Bilkent University
Department of Computer Engineering

CS 319 - Object Oriented Software Engineering
Fall 2023
Section 1
Term Project

BilConnect

(Deliverable 1)

Team 3

Members

Abdullah Samed Uslu
22003598

Celal Salih Türkmen
22102498

Emre Akgül
22102310

Murat Çağrı Kara
22102505

Umut Bora Çakmak
22101806

Table of Contents

Introduction

At BilConnect, our vision is to create a connected Bilkent Campus that bridges the gap between the online and offline worlds of Bilkent community. Our mission is to provide an exclusive social platform designed specifically for the Bilkent community, offering real-time communication and an open marketplace. Inspired by the need for a more specialized and localized platform, we developed BilConnect to address the unique needs of Bilkenters, where users can buy and sell items, find lost possessions, connect with travel partners, and engage in other community activities. We also feature a user-friendly chat system, enhancing individual and group conversations, and encouraging meaningful interactions. It's not just a marketplace; it's a way to build connections, encourage sustainability, and enhance campus life. BilConnect is the go-to platform for all things Bilkent, bringing the campus community closer together.

Non-Functional Requirements

The non-functional requirements of BilConnect can be subdivided into six categories: speed, exclusivity, security, moderation, reliability, and expandability.

Speed

- The messaging functionality used in second-hand sales should provide a messaging experience that does not cause frustration for the user. The messaging should have a latency of less than 3 seconds.
- Posts on second-hand sales and club event posting have more tolerance towards latency, but still minimizing it to less than 15 seconds will prevent confused users from trying to repost cause they think the post didn't work.

Exclusivity

- This app is exclusive to the Bilkent student body, master and doctorate students, faculty members and teaching staff. Keeping it that way could be achieved by authenticating through the e-mail given by Bilkent University.
- Specifically creation of student club accounts should only be allowed after an authentication process.

Security

- This app will handle personal data such as name and e-mail. It will also contain inter-personal messages which is sensitive data. No data leaks should be allowed.
- The app should also be secure against spam. Since account numbers will be limited due to exclusivity, we can limit the number of posts per account to 20 per day. This will also ensure the identity of the app, preventing it from being run over by wholesalers.

Moderation

- Messaging, posts and club events should not be open to spam or any inappropriate content according to Bilkent University's code of ethics. This rule should be reinforced by moderators, number to be decided upon according to traffic and user numbers.

Reliability

- The app should be reliable and should not be down for extended periods of time (2 hours for busy periods, 6 hours for non-busy periods, 24 hours for maintenance). It should especially be reinforced for the start of semesters. Maintenance should be done outside of busy periods. The app should not have bugs that may cause any data loss or data leak.

Expandability

- The app has potential for expansion, especially to graduates and other universities that might be considered for the app. To this end, the app should be easily expandable with a small number of developers.

Tech Stack

We will build BilConnect using a tech stack of popular and updated technologies. Our backend is powered by ASP.NET and C#, and our frontend is powered by React. The architecture used by this trio has many tutorials, templates and support forums on the web that we think that it will be very useful in our project.

Backend with ASP.NET

- BilConnect's backend will be builded using ASP.NET, a web framework that meets our requirements of reliability and security. This framework efficiently operates the server-side processes, ensuring stability. Moreover, it is highly demanded in the industry, which makes it suitable for learning from many sources.

C# for Programming

- We will use C#, a popular object-oriented programming language, to program our app. C# is based on C++ and highly demanded in the industry. It offers high efficiency and many libraries, contributing to the reliability and performance of BilConnect.

SQL for Database

- Our data management system will use SQL, the most popular database language. SQL makes it possible for us to efficiently store, fetch, and manage various data, ensuring that BilConnect operates fast and securely.

Frontend using React

- We will be using React to code the frontend of BilConnect. It is a popular JavaScript library. It provides a fancy user interface, enhancing the appearance of our application. This enables us to create an engaging platform for the users of BilConnect.

Use Case Diagram

The use case diagram of the app is described in Figure 1 with the conventions of UML.

