

## דו"ח פרויקט מסכם בעקרונות שפות תכנות

מגישים :

עידו קסטנבאום – 205767833

מתן שושן – 204196059

שם הפרויקט – sporTinder

### **רעיון ומוטיבציה**

הרעיון שלנו היה ליצור מערכת אשר מטרתה היא קביעת מפגשי ספורט בין משתמשים באמצעות ממשק נוח המציג פרטי משחקים/מגרשים רלוונטים העונים על תנאי השאילתה של המשתמש. המערכת משייכת מגרשי ספורט על פי אזורים ומקשרת את המיקומים של המשתמשים אשר מעוניינים במפגש ספורט.

המוטיבציה שלנו באה ממקום שבו אנחנו מבינים את הצורך באפליקציה שכזו, שכן המון אנשים אוהבים לעשות ספורט ומעוניינים במפגשי ספורט כגון משחקי כדורסל משותפים. עם זאת, רוב האנשים אינם יודעים מתי מתרחשים מפגשי ספורט בין אנשים שהם לא מכירים. ולכן, הפרויקט שלנו בא לענות על הצורך הזה כאשר הוא גם מקשר בין אנשים שרוצים לעשות ספורט וגם מקשר את המשתמש למגרש על פי מרחק שאותו הוא מגדיר כאשר הוא מחפש משחק להצטרף אליו.

למשתמש יש 2 אפשרויות.

**הצטרפות למשחק קיים:** המשתמש מגדיר את המיקום הנוכחי שלו, את המרחק אליו הוא מוכן לנסוע בשביל המשחק, את ענף הספורט, השעות והתאריך בהם הוא מעוניין לשחק.

**יצירת משחק:** המשתמש מגדיר את אותם הפרמטרים לאפשרות של הצטרפות למשחק קיים ובנוסף, מגדיר את מספר המשתתפים המקסימלי למשחק אותו הוא מעוניין ליצור.

במידה המשתמש בחר באפשרות להצטרפות למשחק קיים, לאחר מילוי השאילתה הוא מקבל את פרטי **המשחק** הקרוב ביותר למיקום שלו אשר עונה על תנאי השאילתה. במידה והוא מעוניין במשחק הוא לוחץ על כפתור ירוק ונרשם למשחק, במידה והוא לא מעוניין הוא לוחץ על כפתור אדום והוא יקבל את הפרטים של המשחק העונה לתנאי השאילתה שלו עם הקרבה השנייה ביותר למיקום שלו וכך הלאה.

במידה והמשתמש בחר באפשרות ליצור משחק חדש, לאחר מילוי השאילתה הוא יקבל את פרטי **המגרש** הקרוב אליו ביותר בדומה לתצוגה מהאפשרות הקודמת עם האפשרות ללחוץ כפתור ירוק וכפתור אדום.

נציין כי בחרנו להתמקד בפרויקט שלנו, בעיר אחת ידועה ומרכזית, ניו יורק. כל המידע הגיאוגרפי באפליקציה הוא רק על עיר זו. בנוסף האפליקציה עושה שימוש במידע הנלקח מתוך openStreetMap אשר עליה נרחיב בהמשך הדו"ח.

## סקירת התחום

ראשית, נתחיל בסקירה קצרה על פלטפורמת open street maps עליה הסתמכנו בביצוע הפרויקט

### open street maps - כללי

OpenStreetMap (בקיצור - OSM) הוא מיזם שיתופי ליצירת מפה חופשית של כדור הארץ. המפה נוצרת בעזרת נתונים ממכשירי GPS ניידים, תצלומי אוויר ומקורות חופשיים אחרים. גם קובצי התמונה של המפה וגם המידע הווקטורי, זמינים להורדה ולשימוש. משתמשים רשומים יכולים להעלות רשימות מסלולי GPS, וכן לערוך את המידע הווקטורי בעזרת כלי העריכה של המיזם. ההשראה ל-OpenStreetMap הייתה אתרי אינטרנט כגון ויקיפדיה - לצד המפה מוצגת באופן בולט לשונית "ערוך", ונשמרת היסטוריה מלאה של הגרסאות השונות.

### מאפיינים מרכזיים

- חינמית – שימוש מלא וחינמי לחלוטין, בכל מה שיש לפלטפורמה להציע. זאת בניגוד לפלטפורמות מפות ידועות אחרות, כמו google maps.
- בעלת פונקציונליות עשירה ומגוונת, הכוללת בין היתר: מידע על מיקומים, מפות מסוגים שונים, מסלולים, נקודות עניין, טעינת מפות לתוך gps ועוד..
- שיתופית – כל אחד יכול לערוך ולהוסיף מידע על מיקומים ומקומות בעולם (כמו ויקיפדיה רק של מפות)

בסעיף 4 (שימוש פונקציונלי) נפרט ביתר אריכות על השימוש שלנו בכלי openstreetmaps ונסביר איך הוא תרם לנו בפרויקט.

## **א. סקירת עבודות קודמות**

לאחר בדיקה אחר מערכות דומות, מצאנו מספר דוגמאות לאפליקציות הפותרות בעיה דומה לשלנו.

