Turf Booking Website

Problem Statement

The aim of this website is to take turf bookings online. Turf Bookings are typically carried out over phone calls, however our project is intended so that people can book their turf slots remotely. We have implemented this project using Django Framework.

Explanation

Models

The website has been divided in 2 parts. One for those who are the users of the website i.e. those who want to book turf slots and other for the turf owners/managers who have a turf and want to put it up for bookings.

In order to achieve this we created 2 models, one for the **Users** (Model name: Bookie) and other for the **Turfs** themselves.

```
class Turf(models.Model):
    name = models.CharField(max_length=200)
    location = models.TextField()
    password = models.CharField(max_length=200)
    image = models.ImageField(upload_to='images/', blank=True)

    def __str__(self):
        return self.name
```

```python
class Bookie(models.Model):
    name = models.CharField(max_length=200)
    password = models.CharField(max_length=100)
    phone_number = models.IntegerField(max_length=10)

    def __str__(self):
        return self.name
```

The third and the final model is of a slot. Each **Slot** would exist between 1 turf and 1 user for a given timing. This timing would be in durations of 1hr each.

```python
class Slot(models.Model):
    turf = models.ForeignKey(Turf, default=1, on_delete=models.SET_DEFAULT)
    timing = models.TimeField(default=timezone.now())
    date = models.DateField()
    user = models.ForeignKey(Bookie, default=1, on_delete=models.SET_DEFAULT)
    book_timing = models.DateTimeField(auto_now_add=True)
```

URLs

The URLconf we have created for out App is as follows:

```
urlpatterns = [
    path('', views.home, name="home"),
    path('/create_turf', views.create_turf, name="index"),
    path('user', views.create_user, name="set_user"),
    path('user_login', views.user_login, name="user_login"),
    path('listings/<str:pk>/', views.listings, name="listings"),
    path('listings/<str:upk>/<str:tpk>', views.new_slot, name="new_slot"),
    path('turf_detail/<str:pk>', views.turf_detail, name="turf_detail"),
    path('login_turf', views.login_turf, name="turf_login"),
    path('signup_turf', views.create_turf, name="set_turf"),
]
```

Here, "**set_user"** is used to create a new user. "**listings"** has been passed a parameter pk which is the Primary Key of the user that has logged in. This is required as when the user clicks on book for any of the listings, we know which user is attempting to book a given turf. Following the same reasoning **"new_slot"** takes is 2 parameters, i.e. upk, tpk i.e. the Primary Key for User and Turf respectively so that we know which User (Bookie) is booking which Turf which can be easily identified by their pk values. **"turf_detail"** will show the details of all the bookings a manager/owner has logged in using the Turf Account.

Views

We haven't used the AbstractBaseUser and thus we couldn't use the authenticate method and thus we have to create a way out so as to restrict un-authorized access to the pages.

Thus we have used: 2 global variables which are set to False every time the User or the Turf Logs out. This is to ensure that no one is able to access the pages by manually typing out the URLs.

Log Out in our website is set to Home but it can be changed if features are added, however in our case it works just fine.

```
userLoggedIn = False
turfLoggedIn = True
```

```
def home(request):
    global userLoggedIn
    userLoggedIn = False
    global turfLoggedIn
    turfLoggedIn = False
    return render(request, 'turf_booking/home.html')
```

Thus "home" can be replaced with log out but as our website is not that big this works just fine.

In order to create new user, we use a Django Form that renders out to the Frontend and then the form is checked and then if valid saved, i.e. as the form itself is a Model Form an object for the associated model gets created with the entered fields. We also set the UserLoggedIn = True and a new user has been created.

Create User View:

```
def create_user(request):
    form = NewUser()
    if request.method == "POST":
        form = NewUser(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            form.save()
            pk = Bookie.objects.get(name=name).pk
            global userLoggedIn
            userLoggedIn = True
            return redirect('listings', pk)
    context = {
        'form': form,
    }
    return render(request, 'turf_booking/user.html', context)
```

NewUser Form:

```
class NewUser(forms.ModelForm):
    class Meta:
        model = Bookie
        fields = ['name', 'phone_number', 'password']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control col-lg-12'}),
            'password': forms.PasswordInput(attrs={'class': 'form-control col-lg-12'}),
        }
```

For User (Bookie) Login we use LogUser form to take in the username and password. Find the User match the password to the User's Password and if the user is not found we throw an Exception. If the Username and Passwords match we redirect to the listings page.