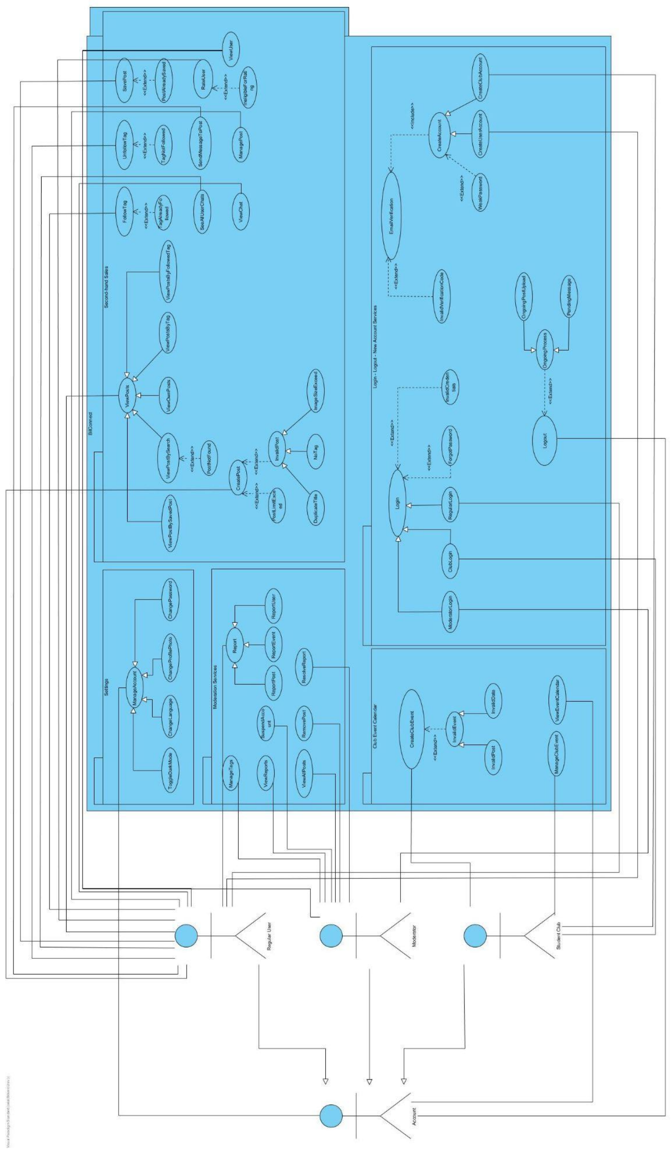

Figure 1: Use Case Diagram

Use Case Flows

In this section, the use case event flows of BilConnect are described with subsections related to their functions. The subsections are Settings, Moderation Services, Club Event Calendar, Second-Hand Sales, and Login/Logout/New Account Services.

1) Settings

This subsection is dedicated to explain the event flows of the use cases for settings.

a) ManageAccount

Participating Actor: Account

Entry Condition:

- *Account* clicks account setting from a drop-down menu on the upper right of the home page.

Exit Condition:

- *Account* decides to use one of the functions that are in **ManageAccount**, which are **ToggleDarkMode**, **ChangeLanguage**, **ChangeProfilePhoto** and **ChangePassword**.
- *Account* decides to return to the main page.

Flow of Events:

1) *Account* clicks on the setting button from the drop-down menu.
2) *Account* is redirected to a new page with account setting change options.
3) *Account* can choose any of **ToggleDarkMode**, **ChangeLanguage**, **ChangeProfilePhoto** and **ChangePassword** to proceed.

Special/Quality Requirements: None.

b) ChangeLanguage

Participating Actor: Account

Entry Condition:

- *Account* selects **ChangeLanguage** from **ManageAccount** page.

Exit Condition:

- Language change successful.
- *Account* decides to return to **ManageAccount** or main page.

Flow of Events:

1) *Account* clicks on the “Change Language” button from the settings.
2) *Account* is redirected to a page that contains language options, English and Turkish.
3) *Account* selects their desired language.

Special/Quality Requirements: None.

c) ToggleDarkMode

Participating Actor: Account

Entry Condition:

- *Account* selects **ToogleDarkMode** from **ManageAccount** page.

Exit Condition:

- *Account* decides to return to **ManageAccount** or the main page.

Flow of Events:

1) *Account* clicks on the “Toggle Dark Mode” button from the settings.
2) *Account* is redirected to a page that has a switch to toggle dark mode.
3) *Account* selects their desired visual mode.

Special/Quality Requirements: None.

d) Change Password

Participating Actor: Account

Entry Condition:

- *Account* selects **ChangePassword** from **ManageAccount** page.

Exit Condition:

- *Account* decides to return to **ManageAccount** or the main page.
- *Account* sets the new password.
- The new password is not accepted.

