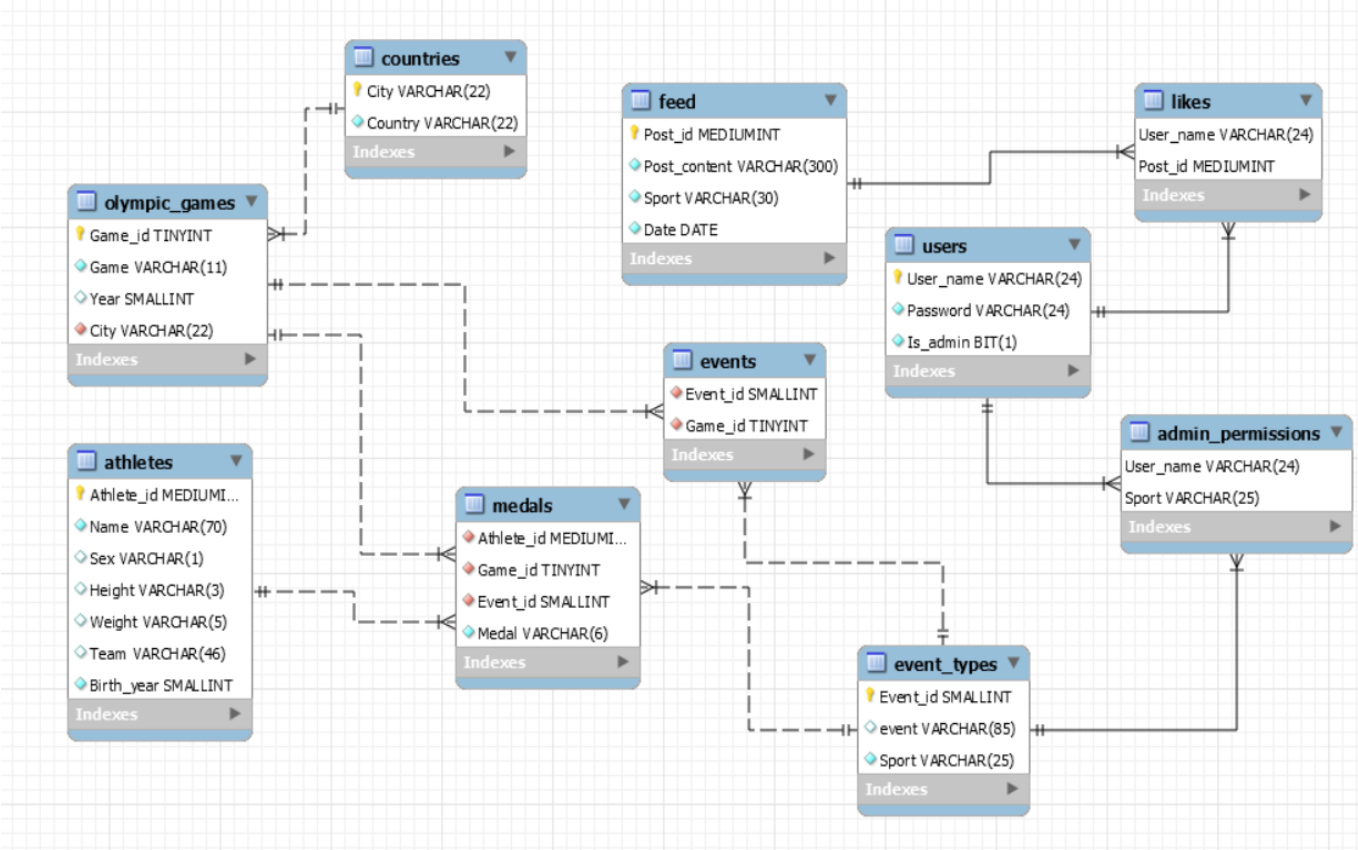
Software Documentation

Sapir Asras 209549625, Omer Abarzel 207041005,
Noam Barak 208787093, Eden Higani 315791434

Schema:



SQL Queries:

הערה: שם הסכמה לפני השינוי עבור ההגשה בשרת היה olympicapp ולכן שם זה מופיע בqueries שביצענו במסמך זה.

