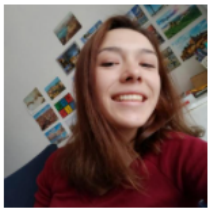
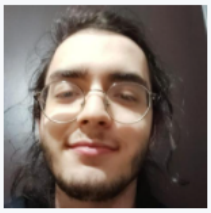
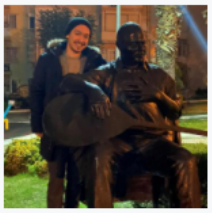
CMPE451-MILESTONE REPORT 2

18/12/2021

Group 2

				
Doğukan Akar	Şefika Akman	Ömer Arslan	İbrahim Kağan Bayat	Berkay Döner
				
A. Necip Görgülü	Ege Can Kaya	Bengisu Özaydın	Mehmet Saraçoğlu	Kerem Zaman

Executive Summary **3**

Summary of Work Done by Each Team Member **4**

Progress According to Requirements **8**
Glossary 8
1. Functional Requirements 8
1.1 User Requirements 8
1.2 System Requirements 9
2. Non-Functional Requirements 13

API Endpoints **14**

Implementation of Core Functionality **29**
Web 29
Android 31

User Interface/User Experience **33**
Homepage and navigation bar/header 33
Signup 35
Login 36
Forgot password 38
Profile Page 39
Create Event Page 41
Search Events 42
Comments & Answers Page 45

Standards **46**
W3C Activity Stream 2.0 46
Wikidata 46

Executive Summary

Android Status: Registration, login, view profile, update profile, search event, view all events functions are completed. Map functionality added to search event and create event pages. All features are connected to the database and working just fine. Some field checks are missing for search event and create event but we are going to fix them as soon as possible.
Android Moving Forward: First we are going to implement the detailed event pages and apply as player (spectator) functionalities. Then we are planning to implement discussion functionality including comments and answers to the detailed event page. Until the final Milestone we are planning to at least implement these functionalities.

Backend status: We have completed the registration, login, forgot password, view profile, update profile, create event, search event, get event, get all events functionalities as of now. These work in compliance with ActivityStream as well. It was our intention to finish the comment and answer APIs and we had started work on them, but we could not finalize them before the milestone 2 presentation.
Backend moving forward: We will first finish the comment and answer APIs as they are work in progress and half-done anyway. We have also already started work on equipment posts right after the past milestone, so ideally that should be the next to be finished. Then, we have all the rest of the requirements left to finish. The first to work on should be the functionality of applying to events, accepting and rejecting users. Then the next target will be the implementation of the follow functionality and the notifications system.

Frontend status: We have completed the home page, registration, login, forgot password, create event with map component and comments/answers. We also implemented the view/update profile pages; however, due to a bug, they are not working correctly on the deployed server for now. Also, the event search page is implemented; however, there are some problems with the date inputs. The map component for rectangular area search is implemented but it is not integrated to the search page yet.
Frontend moving forward: We will first fix the issues in the profile page and the event search page and also integrate the rectangular area search component. We planned to implement the event application functionality for milestone 2; however, we did not implement it since not all teams could catch up. Our next goal is to implement the event application functionality. Then, we will work on equipment posts, follow/block functionalities and notifications.

Summary of Work Done by Each Team Member

Doğukan Akar	• Completed profile Page - Related issue. - pull request • Created Authorization process - Related issue. - Pull request. • Created Feedback Mechanism - Related issue. - Pull request. • Created event search page and added filters. - Related issue. - Pull Request • Created some search filters • Fixed some bugs on signup page - Related issue. - Pull Request • Examined a bug on signup page - Related issue. • Merged frontend and backend pages - Related issue. - Pull Request • Reviewed some pull requests
Şefika Akman	• [Frontend]Created view event page. Its issue number is #188. • [Frontend] Created minimum date, maximum date, minumum creation date and maximum creation date filters for seacrh event page. Its issue number is: #219. • [Frontend] I have implemented the not found page. When somebody enters a page that does not exists, this page opens. Isuue number is: #223 • Created pull request @246 for search date filters. • Created pul request #225 for not found page. • Wrote tests for date filters which are min date, max fate, min creation date and max creation date.

	• Reviewed other group members pull requests.
Ömer Arslan	• Changed signup page password notation for Android. issue • Created loading dialog for Android. issue • Profile page modifications and update profile are implemented for Android. issue • Create event fragment is implemented for Android. issue • Loading Lottie Pull Request • Update Profile Pull Request • Create Event Pull Request • Reviewed Pull Requests: View and Search Events, Search Result Page
İbrahim Kağan Bayat	• I have created comments app and comments branch to work together and clearly. commit pull request • I have contributed to comments and answers api documentation with get all answers of a comment (GET 'api/posts/<post_id>/comments/<comment_id>/answers') and create an answer to a comment (POST 'api/posts/<post_id>/comments/<comment_id>/answers') documentation • I have written get all comments of an event and post a comment to an event methods. GET 'api/posts/<post_id>/comments/' POST 'api/posts/<post_id>/comments/' commit pull request • I have also written models and serializers for comments and answers.commit pull request • I have tried to write some unit tests for urls and views. commit, pull request • To write test functions I watched some videos from Youtube.

	• I reviewed the event posts with activity stream. pull request • I reviewed the seeing events functionality of android part. pull request • Contributing to badge from wikidata
Berkay Döner	• [Frontend] Implemented the map functionality for creating events and improved the create event page. #214, PR: #233 • [Frontend] Implemented the map functionality for rectangular area filtering. #224, PR: #227 • [Frontend] Implemented the comment and answers page. #215, PR: #230 • [Frontend] Wrote unit tests for comments and answers. #238, PR: #229 • [Frontend] Wrote unit tests for create event page. #240, PR: #243 • [Frontend] Wrote unit tests for map functionalities. #239, PR: #245 • Contributed to writing the API documentation in Activity Stream format. • Configured the testing environment for frontend unit tests with Jest library and npm. • Set up GitHub Actions to automate the testing process for frontend unit tests. #250 • Wrote the frontend parts of executive summary, core functionality, user interface sections of the milestone report. Contributed to the status of requirements and standards sections.
Ahmet Necip Görgülü	• Implemented view all events functionality for Android. Issue #203, commit • Implemented Search Events Page for Android. Pull Request • Implemented Search Event Results

	Page for Android. Commit • Implemented the map functionality for Android. Commit • Wrote Implementation of Core Functionality Android part of the Milestone Report. • Presented the Android part of our presentation.
Ege Can Kaya	• Back-end: re-implementation of the creating and getting event functionality, using ActivityStreams • Back-end: improved registration API • Back-end: updated event model • Back-end: view and update profile functionality • Back-end: authentication related changes and checks put into place • Back-end: reviewing work on event pagination • Contributing to the Milestone report by adding the back-end parts of the executive summary and the API documentation
Bengisu Özaydın	• Wrote API documentation for comments & answers functionalities. Wiki Page. Issue #205 • Implemented answers API. Issue #206 • Reviewed PR #231 • Reviewed PR #237 • Wrote tests for event post functionality, did not end up committing those because more functional tests were written by that time.
Mehmet Saraçoğlu	• Implemented the detailed view event page. • Issues: #260 • Pull requests: #258 • Commits: Commit

Kerem Zaman	• Implementing pagination functionality for event posts • Updating the implementation of the pagination for new event post API supporting activity stream format • Adding search functionality to filter events by owner, event date, creation date, age, skill levels, location, coordinates, number of players, number of spectators, numbers of player and spectator capacities, sport type, duration, title and description • Adding unit tests for search functionality of event posts • Editing docker file to run unit tests before starting application automatically • Adding a script to dockerfile for backend service to wait before database service is ready • Fixing a critical bug in event post model causing created event to have wrong event date and search tests to fail • Configuring Github actions to run backend unit tests automatically for each push and pull request on development branch • Deploying latest versions of the app • Issues: #207, #221, #222, #232 • PRs: #220, #231, #237 • Reviewed PRs: #208, #218 • All commits can be found under issues. • Contributing to milestone group report

Progress According to Requirements

Addressed requirements are displayed in green, if they are fully implemented and in yellow if they are implemented partially. Some addressed requirements have explanations at the end.

Glossary

1. Functional Requirements

1.1 User Requirements

1.1.1. Sign Up

1.1.1.1. Users shall register to be a registered user.
1.1.1.2. Users shall specify their email, password and username while registering.
1.1.1.3. Users shall accept the terms and conditions document to register. (Android team implemented a checkbox to accept these but we do not have such documents yet.)
1.1.1.4. Users shall fill personal information form while registering.
1.1.1.5. Users should select their interests and skills to get suggestions from the system. (Frontend part is not deployed correctly due to a bug.)