להלן שמות האפליקציות:

- פלייקרס ספורט חברתי
- Allsportscourts
- Playfinder
- Sportido
- Sport Connect

## **סיכום פתרונות קיימים**

לאחר שראינו דוגמאות של מערכות קיימות הפותרות בעיה דומה לשלנו, עולה כי בחלקן יש דמיון לשלנו, עם זאת ישנם פתרונות בעלי מאפיינים שונים. חלק מהפתרונות מתמקדים אך ורק במגרשים עצמם, ללא התייחסות למשחקים קיימים ולחיבור עם שחקנים אחרים בעלי העדפות דומות. כלומר, השירות שלהם מאפשר קבלת מידע עדכני על מגרשים זמינים על פי דרישת המשתמש (סוג מגרש, זמן) וביצוע הזמנה שלהם. לעומת זאת, ישנם פתרונות אשר מציעים בנוסף להזמנת מגרשים גם חיבור לשחקנים ולמשחקים קיימים בדומה למערכת שלנו. כמו כן, ישנם פתרונות שהתמקדו בסוג ספורט יחיד לעומת אחרים שהציעו פתרון למגוון רחב של סוגי ספורט שונים. כל השירותים שהוצעו, הציגו פלט עבור שאילתת המשתמש באמצעות רשימה (בין אם זה מגרשים או משחקים קיימים).

## **הפתרון שלנו בהשוואה למה שקיים**

הפתרון שלנו, בדומה לחלק מהפתרונות שראינו למעלה, גם נותן אופציה למשתמש לקבל מידע על מגרשים זמינים, על מנת ליצור משחק חדש וגם לקבל מידע על משחקים קיימים ולהצטרף למשחק קיים. בנוסף, הוא מתמקד בשלושה ענפי ספורט, כדורסל, טניס ובייסבול. בשונה מהפתרונות הקיימים אשר מחזירים אופציות לבחירה בתצורה של רשימה, הפתרון שלנו מציע פלט למשתמש בדומה לאפליקציית טינדר. בכל פעם מוצגת למשתמש האופציה המומלצת ביותר עבורו, כאשר יש לו אופציה לאשר או לסרב לאופציה שהוצעה. במידה והוא סירב תוצע לו אפשרות אחרת, עד שהוא יאשר או שיגמרו האופציות הקיימות. היתרון בתצוגה שלנו הוא, האופן שבו מוצגות למשתמש האופציות לבחירה, האפליקציה "מפתה את המשתמש" להשתתף בפעילות ספורט. כמו כן, המשתמש נדרש למינימום מאמץ, הוא רק צריך להזין את המידע הכי בסיסי, כך שהדרך עבורו לקבלת ההמלצות היא קלה ומהירה. בנוסף לכך, את כל המידע שקיבלנו על מיקומים של מגרשים, כתובות, קורדינטות ועוד, קיבלנו מ- openstreetmaps.

## ב. סקירת כלים טכניים

קישור לפרויקט: [https://github.com/njanakiev/openstreetmap-heatmap/blob/master/utis\\_osm.py](https://github.com/njanakiev/openstreetmap-heatmap/blob/master/utis_osm.py)

## תיאור הפרויקט

# Openstreetmap Heatmap

This project is a visualization of OpenStreetMap data with Blender and Python as a 3D barplot. It creates an occurrence heatmap of all points that are collected within a country with a certain tag.

OpenStreetMap (OSM) has a vast geospatial data set containing various tags and attributes besides the geometry. By using the [Overpass API](#) we can query for specific tags, filter specific areas and various other kinds of queries. In this case we want to query for tags within the boundary of a country. We are using the two-letter [country codes](#) and filter for a single [OSM tag](#) across all available [OSM elements](#). You can see the various [Map Features](#) to get a grasp of the variety.

הפרויקט בחבילה overpy שבה אנו השתמשנו, על מנת להריץ שאילתות מול Overpass api וקבלת מידע חזרה. ניתן להגדיר בשאילתה פיצ'רים כמו, תגים ספציפיים, גבולות מקום מבוקש וכו'.

## חברות ושימושים נוספים של openstreetmaps

1. חברת Yahoo התחילה להשתמש במידע של osm בשירות התמונות שלה
2. אפליקציית הניווט moovit
3. חוקרים באקדמיה משמשים ב-osm למחקר.
4. הנתונים של אפליקציית maps.me מבוססים על osm.

## הוראות התקנה

מצורף קובץ requirements.txt המפרט את הוראות ההתקנה. כמו כן, התקנו את הספריות על מערכת הפעלה ווינדוס.

## מאגרי מידע

כל המידע שעבדנו איתו במהלך הפרויקט, התקבל מבסיס הנתונים של osm, ללא מאגרי מידע סטטיים. המידע שקיבלנו מפורט בהמשך הדו"ח.

## שימוש פונקציונלי

נסביר כיצד השימוש ב- openstreetmaps (בקיצור osm) בא לידי ביטוי בפרויקט שלנו. כמו כן, נפרט את התועלת שקיבלנו מהשימוש בכלי. הפרויקט שלנו הסתמך על מידע של מיקומים גיאוגרפיים ובמקרה שלנו, מיקומים של מגרשי ספורט הקיימים בעיר ניו יורק. הדרך שלנו לקבלת המידע באמצעות osm.