For User (Bookie) Login:

```
def user_login(request):
    form = LogUser()
    check = False
    if request.method == "POST":
        form = LogUser(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            password = form.cleaned_data['password']
            try:
                user = Bookie.objects.get(name=name)
                if user.password == password:
                    pk = user.pk
                    global userLoggedIn
                    userLoggedIn = True
                    return redirect('listings', pk)
            except Exception:
                check = True
                print("User absent")
    context = {
        'form': form,
        'check': check,
    }
    return render(request, 'turf_booking/user.html', context)
```

Rendering in HTML:

Note that we have used the same template to render the LoginUser Form as well as NewUser and added a new item in context dictionary.

```
<form method = "post" class = "form-control">
    {% csrf_token %}
    {{ form.as_p }}
    <input type = "submit">
    {% if check %}
    <p>
        Check Your Username and Password.
    </p>
    {% endif %}
</form>
<a class="btn btn-primary" href="{% url 'home' %}" role="button">Home</a>
```

Similarly, we have also designed the Create Turf and Login Turf, the only difference is that Turf object has an image parameter and thus with request.POST we will also request request.FILES for the image.

For Listings we have to show all the turfs that are available to be booked:

```
def listings(request, pk):
    context = {
        'turf_list': Turf.objects.all(),
        'upk': str(pk),
        'user': Bookie.objects.get(pk=pk),
        'userLoggedIn': userLoggedIn,
    }
    return render(request, 'turf_booking/listings.html', context)
```

In order to render the slots that have been booked for a turf we have created the “turf_detail” view.

```python
def turf_detail(request, pk):
    try:
        turf = Turf.objects.get(id=pk)
        slot_list = turf.slot_set.all()
        context = {
            'slot_list': slot_list,
            'turf': turf,
            'turfLoggedIn': turfLoggedIn,
        }
        return render(request, 'turf_booking/turf_detail.html', context)
    except Exception:
        print("Non Found")
    context = {
        'slot_list': [],
    }
    return render(request, 'turf_booking/turf_detail.html', context)
```

We find the “slot_set” of the logged in turf and display that slot_list as a table using Django Standard Template, as:

```
<h1 style="text-align: center;">{{ turf.name }}</h1>
    <hr>
<div class = "container">
    <h2 style="text-align: center; padding-bottom: 0px; margin-bottom: 0px;">Dashboard</h2>-
    <div id = "contents">
        <table class = "table">
            <thead class="thead-dark">
                <tr>
                    <th scope="col">Time</th>
                    <th scope="col">User</th>
                    <th scope="col">Phone Number</th>
                </tr>
            </thead>
    {% if slot_list %}
        {% for slot in slot_list %}
            <tbody>
                    <td>{{ slot.timing }}</td>
                      <td>{{ slot.user }}</td>
                    <td>{{ slot.user.phone_number }}</td>
                </tr>
            </tbody>
        <br>
        {% endfor %}
            </table>
    {% else %}
        <h3 style = "text-align: center;">No Slots Booked Yet.</h3>
    {% endif %}
```

In order to create a new slot we created a “new slot” view which would take in the upk (User Primary Key) and tpk (Turf Promary Key), create a slot with a time that take from the user using another form and create a slot object created from this fields and add it in the turf.slot_set and user.slot_set. This is done as the slot is to be connected to a single user and a single turf at a given time.

Also, note that slots that have been created to a date prior than the current date have no use and thus are deleted. Also as a game lasts about 1hr, once a slot has been added, no other slot can be added within 1hr of the added slot. It will give an error that the slot is already booked.

We achieve this by:

```
user = Bookie.objects.get(pk=upk)
turf = Turf.objects.get(pk=tpk)
slot_list = turf.slot_set.all()
for slot in slot_list:
    if slot.date < date.today():
        slot.delete()
```

And

```
if request.method == "POST":
    form = NewSlot(request.POST)
    if form.is_valid():
        time = form.cleaned_data['timing']
        for slot in slot_list:
            td = datetime.combine(slot.date, slot.timing) - datetime.combine(slot.date, time)
            et = divmod(td.total_seconds(), 60)
            print(et)
            if abs(et[0]) < 60:
                check = True
        if check:
            context = {
                'slot_list': slot_list,
                'turf_id': tpk,
                'user_id': upk,
                'form': form,
                'check': check,
                'userLoggedIn': userLoggedIn,
            }
            print('Slot Booked')
            return render(request, 'turf_booking/slot.html', context)
```

Boolean **check** is used to keep a track of whether a slot has been created or not. It is then sent with the context which will help us print out the error message within the HTML Template.

```
<form method = "POST" class = "form-control">
    {% csrf_token %}
    {{ form.as_p }}
    <input type = "submit">
    {% if check %}
    <p>
        Slot Already Booked. Choose Another Slot.
    </p>
    {% endif %}
</form>
<a class="btn btn-primary" href="{% url 'home' %}" role="button">Home</a>
```

Slot Form:

```
class NewSlot(forms.ModelForm):
    class Meta:
        model = Slot
        fields = ['timing', 'date']
        widgets = {
            'date': forms.DateTimeInput(attrs={'class': 'datetime-input'})
        }
```

The only reason to take date and time not as a DateTime input but separately is that entering a DateTime together might be daunting for the user and so we separate it out.