Tables:

שם הטבלה המקורית ממנה יצרנו את הטבלאות הוא : athlete_events
עבור יצירת הטבלאות השתמשנו בשאילתות הבאות:

athletes:

- CREATE TABLE athletes_temp AS
  (SELECT DISTINCT ID as Athlete_id, Name, Sex, Height, Weight, Team, Age, Year FROM olympicapp.athlete_events);
- UPDATE athletes_temp SET Age = Year - Age;
- CREATE TABLE athletes AS
  (SELECT DISTINCT Athlete_id, Name, Sex, Height, Weight, Team, Age as Birth_year FROM olympicapp.athletes_temp);
- Create table temp as(
  SELECT *

```
FROM athletes
GROUP BY Athlete_id
HAVING COUNT(Athlete_id)>1);
```

- DELETE from athletes WHERE Athlete_id in(select Athlete_id from temp);
- INSERT INTO athletes
  SELECT * FROM temp;
- ALTER TABLE `olympicapp`.`athletes `
  CHANGE COLUMN `Athlete_id` `Athlete_id` SMALLINT NOT NULL ,
  CHANGE COLUMN `Name` `Name` VARCHAR(80) NULL DEFAULT NULL ,
  CHANGE COLUMN `Sex` `Sex` VARCHAR(1) NULL DEFAULT NULL ,
  CHANGE COLUMN `Height` `Height` VARCHAR(3) NULL DEFAULT NULL ,
  CHANGE COLUMN `Weight` `Weight` VARCHAR(5) NULL DEFAULT NULL ,
  CHANGE COLUMN `Team` `Team` VARCHAR(46) NULL DEFAULT NULL ,
  CHANGE COLUMN `Birth_year` `Birth_year` SMALLINT NOT NULL ,
  ADD PRIMARY KEY (`Athlete_id`);

טיפלנו בבפילויות עם השאילתות המתוארות מעלה.

<u>olympic_games:</u>

- CREATE TABLE olympic_games
  AS (SELECT distinct Games, Year, City
  FROM olympicapp.athlete_events);
- ALTER TABLE `olympic_games` ADD COLUMN `Game_id` INT NOT NULL
  AUTO_INCREMENT primary KEY;
- ALTER TABLE `olympicapp`.`olympic_games`
- CHANGE COLUMN `Game_id` `Game_id` TINYINT NOT NULL
  AUTO_INCREMENT FIRST,
  CHANGE COLUMN `Games` `Game` VARCHAR(11) NOT NULL ,
  CHANGE COLUMN `Year` `Year` SMALLINT NULL DEFAULT NULL ,
  CHANGE COLUMN `City` `City` VARCHAR(22) NOT NULL ;

<u>event_types:</u>

- CREATE TABLE event_types AS (SELECT distinct event, Sport
  FROM olympicapp.athlete_events);
- ALTER TABLE `event_types` ADD COLUMN `Event_id` INT NOT NULL
  AUTO_INCREMENT primary KEY;
- ALTER TABLE `olympicapp`.`event_types`
  CHANGE COLUMN `Sport` `Sport` VARCHAR(25) NULL DEFAULT NULL AFTER
  `event_id`,
  CHANGE COLUMN `event_id` `event_id` SMALLINT NOT NULL ,
  CHANGE COLUMN `event` `event` VARCHAR(85) NULL DEFAULT NULL ;

<u>events:</u>

- create table events as
  ((SELECT distinct event_types.Event_id, medals.Game_id
  FROM olympicapp.event_types, olympicapp.medals
  WHERE event_types.Event_id=olympicapp.medals.Event_id
  ORDER BY event_types.Event_id, olympicapp.medals.Game_id asc))

- ALTER TABLE `olympicapp`.`events`
  CHANGE COLUMN `Event_id` `Event_id` SMALLINT NOT NULL ,
  CHANGE COLUMN `Game_id` `Game_id` TINYINT NOT NULL ,
  ADD PRIMARY KEY (`Event_id`, `Game_id`);;

<u>medals:</u>

- create table medals as
  (SELECT ID as Athlete_id , Game_id, Event_id, Medal
  from athlete_events
  LEFT JOIN
  (SELECT event_types.Event_id, event, Game, Game_id
  FROM olympicapp.event_types
  LEFT JOIN
  (SELECT Game_id, events.Game, Event_id
  FROM olympic_games
  LEFT JOIN events ON olympic_games.Game = events.Game
  UNION
  SELECT Game_id, events.Game, Event_id
  FROM olympic_games
  RIGHT JOIN events ON olympic_games.Game = events.Game) as bla
  ON event_types.Event_id = bla.Event_id) as temp3

  ON athlete_events.Event = temp3.Event AND athlete_events.Games = temp3.Game)

