Group 6 - Milestone Report 1

Executive Summary

The work done in the previous semester was already available. In this semester, firstly, we went over that content to check if they need any change and actually we have noticed that sequence diagrams had to be updated; therefore, we spent time on that part. Also we updated the project plan since the plan for this semester was very hypothetical. Lastly, we closed unnecessary issues and similar stuff.

After the review for the existing content was done, all of the back-end, front-end and android teams continued with a period of learning and practicing the tools that they will use. Then each team focused on building infrastructure kind of activities like preparing database.

When all these points were done, all of the teams started implementing most basic functionalities such as registration, login and logout operations, resetting password and viewing profile page for logged in users. All teams completed the intended parts; furthermore, mentioned features were shown to the customer in the first milestone.; therefore, we can say that the root of the building was constructed and now the rest is more about implementation of specific functionalities.

For the future, first of all, we are planning to continue with sport event related parts such as defining a new event, joining it, canceling it and searching to find interesting events. After that, the next objective will be additional features like badges, equipment and fields.

List and Status of Deliverables

Deliverable Name	**Status**
Requirements	Delivered
UML Software Designs	Delivered
Scenarios and Mockups	Delivered

Project plan	Delivered
Communication Plan	Delivered
RAM	Delivered

Evaluation of the Status of Deliverables

Deliverable Name	**Status**
Requirements	Requirements are the main resource of the project and the project plan. We separated the project plan into logical parts and those parts were selected from the requirements. We try to make sure that all requirements are implemented at the correct time and with its related requirements.
UML Software Designs	We checked our use case and class diagrams and did not find any missing parts from the requirements.
Scenarios and Mockups	We checked the scenarios and mockups in the meetings. We decided that they are fine and they don't require any changes so they remained unchanged.
Project plan	Our team was updated. One of our friends left the group and we welcomed a new friend. Then we divided into three sub groups that are focused on different platforms. We updated our plan according to these changes. Everybody had their specific tasks and we gave the first tasks a bit more time than the others because we know that there will be a learning curve.
Communication Plan	Progress of the deliverables and the code depends on the frequency and quality of the communication between team members. Since task distribution is made at the meetings, they work like small milestones and we use the project plan to choose the next tasks.

RAM	We needed to change because one of our team members left and we have a new member. In addition to this, we changed some parts according to our implementations. Now, it is more accurate and it will help us to track our progress and to manage our project.

Evaluation of Tools and Processes

Backend

We use Whatsapp as a tool of communication, Python as the programming language, Django as the web framework, PostgreSQL as the database system, Github issues to track our duties, VsCode as the IDE, and Zoom as the video communication tool. We use Whatsapp because we are only 2 people and we are used to using it in our daily life, it is easier for us to respond quickly. It is harder to track documents in Whatsapp but we managed to track them uıntil now so no problem about this.

We use Python because we both feel comfortable to work with it and also we have some web development experience with it from the CmpE 352. Also, there is huge support for this language and the number of people using it is high which leads to easily finding the solutions to the problems that you can encounter. We chose to use Django because it has more functionalities than Flask. In addition, it is used more commonly in the sector because its power is higher. Therefore, we use this opportunity to learn it and use it to understand the details. Until now, we haven't encountered many problems with Python and Django. However, we could not understand how to use Swagger with Django, it is much more complicated than with Flask. Therefore, this is a downside of Django. We plan to figure out this for the next milestones. For now, we document our API in GitHub Wiki.

We use PostgreSQL because it provides easy to use functionality and also easy configuration for Django. This is also one of the database systems that are supported by our customer. We did not encounter any problem while configuring it but when we need to change some fields in a table that is already created and has some data in it, we do not manage to update it. For this, we deleted the table manually in the server and then created it again. We are not sure if it is a problem related to PostgreSQL for Django support but it wasted some time for us to debug.

We do not specify any IDE since it does not make much difference and every team member may choose to use another one based on their past habits. We prefer using VSCode since we use it in our daily life and it has nice features such as code completion and also it is very lightweight.

We tried to use git issues as effectively as possible. We did not make any work without opening an issue to make sure it does not get forgotten. We also tried

to be as specific as possible on our issue descriptions. It was very useful to have a consistent and managed issue system, whenever we found a bug we were able to tag the corresponding issue and fix the problem.

We used Git Actions to automate the unit testing. We configured it so that whenever a pull request is made to the development branch, all the tests found on the test folder are executed. We use this process to make sure that all the merged code works and does not conflict with existing code.

Zoom was our choice of online meeting software. We tried to have our meetings in person, but whenever we could not we met over Zoom. Since we were only two people, there were no time limit and we were able to have our meetings without a problem.

During the development, we used git for the version management system. We opened a branch named “development” and whenever a new feature is implemented, it is opened in a branch from development with the name of the feature. All feature branches are merged to development. With the automated testing explained above, we were able to keep the development branch as a working and ready to deploy branch.

Frontend

We use slack as the communication medium. We periodically informed each other about the progress and the status of the project. We also planned our meetings via slack. We generally used zoom or google meets for our meetings. We often shared our computer’s screens so that we can show each other what we have done easily.

We are implementing our project using javascript programming language and React.js library. We use React-Bootstrap to use reusable and configurable components. However, they did not feel comfortable to one of our friends. So, we decided not to use only one additional library. Everybody is free to use any of the libraries. For example, we will probably use react-select in the future because it has some strong features such as search button.

We chose React just because we are curious about learning it. We did not discuss any other library/framework because we all agreed on it at first stage.

We reviewed our codes carefully. We opened issues and pull requests via github and notified each other via slack as well.

We did not fully operate it but we are planning to obey some syntax rules. We did not agree on a specific IDE, but when we decide on the syntax, we will probably need to decide on an IDE as well.

Android

We have generally used Slack and WhatsApp for chatting communication purposes. We used Zoom and Discord in our meetings while implementing new

functionalities or reviewing the code together. We have opted for the Kotlin language instead of Java and Android Studio as Integrated Development Environment(IDE). As the design pattern, we have chosen MVVM(Model-View-ViewModel).

We have used most of the time Zoom and sometimes Discord as our online meeting tool. Zoom offers a smooth screen sharing experience while writing code together.

The main reason for us choosing Kotlin as the programming language is that it is really concise when compared to Java. Reading and writing Kotlin code is much easier since there is no verbosity. It allows the implementation of the same tasks with less effort. Moreover, it has many new functionalities such as null safety and data classes. Kotlin is being continuously promoted by Google and most of the companies have already switched to Kotlin worldwide. Its community has been growing and flourishing day by day.

Android Studio was our choice of IDE. Android Studio is the most widely used IDE for Android development. The chief advantage of it is that it provides an Android Emulator, which allows us to see and test our app immediately while writing code. It also has a nice Git graphical user interface, which facilitates the usage of Git. One major problem of Android Studio is that it is heavier when compared to other IDEs.

MVVM is the design pattern we chose and followed. This is a structure that is easy to implement and understand. It separates the graphical user interface from the business logic. It has a part connecting to the backend side. It also has a layout part which is a section that is built for users interacting with.

The Requirements addressed in this Milestone

Backend

- 1.1.1.1 - Guests shall be able to register using username and password that they determine and weren't used by any other user before, kinds of sports they are interested in, their skill level, and email address.
- 1.1.1.2 - Guests shall be able to provide additional information such as name, surname, location, age and gender about themselves at the registration stage.
- 1.1.2.1 Guests shall be able to login using their username and password combination.
- 1.1.2.2 Users shall be able to log out whenever they want.
- 1.1.3.1 Users shall update their username, password, location, age, gender, e-mail, skill level, and favorite sports.
- 1.1.3.2 Users shall be able to see the users they followed.
- 1.1.3.6 Users shall be able to get a new password when they forget their passwords.
- 1.2.4.1 Every user shall have a profile page.
- 1.2.4.2 Every user's profile page is visible to other users.
- 1.2.4.3 Profile page of a user shall include the username.

- 1.2.4.9 Users shall be able to follow other users from their profile pages.

Frontend

- 1.1.1.1 - Guests shall be able to register using username and password that they determine and weren't used by any other user before, kinds of sports they are interested in, their skill level, and email address.
- 1.1.1.2 - Guests shall be able to provide additional information such as name, surname, location, age and gender about themselves at the registration stage.
- 1.1.2.1 Guests shall be able to login using their username and password combination.
- 1.1.2.2 Users shall be able to log out whenever they want.
- 1.1.3.1 Users shall update their username, password, location, age, gender, e-mail, skill level, and favorite sports.
- 1.1.3.6 Users shall be able to get a new password when they forget their passwords.
- 1.2.4.1 Every user shall have a profile page.
- 1.2.4.2 Every user's profile page is visible to other users.
- 1.2.4.3 Profile page of a user shall include the username.

Android

- 1.1.1.2 - Guests shall be able to provide additional information such as name, surname, location, age and gender about themselves at the registration stage.
- 1.1.2.1 Guests shall be able to login using their username and password combination.

