כותרת הפרויקט: פיתוח אפליקציה לניהול קבוצות פנטזיה של שחקני NBA

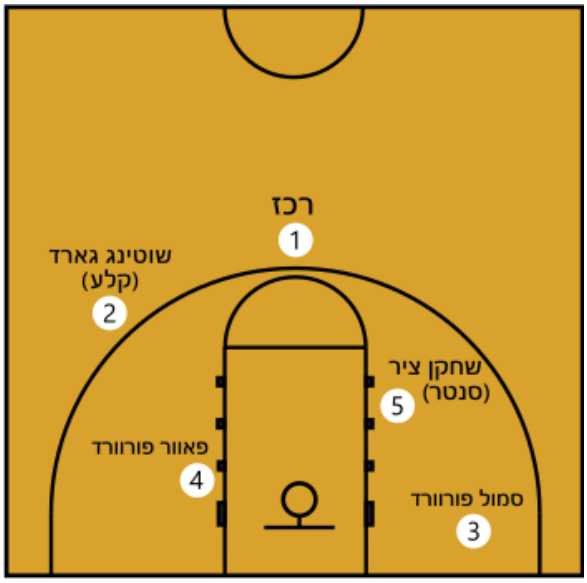
תיאור כללי:

בפרויקט זה, תתבקשו לפתח אפליקציה מבוססת Python המאפשרת למשתמשים ליצור ולנהל קבוצות פנטזיה של שחקני NBA. האפליקציה תכלול שני חלקים עיקריים:

הגשת הפרויקט: הזנת REPO בטופס שישלח לקראת סוף הפרויקט

1. קוד לאיסוף ושמירת נתונים:
   - איסוף נתוני שחקנים מ-API חיצוני עבור שלוש עונות (2022, 2023, 2024).
   - שמירת הנתונים באחד מהפורמטים הבאים לפי בחירתכם:
     - **CSV**
     - **SQLite** (שימוש ב-ORM)
     - **PostgreSQL** (שימוש ב-SQL)
2. שרת **Flask** עם נקודות קצה (**Endpoints**):
   - חיפוש שחקנים לפי עמדה.
   - ניהול קבוצות פנטזיה: יצירה, עריכה, מחיקה וניהול של מספר קבוצות.
   - קבלת סטטיסטיקה על הקבוצות פנטזיה

הסבר כללי על כדורסל (למי שלא מכיר או מבין):



- כדורסל – ויקיספר(wikibooks.org)

דרישות הפרויקט:

איסוף ושמירת נתונים:

- השתמשו ב-API הבא לקבלת נתוני שחקנים: NBA Players Stats API (לא נדרשת הרשמה).
- איסוף נתונים עבור עונות **2023**, **2022** ו-**2024**.
- שמרו את הנתונים בפורמט הנבחר.
- עיצוב מבנה נתונים שיכלול את כל המידע הרלוונטי, כולל נתוני השחקנים, העונות והסטטיסטיקות.

מידע על ה-API:

- אתם תשתמשו ב- API:

http://b8c40s8.143.198.70.30.sslip.io/api/PlayerDataTotals/query?season=2024&&pageSize=1000

- אתר עם מידע נוסף לגבי ה-API: NBA Stats API (getpostman.com)
- דוגמא לתשובת JSON


```
{
    "id": 18345,
    "playerName": "Zach LaVine",
    "position": "SG",
    "age": 28,
    "games": 25,
    "gamesStarted": 23,
    "minutesPg": 872.0,
    "fieldGoals": 170,
    "fieldAttempts": 376,
    "fieldPercent": 0.452,
    "threeFg": 59,
    "threeAttempts": 169,
    "threePercent": 0.349,
    "twoFg": 111,
    "twoAttempts": 207,
    "twoPercent": 0.536,
    "effectFgPercent": 0.531,
    "ft": 88,
    "ftAttempts": 103,
    "ftPercent": 0.854,
    "offensiveRb": 8,
    "defensiveRb": 121,
    "totalRb": 129,
    "assists": 98,
    "steals": 21,
    "blocks": 8,
    "turnovers": 52,
    "personalFouls": 57,
    "points": 487,
    "team": "CHI",
    "season": 2024,
    "playerId": "lavinza01"
},
```