1.1.2. Sign In

1.1.2.1. Registered users shall enter their email or username and password to sign in. (We have implemented a sign in functionality for username only, sign in through email should be implemented.)

1.1.3. Registered Users

1.1.3.1. Registered users shall have a personal profile page. (Frontend part is not displayed in the deployed served to the a bug.)
1.1.3.2. Registered users shall not be blocked by the user to view a user's profile page.
1.1.3.3. User should be registered in order to write comments under posts.

1.1.4. Search

1.1.4.1 Registered users shall use the search feature to find users or posts.

1.1.5. Post

1.1.5.1 Event Post

1.1.5.1.1 Registered users shall enter the values of the event to create an event post.

1.1.5.1.2 Event creators shall specify the type of sports events, manage location and type of sports events.

1.1.5.1.3 Event creators shall select the participants among the applicants.

1.1.5.1.4 Event creators shall report game results along with analytics.

1.1.5.1.5 Users shall indicate the type of sport and location in order to filter events.

1.1.5.1.6 Registered users shall specify their interests and skills to be able to get event recommendations.

1.1.5.2 Equipment Post

1.1.5.2.1 Registered users shall enter the values of the equipment to create an equipment post.

1.1.5.2.2 Registered users shall select the proper post and enter the contents of a comment to enter a comment.

1.1.5.2.3 Users shall indicate the type of sport and location in order to filter equipment.

1.1.5.2.4 Registered users shall specify their interests and skills to be able to get equipment recommendations.

1.1.6. Notification

1.1.6.1 Registered users shall select which notifications that they will see.

1.1.6.2 Registered users shall enable or disable all or a subset of notifications.

1.2 System Requirements

1.2.1 Sign Up

1.2.1.1. System shall ask for username, email and password for regular signup.

1.2.1.2. System shall send verification mail after regular signup.

1.2.1.3. System shall redirect users to profile creation page immediately after signup. (Frontend and Android pages are not the same, should be resolved.)

1.2.1.4. System shall not allow you to sign up with the same email more than once at the same time.

1.2.2 Sign In

1.2.2.1. System shall allow users to sign in with email/username.
1.2.2.2. System shall allow users to logout by clicking a button. (Frontend part is missing.)

1.2.3 User Types

1.2.3.1. Banned User
1.2.3.2. Regular User
1.2.3.3. Administrator

1.2.4 User Profile

1.2.4.1. System shall ask user their interests and skills while creating profile
1.2.4.2. Interests and skills shall be updated according to games registered to system by admins
1.2.4.3. User profile shall display personal information, skills, badges, followers, followings and post history.
1.2.4.4. System shall allow users to edit their profile page and password. (Edit functionality for profile page is implemented but has bugs also password cannot be changed for now.)
1.2.4.5. System shall allow users to delete their profiles.

1.2.5 Events

1.2.5.1. System should recommend sports events to the users according to their interests.
1.2.5.2. System shall send emails and push notifications when a registered event approaches.
1.2.5.3. System shall allow event creators to specify event type, needed number of players, location of event, event time, skills of participants, age range of participants.
1.2.5.4 Events with needed users registered shall be signed as full, so other people can filter them.
1.2.5.5 System shall allow guests to view the events list and individual event pages, without participating in them. (Searching is done without login, however event pages are not fully completed.)
1.2.5.6 All users shall be allowed to list all events on the events page.
1.2.5.7 Event pages shall display the event creator, event location, sports field, applicants, participants and spectators of the event. (Participants and spectators are not displayed yet.)
1.2.5.8 System shall allow all users to view the profile pages of the event creator, any of the applicants, participants and the spectators.
1.2.5.9 System shall allow only registered users to apply to events.

1.2.6 Searching

1.2.6.1 Event Search

1.2.6.1.1. Events shall be filtered by user according to event type, appropriate age range, location, date, hours, skill levels of attendees and availability . (availability is missing)
1.2.6.1.2. Events shall be sorted with respect to name, time and location (e.g. from closer events to far away ones).

1.2.6.2 User Search

1.2.6.2.1. Users shall be filtered according to their ages, skills and interested sports.
1.2.6.2.2. Users shall be sorted with respect to their names and ages.

1.2.7 Sport Equipment Post

1.2.7.1. System shall recommend some websites to the users for their equipment needs.
1.2.7.2. System shall allow filtering and sorting the equipment websites according to the name of the website and name of the sport.
1.2.7.3. System shall allow equipment posts to be edited by their posters.

1.2.8 Reporting

1.2.8.1. System shall allow users to report some problems to event creators in the event.
1.2.8.2. System shall allow users to report some bugs to administrators.
1.2.8.3. System shall allow users to report some users for misusage.

1.2.9 Discussions

1.2.9.1. System shall allow users to discuss under event or equipment post by writing comments. (Backend and Android is missing)
1.2.9.2. System shall also allow users to answer/reply discussion comments. (Backend and Android is missing)
1.2.9.3. Users shall not be able to reply to the answers/replies, thus replies shall be the deepest level of conversations. (Backend and Android is missing)

1.2.10 Notifications

1.2.10.1. System shall notify users when their player/spectator applications are approved/rejected.
1.2.10.2. System shall notify users for any related event happening around if they share their locations.
1.2.10.3. Users shall be notified if some people withdraw from an event after the event is signed as full. (Both registered users and users signed as want to attend but can't register because of capacity)
1.2.10.4. Users shall be notified if the event they applied/selected to be a player/spectator is delayed/cancelled.
1.2.10.5. Users shall be notified when a user follows them.
1.2.10.6. Users shall be notified when a user comments on the discussion page of an event they created.

1.2.11 Badges

1.2.11.1. There shall be badges that are given by the system after some milestones, that are given by the event creators at the end of the event and that are given by the users.
1.2.11.2. There should be badges that indicate bad behavior.
1.2.11.3 Only registered users shall be allowed to give badges to the creator of a post or to fellow players of an event.

1.2.12 User Interactions

1.2.12.1. System shall allow registered users to follow other users, so that they can be notified about the events created by the users they follow.
1.2.12.2. System shall allow users to block other users.

1.2.13 Analytics

1.2.13.1. System shall ask for and store analytics for players, event creators and sports fields after events.
1.2.13.2. System shall allow users to see the analytics for players, event creators and sports fields.

2. Non-Functional Requirements

2.1. Availability

2.1.1. System shall be available as a web application for modern browsers such as Chrome and Firefox.

2.1.2. System shall be available for Android based devices.
2.1.3. English language shall be supported.

2.2. Deployment

2.2.1. System should be containerized using Docker.

2.2.2. System should run on an Amazon EC2 server. (We decided to try Google Cloud and deployed our application there, we might change it to Amazon EC2 in the future.)

2.3. Performance

2.3.1. Response time for any request shall be below 5 seconds.

2.4. Privacy

2.4.1. All personal data shall be stored and processed according to KVKK and GDPR.
2.4.2. Users shall accept the privacy policy when signing up to the system.
2.4.3. Users shall be able to choose which of their personal data to be seen publicly.

2.5. Security

2.5.1. Passwords shall be at least 8 characters long and be a mixture of lowercase characters, uppercase characters and digits.
2.5.2. Passwords shall be encrypted using SHA-256 before storing.
2.5.3. Users shall verify their login information via email after 3 unsuccessful login attempts.
2.5.4. All inputs from users shall be checked to prevent SQL injections.
2.5.5. There shall be a password reset facility that sends a verification code via email or SMS.
2.5.6. System should only allow predetermined known file types to be uploaded. (e.g. PNG and JPG for images)
2.5.7. System should hide the passwords while typing on the password bar.

2.6. Standards

2.6.1. System shall follow the standards introduced by W3C Activity Stream Standard 2.0. (Not every API follows Activity Stream)

2.6.2. The semantic taggings should be supported with Wikidata.

API Endpoints

Link to the API: http://34.68.66.109/

Login (obtain JWT token)

POST 'api/token/obtain/'

- **Author:** Ege Can Kaya
- **Definition:** Takes a set of user credentials and returns an access and refresh JSON web token pair to prove the authentication of those credentials.
- **Example Response Value:**

Example Response Value:

{
"refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTYzODAzMzQ0OCwiaWF0IjoxNjM2ODIzODQ4LCJqdGkiOiJiNDkxZjIwZjU2OTY0YTJiYjI2OTlyNWJjMmZkNTc3YSIsInVzZXJfaWQiOjR9.tCGeBNzTXXBA8fTC2eqxyy2wfa1tYEN8nL0mSCzftP0",
"access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM2ODI0MTQ4LCJpYXQiOjE2MzY4MjM4NDgsImp0aSI6ImY0ZWIyNDY5ZWIwZDQyMDQ4NzI1NTAyNDQ5YjZlYTM2IiwidXNlcl9pZCI6NH0.HgN9BIi3F-1lnio_s32IHdsHNDUlAuNcWaE_s9A11vY"
}

Example Failure Value:

{
"detail": "No active account found with the given credentials"
}

- **Parameters**

Parameter	Description	Parameter Type	Data Type

username	Username	Body	String
password	Password	Body	String

- **Response Messages**
  - 200: OK.
  - 401: Unauthorized.