- ALTER TABLE `olympicapp`.`medals`
  CHANGE COLUMN `Athlete_id` `Athlete_id` MEDIUMINT NOT NULL ,
  CHANGE COLUMN `Game_id` `Game_id` TINYINT NOT NULL ,
  CHANGE COLUMN `Event_id` `Event_id` SMALLINT NOT NULL ,
  CHANGE COLUMN `Medal` `Medal` VARCHAR(6) NULL DEFAULT NULL ,
  ADD PRIMARY KEY (`Athlete_id`, `Game_id`, `Event_id`);

- ALTER TABLE `world`.`medals`
  ADD PRIMARY KEY (`Athlete_id`, `Game_id`, `Event_id`);

<u>countries:</u>

- create table countries as
  (select City, Country from countries_old where City in(SELECT City FROM olympicapp.olympic_games));
- ALTER TABLE `olympicapp`.`countries`
  CHANGE COLUMN `City` `City` VARCHAR(22) NOT NULL ,
  CHANGE COLUMN `Country` `Country` VARCHAR(22) NOT NULL ,

ADD PRIMARY KEY (`City`);

users:

- CREATE TABLE `users`
  (`User_name` VARCHAR(24) NOT NULL,
  `Password` VARCHAR(24),
  `Is_admin` BIT(1) NOT NULL,
  PRIMARY KEY(`User_name`)) ;

admin_permissions:

- CREATE TABLE `admin_permissions`
  (`User_name` VARCHAR(24) NOT NULL,
  `Sport` VARCHAR(30) NOT NULL,
  PRIMARY KEY(`User_name`, `Sport`));

feed:

- CREATE TABLE `feed`
  (`Post_id` INT NOT NULL AUTO_INCREMENT,
  `Post_content` VARCHAR(300) NOT NULL,
  `Sport` VARCHAR(30)NOT NULL,
  `Date` DATE NOT NULL,
  PRIMARY KEY(`Post_id`),
  UNIQUE KEY `Post_content_UNIQUE`(`Post_content`));

likes:

- CREATE TABLE `likes`
  (`User_name` VARCHAR(24) NOT NULL,
  `Post_id` INT NOT NULL,
  PRIMARY KEY(`User_name`, `Post_id`));

Foreign Keys:

- events : fk to events_types on column Event_id
  fk to olympic_games on column Game_id
- medals : fk to event_types on column Event_id
  fk to athletes on column Athlete_id
  fk to olympic_games on column Game_id
- likes: fk to feed on column Post_id
  fk to users on column User_name
- olymic_games: fk to countries on column City
- admin_permissions: fk to users on column User_name
  fk to event_types on column Sport

Indexes:

primary and default from the foreign keys.

Queries From Code:

שאילתות מ DBGeneral:

- שם הפונקציה: SelectColFromTable
  מקבלת: col - עמודה , table - טבלה, helper - הוספה של עוד אלמנטים.
  מחזירה: עמודה מטבלה מסוימת col.

SELECT DISTINCT + col + FROM olympicapp.+ table + helper + ;

- שם הפונקציה: TheMostXAthlete
  מקבלת: X - פרמטר, sport - ספורט, order- הסדר.
  מחזירה: את שמות 4 האתלטים הכי X בסדר **order** מספורט מסוים ואת ערך הפרמטר X עצמו. לדוגמא- שם האתלט הכי גבוה בבדורסל ואת גובהו.

SELECT Name,**X** FROM
(SELECT Athlete_id, Name, **X** FROM
(SELECT Athlete_id, Name, **X** FROM olympicapp.athletes AS temp WHERE Athlete_id IN (SELECT Athlete_Id FROM olympicapp.medals WHERE (event_id IN (SELECT event_id FROM olympicapp.event_types WHERE sport = "**sport**")) GROUP BY Athlete_Id) AND **X** <> "NA") AS temp2 ORDER BY cast(X as unsigned) **order** LIMIT 4) AS temp3;

- שם הפונקציה: TheBestXAthlete
  מקבלת: sport - ספורט, helper - הוספת אלמנטים נוספים לשאילתה.
  מחזירה: את 4 האתלטים הכי טובים בספורט מסוים עם ה helper - AND medal <> "NA" ואת האתלטים שהשתתפו בהכי הרבה אולימפיאדות בספורט מסוים אם ה helper = "".

SELECT Name FROM olympicapp.athletes WHERE Athlete_Id IN (SELECT Athlete_Id FROM (
SELECT Athlete_Id, COUNT(*) AS magnitude FROM (SELECT Athlete_Id, Medal FROM olympics.medals WHERE ((event_id IN
(SELECT event_id FROM olympicapp.event_types WHERE sport = "**sport**"))) **helper**) AS temp
GROUP BY Athlete_Id
ORDER BY magnitude DESC
LIMIT 4) AS temp2);

- שם הפונקציה: LocationOfOlympicGame
  מקבלת: משחק אולימפי לדוגמה Summer 2012
  מחזירה: את העיר והמדינה בה האולימפיאדה התקיימה.