Code

Models.py

```
from django.db import models
from django.utils import timezone


class Turf(models.Model):
    name = models.CharField(max_length=200)
    location = models.TextField()
    password = models.CharField(max_length=200)
    image = models.ImageField(upload_to='images/')


    def __str__(self):
        return self.name


class Bookie(models.Model):
    name = models.CharField(max_length=200)
    password = models.CharField(max_length=100)
    phone_number = models.IntegerField(max_length=10)


    def __str__(self):
        return self.name


class Slot(models.Model):
    turf = models.ForeignKey(Turf, default=1, on_delete=models.SET_DEFAULT)
    timing = models.TimeField(default=timezone.now())
```

```
date = models.DateField()

user = models.ForeignKey(Bookie, default=1, on_delete=models.SET_DEFAULT)

book_timing = models.DateTimeField(auto_now_add=True)
```

views.py

```
from django.shortcuts import render, redirect, HttpResponse,
get_object_or_404

from .forms import NewUser, NewTurf, NewSlot, LogUser, LogTurf

from .models import Turf, Slot, Bookie

from datetime import datetime, date


userLoggedIn = False

turfLoggedIn = True



def create_turf(request):

    form = NewTurf()


    if request.method == "POST":

        form = NewTurf(request.POST, request.FILES)

        if form.is_valid():

            name = form.cleaned_data.get('name')

            location = form.cleaned_data.get('location')

            password = form.cleaned_data.get('password')

            image = form.cleaned_data.get('image')

            obj = Turf.objects.create(

                name=name,

                password=password,

                location=location,

                image=image

            )

            obj.save()

            return redirect(login_turf)
```

```
    context = {
        'form': form,
    }
    return render(request, 'turf_booking/index.html', context)


def login_turf(request):
    form = LogTurf()
    check = False
    if request.method == "POST":
        form = LogTurf(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            password = form.cleaned_data['password']
            try:
                turf = Turf.objects.get(name=name)
                if turf.password == password:
                    pk = turf.pk
                    global turfLoggedIn
                    turfLoggedIn = True
                    return redirect('turf_detail', pk)
            except Exception:
                check = True
                print("User absent")
    context = {
        'form': form,
        'check': check,
    }
```

```
    return render(request, 'turf_booking/login_turf.html', context)



def create_slot(request, pk):
    turf = Turf.objects.get(id=pk)
    slot_list = turf.slot_set.all()
    for slot in slot_list:
        if slot.book_timing.date() != datetime.date.today():
            slot.delete()
    form = NewSlot()
    if request.method == "POST":
        form = NewSlot(request.POST)
        if form.is_valid():
            time = form.cleaned_data['timing']
            check = False
            for slot in slot_list:
                if slot.timing == time:
                    check = True
            if check:
                HttpResponse("Slot Already Booked")
            else:
                slot = form.save()
                turf.slot_set.add(slot)
        return redirect('set_slot', pk)
    context = {
        'slot_list': slot_list,
        'turf_id': pk,
        'form': form,
```

```
        'userLoggedIn': userLoggedIn,
    }
    return render(request, 'turf_booking/slot.html', context)


def create_user(request):
    form = NewUser()
    if request.method == "POST":
        form = NewUser(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            form.save()
            pk = Bookie.objects.get(name=name).pk
            global userLoggedIn
            userLoggedIn = True
            return redirect('listings', pk)
    context = {
        'form': form,
    }
    return render(request, 'turf_booking/user.html', context)


def user_login(request):
    form = LogUser()
    check = False
    if request.method == "POST":
        form = LogUser(request.POST)
        if form.is_valid():
```

```
            name = form.cleaned_data['name']
            password = form.cleaned_data['password']
            try:
                user = Bookie.objects.get(name=name)
                if user.password == password:
                    pk = user.pk
                    global userLoggedIn
                    userLoggedIn = True
                    return redirect('listings', pk)
            except Exception:
                check = True
                print("User absent")
    context = {
        'form': form,
        'check': check,
    }
    return render(request, 'turf_booking/user.html', context)


def listings(request, pk):
    context = {
        'turf_list': Turf.objects.all(),
        'upk': str(pk),
        'user': Bookie.objects.get(pk=pk),
        'userLoggedIn': userLoggedIn,
    }
    return render(request, 'turf_booking/listings.html', context)
```

```
def new_slot(request, upk, tpk):
    user = Bookie.objects.get(pk=upk)
    turf = Turf.objects.get(pk=tpk)
    slot_list = turf.slot_set.all()
    for slot in slot_list:
        if slot.date < date.today():
            slot.delete()
    form = NewSlot()
    check = False
    if request.method == "POST":
        form = NewSlot(request.POST)
        if form.is_valid():
            time = form.cleaned_data['timing']
            for slot in slot_list:
                td = datetime.combine(slot.date, slot.timing) -
datetime.combine(slot.date, time)
                et = divmod(td.total_seconds(), 60)
                print(et)
                if abs(et[0]) < 60:
                    check = True
            if check:
                context = {
                    'slot_list': slot_list,
                    'turf_id': tpk,
                    'user_id': upk,
                    'form': form,
                    'check': check,
                    'userLoggedIn': userLoggedIn,
```

```
                }
                print('Slot Booked')
                return render(request, 'turf_booking/slot.html', context)
            else:
                slot = form.save()
                turf.slot_set.add(slot)
                user.slot_set.add(slot)
                return redirect('new_slot', upk, tpk)
    context = {
        'slot_list': slot_list,
        'turf_id': tpk,
        'user_id': upk,
        'form': form,
        'check': check,
        'userLoggedIn': userLoggedIn,
    }
    return render(request, 'turf_booking/slot.html', context)



def turf_detail(request, pk):
    try:
        turf = Turf.objects.get(id=pk)
        slot_list = turf.slot_set.all()
        context = {
            'slot_list': slot_list,
            'turf': turf,
            'turfLoggedIn': turfLoggedIn,
        }
```

```
        return render(request, 'turf_booking/turf_detail.html', context)
    except Exception:
        print("Non Found")
    context = {
        'slot_list': [],
    }
    return render(request, 'turf_booking/turf_detail.html', context)


def home(request):
    global userLoggedIn
    userLoggedIn = False
    global turfLoggedIn
    turfLoggedIn = False
    return render(request, 'turf_booking/home.html')
```