Login (refresh JWT token)

POST 'api/token/obtain/'

- **Author:** Ege Can Kaya
- **Definition:** Takes a refresh type JSON web token and returns an access type JSON web token if the refresh token is valid.
- **Example Response Value:**

Example Response Value:

{
"access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM2ODI0NDUxLCJpYXQiOjE2MzY4MjM4NDgsImp0aSI6IjY3ZTRlZDE2NzQyYTQyYTZiOGY2OTBiZTFmMTUzZjQ1IiwidXNlcl9pZCI6NH0.J9WhNq04miuDSABsobZtPHS5drcXxcq0eQE6MH44-zU",
"refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTYzODAzMzc1MSwiaWF0IjoxNjM2ODI0MTUxLCJqdGkiOiI0NjQ0MDFjNGY2ZjA0OTA3YjZlZTA3NDA0ZmQ0ZmI0MiIsInVzZXJfaWQiOjR9.r4q3jpcilwT6bqXupFqhSnu7pc58IzPNJXilnfwhwwM"
}

Example Failure Value:

{
"detail": "Token is invalid or expired",
"code": "token_not_valid"
}

- **Parameters**

Parameter	Description	Parameter Type	Data Type
refresh	Refresh type JWT token	Body	String

- **Response Messages**
    - 200: OK.
    - 401: Unauthorized.

Register

POST 'api/user/create/'

- **Author:** Ege Can Kaya
- **Definition:** Allows a user to register
- **Example Response Value:**

Example Response Value:

```
{
  "email": "egekaya@somemail.com",
  "username": "egekaya"
}
```

Example Failure Value:

```
{
  "username": [
    "This field is required."
  ],
  "password": [
    "This field is required."
  ]
}
```

- **Parameters**

Parameter	Description	Parameter Type	Data Type
username	Username of registering user	Body	String
email	Email address of registering user	Body	String
password	Password of registering user	Body	String

- **Response Messages**
    - 201: Created.
    - 401: Bad request.

Reset password (send mail)

POST 'api/password/reset/'

- **Author:** Ege Can Kaya
- **Definition:** Provides a method to request a password reset token based on an e-mail address
- **Example Response Value:**

Example Response Value:

```
{
  "status": "OK"
}
```

Example Failure Value:

```
{
  "email": [
```

"We couldn't find an account associated with that email. Please try a different e-mail address."
]
}

- **Parameters**

Parameter	Description	Parameter Type	Data Type
Email	Email address of user trying to reset their password	Body	String

- **Response Messages**
  - 200: OK.
  - 401: Bad request.

Reset password (confirmation)

POST 'api/password/reset/'

- **Author:** Ege Can Kaya
- **Definition:** Provides a method to reset a password based on a unique token
- **Example Response Value:**

Example Response Value:

```
{
  "status": "OK"
}
```

Example Failure Value:

```
{
  "detail": "Not found."
}
```

- **Parameters**

Parameter	Description	Parameter Type	Data Type
Password	New password for the user	Body	String
Token	Token received by email by the user allowing them to choose a new password	Body	String

- **Response Messages**
    - 200: OK.
    - 401: Bad request.

View profile

GET 'api/users/<username>/'

- **Author:** Ege Can Kaya
- **Definition:** Views the profile of a user
- **Example Response Value:**

Example Response Value:

```
{
   "first_name": "Ege",
   "last_name": "Kaya",
   "bio": "Hello, I love sports.",
   "fav_sport_1": "football",
   "fav_sport_2": "basketball",
   "fav_sport_3": "tennis",
   "location": "Istanbul"
}
```

Example Failure Value:


```
{
  "detail": "Not found."
}
```


- **Parameters**

Parameter	Description	Parameter Type	Data Type
username	Username of profile to view	Path	String

- **Response Messages**
  - 200: OK.
  - 404: Not Found.

Edit profile

PUT 'api/users/<username>/'

- **Author:** Ege Can Kaya
- **Definition:** Allows a user to edit their profile
- **Example Response Value:**

Example Response Value:


```
{
  "first_name": "Lewis",
  "last_name": "Hamilton",
  "bio": "7-time world champion",
  "fav_sport_1": "Formula 1",
  "fav_sport_2": "Formula 2",
  "fav_sport_3": "Formula E",
  "location": "Monaco",
```

"avatar": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg",
"privacy": false
}

Example Failure Value:

{
"detail": "Unauthorized."
}

- **Parameters**

Parameter	Description	Parameter Type	Data Type
username	Username of profile to edit	Path	String
first_name	New value for first name	Body	String
last_name	New value for last name	Body	String
bio	New value for bio	Body	String
fav_sport_1	New value for favorite sport 1	Body	String
fav_sport_2	New value for favorite sport 2	Body	String

fav_sport_3	New value for favorite sport 3	Body	String
location	New value for location	Body	String
avatar	New value for avatar	Body	String
privacy	New value for privacy	Body	Boolean

- **Response Messages**
  - 200: OK.
  - 401: Unauthorized.

Get event by ID

GET 'api/posts/<id>/'

- **Author:** Doğukan Akar
- **Definition:** Allows a user to fetch an event from database
- **Example Response Value:**

Example Response Value:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "Sally created an event",
  "type": "Create",
  "actor": {
    "type": "Person",
    "name": "Sally"
  },
  "object": {
    "type": "Event",
    "postId": "",
    "ownerId": "",
    "content": "",
```

```
        "title": "Beginner friendly tennis game",
        "creationDate": "2014-11-31T23:00:00-08:00",
        "lastUpdateDate": "2014-11-31T23:00:00-08:00",
        "numberOfClicks": 0,
        "location": {
            "name": "Etiler Tennis Club",
            "type": "Place",
            "longitude": 12.34,
            "latitude": 56.78,
            "altitude": 90,
            "units": "m"
        },
        "eventDate": "2014-12-31T23 00 00-08 00",
        "eventSport": "Tennis",
        "eventMinAge": 16,
        "eventMaxAge": 18,
        "eventMinSkillLevel": 1,
        "eventMaxSkillLevel": 5,
        "eventPlayerCapacity": 12,
        "eventSpectatorCapacity": 12,
        "eventApplicants": [1,2,3],
        "eventPlayers": [1,2,3]
    }
}
```


Example Failure Value:


```
{
    "detail": "Not found."
}
```


- **Parameters**

Parameter	Description	Parameter Type	Data Type
id	ID of the searched event	Path	Int

- **Response Messages**
    - 200: OK.
    - 404: Not Found.

Get all events

GET 'api/posts/'

- **Author:** Doğukan Akar
- **Definition:** Returns a list of post ids using filters.
- **Example Response Value:**

Example Response Value:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "Object history",
  "type": "OrderedCollection",
  "totalItems": 2,
  "totalPages": 2,
  "orderedItems": [
    {
      "type": "Create",
      "actor": {
        "type": "Person",
        "name": "Sally"
      },
      "object": "/api/posts/1"
    },
    {
      "type": "Create",
      "actor": {
        "type": "Person",
        "name": "Sally"
      },
      "object": "/api/posts/2"
    }
  ]
```

}

- **Parameters**

Parameter	**Description**	**Parameter Type**	**Data Type**
query	Query string	Query	String
creation_date	Creation date of the searched event	Body	DateTime
location	Location of the searched event	Body	String
date	Date of the searched event	Body	DateTime
duration	Duration of the searched event	Body	DateTime
sport	Sport type of the searched event	Body	DateTime
age_group	Which age group is preferred for the searched event	Body	Integer
player_capacity	Player capacity of the searched event	Body	Intege

spec_capacity	Spectator capacity of the searched event	Body	Integer
players	Players of the searched event	Body	Integer
spectators	Spectators of the searched event	Body	Integer
latitude	Latitude of the searched event	Body	Float
longitude	Longitude of the searched event	Body	Float
owner	Owner of the searched event	Body	User

- **Response Messages**
  - 200: OK.
  - 500: Internal server error.

