admin.py

Register your models here.

```
from django.contrib import admin
```

apps.py

```python
from django.apps import AppConfig


class ManagerConfig(AppConfig):
    name = 'manager'
```

Coach.py

```
from manager.Player import Player

class Coach(Player):

    def __init__(self):
        self.wins = 0
        self.losses = 0

    def getWins(self):
        return self.wins

    def addWin(self):
        self.wins = self.wins + 1

    def getLosses(self):
        return self.losses

    def addLoss(self):
        self.losses = self.losses + 1

    def getRecord(self):
        return self.wins/self.losses

    def setRecord(self, w, l):
        self.wins = w
        self.losses = l
```

Game.py

```
class Game:

    def __init__(self, id, ht=None, at=None, d=0, m=0, y=0, t=None,
        self.id = id
        self.homeTeam = ht
        self.awayTeam = at
        self.scoreHome = 0
        self.scoreAway = 0
        self.forefit = False
        self.complete = False
        self.day = d
        self.month = m
        self.year = y
        self.time = t
        self.location = loc


    def getId(self):
        return self.id


    def getDay(self):
        return self.day


    def getMonth(self):
        return self.month


    def getYear(self):
        return self.year


    def getTime(self):
        return self.time


    def getDatestring(self):
        return self.month + "/" + self.day + "/" + self.year


    def setDay(self, day):
        self.day = day


    def setMonth(self, month):
        self.month = month


    def setYear(self, year):
        self.year = year


    def setTime(self, time):
        self.time = time


    def getHomeTeam(self):
        return self.homeTeam


    def setHomeTeam(self, home):
        self.homeTeam = home


    def getAwayTeam(self):
        return self.awayTeam


    def setAwayTeam(self, away):
        self.awayTeam = away


    def getLocation(self):
        return self.location


    def setLocation(self, loc):
        self.location = loc
```

```
    def getScoreHome(self):
        return self.scoreHome


    def setScoreHome(self, score):
        self.scoreHome = score


    def getScoreAway(self):
        return self.scoreAway


    def setScoreAway(self, score):
        self.scoreAway = score


    def isForefit(self):
        return self.forefit


    def gameForefit(self):
        self.forefit = True


    def isComplete(self):
        return self.complete


    def gameComplete(self):
        self.complete = True


    def getWinningTeam(self):
        if self.complete:
            if self.scoreAway > self.scoreHome:
                return self.awayTeam
            else:
                return self.homeTeam


    def __str__(self):
        return '{} at. {}'.format(str(self.awayTeam), str(self.home
```

main.py

```
import uuid
from manager.Game import Game
from manager.Player import Player
from manager.Coach import Coach
from manager.Team import Team
from manager.Tournament import Tournament


class Runner:


    def __init__(self):

        self.players = []
        self.teams = []
```

Create Players

```
        self.p1 = Player(uuid.uuid4(), 'Drew', 'Casner')
        self.p2 = Player(uuid.uuid4(), 'RJ', 'Morley')
        self.p3 = Player(uuid.uuid4(), 'Jesper', 'Stryen')
        self.p4 = Player(uuid.uuid4(), 'Austin', 'Smith')
        self.p5 = Player(uuid.uuid4(), 'Lucas', 'Sward')
        self.p6 = Player(uuid.uuid4(), 'Ben', 'Settlerquist')
        self.p7 = Player(uuid.uuid4(), 'Powell', 'Hinson')
        self.p8 = Player(uuid.uuid4(), 'Quinn', 'Mahone')
        self.p9 = Player(uuid.uuid4(), 'Colt', 'Wise')
        self.p10 = Player(uuid.uuid4(), 'Ryan', 'Becker')
        self.p11 = Player(uuid.uuid4(), 'Matt', 'Skogen')
        self.p12 = Player(uuid.uuid4(), 'Isaiah', 'Jones')
        self.p13 = Player(uuid.uuid4(), 'John', 'Gadbois')
        self.p14 = Player(uuid.uuid4(), 'Kian', 'Tanner')
        self.p15 = Player(uuid.uuid4(), 'Pete', 'Snowden')
        self.p16 = Player(uuid.uuid4(), 'Nick', 'Hearon')

        self.players.append(self.p1)
        self.players.append(self.p2)
        self.players.append(self.p3)
        self.players.append(self.p4)
        self.players.append(self.p5)
        self.players.append(self.p6)
        self.players.append(self.p7)
        self.players.append(self.p8)
        self.players.append(self.p9)
        self.players.append(self.p10)
        self.players.append(self.p11)
        self.players.append(self.p12)
        self.players.append(self.p13)
        self.players.append(self.p14)
        self.players.append(self.p15)
        self.players.append(self.p16)
```

