CMPE 451 MILESTONE 2 GROUP REPORT

Prepared By: Gözde Ünver, Zuhal Didem Aytaç, Kemal Çağın Sertkaya, Umut Gün, Muhammed Furkan Atasoy
Group number: 5
Group name: LUDO
Sub-group:
Backend: Gözde Ünver, Mehmet Fatih Aksoy, Salih Bedirhan Eker
Frontend: Zuhal Didem Aytaç, Umut Gün, Muhammed Furkan Atasoy
Android: Kemal Çağın Sertkaya
Tag id: v.0.0.1

Executive summary

a) Project status

Backend has implemented create, change, delete, search event and equipment posts, login, register, apply to event, mail verification, access token generator functionalities.

Android has implemented the following pages: "Welcome Page, Login, Register, Forgot Password, Home Page, Search, Sort, Filter Functions, Create Event Page, Event Details Pop-up." Some of the pages are connected to the back-end. Others will be connected as soon as possible.

Frontend has implemented the following pages: Login, Register, Forgot Password, Search, Sort, Filter Functions, Profile Page, Create Event Page, Event Detail Page, Comment Feature. Some of the pages are connected to the back-end. Others will be connected as soon as possible.

b) The changes our team has planned/implemented to improve the development process

We assigned each person tasks for the milestone and we decided that 1 person from frontend, 1 from mobile and 1 from backend should communicate with each other regularly for the same tasks they work on to accelerate the development and decrease the bugs.

c) The improvements we have planned for the remainder of our project

In the backend, send badges, home page where the upcoming events of the users who the current user follows will be listed, event analytics will be implemented.
For the mobile part, we firstly planned to connect all pages to the back-end successfully. Later on,we'll create profile pages, comment functions, log-out functions, and equipment post related pages.

Summary of work

Gözde Ünver	- Updating and removing bugs in the create event and equipment posts, change event and equipment posts, delete equipment post functions after the milestone 1 - Implementing get event post and equipment post details functions - updating ouput jsons of create and change event and equipment posts in backend - changing location fields of the event and equipment post models - deploying backend to Amazon - adding "is_custom" field to backend so that a custom sport that is created by a user is not visible to other users - correcting created_date fields of event and equipment posts because django was storing UTC values instead of Europe/Istanbul - implementing the option of creating multiple events at a time in create event post functionality - returning users whose skill levels were inadequate to apply to an event in event post detail function - reviewing Mehmet Fatih's pull requests - reviewing Salih's pull requests - implementing code that sends post requests to Amazon to store the images and to store the url of the stored image in the postgresql db and then returning this url to frontend - adding CORS library to backend to resolve the CORS error that frontend was receiving - searching for a solution to resolve the https error we receive from backend Amazon instance while sending requests from deployed frontend - sending requests to deployed backend so that there can be some created posts to show them in the presentation - writing down the description of get event post details function, get equipment post details function, delete equipment post function, change event post information, change equipment post information, create event post function, create equipment post function, save sports to database function, save skill levels to database function and save badges to database function in the API documentation for this project on a wiki page that I created - Creating initial documents to write the milestone 2 group report - writing down how I used acitivty stream in the project in the group report - writing down the changes our team has planned/implemented to improve the development process in the group report - writing down the improvements we have planned for the remainder of our project for the backend - Writing down my weekly efforts - attending weekly lab sessions - attending customer 2 meeting

Zuhal Didem Aytaç	Attending weekly LAB Sessions Attending Frontend Meetings & Group Meetings Attending Customer Milestone 2 Meeting Taking meeting notes of frontend meeting 4 and group meeting 26 Implementing profile page Implementing user service with mocks that provides user data to be shown on profile page Implementing Terms and Conditions on register page Implementing logical input checks on login-register and forgot password pages Implementing error messages on register page Implementing Authorization and Authentication on frontend (token handling) Implementing auth context and authenticated routes Researching CORS error & communicating with the backend team Connecting frontend app to backend Fixing frontend bugs on register page, search page and profile page Implementing necessary navigations on frontend (register page and login page) Implementing routing in frontend Providing tests for register, forgot password and login pages Implementing helper functions for input checks Providing unit tests for helper functions Documenting personal weekly efforts Documenting personal efforts for the tabular work summary section of Milestone Report Documenting progress according to requirements for Milestone Report (my part only) Documenting user interface / user experience for Milestone Report (my part only)

Mehmet Fatih Aksoy	-Updating and removing bugs in the register, login and mail verification functions after milestone 1 -Attending weekly LAB meetings -Attending urgent meetings in discord -Studying about Amazon S3 Bucket and reading documents about it -Learning Amazon API Gateway and Amazon Lambda Services -Changing return response of login function -Creating mail verification message template and adding it to project -Creating Amazon S3 Bucket Server -Creating Amazon API to perform delete, create and update event,equipment post images in S3 Bucket. -Adding jwt-simple authentication app to main project -Creating backend in Amazon Lambda to take data from Amazon API and make neccessary works on S3 Bucket -Reviewing Salih's Pull requests -Reviewing Gözde's Pull Requests -Informing my team for S3 Bucket and sharing get/post/delete input/output format and URL of that API -Creating a function that generates access token for each user when logged in -Creating a function that takes an access token and return user bound to that token -Changing field of sportskill class -Writing my personal efforts on here -Writing my individual milestone #2 report -Attending customer 2 meeting -Writing API endpoints

Kemal Çağın Sertkaya	Attending weekly LAB Sessions. Attending Android Meetings. Attending Customer Milestone 2 Meeting. Taking meeting notes of group meeting #25. Creating Google Developer account and search for Maps API. Implementing main event screen and event detail pop-up. Implementing A-Z and by date sorting functions. Implementing Filter Page. Implementing Search Functionality. Implementing Create Event Page. Implementing Google Maps API with mark a point and draw circle functionality. Reviewing Berkay's Pull Request. Updating personal weekly efforts page. Documenting personal efforts for Group Milestone Report. Adding Android application's user interfaces and GitHub links to the Group Milestone Report. Presenting Ludo Android Application on Customer Demo 2.

Umut Gün	- Attending weekly LAB Sessions - Attending Frontend Meetings & Group Meetings - Attending Customer Milestone 2 Meeting - Implementing Creating Event Post Page - Implementing Form Control Validity in Creation Event Post Page - Implementing Other Category Option - Implementing Event Detail Page - Implementing Comment Feature as Design - Implementing Routing in Frontend - Fixing Frontend Bugs with Creation Event and Event Detail Page - Changing Some Minor Texts - Changing Backend IP in Frontend - Review pull requests - Preparing Customer Meeting 2 - Presentation Scenario for Customer Meeting 2 - Watch and Feedback Other Groups - Documenting Personnel Weekly Efforts - Documenting Personal Work Summary for Group Milestone 2 - Documenting Executive Summary for Group Milestone 2 - Documenting User Interface and User Experience for Group Milestone 2

Berkay Bör	Attending weekly LAB Sessions Attending Android sub-group meetings Attending Customer Milestone 2 Meeting Taking meeting notes of Android meeting 2 Implementing sign up flow Implementing login flow Implementing user create Implementing connecting to backend Reviewing Kemal's pull request Reviewing Kemal's work on maps Contributing to group milestone report Updating personal weekly efforts page Documenting personal efforts for Group Milestone Report

Salih Bedirhan Eker	- Backend geolocation implementation research - Decide JSON format of Search, Filter, and Sort page - Implement Application functionality - Implement Search functionalities backend - Write test cases for Application for event functionality - Implementation of getApplicant functions - Implementation of Spectate functionality - Write tests for deleting event functionality - Write tests for spectate functionality - Write tests for search event/equipment functionalities - Implement accept application functionality - Remove login requirement - Set server security configurations - Enter example data to deployed database - Add example api calls - Write api documentation - Write weekly efforts

Muhammed Furkan Atasoy	● Attend Weekly Lab sessions
	● Attend Frontend meetings
	● Attend Customer Milestone Meeting
	● Document personal summary of work for milestone 2
	● Contribute to milestone 2 group report
	  ○ Progress according to requirements
	  ○ Frontend Mapping Functionality
	  ○ User Interface/User Experience
	● Write individual milestone 2 report
	● Create a mockup path for presentation in search page
	● Watch and review other groups presentations
	● Deploy Frontend to Amazon server
	● Connect search page to backend

	• Writing Test for Search Page • Review pull request: implementation of the creation event page #398 • Review pull request: Event detail page #407 • Review pull request: Comment feature #408 • Review pull request: fixed some bugs and changed backend ip #440 • Review pull request: feature/terms-and-conditions #397 • Review pull request: Frontend-Backend Connection #419 • Review pull request: Profile Page #420 • Implement Search Page 2 • Implement Map component in React • Decide JSON format of Search, Filter, and Sort page • Search Map Component

Progress according to requirements (dönem başından beri yapılanlar)
Completed:

- **Implemented:**
  - Backend
    - Get event post details function
    - Get equipment post details function
    - Create event and equipment post
    - Change event and equipment post
    - Delete event and equipment post
    - Apply to event
    - Spectate to event
    - Accept application
    - Search event post
    - Search equipment post
    - Login
    - Register
    - Mail verification
    - Access token generator
  - Frontend
    - register page
    - forgot password page

        - Login page
        - Profile Page
        - Create Event Page
        - Event Detail Page
        - Comment
        - Search and Filter Page
        - Map Component
        - Header and Sidebar
    - Android
        - Welcome page
        - Login page
        - Register Page
        - Forgot Password Page
        - Filter Page
        - Search Function
        - Sort Function
        - Map Function
        - Create Event Page
        - Event details pop-up

- **Tested:**
    - Backend
        - Get event post details function
        - Get equipment post details function
        - Create event and equipment post
        - Change event and equipment post
        - Delete event and equipment post
        - Apply to event
        - Spectate to event
        - Accept application
        - Search event post
        - Search equipment post
        - Login
        - Register
        - Mail verification

    - Frontend
        - register page
        - forgot password page
        - Login page
        - Search and Filter Page
- **Documented:**
    - Backend
        - Get event post details function
        - Get equipment post details function
        - Create event and equipment post
        - Change event and equipment post
        - Delete event and equipment post
        - Apply to event
        - Spectate to event
        - Accept application
        - Search event post
        - Search equipment post
- **Deployed:**
    - Backend
        - Get event post details function
        - Get equipment post details function
        - Create event and equipment post
        - Change event and equipment post
        - Delete event and equipment post
        - Apply to event
        - Spectate to event
        - Accept application
        - Search event post
        - Search equipment post
        - Login
        - Register
        - Mail Verification
    - Frontend
        - register page
        - forgot password page

        - Login page
        - Profile Page
        - Search and Filter page
        - Map component
        - Header and Sidebar
    - Android
        - Login Page
        - Register Page
        - Search Function
        - Map Function

In progress:

- Android (pages are ready, now connecting to backend)
    - Create event page
    - Filter page

- Frontend (pages are ready, now connecting to backend)
    - Create Event Page
    - Event Detail Page
    - Comment

Not started:

- Backend
    - Home page
    - Send badge
    - Write comment to event and equipment post
    - Postpone event
    - Reject application
    - Write comment
    - Event analytics

    - Frontend
        - Equipment Post Page (Create-Edit)
        - Homepage
        - Equipment Search and Filter page

- Android
    - Log out
    - Profile Page
    - Equipment post page
    - Send badge
    - Comment function

API Documentation

Register Function

This function takes username, name, surname, mail, password, interests and level of that interests and creates a user with these given credentials. Register function also sends mails to given mail address to mail verification. This function controls if given 2 passwords are equal to each other, given username length is smaller than 8 or not and given username, mail address already exist or not. Backend endpoint: http://3.122.41.188:8000/register

Login Function

This function takes username and password and find a user with these given credentials. If user's mail is not verificated, login functionreturns a error response. If user is not found, also there is a response for that situtation. After checks, functions logs user in, and creates token for user. This function returns token generated for user. Backend endpoint: http://3.122.41.188:8000/login

Logout function

This function simply logs out user from system. Backend endpoint: http://3.122.41.188:8000/logout_user

Activate Function

This function is triggered when registered user clicks link on the verification mail. It takes generated token and uid, tries to find a user with that token, when find it changes the field is_mail_verificated to True. Backend endpoint: http://3.122.41.188:8000/activate-user//

Create event post function

This function creates an event post using these information: post name, owner of the post, sport name that this post refers to, description of the post, longitude and latitude of the post that this event will take place, the day and time that this event will take place, player participation limit number of the event, spectator limit number, event rules which the creator of the event may want to specify, event's equipment requirements which the creator of the event may want to specify, location requirement of the event if the creator of the event may want to specify such as only people from Istanbul should join etc., contact information of the creator of the event if he/she wants to specify such as telefon number or email etc., image of the event post, skill requirement of the event so that only people who satisfy the skill requirement can join, badges of the event that event creator or the applicants of the event would send to each other. When an event is created, its created day and time are automatically recorded, the capacity is initially "open to applications", its status is upcoming, current number of players that have been accepted by the event creator is 0, current spectator is 0. Users can either choose one of the sports names stored in the system or choose the "other" option and specify their unique sport name for the event.