Summary of Work

Team Member	**Work**
Berk Atıl	I have implemented register, get user, and get sports endpoints and they are used by the client side. Additionally, I have initialized our API in the beginning and made the configuration for PostgreSQL. Also, I dockerized our application and with Elif deployed our application on AWS EC2 Also, we wrote unit tests for each of our endpoints. I created an API Plan wiki page so that the client side can easily access the url, parameters and return types of the endpoints. Also, I reviewed the Pull Requests that are

	opened by Elif and gave some feedback. Also, I have changed one sequence diagram related to user blocking. Lastly, We have reviewed the requirements together.
Deniz Arda Budak	I have edited and improved the “sending request to attend an event” sequence diagram. I reviewed the merge request opened by Ömer. We reviewed the requirements together with my team. We have implemented the login functionality in our Android app with Ömer and Melih. I have attended all the group meetings. I have opened necessary issues and assigned them accordingly.
Ekrem Yusuf Ekmekci	Contributed to the discussion on old versions of our sequence diagrams. Also modified my sequence diagram (creating an event) corresponding to the feedback that we created as a group. I communicated with my teammates in the front-end team, and listed them the information that they should learn. I also prepared some suggestions that we can use during the implementation such as React Router. Then I started studying React, React Router, React-Bootstrap, axios etc. I recreated our app layout. I used rows and columns but at some point I needed dynamic column width. Then I prepared a flexbox that contains sidebar and main content. I also created containers for header and footer parts. Created route handler, context manager and local storage variable. Then I converted our application into a conditionally rendered app. I prepared components to be rendered depending on the route. I modified the sidebar so that it redirects to the profile page properly. I also modified the header and footer to give them a nicer look. I modified the login page and added a function that is triggered when the user is logged in. The function updates user information stored as both context variable and local storage variable. I created profile page. It consists of two cards. One for basic information and the other for sport skills. When user clicks to save button, it updates the user via sending a put request to the backend.
Elif Sema Balcıoğlu	I updated the sequence diagram I made last year with the necessary changes. I created the Reviewing Repository page on GitHub and added the changes made to the deliverables of last year. My first responsibility was to find a data validation solution. I decided it would be best for us to use

	serializers. I then implemented the login functionality with token authentication, after that I created the logout endpoint for all these endpoints I also wrote the unit tests. I worked on password recovery mechanism. I also worked on following-related endpoints. These include following a user, unfollowing, getting the users who follow a user, users that are followed by a user. The responses of these endpoints follow the Jsonld format and Activity Streams and unit tests are written. I reviewed all the pull request opened by Berk. I then created a GitHub workflow for automated testing. Whenever a pull request or a commit to the pull request is made all unit test are executed to make sure no conflicting code is merged to development branch. With Berk, we deployed our application to EC2.
İbrahim Melih Aktaş	I made the necessary changes in the sequence diagram 5. As the Android team with Ömer and Deniz, we created a simple Android project with the MVVM(Model view viewmodel) structure. I helped the implementation of Login functionality. I added the group symbol to the login and register pages. I designed the sign up page. However It is not functional and completed. I tried to solve the problems we encountered before the demo.
Musa Nuri İhtiyar	* I have reviewed sequence diagram 1 since I was the one who prepared it previously. * I have kept meeting notes and uploaded it to the github for one of the front-end meetings * I have initialized a new react project in order to form the project structure for the front-end. * I have tried to prepare header, sidebar and footer elements, all of which are permanent components; nonetheless, more efforts were put by other members in order to make styling better. * I have prepared a password reset page for the front-end.
Ömer Faruk Süve	I have created the boilerplate of our Android Application. I tried to make our opening page layout in Android. In addition, I have implemented the login functionality with Deniz and Melih. I opened a pull request for the login functionality. I asked Deniz and Melih to review my code and give feedback whether it should have been merged or not. Besides coding, I have taken three meeting notes of group meetings for three weeks. I have opened several issues and also I have been assigned several issues by Deniz and Melih. I attended the group meetings and Android

	meetings. Also, I have changed the unblocking user operation sequence diagram.
Salih Can Özçelik	I was a new member of the team. I realized there was some problem with communication so I created Slack and reorganized the channels. Since I was a new member I updated the group photo and updated the project plan. Also there were some issues with merge's and as a team we merged the correct branch to our front-end base. I created login and register page with reactJS, and I wrote the code to be fully functional.

Deliverables

Requirements

1. Functional Requirements

1.1 User Requirements

- 1.1.1 Registration
  - 1.1.1.1 Guests shall be able to register using username and password that they determine and weren't used by any other user before, kinds of sports they are interested in, their skill level, and email address.
  - 1.1.1.2 Guests shall be able to provide additional information such as name, surname, location, age and gender about themselves at the registration stage.
  - 1.1.1.3 Users shall be able to delete their accounts whenever they want.
- 1.1.2 Login
  - 1.1.2.1 Guests shall be able to login using their username and password combination.
  - 1.1.2.2 Users shall be able to log out whenever they want.
- 1.1.3 Profile Management
  - 1.1.3.1 Users shall update their username, password, location, age, gender, e-mail, skill level, and favorite sports.
  - 1.1.3.2 Users shall be able to see the users they followed.
  - 1.1.3.3 Users shall be able to see their event history.
  - 1.1.3.4 Users shall be able to decide which information is visible to others.
  - 1.1.3.5 Users shall be able to see the events they are interested in.

        - 1.1.3.6 Users shall be able to get a new password when they forget their passwords.

1.2 System Requirements

- 1.2.1 Event Recommendations
    - 1.2.1.1 The system shall recommend the events that are for sports selected by the user.
    - 1.2.1.2 The system prioritizes the events for users with the close skill level as the user.
    - 1.2.1.3 The recommended events shall be automatically sorted by ascending distance.
    - 1.2.1.4 The events created by the event creators, whom the user is following, shall be recommended.
- 1.2.2 Notifications
    - 1.2.2.1 The system shall send notifications on events, to which the user is "Interested" when the user is approved or rejected by the event creator.
    - 1.2.2.2 The system shall send notifications when an event in which the user is "Interested" only has few available spots left. The meaning of few shall be determined for each sport separately.
    - 1.2.2.3 The system shall send notifications when all the spots in an event to which the user is approved to participate are taken.
    - 1.2.2.4 The system shall send notifications when an event to which the user has approved starts in a week, a day, and 3 hours.
    - 1.2.2.5 The system shall send notifications if an event to which the user has approved has been canceled by the event creator.
    - 1.2.2.6 The system shall send notifications if details of an event to which the user has been approved has been changed by the event creator.
- 1.2.3 Search Engine
    - 1.2.3.1 The user shall be able to search events.
    - 1.2.3.2 The user shall be able to filter the event results based on sport type, skill level and time.
    - 1.2.3.3 The user shall be able to sort search results based on their distance if location data is shared, starting time, skill level for events.
    - 1.2.3.4 The user shall be able to search users based on their name and username.
    - 1.2.3.5 The user shall be able to search equipment.
    - 1.2.3.6 The user shall be able to filter equipment based on the sport type.

        - 1.2.3.7 The user shall be able to search field.
        - 1.2.3.8 The user shall be able to sort field based on its distance.
        - 1.2.3.9 The user shall be able to search events on map.
    - 1.2.4 Profile Page
        - 1.2.4.1 Every user shall have a profile page.
        - 1.2.4.2 Every user's profile page is visible to other users.
        - 1.2.4.3 Profile page of a user shall include the username.
        - 1.2.4.4 Users shall be given the option to show and hide the full name, profile picture, age, user location, skill level for each sport, personal badges, previously attended events, previously created events.
        - 1.2.4.5 If the user is a field owner, field details, previous events that took place in the fields the user has shall be shown on the profile page of the user.
        - 1.2.4.6 Users shall be able to block other users from their profile pages.
        - 1.2.4.7 Users cannot see the profile of the user who blocks them.
        - 1.2.4.8 Users cannot move to the profile page of the user who blocks them when they see him/her on the event attendee list.
        - 1.2.4.9 Users shall be able to follow other users from their profile pages.

1.3 Event Requirements

- 1.3.1 Creation and Cancellation
    - 1.3.1.1 Any user shall be able to create an event. After creating an event, the host is going to get the "Event Creator" status and have privileges merely for that specific event.
    - 1.3.1.2 The creator for an event shall be able to cancel the event before it starts. In that case, anyone who stated that they are "Coming" to the event or "Interested" in the event shall receive notifications about the cancellation.
    - 1.3.1.3 Every event must have one host and any number of co-hosts.
- 1.3.2 Event Page
    - 1.3.2.1 General Information
        - 1.3.2.1.1 Users can send registration request via event page.
        - 1.3.2.1.2 Users can declare themselves as spectator via event page.
        - 1.3.2.1.3 Users can access information and comments/discussions about the specific event and place via the event page.

        - 1.3.2.2 "About" Tab
            - 1.3.2.2.1 There will be some basic information about the event on this page like "Time", "Location", "Min and max number of participants" and "Description".
            - 1.3.2.2.2 Also, some events might have some constraints for the users who want to participate. These "Requirements" are also shown in the "About" Tab.
            - 1.3.2.2.3 There will be information about the badges to be delivered. After the event, the ones who gained the badges will be displayed.
            - 1.3.2.2.4 Badges to be delivered can be changed by the creator. That is, "event owner to players badges" is customizable. See badges section.
        - 1.3.2.3 "Discussion" Tab
            - 1.3.2.3.1 The event page will also include a "Discussion Page" for people to post their questions, talk about where to find the necessary equipment or offer changes in the plan.
            - 1.3.2.3.2 Who can post or see the posts on this page can be adjusted by the event creator.
        - 1.3.2.4 Unless otherwise stated, users shall be able to see who is "Coming" to the event, "Interested" in the event, and "Wants to Watch" the event.
        - 1.3.2.5 Participants shall be able to share photos and videos after the event if the event creator allows, in the Event Page.
- 1.3.3 Management
    - 1.3.3.1 Depending on the event creator's choices, some events might require an approval from the creator before the participation. In these cases, the user shall seem to be "Interested" in the event and wait for the event creator to approve their participation. They can send an "approval request message" as to why they want to participate to obtain a higher chance of getting an approval.
    - 1.3.3.2 Event creator can make changes in the activity, even after its creation.
- 1.3.4 Communication
    - 1.3.4.1 Users can ask their questions about the event in the "Discussion Page" if permitted.
    - 1.3.4.2 For approval-required events, the users who want to participate can send an approval request and explain why they want to attend.
- 1.3.5 Spectators
    - 1.3.5.1 The users who do not participate as player can be spectator. They also can undo it.

