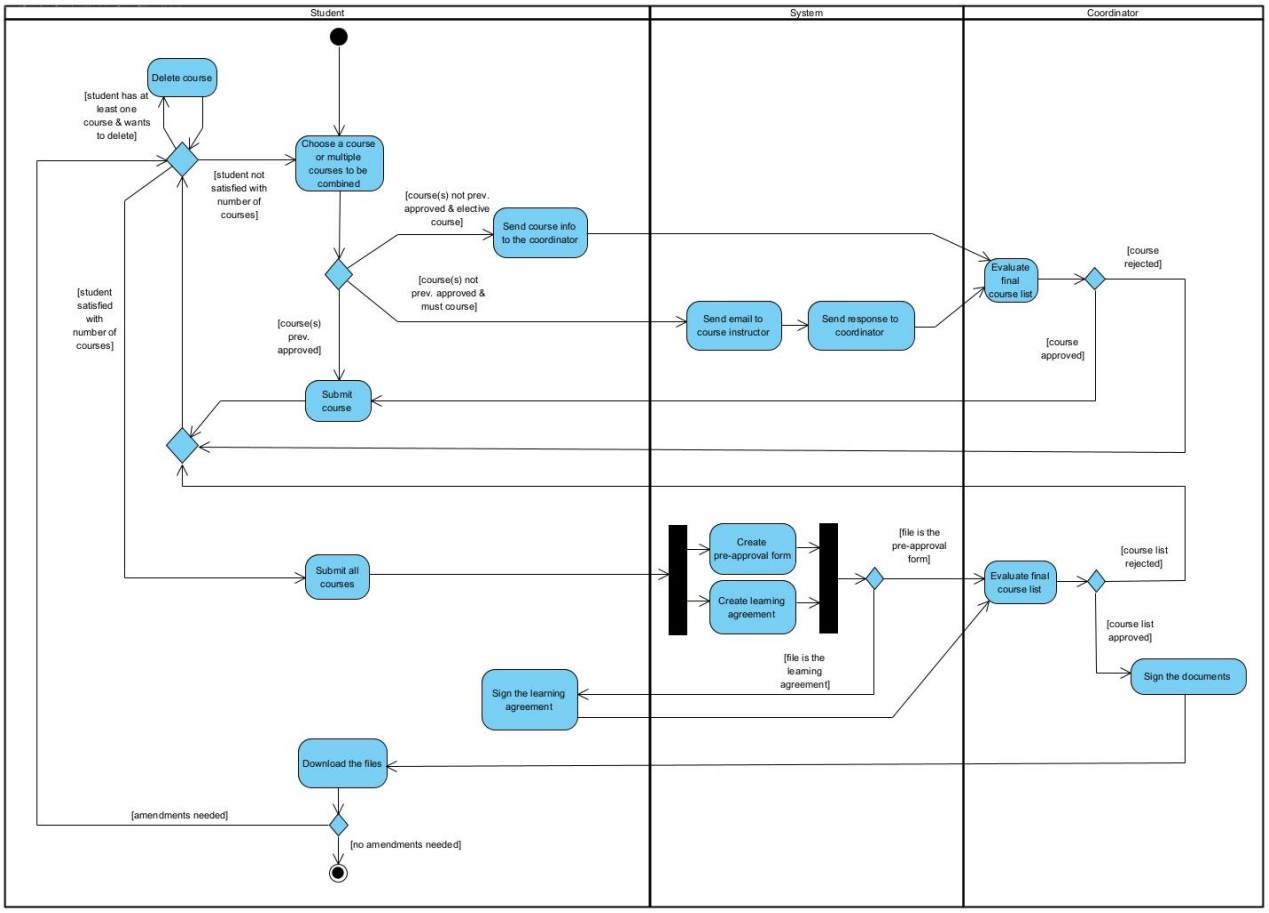


Fig. 2.3.3.1.2.: Course Choosing Activity Diagram

Course transfer activity diagram:

The diagram below shows the process in which, after the mobility period, the courses students took get transferred to Bilkent University.

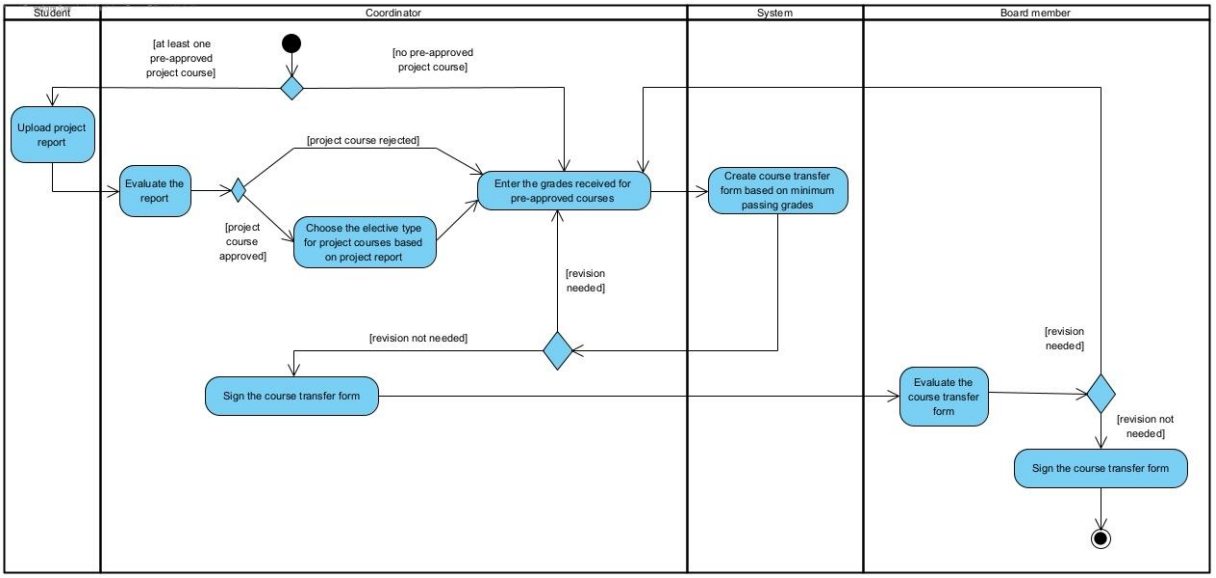


Fig. 2.3.3.1.3.: Course Transfer Activity Diagram

2.3.3.2. State Diagrams

Course list state diagram:

The diagram below shows how a student's course list's state changes. "Private" course list can only be seen by the student. Rejections and approvals are done by the coordinator and the submissions and amendments are done by the student.