Backend endpoint: http://3.122.41.188:8000/post/create_event_post/

Create equipment post function

This function creates equipment posts using this information: Name of the equipment post, creator of the post, sport name related to this post, description of the post in which the user could explain further his/her intention in this post, longitude and latitude of the location where the user want to specify an exact location of a shop, link of the store that the user want to put on the post, information about the post being active, image of the equipment post. Created day and time are automatically recorded, the status of the post is initially active=True.

Backend endpoint: http://3.122.41.188:8000/post/create_equipment_post/

Get event post details function

This function returns all information of an event post. These information are post name, owner of the post, sport name that this post refers to, created day and time of the post, description of the post, longitude and latitude of the post that this event will take place, the day and time that this event will take place, player participation limit number of the event, spectator limit number, event rules which the creator of the event may want to specify, event's equipment requirements which the creator of the event may want to specify, status of the event such as upcoming/deleted/passed, capacity of the event such as open to applications/full, location requirement of the event if the creator of the event may want to specify such as only people from Istanbul should join etc., contact information of the creator of the event if he/she wants to specify such as telefon number or email etc., url of the image of the event post, skill requirement of the event so that only people who satisfy the skill requirement can join, current number of players that have been accepted by the event creator, badges of the event that event creator or the applicants of the event would send to each other, current number of spectators that have joined the event, comments written for that posts, accepted user information of the users that are players of the event posts, rejected user information of the users that wanted to be players of the post, spectator user information of the users that will spectate to the event, waiting user information of the users that are waiting to be accepted and user information of the users that had inadequate skill level to join the event.

Backend endpoint: http://3.122.41.188:8000/post/get_event_post_details/

Get equipment post details function

This function returns everything about an equipment post. Name of the equipment post, information of the user who created the post, sport name related to this post, created day and time of the post, description of the post in which the user could explain further his/her intention in this post, longitude and latitude of the location where the user want to specify an exact location of a shop, link of the store that the user want to put on the post, information about the post being active, url of the image that is present on the equipment post, comments written for the post.

Backend endpoint: http://3.122.41.188:8000/post/get_equipment_post_details/

Delete equipment post function

This function deletes the equipment post. It is actual not removing the function from the database, instead its "active" status field becomes passive. In the frontend it is not shown anymore when equipment posts are searched.

Backend endpoint: http://3.122.41.188:8000/post/delete_equipment_post/

Delete event post function

This function deletes an event post. It updates the event post but the data still remains in the database. Function make 'false' 'active' field. While searching this event post is not shown. Also as return values function returns a message 'Event post is deleted'.

Backend endpoint: http://3.122.41.188:8000/delete_event_post/

Change event post information function

This function changes values of the fields that user modified on the event post. These fields can be badges of the event, name of the post, sport name related to the post, description of the post, longitude and latitude of the post, day and time that the event will take place, participation limit, spectator limit, rules of the event, equipment requirements of the event, location requirement of the event, contact information of the creator of the event, skill requirement of the event.

Backend endpoint: http://3.122.41.188:8000/post/chane_event_post/

Change equipment post information function

This function changes values of the fields that user modified on the equipment post. These fields can be name of the post, sport name that is related to this equipment post, description of the post, longitude and latitude of the post, link of the post.

Backend endpoint: http://3.122.41.188:8000/post/change_equipment_post/

Apply to event

This function basically creates an entry in 'Application' table if necessarry conditions are met. Function checks if event post is available to apply. After that function creates an entry and its field 'status' is 'waiting' or 'inadequate' (if applicants skill level is not adequate then status will be inadequate) and 'applicant_type' 'player'. In return it responses with a success message.

Backend endpoint: http://3.122.41.188:8000/post/apply_to_event/

Spectate to event

This function basically creates an entry in 'Application' table if necessarry conditions are met. Function checks if event post is available to spectate. After that function creates an entry and its field 'status' is 'accepted' (note that it does not become waiting) and 'applicant_type' 'spectator'. In return it responses with a success message.

Backend endpoint: http://3.122.41.188:8000/post/spectate_to_event/

Accept applications

This function just updates an application entry. If necessarry conditions are met, field 'status' become 'accepted' (from 'waiting' or 'inadequate'). In return it responses with a success message.

Backend endpoint: http://3.122.41.188:8000/post/accept_application/

Search event post

This function allows to search for an equipment post. You can provide a search keyword, a sort function results are sorted by location or by created date (default is date), and a filter function. All filter shoices are optional. In filter function you can filter posts by location and a radius, by sport type, by date interval. Also if you do not provide a search keyword function brings you all posts. The function retuns a sorted list contains events.

Backend endpoint: http://3.122.41.188:8000/search/search_event/

Search equipment post

This function allows to search for an event. You can provide a search keyword, a sort function results are sorted by location or by date (default is date),

and a filter function. All filter shoices are optional. In filter function you can filter posts by location and a radius, by sport type, by date interval. Also if you do not provide a search keyword function brings you all posts. The function retuns a sorted list contains events.

Backend endpoint: http://3.122.41.188:8000/search/search_equipment/

Save sports to the database

This function saves sport names retrieved from Decathlon API to the database. This function is called for once by the admin.

Backend endpoint: http://3.122.41.188:8000/post/save_sports/

Save skill levels to the database

This function saves skill levels to the database. Skill levels are hardcoded in the code. This function is called once by the admin

Backend endpoint: http://3.122.41.188:8000/post/save_skill_levels/

Save badges to the database

This function saves badges to the database. Badges are hardcoded in the code but they are not taken from wiki data yet. This function is called once by the admin
Backend endpoint: http://3.122.41.188:8000/post/save_badges/

Link to API

https://github.com/bounswe/2021SpringGroup5/wiki/LUDO-Project-API-Documentation

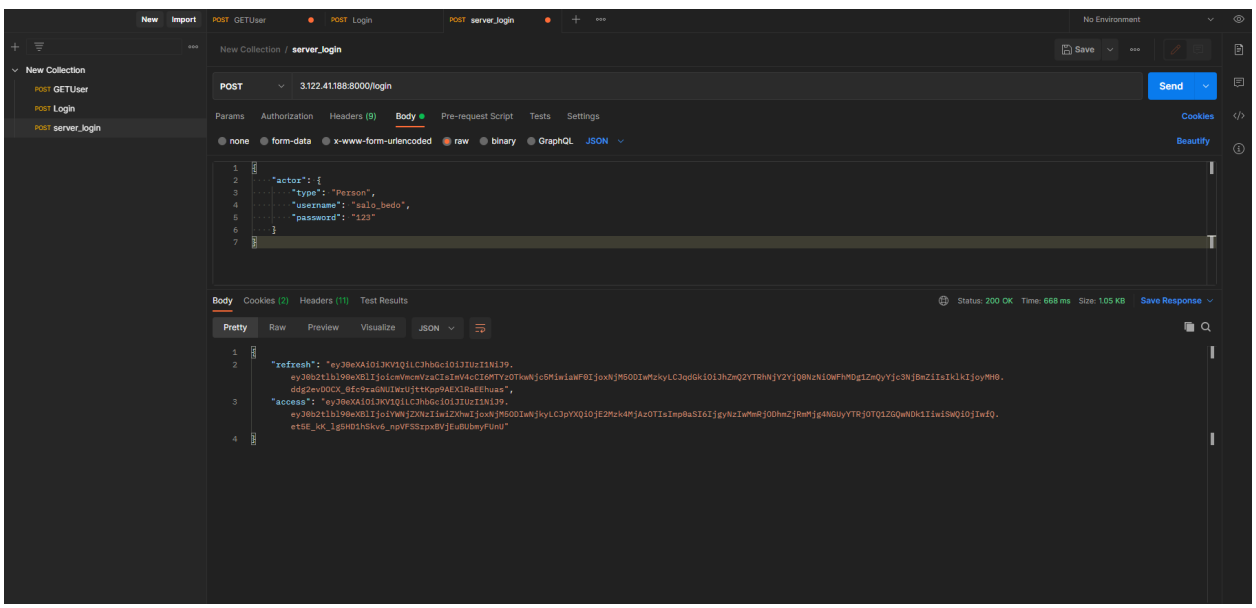
Three examples of API calls corresponding to core functionality of our application (All instructions to access the functionality must be provided. (You may share Postman collections))

Three core functionalities are:

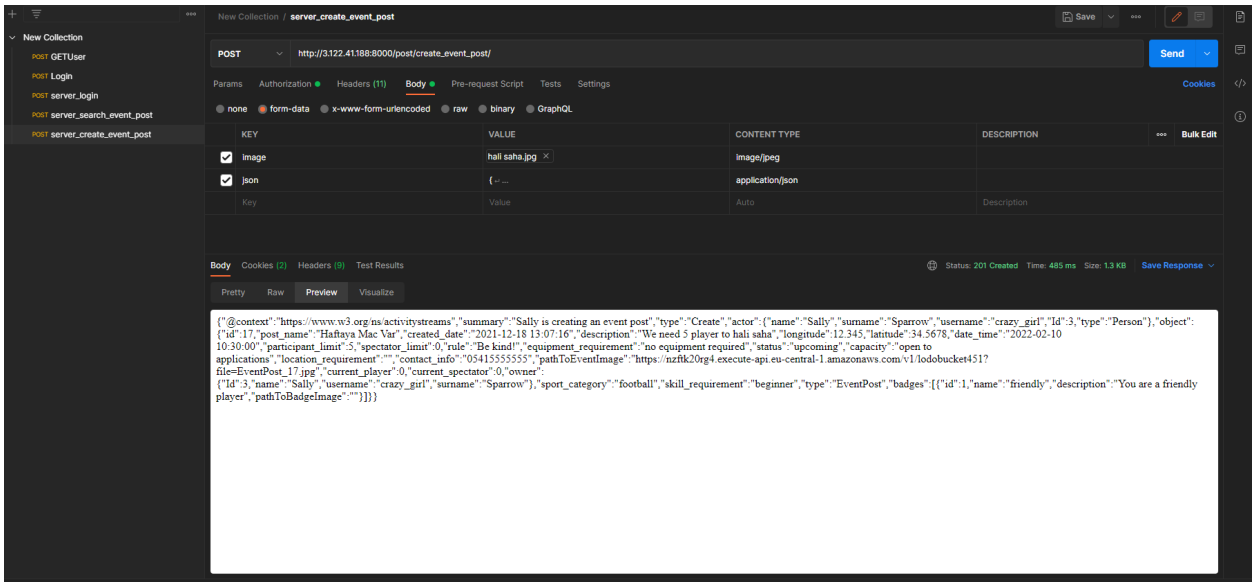
- Create event post
- Create equipment post
- Search event

But first you need to login.