SELECT Country,City FROM countries WHERE City =(SELECT City FROM olympic_games WHERE Game = “**game**“);

שאילתות מ DBSearch:

- שם הפונקציה: Filter
  מקבלת: מילון של ערכים לחיפוש לפיהם.
  מחזירה: 30 תוצאות חיפוש רנדומליות במקסימום .

SELECT distinct **selectStr** FROM **table** WHERE **whereVals** ORDER BY RAND() LIMIT 30;

שאילתות מ DBUsers:

- שם הפונקציה: NewUserRegister
  מקבלת: שם משתמש וסיסמה.
  מחזירה: אובייקט של משתמש עם הערכים שהתקבלו במידה והמשתמש לא קיים אובייקט של יוזר עם ערכי null במידה והמשתמש כבר קיים.

INSERT INTO olympicapp.users (User_name,Password,Is_admin) VALUES ("**username** "," **password** ",0);

- שם הפונקציה: Login
  מקבלת: שם משתמש וסיסמה.
  מחזירה: אובייקט של משתמש עם הערכים שהתקבלו במידה והמשתמש קיים אובייקט של יוזר עם ערכי null במידה והמשתמש לא קיים.

SELECT User_name, Password, Is_admin FROM olympicapp.users WHERE User_name = "**username** " AND Password =" **password** ";

- שם הפונקציה: DeleteUser
  מקבלת: שם משתמש וסימן האם הוא אדמין.
  במידה והמשתמש הוא משתמש רגיל , מוחקת אותו מטבלת המשתמשים בלבד. במידה ולמשתמש יש הרשאות נוספות מוחקת אותו גם מטבלת הרשאות האדמין.
  בנוסף מוחקת את הלייקים שעשה המשתמש על פוסטים.

DELETE FROM olympicapp.users WHERE User_name ="**username**";

DELETE FROM olympicapp.admin_permissions WHERE User_name = "**username**";

DELETE FROM olympicapp.likes WHERE User_name = "**username** ";

- שם הפונקציה:ChangePassword
  מקבלת:שם משתמש וסיסמה חדשה.
  מחזירה: true אם עדכון הסיסמה עבר בהצלחה false במידה ולא.

UPDATE olympicapp.users SET Password =" **new_password** " WHERE User_name = "**username**";

- שם הפונקציה: UpdateAdmin
  מקבלת: שם משתמש , סיסמה, האם הוא אדמין.
  מחזירה: true אם עדכון פרטי האדמין עבר בהצלחה false במידה ולא.
  במידה והמשתמש כבר אדמין יש רק לעדכן את טבלת ההרשאות, במידה והמשתמש הוא אדמין חדש יש לעדכן זאת גם בטבלת המשתמשים.

UPDATE olympicapp.users SET Is_admin = 1 WHERE User_name = “**username**”
INSERT INTO admin_permissions (User_name, Sport) VALUES(“**username**",”**sport**”);

- שם הפונקציה: GetAdminList
  מקבלת: שם משתמש
  מחזירה: רשימת תחומי ספורט בהם המשתמש קיבל הרשאה.

SELECT Sport FROM olympicapp.admin_permissions
WHERE User_name = “**username**”;

שאילתות מ DBQuiz:

- שם הפונקציה: Get4RandomAthletesBySport
  מקבלת: ספורט.
  מחזירה: רשימה של 4 אתלטים מהספורט שקיבלה.

SELECT DISTINCT olympicapp.athletes.Name, olympicapp.medals.Medal
FROM olympicapp.medals JOIN olympicapp.event_types
ON olympicapp.medals.Event_id = olympicapp.event_types.Event_id JOIN olympicapp.athletes
ON olympicapp.athletes.Athlete_id = olympicapp.medals.Athlete_id
WHERE olympicapp.event_types.Sport = "**sport**")
ORDER BY RAND() LIMIT 4;

- שם הפונקציה: GetXByYWhereZFromAthletes
  מקבלת: x- העמודה הרצויה, y - התנאי שרוצים לבדוק, z - הערך שהתנאי דורש.
  מחזירה: את האתלט עבורו מתקיים התנאי.

SELECT **x** From olympicapp.athletes WHERE **y** = “**z**”;

- שם הפונקציה: WrongYears
  מקבלת: שנה.
  מחזירה: רשימה של של 3 שנים שאינה מכילה את השנה שהתקבלה לפונקציה.