forms.py

```
from django import forms

from .models import Turf, Slot, Bookie


# Form for creation of new Turf:

class NewTurf(forms.ModelForm):

    class Meta:

        model = Turf

        fields = ['name', 'location', 'password', 'image']

        widgets = {

            'name': forms.TextInput(attrs={'class': 'form-control col-lg-
12'}),

            'location': forms.Textarea(attrs={'class': 'form-control col-lg-
12'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control
col-lg-12'}),

        }



# Form for create of new slot:

class NewSlot(forms.ModelForm):

    class Meta:

        model = Slot

        fields = ['timing', 'date']

        widgets = {

            'date': forms.DateTimeInput(attrs={'class': 'datetime-input'})

        }



# Form for creation of new User:
```

```
class NewUser(forms.ModelForm):

    class Meta:

        model = Bookie

        fields = ['name', 'phone_number', 'password']

        widgets = {

            'name': forms.TextInput(attrs={'class': 'form-control col-lg-
12'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control
col-lg-12'}),

        }



class LogUser(forms.ModelForm):

    class Meta:

        model = Bookie

        fields = ['name', 'password']

        widgets = {

            'name': forms.TextInput(attrs={'class': 'form-control col-lg-
12'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control
col-lg-12'}),

        }



class LogTurf(forms.ModelForm):

    class Meta:

        model = Turf

        fields = ['name', 'password']


        widgets = {
```

```
            'name': forms.TextInput(attrs={'class': 'form-control col-lg-
12'}),
            'password': forms.PasswordInput(attrs={'class': 'form-control
col-lg-12'}),
        }
```

urls.py

```
from django.contrib import admin
from django.urls import path, include
from . import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('', views.home, name="home"),
    path('/create_turf', views.create_turf, name="index"),
    path('user', views.create_user, name="set_user"),
    path('user_login', views.user_login, name="user_login"),
    path('listings/<str:pk>/', views.listings, name="listings"),
    path('listings/<str:upk>/<str:tpk>', views.new_slot, name="new_slot"),
    path('turf_detail/<str:pk>', views.turf_detail, name="turf_detail"),
    path('login_turf', views.login_turf, name="turf_login"),
    path('signup_turf', views.create_turf, name="set_turf"),
]

# For Image:
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
                          document_root=settings.MEDIA_ROOT)
```

admin.py

```
from django.contrib import admin
from .models import Turf, Slot, Bookie

admin.site.register(Turf)
admin.site.register(Slot)
admin.site.register(Bookie)
```

urls.py (Base - Not Created App)