Create an event

POST 'api/posts/'

- **Author:** Doğukan Akar
- **Definition:** Allows a user to create an event
- **Example Response Value:**

Example Response Value:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
```

```
  "summary": "Sally created an event",
  "type": "Create",
  "actor": {
    "type": "Person",
    "name": "Sally"
  },
  "object": {
    "type": "Event",
    "name": "A Simple Event",
    "postId": "",
    "ownerId": "",
    "content": "",
    "title": "Beginner friendly tennis game",
    "creationDate": "2014-11-31T23:00:00-08:00",
    "lastUpdateDate": "2014-11-31T23:00:00-08:00",
    "numberOfClicks": 0,
    "location": {
      "name": "Etiler Tennis Club",
      "type": "Place",
      "longitude": 12.34,
      "latitude": 56.78,
      "altitude": 90,
      "units": "m"
    },
    "eventDate": "2014-12-31T23 00 00-08 00",
    "eventSport": "Tennis",
    "eventMinAge": 16,
    "eventMaxAge": 18,
    "eventMinSkillLevel": 1,
    "eventMaxSkillLevel": 5,
    "eventPlayerCapacity": 12,
    "eventSpectatorCapacity": 12,
    "eventApplicants": [1,2,3],
    "eventPlayers": [1,2,3]
  }
}
```


- **Parameters**

Parameter	Description	Parameter Type	Data Type
query	Query string	Query	String
creation_date	Creation date of the searched event	Body	DateTime
location	Location of the created event	Body	String
date	Date of the created event	Body	DateTime
duration	Duration of the created event	Body	DateTime
sport	Sport type of the created event	Body	DateTime
age_group	Which age group is preferred for the created event	Body	Integer
player_capacity	Player capacity of the created event	Body	Intege
spec_capacity	Spectator capacity of the created event	Body	Integer

players	Players of the created event	Body	Integer
spectators	Spectators of the searched event	Body	Integer
latitude	Latitude of the created event	Body	Float
longitude	Longitude of the created event	Body	Float
owner	Owner of the created event	Body	User

- **Response Messages**
  - 201: Created.
  - 400: Bad request.

GET http://34.68.66.109/api/posts/1/

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "@context": "https://www.w3.org/ns/activitystreams",
    "summary": "AnonymousUser created an event",
    "type": "Create",
    "actor": {
        "type": "Person",
        "name": "AnonymousUser"
    },
    "object": {
        "type": "Event",
        "name": "A Simple Event",
        "postId": 1,
        "ownerId": 3,
        "content": "very good event",
        "title": "My Good event",
        "creationDate": "2021-12-13T17:54:02.248041Z",
        "numberOfClicks": 0,
        "location": {
            "name": "",
            "type": "Place",
            "longitude": 1.0,
            "latitude": 1.0,
            "units": "m"
        }
    },
    "eventDate": "2021-12-13T17:54:02.248067Z",
    "eventSport": "event event",
    "eventMinAge": 18,
    "eventMaxAge": 75,
    "eventMinSkillLevel": 0,
    "eventMaxSkillLevel": 0,
    "eventPlayerCapacity": 10,
    "eventSpectatorCapacity": 0,
    "eventApplicants": [],
```