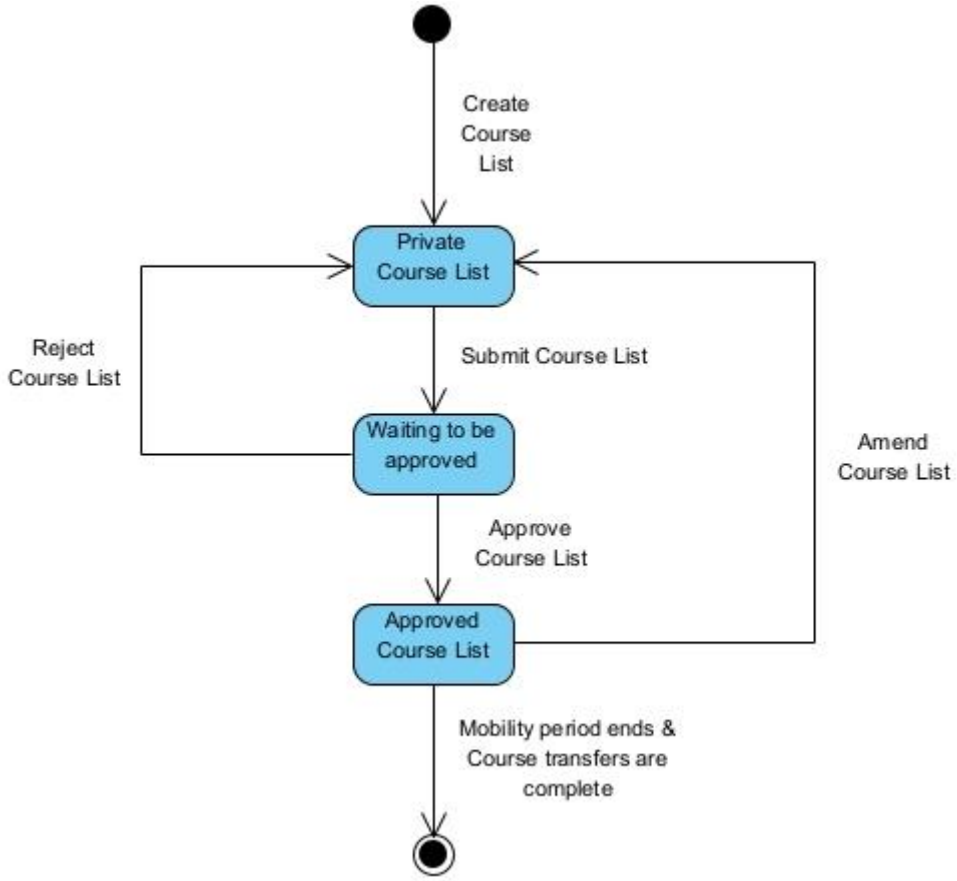


Fig. 2.3.3.2.1.: Course List State Diagram

ToDo item state diagram:

The diagram below shows the changes in a user's ToDo items' state.

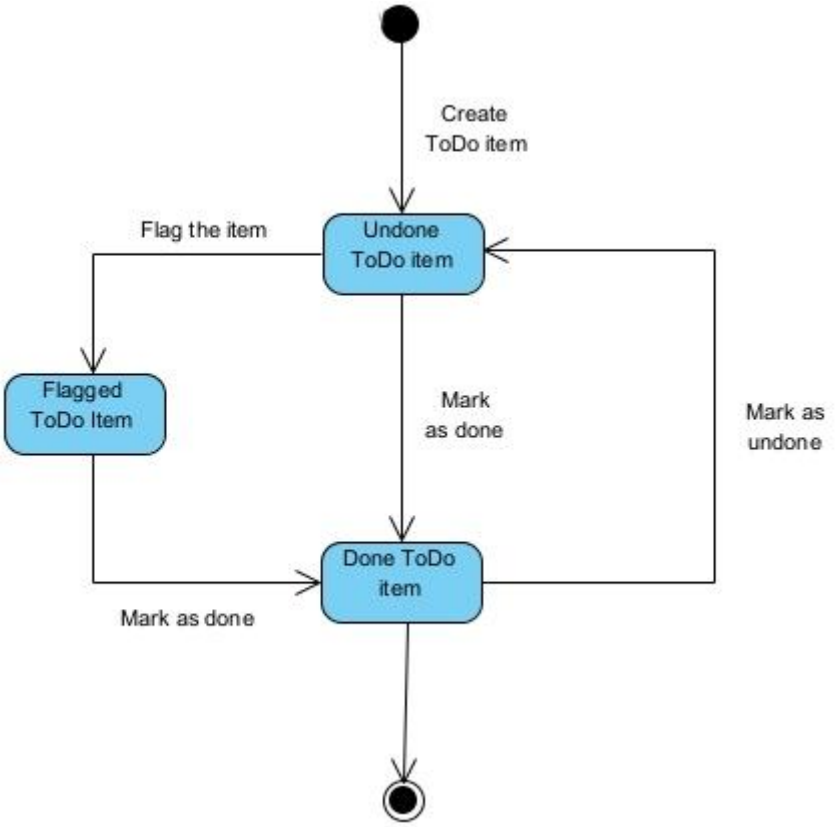


Fig. 2.3.3.2.2.: ToDo Item State Diagram

2.3.4. Sequence Diagram

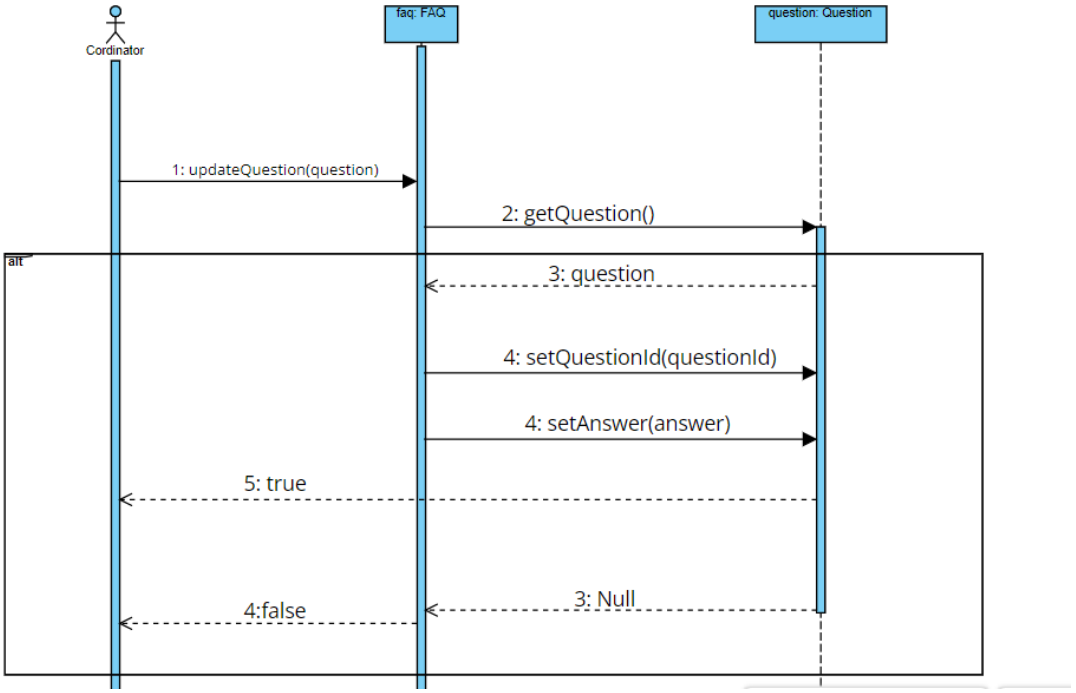


Fig. 2.3.4.1.: Sequence Diagram of Update FAQ Question

The coordinator can update the content of the question from the buttons appearing on the right side of the questions in the coordinator´s user interface. Unless there is a question in the FAQ, update functionality will work.

2.3.5. User Interface

2.3.5.1. Common Interfaces

Common interfaces are used by both users. For these pages, the only noticeable change between users is on the menu to the left. Coordinators have a “student list” menu button whereas students have a “courses” button.

Log-In Page

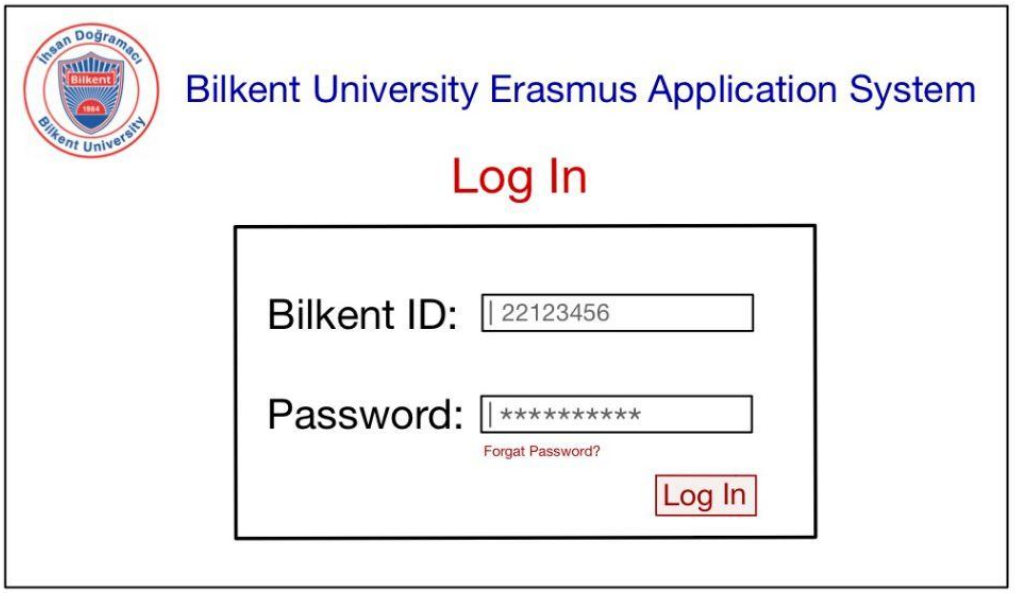


Fig. 2.3.4.1.1. Login user interface

Notifications

Users can select notifications to delete or flag them by clicking on the circles.