        - 1.3.5.2 Users can declare themselves as spectator via event page. (See 1.3.2.1.2)
        - 1.3.5.3 If the field capacity is less than or equal to the number of spectators, new spectators will be warned about it.
        - 1.3.5.4 If the field capacity is less than the number of spectators, current spectators will be warned about it.

1.4 Badges

- 1.4.1 Badge Types
    - 1.4.1.1 There will be certain badges already designed in the system.
    - 1.4.1.2 Users shall be able to request new type of badges from the system.
    - 1.4.1.3 There will also be badges indicating misbehaviour.
- 1.4.2 Badge Acquirement
    - 1.4.2.1 Users shall be able to acquire badges as event participators.
    - 1.4.2.2 Users shall be able to acquire badges as event creators.
    - 1.4.2.3 The badges shall be given by the system, event participators and event creators.

1.5 Equipment

- 1.5.1 Creation and Deletion
    - 1.5.1.1 Any user shall be able to create an equipment post.
    - 1.5.1.2 The creator of equipment shall be able to delete the equipment from the system.
- 1.5.2 Equipment Page
    - 1.5.2.1 General Information
        - 1.5.2.1.1 Users can access information and comments/discussions about the specific equipment and place via the equipment page.
    - 1.5.2.2 "About" Tab
        - 1.5.2.2.1 There will be some basic information about the equipment on this page like "Location", "Related Sports" and "Description".
    - 1.5.2.3 "Discussion" Tab
        - 1.5.2.3.1 The equipment page will also include a "Discussion Page" for people to post their questions, talk about where to find the equipment.
        - 1.5.2.3.2 Participants shall be able to share photos and videos about the equipment.

2. Non-Functional Requirements

2.1 Design

- 2.1.1 There will be an Android application and a web platform. Both of them will have identical functionalities.

2.2 Annotations

- 2.2.1 W3C Activity Stream standard 2.0 will be used in order to implement subscription and notification functionalities.
- 2.2.2 Wikidata.org will be used to support semantic taggings.

2.3 Legal and Ethical Issues

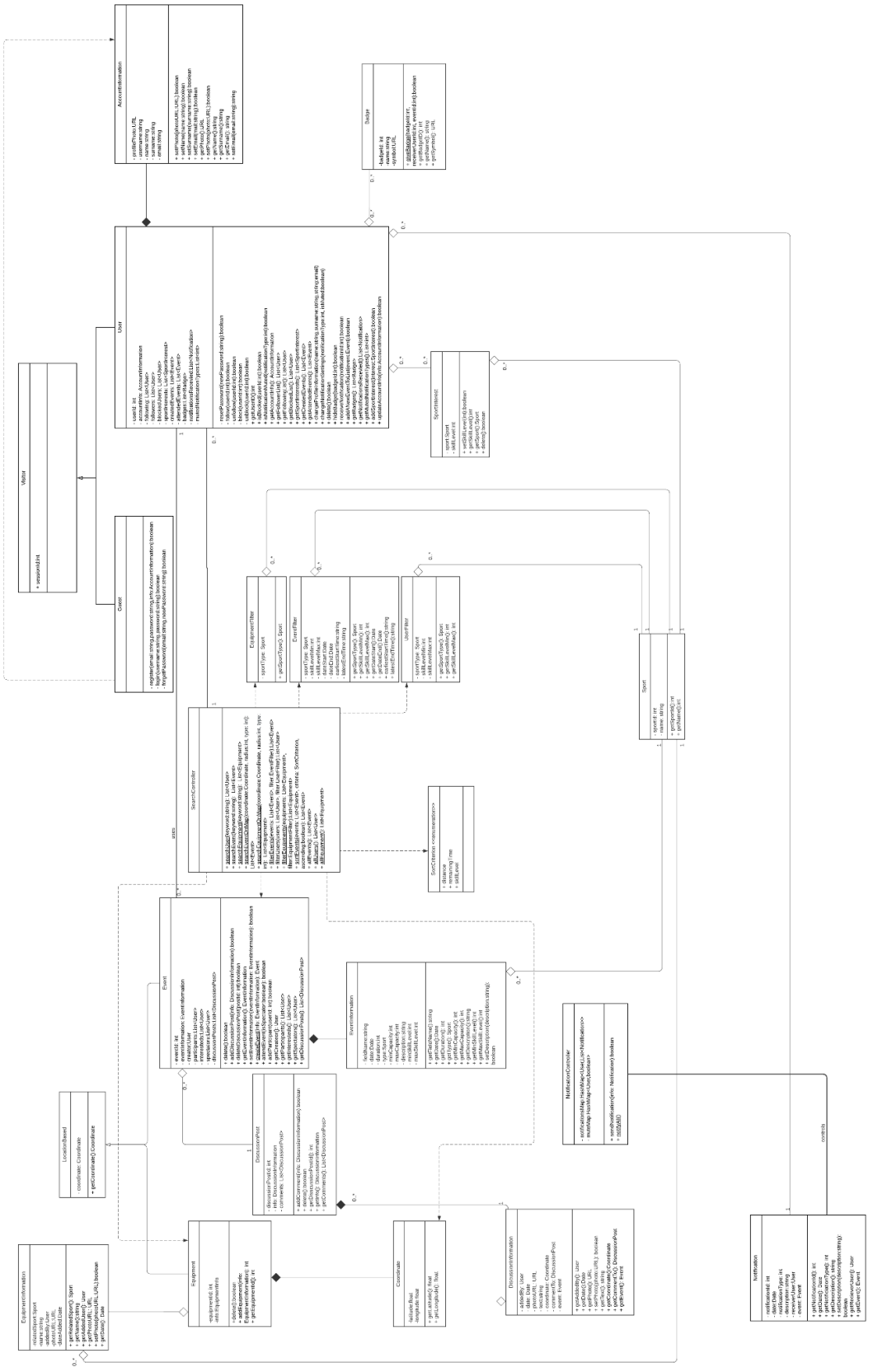
- 2.3.1 The rules of GDPR and KVKK will be followed regarding the "personal information", "contact information", "copyrighted contents", "licence issues" etc.
- 2.3.2 There will be documents about "Terms of Use" and "Privacy Policy" of the application and they must be accepted during the registration.
- 2.3.3 The Project will consider the Ethical Values.
- 2.3.4 There will be no tolerance for discriminatory speech or content.

2.4 Implementation Details

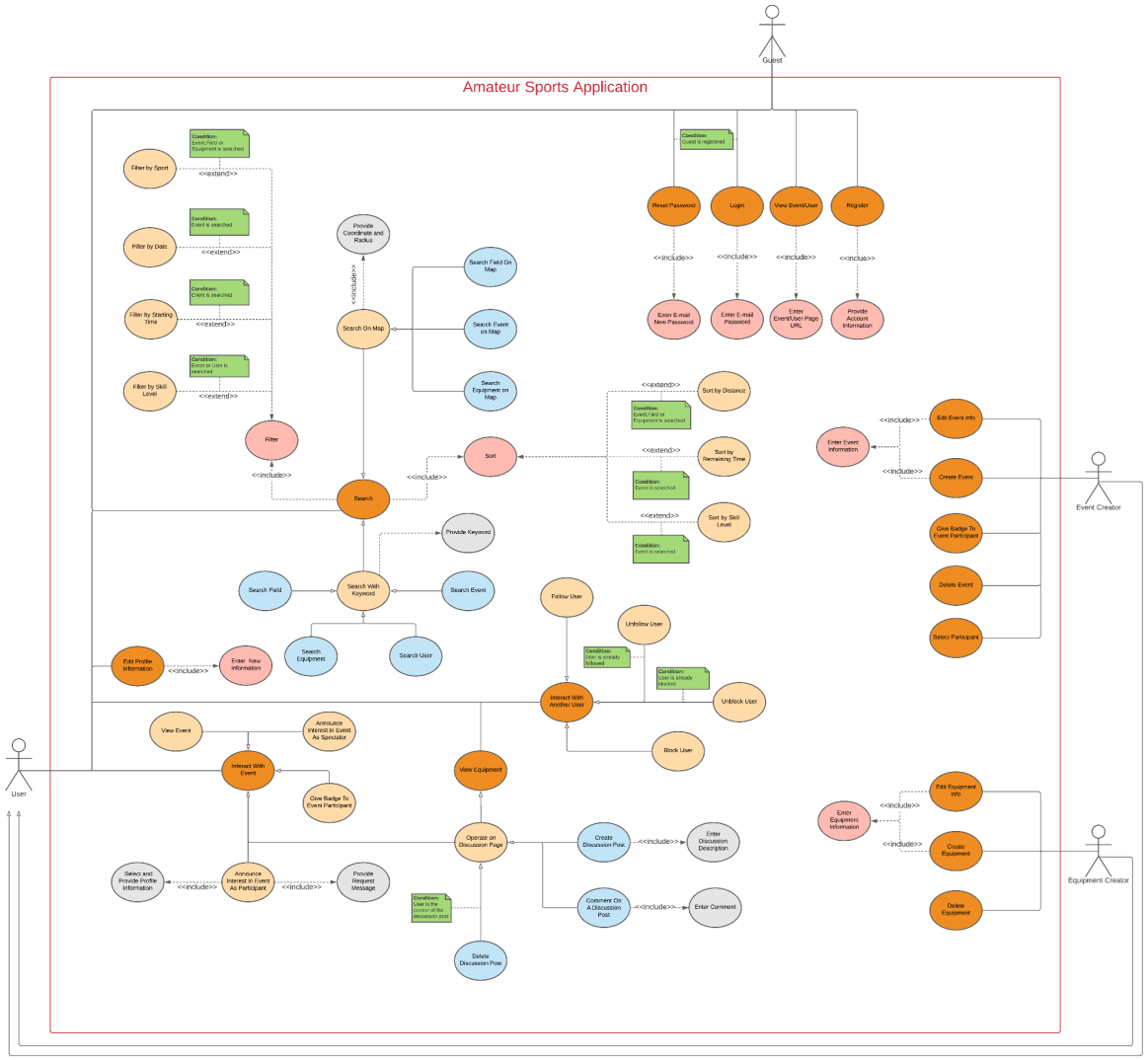
- 2.4.1 Front-end will be created using React.
- 2.4.2 Back-end will be created using Django.
- 2.4.3 Android will be created using Kotlin.

