

The University of Akron

IdeaExchange@UAkron

---

Williams Honors College, Honors Research  
Projects

The Dr. Gary B. and Pamela S. Williams Honors  
College

---

Spring 2022

## Sportiasts

Yuvraj Subedi  
ys96@uakron.edu

Follow this and additional works at: [https://ideaexchange.uakron.edu/honors\\_research\\_projects](https://ideaexchange.uakron.edu/honors_research_projects)



Part of the [Other Computer Sciences Commons](#)

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

---

### Recommended Citation

Subedi, Yuvraj, "Sportiasts" (2022). *Williams Honors College, Honors Research Projects*. 1519.  
[https://ideaexchange.uakron.edu/honors\\_research\\_projects/1519](https://ideaexchange.uakron.edu/honors_research_projects/1519)

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact [mjon@uakron.edu](mailto:mjon@uakron.edu), [uapress@uakron.edu](mailto:uapress@uakron.edu).

**An exploration of Django and PostgreSQL to  
connect sport enthusiasts: Sportiasts**

Yuvraj Subedi  
Computer Science - Systems Track  
The University of Akron Honors Project  
Sponsor: Professor Dr. Zhong-Hui Duan

Spring 2022

# Table of Contents

---

- [Introduction](#)
  - [Goals and Objectives](#)
  - [Materials and Methods](#)
    - [Front-end](#)
      - [HTML](#)
      - [CSS](#)
      - [JavaScript](#)
      - [React.JS](#)
    - [Back-end](#)
      - [Python](#)
      - [Django](#)
      - [PostgreSQL](#)
  - [Design](#)
    - [Home Page](#)
    - [Sign Up](#)
    - [Log In](#)
    - [Homepage after logged in](#)
    - [Profile](#)
    - [Search](#)
    - [Create Event](#)
    - [Event Page](#)
    - [Chats](#)
  - [Implementation](#)
    - [manage.py](#)
    - [Project Files](#)
    - [Application Files](#)
    - [Template Files](#)
    - [Static Files](#)
  - [Results and Discussion](#)
  - [Conclusions](#)
  - [References](#)
- 
-

# Introduction

---

In this 21st century, the use of technology is rapidly growing. From children to seniors, technology has occupied its place in almost everyone's life. Along with the development of technologies, social media use has grown exponentially in the last decade or two. From regular family conversations to professional meetings, social media has been the most used platform lately. Big tech companies like Facebook, Google, Microsoft are social media platforms and are being used by majority of the world population. The popularity of social media comes from their versatility. Social media don't only let you chat with people or conduct meetings, but they have huge number of features like posting your daily photos, videos, or going live to show something, or searching people/event and many more features.

Social media platforms have made people's life easier. In the sports or athletic world too, there are many online/offline platforms to guide one's training or counting daily exercise or food intake level. There is a huge population of people who are not professional athletes but love sports in daily life. The main goal of this project is to build a platform to help people find or connect with other people who have similar sport/athletic interest. In simple terms, this application connects sport enthusiasts.

As of 2019, 95.6% of US population are engaged in sports or leisure activities directly or indirectly (Statista, 2022). This number is insanely high because of the human nature to be attracted by recreational activities. Unlike big tech platforms like Facebook or Google, this platform focuses mainly on sports. People who would like to play sport but haven't been able to find their partner, or team, or some sport events nearby can use this application to search, locate, and connect with people of similar sport interest. Apart from searching people and events, this platform lets people organize sport events and invite others to attend the event. With these being main features, there are many other features implemented on this project which focuses on connecting sport lovers through online platforms.

The first version of the application is an online web application with the flexibility to continue the project for ios/android phone applications.

---

---

## Goals and Objectives

---

The motivation to build this project comes from people's daily life need. For example, if someone travels from New York to Paris for 2 weeks business trip, there is a big chance that they want to play some sport but won't find anyone to play with. A small statistical example can explain this better; about 2.29 billion trips are made in US every year domestically and 93.1 million international trips are made from/to US every year (Ferris, condorferris.co.uk, 2022). A big percentage of this travelers are not able to enjoy their favorite sport because of their unfamiliarity with new place and people. A person wanting to go hiking will end up doing nothing in the room just because s/he couldn't find people to go hiking with.

Thus, this platform is focused on solving these problems by bringing the sport community together. The major goal of this project is to build a web application that lets user to do the following things:

- Let users register to the app (login/sign up)
- View future events/ currently happening events/ expired events.
- View all the events that the user has joined.

- Search any sporting event based on sport type/ location/ event title.
- Create new sporting event.
- Join current available sport event.
- Withdraw from a joined event.
- Organizer of the event can remove other users.
- A user can invite another user to an event.
- Direct message other users of the same event.
- View event location in a map.
- Notify users about the events happening nearby.
- Notify users about their event in advance.
- Have a group discussion/ group chat room for users of a particular event.
- User-friendly UI.

The main objective of these features is to provide users a comprehensive solution in connecting/creating a sport community.

---



---

## Materials and Methods

---

This project is based on web, so different web application frameworks and programming languages have been used to build this project. The major tools or frameworks used to design and develop this project are as follows:

### Front-end

The front-end of a website is everything the user either sees or interacts with when they visit the website (Airfocus, 2022). It defines the whole user interface of the website. It is responsible for the total look and feel of an online experience. The materials and tools used for the front-end of this application are described below.

- HTML

HyperText Markup Language (HTML) is the most basic building block of the Web development. It defines the framework and structure of the web content. HTML uses markup language to annotate text, images, and other content for display in a Web browser (developer.mozilla.org, 2022). Some examples of HTML elements are `<head>`, `<title>`, `<body>`, `<div>`, `<span>`, etc. These tags which are inside `< >` are case insensitive.

- CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML. One metaphor of explaining HTML and CSS would be HTML is a skeleton system of the web development whereas CSS is the muscular system. CSS adds presentation design and fills the framework defined by HTML.

- JavaScript

JavaScript, also known as JS, is a lightweight, interpreted compiled programming language. While it is best recognized as a Web page scripting language, it is also used in a variety of non-browser settings, including Node.js, Apache CouchDB, and Adobe Acrobat (developer.mozilla.org, 2022). JavaScript is a single-threaded, dynamic, prototype-based language that supports object-oriented, imperative, and declarative (e.g. functional programming) programming styles.

- React.JS

React is a JavaScript library for building user interfaces. React makes it easier to create interactive User Interfaces (Boduch et al, 2020). Implementation of React needs basic understanding of HTML and Javascript. React is currently maintained by Meta (formerly Facebook).

## Back-end:

Back end development refers to the server side of an application. Users can't see the backend work, but this part of the application is what communicates the database information to the browser (Airfocus, 2022). Different tools and programming languages have been used on this project for the backend part of the application which are described below.

- Python

Python is one of the most popular programming language in the world. Python lets programmers work quickly and integrate system more effectively and efficiently. It is most popular in machine learning, data science, mathematical computations, web development, and biological simulations.

- Django

Django is a high-level Python web framework that promotes speedy development as well as a clean, pragmatic design (Forcier et al, 2008). It was built to take care of a lot of the hassle of web development so developers can focus on developing the application instead of reinventing the wheel. It's free and open source. Django also helps developers avoid many common security mistakes (developer.mozilla.org, 2022).