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('turf_booking.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
                          document_root=settings.MEDIA_ROOT)
```

IMPORTANT HTML PAGES (USES DJANGO TEMPLATE LANGUAGE)

MULTIPART FORM

templates/turf_booking/index.html (Turf Creation)

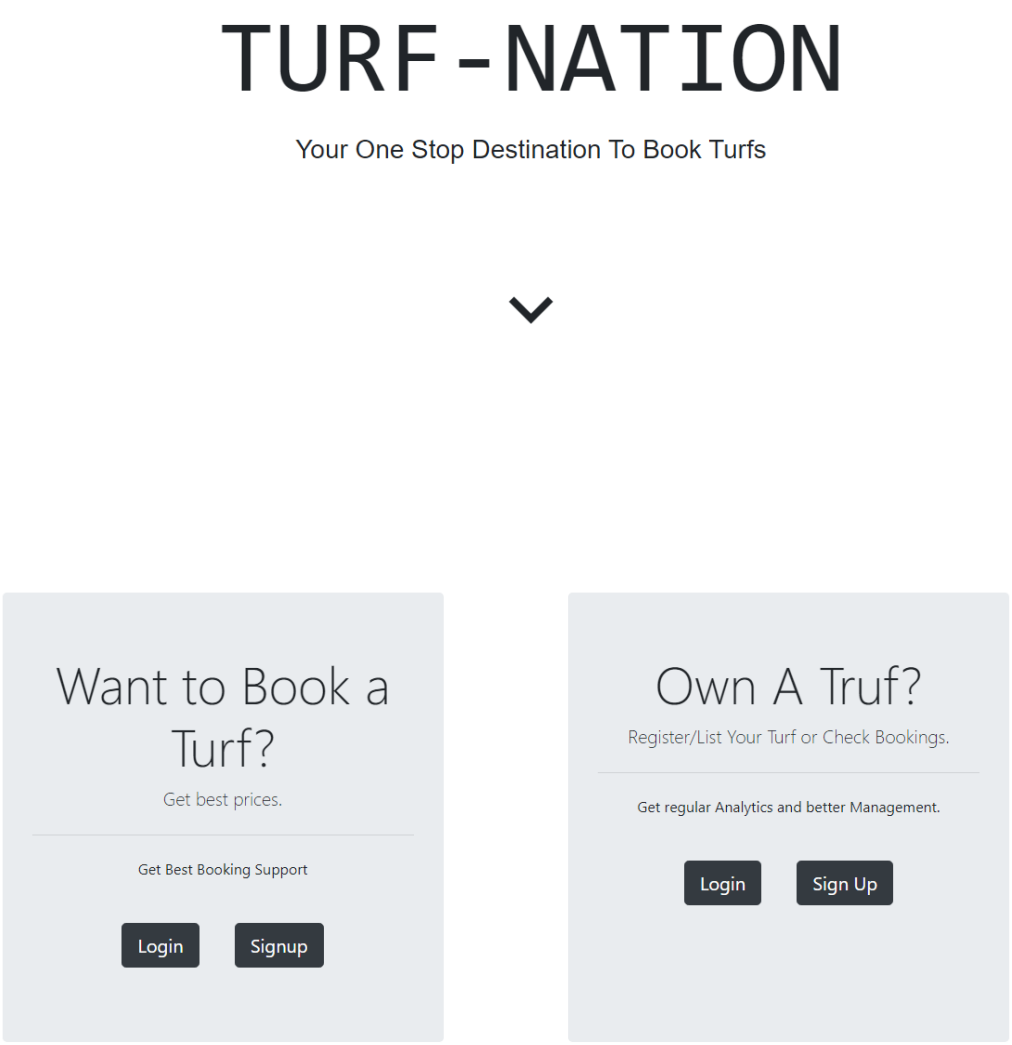
```
<head>

<!— Bootstrap for Styling -->

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

</head>

<body>

<div class = "container">

<h1>Turf Form</h1>

<form class = "form-control" method = "post" enctype="multipart/form-data">

    {% csrf_token %}

    {{ form.as_p }}

    <input type = "submit">

</form>

<a class="btn btn-primary" href="{% url 'home' %}" role="button">Home</a>

</div>

</body>
```

templates/turf_booking/slot.html (Creating Slot)

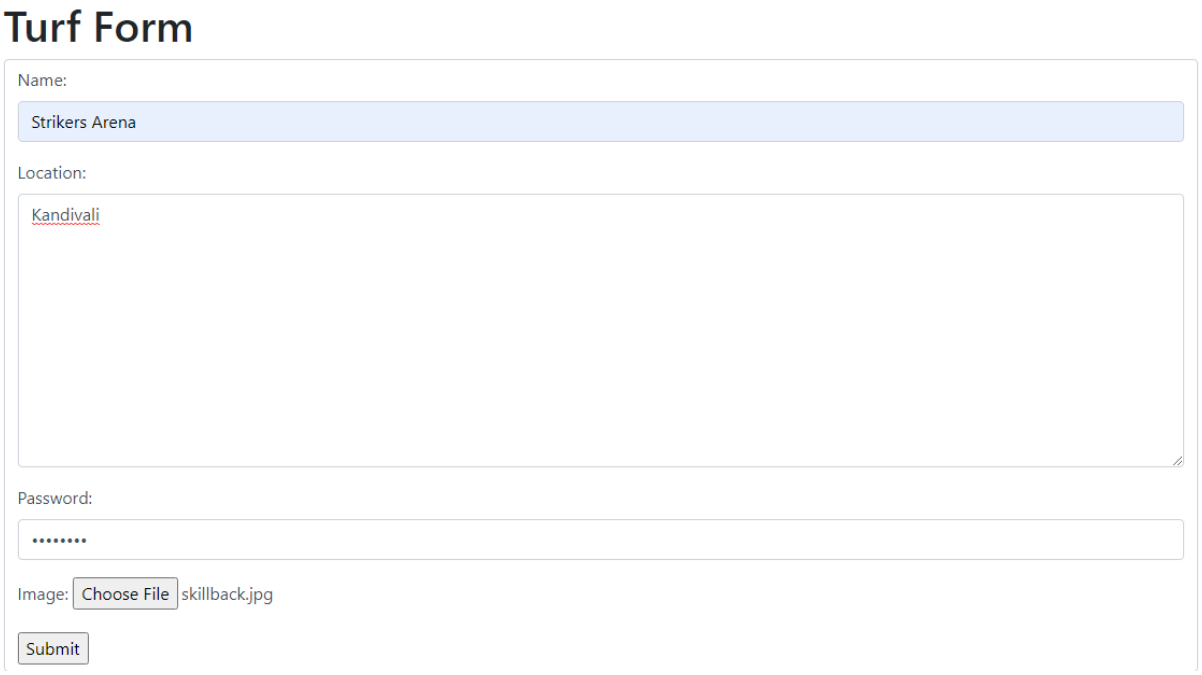
```
<head>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS
6JXm" crossorigin="anonymous">