UML Software Designs

Class Diagram



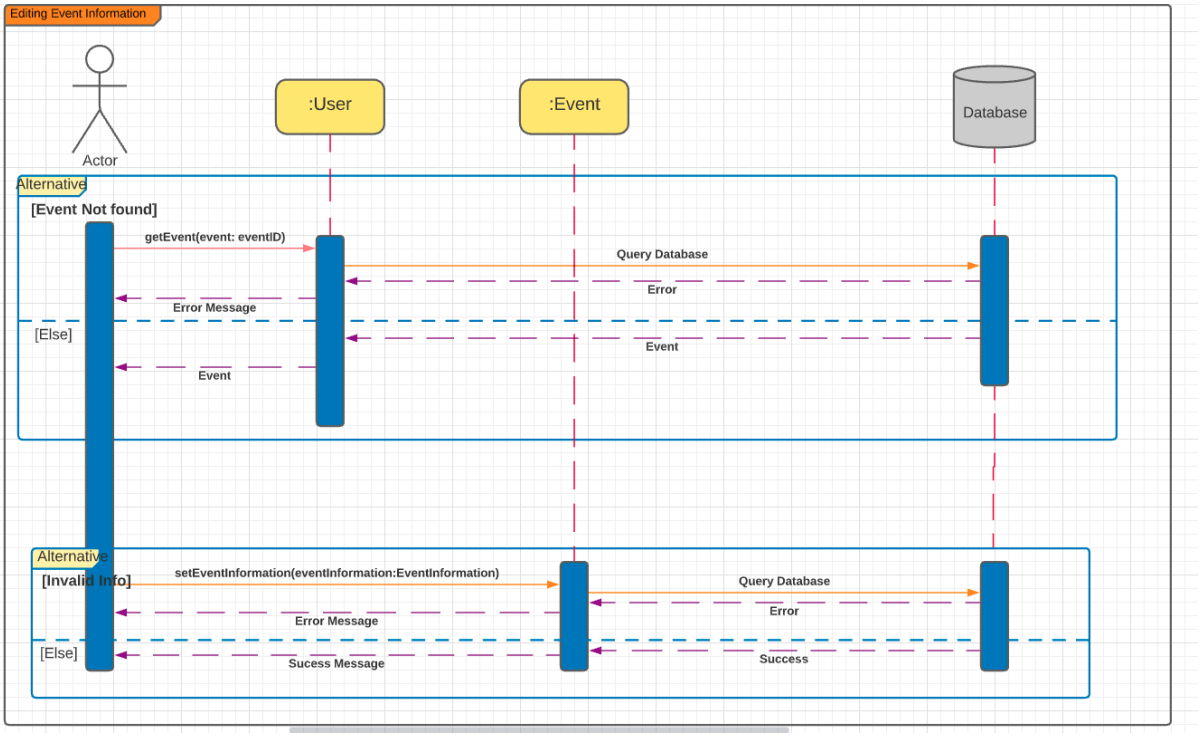
Use Case Diagram



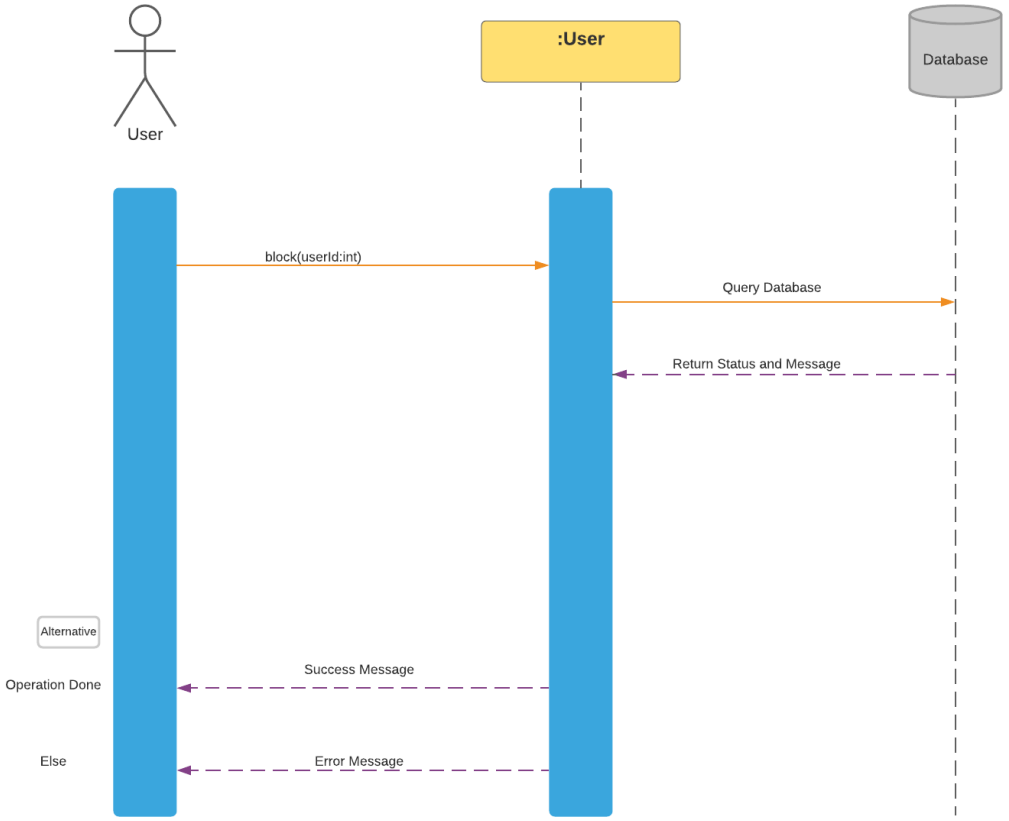
Sequence Diagrams

We have 9 sequence diagrams. Each of them corresponds to a different mockup scenario.