- PostgreSQL

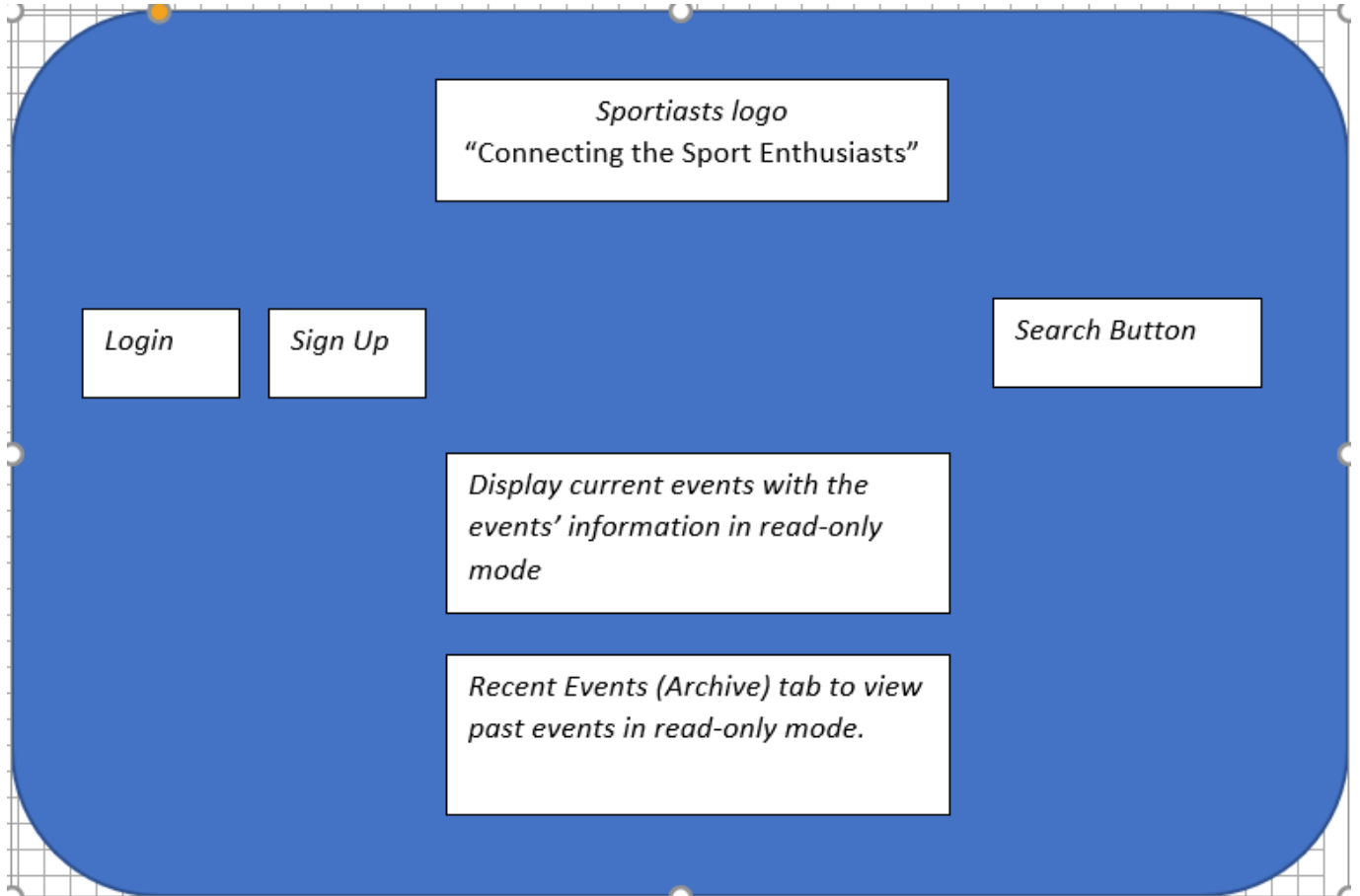
PostgreSQL is a sophisticated, open source object-relational database system that leverages and extends the SQL language, as well as a variety of other capabilities to reliably store and scale even the most complex data demands (Momjian, 2001). PostgreSQL is a robust database management system that operates on all major operating systems and includes powerful add-ons like the PostGIS geographical database extender. PostgreSQL is very extendable in addition to being free and open source. Developers can, for example, define their own data types, create custom functions, and even write code in a variety of programming languages without having to recompile the database.

---

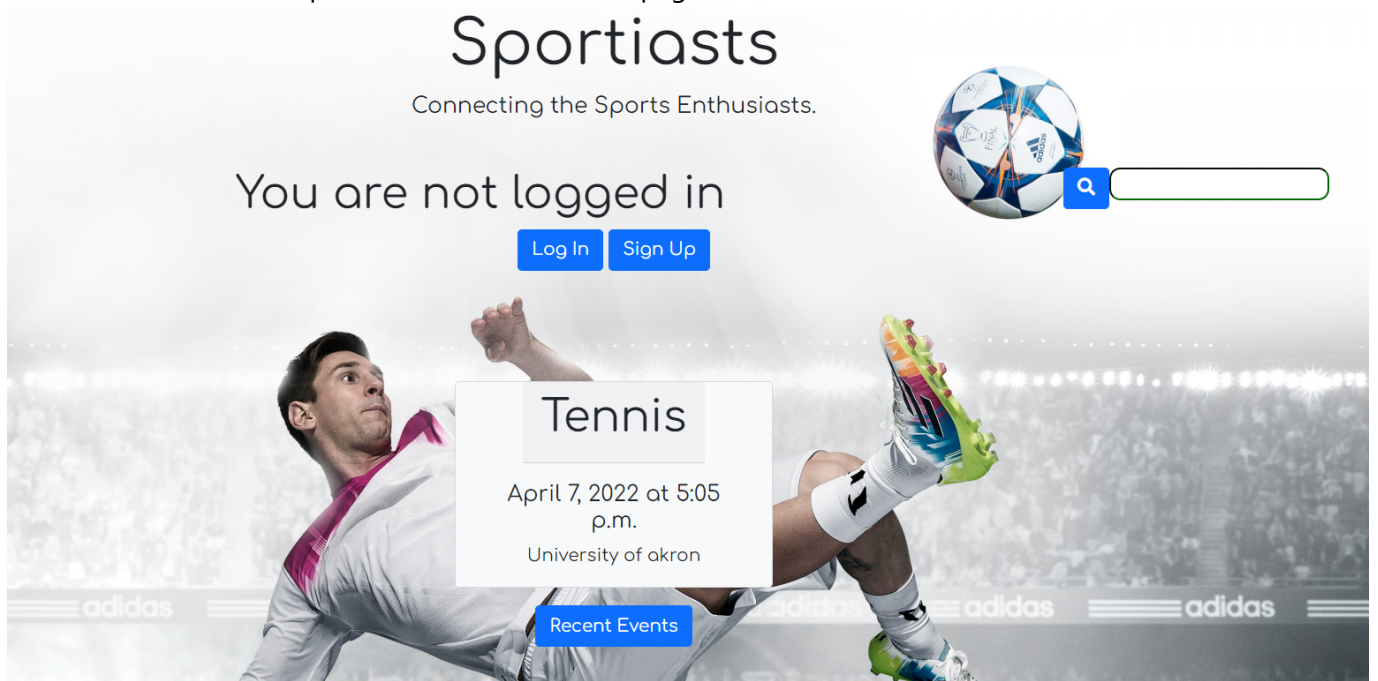
# Design

## Home Page

The current idea is to keep the home page as simple as possible. This page contains the name and logo of the project, login and signup buttons, search tab, and displays near future events. Underneath the events, there is a button to display past events too. The template for the home page is as follows:



The current front end implementation of the home page looks as:

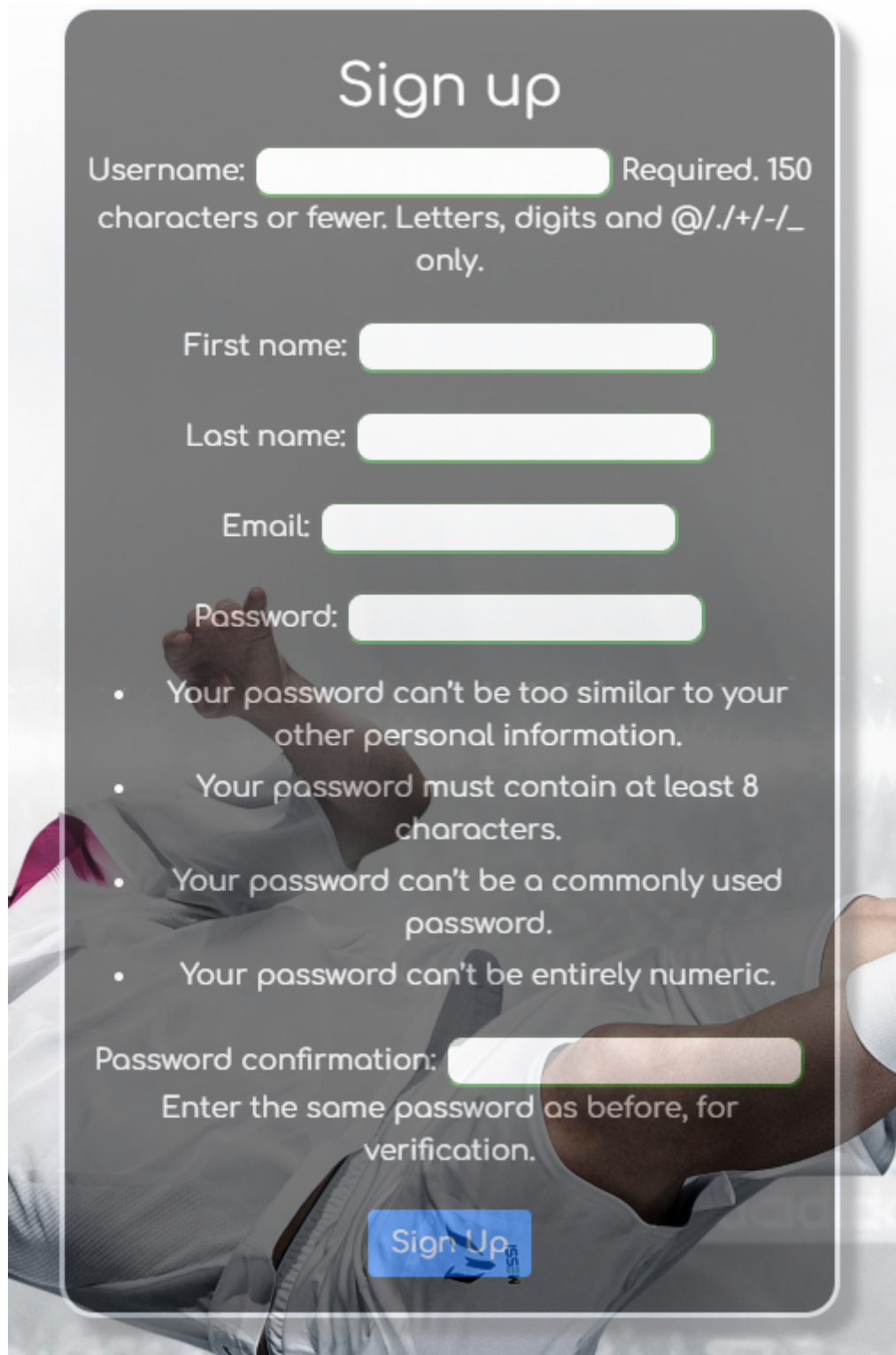


When the user is not logged in to their Sportiasts account, they have options to log in, sign up, search for

the events, view the future events, and go to the **Recent events** tab to view past events. The events viewed in this page are read-only because the user is not logged in to their account yet.

## Sign Up

This page lets new users to sign up with the Sportiast and create their own account. When the user signs up, the application asks for *Username, First Name, Last Name, Email, and Password*. The password is validated to make the passwords strong enough. The password can't be too similar to other personal information, the password must contain at least 8 characters, the password can't be commonly used password like hello123, and the password can't be entirely numeric. The raw password is not stored in the database so that the admin of the application too can't see the users password. The password is hashed using the built-in feature of the Django which uses pbkdf2\_sha256 algorithm with 216000 iterations to hash the passwords.



**Sign up**

Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

First name:

Last name:

Email:

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:

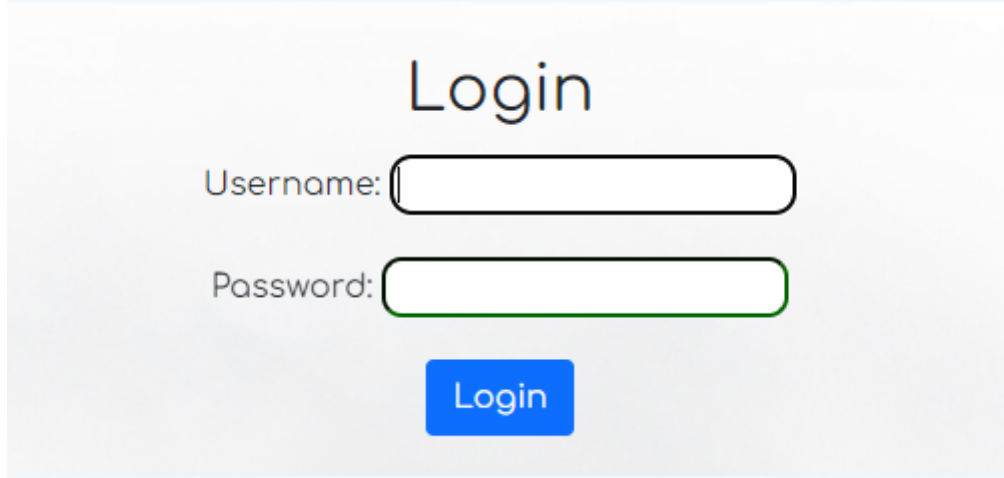
Enter the same password as before, for verification.