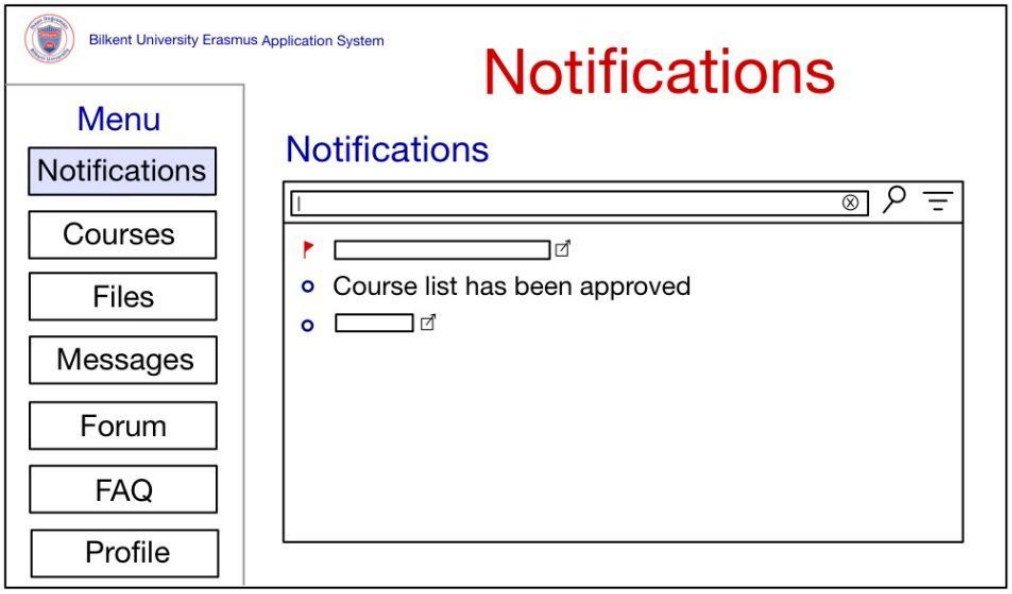


Fig. 2.3.4.1.2. Notifications user interface

Files

Through this page, users can access their files and download them.

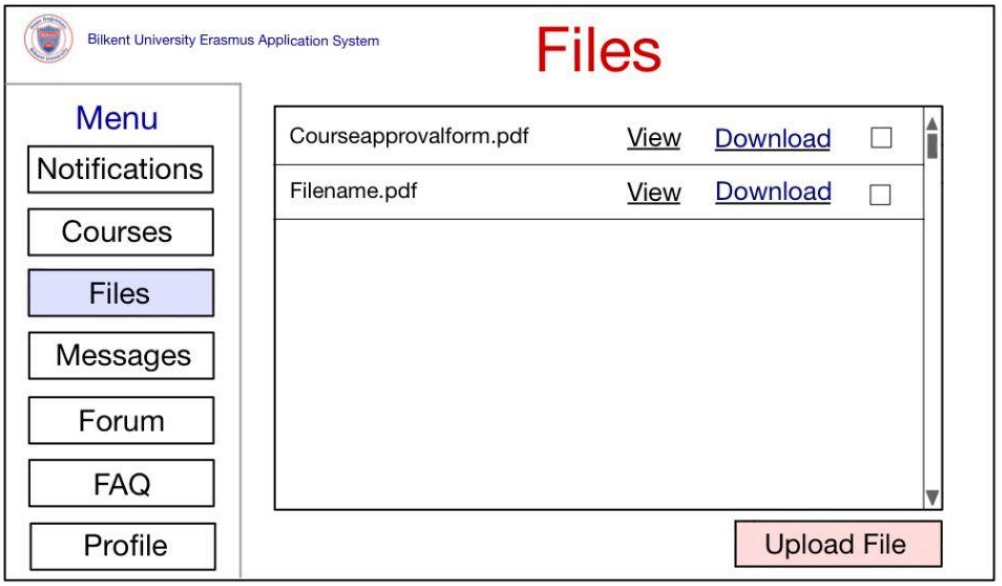


Fig. 2.3.4.1.3. List of files user interface

Upload File

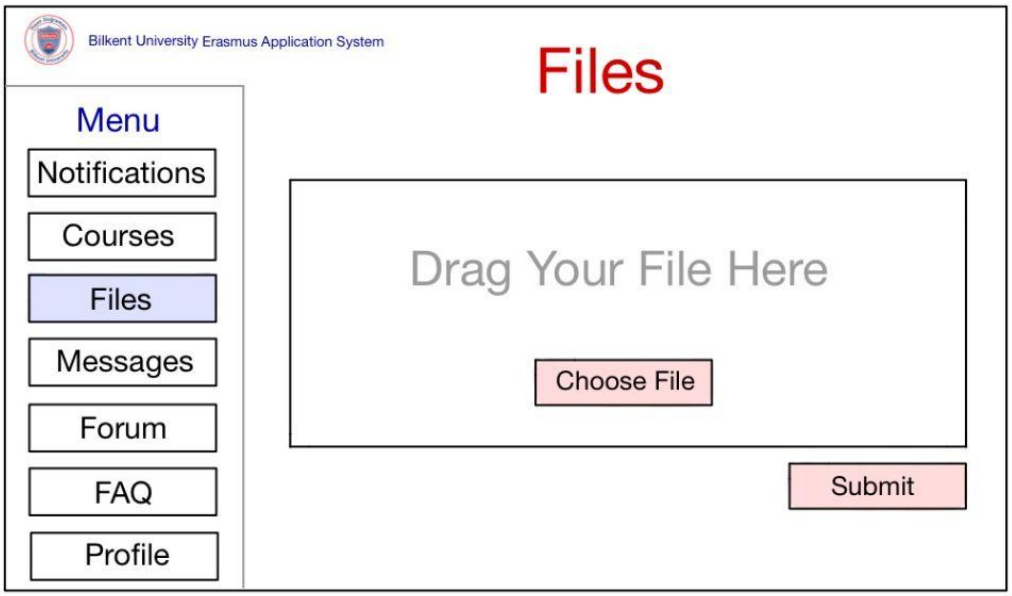


Fig. 2.3.4.1.4. Uploading a file user interface

Forum

Any keyword can be searched to find a particular post on the forum. Posts can also be filtered by their subjects, posters, and dates.