Flow of Events:

1) *Account* clicks on the “Change Password” button from the settings.
2) *Account* is redirected to a page that has the two password boxes for current password and new password.
3) *Account* enters the current and new passwords.

Special/Quality Requirements: The current password must be correct. The new password must meet the password requirements. The new password cannot be the same with the old password. The passwords must be censored.

e) Change Profile Photo

Participating Actor: Account

Entry Condition:

- *Account* selects **ChangeProfilePhoto** from **ManageAccount** page.

Exit Condition:

- *Account* decides to return to **ManageAccount** or the main page.
- *Account* sets the new profile photo.
- The new profile photo is not accepted.

Flow of Events:

1) *Account* clicks on the “Change Profile Photo” button from the settings.
2) *Account* uploads a new photo.

Special/Quality Requirements: The new photo must be in correct file format, size and pixel ratio. It must be cropped to be a square.

2) Moderation Services

This subsection is dedicated to explain the event flows of the use cases for moderation services.

a) ManageTags

Participating Actor: Moderator

Entry Condition:

- *Moderator* selects **ManageTags** from the main page.

Exit Condition:

- *Moderator* decides to return to the main page.

Flow of Events:

1) *Moderator* enters "Manage Tags" page.
2) *Moderator* chooses to either add, change or remove a tag.
3) If removing or changing, the *Moderator* chooses the tag to operate on.
4) If adding or changing, the *Moderator* enters the title for the tag. Adding or changing a tag cannot create a tag that already exists.

Special/Quality Requirements: None.

b) ViewReports

Participating Actor: Moderator

Entry Condition:

- *Moderator* selects **ViewReports** from the main page.

Exit Condition:

- *Moderator* decides to return to the main page.

Flow of Events:

1) *Moderator* enters "View Reports" page.
2) *Moderator* selects report to view in detail in addition to posts and users affiliated with the report.

Special/Quality Requirements: None.

c) ViewAllPosts

Participating Actor: Moderator

Entry Condition:

- *Moderator* selects **ViewAllReports** from the main page.

Exit Condition:

- *Moderator* decides to return to the main page.

Flow of Events:

3) *Moderator* enters "View Reports" page. Moderator can view old as well as new posts in this page.
4) *Moderator* selects posts to view in detail.

Special/Quality Requirements: None.

d) SuspendAccount

Participating Actor: Moderator

Entry Condition:

- *Moderator* selects **SuspendAccount** from the accounts' profile.

Exit Condition:

- Suspension is successful.

Flow of Events:

1) *Moderator* presses "Suspend" button located on the *Account's* profile which is only visible to the *Moderator*.
2) *Moderation* will designate a time period and reason for the suspension.
3) *Account* and posts related to it will cease to be visible to *Regular User*'s.

Special/Quality Requirements: None.

e) RemovePost

Participating Actor: Moderator

Entry Condition:

- *Moderator* selects **RemovePost** from the post page.

Exit Condition:

- Post is successfully removed.

Flow of Events:

1) *Moderator* presses "Remove Post" button in the post page.
2) *Moderation* will designate a reason for the removal.
3) Post will no longer be viable for *Redular User.*

Special/Quality Requirements: None.

f) ReportPost

Participating Actor: Regular User

Entry Condition:

- *Regular User* selects **ReportPost** from the post page.

Exit Condition:
- Successfully reported.

Flow of Events:
1) *Regular User* presses actions menu and then “Report Post” dropdown menu button in the post page.
2) *Regular User* will specify the reasoning behind the report of the post by including description and if possible visual proof.
3) “Report post ” request will be delivered to *Moderator.*
4) *Moderator* takes action upon whether the report is valid or not, if valid then *Moderator* will remove the post, else the post will still be displayed.

Special/Quality Requirements: None.

g) ReportEvent

Participating Actor: Regular User

Entry Condition:
- *Regular User* selects **ReportEvent** from the event details page.

Exit Condition:
- Successfully reported.

Flow of Events:
1) *Regular User* presses the “Report” button.
2) *Regular User* provides a description and if possible visual proof of the report cause.

Special/Quality Requirements: None.

h) ReportUser

Participating Actor: Regular User

Entry Condition:
- *Regular User* selects **ReportUser** from the user page.

Exit Condition:
- Successfully reported.

Flow of Events:

1) *Regular User* presses the “Report” button.
2) *Regular User* provides a description and if possible visual proof of the report cause.

Special/Quality Requirements: None.

i) ResolveReport

Participating Actor: Moderator

Entry Condition:
- *Moderator* wants to resolve and get rid of a report.