**Sign Up**



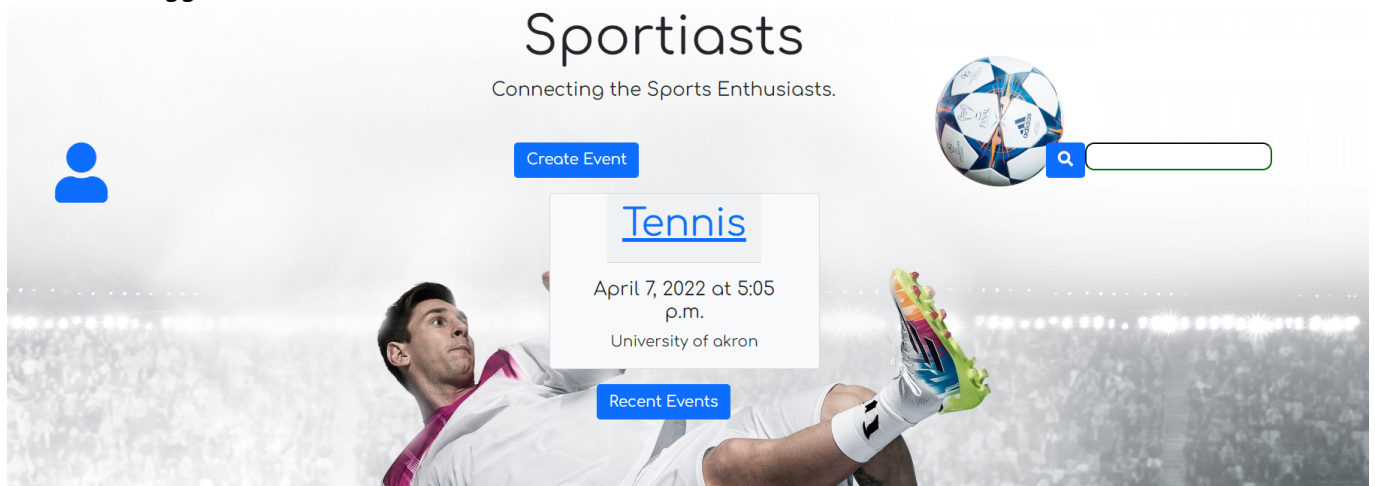
## Log In

This page lets the users to log in to their **Sportiasts** account. It asks for the username/password with the user and if the username/password combination matches the database, the user is logged in to their account.

A login form with a light gray background. At the top, the word "Login" is centered in a large, dark gray font. Below it, there are two input fields. The first is labeled "Username:" and the second is labeled "Password:". Both labels are in a dark gray font. The input fields are white with rounded corners and a thin border. Below the password field, there is a blue button with the word "Login" in white text.

## Homepage after logged in

Once the user is logged in to their account, they can view and/or join and create events. The **Homepage** after the user is logged in looks as below:



### Recents Events

The **Recent Events** tab in the homepage is to display all the past events so that the past events do not make the homepage crowded.

# Sportiasts

Archived Events.

[Home](#)

## [Cricket Competition](#)

May 5, 2021 noon  
Beni, Nepal  
It was 328 days ago

## [Table Tennis](#)

Sept. 17, 2021 noon  
Rec Center, University of Akron  
It was 193 days ago

## [Ping Pong](#)

Sept. 18, 2021 noon  
Akron downtown area  
It was 192 days ago

## [Biking](#)

Sept. 23, 2021 3:30 p.m.  
Akron downtown area  
It was 187 days ago

## [Volleyball](#)

Sept. 24, 2021 6 p.m.  
Akron downtown area  
It was 186 days ago

## [Football](#)

Sept. 30, 2021 3:20 p.m.  
Amsterdam, Netherlands  
It was 180 days ago

## [Check](#)

Oct. 28, 2021 3:03 p.m.  
Beni, Nepal  
It was 152 days ago

## [No Name](#)

Oct. 28, 2021 noon  
Beni, Nepal  
It was 152 days ago

## [Test](#)

Oct. 29, 2021 12:02 p.m.

## [Check - 1](#)

Oct. 29, 2021 3:03 p.m.

## [Swimming](#)

Oct. 29, 2021 noon

## [Check - 2](#)

Oct. 30, 2021 3:03 p.m.

## Profile

Apart from displaying current events, the homepage after the user is logged in displays the profile icon on the top left of the home page. The profile icon leads to the user's profile where the user can see all the events they have created. The **Log Out** tab is also present in the **Profile** page. The **Log Out** button logs the user out of their account for the current session. The **notification** bell on the user's profile notifies the users about the events that are in the near future. In the current version of the application, the notification button gives the notification for the events that are scheduled in the next 7 days.

[Home](#)

Name:

Email: ys96@uakron.edu



[Log Out](#)



## [Cricket Competition](#)

May 5, 2021 noon  
335 days ago.  
Beni, Nepal

## [Football](#)

Sept. 30, 2021 3:20 p.m.  
187 days ago.  
Amsterdam, Netherlands

## [Biking](#)

Sept. 23, 2021 3:30 p.m.  
194 days ago.  
Akron downtown area

## [Test](#)

Oct. 29, 2021 12:02 p.m.  
158 days ago.  
University of akron

## [Swimming](#)

Oct. 29, 2021 noon  
158 days ago.  
Beni, Nepal

## [Football - 2](#)

Nov. 20, 2021 2:03 p.m.  
136 days ago.  
Beni, Nepal

## [Football - 3](#)

Nov. 27, 2021 2:03 p.m.  
129 days ago.  
Beni, Nepal

## [Tennis](#)

April 7, 2022 5:05 p.m.  
2 days later.  
University of akron

## Search

The search button lets users search for the event with the event name, event time, or the venue. So, the user can search all the events that are happening in a certain place, or in a certain time, or search for a particular event type.

An exmple of search result for the location specific search:

## Results for akron

### Tennis

April 7, 2022

University of akron

### Test

Oct. 29, 2021

University of akron

### Volleyball

Sept. 24, 2021

Akron downtown area

An example of search result for the event specific search:

# Results for Football

## Football - 4

Dec. 4, 2021

Beni, Nepal

## Football - 3

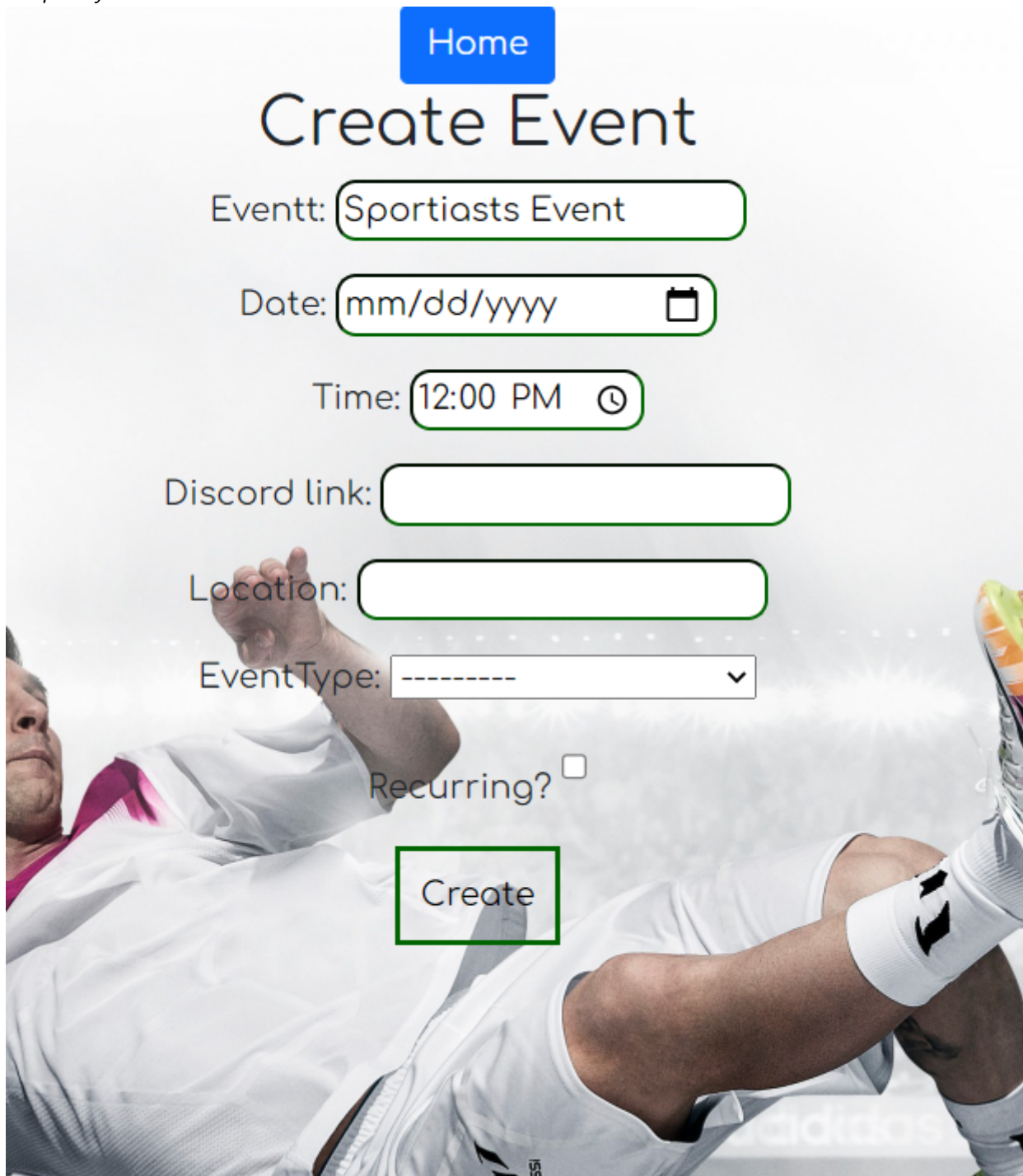
Nov. 27, 2021

Beni, Nepal

## Create Event

This page lets users create new event. The fields that user should enter are **Event Title**, **Date**, **Time**, **Discord Link** (if any), **Location**, **Event Type**, and **Recurring?**. The event title is the name of the event. The date field lets users to either enter a date from keyboard or choose from a calendar. The time is the time of the event. If a user wishes to include a discord link to the event, this page lets user do that. The location field is the venue of the event. The event type field in this page is the drop down menu from where user can select an option from *Indoor Event*, *Outdoor Event*, *Water event*, *Adventurous Event*, or *Other*. A user can also make the event recurring for a certain number of time in a certain number of days. If a user checks *Recurring?* field, the user is asked to enter more data about the frequency of the event. The user should enter the number of days that the event is recurring in, and the number of times the event is happening. For example, if I wish to create an football event for 4 weeks regularly, then my entries should be 7 for the *Recurr in* field and 4 for the

Frequency field.

A screenshot of a 'Create Event' form overlaid on a background image of a soccer player in a white jersey. The form includes a blue 'Home' button at the top, followed by the title 'Create Event'. Below the title are several input fields: 'Eventt:' with the value 'Sportiasts Event', 'Date:' with a placeholder 'mm/dd/yyyy' and a calendar icon, 'Time:' with the value '12:00 PM' and a clock icon, 'Discord link:', 'Location:', and 'EventType:' with a dropdown menu showing '-----'. There is also a 'Recurring?' checkbox which is unchecked. At the bottom of the form is a green 'Create' button.

Home

## Create Event

Eventt: Sportiasts Event

Date: mm/dd/yyyy

Time: 12:00 PM

Discord link:

Location:

EventType: -----

Recurring? ☐

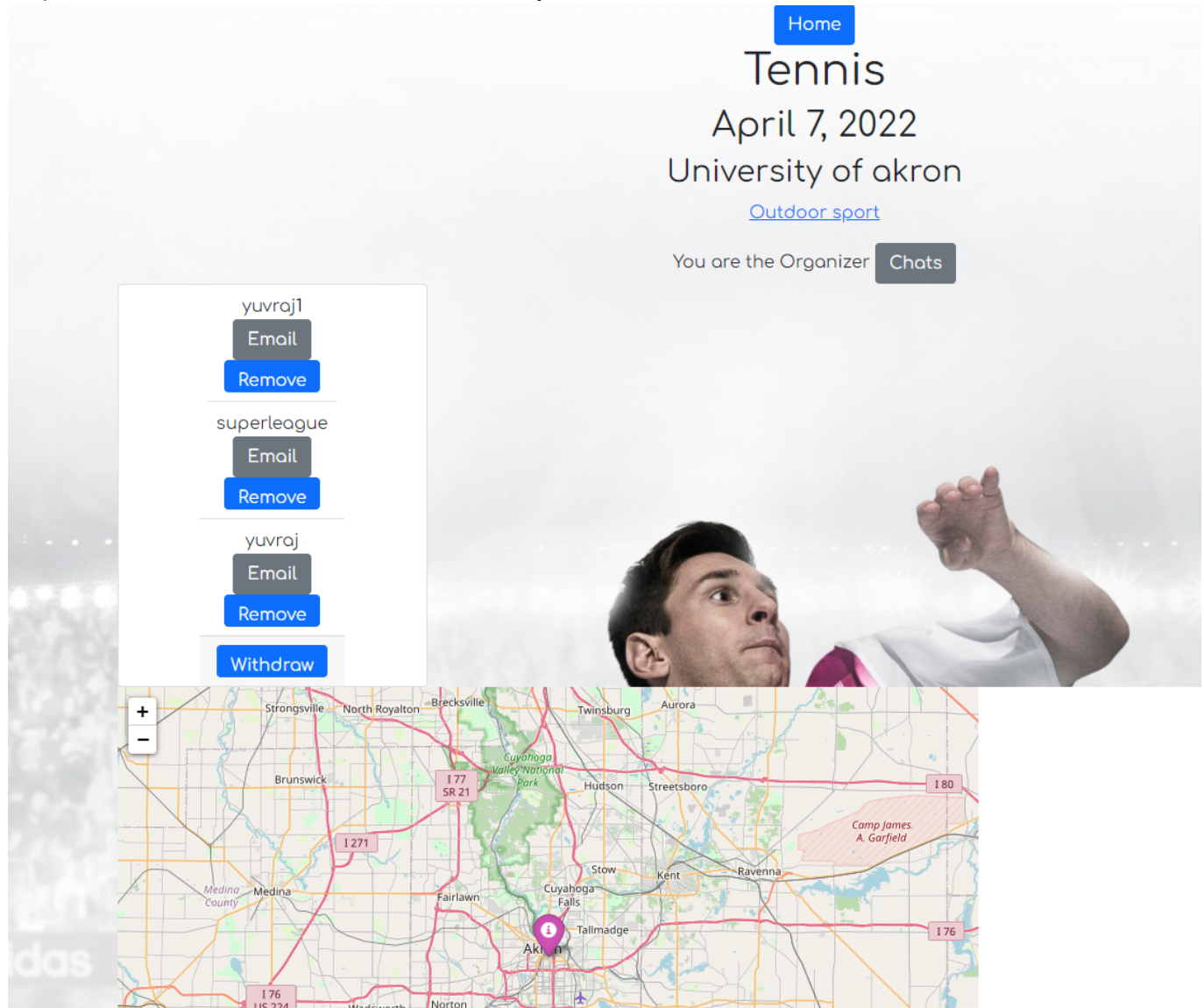
Create

## Event Page

This page is the main page for the event information. This page displays all the information related to the event and everything that the user needs to know about the event. It displays the event name, event time, event venue, event type, and the organizer of the event at the top of the page. On the left side of the page, the participants of the events are displayed. An organizer of the event can remove any participant from the event. An user who has joined the event can withdraw themselves from the event. This page also displays the



map to the event location which makes users easy to find the event venue.



## Chats

This is the group chat feature of the certain event. There is one group chat for one event where the joined users and the organizer have access too. The group chat page displays the name of the sender, the message that was sent, the date when the message was sent, and the time when the message was sent. Any users who have joined the event can see or send a message to the group chat.

yuvraj:Hello participants, ready?, April 1, 2022, 6:58 p.m.

yuvraj1:Yes, I am ready., April 1, 2022, 6:58 p.m.

superleague:I am ready too, April 1, 2022, 6:58 p.m.

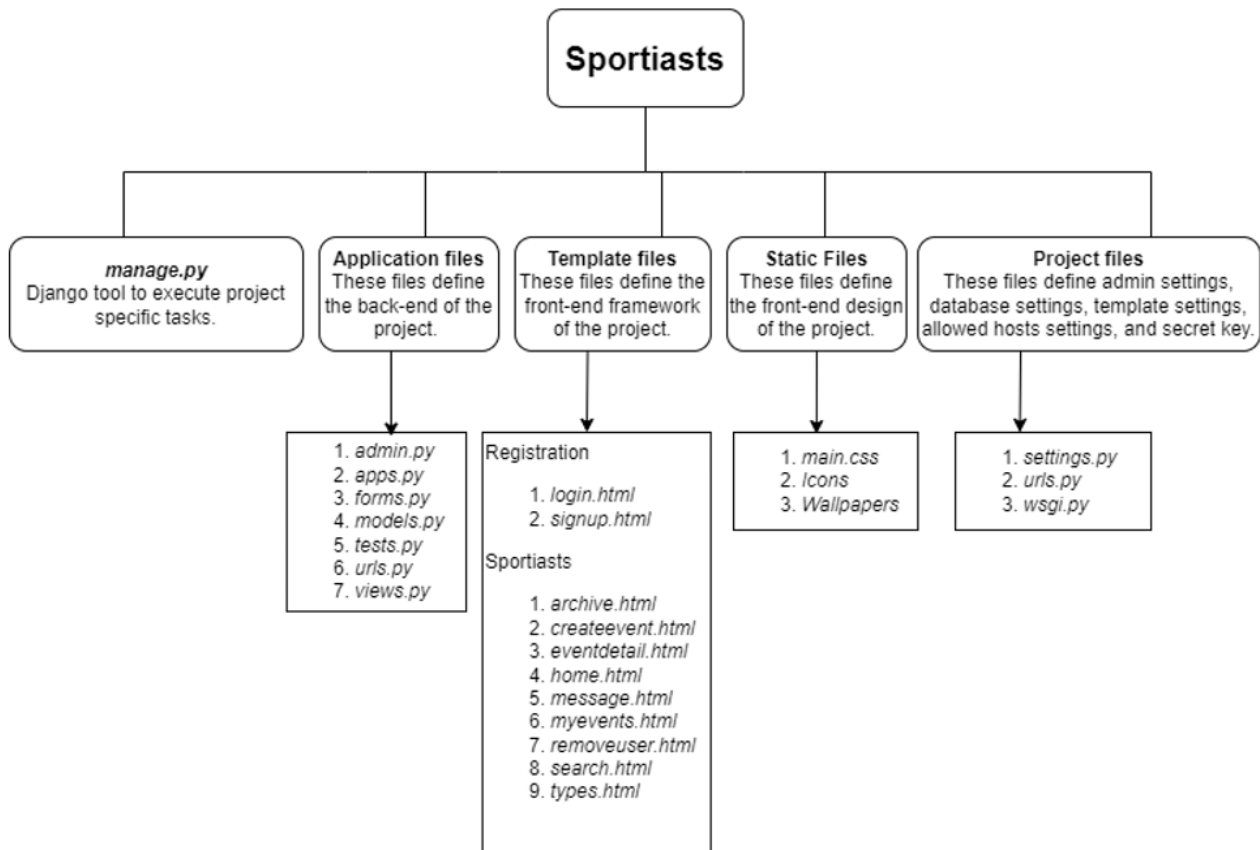
Message:

---

# Implementation

---

As previously mentioned, the programming languages and tools used for this project are HTML, CSS, React.JS, Django, and PostgreSQL. Different type of files created and wrote to make this project are designed and connected in such a way that different directories contain similar type of files. The main project directory contains different type of files such as python files, HTML files, CSS files, images, etc. Therefore, a specific hierarchy of files storage was needed to store related files in their respective directory. The design of the files storage can be explained by the chart below:



## manage.py

It is one of the tools that a Django provides when starting a new project. This tool is used to execute project specific tasks such as - starting a new app within a project, running the development server, running tests, etc (docs.djangoproject.com, 2022). In this project, this tool defines the sportiasts application and runs the server when starting the project.

```
#!/usr/bin/env python3
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'enthusportssettings')
```

```

try:
    from django.core.management import execute_from_command_line
except ImportError as exc:
    raise ImportError(
        "Couldn't import Django. Are you sure it's installed and "
        "available on your PYTHONPATH environment variable? Did you "
        "forget to activate a virtual environment?"
    ) from exc
execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

## Project Files

These are the project-level files that define entire web application. The files under this category define admin settings for django, URLs, database settings, template settings, allowed hosts settings, and secret key. The project-level files created and defined on this project are as follows:

- *settings.py*

This tool defines the most sensitive information of the whole web application. It contains information about the database settings, database admin login credentials, production secret key, allowed hosts information, authorized password validators, and other project settings. The basic level definition of the file is as below:

```

"""
Django settings for sportiasts project.