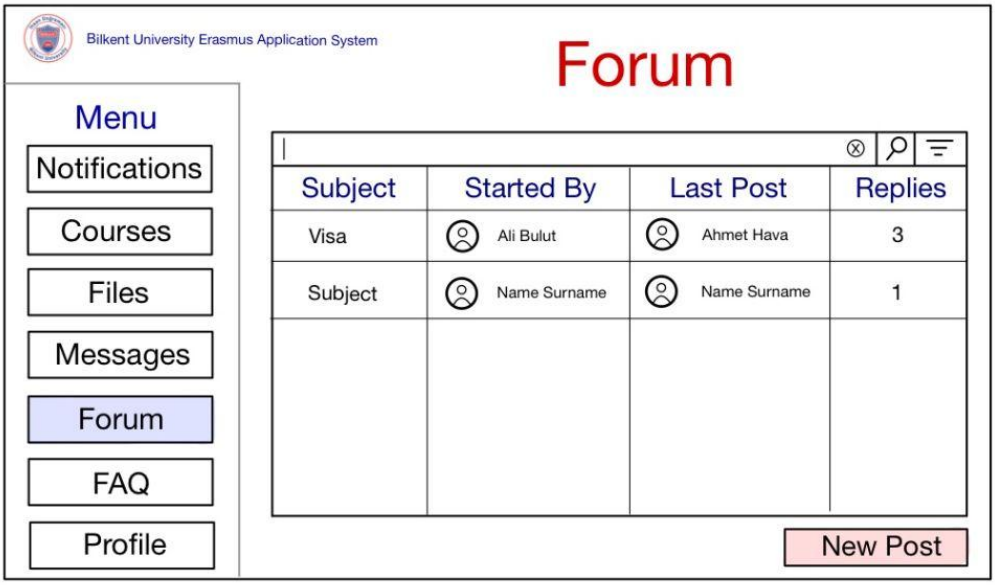


Fig. 2.3.4.1.5. List of forum posts user interface

Viewing Post

All replies to a particular post are aligned in the same line. A reply is indented a little right of the replied text.

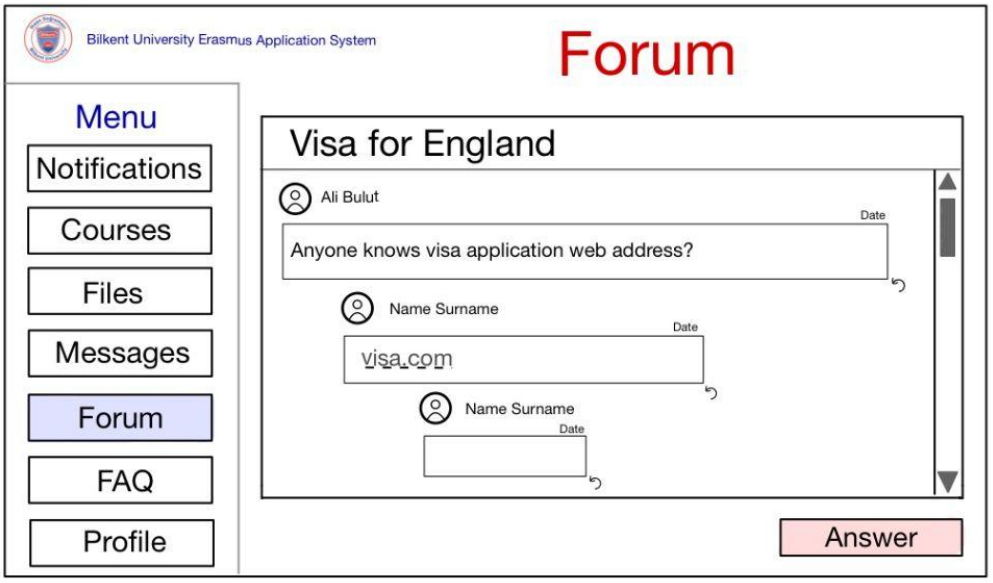


Fig. 2.3.4.1.6. Single forum post user interface

New Post or Replying to a Post

Although the interfaces are quite similar, the subject option is not shown if a user is replying to a post.

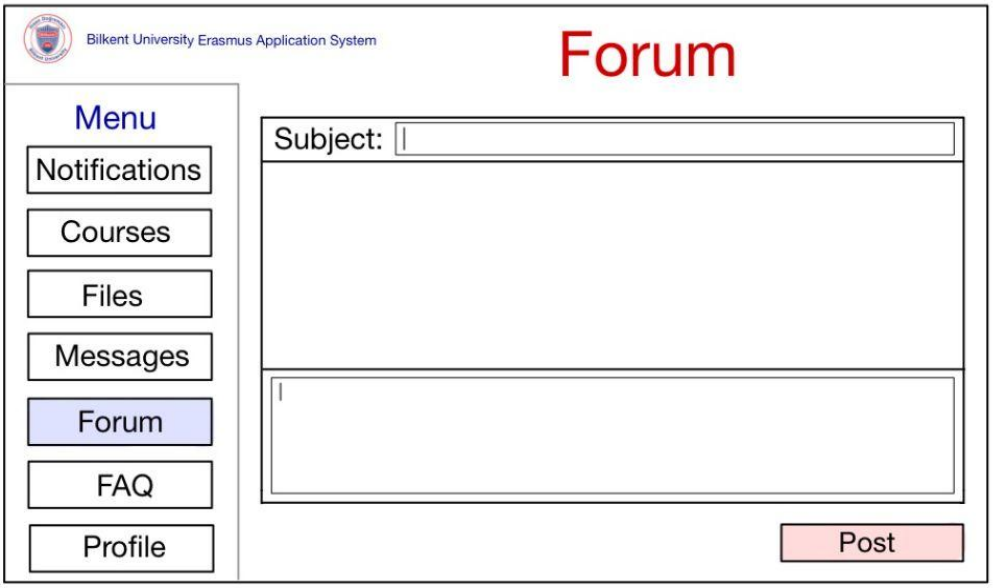


Fig. 2.3.4.1.7. Writing a forum post user interface

Viewing Own Profile

According to whether the user is a student or a coordinator, the information displayed may change. Along with viewing information, users also can change their password, and mail address. From the preferences, they can choose to share their specific information like their phone number, mail address with other users.

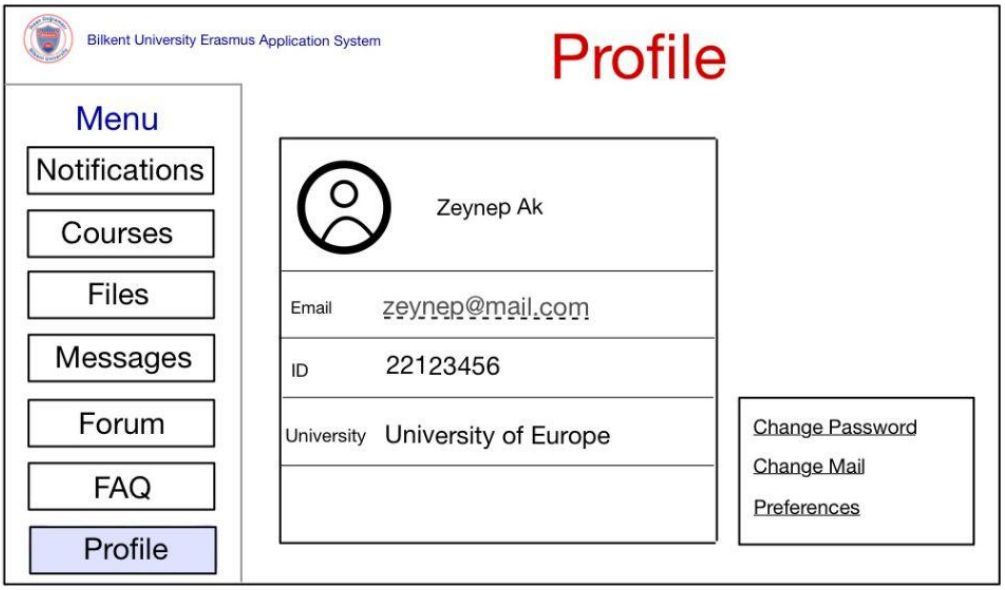


Fig. 2.3.4.1.8. Viewing own profile user interface

2.3.5.2. Student Interfaces

Main Page

Status shows the current state of the Erasmus process. For example, a student may be waiting for approval from the coordinator or the student may be expected to upload a specific file etc.