Login:



Create event post:



A clear view of jsons:

Json in request:

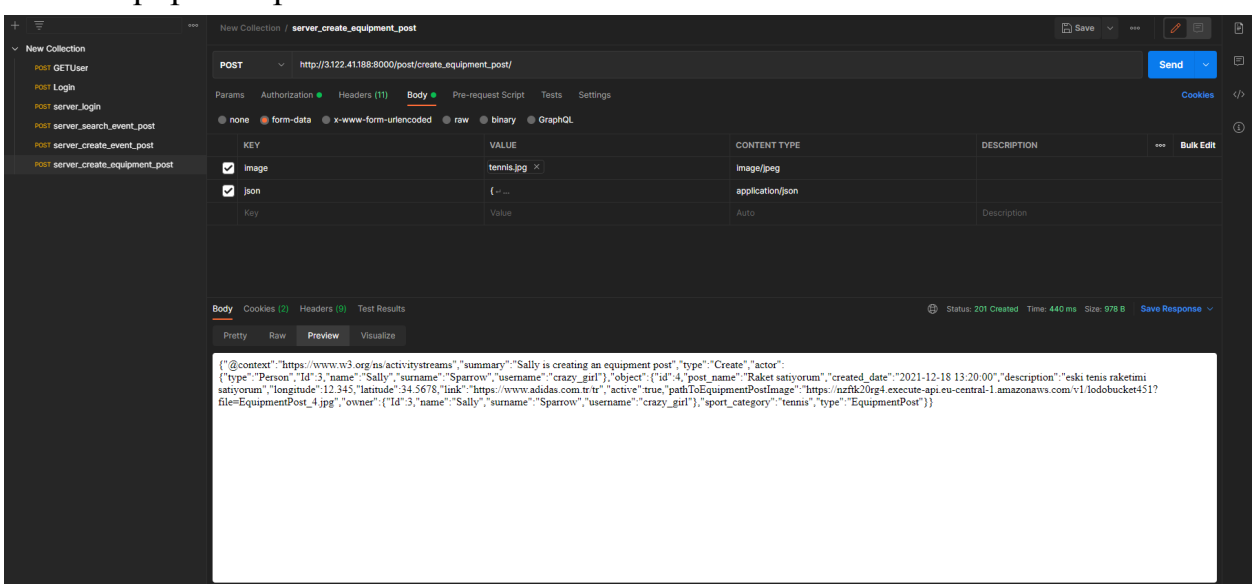

```
{
   "@context":"https://www.w3.org/ns/activitystreams",
   "summary":"Sally is creating an event post",
   "type":"Create",
   "actor":{
      "type":"Person",
      "name":"Sally",
      "surname":"Sparrow",
      "username":"crazy_girl",
      "Id":3
   },
   "object":{
      "type":"EventPost",
      "owner_id":3,
      "post_name":"Haftaya Mac Var",
      "sport_category":"football",
      "longitude":12.345,
      "latitude":34.5678,
      "description":"We need 5 player to hali saha",
      "pathToEventImage":null,
      "date_time":"2022-02-10 10:30",
      "participant_limit":5,
      "spectator_limit":0,
      "rule":"Be kind!",
      "equipment_requirement":"no equipment required",
      "location_requirement":"",
      "contact_info":"05415555555",
      "skill_requirement":"beginner",
      "repeating_frequency":1,
      "badges":[
         {
            "id":1,
            "name":"friendly",
            "description":"You are a friendly player",
            "pathToBadgeImage":""
         }
      ]
   }
}
```

Json in response:


```
{
  "@context":"https://www.w3.org/ns/activitystreams",
  "summary":"Sally is creating an event post",
  "type":"Create",
  "actor":{
    "name":"Sally",
    "surname":"Sparrow",
    "username":"crazy_girl",
    "Id":3,
    "type":"Person"
  },
  "object":{
    "id":17,
    "post_name":"Haftaya Mac Var",
    "created_date":"2021-12-18 13:07:16",
    "description":"We need 5 player to hali saha",
    "longitude":12.345,
    "latitude":34.5678,
    "date_time":"2022-02-10 10:30:00",
    "participant_limit":5,
    "spectator_limit":0,
    "rule":"Be kind!",
    "equipment_requirement":"no equipment required",
    "status":"upcoming",
    "capacity":"open to applications",
    "location_requirement":"",
    "contact_info":"05415555555",
    "pathToEventImage":"https://nzftk20rg4.execute-api.eu-central-1.amazonaws.com/v1/lodobucket451?file=EventPost_17.jpg",
    "current_player":0,
    "current_spectator":0,
    "owner":{
      "Id":3,
      "name":"Sally",
      "username":"crazy_girl",
      "surname":"Sparrow"
    },
    "sport_category":"football",
    "skill_requirement":"beginner",
    "type":"EventPost",
    "badges":[
      {
        "id":1,
        "name":"friendly",
        "description":"You are a friendly player",
        "pathToBadgeImage":""
      }
    ]
  }
}
```


Press Esc to exit full screen

Create equipment post:



A clear view of jsons:

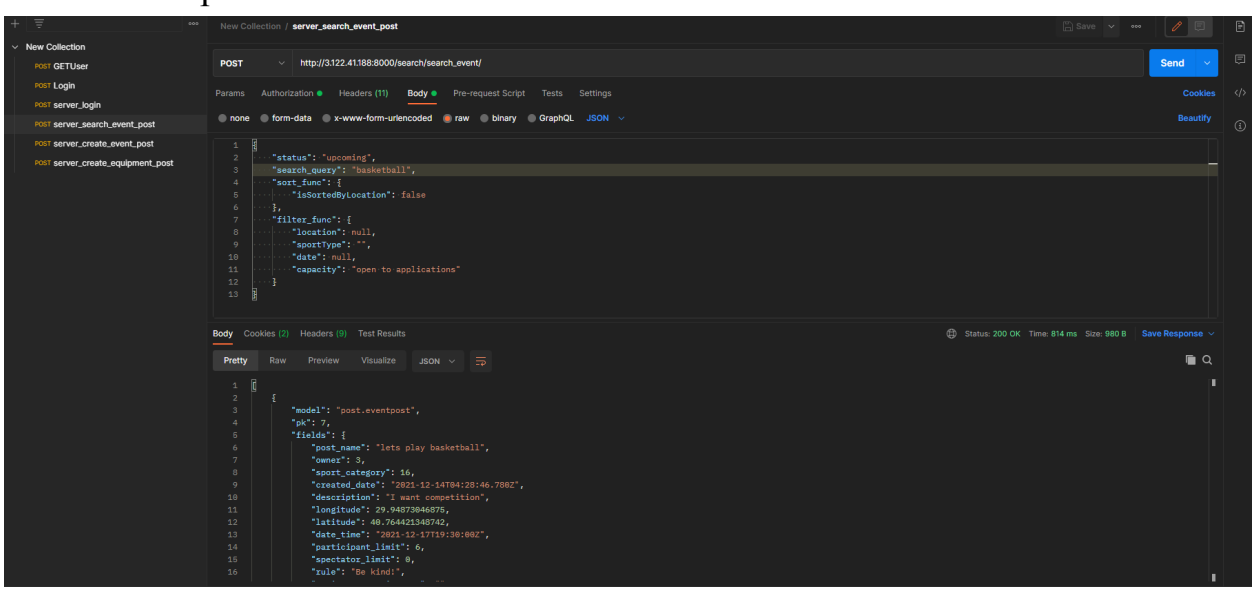
Json in request:


```
{
   "@context":"https://www.w3.org/ns/activitystreams",
   "summary":"Sally is creating an equipment post",
   "type":"Create",
   "actor":{
      "type":"Person",
      "Id":3,
      "name":"Sally",
      "surname":"Sparrow",
      "username":"crazy_girl"
   },
   "object":{
      "type":"EquipmetPost",
      "owner_id":3,
      "post_name":"Raket satiyorum",
      "sport_category":"tennis",
      "longitude":12.345,
      "latitude":34.5678,
      "description":"eski tenis raketimi satiyorum",
      "pathToEquipmentPostImage":null,
      "link":"https://www.adidas.com.tr/tr"
   }
}
```

Json in response:


```
{
  "@context":"https://www.w3.org/ns/activitystreams",
  "summary":"Sally is creating an equipment post",
  "type":"Create",
  "actor":{
    "type":"Person",
    "Id":3,
    "name":"Sally",
    "surname":"Sparrow",
    "username":"crazy_girl"
  },
  "object":{
    "id":4,
    "post_name":"Raket satiyorum",
    "created_date":"2021-12-18 13:20:00",
    "description":"eski tenis raketimi satiyorum",
    "longitude":12.345,
    "latitude":34.5678,
    "link":"https://www.adidas.com.tr/tr",
    "active":true,
    "pathToEquipmentPostImage":"https://nzftk20rg4.execute-api.eu-central-1.amazonaws.com/v1/lodobucket451?file=EquipmentPost_4.jpg",
    "owner":{
      "Id":3,
      "name":"Sally",
      "surname":"Sparrow",
      "username":"crazy_girl"
    },
    "sport_category":"tennis",
    "type":"EquipmentPost"
  }
}
```


Search event post:



Json in response:


```
[
  {
    "model":"post.eventpost",
    "pk":7,
    "fields":{
      "post_name":"lets play basketball",
      "owner":3,
      "sport_category":16,
      "created_date":"2021-12-14T04:28:46.780Z",
      "description":"I want competition",
      "longitude":29.94873046875,
      "latitude":40.764421348742,
      "date_time":"2021-12-17T19:30:00Z",
      "participant_limit":6,
      "spectator_limit":0,
      "rule":"Be kind!",
      "equipment_requirement":"",
      "status":"upcoming",
      "capacity":"open to applications",
      "location_requirement":"",
      "contact_info":"054155555",
      "pathToEventImage":"https://nzftk20rg4.execute-api.eu-central-1.amazonaws.com/v1/lodobucket451?file=EventPost_7.jpg",
      "skill_requirement":1,
      "current_player":0,
      "current_spectator":0
    }
  }
]
```


Implementation of core functionality

Answer the following questions for path functionality for SDP groups and Mapping functionality for all other groups.
- Describe the status of this functionality
- Describe in detail the work that has been completed regarding this functionality.
- Describe the work that remains for this functionality.

Android Mapping Functionality

Map Functionality for mobile application is working properly. Firstly, I created a Google Dev. account and got an API key including the Google Maps SDK for Android. After that, I searched and thought about how to implement “marking a location” and “drawing circles around marked locations”

functionalities. Then, I implemented the code providing those marking and circling functionalities. The code part can be found in the following link: [Google Maps Functionality for Android](). If the user taps on a location once on the map, there will appear a red mark icon on the tapped point. Later on, if s/he long presses to a point on the map, there will appear a blue circle whose center is firstly tapped point and, whose radius is the difference between long pressed point and firstly tapped point (the firstly marked location). If the user wants to change the radius of the circle or covered area, s/he can simply long press another point on the map and the new radius will be set accordingly. Since my Google Developer Account has limited functionality, to prevent the extra costs and attacks, I removed the API key from our Android Manifest file on GitHub. So, in the next, -if exists- I will search for a free Google Developer Account for development purposes only and will use its public API in our Android Manifest.

Backend Mapping Functionality

Map functionality for backend works with latitude and longitude information that frontend and android provided. Backend records all location information for an event in post tables. Then when a search function is called, this latitude and longitude information is used for filtering. Backend is provided with additional radius information. This is basically the search radius. Backend filters the events whether they reside in a circle with provided radius or not.

Frontend Mapping Functionality

Map functionality for the frontend working properly. There are not any missing parts. The processes done are listed like:

- I searched for map component. I found out there are multiple implementation of Google Maps in React, after elaborate investigation, i found one component called @react-google-maps/api. As I understand, it is what we want

  - I discovered that Google Maps is no longer free. Therefore, there is a watermark which says 'development purpose only' on the map.