GET http://34.68.66.109/api/users/cemilacar/

Params Authorization Headers (6) Body Pre-request Scrip

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "id": 7,
    "first_name": "cemil",
    "last_name": "acar",
    "bio": "",
    "fav_sport_1": "",
    "fav_sport_2": "",
    "fav_sport_3": "",
    "location": "Riva",
    "avatar": ""
}
```


Implementation of Core Functionality

Web

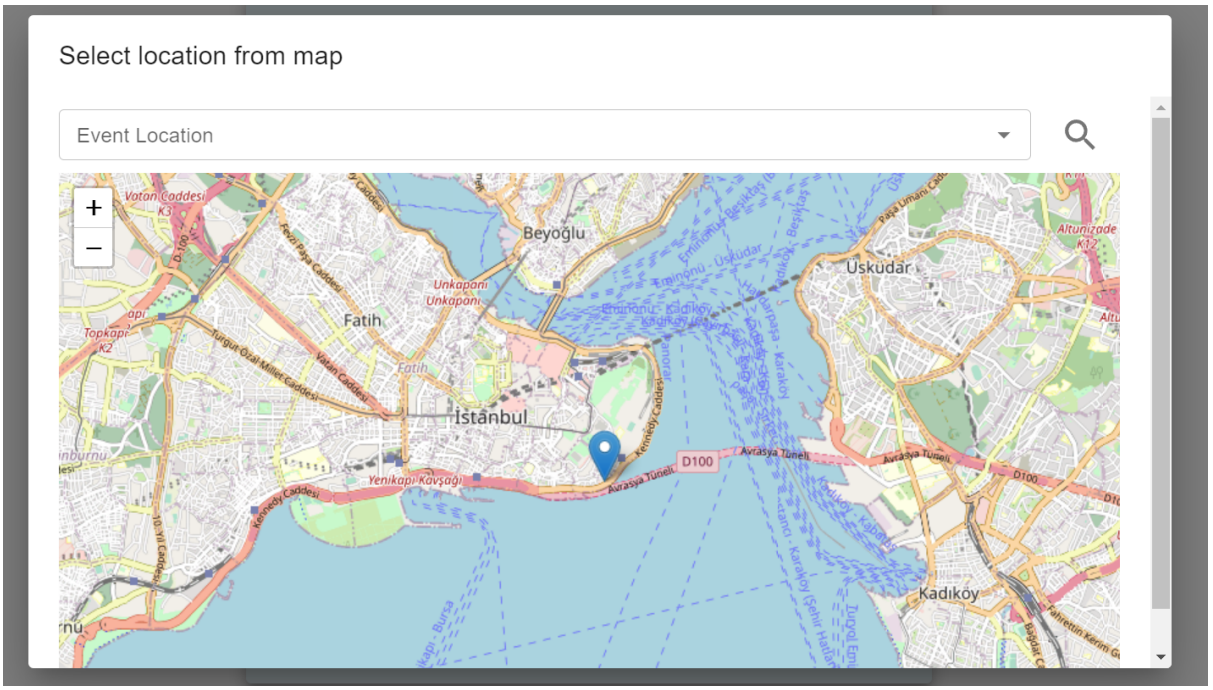
Status: We used maps to specify the event location in the create event page and to filter the events in the search page. Create event page is implemented with the necessary map and it is deployed. Although the map component for rectangular area search is implemented, the search page does not integrate it yet.

Detailed Work: For both maps, we used the leaflet and react-leaflet libraries, which support OpenStreetMap. We tried to use open source libraries in our application since our application itself is open source and we want to encourage collaboration.

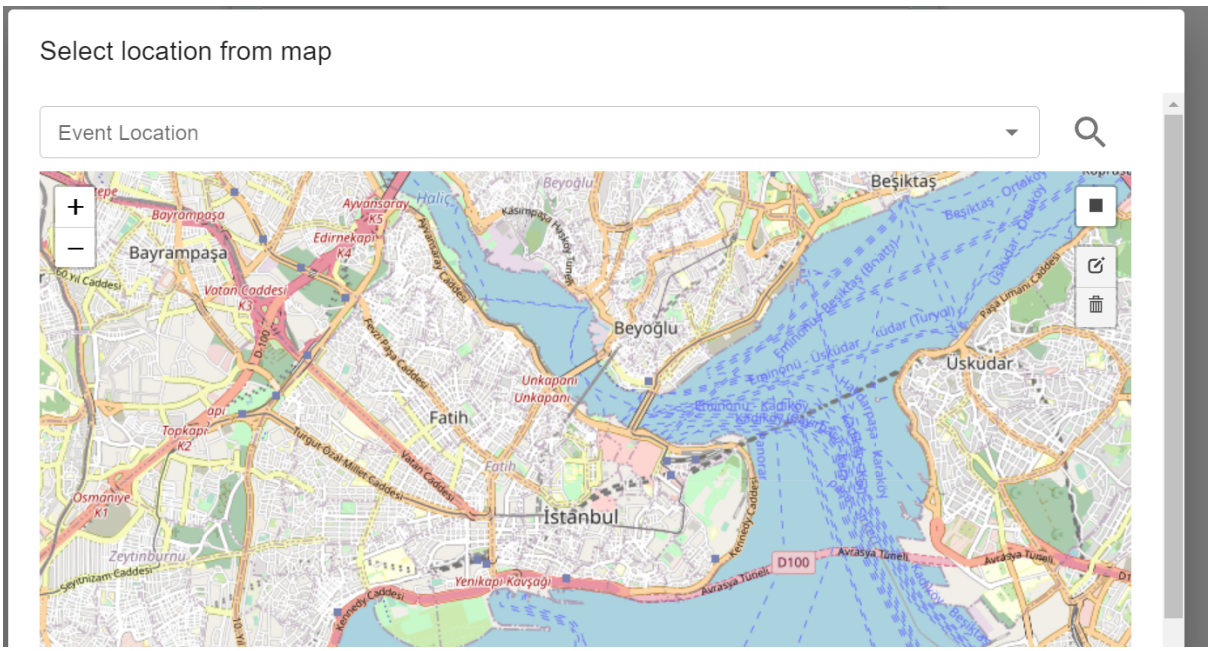
In the create event page, we implemented a draggable marker which users can drag to the precise location they want. Furthermore, we implemented a search bar on the top which the user can use to find general locations. When the user clicks the button after writing some text, options are fetched using the Nominatim API, which is also open source, and displayed. When the user selects one of these options, the map gets centered at the chosen location.

In the search page, we implemented rectangular area selection using the leaflet-draw and react-leaflet-draw libraries in addition to leaflet. Users can select the rectangle button on the right to start the drawing process. When a rectangle is drawn, the drawing button gets disabled since users are only allowed to specify one area.

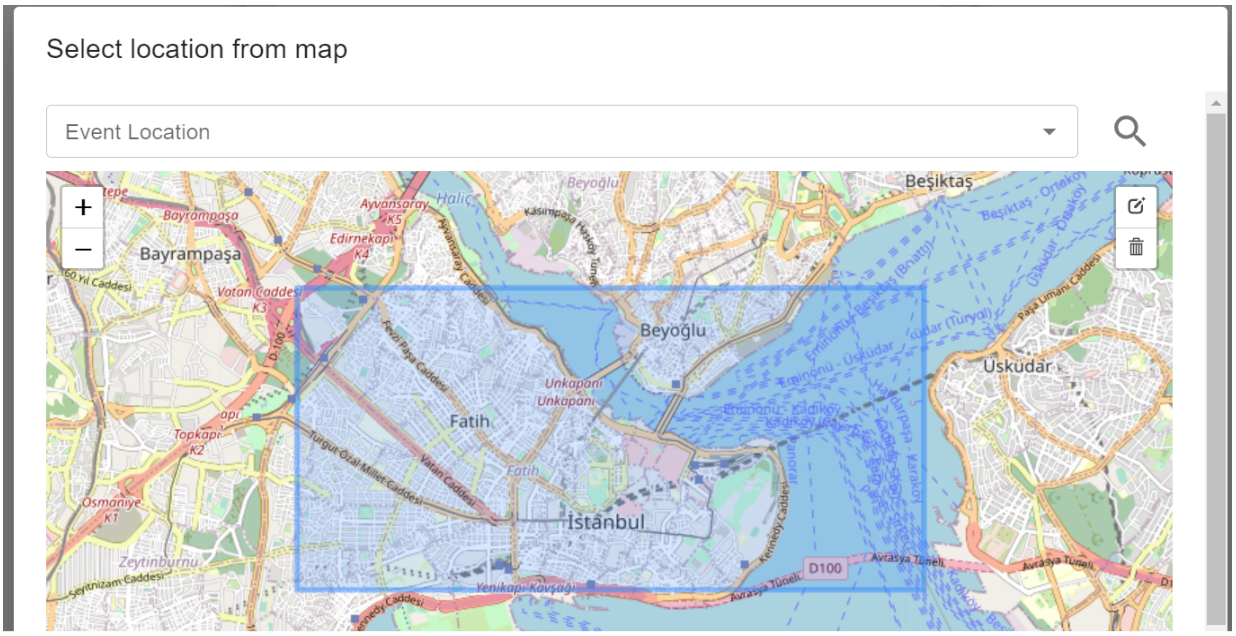
Future work: Search page must integrate the map with rectangular area selection to filter events by location. Also, there is a bug that causes the drawing button to stay disabled even though the drawn rectangle is deleted. It is related to the react-leaflet-draw library; however, we still have to find a solution to it.



Map in the create event page



Map in the search page, before choosing a rectangular area



Map in the search page, after choosing a rectangular area

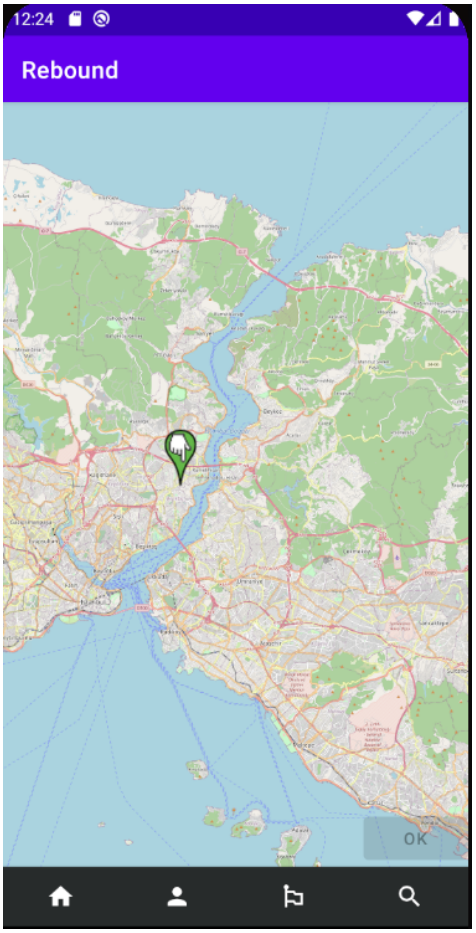
Android

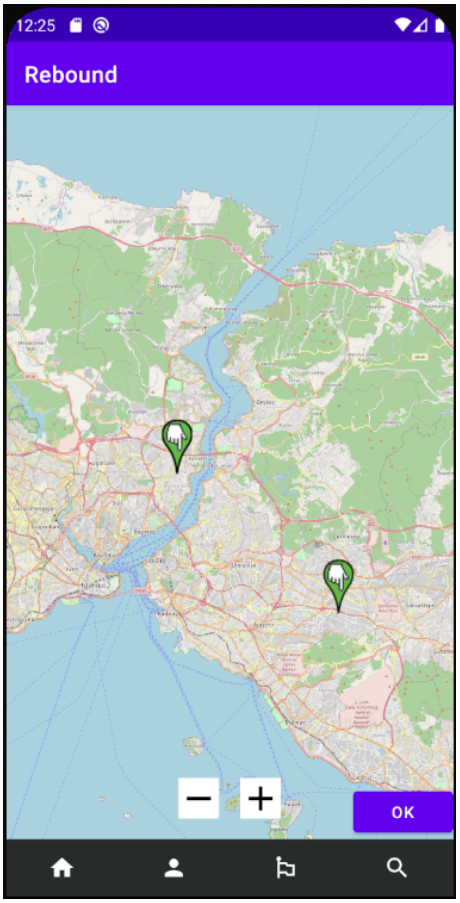
Status: Map functionality for create event and search event functions is completed and is working as intended.