1. Editing Event Information

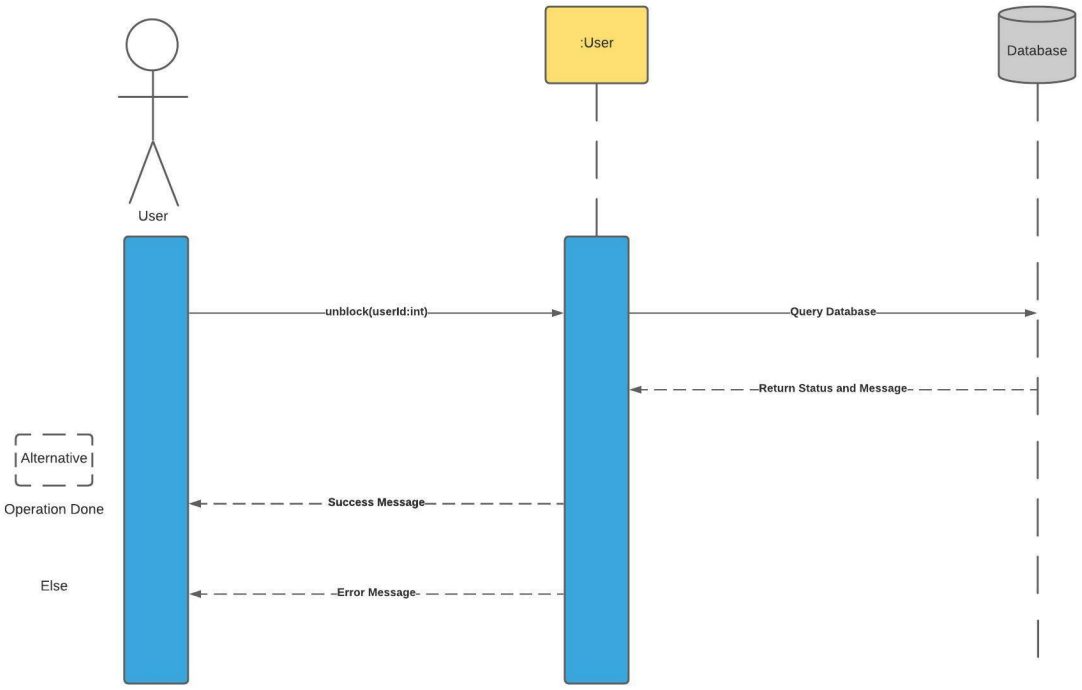


2. Blocking a User



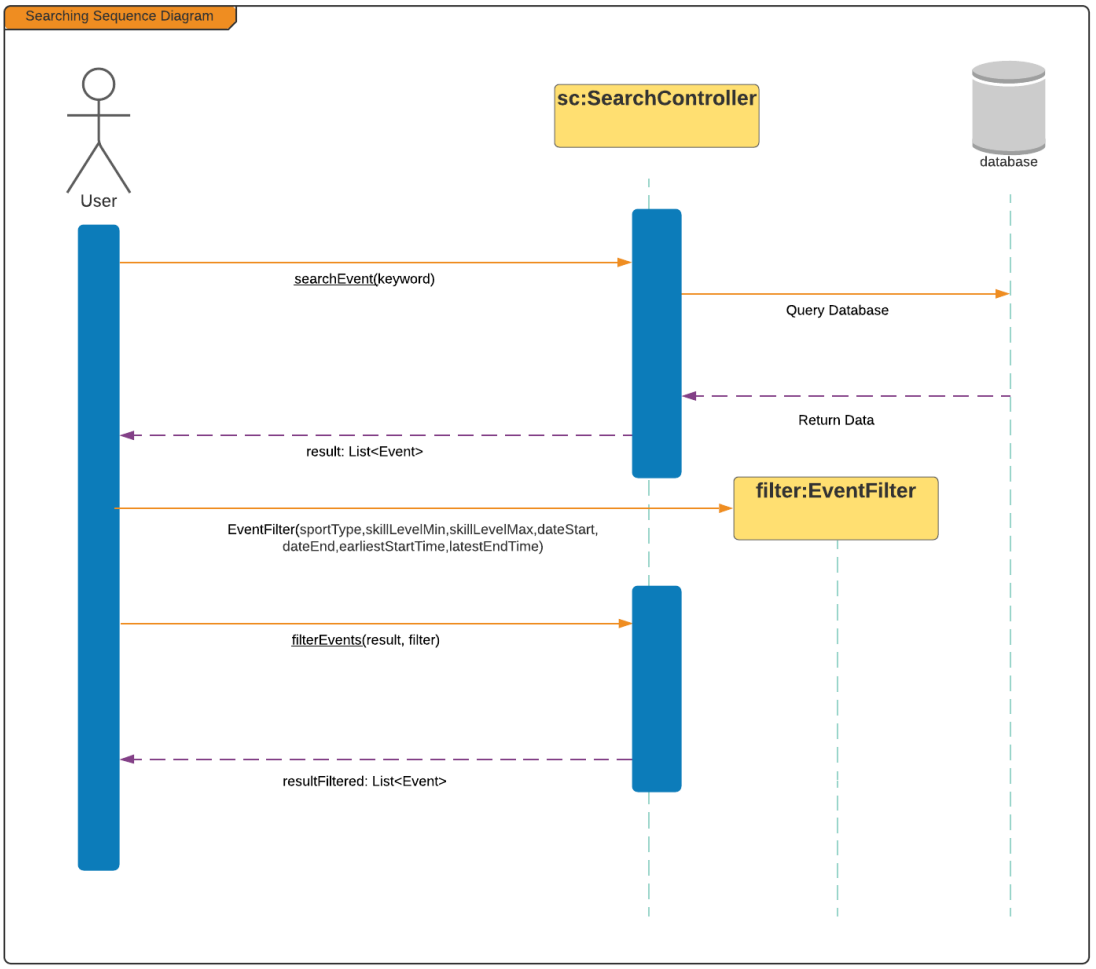
3. Unblocking a User

User unblocks another user.

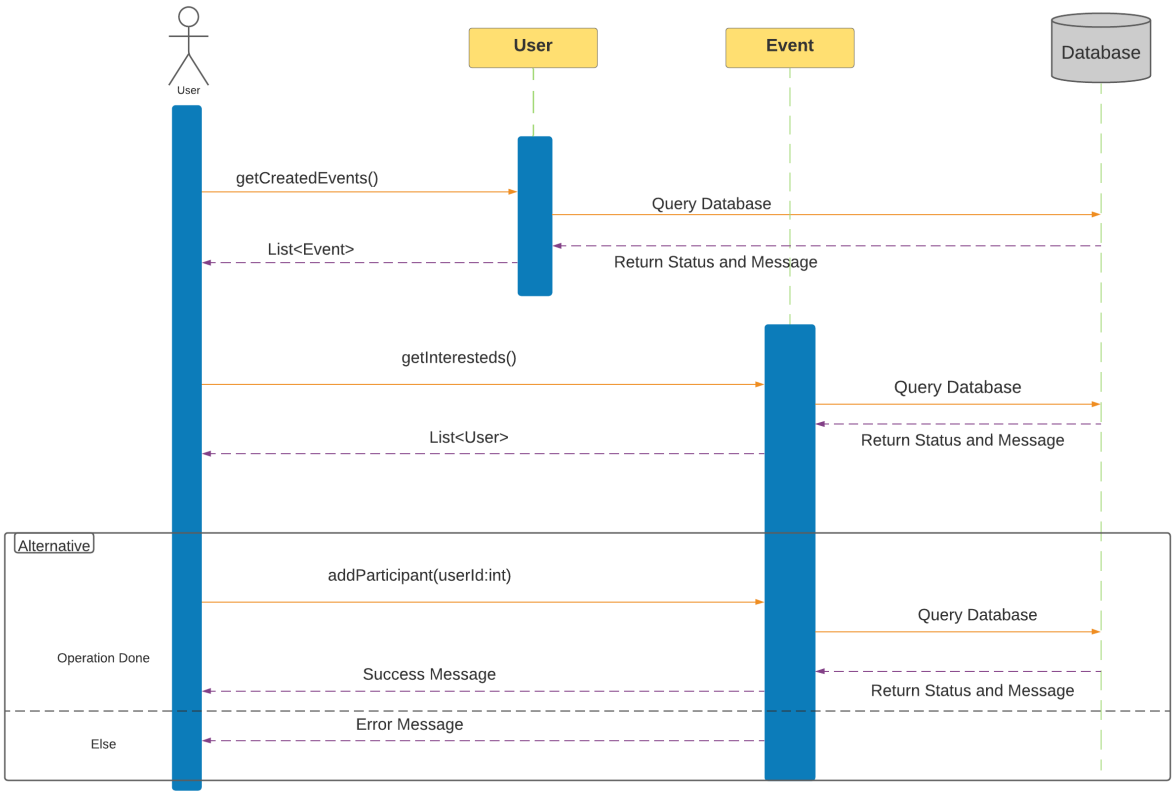


4. Searching and Filtering Events

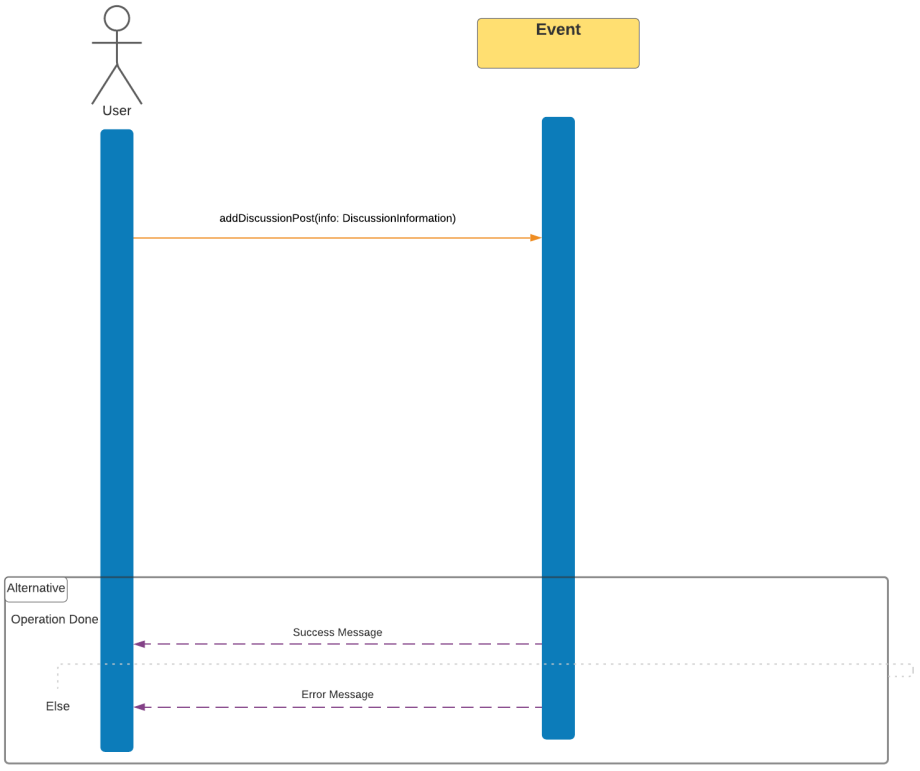
User searches events and after seeing the results, decides to filter and sort these items.