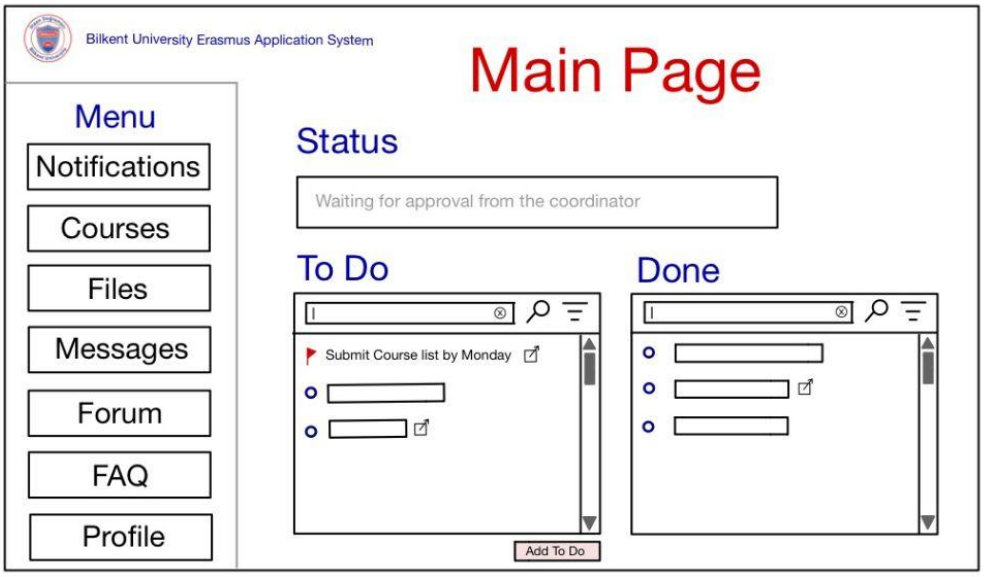


Fig. 2.3.4.2.1. Main page user interface

Courses

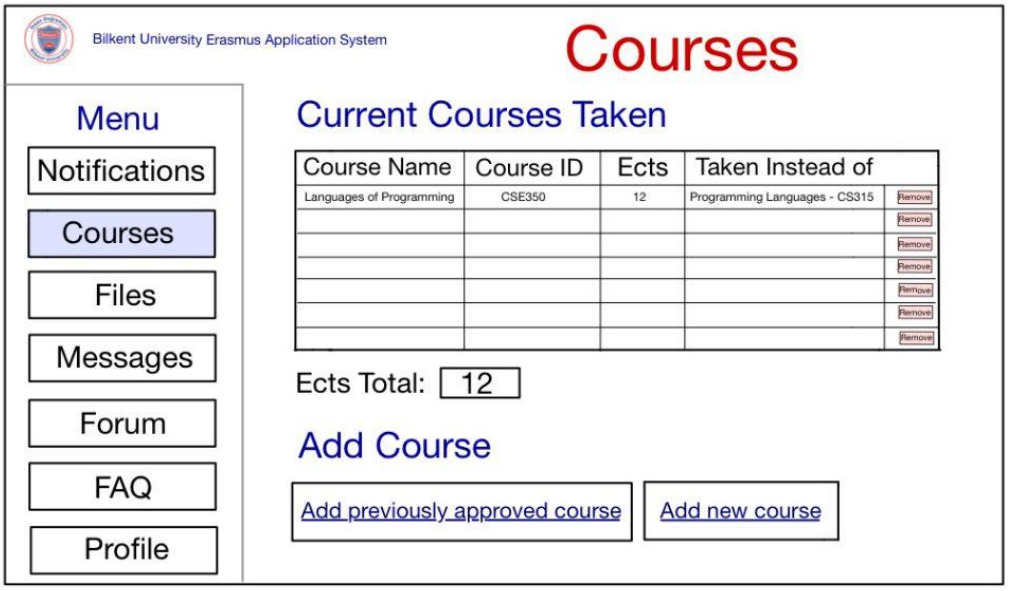


Fig. 2.3.4.2.2. Courses user interface

Add Previously Approved Course

Currently taken courses' list is shown above for every course adding process.

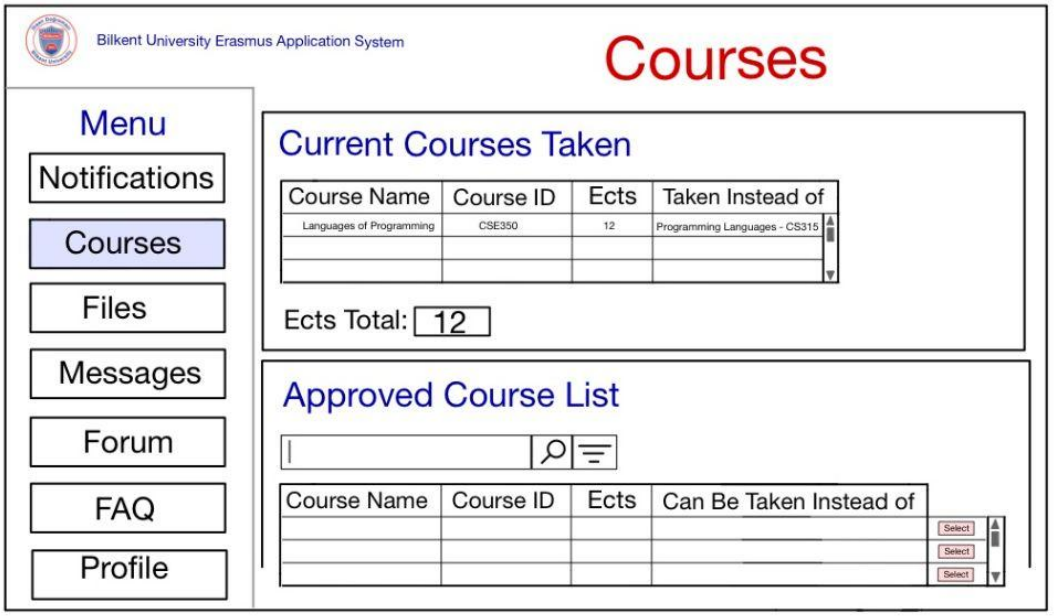


Fig. 2.3.4.2.3. Add previously approved course user interface

Add New Course

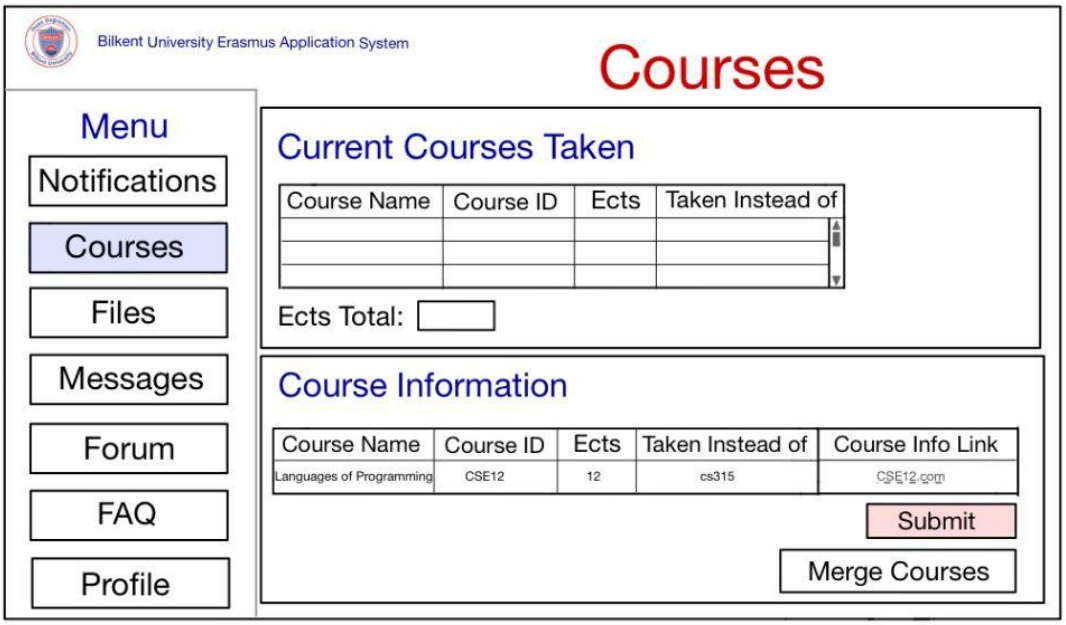


Fig. 2.3.4.2.4. Add new course user interface

Merge Courses

Multiple courses can be merged into one to cover a Bilkent course. Students can give required information about courses in required areas.