- הסבר על סוגי הקליעות שיש בכדורסל וב-API
  - "fieldGoals": 170, → סה"כ קליעות מוצלחות ל-2 ו-3 נקודות
  - "fieldAttempts": 376, → סה"כ ניסיונות לקליעה ל-2 או 3
  - "fieldPercent": 0.452, → אחוז הצלח הקליעה ל-2 או ל-3
  - "threeFg": 59, → סה"כ קליעות ל-3
  - "threeAttempts": 169,
  - "threePercent": 0.349,
  - "twoFg": 111, → סה"כ קליעות ל-2
  - "twoAttempts": 207,

- "twoPercent": 0.536,
- הוא מאפשר לכם למשוך מידע על עונה אחת. כמובן אתם צריכים למשוך 3 עונות ולשמור במבנה נתונים

חישובים וניתוחים:

החישובים והניתוחים הללו צריכים לחזור עם המידע לשחקנים

- **מדד מחושב מקומי:**
  1. יחס אסיסטים לאיבודי כדור (**Assist-to-Turnover Ratio - ATR**):
  2. נוסחה

$$\frac{\text{assists}}{\text{turnovers}} = \text{ATR}$$

  3. **הסבר**: יחס זה מודד את היכולת של השחקן ליצור הזדמנויות לנקודות לעומת מספר התעבורות שהוא עושה. יחס גבוה מצביע על ניהול כדור טוב ויעיל.

- **מדד מחושב גלובלי:**
  1. יחס נקודות למשחק לעומת ממוצע העמדה (**Points Per Game Ratio Compared to Position Average - PPG Ratio**): **מדד מחושב גלובלי:**
  2. נוסחה

$$\frac{\text{points per game של השחקן}}{\text{ממוצע הנקודות למשחק בעמדה שלו בעונה}} = \text{PPG Ratio}$$

  - **הסבר**: יחס זה משווה את מספר הנקודות שהשחקן מכניס בממוצע לכל משחק לממוצע הנקודות למשחק של כל השחקנים בעמדה שלו באותה עונה. יחס גבוה מצביע על ביצועים טובים יותר מהממוצע בעמדה.

פיתוח שרת Flask:

Endpoint לחיפוש שחקנים לפי עמדה:

3. **URL:** `/api/players`?position={position}&season={season}
4. שיטת **HTTP:** `GET`
5. פרמטרים:
   - `position` (אופציות: `PG`, `SG`, `SF`, `PF`, `C`) – חובה (הסבר: כדורסל/עמדות של שחקנים – ויקיספר(wikibooks.org))
   - `season` (אופציונלי; אם לא מצוין, יש להחזיר את כל העונות).
6. תיאור:
   - מחזיר רשימה של שחקנים בעמדה המבוקשת.
   - לכל שחקן יש להציג את הנתונים הבאים:
     - שם השחקן - playerName
     - קבוצה - team
     - עמדה - position
     - עונות – רשימה של season (העונות שבהן שיחק)
     - כמות נקודות - points
     - כמות משחקים - games
     - אחוז קליעה לשתיים – twoPercent (מחושב נכון ל-3 העונות)
     - אחוז קליעה לשלוש - threePercent (מחושב נכון ל-3 עונות)
     - יחס אסיסטים לאיבודי כדור – ATR
     - יחס נקודות למשחק לעומת ממוצע העמדה – PPG Ratio

Endpoint לניהול קבוצות פנטזיה:

7. יצירת קבוצה:
   - **URL:** `/api/teams`
   - שיטת **HTTP:** `POST`
   - JSON **Body:** המכיל:
     - `team_name`: שם הקבוצה.
     - `player_ids`: רשימת מזהי השחקנים בקבוצה.
   - תיאור:
     - יוצר קבוצה חדשה ושומר אותה בבסיס הנתונים.
   - ואלידציה:
     - לא ניתן לייצר קבוצה בלי 5 שחקנים
     - חייב שחקן אחד בכל עמדה (יש 5 עמדות)
8. עריכת קבוצה:
   - **URL:** `/api/teams/<team_id>`
   - שיטת **HTTP:** `PUT`
   - JSON **Body:** המכיל את רשימת השחקנים (player_ids) המעודכנת
   - תיאור:
     - מעדכן את פרטי הקבוצה.
   - ואלידציה:

- לא ניתן לעדכן קבוצה בלי 5 שחקנים
- חייב שחקן אחד בכל עמדה (יש 5 עמדות)
- לא ניתן לעדכן יותר מקבוצה אחת עם אותו שחקן (הודעת שגיאה מתאימה)