אלטרנטיבה אחרת שנותנת מידע איכותי למטרה דומה לשלנו הינה google maps. הבעייתיות עם google היא שה- api החינמי שלו מוגבל והמלא בתשלום, לעומת osm הנותן את כל שירותיו בחינם, ללא יוצא מן הכלל. אלטרנטיבה אחרת יכלה להיות, מאגר סטטי קיים. לא רצינו להסתמך על מאגרים סטטיים כיוון, שגם לא נוכל להבטיח את עדכניות המידע, ובנוסף נהיה תלויים במאגרים קיימים. כלומר, באזור אחר לא בהכרח יהיה לנו מאגר מידע קיים. רצינו שתהיה לנו עצמאות באפשרות ה- scaling לערים ומדינות אחרות.

חשוב לנו לציין, כי הממשק וקבלת המידע מול - osm היה מאתגר, ודרש מאיתנו עבודה רבה ומחקר מקיף בנושא. שאילתות החיפוש מול המנועים של osm הינן שונות בתכלית מפלטפורמות ניווט אחרות. על כך נפרט בהמשך.

## פונקציונליות מרכזית המתבססת על osm

1. קבלת מידע מלא על מיקומים של מגרשי ספורט (כדורסל, בייסבול וטניס) בעיר ניו יורק
2. עבור קלט של כתובת המכילה (מדינה, עיר ורחוב), קבלת פלט מידע גיאוגרפי מלא על הכתובת (קו אורך ורוחב)
3. עבור קלט מיקום המיוצג באמצעות קו אורך ורוחב, קבלת פלט תיאור מלא של כתובת מיקום זה.

### קבלת מידע מלא על מיקומים של מגרשי ספורט (כדורסל, בייסבול וטניס) בעיר ניו יורק

את המידע קיבלנו באמצעות בקשת שאילתה לשרת הנקרא overpass api. שרת זה מתממשק מול בסיס הנתונים של osm ומחזיר מידע בפורמט json או בתצורת מפה. אנחנו השתמשנו בקבלת json. כמו כן, התקשורת מול השרת מתבצעת באמצעות בקשת GET וקבלת response. הסינטקס של השאילתה הינו מיוחד (דומה קצת ל-sql). בשאילתה עצמה ניתן להגדיר בצורה יחסית מדויקת מה אנחנו מחפשים על מנת שהשרת ידע איזה מידע לאחזר. ניתן לעשות זאת באמצעות הגדרת tags ספציפיים (דוגמה תינתן בהמשך). אנו ביצענו את הבקשה לשרת באמצעות ספריית request, ושליחת בקשת GET באמצעותה. נראה דוגמא לבקשה על ידי דוגמא מהקוד:

```
overpass_query = """
    [out:json][timeout:25];
    (area[name = "New York"]; node(area)[sport= "" + pitch_type + "" ][leisure=pitch]);out body;
    """
response = requests.get(OSM.overpass_url, params={'data': overpass_query})
```

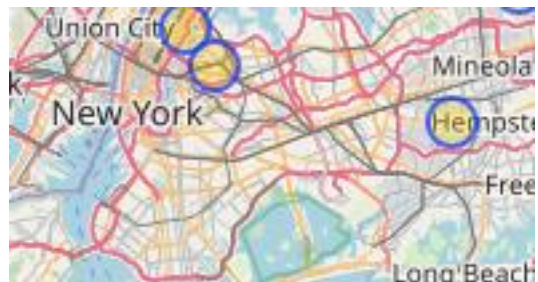
נעבור לדוגמא של בקשה ותשובה :

```
""
[out:json][timeout:25];
(area[name = "New York"];node(area)[sport=tennis][leisure=pitch]);out
body;
""
```

נסביר בקצרה את הדוגמא : ניתן לראות סינטקס של בקשת מגרשי טניס באזור ניו יורק. בשורה 1, אנו מבקשים לקבל את המידע בפורמט json. בשורה 2, נבקש את המידע מאזור ניו יורק. לאחר מכן, נרצה לקבל node (מייצג מיקומים) של התגים הבאים (מקבלים כמו מילון מפתח וערך), leisure=pitch sport=tennis, כלומר תביא לי מגרשים של טניס.

דוגמא לפלט שחזר בתצורת json ומפה :

```
{
  "type": "node",
  "id": 4742424227,
  "lat": 40.8490042,
  "lon": -72.9884169,
  "tags": {
    "leisure": "pitch",
    "name": "Country Point Tennis Court",
    "sport": "tennis"
  }
}
```



**עבור קלט של כתובת המכילה (מדינה, עיר ורחוב), קבלת פלט מידע גיאוגרפי מלא על הכתובת (קו אורך ורוחב)**  
מטרת הפונקציונליות הינה מציאת קו רחוב ואורך של מיקום המשתמש, על מנת לחשב מרחקים בינו לבין המגרשים המוצעים לו. את המידע קיבלנו באמצעות בקשת שאילתה לשרת הנקרא nominatim api. גם שרת זה מתממשק מול בסיס הנתונים של osm ומחזיר מידע בפורמט json או בתצורת מפה. גם כאן, השתמשנו בקבלת json. בדומה לשרת הקודם, התקשורת מול השרת מתבצעת באמצעות בקשת GET וקבלת response. גם כאן השתמשנו בספריית request לשליחת הבקשה. בנוסף, השימושים של 2 השרתים שונים בהתאם למטרת החיפוש. במקרה שלנו רצינו לקבל את מידע גיאוגרפי על הכתובת שהוזנה. אנו שלחנו את הבקשה כבקשת request רגילה עם פרמטרים בהתאם. ה-url הבסיסי הינו :