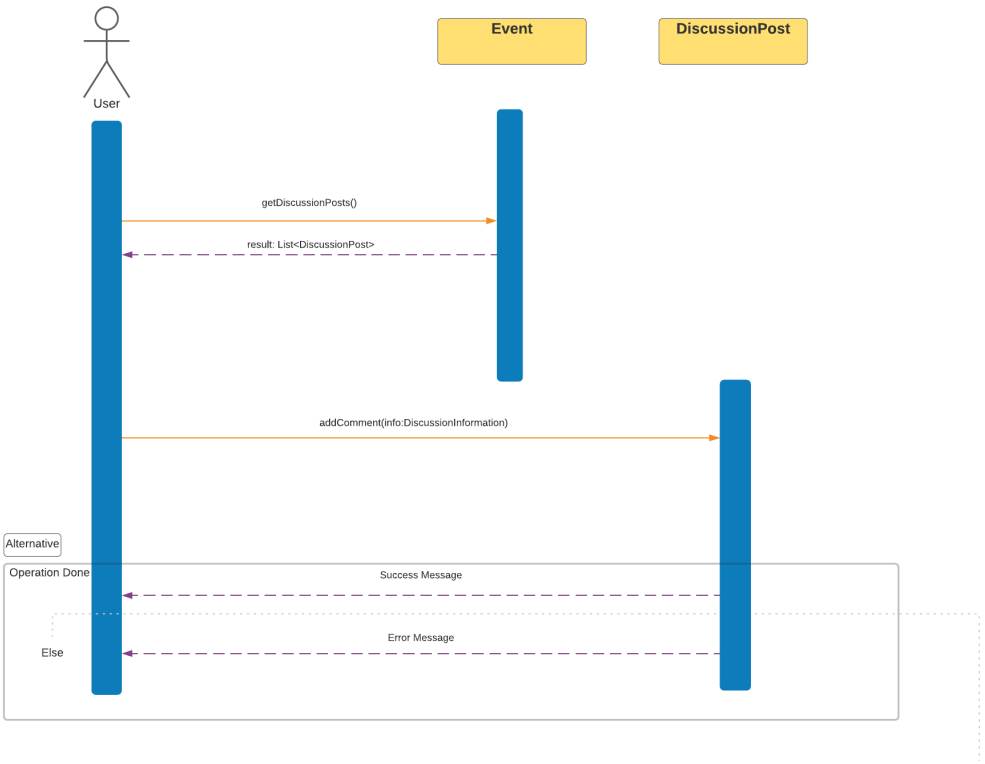
5. Adding Participants



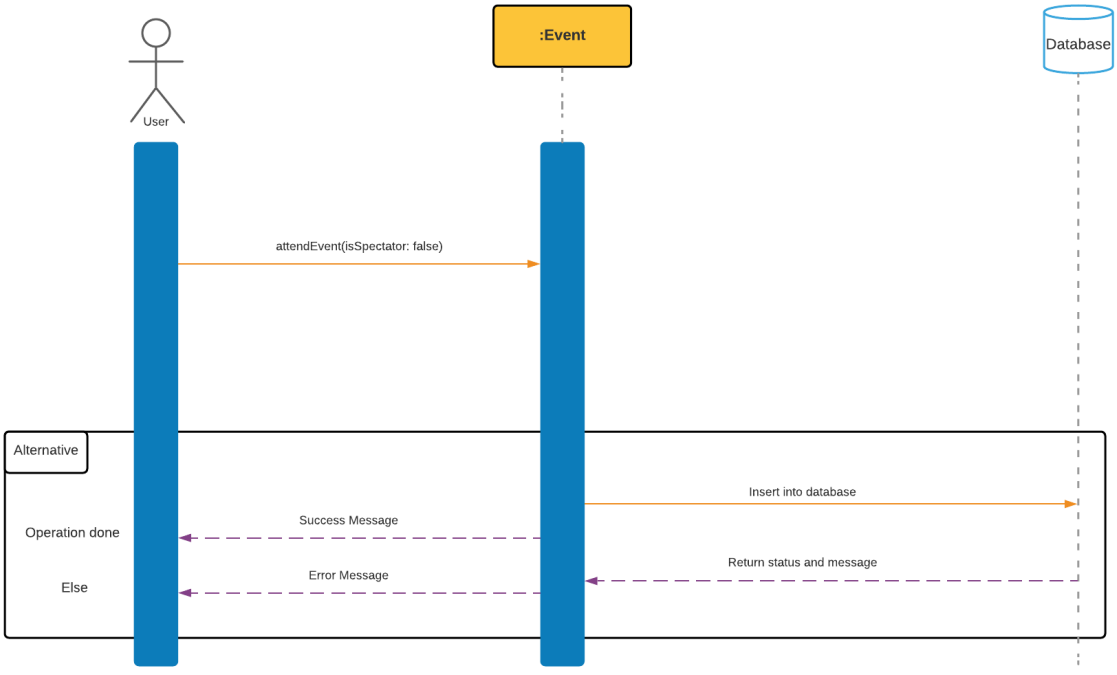
6. Creating a New Discussion Post



7. Posting Comments in the Discussion Page

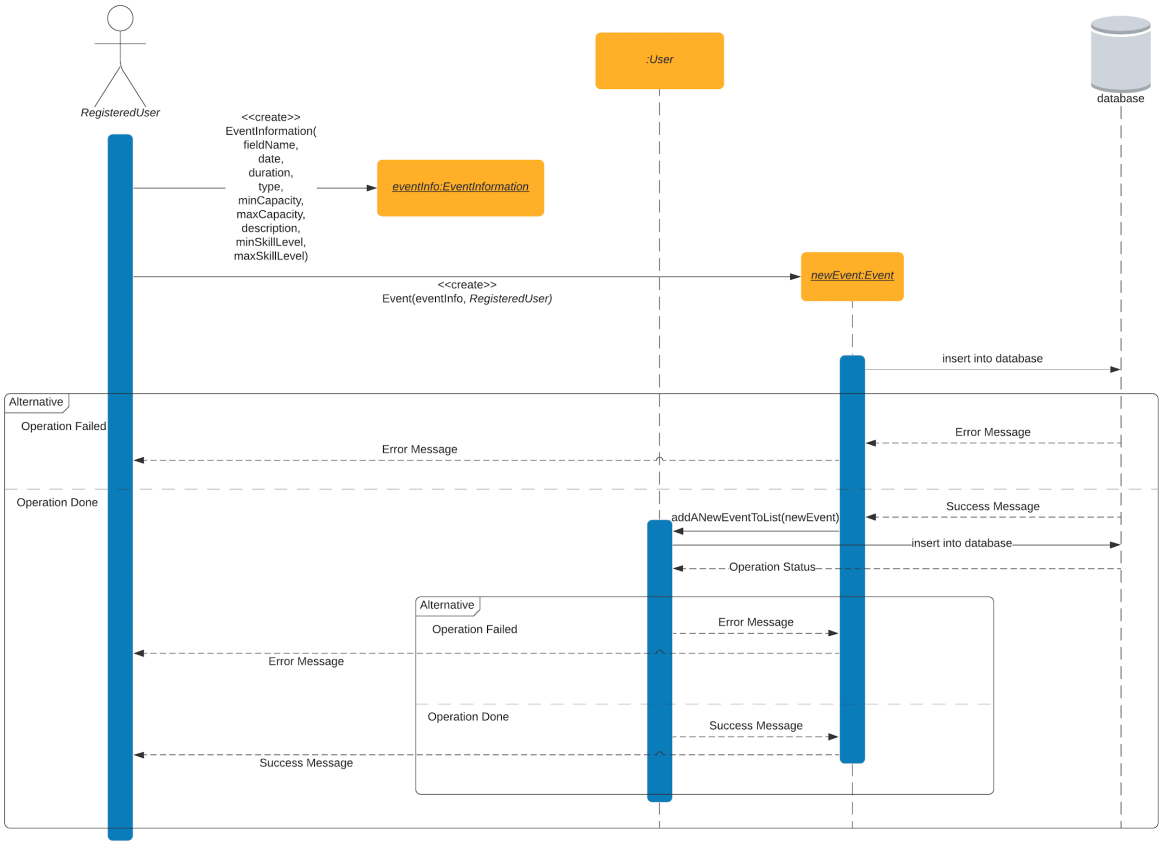


8. Sending request to attend an event



Note: It is assumed that the event has already been found by the user

9. Creating an Event



Scenarios and mockups

User Block Scenario 1	https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Scenario • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Website-Mockup-1 • Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Mobile-Mockup-1
User Block Scenario 2	https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Scenario-2 • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Website-Mockup-2

	• Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Mobile-Mockup-2
User Block Scenario 3	https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Scenario-3 • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Website-Mockup-3 • Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/User-Block-Mobile-Mockup-3
Search an Event Scenario	https://github.com/bounswe/2021SpringGroup6/wiki/Search-an-Event-Scenario • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Search-an-event--website-mock-up • Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Search-an-event-mobile-mock-up
Send Request to Attend an Event Scenario	https://github.com/bounswe/2021SpringGroup6/wiki/Send-Request-to-Attend-an-Event-Scenario • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Send-request-to-attend-an-event-website-mock-up • Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Send-request-to-attend-an-event-website-mobile-mock-up
Person Selection for an Event Scenario	https://github.com/bounswe/2021SpringGroup6/wiki/Person-Selection-for-an-Event-Scenario • Website mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Person-Selection-Website-Mockup • Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Person-Selection-Mobile-Mockup
Event Creation Scenario	https://github.com/bounswe/2021SpringGroup6/wiki/Event-Creation-Scenario • Website mockup:

	https://github.com/bounswe/2021SpringGroup6/wiki/Event-Creation-Mockup-(Website) ● Mobile mockup: https://github.com/bounswe/2021SpringGroup6/wiki/Event-Creation-Mockup-(Mobile)
Event Creation Scenario	● Website Mockup ● Mobile Mockup
Checking the Event Page and Posting a Discussion Scenario	● Website Mockup ● Mobile Mockup