9. מחיקת קבוצה:
   - **URL:** `/api/teams/<team_id>`
   - שיטת **HTTP:** `DELETE`
   - תיאור:
     - מוחק את הקבוצה מהבסיס נתונים.
   - ולידציה
     - לא ניתן למחוק קבוצה לא קיימת (הודעת שגיאה מתאימה)
10. קבלת פרטי קבוצה:
   - **URL:** `/api/teams/<team_id>`
   - שיטת **HTTP:** `GET`
   - תיאור:
     - מחזיר את פרטי הקבוצה, כולל:
       - שם הקבוצה.
       - רשימת השחקנים עם הנתונים הבאים (הנתונים הם לכל העונות יחד):
         - שם השחקן - playerName
         - קבוצה - team
         - עמדה - position
         - כמות נקודות - points
         - כמות משחקים - games
         - אחוז קליעה לשתיים - twoPercent
         - אחוז קליעה לשלוש - threePercent
         - יחס אסיסטים לאיבודי כדור – ATR
         - יחס נקודות למשחק לעומת ממוצע העמדה – PPG Ratio

Endpoint לסטטיסטיקות:

11. השוואה בין קבוצות פנטזיה
   - **URL:** `/api/teams/compare?team1={team_id}&team2={team_id}&…`
   - שיטת **HTTP:** `GET`
   - תיאור:
     - מבצע השוואה בין 2 או יותר קבוצות פנטזיה
     - קלט הוא ב-query string, שמות המשתנים הם team1, team2, וכן הלאה. הנתון עבור כל קבוצה הוא מזהה קבוצה
     - אין הגבלה לכמות הקבוצות שניתן להשוות
   - ולידאציה
     - לוודא שהקבוצות קיימות בבסיס נתונים
     - שיש לפחות 2 קבוצות בבקשה
     - לוודא ששמות הפרמטרים מדויקים ונכונים
   - תשובה
     - רשימה ממוינת של הקבוצות לפי **PPG Ratio** הקבוצתי מהגדול לקטן

- רשימת הקבוצות בהשוואה כאשר לכל קבוצה הנתונים הבאים (כל המידע מתייחס לכל העונות)
  - קבוצה - team
  - כמות נקודות - points
  - אחוז קליעה לשתיים – twoPercent (ממוצע של כל השחקנים)
  - אחוז קליעה לשלוש - threePercent (ממוצע של כל השחקנים)
  - יחס אסיסטים לאיבודי כדור – ATR **(ממוצע של** ATR **של כל השחקנים)**
  - יחס נקודות למשחק לעומת ממוצע העמדה – PPG Ratio (ממוצע של כל ה-PPG Ratio של כל שחקן בקבוצה)

12. בונוס: השוואה בין קבוצות בבסיס הנתונים

- **URL:** `/api/teams/stats?team1={team_name1}&team2={team_name2}&…`
- שיטת **HTTP:** `GET`
- תיאור:
  - מבצע השוואה בין 2 או יותר קבוצות רגילות
  - קלט הוא ב-query string, שמות המשתנים הם team1, team2, וכן הלאה. הנתון עבור כל קבוצה הוא שם הקבוצה (לדוגמא: CHI)
  - יש הגבלה להשוואה ל-3 קבוצות
  - יש צורך למצוא את כל השחקנים באותה קבוצה. יש להתחשב בכל השחקנים בקבוצה. הפעולה מתבצעת כלפי כל העונות
- ולידאציה
  - לוודא שהקבוצות קיימות בבסיס נתונים
  - שיש לפחות 2 קבוצות בבקשה
  - לוודא ששמות הפרמטרים מדויקים ונכונים
- תשובה
  - רשימה ממוינת של הקבוצות לפי **PPG Ratio** הקבוצתי מהגדול לקטן
  - רשימת הקבוצות בהשוואה כאשר לכל קבוצה הנתונים הבאים (כל המידע מתייחס לכל העונות)
    - קבוצה - team
    - כמות נקודות - points
    - אחוז קליעה לשתיים – twoPercent (ממוצע של כל השחקנים)
    - אחוז קליעה לשלוש - threePercent (ממוצע של כל השחקנים)
    - יחס אסיסטים לאיבודי כדור – ATR **(ממוצע של** ATR **של כל השחקנים)**
    - יחס נקודות למשחק לעומת ממוצע העמדה – PPG Ratio (ממוצע של כל ה-PPG Ratio של כל שחקן בקבוצה)