<https://nominatim.openstreetmap.org/search?<params>>

דוגמה לבקשה עם פרמטרים של כתובת (רחוב, עיר ומדינה), פורמט ג'ייסון, והחזרת פרטי כתובת:

<https://nominatim.openstreetmap.org/search?q=20 W 34th, New York, ארצות הברית&format=json&addressdetails=1>

Response שהתקבל עבור הבקשה:

```
{
  "place_id": 193784657,
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
  "osm_type": "way",
  "osm_id": 469875416,
  "boundingbox": [
    "40.7447808",
    "40.7449468",
    "-73.3761402",
    "-73.3759576"
  ],
  "lat": "40.7448487",
  "lon": "-73.37604890262162",
  "display_name": "20, South 34th Street, Suffolk County, New York, 11798, United States",
  "class": "building",
  "type": "house",
  "importance": 0.511,
  "address": {
    "house_number": "20",
    "road": "South 34th Street",
    "county": "Suffolk County",
    "state": "New York",
    "postcode": "11798",
    "country": "United States",
    "country_code": "us"
  }
}
```

עבור קלט מיקום המיוצג באמצעות קו אורך ורוחב, קבלת פלט תיאור מלא של כתובת מיקום זה  
אופן הפעולה הינו זהה לחלוטין לפונקציות הקודמת שהצגנו. נעבור ישר לדוגמא:

בקשה:

<https://nominatim.openstreetmap.org/reverse.php?lat=40.7499168&lon=-73.914521&format=jsonv2>

פרמטרים:

		lat	40.7499168
		lon	-73.914521
		format	jsonv2

תשובה:

```
"place_id": 55966101,
"licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
"osm_type": "node",
"osm_id": 4812375400,
"lat": 40.7497319,
"lon": -73.9141616,
"place_rank": 30,
"category": "leisure",
"type": "pitch",
"importance": 0,
"addresstype": "leisure",
"name": "Baseball field",
"display_name": "Baseball field, Barnett Avenue, Sunnyside Gardens, Queens, Queens County, New York, 11104, United States",
"address": {
  "leisure": "Baseball field",
  "road": "Barnett Avenue",
  "neighbourhood": "Sunnyside Gardens",
  "suburb": "Queens",
  "city_district": "Queens County",
  "city": "New York",
  "state": "New York",
  "postcode": "11104",
  "country": "United States",
  "country_code": "us"
},
"boundingbox": [
  40.7496819,
  40.7497819,
  -73.9142116,
  -73.9141116
]
```



כעת נפרט בקצרה על המודולים השונים המרכיבים את הפרויקט שלנו ועל המחלקות הקיימות. לאחר מכן, נפרט על הפונקציות העיקריות.

**מודלים:**

**Game.py**

- **Class Game** – מייצג אובייקט משחק

**GUI.py**

- **class First\_screen** – אובייקט המייצג עמוד ראשי
- **Class create\_Game** – אובייקט המייצג תצוגה של מגרשים רלוונטיים העונים לתנאי השאילתה של המשתמש לשם יצירת משחק
- **Class Join\_Game** – אובייקט המייצג תצוגה של משחקים רלוונטיים העונים לתנאי השאילתה של המשתמש לשם הצטרפותו למשחק קיים.

**location.py**

- **class Location** – מייצג אובייקט מיקום, כלומר מיקום גיאוגרפי המיוצג באמצעות כתובת וקו גובה ורוחב.

**mybackend.py**

- **Class Database** – מייצג אובייקט בסיס נתונים. כל העבודה והממשק אל מול בסיס הנתונים מתבצעת רק דרכו.

**openstreetmaps.py**

- **class OSM** – מייצג אובייקט **osm**, כלומר בעבודה והממשק אל מול ה- data וה- api של **osm** מתבצע דרכו

**pitch.py**

- **class Pitch** – מייצג אובייקט מגרש, יורש ממחלקת **Location** ומייצג מגרש (טניס, כדורסל או בייסבול)

**programmanager.py**

- **class ProgramManager** – מייצג אובייקט מנהל התוכנית, אחראי על תיאום בין שכבת הלוגיקה התצוגה. כלומר מתאם בין הגוי לבין שאר המודולים בלוגיקה – בסיס הנתונים ו- **osm**.  
\*\*\*פירוט מקיף יותר על המודולים והמחלקות השונות ניתן למצוא ב- **code documentation**.

**בסיס נתונים**

הפרויקט שלנו עובד עם בסיס נתונים **sqlite**. בסיס הנתונים מכיל 2 טבלאות: מגרשים ומשחקים.

## פירוט פונקציות עיקריות

### מודל OSM.py

def get\_pitches():

הפונקציה פונה ל- openstreetmaps data באמצעות ספריית overpy (המתממשק מול overpass api) ומקבלת ממנו רשימת מקומות המתויגים כמגרשי ספורט (כדורסל, טניס ובייסבול). הפונקציה מפרסרת את המידע שקיבלה (בפורמט json) ומחזירה רשימת מגרשים עם המידע הגיאוגרפי עליהם.

def get\_location(street, city="New York", country='ארצות הברית')

הפונקציה מקבלת רחוב כקלט. הפונקציה מקבלת את המידע הגאוגרפי המלא על הרחוב (תיאור מיקום, קו רחב, קו אורך, מס' זיהוי) באמצעות בקשת GET משרת nominatim השולף את המידע המתממשק עם המאגר של openstreetmaps. הפונקציה מחזירה את הנתונים שהתקבלו.