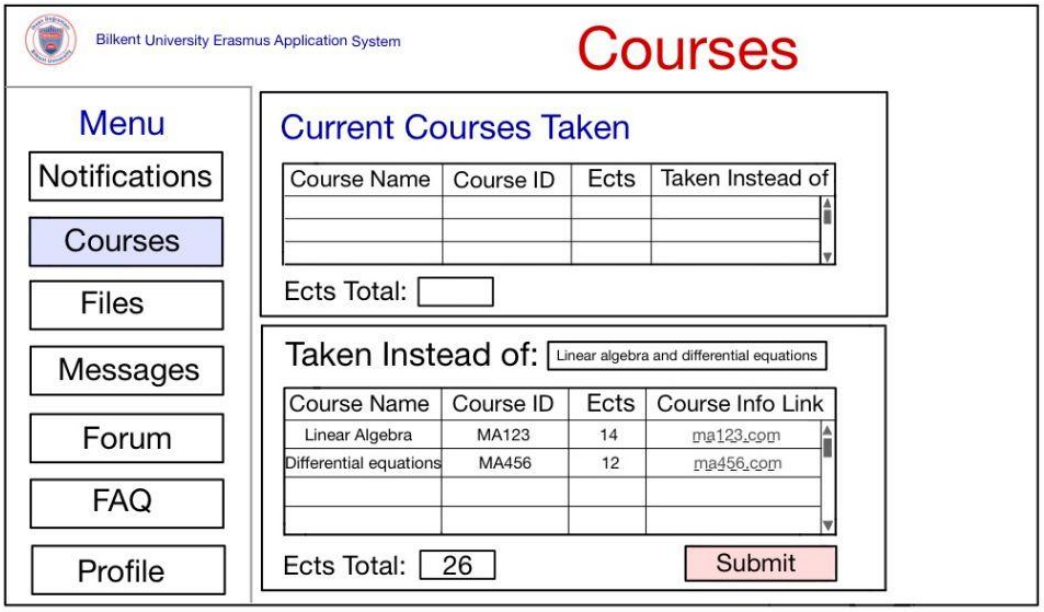


Fig. 2.3.4.2.5. Merge courses user interface

Messages

Students can only see other students' names, and other information if the students are allowed to share.

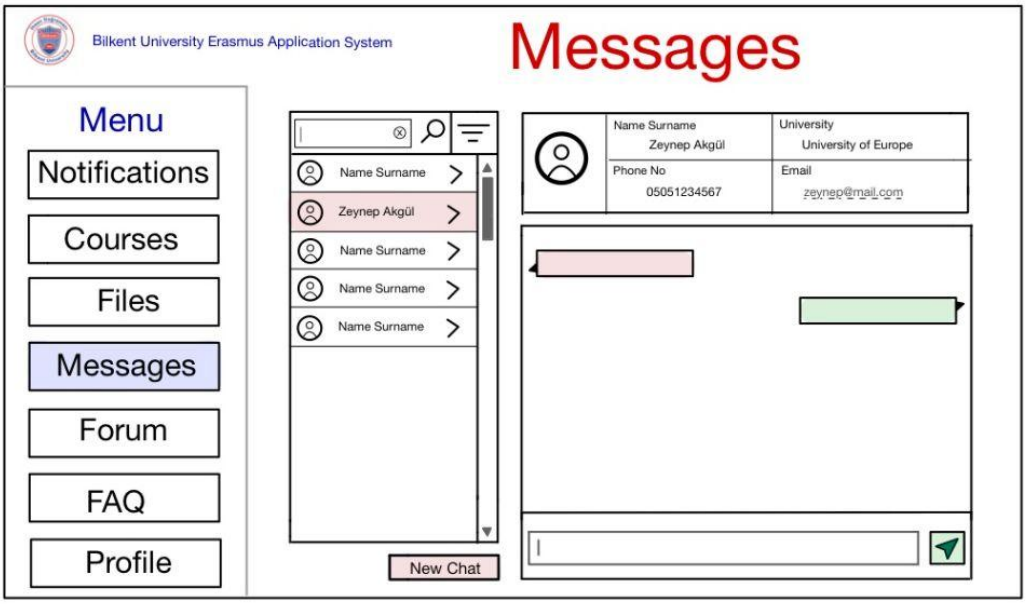


Fig. 2.3.4.2.6. Messages user interface

Frequently Asked Questions

Plus signs extend the questions to display their answers and the minus signs remove the answers from the view.

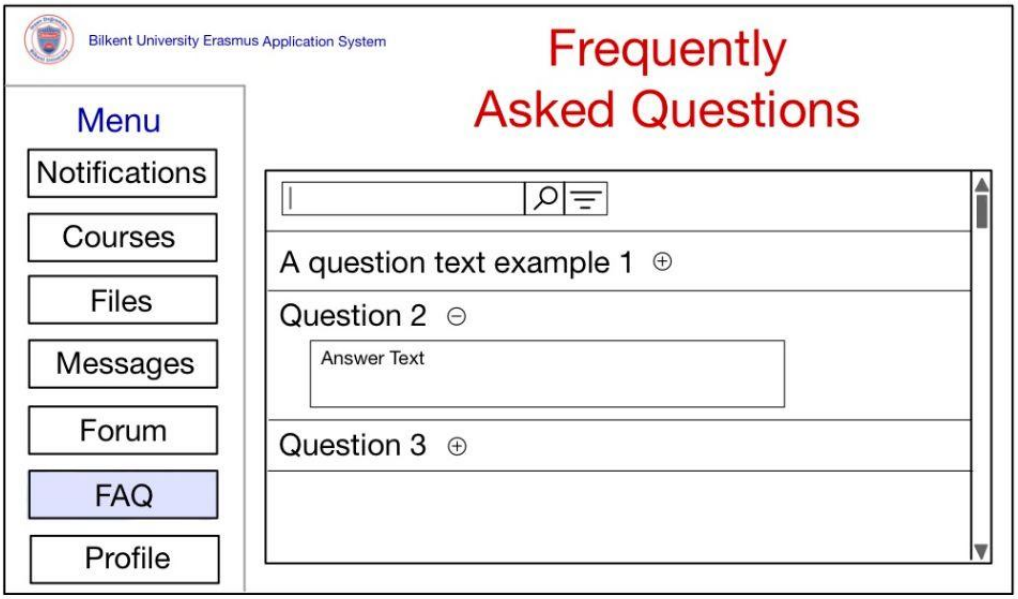
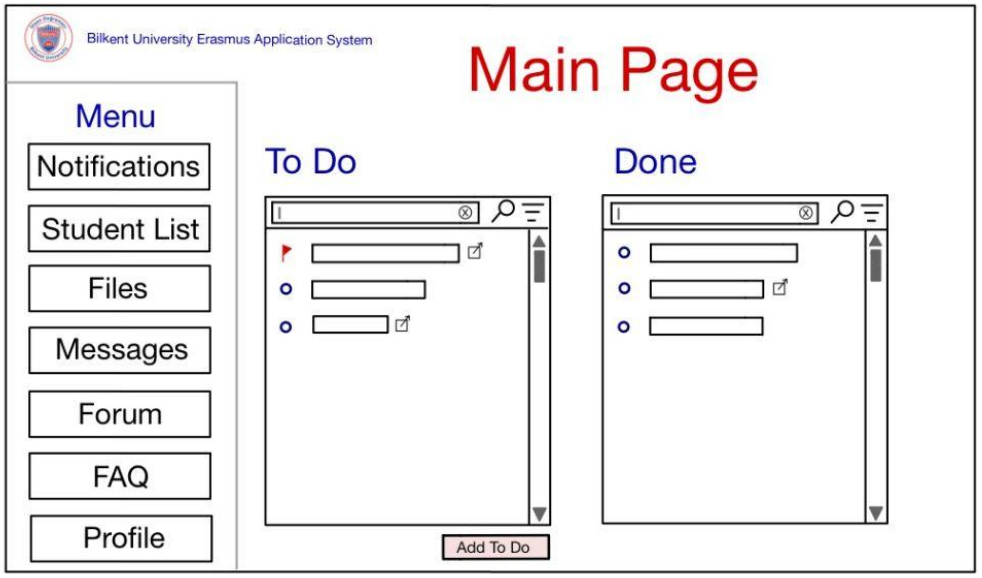


Fig. 2.3.4.2.7. FAQ user interface

2.3.5.3. Coordinator Interfaces

Main Page



Regarding options are displayed when To Do's or Done's chosen to be flagged, deleted or marked as done.