</head>

<body>

<div class = "container">

    {% if userLoggedIn %}

    <h1>Slot Form</h1>

    {% for slot in slot_list %}

        <p>{{ slot.date}}, {{ slot.timing }}: {{ slot.user }}</p>

    {% endfor %}

    <form method = "POST" class = "form-control">

        {% csrf_token %}

        {{ form.as_p }}

        <input type = "submit">

        {% if check %}

        <p> Slot Already Booked. Choose Another Slot.</p>

        {% endif %}

    </form>

    <a class="btn btn-primary" href="{% url 'home' %}" role="button">Home</a>

    </div>

    {% endif %}

    {% if userLoggedIn == False %}

        <h2>You aren't Authorized to view this page</h2>

    {% endif %}

</body>
```

Output

Home

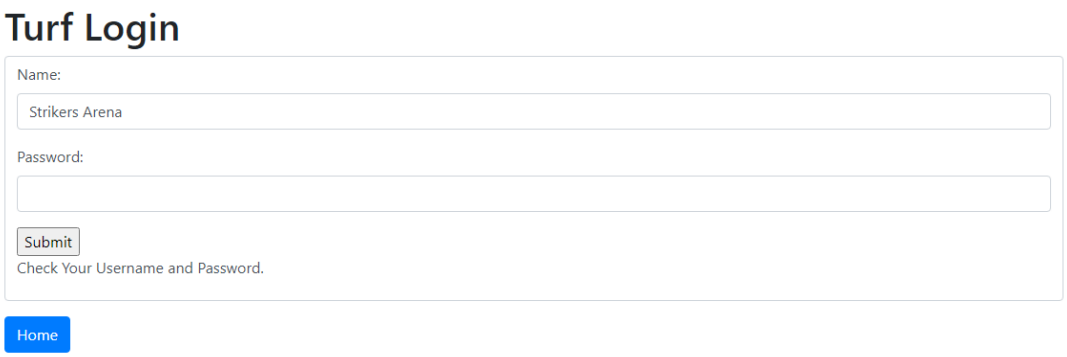

TURF-NATION
Your One Stop Destination To Book Turfs
Want to Book a Turf?
Get best prices.
Get Best Booking Support
Login
Signup
Own A Truf?
Register/List Your Turf or Check Bookings.
Get regular Analytics and better Management.
Login
Sign Up

Turf Sign-Up

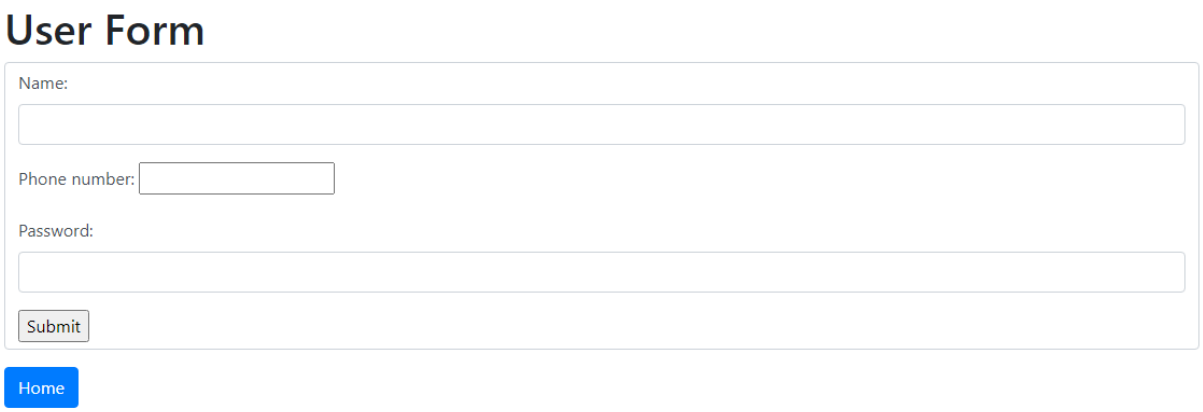


Turf Login

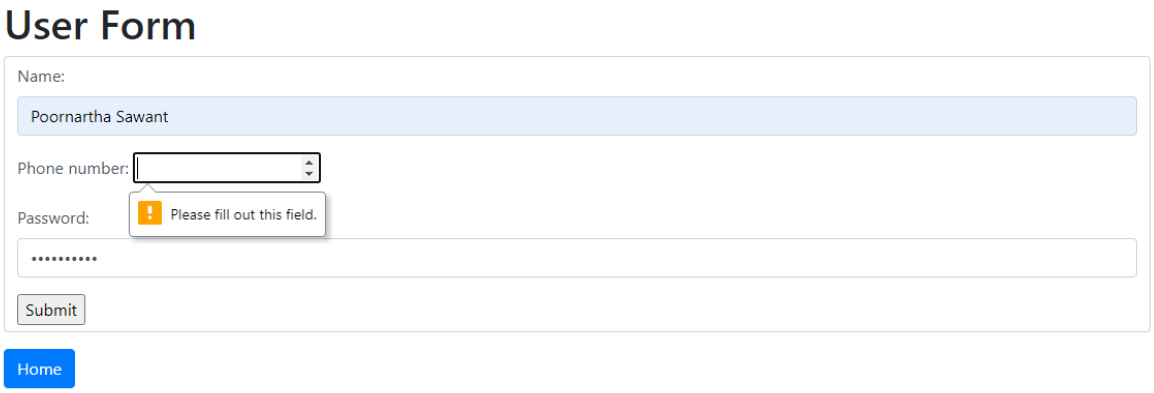
Failed Login



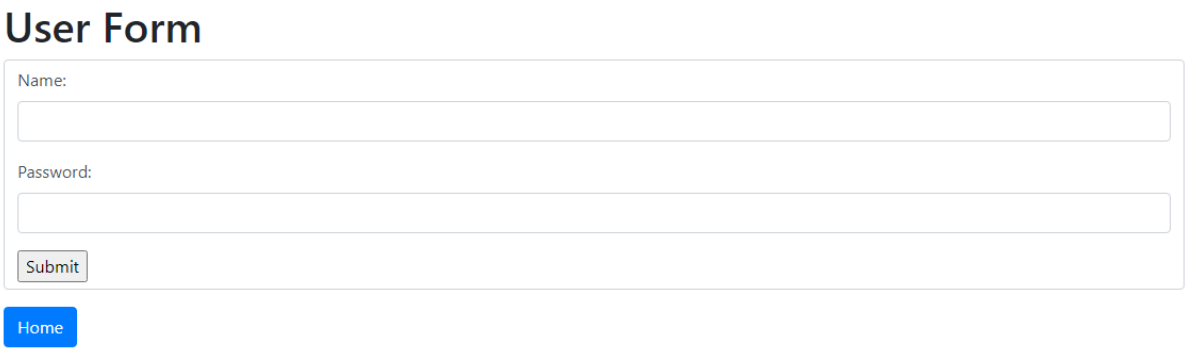
User Sign-Up



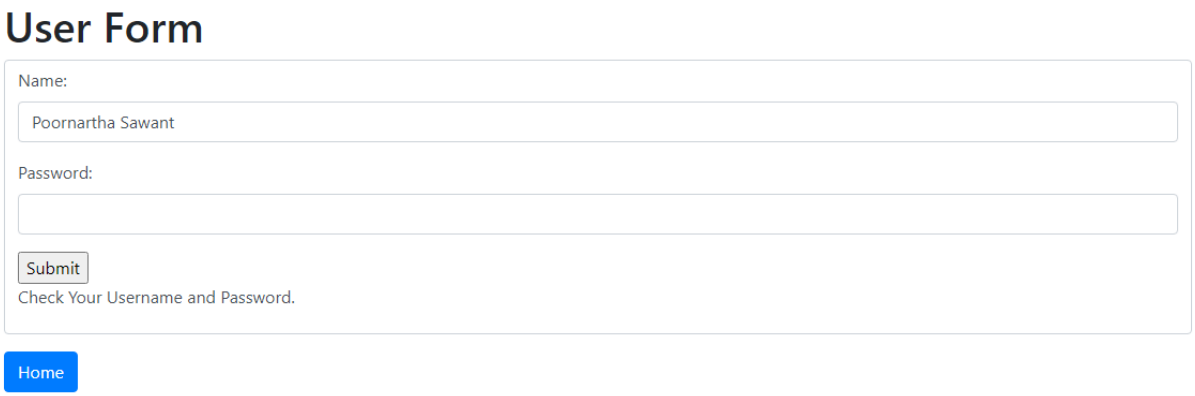
Missing Field



User Login



Failed Login



Listings

Poornartha

Turfs Available For You

Strikers Arena