Hence, i decided to use another map component since it's not appropriate in our condition to have watermark on the map.

- I searched and I found out that Open Street Map is free, and there are official implementation in React called 'React Leaflet'so I will use it.
- React Leaflet has support to add a marker, a circle around the marker, and functionality of changing radius.
- I had problem with radius since it was not in kilometer or lan/ltd type. I made several experiences and researches, then I found coefficient to convert it kilometer and lan/ltd.
- I added the radius functionality with help of official documentation. There was problem with non-number values, I solved it with help of Material ui number api
- I also added functionality to locate users since it will give a better user experience to the user while he/she is searching for events.

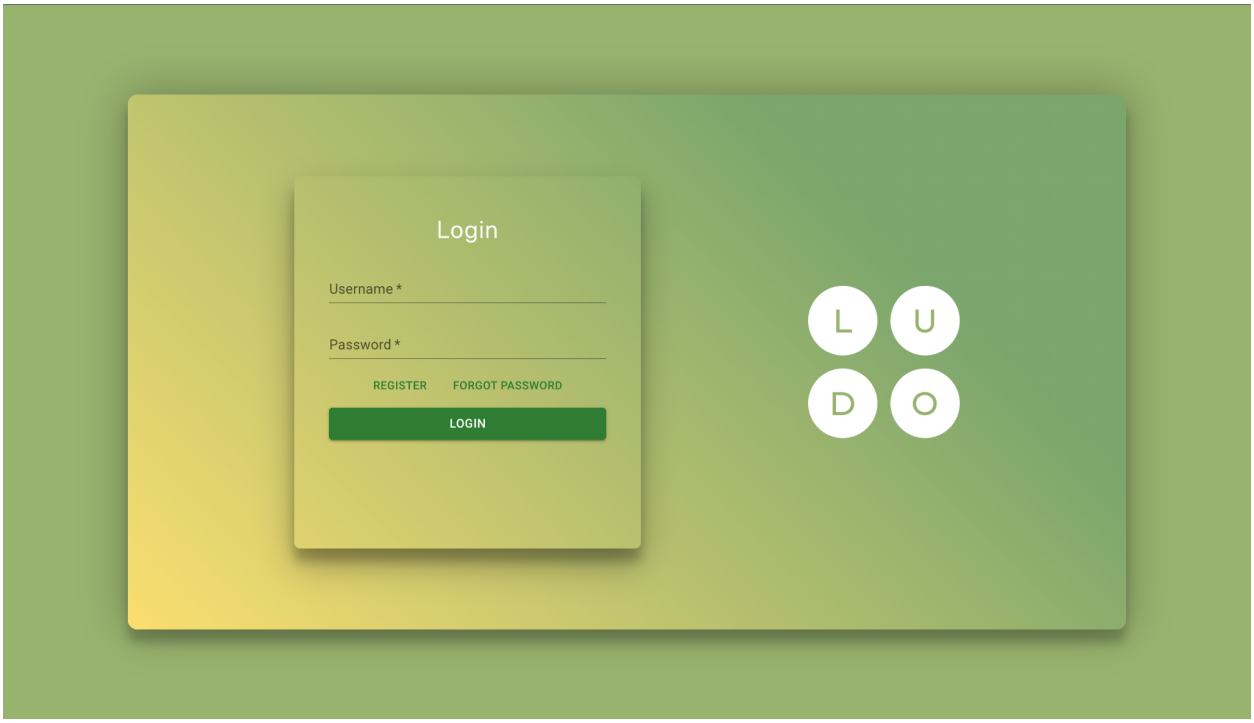
Functionalities are listed like:

- Show the map
- Choose a place and add marker
- Draw a circle around the marker
- Change the radius of the circle (unit is kilometer)
- Locate the user and jump map to that location

User Interface/User Experience
For each user interface design provide the following: link to the code in the project repository ,screenshot of all implemented Web interfaces,screenshot of all implemented Mobile interfaces

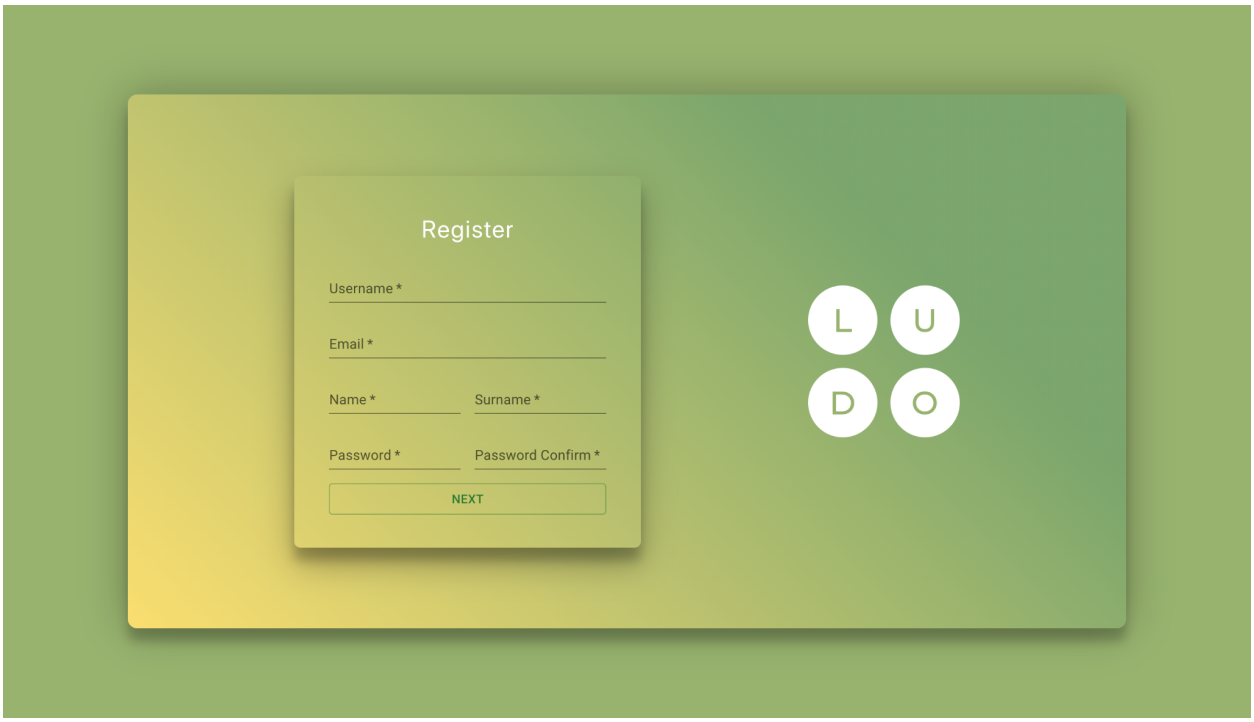
Login Page

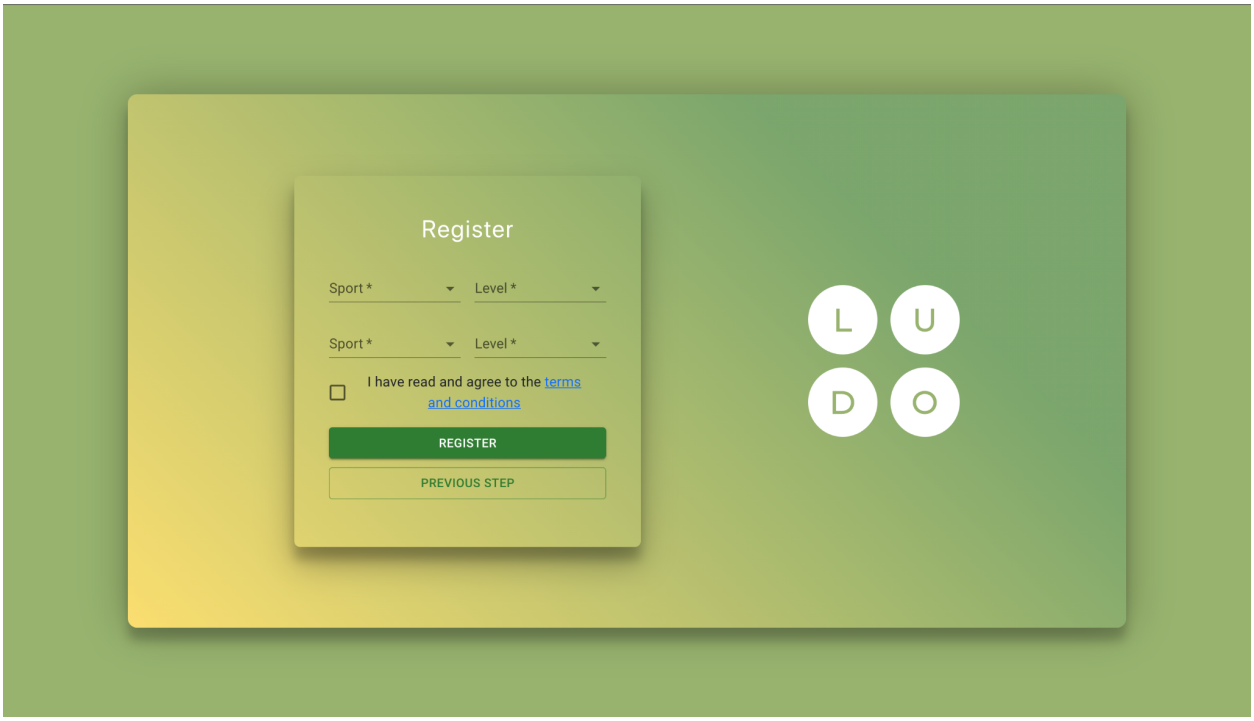
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/LoginScreen.jsx



Register Page

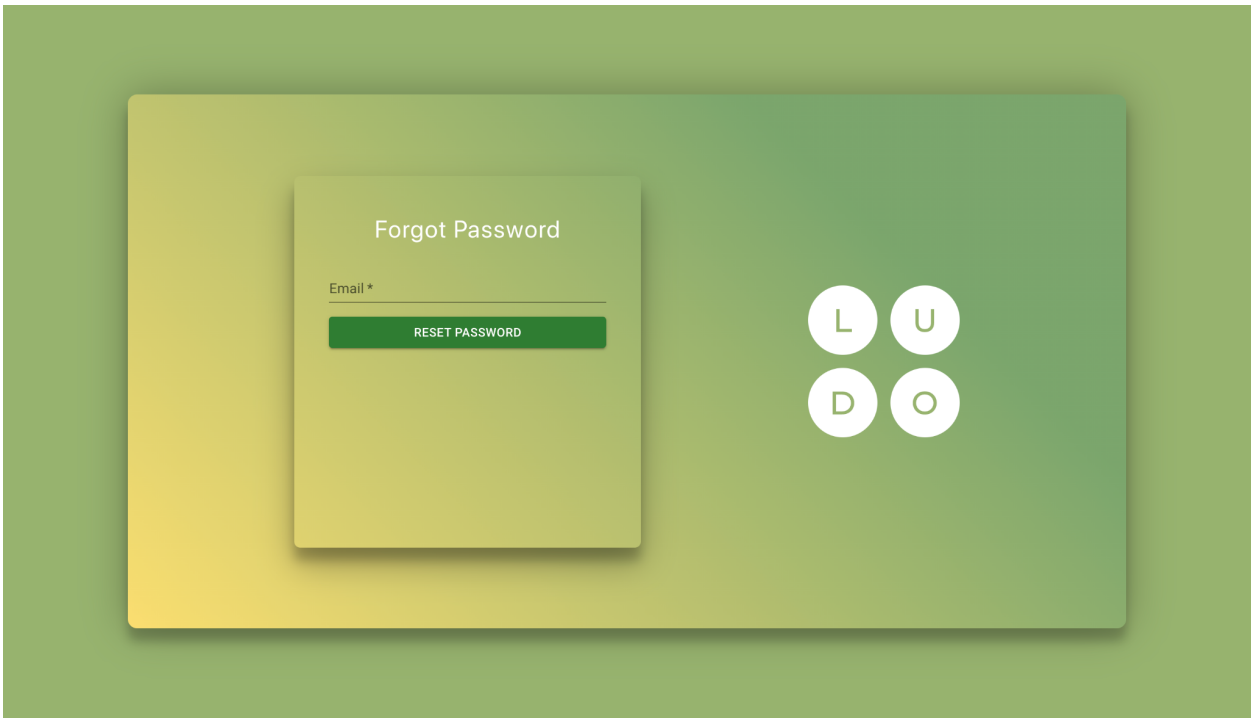
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/RegisterScreen.jsx