Detailed Work: We've used osmdroid library in order to implement map functionality in our application. We have decided on osmdroid since we wanted to

use an open source map provider. We first created a map fragment so we can handle all things map related there. We have created the default world map based on [How to use the osmdroid library (Kotlin) · osmdroid/osmdroid Wiki (github.com)](). After that we implemented placing marker functionality and with a MapEventsReceiver obtained the coordinates where a marker is placed. We have implemented placing markers in a way that only two markers can be present on the screen maximum. On the map page after clicking “OK” button map fragment sends the latitude and longitude of these two markers to its parent fragment with the help of FragmentManager.setFragmentResult(). We set the map center to Istanbul and set the map zoom to 12 in order to make users’ life a little bit easier. We have used map fragment for both create event and search events functionalities.

Future Work: We need to demonstrate the rectangle that is going to be created by markers placed by users. After 2 markers are selected a rectangle is going to appear on the screen to let the user know which area they are selecting exactly.





User Interface/User Experience

Homepage and navigation bar/header

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Home

Android Code:
2021SpringGroup2/FragmentViewAllEvents.kt at android · bounswe/2021SpringGroup2 (github.com)

rebound

SEARCH | CREATE A NEW EVENT | HOME | SIGN UP

Discover Sports

Soccer

Sport that is played mostly outdoors, on grass, soccer (or football in Europe) opposes two teams. The objective is simple: send, by using mostly your feet, the ball to the opposing team's goal to score points. Ready to use a quick feint to shimmy past your opponent?

SEE SOCCER EVENTS

Basketball

Competitive and recreative sport opposing two teams, basketball is played on a court where two hoops are suspended in height at each extremity. With your team, your goal is to score points by throwing the ball into the basket. You will have to do some feints and dribble the ball to progress on the court and get as close as possible to the basket before throwing the ball. Can you jump high enough to make a dunker?

SEE BASKETBALL EVENTS

Volleyball