Create Teams

```
        self.t1 = Team(uuid.uuid4(), 'The Killerz', 'Boulder', 'Col
        self.t1.addPlayer(self.p1)
        self.t1.addPlayer(self.p2)
        self.t2 = Team(uuid.uuid4(), 'The Vikings', 'New York', 'Ne
        self.t2.addPlayer(self.p3)
        self.t2.addPlayer(self.p4)
        self.t3 = Team(uuid.uuid4(), 'The High Flyers', 'Austin', '
        self.t3.addPlayer(self.p6)
        self.t3.addPlayer(self.p7)
        self.t4 = Team(uuid.uuid4(), 'The Ballers', 'Aurora', 'Colo
        self.t4.addPlayer(self.p5)
        self.t4.addPlayer(self.p8)
        self.t5 = Team(uuid.uuid4(), 'The Wurst', 'Boulder', 'Color
        self.t5.addPlayer(self.p9)
        self.t5.addPlayer(self.p10)
        self.t6 = Team(uuid.uuid4(), 'Da N3rds', 'Seattle', 'Washir
        self.t6.addPlayer(self.p11)
        self.t6.addPlayer(self.p12)
        self.t7 = Team(uuid.uuid4(), 'The Climbers', 'LA', 'Cali')
        self.t7.addPlayer(self.p13)
        self.t7.addPlayer(self.p14)
        self.t8 = Team(uuid.uuid4(), 'Rockstars', 'Summit County',
        self.t8.addPlayer(self.p15)
        self.t8.addPlayer(self.p16)
```

```
            self.teams.append(self.t1)
            self.teams.append(self.t2)
            self.teams.append(self.t3)
            self.teams.append(self.t4)
            self.teams.append(self.t5)
            self.teams.append(self.t6)
            self.teams.append(self.t7)
            self.teams.append(self.t8)
```

Test Game g1 = Game(uuid.uuid4(), self.t1, self.t2)

Create Tournament

```
            self.turny1 = Tournament(uuid.uuid4(), 'Champions Club', 1,
            self.turny1.addTeam(self.t1, 0)
            self.turny1.addTeam(self.t2, 0)
            self.turny1.addTeam(self.t3, 0)
            self.turny1.addTeam(self.t4, 0)
            self.turny1.addTeam(self.t5, 0)
            self.turny1.addTeam(self.t6, 0)
            self.turny1.addTeam(self.t7, 0)
            self.turny1.addTeam(self.t8, 0)
            self.turny1.start()
```

Update results

```
            '''
            self.turny1.getGames()[2][0].setScoreHome(90)
            self.turny1.getGames()[2][0].setScoreAway(80)
            self.turny1.getGames()[2][0].gameComplete()
            print('')
            print(self.turny1.getGames()[2][0].getWinningTeam())
            print('')
            self.turny1.update()
    #DIVIDER

    #DIVIDER
        def getPlayers(self):
            return self.players
    #DIVIDER
        def getTeams(self):
            return self.teams
    #DIVIDER
        def getTourny(self):
            return self.turny1
```

”’

models.py

Create your models here.

```
from django.db import models
```

Player.py

```
class Player:

    def __init__(self, id, fname='', lname=''):
        self.id = id
        self.firstName = fname
        self.lastName = lname


    def getId(self):
        return self.id


    def getFirstName(self):
        return self.firstName


    def getLastName(self):
        return self.lastName


    def getFullName(self):
        return self.firstName + " " + self.lastName


    def editName(self, first, last):
        self.firstName = first
        self.lastName = last


    def __str__(self):
        return '{} {}'.format(self.firstName, self.lastName)
```

Team.py

```
from manager.Coach import Coach

class Team:

    def __init__(self, id, name=None, city=None, state=None):
        self.id = id
        self.name = name
        self.players = []
        self.coach = None
        self.city = city
        self.state = state
        self.seed = 0


    def getId(self):
        return self.id


    def getName(self):
        return self.name


    def setName(self, name):
        self.name = name


    def getCoach(self):
        return self.coach


    def setCoach(self):
        self.coach = coach


    def getPlayers(self):
        return self.players


    def addPlayer(self, player):
        self.players.append(player)


    def removePlayer(self, playerIn):
        itr = 0
        popIdx = -1
        for player in self.players:
            if player == playerIn:
                popIdx = itr
            itr = itr + 1

        if popIdx != -1:
            self.players.pop(popIdx)


    def getCity(self):
        return self.city


    def setCity(self, city):
        self.city = city


    def getState(self):
        return self.state


    def setState(self, state):
        self.state = state


    def setSeed(self, seed):
        self.seed = seed


    def getSeed(self):
        return self.seed
```

```
        def __str__(self):
            players = ''
            for player in self.players:
                players = players + str(player) + ' '
            return '#{} {}, {}, {}, {}'.format(self.seed, self.name, se
```