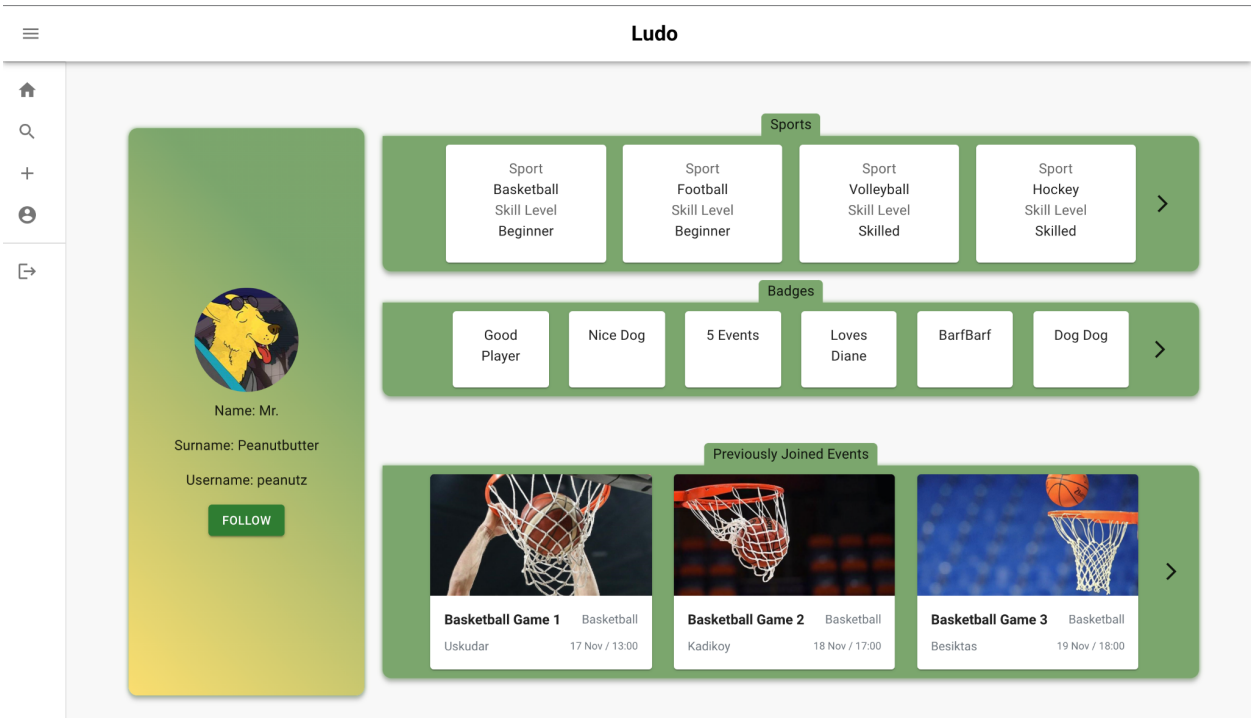
Forgot Password Page

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/ForgotPasswordScreen.jsx



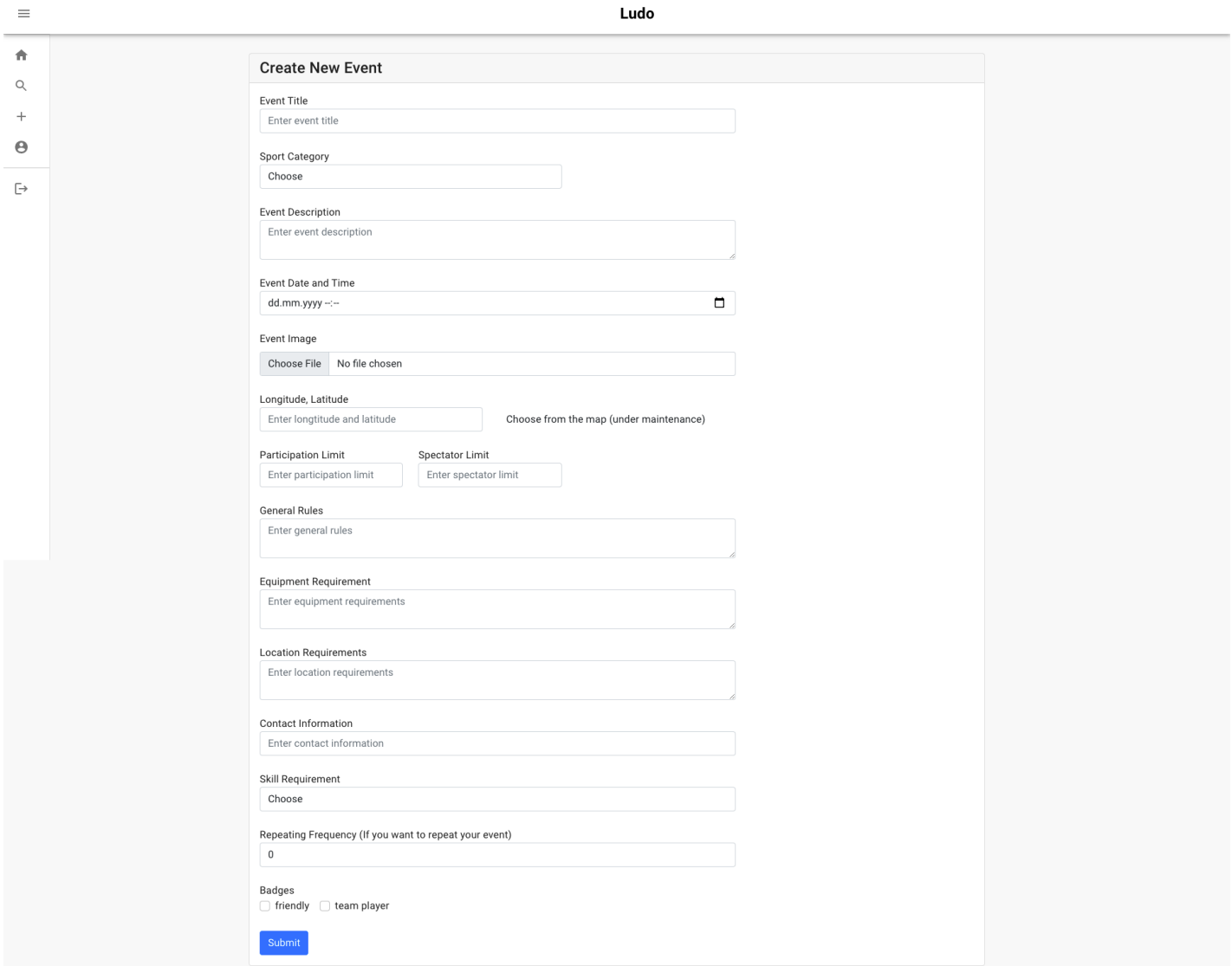
Profile Page

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/Profile.jsx



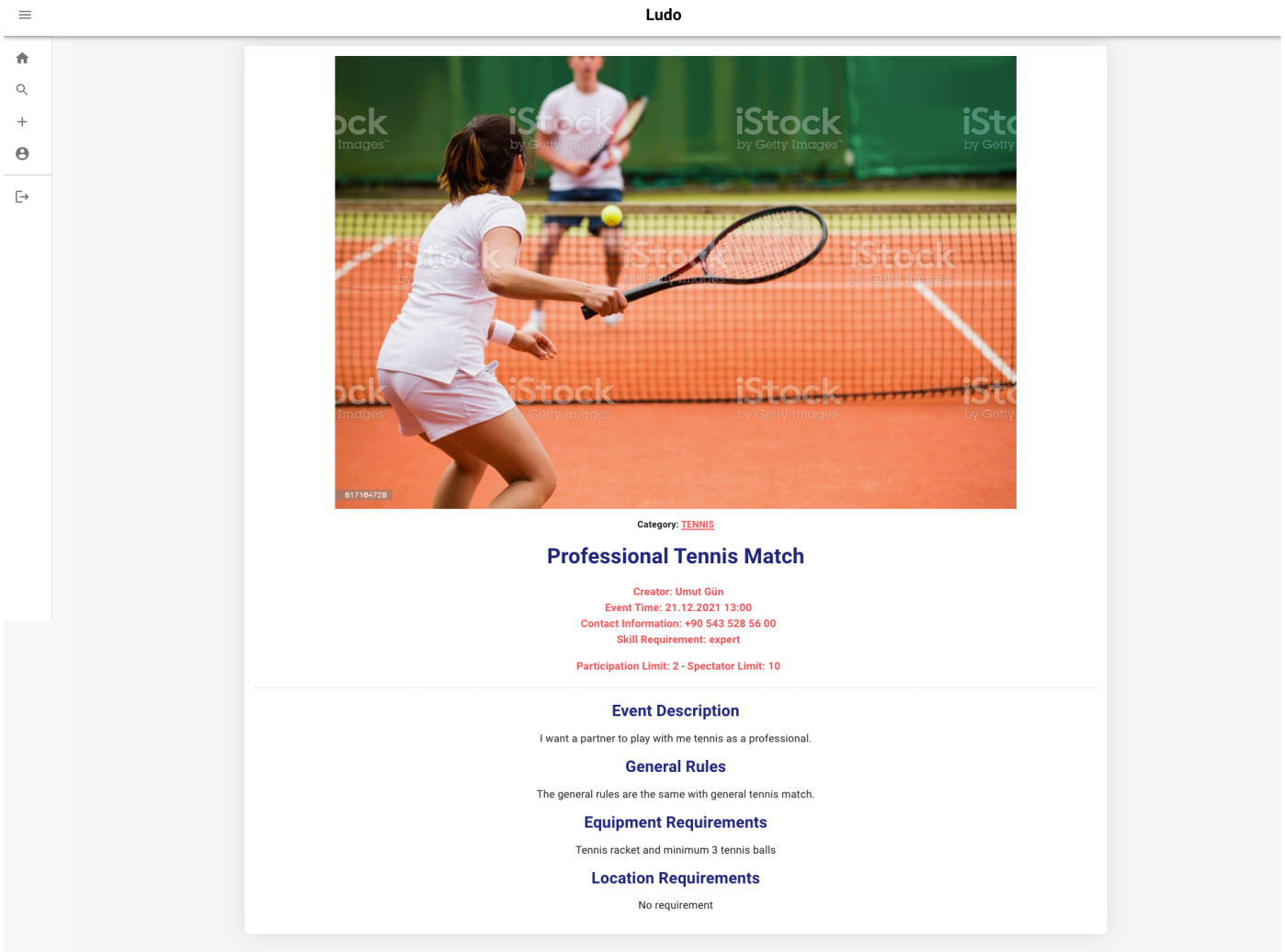
Creation Event Post Page

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/CreateEventScreen.js



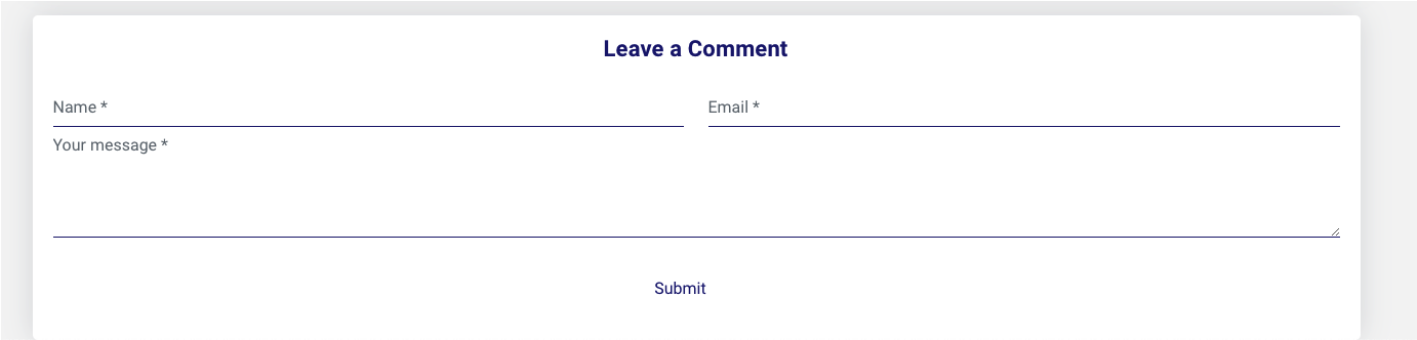
Event Detail Page

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/EventDetailScreen.js