Generated by 'django-admin startproject' using Django 2.2.18.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '*****'

# SECURITY WARNING: don't run with debug turned on in production!

```



```

DEBUG = True

ALLOWED_HOSTS = ['*****']

# Application definition

INSTALLED_APPS = [
    '*****',
    '*****',
    '*****',
    '*****',
    '*****',
    '*****',
    '*****'
]

MIDDLEWARE = [
    '*****',
    '*****',
    '*****',
    '*****',
    '*****',
    '*****',
    '*****',
    '*****'
]

LOGIN_REDIRECT_URL = '*****'
LOGOUT_REDIRECT_URL = '*****'
ROOT_URLCONF = '*****'
TEMPLATES_DIRS = os.path.join(BASE_DIR, 'templates')
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'enthusports.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',

```

```

        'NAME': '*****',
        'USER': '*****',
        'PASSWORD': '*****',
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = '*****'

TIME_ZONE = '*****'

USE_I18N = *****

USE_L10N = *****

USE_TZ = *****

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_ROOT = '*****'

STATIC_URL = '*****'

STATICFILES_DIRS = ( os.path.join('static'), )

```

- *urls.py*

This file handles the application URLs. The `urlpatterns` list routes URLs to views.

```
"""sportiasts URL Configuration

The `urlpatterns` list routes URLs to views.
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path('', include('sportiasts.urls'))
]
```

- *wsgi.py*

This file is basically used for deployment only rather than development, except if the developer is using docker. It is used as an interface between application server to connect with django or any python framework which implements wsgi specified by python community.

```
"""
WSGI config for sportiasts project.
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'enthusports.settings')

application = get_wsgi_application()
```

## Application Files

- *admin.py*

This file defines the admin page of the project. The django admin can see and edit different database entries from the admin page. This file is used to define and create the views of the admin page.

```
from django.contrib import admin
from .models import *Model Names*

# define the view of admin for the event details
class EventAdmin(admin.ModelAdmin):
    list_display = ('eventtt', 'slug', 'date', 'organizer')
    search_fields = ['eventtt', 'date',]
    prepopulated_fields = {'slug': ('eventtt',)}

admin.site.register(Events, EventAdmin)
admin.site.register(EventType)
admin.site.register(Message)
```

- *apps.py*

This file defines the name of the application in the project.

- *forms.py*

This file handles the details of the different forms used in the application. This project lets users fill up different kind of forms such as registration form (sign up), event details form, and so on. The classes defined in this file help developers to add extra features to the forms such as adding calendar widget to the date field, adding placeholders, etc. The example of class EventForm defined in this file for the application's event details form is as below:

```
class EventForm(forms.ModelForm):
    class Meta:
        model = Events
        fields = (
            "eventtt", "date", "time", "discord_link", "location", "EventType", "recur_in", "re
            cur_man")
        widgets = {
            'date': forms.DateInput(attrs={'type': "date"}),
            'time': forms.TimeInput(attrs={'type': "time"}),
            'discord_link': forms.URLInput(attrs={'required': "false"}),
            'recur_in': forms.NumberInput(attrs={"placeholder": "In how many
            days should the event recurr?"}),
            'recur_man': forms.NumberInput(attrs={"placeholder": " Input 0 for
            non recurring events"})
        }
```

- *models.py*

This is one of the application level files which lets developers define models through different classes. The classes in the models file define the features of the database tables. This is where all the attributes

of a database table are defined. An example of event model is as follows:

```
# A model for the events
class Events(models.Model):
    id = models.AutoField(primary_key=True)
    eventtt = models.CharField(max_length=500, default="Sportiasts Event")
    slug = models.SlugField(unique=True, auto_created=True)
    date = models.DateField()
    time = models.TimeField(default="12:00:00")
    location = models.CharField(max_length=40)
    discord_link = models.URLField(null=True)
    EventType = models.ForeignKey(EventType,
on_delete=models.CASCADE, null=True, related_name="EventType")
    organizer = models.ForeignKey(User, on_delete=models.CASCADE, null=True,
related_name='Organizer',)
    player = models.ManyToManyField(to=User, related_name="players")
    recur_in = models.IntegerField()
    recur_man = models.IntegerField()

    def save(self, *args, **kwargs):
        self.slug = slugify(self.eventtt)
        super(Events, self).save(*args, **kwargs)

    def __str__(self):
        return self.eventtt

    @property
    def is_today(self) -> bool:
        return date.today() == self.date

    @property
    def is_due(self):
        return date.today() > self.date

    @property
    def diff(self):
        if date.today() > self.date:
            return (date.today() - self.date).days
        else:
            return (self.date - date.today()).days
```

Different methods to get event information from the stored event data are defined in this file as shown above.

- *tests.py*

In this file, the test cases for the application are written and run through *manage.py*.

- *urls.py*

This file handles the URLs of the application only. For example, it routes the application URL `application_path/signup` to the signup page defined in `views.py`. This file uses `urlpatterns` to route application URLs to their respective views.

```
urlpatterns = [
    path('signup/', views.SignUpView.as_view(), name='signup')
]
```

- *views.py*

This file defines the display or view of the pages in the application. The views method or classes created in this file load a `html` template with predefined django function `loader.get_template()`. This features of the page views are also defined in this file. For example, the search page displays the filtered result in the reversed date order (latest to oldest). Such features of the page views are also defined in this file. The views methods return the `HttpResponse`. As an example of the views definitions, `search` method is as below:

```
def Search(request):
    template = loader.get_template('sportiasts/search.html')
    query = request.GET['query']
    events = models.Events.objects.filter(
        eventt__icontains=q).order_by('date').reverse() |
models.Events.objects.filter(
    location__icontains=q).order_by('date').reverse() |
models.Events.objects.filter(
    EventType__title__icontains=q).order_by('date').reverse()
    print(events.count())
    if events.count()==0:
        empty=True
    else:
        empty = False
    return HttpResponse(template.render({'events':events, 'query':query,
'empty':empty},request))
```

In `views.py`, the classes are defined for the views that do not return `HttpResponse`. The classes for the views define how the different models or forms defined already in the application are viewed. An example of class definition for the event details view is as:

```
class EventDetailView(generic.DetailView):
    model = models.Events
    template_name='sportiasts/eventdetail.html'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        #make map
        geolocator = Nominatim(user_agent='sportiasts')
        location= self.get_object().location
```

```

full_location= geolocator.geocode(location)
#full location has (address,(lat,long))
m=folium.Map(width=800, height=500, location=full_location[1])
folium.Marker(full_location[1], tooltip=location,
icon=folium.Icon(color='purple')).add_to(m)
m=m._repr_html_()
#add map to context
context["fol_map"]= m
context["players"] = self.get_object().player.all()
print(self.request.user in context['players'])
if self.request.user in context['players']:
    context['join']=False
else:
    context['join']=True
return context

def dispatch(self, request, *args, **kwargs):
    print(self.http_method_names)
    if request.method.lower() in self.http_method_names:
        handler = getattr(self, request.method.lower(),
self.http_method_not_allowed)
    else:
        handler = self.http_method_not_allowed

    if self.request.GET.get('join'):
        print(request.user)
        print(self.get_object().player.all())
        self.get_object().player.add(request.user)

    elif self.request.GET.get('withdraw'):
        self.get_object().player.remove(request.user)
    return handler(request, *args, **kwargs)

```

## Template Files

The template files are the HTML files. Django uses {%\_\_%} format to generate the final HTML. It processes plain text from variables and functions, does text replacements, allows conditional evaluations and loops (docs.djangoproject.com, 2022).

A HTML file called `base.html` is written as the basic HTML file for other template files. This file contains the most basic features of the web application like application icon and application title. All other template files extend this `base.html` file.

The registration directory contains the HTML files for sign up and log in. The files are named `signup.html` and `login.html`. These file define how the front end framework of the signup and login pages.

The Sportiasts directory contain HTML files for other web application pages such as:

- home.html
- myevents.html
- createevent.html

- eventdetail.html
- archive.html
- message.html
- removeuser.html
- search.html
- types.html

These HTML files provide the framework for how the front-end should be displayed in each web application page. Each HTML file account for one application page. Django provides template engine for HTML which allows for conditional evaluations and loops.

For instance, for search functionality of the application, `search.html` file contains conditional evaluations and loops to check if the search returned no result. In such case, "No result found" message is displayed. In a case that more than one result is return, the loop is used to display all the events. Such template engine features of django for HTML files can be seen below in `search.html` file.