Volleyball is played for fun or during competition, inside or outside, with two teams on each side of a net on a delimited court. By only using your hands and your arms to send the ball on the other side of the net to score points, you'll make wonderful exchanges with your teammates. But for that, you will need to use your calm to receive the ball with precision and power to return it with great propulsion!

SEE VOLLEYBALL EVENTS

Browse Events

Beginner Friendly Tennis Match

Beginners are welcome

hatice_gun

Etiler Tennis Club

13/11/2021 15:00-16:00

Beginner Friendly Tennis Match

Beginners are welcome

hatice_gun

Etiler Tennis Club

13/11/2021 15:00-16:00

Beginner Friendly Tennis Match

Beginners are welcome

hatice_gun

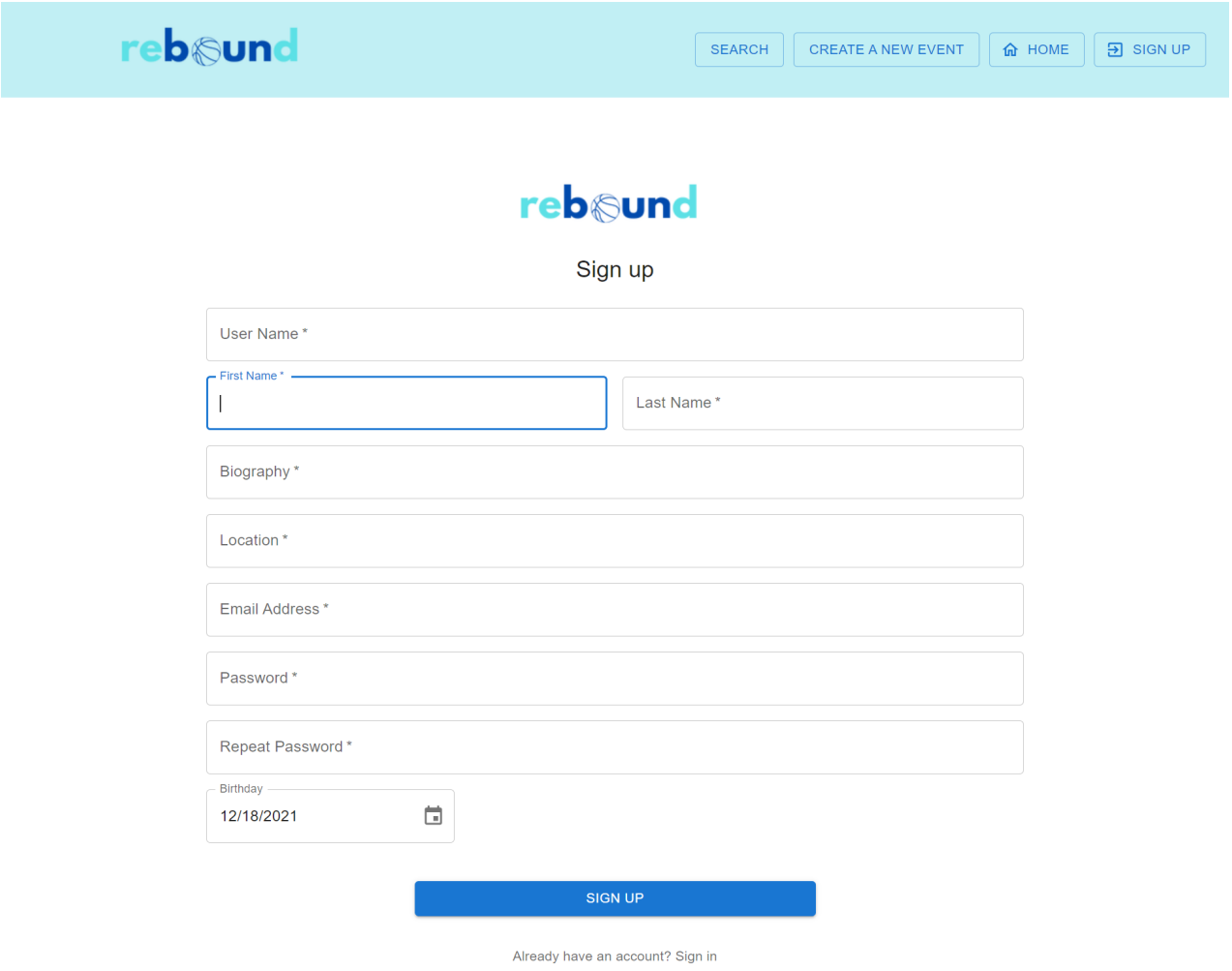
Etiler Tennis Club

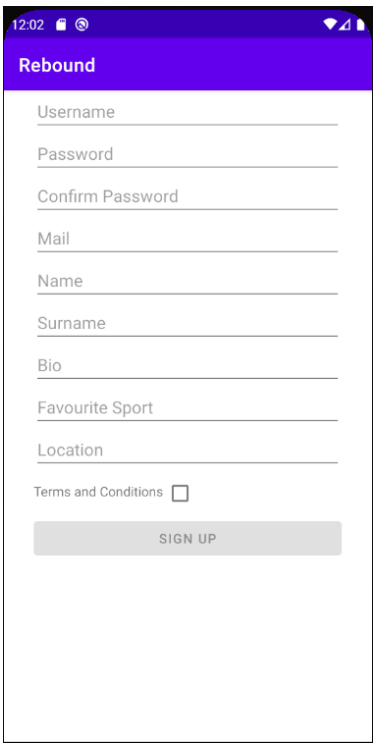
13/11/2021 15:00-16:00

Signup

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Signup

Android Code:
2021SpringGroup2/SignupActivity.kt at android · bounswe/2021SpringGroup2 (github.com)

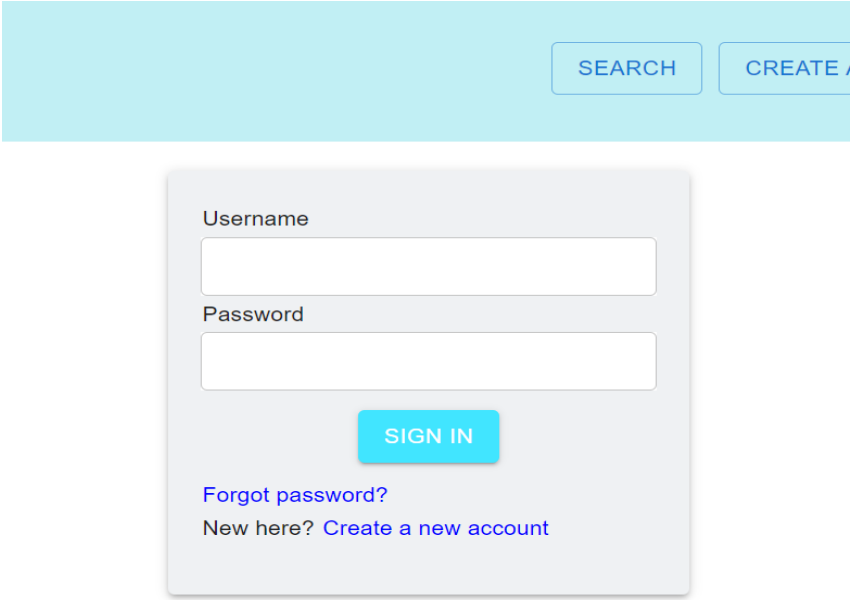


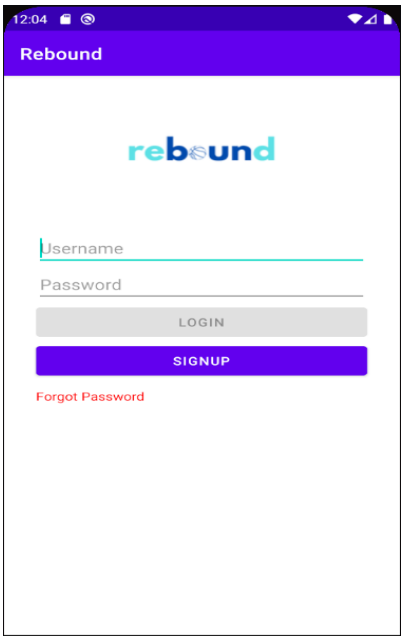


Login

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Login

Android Code:
2021SpringGroup2/LoginActivity.kt at android · bounswe/2021SpringGroup2 (github.com)

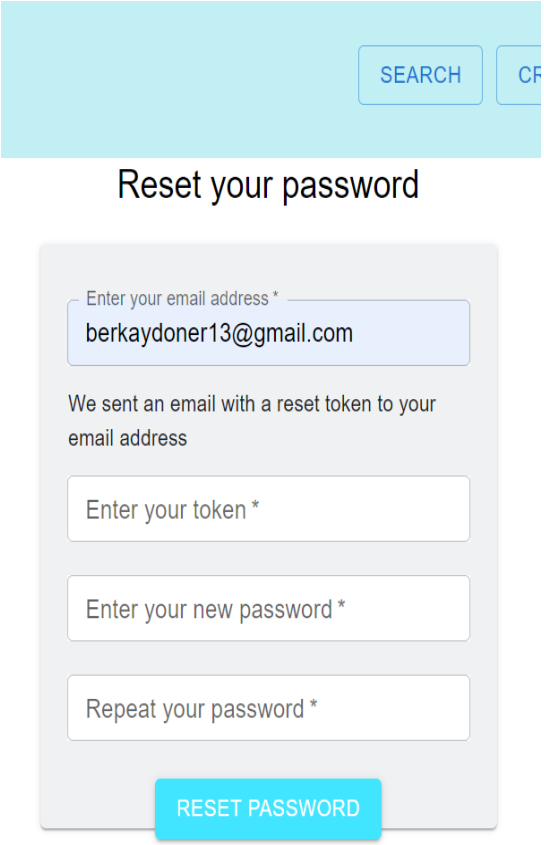

SEARCH
CREATE
Username
Password
SIGN IN
Forgot password?
New here? Create a new account

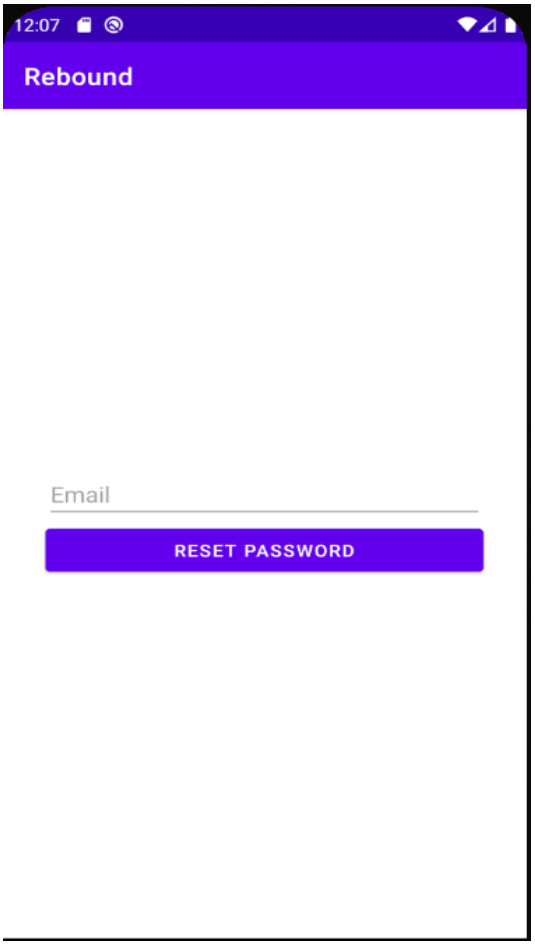

12:04
Rebound
rebound
Username
Password
LOGIN
SIGNUP
Forgot Password

Forgot password

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/blob/development/frontend/src/Views/Login/ForgotPassword.js

Android Code:
2021SpringGroup2/ForgotPasswordActivity.kt at android · bounswe/2021SpringGroup2 (github.com) - 2021SpringGroup2/ResetPasswordActivity.kt at android · bounswe/2021SpringGroup2 (github.com)

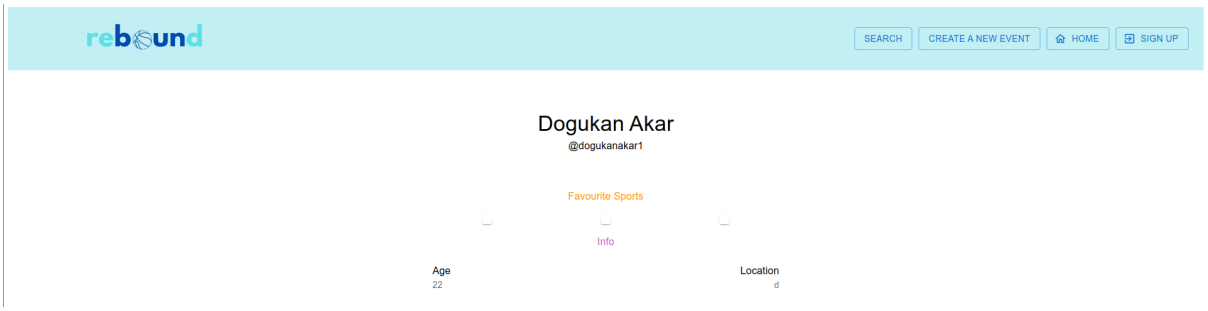


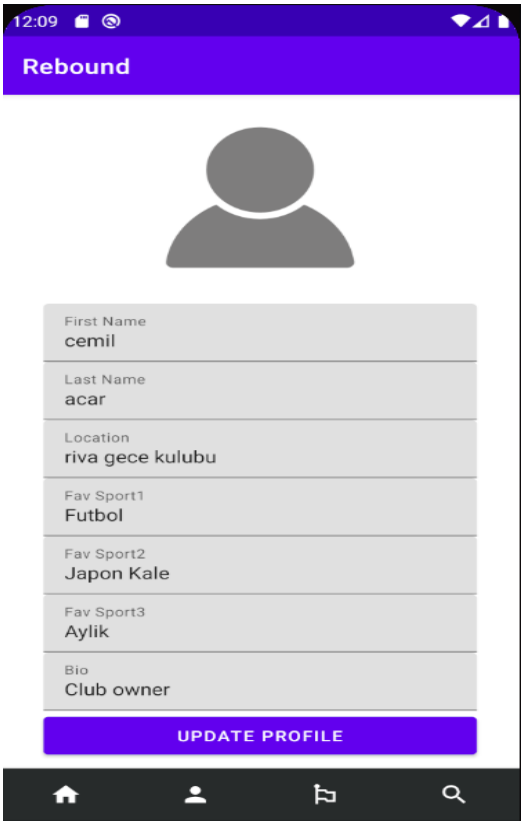


Profile Page

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Profile

Android Code:
2021SpringGroup2/FragmentProfile.kt at android · bounswe/2021SpringGroup2 (github.com)


rebound
SEARCH
CREATE A NEW EVENT
HOME
SIGN UP
Dogukan Akar
@dogukanakar1
Favourite Sports
Info
Age
22
Location
d

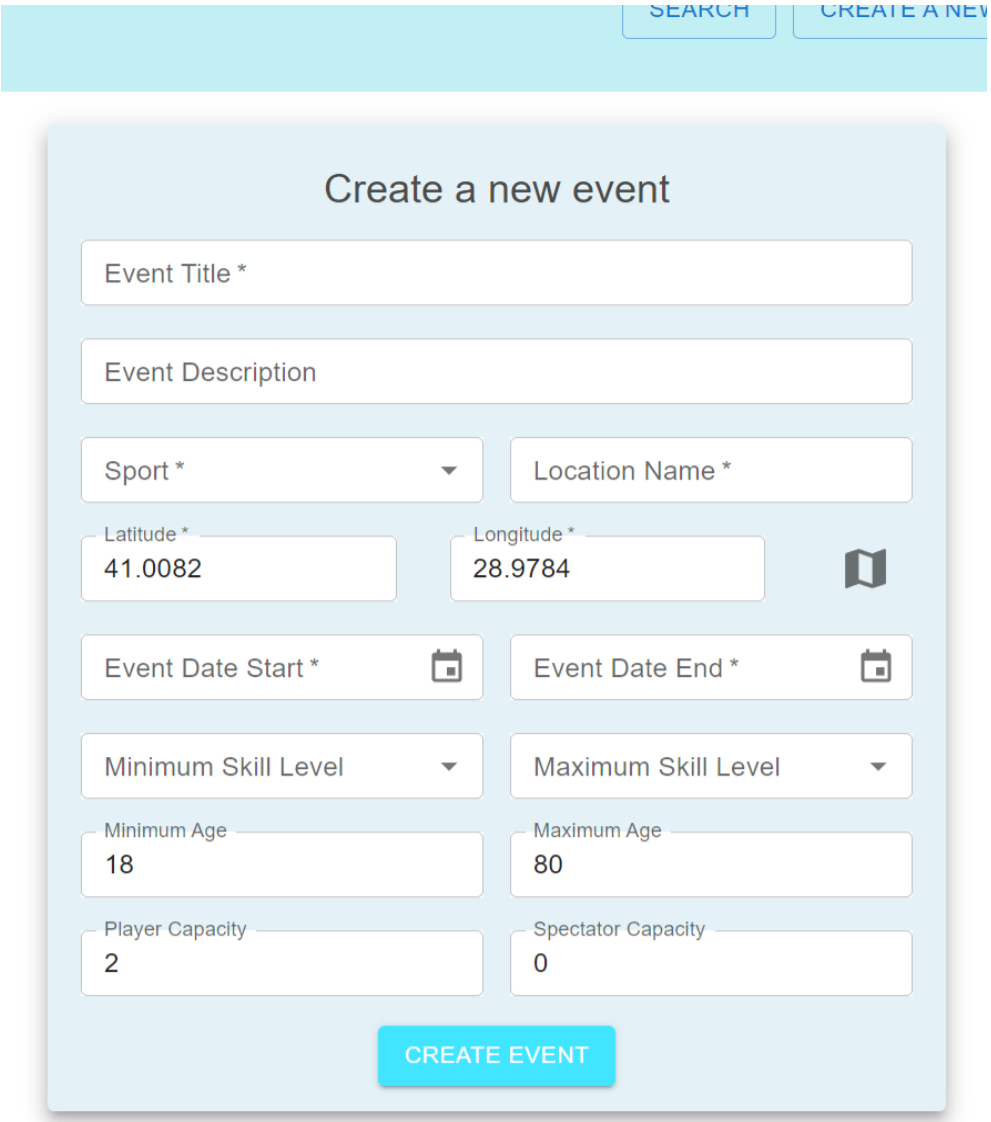

12:09
Rebound
First Name
cemil
Last Name
acar
Location
riva gece kulubu
Fav Sport1
Futbol
Fav Sport2
Japon Kale
Fav Sport3
Aylik
Bio
Club owner
UPDATE PROFILE

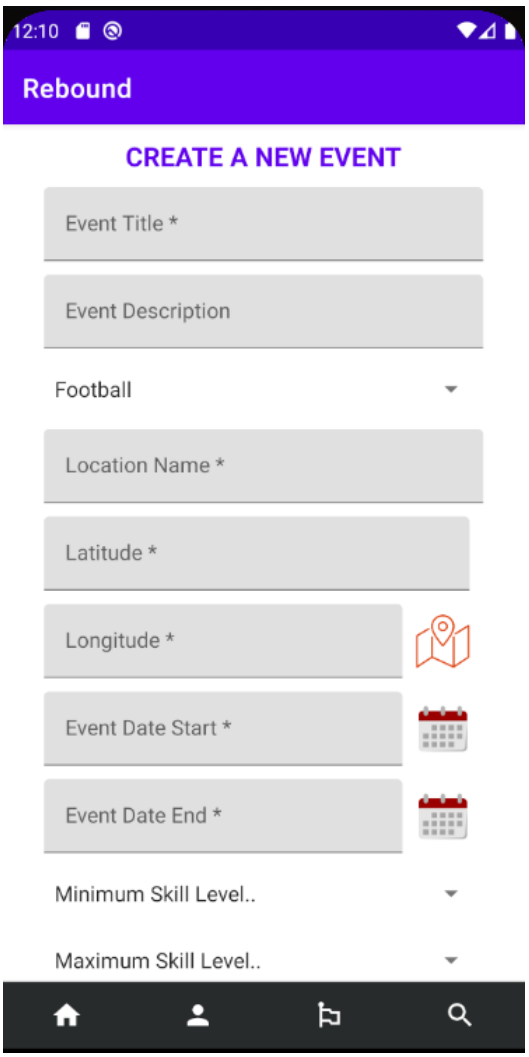
Create Event Page

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Create%20Event

Android Code:
[2021SpringGroup2/FragmentCreateEvent.kt at android · bounswe/2021SpringGroup2 (github.com)]