RAM

Lead(L) Contributor(C) Review(R) Approval(A) None(N)	Berk Atıl	Salih Can Özçelik	Deniz Arda Budak	Ekrem Yusuf Ekmekci	Elif Sema Balcıoğlu	İbrahim Melih Aktaş	Musa Nuri İhtiyar	Ömer Faruk Süve
Group Meetings	C	C	C	C	C	C	C	C
Back-end Meetings	C	N	N	N	C	N	N	N
Front-end Meetings	N	C	N	C	N	N	C	N
Android Meetings	N	C	N	N	N	C	N	C
Implementation Start								
Front-End: First Design	N	R	N	C	N	N	L	N
Back-End: Server Trial	L	N	N	N	A	N	N	N
Android: First Design	N	N	R	N	N	L	N	L
Login-Register								
Front-End								
Front-End: Register-Login Page	N	L	N	R	N	N	R	N
Front-End: Password change-reset Page	N	R	N	R	N	N	L	N
Front-End: Home Page	N	R	N	L	N	N	R	N
Back-End								
Back-End: Login endpoint	A	N	R	R	L	N	N	N
Back-End: Register endpoint	L	N	N	R	A	R	N	N
Back-End: Password change-reset Page	A	N	N	N	L	N	R	N
Android								
Android: Register-Login Page	N	A	N	R	N	L	N	L
Android: Password change-reset Page	N	L	N	N	R	A	N	L
Android: Home Page	N	L	N	N	N	L	R	A
Create Milestone 1 (451)	C	C	C	C	C	C	C	C
Event - User Profile								
Front-End								
Front-End: Event Page Design	N	R	N	R	N	N	L	N
Front-End: Event Creation	N	L	N	R	N	N	R	N

Lead(L) Contributor(C) Review(R) Approval(A) None(N)	Berk Atıl	Salih Can Özçelik	Deniz Arda Budak	Ekrem Yusuf Ekmekci	Elif Sema Balcıoğlu	İbrahim Melih Aktaş	Musa Nuri İhtiyar	Ömer Faruk Süve
Front-End: Event Discussion Page	N	R	N	L	N	N	R	N
Front-End: Profile Page	N	R	N	R	N	N	L	N
Front-End: Profile Creation	N	L	N	R	N	N	R	N
Front-End: Follow-Unfollow mechanism	N	R	N	L	N	N	R	N
Back-End								
Back-End: Event System and database	L	N	R	N	A	N	N	N
Back-End: Event Creation	A	N	N	R	L	R	N	N
Back-End: Event Discussion Page	A	N	N	N	L	N	R	N
Back-End: Follow-Unfollow mechanism	A	N	R	N	L	N	N	N
Back-End: Get User	L	N	N	N	A	N	R	N
Back-End: Block-Unblock Mechanism	L	N	N	N	A	N	R	N
Back-End: Get Sports	L	N	N	R	A	N	N	N
Android								
Android: Event Page Design	R	A	N	N	N	L	N	L
Android: Event Creation	N	L	R	N	N	A	N	L
Android: Event Discussion Page	N	L	N	R	N	L	N	A
Android: Profile Page	N	A	N	N	R	L	N	L
Android: Profile Creation	N	L	N	N	N	A	R	L
Android: Follow-Unfollow mechanism	R	L	N	N	N	L	N	A
Search								
Front-End								
Front-End: Search Page	N	R	N	R	N	N	R	N
Front-End: Google Maps	N	R	N	R	N	N	L	N
Front-End: Search Sort	N	L	N	R	N	N	R	N
Front-End: Search Filter	N	R	N	L	N	N	R	N
Back-End								

Lead(L) Contributor(C) Review(R) Approval(A) None(N)	Berk Atıl	Salih Can Özçelik	Deniz Arda Budak	Ekrem Yusuf Ekmekci	Elif Sema Balcıoğlu	İbrahim Melih Aktaş	Musa Nuri İhtiyar	Ömer Faruk Süve
Back-End: Search system	L	N	N	R	A	R	N	N
Back-End: Location implementation	A	N	N	N	L	N	R	N
Back-End: Search Sort	A	N	N	N	L	N	N	R
Back-End: Search Filter	L	R	N	N	A	N	N	N
Android								
Android: Search Page	N	A	R	N	N	L	N	L
Android: Google Maps	N	L	N	R	N	A	N	L
Android: Search Sort	N	L	N	N	R	L	N	A
Android: Search Filter	N	A	N	N	N	L	R	L
Create Milestone 2 (451)	C	C	C	C	C	C	C	C
Badge and Equipment								
Front-End								
Front-End: Badge Design	N	R	N	R	N	N	L	N
Front-End: Badge Implementation	N	L	N	R	N	N	R	N
Front-End: User block system	N	R	N	L	N	N	R	N
Front-End: Participant Selection	N	R	N	R	N	N	L	N
Front-End: Equipment Creation	N	L	N	R	N	N	R	N
Front-End: Addition of equipment to search	N	R	N	L	N	N	R	N
Back-End								
Back-End: Badge Implementation	A	N	N	N	L	N	R	N
Back-End: User block system	A	N	N	N	L	N	N	R
Back-End: Participant Selection	L	R	N	N	A	N	N	N
Back-End: Equipment Creation	A	N	N	N	L	R	N	N
Back-End: Addition of equipment to search	L	N	R	N	A	N	N	N
Android								
Android: Badge Design	R	A	N	N	N	L	N	L

Lead(L) Contributor(C) Review(R) Approval(A) None(N)	Berk Atıl	Salih Can Özçelik	Deniz Arda Budak	Ekrem Yusuf Ekmekci	Elif Sema Balcıoğlu	İbrahim Melih Aktaş	Musa Nuri İhtiyar	Ömer Faruk Süve
Android: Badge Implementation	N	L	R	N	N	A	N	L
Android: User block system	N	L	N	R	N	L	N	A
Android: Participant Selection	N	A	N	N	R	L	N	L
Android: Equipment Creation	N	L	N	N	N	A	R	L
Android: Addition of equipment to search	R	L	N	N	N	L	N	A
Create Milestone 3 (451)	C	C	C	C	C	C	C	C

Project Plan

It's a very big file, in terms of the number of pages. You can check it from that link: https://github.com/bounswe/2021SpringGroup6/blob/master/deliverables/451%20Project%20Plan.pdf

Communication Plan

Participants	Aim	Place	Time
All Team Members	Evaluating the current progress and distribution of the tasks for the next week	Zoom	Every Tuesday @19.00
All Team Members	Urgent problems/issues	WhatsApp, Slack, Phone Call	Anytime Needed
All Team Members	Progress and issue tracking	Github	Anytime Needed
Customer/Instructor and Available Team Members	Discussing the details of the product and clarifying confusing parts of the project	Zoom	Anytime Needed
Back-end Team	Discussing what we have done and what should be done for the next week. New features and bugs are assigned to members	Face-to-face, at the campus	Every Wednesday @19.00
Front-end Team	Discussing what has been done and how it was done. Also talking about our plan for the future, especially next week.	Zoom	Every Tuesday @21.00

Android Team	Discussing what has been done and how it was done with talking about what should be done for the future and what we need to learn.	Zoom	Every Tuesday @20.00

API

Deployed API Base Url: 18.217.235.17:8080

API Documentation

- User Related Endpoints
  - /users POST - Create a User
  - /users/login POST - Login
  - /users/logout POST - Logout
  - /users/<user_id>/following POST - Follow User
  - /users/<user_id>/following DELETE - Unfollow User
  - /users/<user_id>/following GET - Get Users Followed By User
  - /users/<user_id>/follower GET - Get Users Follow User
  - /users/<user_id> GET - Get User
  - /users/<user_id> PUT - Update User Information
  - /users/recover POST - Recover Password
- Sport Related Endpoints
  - /sports GET - Get Sports

User Related Endpoints

/users POST - Create a User

Endpoint: /users
Method: POST
Authorization: Not Needed
Author: Berk Atıl
Body Parameters:

```
 {
             “email” :Email of the user, required
             “password” : Password of the user. It should be at least 8 characters
and at most 15 characters. Alphanumeric characters, _, *, and . are allowed.
required
             “identifier” : Identifier(username) of the user. required
             “name” : First name of the user.
             “familyName” : Last name of the user.
             “birthDate” : Birth date of the user. It should be YYYY-MM-DD format.
             “gender” : Gender of the user. It can be male, female or
decline_to_report.
             “sports” : Skill levels of the user for different sports. It is a list
of dictionaries and each dictionary contain sport name and skill level
}
```

Responses:
201: ""
400: "Body parameters are not correct."
Example Response:

```
{
        "message": "There is an error regarding the provided data"
}
```

400: "Given sport is not supported."
Example Response:

```
{
        "message": "Given sport is not supported."
}
```

400: "Username is already taken"
Example Response:

```
{
        "message": "Username is already taken"
}
```