Kandivali

Book!

Bayern

Borivali

Book!

Slot Booking (After Clicking on Book)

Initial Slot Form

Slot Form

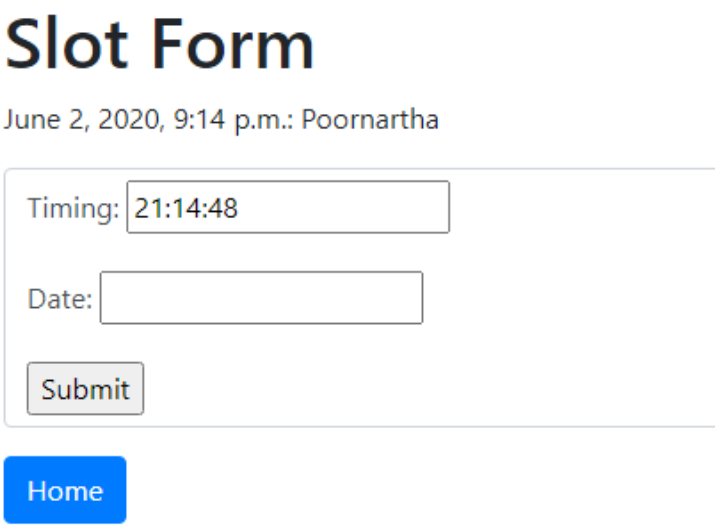
Timing: 21:14:48

Date:

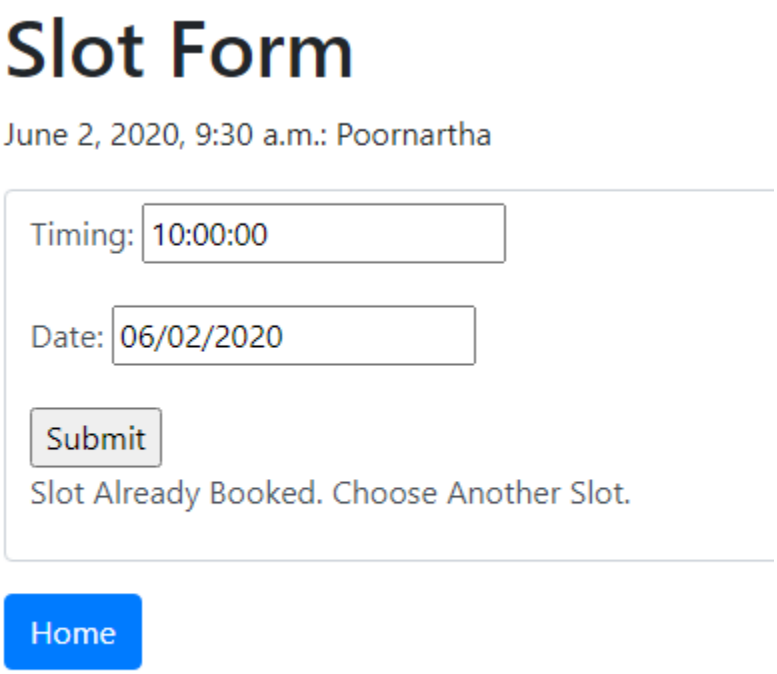
Submit

Home

After Booking a Slot



Booking a Slot Within 1hr of Previously Booked Slot



Turf Dashboard (After Login)

Strikers Arena

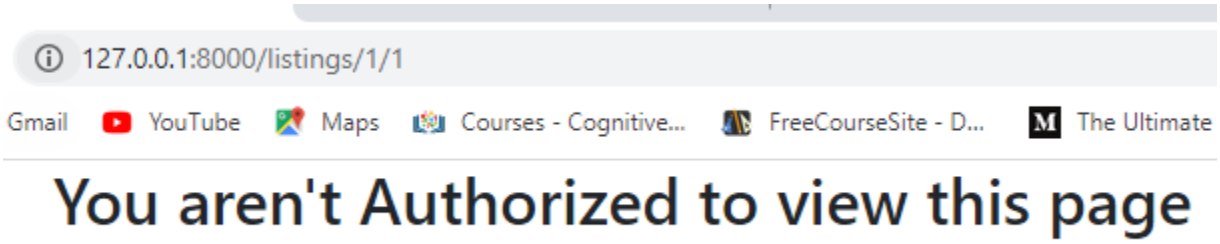
Dashboard

-

Time	Date	User	Phone Number
9:30 a.m.	June 2, 2020	Poornartha	7977925014
11:30 a.m.	June 2, 2020	Shubh	9987206919
12:30 p.m.	June 2, 2020	Arpit	7260458127

Home

Trying to Access by Manually Entering the URL



You aren't Authorized to view this page