Exit Condition:
- Successfully resolved.

Flow of Events:
1) *Moderator* enters report details.
2) *Moderator* presses the “Resolve Report”.

Special/Quality Requirements: None.

3) Club Event Calendar

This subsection is dedicated to explain the event flows of the use cases for the club event calendar.

a) CreateClubEvent

Participating Actor: Student Club

Entry Condition:
- *Student Club* decides to post a new event.

Exit Condition:
- Event successfully posted.

Flow of Events:
1) *Student Club* selects “Create New Event” on the main page.
2) *Student Club* designates the time and place for the event as well as an explanation of the event.

Special/Quality Requirements: None.

b) ManageClubEvent

Participating Actor: Student Club

Entry Condition:

- *Student Club* decides to change the details of the event.

Exit Condition:

- Post successfully edited.

Flow of Events:

1) *Student Club* selects “Create Edit Event” on the page detailing the extent.
2) *Student Club* designates the new time and new place for the event.

Special/Quality Requirements: None.

c) ViewEventCallendar

Participating Actor: Account

Entry Condition:

- *Account* decides to view Event Calendar from a button on the top left of the main page.

Exit Condition:

- *Account* decides to go back to the main page.

Flow of Events:

1) *Account* enters the Event calendar page.
2) *Account* can view future and past club events on the calendar.

Special/Quality Requirements: None.

4) Second-hand Sales

This subsection is dedicated to explain the event flows of the use cases for second-hand sales services.

a) ViewPostBySavedPosts

Participating Actor: Regular User

Entry Condition:

- *Regular User* clicks “Saved Posts” from a drop-down menu on the upper right of the home page.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* can view saved posts in a list form, ordered by date saved.
2) *Regular User* can search by a function provided in the “Saved Post” page to better find what they are looking for.

Special/Quality Requirements: None.

b) ViewPostBySearch

Participating Actor: Regular User

Entry Condition:

- *Regular User* uses the search function on the main page without tags only by name.

Exit Condition:

- *Regular User* returns to the main page.
- *Regular User* uses **ViewPostByTag**.

Flow of Events:

1) *Regular User* types in keywords for the search.
2) *Regular User* is provided with related results based on their title.
3) *Regular User* can sort them by date or by cost.

Special/Quality Requirements: None.

c) ViewOwnPosts

Participating Actor: Regular User

Entry Condition:

- *Regular User* clicks “View Own Posts” from a drop-down menu on the upper right of the home page.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* can view their own posts in a list form, ordered by date uploaded to the system with the most recent post will be displayed at the top.
2) *Regular User* can search by a function provided in the "View Own Post" page to better find what they are looking for.

Special/Quality Requirements: None

d) ViewPostsByTag

Participating Actor: Regular User

Entry Condition:

- *Regular User* uses the search function on the main page with tags.

Exit Condition:

- *Regular User* returns to the main page.
- *Regular User* uses **ViewPostBySearch**.

Flow of Events:

1) *Regular User* selects tags for the search, if desired combined with keywords.
2) *Regular User* is provided with related results based on their tag and title.
3) *Regular User* can sort them by date or by cost.

Special/Quality Requirements: None.

e) ViewPostsByFollowedTag

Participating Actor: Regular User

Entry Condition:

- *Regular User* clicks "Followed Tags" from a drop-down menu on the upper right of the home page.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* is shown posts sorted by date on all the followed tags by default.
2) *Regular User* can further search from these posts ore focus on a few tags.

Special/Quality Requirements: None.

f) CreatePost

Participating Actor: Regular User

Entry Condition:

- *Regular User* clicks “Create Post” from the middle lower portion of the main page.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* enters the post-creation screen.
2) *Regular User* selects a title, description and tag for the post.
3) *Regular User* uploads photos of the product.
4) If the tag of the post permits it (Second-hand tag will, while Donation tag will not), *Regular User* will set a price.

Special/Quality Requirements: None.

g) FollowTag

Participating Actor: Regular User

Entry Condition:

- From the **ViewPostsByFollowedTag** screen, *Regular User* presses the “Manage Followed Tags” button.

Exit Condition:

- *Regular User* returns to the main page.
- *Regular User* returns to the **ViewPostsByFollowedTag** screen.

Flow of Events:

1) *Regular User* is presented with a screen with list of all tags.
2) They can search from the list and click on an unfollowed tag to follow it.