400: "Email is already taken"
Example Response:

```
{
        "message": "Email is already taken"
}
```

400: "There is an integrity error."
Example Response:

```
{
        "message": "There is an integrity error."
}
```

500:"There is an internal error, try again later."

```
{
        "error": "There is an internal error, try again later."
}
```

/users/login POST - Login

Endpoint: /users/login/
Method: POST
Authorization: Not Needed
Author: Elif Sema Balcıoğlu
Body Parameters:

```
 {
            “identifier” : Identifier(username) of the user. required
            “password” : Password of the user. required
}
```

Expected Response When No Error:

```
{
            "token": Authorization token for the user.
            "user_id": Id of the user.
}
```

Responses:
200: "User is authenticated." Example Response:

```
{
             "token": "952c07e2afc8fd0307fd3eb5c130526d3d43ac3c",
             "user_id": 5,
}
```

400: "Body parameters are not correct."
Example Response:

```
{
    "message": {
        "identifier": [
            "Ensure this field has at least 3 characters."
        ]
    }
}
```

403: "Credentials are not correct."
Example Response:

```
{
    "message": "Check credentials."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/logout POST - Logout

Endpoint: /users/logout/
Method: POST
Authorization: Needed
Author: Elif Sema Balcıoğlu
Header Parameters:

```
Authorization: Token <Token of the user>
```

Expected Response When No Error:

```
{
    "message": "Successfully logged out."
}
```

Responses:
200: "User successfully logged out." Example Response:

```
{
    "message": "Successfully logged out."
}
```

401: "Invalid token."
Example Response:

```
{
    "message": "Invalid token."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/<user_id>/following POST - Follow User

Endpoint: /users/<user_id>/following/
Method: POST
Authorization: Needed
Author: Elif Sema Balcıoğlu
Path Parameters:

```
user_id: user_id of the following user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Body Parameters:

```
{
            “user_id”: Id of the user to follow. required
}
```

Responses:
200: "User successfully followed."
401: "Invalid token."
Example Response:

```
{
    "message": "Invalid token."
}
```

```
{
    "message": "Login required."
}
```

403: "Not allowed to follow for another user."

```
{
    "message": "Not allowed to follow for another user."
}
```

400: "Input not correct."

```
{
    "user_id": [
        "A valid integer is required."
    ]
}
```

```
{
    "message": "User cannot follow itself."
}
```

```
{
    "message": "Enter a valid user_id to follow."
}
```

```
{
    "message": "User already followed."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/<user_id>/following DELETE - Unfollow User

Endpoint: /users/<user_id>/following/
Method: DELETE **Authorization:** Needed
Author: Elif Sema Balcıoğlu
Path Parameters:

```
user_id: user_id of the following user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Body Parameters:

```
{
            “user_id”: Id of the user to unfollow. required
}
```

Responses:
200: "User successfully unfollowed."
401: "Invalid token."
Example Response:

```
{
    "message": "Invalid token."
}
```

```
{
    "message": "Login required."
}
```

403: "Not allowed to unfollow for another user."

```
{
    "message": "Not allowed to unfollow for another user."
}
```

400: "Input not correct."

```
{
    "user_id": [
        "A valid integer is required."
    ]
}
```

```
{
    "message": "Enter a valid user_id to follow."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/<user_id>/following GET - Get Users Followed By User

Endpoint: /users/<user_id>/following **Method:** GET
Authorization: Needed
Author: Elif Sema Balcıoğlu
Path Parameters:

```
user_id: user_id of the following user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Responses:
200: Example Response:

```
{
    "@context": "https://www.w3.org/ns/activitystreams",
    "summary": "lion3's following activities.",
    "type": "Collection",
    "total_items": 1,
    "items": [
        {
            "@context": "https://www.w3.org/ns/activitystreams",
            "summary": "lion3 followed esb",
            "type": "Follow",
            "actor": {
                "type": "https://schema.org/Person",
                "@id": 16,
                "identifier": "lion3"
            },
            "object": {
                "type": "https://schema.org/Person",
                "@id": 3,
                "identifier": "esb"
            }
        }
```

```
    ]
}
```

401: "Invalid token."
Example Response:

```
{
    "message": "Invalid token."
}
```

```
{
    "message": "Login required."
}
```

400: "Input not correct."

```
{
    "message": "User does not exist."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/<user_id>/follower GET - Get Users Follow User

Endpoint: /users/<user_id>/follower **Method:** GET
Authorization: Needed
Author: Elif Sema Balcıoğlu
Path Parameters:

```
user_id: user_id of the following user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Responses:
200: Example Response:


```
{
    "@context": "https://www.w3.org/ns/activitystreams",
    "summary": "esb's being followed activities.",
    "type": "Collection",
    "total_items": 3,
    "items": [
        {
            "@context": "https://www.w3.org/ns/activitystreams",
            "summary": "lion followed esb",
            "type": "Follow",
            "actor": {
                "type": "https://schema.org/Person",
                "@id": 8,
                "identifier": "lion"
            },
            "object": {
                "type": "https://schema.org/Person",
                "@id": 3,
                "identifier": "esb"
            }
        },
        {
            "@context": "https://www.w3.org/ns/activitystreams",
            "summary": "esb18 followed esb",
            "type": "Follow",
            "actor": {
                "type": "https://schema.org/Person",
                "@id": 10,
                "identifier": "esb18"
            },
            "object": {
                "type": "https://schema.org/Person",
                "@id": 3,
                "identifier": "esb"
            }
        },
        {
            "@context": "https://www.w3.org/ns/activitystreams",
            "summary": "lion3 followed esb",
            "type": "Follow",
            "actor": {
                "type": "https://schema.org/Person",
                "@id": 16,
                "identifier": "lion3"
            },
            "object": {
                "type": "https://schema.org/Person",
```

```
                "@id": 3,
                "identifier": "esb"
            }
        }
    ]
}
```

401: "Invalid token."
Example Response:

```
{
    "message": "Invalid token."
}
```

```
{
    "message": "Login required."
}
```

400: "Input not correct."

```
{
    "message": "User does not exist."
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

/users/<user_id> GET - Get User

Endpoint: /users/<user_id>
Method: GET
Authorization: Not Needed to see other users but needed to see own data.
Author: Berk Atıl

Path Parameters:

```
user_id: user_id of the desired user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Expected Response When No Error:

```
{
            "email": Email of the user.
            "user_id": Id of the user.
            "name": First name of the user
            "familyName": Last name of the user.
            "birthDate": Birth date of the user.
            "gender": Gender of the user.
            "@context": Context of the user based on scheme.
            "@id": Id of the user.
            "knowsAbout": Skill levels of the user for different sports.
            "@type":
}
```

Responses:
200: "User is returned." Example Response:

```
{
            "email": "berk@g.com",
            "user_id": 1,
            "name": "Berk",
            "familyName": "",
            "birthDate": null,
            "gender": "male",
            "@context": "https://schema.org/Person",
            "@id": 1,
            "knowsAbout": [
                          {
                           "@type": "PropertyValue",
                           "name": "soccer",
                           "value": 2
                          }]
            "@type": "Person"
}
```

401: "User does not exist"
Example Response:

```
{
    "message": "User id does not exist"
```

```
}
```

500: "An error occurred, please try again later"

```
{
    "message": "An error occurred, please try again later"
}
```

/users/<user_id> PUT - Update User Information

Endpoint: /users/<user_id>
Method: PUT
Authorization: Needed
Author: Berk Atıl

Path Parameters:

```
user_id: user_id of the desired user, integer.
```

Header Parameters:

```
Authorization: Token <Token of the user>
```

Body Parameters:

```
 {
            “email” :Email of the user
            “name” : First name of the user.
            “familyName” : Last name of the user.
            “birthDate” : Birth date of the user. It should be YYYY-MM-DD format.
            “gender” : Gender of the user. It can be male, female or
decline_to_report.
            “sports” : Skill levels of the user for different sports. It is a list
of dictionaries and each dictionary contain sport name and skill level
}
```

Responses:
200: "User information is updated." **400**: "Input not correct."

```
{
    "gender": [
        "The gender could be male, female or decline_to_report."
```

```
    ]
}
```

401: "Login required"
Example Response:

```
{
    "message": "Login required."
}
```

500: "An error occurred, please try again later"

```
{
    "message": "An error occurred, please try again later"
}
```

/users/recover POST - Recover Password

Endpoint: /users/recover
Method: POST
Authorization: Needed
Author: Elif Sema Balcıoğlu
Body Parameters:

```
{
            “email”: email of the user who forgot their password. required
}
```

Responses:
200: "User successfully unfollowed." Example Response:

```
{
    "message": "If email provided is correct, a reset password is sent, please
check spam."
}
```

401: "Already logged in."
Example Response:

```
{
    "message": "Already logged in use change password on settings instead."
```

```
}
```

400: "Input not correct."

```
{
    "message": {
        "email": [
            "Enter a valid email address."
        ]
    }
}
```

500:"There is an error, try later."

```
{
    "message": "Try later."
}
```

Sport Related Endpoints

/sports GET - Get Sports

Endpoint: /sports
Method: GET
Authorization: Not Needed.
Author: Berk Atıl

Expected Response When No Error:

```
{
            "sport_names": ['soccer','basketball',...]
}
```

Responses:
200: "Sports are returned" Example Response:

```
{
            "sport_names": ['soccer','basketball',...]
}
```

500:"There is an internal error, try again later."

```
{
    "message": "There is an internal error, try again later."
}
```