def get\_address\_by\_coordinates(lat, lon):

הפונקציה מקבלת מיקום המיוצג באמצעות קו רחב וקו אורך. הפונקציה מקבלת את הכתובת המלאה של המיקום (עיר, רחוב, מס' בית וכו') באמצעות בקשת GET משרת nominatim השולף את המידע ממאגר osm. הפונקציה מחזירה את הנתונים שהתקבלו

### מודל Location.py

def calculate\_distance(self, location):

הפונקציה מקבלת אובייקט מסוג Location. הפונקציה מחשבת את המרחק ביניהם על ידי נוסחת haversine ובאמצעות שימוש בספריית haversine, המגיעה עם פונקציית haversine לחישוב הנוסחה (הנוסחה מחשבת מרחק בין 2 נקודות המיוצגות באמצעות קו גובה ואורך)

## מודל mybackend.py

- def create\_db():

הפונקציה יוצרת את בסיס הנתונים. היא מקבלת רשימה של המגרשים הקיימים באמצעות הפונקציה get\_pitches מתוך OSM (נפרט עליה בהמשך), בונה את הטבלאות ומכניסה אליהן את ה-data באמצעות פונקציות העזר (insert\_data\_into\_pitches\_table, create\_games\_table, create\_pitches\_table).  
\*הפונקציה תיצור את בסיס הנתונים אך ורק במידה והוא עדיין לא נוצר, כלומר פעם אחד בלבד.

def get\_available\_pitches(type, start\_game):

הפונקציה מקבלת 2 פרמטרים: סוג ספורט וזמן תחילת משחק. הפונקציה מחזירה את כל המגרשים הפנויים למשחק אשר מתאימים לסוג הספורט בקלט ואינם כבר "תפוסים" במשחק שנקבע בזמן המבוקש. הפונקציה נעזרת בטבלת Games ובטבלת Pitches להחזרת המידע הרלוונטי.

def get\_available\_games(type, start\_game):

הפונקציה מקבלת 2 פרמטרים: סוג ספורט וזמן תחילת משחק. הפונקציה מחזירה את כל המשחקים אשר מתאימים לסוג הספורט בקלט ואינם כבר עדיין בתפוסה מלאה. הפונקציה נעזרת בטבלת Games לטובת החזרת המידע הרלוונטי.

def create\_game(id\_game, type, pitch\_id, start\_game, end\_game, max\_participants):

הפונקציה מכניסה רשומה חדשה לתוך טבלת Games המייצגת משחק חדש, בהתאם לפרמטרים שהתקבלו כקלט.

: def joining\_to\_game(id\_game, num\_participants)

הפונקציה מקבלת מס' משחק, ומס' משתתפים. הפונקציה מגדילה את כמות המשתתפים במשחק קיים בטבלת Games באחד. כלומר מייצגת הצטרפות שחקן למשחק קיים.

def delete\_finished\_games(list\_id\_games):

הפונקציה מוחקת את כל הרשומות מטבלת Games אשר זמן הסיום שלהם כבר עבר. כלומר משחקים שכבר נגמרו.

## מודול programanager.py

**: def set\_user\_location(street)**

הפונקציה מקבלת רחוב כקלט. הפונקציה מקבלת את המידע הגאוגרפי המלא על הרחוב (תיאור מיקום, קו רחב, קו אורך, מס' זיהוי) באמצעות הפונקציה get\_location(עליה פירטנו לפניכן) באמצעות מחלקת OSM. הפונקציה מאתחלת את השדה location בנתונים אלו.

**def is\_legal\_address(street):**

הפונקציה מקבלת רחוב כקלט. הפונקציה בודקת האם הרחוב שהוזן הינו חוקי וקיים בניו יורק. אם כן תחזיר True אחרת תחזיר False. הפונקציה מקבלת את המידע על הרחוב באמצעות הפונקציה get\_location(עליה פירטנו לפניכן) באמצעות מחלקת OSM.

**def create\_db():**

הפונקציה אחראית על יצירת בסיס הנתונים. היא מקבלת רשימה של מגרשי ספורט (כדורסל, בייסבול וטניס) קיימים בניו יורק מתוך ה-data של openstreetmaps באמצעות הפונקציה get\_pitches מתוך מחלקת OSM (עליה פירטנו לפניכן). הפונקציה מתקשרת עם מחלקת Database על מנת ליצור את בסיס הנתונים.  
**\*הפונקציה תיצור את בסיס הנתונים אך ורק במידה והוא עדיין לא נוצר, כלומר פעם אחד בלבד.**

**get\_available\_pitches(type, year, month, day, start\_hour, requested\_distance):**

הפונקציה מחזירה את כל המגרשים הפנויים למשחק אשר מתאימים לסוג הספורט בקלט ואינם כבר "תפוסים" במשחק שנקבע בזמן המבוקש. הפונקציה מקבלת ממחלקת Database את המידע הרלוונטי.

**def get\_available\_games(type, year, month, day, start\_hour, requested\_distance):**

הפונקציה מחזירה את כל המשחקים אשר מתאימים לסוג הספורט בקלט ואינם כבר עדיין בתפוסה מלאה. הפונקציה מקבלת ממחלקת Database את המידע הרלוונטי.

**def create\_game(type, pitch\_id, year, month, day, start\_hour, max\_participants):**

הפונקציה יוצרת משחק חדש בהתאם לפרמטרים שקיבלה. הפונקציה מתקשרת עם מחלקת Database לטובת פעולה זו.