Maps are shown above in the mapping functionality section.





Search Events

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/SearchEvents

Android Code:
[2021SpringGroup2/FragmentSearchEvent.kt at android · bounswe/2021SpringGroup2 (github.com)]

rebound

SEARCH | CREATE A NEW EVENT | HOME | SIGN UP

Query

SEARCH

Filters

Minimum Skill Level

☆ ☆ ☆ ☆ ☆

Maximum Skill Level

★ ★ ★ ★ ★

Age Group Range

Players Range

Spectators Range

Player Capacity Range

Spectator Capacity Range

Minimum Creation Date

01/01/1970

Maximum Creation Date

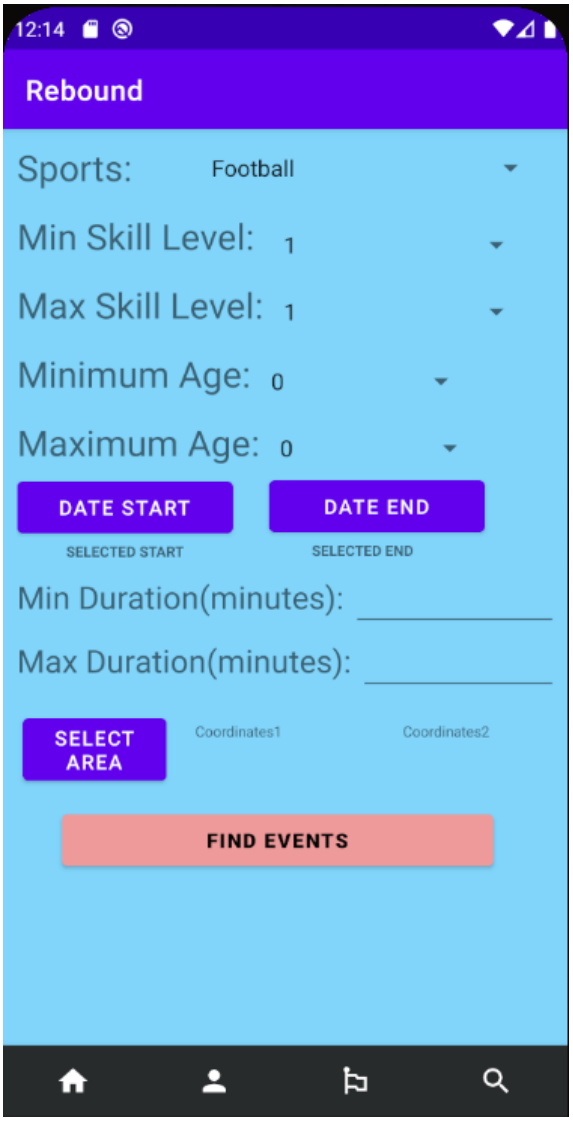
01/01/1970

Date Range (Start)

01/01/1970

Date Range (End)

01/01/1970


12:14
Rebound
Sports:
Football
Min Skill Level:
1
Max Skill Level:
1
Minimum Age:
0
Maximum Age:
0
DATE START
DATE END
SELECTED START
SELECTED END
Min Duration(minutes):
Max Duration(minutes):
SELECT AREA
Coordinates1
Coordinates2
FIND EVENTS

Comments & Answers Page

Frontend Code:
https://github.com/bounswe/2021SpringGroup2/tree/development/frontend/src/Views/Comments

Standards

W3C Activity Stream 2.0

We used the Activity Stream format for some JSON requests/responses between the frontend&android and backend parts.

As can be seen from Event API Documentation, responses related to events from the backend are in the Activity Stream. However, requests sent to the backend are not in the Activity Stream format yet. We are planning to convert all the requests sent from frontend and android for creating events/equipment posts to the Activity Stream format.

For comments and answers, we decided on the format, as can be seen from Comments&Answers API Documentation. Only the frontend part is implemented for this functionality, its controllers are implemented to get/send JSON objects in Activity Stream format but they might change according to backend.

Wikidata

We have not used Wikidata in our application yet; however, we are planning to use it for the badges in our application. These badges will be given to users to highlight their behaviour and skills. There will also be badges that represent bad behavior. These badges will be displayed on the profile pages, users can view them to decide on which applications to approve.

MVP - https://www.wikidata.org/wiki/Q652965
Fair play - https://www.wikidata.org/wiki/Q995616
Friendly - https://www.wikidata.org/wiki/Q491
Team leader - https://www.wikidata.org/wiki/Q743438
Cheater - https://www.wikidata.org/wiki/Q2488105
Agression - https://www.wikidata.org/wiki/Q191797
Bidone d'oro - https://www.wikidata.org/wiki/Q857020