Special/Quality Requirements: None.

h) UnfollowTag

Participating Actor: Regular User

Entry Condition:

- From the **ViewPostsByFollowedTag** screen, *Regular User* presses the “Manage Followed Tags” button.

Exit Condition:

- *Regular User* returns to the main page.
- *Regular User* returns to the **ViewPostsByFollowedTag** screen.

Flow of Events:

1) *Regular User* is presented with a screen with list of all tags.
2) They can search from the list and click on a followed tag to unfollow it.

Special/Quality Requirements: None.

i) SavePost

Participating Actor: Regular User

Entry Condition:

- When viewing a post, user clicks on the heart icon on the post. Empty icon will be filled.

Exit Condition:

- When viewing a post, user clicks on the heart icon on the post. Filled icon will be emptied.

Flow of Events:

1) *Regular User* enters post details.
2) *Regular User* clicks on the heart icon.

Special/Quality Requirements: None.

j) SeeAllUserChats

Participating Actor: Regular User

Entry Condition:

- *Regular User* clicks “Chats” from a drop-down menu on the upper right of the home page.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* is redirected to a page containing all their chats sorted by last message time.

Special/Quality Requirements: None.

k) SendMessageToPost

Participating Actor: Regular User

Entry Condition:

- When viewing a post, user clicks on the “I am interested” button on the post.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) *Regular User* can send messages to and receive messages from the post owner.

Special/Quality Requirements: None.

l) RateUser

Participating Actor: Regular User

Entry Condition:

- The *Regular User* presses “Rate User” button on the **ViewPosts** page.

Exit Condition:

- The *Regular User* presses any position of the 5 star UI element and the “Rate” button after “Rate User” button creates a pop-up.
- The *Regular User* presses “Exit” button after “Rate User” button creates a pop-up.

Flow of Events:

1) The web application produces a automated pop-up element that includes 5 star UI element that can be pressed in any region of the space that it is placed and “Rate User” button.
2) The *Regular User* presses any position of the 5 star UI element and the “Rate” button or presses the “Exit” button instead.
3) The rate of the *Regular User* who created the post is recalculated if “Rate” button is pressed and displayed on the *Regular User* UI element, else UI element remains the same.

Special/Quality Requirements: None.

m) ViewChat

Participating Actor: Regular User

Entry Condition:

- From the **SeeAllUserChats** page, select a message thread that already started.

Exit Condition:

- *Regular User* chooses to return to **SeeAllUserChats**.

Flow of Events:

1) *Regular User* sees previous chats with the post owner on this specific post.
2) *Regular User* can send text messages and/or images to the chat.

Special/Quality Requirements: None.

n) ManagePost

Participating Actor: Regular User

Entry Condition:

- *Regular User* has a post to click meaning he at least created one post.
- *Regular User* chooses the intended post from the **ViewOwnPosts** page.

Exit Condition:

- *Regular User* returns to the main page.
- *Regular User* saves any changes being made using “Save” button.

Flow of Events:

1) *Regular User* is redirected to **Post** page.
2) *Regular User* chooses to edit, delete post or open chat feature.
3) *Regular User* can close the chat if opened and can save any changes that they made by clicking “Save” button or they can redict themselves to the main page by clicking the icon.

Special/Quality Requirements: None.

o) ViewUser

Participating Actor: Regular User, Moderator

Entry Condition:

- *Regular User* must be on a **Post** page.
- *Regular User* must click the user icon.

Exit Condition:

- *Regular User* returns to the main page.

Flow of Events:

1) Web application displays *Regular User*’s profile elements such as its rate and the posts queried by the *Regular User* in the past.

Special/Quality Requirements: None.

5) Login/Logout/New Account Services

This subsection is dedicated to explain the event flows of the use cases for Login/Logout and new account registration services.

a) Login

Participating Actor: Account

Entry Condition:

- *Account* presses the log-in button on the home page.
- *Account* enters a valid username and password.
- *Account* presses the log-in button in the log-in page.

Exit Condition:

- *Account* enters their account.

Flow of Events:

1) *Account* enters the login page.
2) *Account* enters their account's username or email.
3) *Account* enters their account's password.
4) Website directs the user to the home page of their account.

Special/Quality Requirements: The login process must be secured using HTTPS. Passwords should be stored encrypted and should never be displayed in any readable format.

b) ModeratorLogin

Participating Actor: Moderator

Entry Condition:

- *Moderator* presses the login button on the home page.
- *Moderator* enters a valid username and password.
- *Moderator* presses the login button in the log-in page.