**def join\_to\_game(id\_game):**

הפונקציה מצרפת את המשתמש למשחק ספציפי. הפונקציה מתקשרת עם מחלקת Database לטובת פעולה זו.

**def delete\_finished\_games() :**

הפונקציה מוחקת את כל המשחקים שנגמרו. הפונקציה מתקשרת עם מחלקת Database לטובת פעולה זו.

## מודל GUI.py

**def clicked\_join(self):** - פונקציה זו מופעלת כאשר המשתמש לוחץ במסך הראשי על כפתור **join game**, הפונקציה מעבירה אותו למסך שבו המשתמש ממלא פרטים על המשחק אשר אליו הוא רוצה להצטרף כמו: מרחק מקסימלי אליו הוא מוכן לנסוע, שעות ותאריך בהם הוא מעוניין להצטרף, סוג הספורט אותו הוא רוצה לשחק.

**def clicked\_create(self):** - פונקציה זו מופעלת כאשר המשתמש לוחץ במסך הראשי על כפתור **create game**, הפונקציה מעבירה אותו למסך שבו המשתמש ממלא פרטים על המשחק אותו הוא רוצה ליצור: מרחק מקסימלי אליו הוא מוכן לנסוע, שעות ותאריך בהם הוא מעוניין להצטרף, סוג הספורט אותו הוא רוצה לשחק ומספר המשתתפים המקסימלי של אותו משחק.

**def go\_back(self):** - פונקציה זו היא מימוש של כפתור **go back** אשר מחזיר את המשתמש אל מסך הבית.

**def validation(self):** - פונקציה זו מבצעת בדיקות ולידציה על השאילתה שהמשתמש ממלא. במידה וישנה בעיה קלט לא תקין או נכון, הפונקציה מציגה הודעה אש מפרטת על כל בעיות הקלט בשאילתה. במידה ואין בעיות, הפונקציה מבצעת מעבר על בחירת המשחק/המגרש בהתאם לבחירת המשתמש.

### כאשר לוחצים על יצירת משחק חדש:

**register\_game\_in\_db(self):** - כאשר המשתמש אשר יוצר את המשחק מאשר את פרטי המגרש המוצגים בפניו ולוחץ על הכפתור הירוק, הפונקציה שומרת את המשחק החדש ב**db** בטבלת משחקים וכמו כן מציגה למשתמש הודעה כי המשחק נוצר בהצלחה.

**def next\_pitch(self):** - כאשר המשתמש אשר יוצר את המשחק דוחה את פרטי המגרש המוצגים בפניו ולוחץ על הכפתור האדום, הפונקציה מחזירה את פרטי המגרש הבא אשר עונה על תנאי השאילתה. במידה והמשתמש יבחר ללחוץ על הכפתור האדום פעם נוספת, הפונקציה תציג בפניו פרטים של המגרש הבא אשר מתאים לתנאי השאילתה. הפונקציה מציגה את פרטי המגרשים על פי המיקום הקרוב ביותר אל המיקום של המשתמש. למשל, בהצגה הראשונה הפרטים של המגרש יהיו זה של המגרש הקרוב ביותר אל המשתמש ובכל לחיצה על הכפתור האדום יוצג המגרש הבא הקרוב ביותר אל המשתמש וכך הלאה.

### כאשר לוחצים על הצטרפות למשחק קיים:

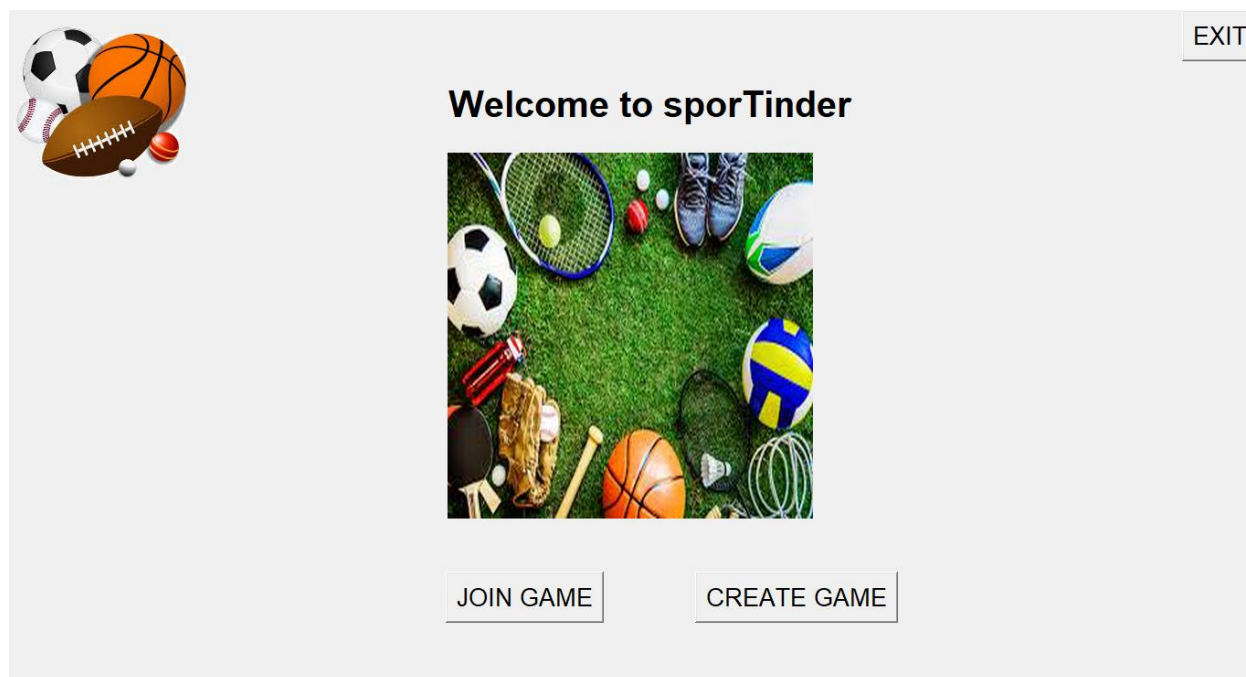
**def register\_in\_db(self):** - כאשר המשתמש אשר רוצה להצטרף למשחק קיים מאשר את פרטי המשחק אשר מוצגים בפניו ולוחץ על הכפתור הירוק, הפונקציה שומרת את הצטרפותו אל המשחק ומעדכנת את פרטי המשחק הקיים בכך שישנו אדם נוסף אשר הצטרף למשחק. כמו כן, המערכת תציג למשתמש כי נרשם למשחק הקיים בהצלחה.