ניהול נתונים ובסיס נתונים:

- תכננו את מבנה בסיס הנתונים כך שיתמוך בדרישות הפרויקט.
- יש לקבל החלטה איך ואיזה נתונים לשמור בבסיס הנתונים ואיך הבסיס נתונים יראה. אתם מחליטים את מבנה הבסיס נתונים (חייב כמובן לתמוך בדירושת הפרויקט)
- אם אתם משתמשים ב-SQLite או PostgreSQL, ניתן להשתמש ב-ORM כמו SQLAlchemy.

דגשים:

- מבנה קוד ואדריכלות:
  - השתמשו בתכנון מודולרי וארכיטקטורה נכונה.
  - הקפידו על עקרונות תכנות נקי (Clean Code).
- תיעוד:
  - תעדו את הקוד באמצעות הערות והסברים.
  - צרו קובץ `README` המסביר כיצד להפעיל את האפליקציה, כולל הוראות התקנה.
- שימוש ב-**Git**:
  - השתמשו ב-Git לניהול גרסאות.
  - בצעו commit-ים משמעותיים עם הודעות ברורות המתארות את השינויים (טיפ: אחרי פירוק הפרויקט לחלקים – כמו שהוסבר אתמול, זה יעזור לכם להחליט מתי לבצע Commit, בסיום פיתוח רכיב מסוים לדוגמא)

רובריקה להערכת הפרויקט:

פונקציונליות כללית (70%):

- איסוף ושמירת נתונים (25%):
  - איסוף נתונים מ-API עבור שלוש העונות ושמירתם בפורמט הנבחר.
- חישובים וניתוחים (25%):
  - חישוב נכון של המדדים המחושבים
- מימוש ה-**Endpoints** (20%):
  - מימוש מלא של ה-Endpoints לפי הדרישות.
  - יכולת חיפוש שחקנים וניהול קבוצות (יצירה, עריכה, מחיקה).
- שאלת בונוס (7%) – endpoint של סטטיסטיקה לקבוצות בבסיס נתונים

ארכיטקטורה ותכנון (15%):

- מבנה הנתונים ובסיס הנתונים (7.5%):
  - תכנון נכון של הטבלאות, השדות והקשרים ביניהן.
  - שימוש יעיל ב-ORM או ב-SQL.
- מבנה הקוד והמודולריות (7.5%):
  - חלוקה נכונה של הקוד למודולים ופונקציות.
  - עמידה בעקרונות תכנות נקי ומודולרי.

איכות הקוד ופורמטינג (10%):

- קריאות הקוד (5%):
  - שימוש בשמות משתנים ופונקציות מובנים וברורים.

        - הוספת הערות במקומות הנכונים להסבר.
    - סטנדרטים ופורמטינג (5%):
        - שימוש בקווי קידוד אחידים ועקביים.

שימוש ב-Git (5%):

- ניהול גרסאות (2.5%):
    - היסטוריית commit-ים עקבית ומשמעותית.
- הודעות **commit** (2.5%):
    - הודעות ברורות ומפורטות המתארות את השינויים שבוצעו.

מטרות למידה:

- עבודה עם APIs ושילובם באפליקציה.
- הבנה ותכנון של בסיסי נתונים ובחירת הטכנולוגיה המתאימה.
- פיתוח אפליקציית Flask ויצירת Endpoints לשירותים.
- ניהול נתונים ויישום פעולות CRUD.
- ביצוע חישובים וניתוחים על הנתונים שנאספו.
- שימוש ב-Git לניהול גרסאות ויישום פרקטיקות עבודה נכונות.

הנחיות נוספות:

- בחירת טכנולוגיות:
    - אתם מחליטים באיזו טכנולוגיה להשתמש לאחסון הנתונים (CSV, SQLite, PostgreSQL).
    - שקלו את היתרונות והחסרונות של כל טכנולוגיה והסבירו את הבחירה שלכם.
    - חובה להראות שימוש נכון בפונקציות שנלמדו:map, reduce, filter וכו'
- הרחבות אפשריות:
    - הוסיפו אפשרות לסינון שחקנים לפי קריטריונים נוספים.
    - אפשרו למשתמשים להציג סטטיסטיקות נוספות על הקבוצה שלהם.