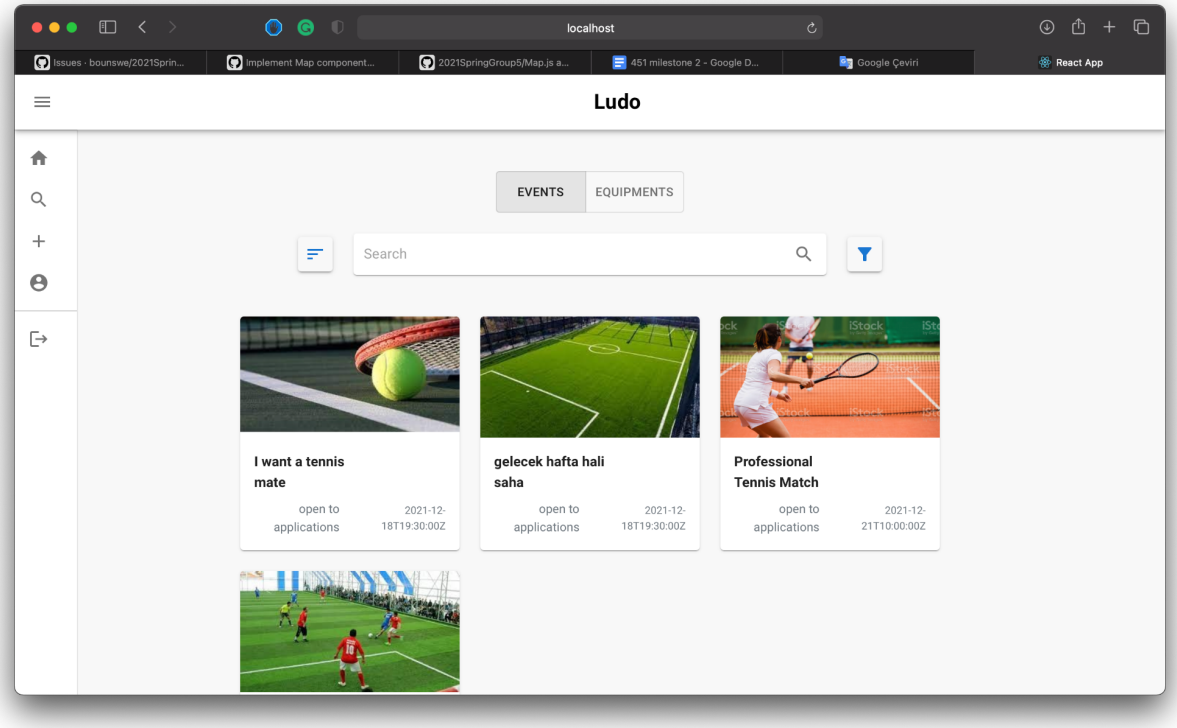
Comment Feature

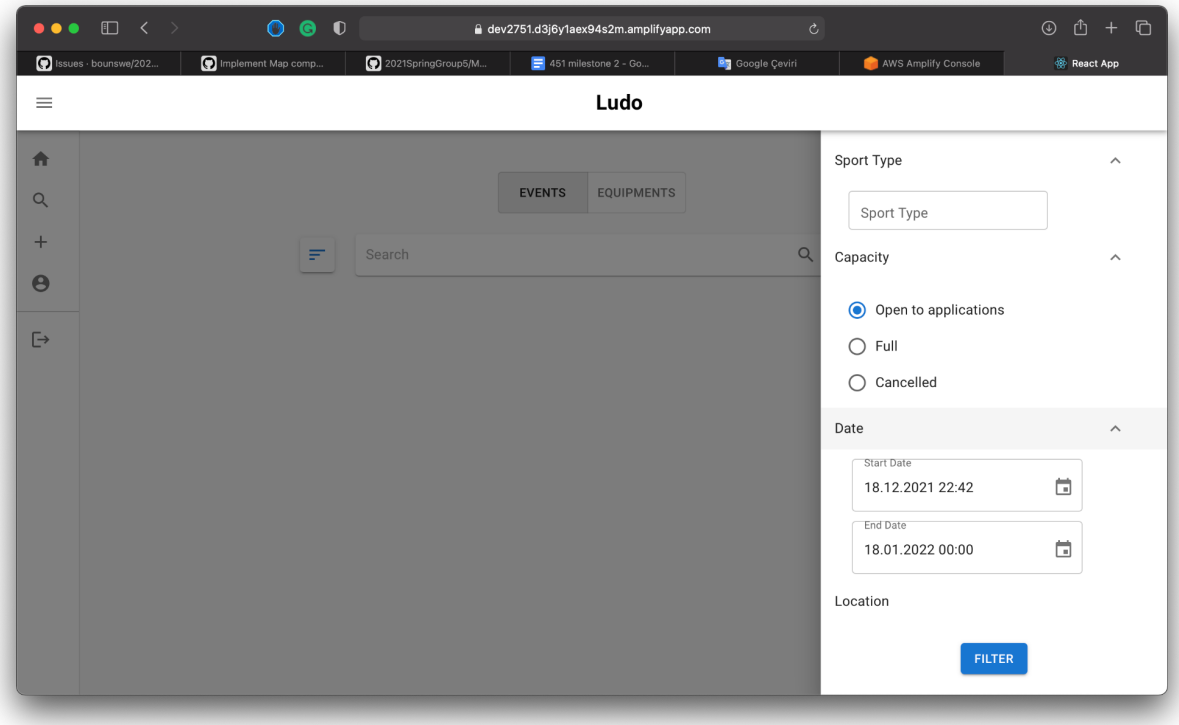
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/components/Common/Comment.js



Search and Filter Page

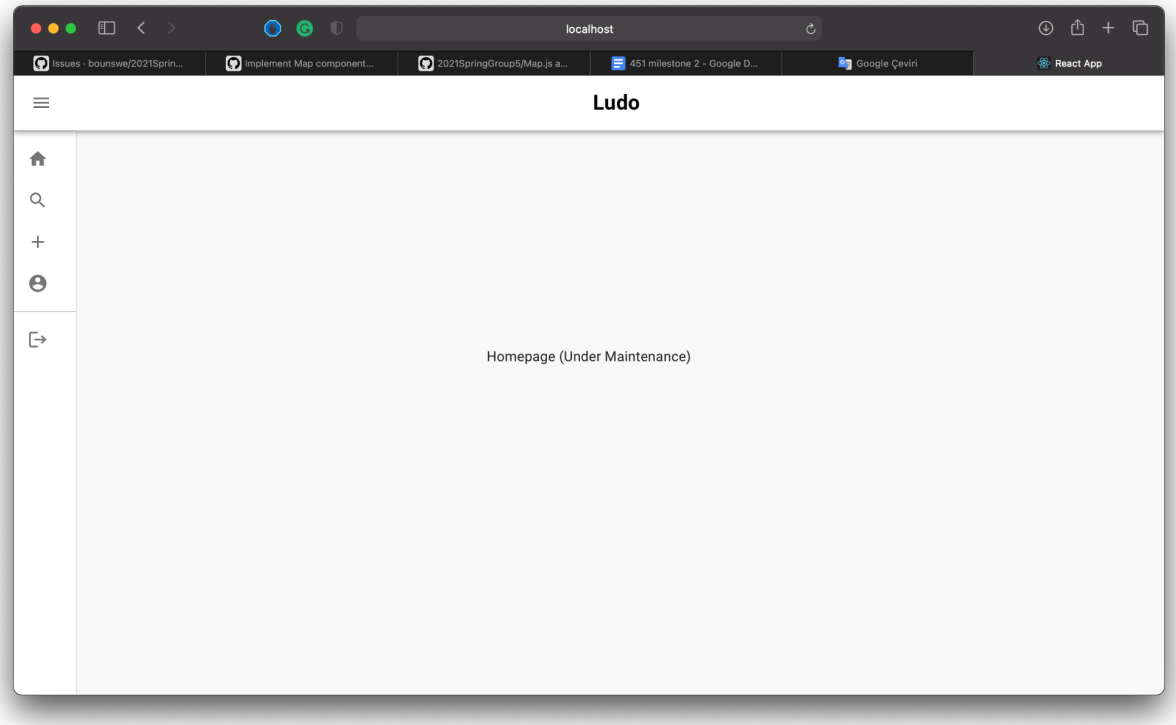
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/screens/SearchScreen.js





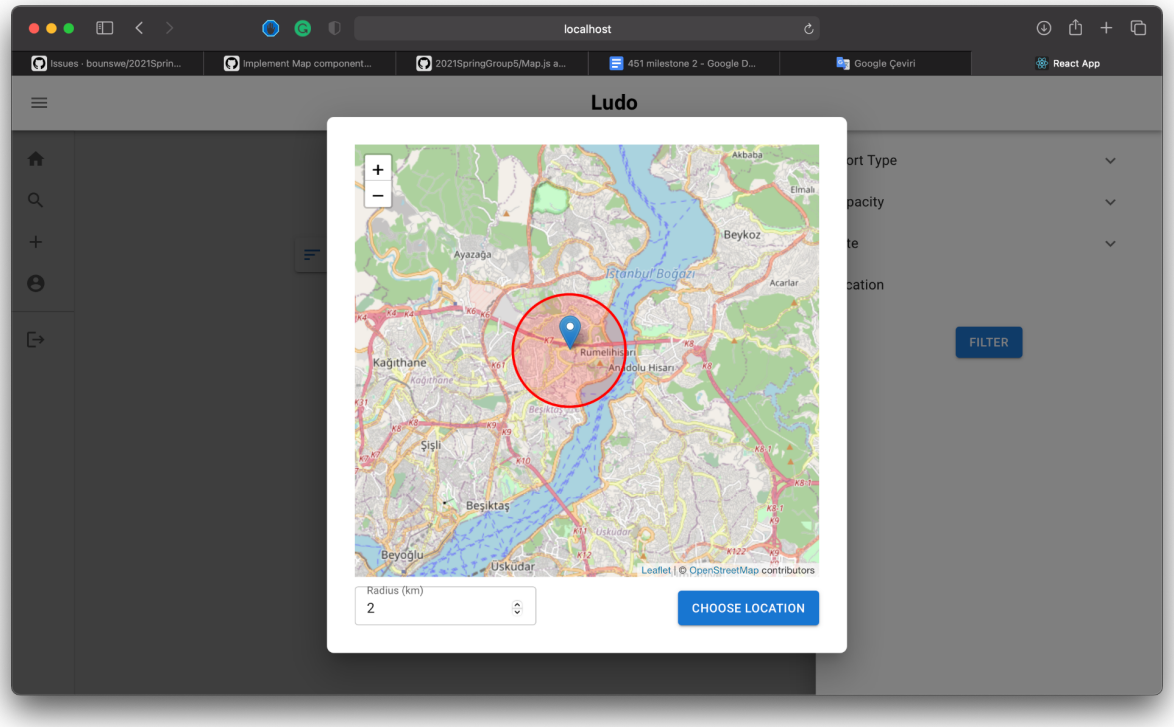
Header and Drawer

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/components/Common/Header.js