**def next\_match(self):** - כאשר דוחה את פרטי המשחק המוצגים בפניו ולוחץ על הכפתור האדום, הפונקציה מחזירה את פרטי המשחק הבא אשר עונה על תנאי השאילתה. במידה והמשתמש יבחר ללחוץ על הכפתור האדום פעם נוספת, הפונקציה תציג בפניו פרטים של המשחק הבא אשר מתאים לתנאי השאילתה בדומה לפונקציה **def next\_pitch(self)** אשר מתאימה לאפשרות יצירת משחק חדש.


## תוצאות

להלן תצלומי מסך מן האפליקציה בשלבים שונים של המערכת :

עמוד ראשי :



טופס למילוי השאילתה לשם יצירת משחק חדש :



EXIT

Enter your username

What your current street

How far would you go?(km)

Choose your sport

basketball

Choose your hours to play

10:00 - 12:00


Choose date

12/27/20

Max participants

FIND ME A MATCH

GO BACK



EXIT

Enter your username

ido

What your current street

mercerc

How far would you go?(km)

1000

Choose your sport

basketball

Choose your hours to play

10:00 - 12:00

Choose date

12/28/20

Max participants

6

FIND ME A MATCH

GO BACK

לאחר לחיצה על הכפתור "FIND ME A MATCH" 15 נקבל את פרטי המגרש הראשון אשר מתאים לתנאי השאלתה.



EXIT

Pitch Address: Gun Club Road, Altamont, Town of Guilderland, Albany County, New York, 12009, United States

Distance(km): 59

Sport: basketball

Time: 10:00 - 12:00



במידה ונלחץ על הכפתור האדום נקבל פרטי מגרש נוסף :