Exit Condition:

- *Moderator* enters their account.

Flow of Events:

1) *Moderator* enters the login page.
2) *Moderator* enters their account's username or email.
3) *Moderator* enters their account's password.
4) Website directs the user to the home page of their account.

Special/Quality Requirements: The login process must be secured using HTTPS. Passwords should be stored encrypted and should never be displayed in any readable format.

c) ClubLogin

Participating Actor: Student Club

Entry Condition:

- *Student Club* presses the log-in button on the home page.
- *Student Club* enters a valid username and password.
- *Student Club* presses the log-in button in the log-in page.

Exit Condition:

- *Student Club* enters their account.

Flow of Events:

1) *Student Club* enters the login page.
2) *Student Club* enters their account’s username or email.
3) *Student Club* enters their account’s password.
4) Website directs the user to the home page of their account.

Special/Quality Requirements: The login process must be secured using HTTPS. Passwords should be stored encrypted and should never be displayed in any readable format.

d) RegularLogin

Participating Actor: Regular User

Entry Condition:
- *Regular user* presses the log-in button on the home page.
- *Regular user* enters a valid username and password.
- *Regular user* presses the log-in button in the log-in page.

Exit Condition:
- *Regular user* enters their account.

Flow of Events:
1) *Regular user* enters the login page.
2) *Regular user* enters their account’s username or email.
3) *Regular user* enters their account’s password.
4) Website directs the user to the home page of their account.

Special/Quality Requirements: The login process must be secured using HTTPS. Passwords should be stored encrypted and should never be displayed in any readable format.

e) Logout

Participating Actor: Account

Entry Condition:
- An account must already be logged in.
- The user presses the logout button.

Exit Condition:
- User's account is closed.

Flow of Events:
1) *Account* press logout button.
2) The website lets the user logout of the account and redirects the user to the home page.

Special/Quality Requirements: Ensure complete session termination so no data or operations from the previous session can be accessed without a new login.

f) CreateAccount

Participating Actor: Account

Entry Condition:
- *Account* presses the register button on the home page.
- *Account* enters a valid username and password, and enters the email verification code correctly.
- *Account* presses the sign up button in the log-in page.

Exit Condition:
- *Account* does an invalid operation and the page redirects to the login/register page.
- *Account* signs up successfully and logged in. *Account* is redirected to the main page.

Flow of Events:
1) *Account* enters the register page.
2) *Account* enters their account's username or email.
3) *Account* enters the verification code sent by email.
4) Website directs the user to the home page of their account if they entered every input correctly.

Special/Quality Requirements: Password must be strong, and must have at least 8 characters. The registered email must end with "bilkent.edu.tr" and must not be registered before. A new email verification code can only be sent after 60 seconds.

g) CreateUserAccount

Participating Actor: Regular User

Entry Condition:
- *Regular user* presses the register button on the home page.
- *Regular user* enters a valid username and password, and enters the email verification code correctly.
- *Regular user* presses the sign up button in the log-in page.

Exit Condition:

- *Regular user* does an invalid operation and the page redirects to the login/register page.
- *Regular user* signs up successfully and logged in. User is redirected to the main page.

Flow of Events:

1) *Regular user* enters the register page.
2) *Regular user* enters their account's username or email.
3) *Regular user* enters the verification code sent by email.
4) Website directs the user to the home page of their account if they entered every input correctly.

Special/Quality Requirements: Password must be strong, and must have at least 8 characters. The registered email must end with "bilkent.edu.tr" and must not be registered before. A new email verification code can only be sent after 60 seconds.

h) CreateClubAccount

Participating Actor: Student Club

Entry Condition:

- *Student Club* presses the register button on the home page.
- *Student Club* enters a valid username and password, and enters the email verification code correctly.
- *Student Club* presses the sign up button in the log-in page.

Exit Condition:

- *Student Club* does an invalid operation and the page redirects to the login/register page.
- *Student Club* signs up successfully and logged in. User is redirected to the main page.

Flow of Events:

1) *Student Club* enters the register page.
2) *Student Club* enters their account's username or email.
3) *Student Club* enters the verification code sent by email.
4) Website directs the user to the home page of their account if they entered every input correctly.

Special/Quality Requirements: Password must be strong, and must have at least 8 characters. The registered email must be a verified Bilkent student club account and must not be registered before. A new email verification code can only be sent after 60 seconds.