main

rosters

```
from teams import teams_database
from overall import calculate_overall
rosters = {
    "Corry": {
        "Varsity": {
            "rating": 65,
            "players": ["Tucker Jackson", "Kamdyn Moon", "Wyatt Swan",
"Eli Gates", "Kaiden Krasa", "Alex Hellyer", "Jacob Anderson", "Ian
Shotts", "Bryce Chelton"]
        },
        "JV": {
            "rating": 50,
            "players": ["Jeremiah Johnson", "Jackson Lathrop", "Connor
Kent", "Chaz Bird", "Lennon Hill", "Mac Hulings", "Hunter Donoghue",
"Jayden Bolyard"]
        }
    },
    "Austintown Fitch": {
        "Varsity": {
            "rating": 79,
            "players": ["Cole Gamertsfelder", "Brady Stovall", "Daniel
Heddleson", "TJ McGagahan", "Cooper Jobe", "Nicholas Pugliese", "AJ
Marinelli", "Gavin Loomis", "Reggie Danko"]
        },
        "JV": {
            "rating": 55,
            "players": ["Ross Watkins", "Bryce Stovall", "Karter Oyster",
"Gunner Jack", "Evan Warmouth", "Mekhi DeMain", "Sebastian Pryor", "Jacob
Trautman"]
        }
    },
    "Erie High": {
        "Varsity": {
            "rating": 68,
            "players": ["Chase Benedict", "Jack Musone", "Brecken
McLaughlin", "Keishaun Richmond", "Joey Stephan", "Jacoby Thompson", "Alex
Wilpula", "Lucas Hanson", "Evan Bliley"]
        },
        "JV": {
            "rating": 48,
```

```
            "players": ["Logan Lazar", "Aiden Acevedo", "Montrell
Richmond", "Derrick Wolfram", "Saxon Emhoff", "Joey Isenburg", "Jackson
Flarity", "Maddox Burnett"]
        }
    }
}
```

ideas (viewer only edit)

standings

```python
import random
from teams import teams_database



class SeasonTracker:
    def __init__(self):
        self.standings = {}

    def update_result(self, team_name, won):
        if team_name not in self.standings:
            self.standings[team_name] = {"W": 0, "L": 0}
        if won:
            self.standings[team_name]["W"] += 1
        else:
            self.standings[team_name]["L"] += 1

    def display_standings(self):
        print("\n--- 2026 SEASON STANDINGS ---")
        print(f"{'Team':<20} | {'W':<3} | {'L':<3} | {'PCT':<5}")
        print("-" * 40)
        for team, record in self.standings.items():
            total = record["W"] + record["L"]
            pct = record["W"] / total if total > 0 else 0
            print(f"{team:<20} | {record['W']:<3} | {record['L']:<3} |
{pct:.3f}")

# --- SIMULATION ENGINE ---
tracker = SeasonTracker()

def play_game(team1, team2):
    total_rating = team1['rating'] + team2['rating']
    win_prob = team1['rating'] / total_rating

    if random.random() < win_prob:
        winner, loser = team1, team2
    else:
        winner, loser = team2, team1

    score_win = random.randint(4, 11)
```

```
    score_loss = random.randint(0, score_win - 1)

    print(f"Final: {winner['name']} {score_win}, {loser['name']}
{score_loss}")
    tracker.update_result(winner['name'], True)
    tracker.update_result(loser['name'], False)

def run_season_loop(my_school_name, game_count=10):
    # Find school
    my_team = None
    my_state = None
    for state, teams in teams_database.items():
        for t in teams:
            if t['name'] == my_school_name:
                my_team, my_state = t, state
                break

    if not my_team: return print("School not found!")

    for i in range(game_count):
        # 70% chance of local/state rival, 30% chance of national showcase
        if random.random() < 0.7:
            rivals = [t for t in teams_database[my_state] if t['name'] !=
my_school_name]
            opp = random.choice(rivals) if rivals else
random.choice(teams_database["TX"]) # Fallback
        else:
            other_states = [s for s in teams_database.keys() if s !=
my_state]
            opp = random.choice(teams_database[random_state :=
random.choice(other_states)])

        play_game(my_team, opp)

# --- EXECUTION ---
run_season_loop("Corry", game_count=5)
tracker.display_standings()
```

NHSI

```python
import random
import math

class Team:
    def __init__(self, name, state, rating, seed=1):
        self.name = f"{name} ({state})"
        self.rating = rating
        self.seed = seed

    def __repr__(self):
        return f"({self.seed}) {self.name}"

class NationalTournamentSim:
    def __init__(self):
        # Top 2026 Seeds based on current rankings (April 2026)
        self.field = [
            Team("Venice", "FL", 99, seed=1),      # No. 1 in Perfect Game
            Team("Orange Lutheran", "CA", 98, seed=1), # No. 2 in
MaxPreps/PG
            Team("Tomball", "TX", 97, seed=2),     # No. 4 in PG, No. 1 in
MaxPreps
            Team("Barbe", "LA", 96, seed=2),       # Longest win streak
(32 games)
            Team("St. John Bosco", "CA", 95, seed=3),
            Team("Grapevine", "TX", 94, seed=3),
            Team("Harvard-Westlake", "CA", 93, seed=4),
            Team("IMG Academy", "FL", 92, seed=4),
        ]

    def simulate_game(self, t1, t2):
        # Baseball scoring logic (lower scores, rare shutouts)
        win_prob = t1.rating / (t1.rating + t2.rating)
        winner = t1 if random.random() < win_prob else t2

        score_w = random.randint(2, 9)
        score_l = random.randint(0, score_w - 1)
        # Extra innings simulation if needed
        return winner, (score_w, score_l)

class UmpireCalls:
```

```
    def __init__(self):
        # 2026 NFHS specific rules
        self.calls = [
            "Excessive Player-to-Player Meeting (Limit 1 per inning)",
            "Running Lane Violation (Double First Base rule)",
            "Illegal Audio/Video Recording Device on Player",
            "Charged Defensive Conference (Coach Visit)",
            "Balk (Pivot Foot Parallel in Wind-up)"
        ]

    def throw_flag(self):
        return f"Umpire Decision: {random.choice(self.calls)}"

# --- Execution ---
sim = NationalTournamentSim()
ump = UmpireCalls()

print("🏆 --- 2026 NATIONAL HIGH SCHOOL INVITATIONAL (NHSI) --- 🏆")
# Simplified Quarterfinal Demo
for i in range(0, len(sim.field), 2):
    t1, t2 = sim.field[i], sim.field[i+1]
    winner, score = sim.simulate_game(t1, t2)
    print(f"Final: {t1} vs {t2} | {winner.name} wins
{score[0]}-{score[1]}")
    if random.random() > 0.7:
        print(f"   🚩 {ump.throw_flag()}")
```

Basically this is a 64 team tournament that will sim at the end of year it will have 3 seasons in one school year with the 3 state champions getting automatic bids in this tournament

state playoff

All states split in 2 parts eastern and western playing best of 3 expect the state finals 7 games