EXIT

Pitch Address: Red Mills Road, Town of Crawford, New York, 12566, United States

Distance(Km): 107

Sport: basketball

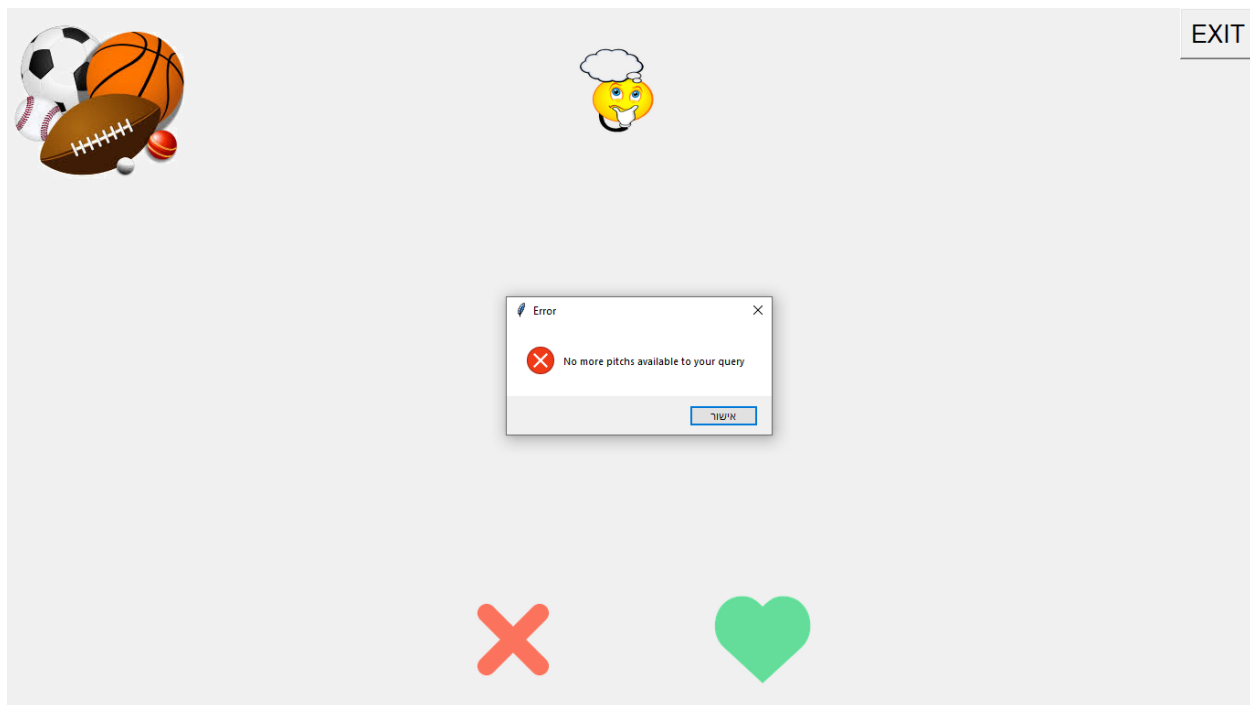
Time: 10:00 - 12:00



במידה ונאשר נקבל הודעה כי יצרנו את המשחק בהצלחה והאפליקציה תיסגר :

במידה ויגמרו המשחקים הרלוונטים נקבל את ההודעה הבאה :





להלן מסך הצטרפות למשחק קיים:

Enter your username

What your current street

How far would you go?(km)

Choose your sport

Choose your hours to play

Choose date

להלן פרטי המשחק הראשון המוצג מתוצאות השאילתה



EXIT

Pitch address: Red Mills Road, Town of Crawford, New York, 12566, United States

Distance(Km): 107

Sport: basketball

Time: 10:00 - 12:00

Max participants: 6

Current participants: 1



גם בהצטרפות למשחק קיים, פורמט ההצגה והמעבר בין משחקים נשאר זהה.



EXIT

Pitch address: Red Mills Road, Town of Crawford, New York, 12566, United States

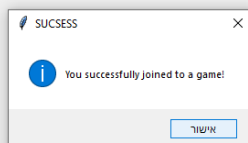
Distance(Km): 107

Sport: basketball



Time: 10:00 - 12:00

Max participants: 6

Current participants: 1



בחיפוש אחר משחק קיים אחר, נראה כי ההצטרפות שעשינו מקודם עדכנה את ה-current participants ל-2.



EXIT

Pitch address: Red Mills Road, Town of Crawford, New York, 12566, United States



Distance(Km): 107

Sport: basketball


Time: 10:00 - 12:00

Max participants: 6

Current participants: 2



להלן דוגמאות עבור קלט לא תקין והשגיאה אשר הוצגה עבורם:



EXIT

Enter your username

What your current street

How far would you go?(km)


Choose your sport

Choose your hours to play

Choose date


Max participants

Error

 Distance must contains only digits.  
Max participants must contains only digits

אישור

בגלל שהרחובות נלקחו רק ממדינת ניו יורק, אין זיהוי לרחובות מחוץ למדינה זו.



EXIT

Enter your username

ido

What your current street

rager

How far would you go?(km)

4

Choose your sport

basketball

Choose your hours to play

10:00 - 12:00

Choose date

12/28/20

Max participants

4

FIND ME A MATCH


GO BACK

Error

Not legal address

אישור

התאריך שנבחר כבר עבר:



EXIT

Enter your username

ido

What your current street

mercerc

How far would you go?(km)

4

Choose your sport

basketball

Choose your hours to play

10:00 - 12:00

Choose date

12/22/20

Max participants

4

FIND ME A MATCH

GO BACK

Error

Selected date cant be in the past.

אישור

קלט ריק:



EXIT

Enter your username

What your current street

How far would you go?(km)

Choose your sport

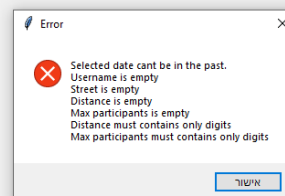
Choose your hours to play

Choose date

Max participants

FIND ME A MATCH

GO BACK



**מסקנות –** המסקנות שלנו מהשימוש בספרייה של OSM היא, שהיא עונה על הצורך בכך שהיא מזהה תבניות של מגרשי ספורט, ובכך הצלחנו לאחזר את כל מגרשי הספורט המתויגים מתוך המדינה ניו יורק שבארצות הברית. מתוך המיקומים, קיבלנו מידע על מגרשים אלו כמו מזהה מיקום, כתובת, וקווי אורך ורוחב. עם זאת, במהלך העבודה עם הכלי הבנו כי הוא לא מציג את כל האפשרויות אשר תכננו לממש בתחילת הפרויקט.

כמו למשל, חיפוש מיקום מסוים על פי מרחק הוא לא משהו שהצלחנו לממש באמצעות כלי זה וכדי לממש את האפשרות הזו היינו צריכים להשתמש בכלי נוסף אשר מחפש מרחק בין קואורדינטות.

דוגמה נוספת היא להצגת מפה – בתחילת הפרויקט חשבנו על מימוש של הצגת תצלום מפה, אך המימוש שלו באמצעות ספרייה זו היה מורכב ומסורבל מאוד ולכן העדפנו להתמקד אך ורק בתצוגת כרטיסיות בדומה לאפליקציית Tinder אשר עונה באופן טוב על ההגדרה של הפרויקט שלנו, והוא קישור משתמשים למגרשי ספורט קיימים באמצעות תיוגים.

אנחנו מאמינים כי הפיתוח שלנו אכן עונה על הציפיות אשר הגדרנו בתחילת העבודה ומהווה פלטפורמה נוחה וטובה ליצירה של משחקי ספורט.