```
{% extends 'base.html' %}
{% block title %}Search results for {{q}}{% endblock title %}

{% block content %}
<h1>
    {% if empty == True %} No results found for {{q}} {% else %} Results for {{q}}
    {% endif %}
</h1>

{% if user.is_authenticated %}
<div class="events">
    {% for event in events %}
    <div class="card text-dark bg-light mb-3" style="max-width: 18rem">
        <div class="card-header">
            <h1><a href="{% url 'eventdetail' event.slug%}">{{event.eventtt}}</a></h1>
        </div>
        <div class="card-body">
            <h5 class="card-title">{{event.date}}</h5>
            <p class="card-text">{{event.location}}</p>
        </div>
    </div>
    {% endfor %}
</div>

    <h1> {% if empty2 == True %} {%endif%}{% if empty2 == False %} If you mean
    {{q2}}</h1> {% endif %}

    <div class="events">
    {% for event in events2 %}
    <div class="card text-dark bg-light mb-3" style="max-width: 18rem">
        <div class="card-header">
            <h1><a href="{% url 'eventdetail' event.slug%}">{{event.eventtt}}</a></h1>
        </div>
        <div class="card-body">
            <h5 class="card-title">{{event.date}}</h5>
```



```

        <p class="card-text">{{event.location}}</p>
    </div>
</div>
{% endfor %}
</div>

{% else %}
<div class="events">
    {% for event in events %}

        <div class="card text-dark bg-light mb-3" style="max-width: 18rem">
            <div class="card-header"><h1>{{event.eventt}}</h1></div>
            <div class="card-body">
                <h5 class="card-title">{{event.date}}</h5>
                <p class="card-text">{{event.location}}</p>
            </div>
        </div>
    {% endfor %}
</div>

<h1> {% if empty2 == True %} ""{% else %} If you mean {{q2}}</h1>
<div class="events">
    {% for event in events2 %}

        <div class="card text-dark bg-light mb-3" style="max-width: 18rem">
            <div class="card-header"><h1>{{event.eventt}}</h1></div>
            <div class="card-body">
                <h5 class="card-title">{{event.date}}</h5>
                <p class="card-text">{{event.location}}</p>
            </div>
        </div>
    {% endfor %}
</div>

{% endif %}
{% endif %}
{% endblock content %}

```

## Static Files

These files are the static files of the application. This directory includes css files, application icon image, application background image, and so on.

The css file provides design and decoration to the template files (HTML files). For this version of the project, the very basic css features are implemented to keep user-interface as simple as possible.

---

## Results and Discussions:

---

The current version of the project **Sportiasts** includes only the basic features needed to complete the first version of the application. This application lets people connect and communicate with each other who have

similar sport interests.

The application displays present, future, and past events under different categories sorted by their event date. The users can search for the sport event happening in any location and filter the results by the date. This is one of the most important features that is needed for people to find their sport of interest. Also, the users joined in a same event can have group chat to discuss further about the event. This enhances the networking and communication among the people of similar sport interest. The application allows users to create their own event and invite people to join the event. The organizer of the event has a right to remove the user from their event and the user can withdraw themselves from the event that they have joined.

Another most important feature of the application is that the application displays the event location in a map. This helps users to find their event location even if they are unfamiliar about the venue. The application sends notification about the events that are happening in the near future so that the users don't forget or miss their event. These basic features of the application complete the first version of the application.

There are to-do lists that can be implemented on this project to enhance the usage and reach of the project to a common public. These are the future works for upcoming versions of the projects. Some of them are as follows:

- Track user's location (with user's permission) to get better results to their sport event search.
- Let users follow each other.
- Let users rate the event they joined and attended.
- Display weather information in the event location on the event date.
- Let users share the pictures/videos of the event they attended.
- Categorize the events by kids/ adults/ family/ professionals.

The implementation of these features in the upcoming versions will definitely enhance the strength of the project. However, factors like security and data storage will be the first priority for future designs and implementations.

---

## Conclusions:

---

This version of the project is the first stepping stone of the idea. During this project, I have been able to explore many features of django, PostgreSQL, and different front-end tools. Django is one of the smoothest and the most powerful tools in developing web applications. It is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is free and open source. Some of the best features of Django are that the django is very fast, secure, and scalable.

Another very powerful database system learned from this project is PostgreSQL. It is a powerful, open source object-relational database system which is reliable and robust. It safely store and scale the most complicated data workloads. PostgreSQL runs on all major operating systems and has many powerful add-ons.

Apart from these two new tools used in the project, other common web development tools such as HTML, CSS, Javascript have been extensively explored and implemented on this project.

From the motivation idea of this project to completing the first version of the project, many challenges have been faced and overcome. This motivation of learning new tools and implementing them on the project will lead to higher versions of the application. In summary, this application helps a sport community to connect with each other, communicate, and expand their sport network while making them physically and mentally healthy by providing a platform to participate in different kinds of sporting event.

---

---

# References

---

Airfocus.com. 2022. Glossary. [online] Available at: <https://airfocus.com/glossary/>.

Boduch, A. and Derks, R., 2020. React and React Native: A complete hands-on guide to modern web and mobile development with React. js. Packt Publishing Ltd.

Condor Ferries. 2022. 70+ U.S. Tourism & Travel Statistics (2020-2021). [online] Available at: <https://www.condorferries.co.uk/us-tourism-travel-statistics>.

Developer.mozilla.org. 2022. HTML: HyperText Markup Language | MDN. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTML>.

Developer.mozilla.org. 2022. JavaScript | MDN. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

Docs.djangoproject.com. 2022. Django documentation | Django documentation | Django. [online] Available at: <https://docs.djangoproject.com/en/4.0/>.

Forcier, J., Bissex, P. and Chun, W.J., 2008. Python web development with Django. Addison-Wesley Professional.

Graft, A., 2022. Travel and Tourism Statistics: The Ultimate Collection. [online] Blog.accessdevelopment.com. Available at: <https://blog.accessdevelopment.com/tourism-and-travel-statistics-the-ultimate-collection>.

Momjian, B., 2001. PostgreSQL: introduction and concepts (Vol. 192). New York: Addison-Wesley.

Statista. 2022. Percentage of U.S. Americans engaged leisure/sports activities 2020 | Statista. [online] Available at: <https://www.statista.com/statistics/189548/daily-engagement-of-the-us-population-in-leisure-and-sports/#:~:text=This graph depicts the daily,and sports activities in 2019..>