Map Component

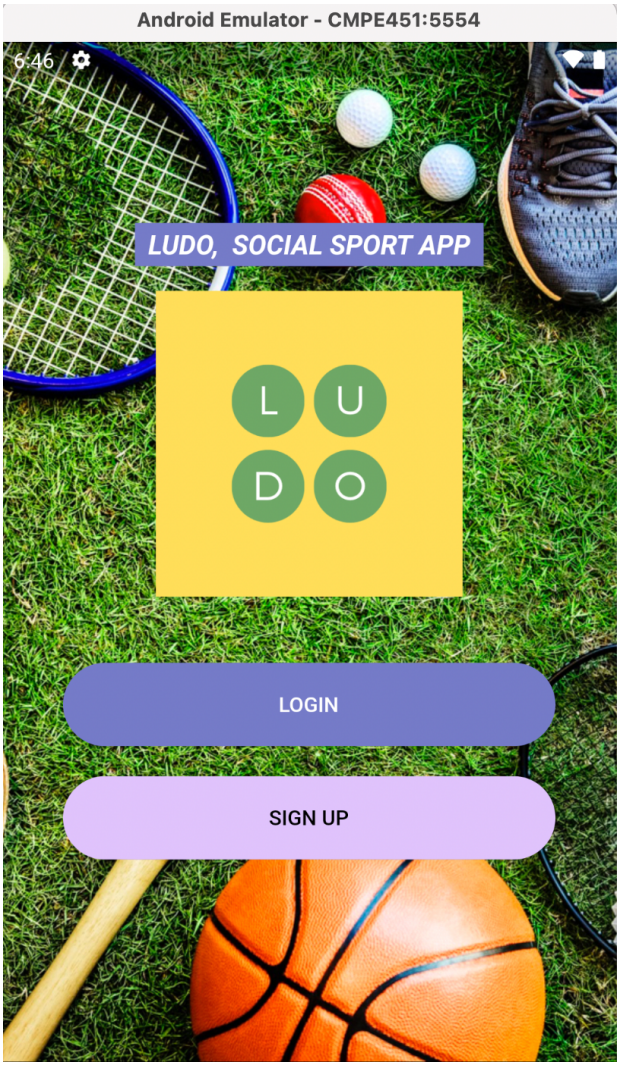
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/frontend/app/src/components/Common/Map.js



Android Interfaces

<u>Welcome Screen</u>

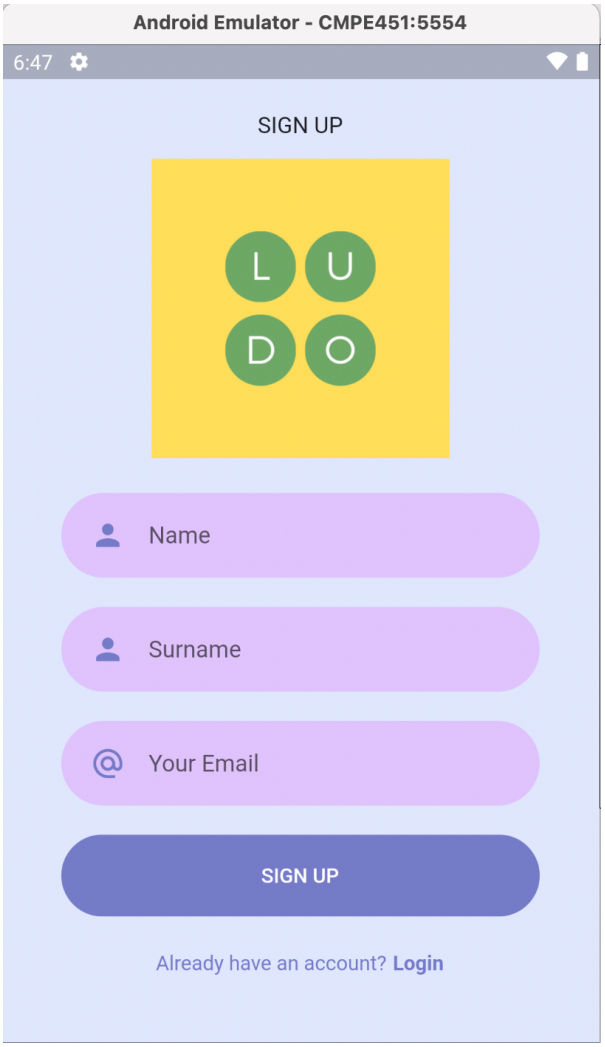
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/welcome

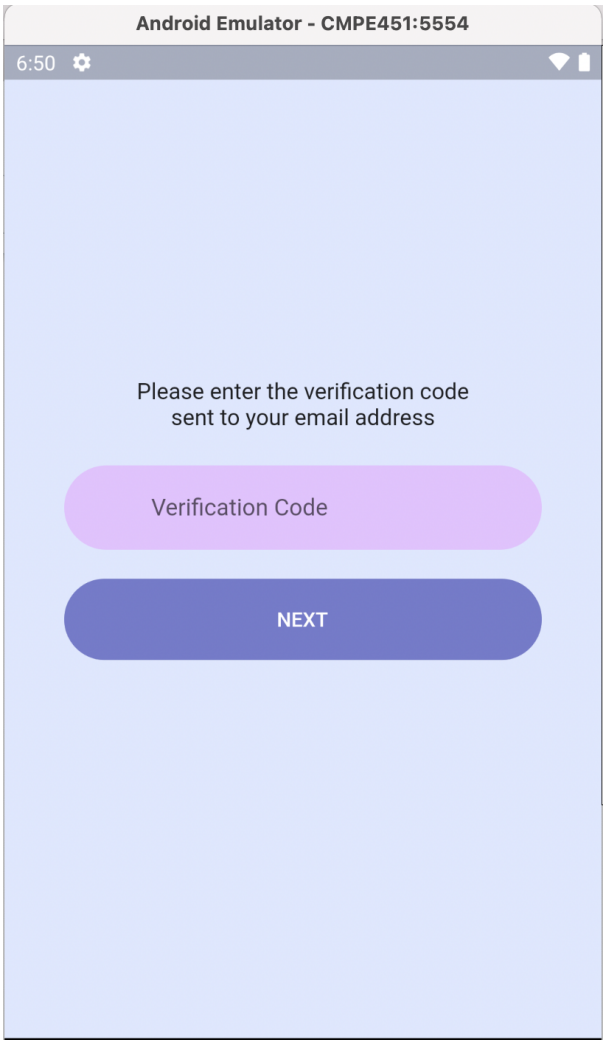


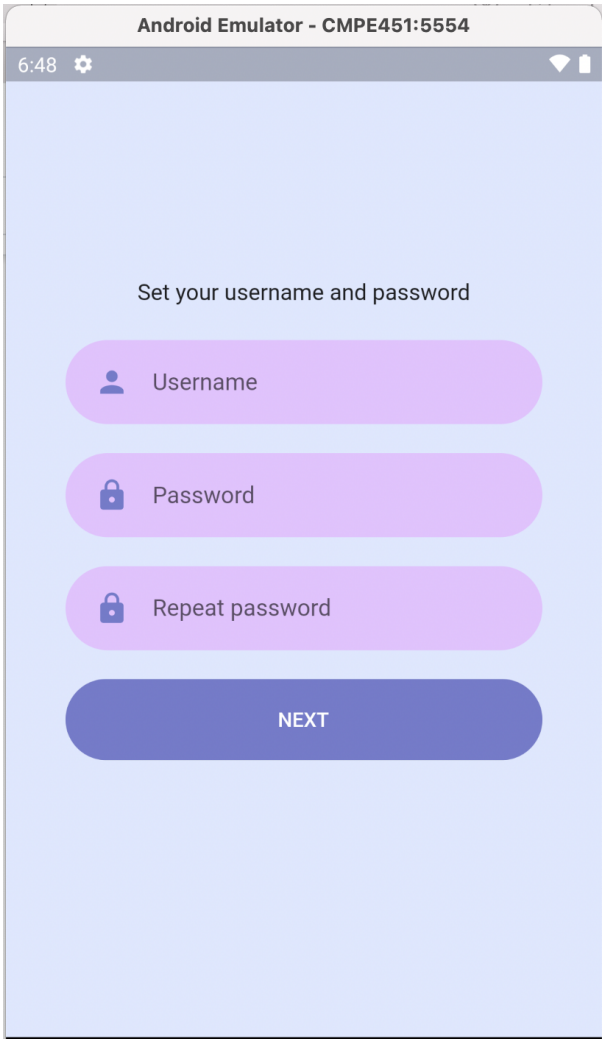
Register Screens

- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/signup
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/email_confirm

- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/user_create/components
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/fav_sports
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/email_confirm