tests.py

Create your tests here.

```
from django.test import TestCase
```

Tournament.py

```
import random
import math
import uuid
from manager.Game import Game

class Tournament:

    def __init__(self, id, name, elims, st, gt):

        self.id = id
        self.name = name
        self.games = []
        self.teams = []
        self.winner = None
        self.elims = elims
        self.started = False
        self.startTime = st
        self.gameLength = gt

    def getID(self):
        return self.id

    def getName(self):
        return self.name

    def setName(self, name):
        self.name = name

    def getTeams(self):
        return self.teams

    def getGames(self):
        return self.games

    def addTeam(self, team, seed):

        if seed == 0:
            seed = random.randint(1, 101)
        if not self.started:
            self.teams.append(team)
            team.setSeed(seed)
        else:
            print('Tournament has started, cannot add team')

    def removeTeam(self, teamIn):
        itr = 0
        popIdx = -1
        for team in self.teams:
            if team == teamIn:
                popIdx = itr
            itr = itr + 1

        if popIdx != -1:
            self.teams.pop(popIdx)

    def getWinner(self):
        return self.winner

    def setWinner(self, winner):
        self.winner = winner

    def start(self):
        if not self.started:
            self.started = True
            if self.elims == 1: #Single Elimination
                numTeams = len(self.teams)
```

Generate Random Seed if non seeeded tourny

Sort teams by seed

Make sure home and away is correct

```
                depth = math.ceil(math.log(numTeams, 2))
                totGames = numTeams - 1


                self.teams.sort(key=lambda x: x.seed, reverse=False
                for idx in range(0, numTeams):
                    self.teams[idx].setSeed(idx + 1)

                gAdded = 0
                botGames = 0
                for i in range(0, depth):
                    botGames = 0
                    games = []
                    for j in range(0, (2**i)):
                        if gAdded < totGames: #Add Game
                            gAdded+=1
                            botGames+=1
                            newGame = Game(uuid.uuid4())
                            games.append(newGame)
                    self.games.append(games)


                teamSet = 0
                curDepth = depth
                teamAdded = []
                teamAdded.append(0)
                for round in reversed(self.games):
                    seed = 2**curDepth + 1
                    for game in reversed(round):
                        teamIdx = len(self.teams) - len(teamAdded)
                        if teamIdx not in teamAdded:
                            game.setAwayTeam(self.teams[teamIdx-1])
                            teamAdded.append(teamIdx)
                        teamIdx = seed - teamIdx
                        if teamIdx not in teamAdded:
                            game.setHomeTeam(self.teams[teamIdx-1])
                            teamAdded.append(teamIdx)
                        else:
                            if game.getAwayTeam():
                                game.setHomeTeam(game.getAwayTeam()
                                game.setAwayTeam(None)


                        if game.getAwayTeam() and game.getHomeTeam(
                            if game.getAwayTeam().seed < game.getHo
                                tempTeam = game.getAwayTeam()
                                game.setAwayTeam(game.getHomeTeam()
                                game.setHomeTeam(tempTeam)


                    curDepth-=1

                self.teams.sort(key=lambda x: x.seed, reverse=False
                roundIdx = 1
                for round in self.games:
                    print('====Round {}===='.format(roundIdx))
                    for game in round:
                        print(game)
                    roundIdx+=1

            else: # Double Elimination
                pass
        else:
            print('Tournament Already Started')


    def update(self):
        for idx in range(len(self.games)-2, -1, -1):
            count = 0
            for game in self.games[idx]:
                if not game.getAwayTeam():
                    game.setAwayTeam(self.games[idx+1][count].getWi
                    count+=1
                if not game.getHomeTeam():
                    game.setHomeTeam(self.games[idx+1][count].getWi
                    count+=1

        print('')
        roundIdx = 1
        for round in self.games:
            print('====Round {}===='.format(roundIdx))
            for game in round:
```

```
                print(game)
            roundIdx+=1
```

views.py

Create your views here.

```
from django.shortcuts import render
from manager.main import Runner


r = Runner()


def index(request):
    turny = r.getTourny()
    return render(request, "index.html", {'turny': turny})


def teams(request):

    teams = r.getTeams()
    return render(request, "teams.html", {'teams': teams})


def players(request):
    players = r.getPlayers()
    return render(request, "players.html", {'players': players})


def tournament(request):
    turny = r.getTourny()
    return render(request, "tournament.html", {'turny': turny})


def login(request):
    return render(request, "login.html")
```