Fig. 2.3.4.3.1. Coordinator main page user interface

Student List

Student names have two links, one directed to the message interface and the other to the profile. The coordinator link directs to the message interface only.

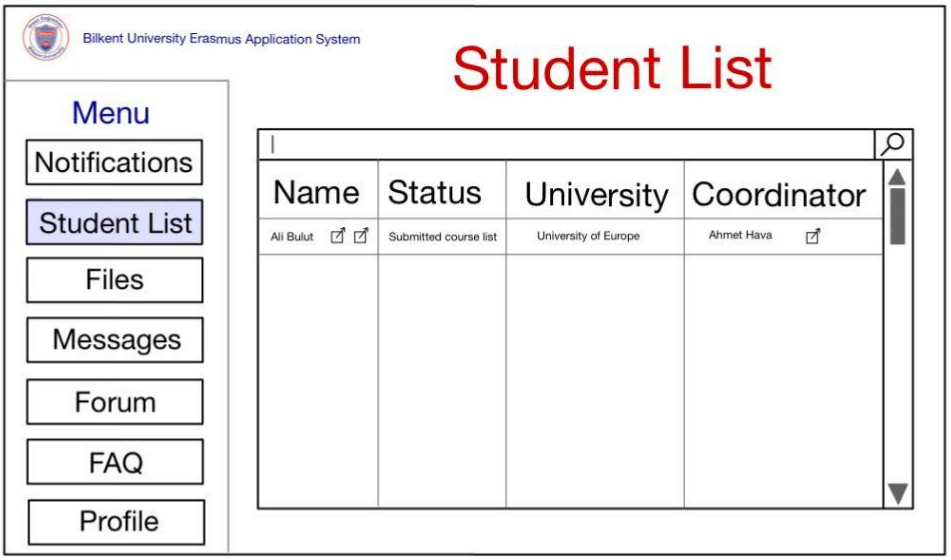


Fig. 2.3.4.3.2. Student list user interface

Messages

Coordinators can see every information submitted on a student profile in a messaging view.

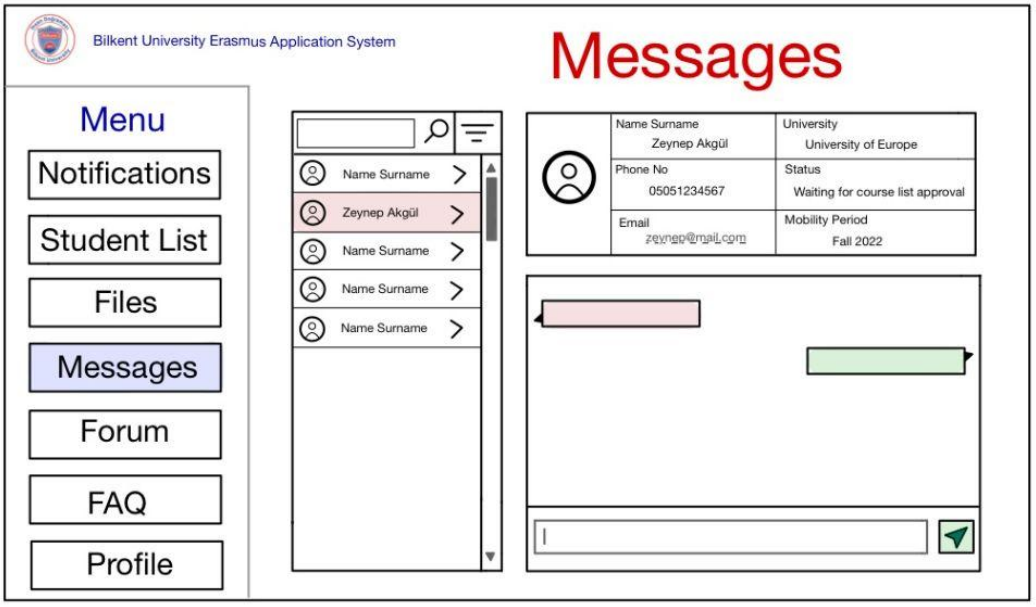


Fig. 2.3.4.3.3. Messages user interface

Viewing Student's Profile

Courses waiting to be approved and all courses taken by the student an be seen in student's courses

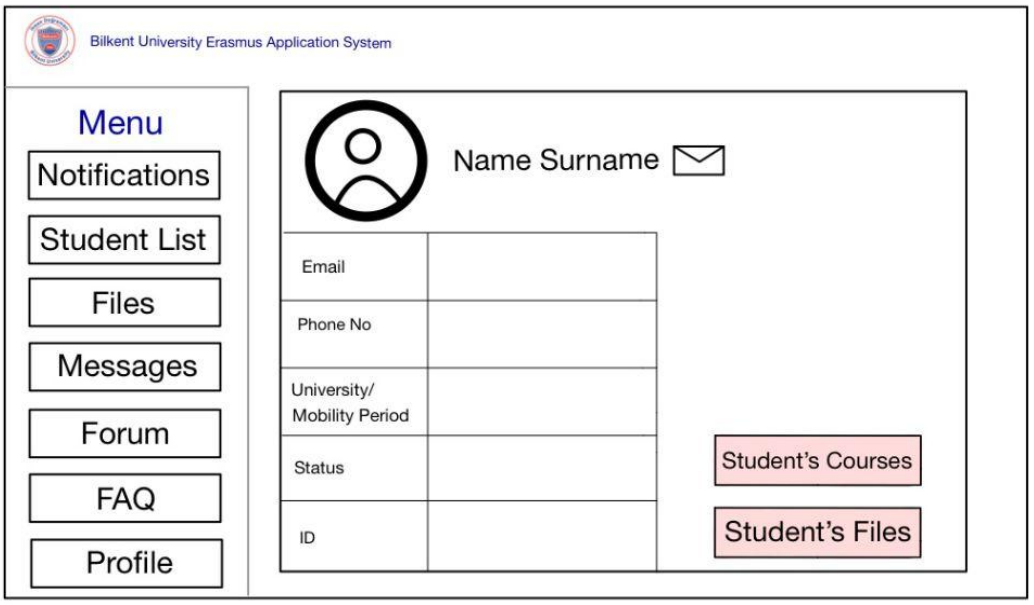


Fig. 2.3.4.3.4. Viewing student's profile user interface

Frequently Asked Questions

A coordinator can remove a frequently asked question by marking the circles and choosing to delete.