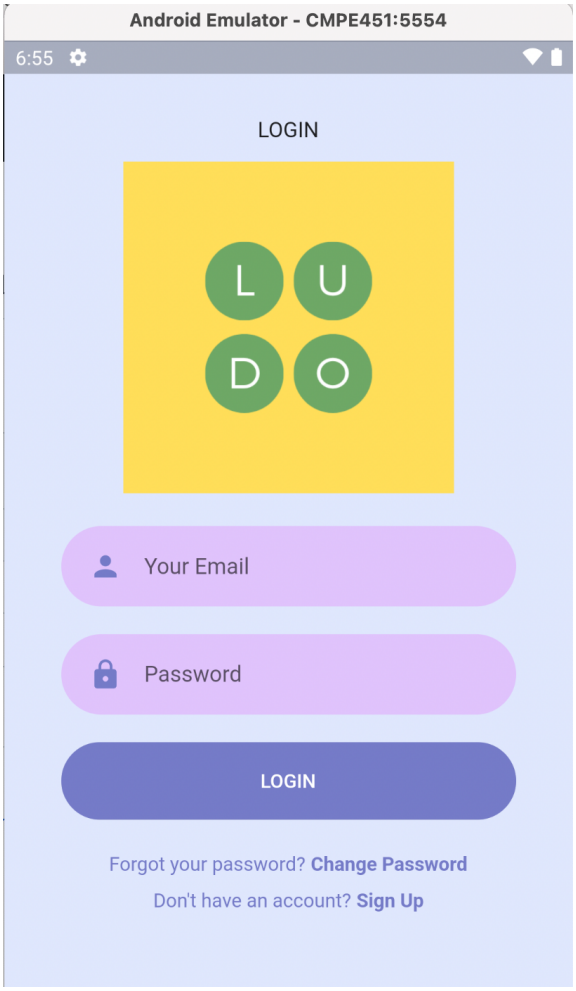






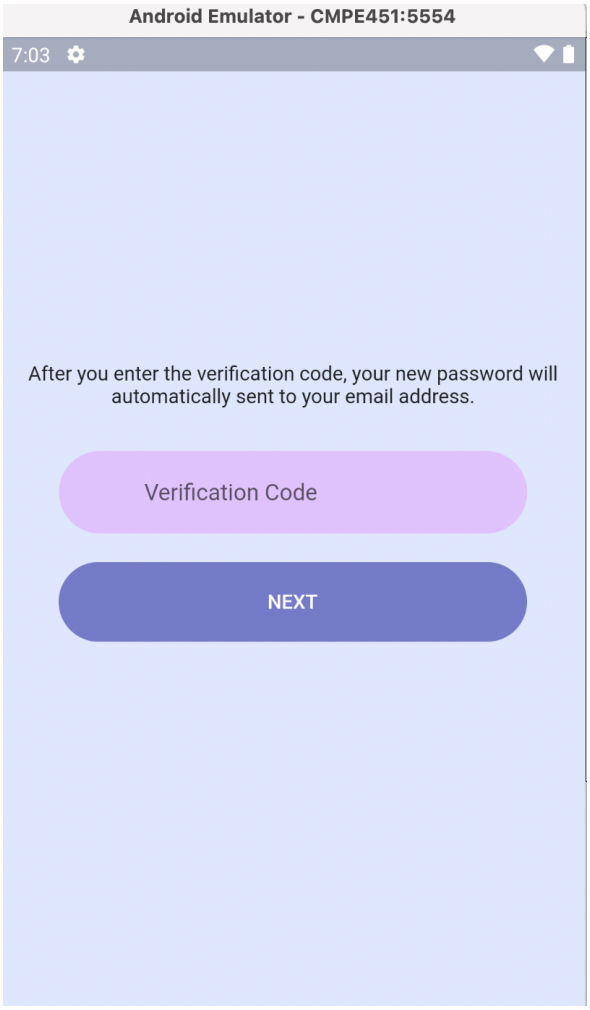
Login Screen

- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/login



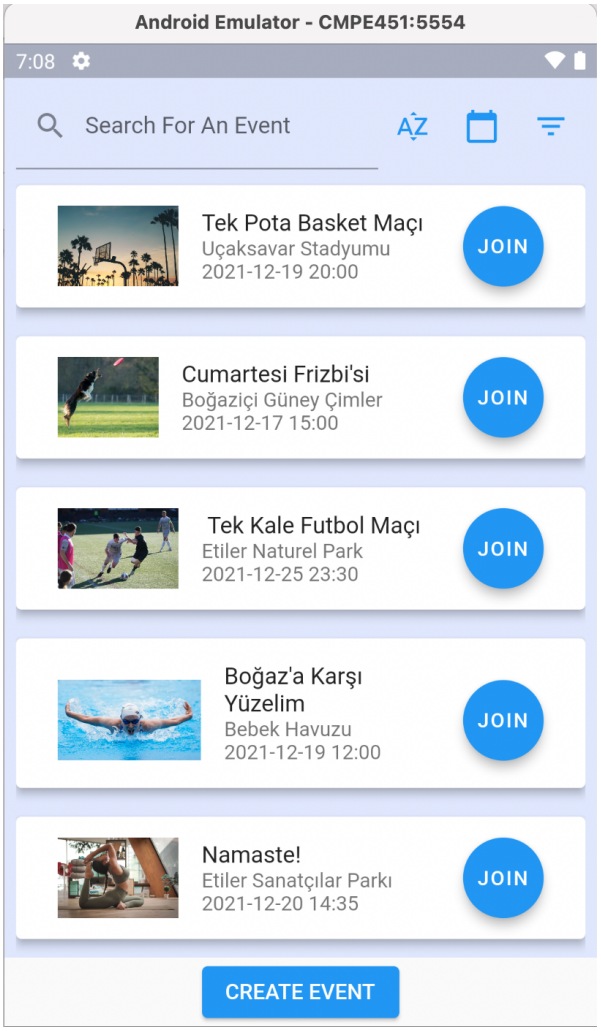
Forgot Password Screen

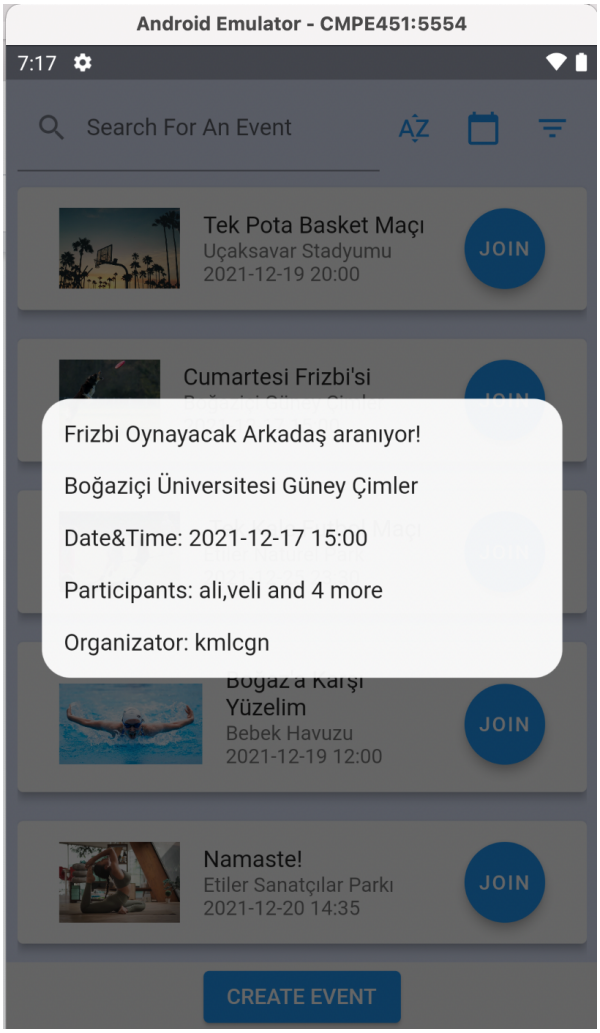
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/change_password



Home Screen (Search and Sort Screens also on this Page) and Event Detail Pop Up

- https://github.com/bounswe/2021SpringGroup5/blob/master/project/android/app/lib/screens/main_events/main_event_screen.dart

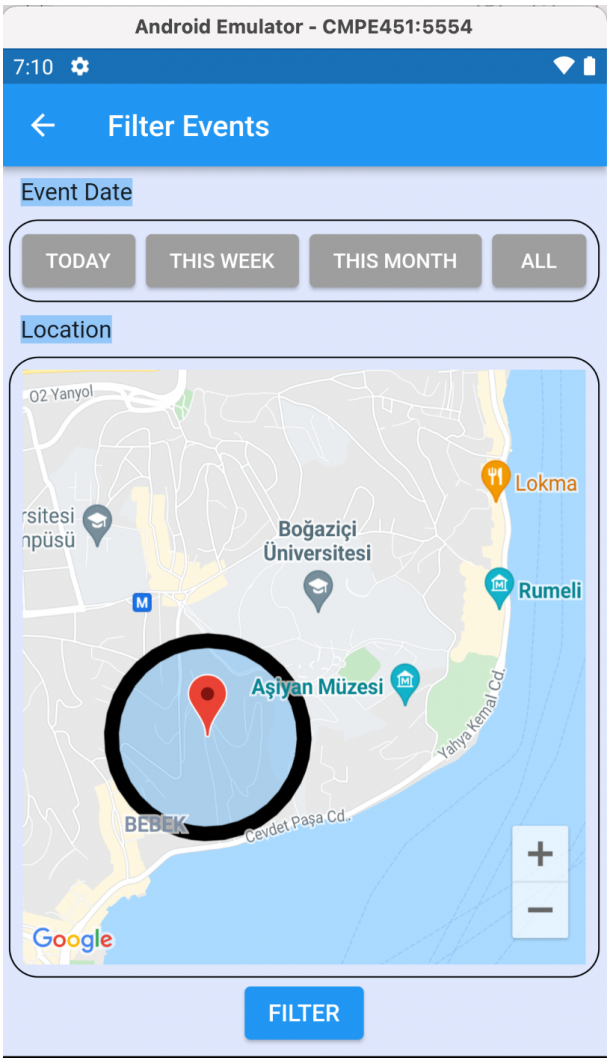




Filter Screen

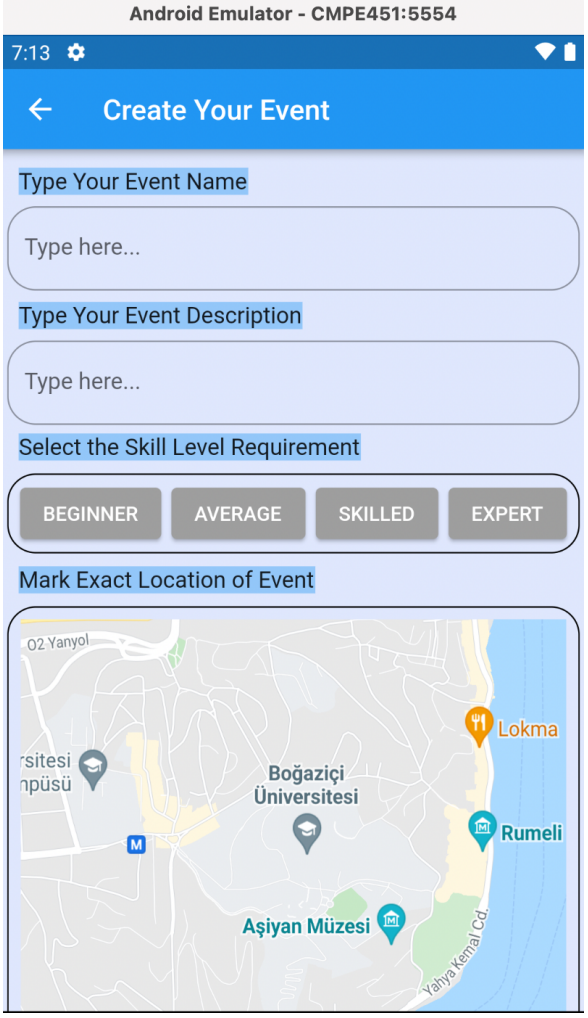
- https://github.com/bounswe/2021SpringGroup5/blob/master/project/android/app/lib/screens/filter_popup/filter_popup_screen.dart

- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/google_maps



Create Event Screens

- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/create_event
- https://github.com/bounswe/2021SpringGroup5/tree/master/project/android/app/lib/screens/google_maps



Standards

a) Activity stream

We examined the activity stream structıre and applied its json convention to the requests that are covered by the activity stream. We wrote down input and output jsons of these functions. Those functions are create event post, create equipment post, delete event post, delete equipment post, view event post details, view equipment post details, change event post information and change equipment post information. For these functions, frontend and mobile sends requests according to the activity stream conventions. Backend retrieves these jsons, processes them and also stores them in either EventPostActivityStream table or EquipmentPostActivityStream tables. Even though we haven't written down create comment jsons according to activity stream, we also created EventCommentActivityStream and EquipmentCommentActivityStream tables because we will use them as well when we implement these functionalities. Before saving the activity stream information into the database we save the actual "object", in our case this is an event or equipment post. Then we save the activity stream information into the database referring to the object that we have just created/deleted/updated as a foreign key. In all 4 of these activity stream tables there are generally 5 fields: object, context, summary, actor and type. This information comes from jsons that are created according to the w3c activity stream convention. "Object" field refers to the comment, event or equipment object and in the models they are foreign keys to the event, equipment or comment objects. "Context" field refers to the w3c activity stream documentation url and this is always this url "https://www.w3.org/ns/activitystreams", "summary" is the summary of the activity explaining what the user has just done, "actor" is referring to the user object who made this activity and actor is stored as a foreign key to this user object and "type" is the type of the activity such as create, delete or update. The communication between the backend and frontend/mobile is done with activity stream jsons for these activities. Frontend and mobile send requests according to this convention and backend returns the information according to this convention. But we also modified a little these jsons in this way: When the user creates an event and the information must be sent to the backend, w3c activity stream sends very little information, instead of that we list all the information of the post in the

“object” part of the json. Similarly, we return all the information of the object in the “object” part of the json. For the update case, in addition to context, type, summary, object keywords we added another keyword in the json that mobile and frontend sends to backend, “modifications”. The parts where the user changed the information are sent only here so the backend updates the object using only the information in the “modifications” keyword. As the output jsons, backend sends all the information about the object in the “object” keyword of the activity stream json structure, for the delete case, only success message is returned to the frontend and mobile because they only check the status code if everything is okay and don’t display anything.

b) Wiki data

We tried to find badges of sport on wiki data, but couldn't find any badges about sports, personal characteristics or something else. Currently we have 5 badges that we made up but we will replace them with the keywords that we will find in wiki data.