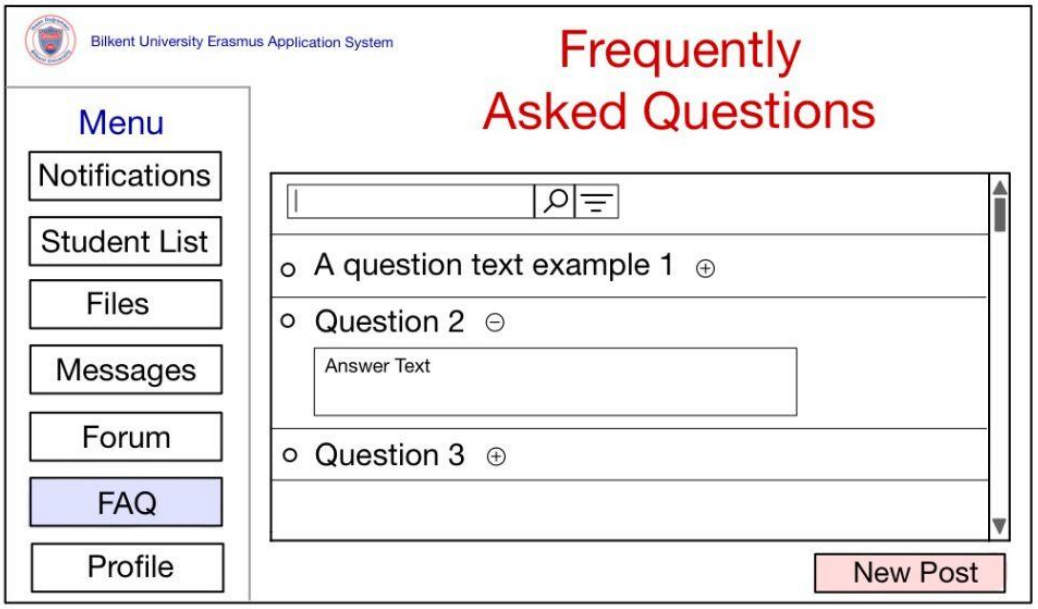


Fig. 2.3.4.3.5. Coordinator's FAQ user interface

Posting a Frequently Asked Question

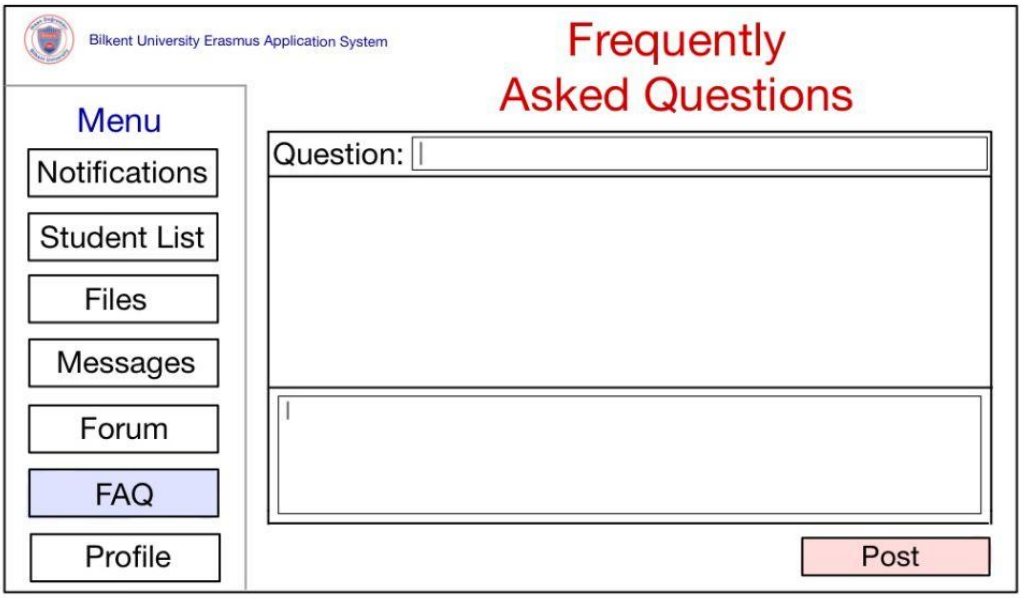


Fig. 2.3.4.3.6. Posting a FAQ user interface

Course Approval

Coordinators will be able to see submitted courses, course lists or merged courses and can reject or approve them along with an optional comment to be received by the student.

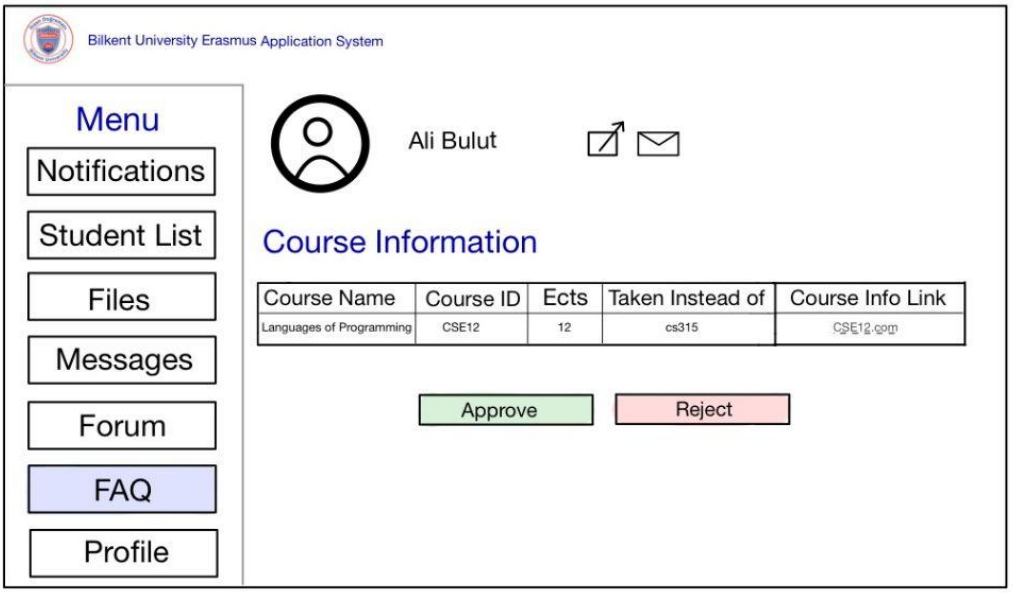


Fig. 2.3.4.3.7. Approving a course user interface

3. Improvements Summary

- Placement functionality is explained in the functional requirements part.
- Non functional requirements were updated and renewed.
- Activity diagrams were improved based on the given feedback. Namely, condition guards were added where they were missing, and the diagrams were made more readable by splitting decision nodes if they had too many connecting paths.
- One sample sequence diagram was added.
- State diagrams were redone to show the changes in an object (Course List and ToDo item) instead of UI navigation.
- Figure Captions are added
- Example information has been added to user interface diagrams
- The class diagram was modified based on the feedback given by the T.A., and some new decisions taken by our group. Small changes in the explanations of classes have been made.
- 19 many use cases and 2 external users are introduced. The use cases are as follows: SendEmail, ViewToDoList, ViewNotifications, ViewMessages, ViewProfile, ViewFiles, ViewFAQ, ViewForum, DeletePostFromForum, ViewUnapprovedCourses, ViewApprovedCourses, AddPreviouslyApprovedCourse, ObtainCourseData, AddNewCourse, MergeCourses, DeleteCourse, SubmitCourseList, ViewStudentCourses, and ViewStudentFiles. ChangeStatusOfNotifications is changed as UpdateNotifications, MessageContact is changed as SendMessage, ChangeFAQContents is changed as UpdateFAQContents, SeeStatus is changed as ViewStatus. Instead of AddCourse, we now have AddPreviouslyApprovedCourse and AddNewCourse. The newly introduced external users are MailServer and CourseDatabase. Moreover, the include and exclude relations are updated.

4. References

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

[2] “Use Case Diagram Notations Guide - Visual Paradigm.” Visual Paradigm Community Circle, May 11, 2018. https://circle.visual-paradigm.com/docs/uml-and-sysml/use-case-diagram/use-case-diagram-notations-guide/.

[3] Cysneiros, Luiz M. and Eric Yu. "Non-functional requirements elicitation." Perspectives on software (n.d.): 2004.