SELECT Birth_year From olympicapp.athletes WHERE Birth_year <> "**year**” LIMIT 3;

- שם הפונקציה: WrongCountries
  מקבלת: מדינה.
  מחזירה: רשימה של של 3 מדינות שאינה מכילה את המדינה שהתקבלה לפונקציה.

SELECT DISTINCT Country From olympicapp.countries WHERE Country <> “**country**“ LIMIT 3;

שאילתות מ DBFeed:

- שם הפונקציה:InsertIntoFeedTable
  מקבלת: תוכן של פוסט, הספורט אליו הפוסט קשור.
  הפונקציה מכניסה לטבלה של ה Feed את הפוסט שקיבלנו יחד עם התאריך הנוכחי.

INSERT INTO olympicapp.feed (Post_content,Sport,Date) VALUES (“**content”**, “**sport”**, “**date”**);

- שם הפונקציה: FeedPosts
  מחזירה: רשימה של פוסטים.
  השאילתה בוחרת 10 פוסטים רנדומליים מהטבלה של ה Feed. עבור כל פוסט נבחר יוצרת אובייקט של POST ובודקת ומעדכנת את מספר הלייקים. לבסוף מחזירה רשימה של POST

SELECT * FROM olympicapp.feed ORDER BY RAND() LIMIT 10;

- שם הפונקציה: GetNumberOfLikes
  מקבלת: id של פוסט מסוים.
  מחזירה: מספר הלייקים שהפוסט קיבל.

SELECT COUNT(“**Post_id”**) AS NumberOfLikes FROM (SELECT Post_id FROM olympicapp.likes WHERE Post_id = “**post_id”**) AS temp";

- שם הפונקציה: GetNumberOfAthletesFromTeam
  מקבלת:נבחרת מסוימת.
  מחזירה: מספר האתלטים שייצגו את הנבחרת שהתקבלה.

SELECT COUNT(distinct athletes.Name) as number FROM olympicapp.athletes WHERE athletes.Team='**team**";

- שם הפונקציה: GetDistinctEvents
  מקבלת:,תחום ספורט.
  מחזירה: מספר האירועים בספורט זה.

SELECT COUNT(distinct event_types.event) as number FROM olympicapp.event_types WHERE event_types.Sport=”**sport**”;

- שם הפונקציה: GetAvgOfGender
  מקבלת:,תחום ספורט, מין.
  מחזירה: ממוצע הגובה של המין שהתקבל בתחום הספורט שהתקבל.

SELECT ROUND(AVG(athletes.Height),2) as avg FROM olympicapp.athletes WHERE athletes.Sex= **“gender”** AND NOT athletes.Height='NA'
and Athlete_id IN
(SELECT Athlete_Id FROM olympicapp.medals WHERE((event_id IN (SELECT event_id FROM olympicapp.event_types WHERE sport = **“sport”**))));

- שם הפונקציה: GetRandomWin
  מקבלת:תחום ספורט.
  מחזירה: זכייה רנדומלית בתחום הספורט שהתקבל.
  זכייה- שם של אתלט מהתחום שהתקבל ומדליה שזכה בה.

(SELECT olympicapp.athletes.Name, olympicapp.medals.Medal
FROM olympicapp.medals JOIN olympicapp.event_types ON olympicapp.medals.Event_id = olympicapp.event_types.Event_id JOIN olympicapp.athletes
ON olympicapp.athletes.Athlete_id = olympicapp.medals.Athlete_id WHERE NOT olympicapp.medals.Medal = "NA" AND olympicapp.event_types.Sport =**”sport”**)
ORDER BY RAND() LIMIT 1;

- שם הפונקציה: LikePost
  מקבלת: id של פוסט, שם משתמש שאהב את הפוסט.
  מחזירה: true אם הצליחה בעדכון הטבלה, אחרת false.
  הפונקציה מוסיפה לטבלת ה Likes את הרשומה.

INSERT INTO olympicapp.likes (User_name,Post_id)
VALUES (“**username**”,"**post_id**");

- שם הפונקציה: DislikePost
  מקבלת: id של פוסט, שם משתמש שאהב את הפוסט.
  מחזירה: true אם הצליחה בעדכון הטבלה, אחרת false.
  הפונקציה מוחקת מטבלת ה Likes את הרשומה.

DELETE FROM olympicapp.likes WHERE User_name=**”username”** and Post_id = **“post_id”**;

Data Sources:

את המידע שהשתמשנו בו באפליקציה לקחנו מהאתר kaggle.
השתמשנו ב 3 דאטה סטים.
המרכזי המכיל את כל רשומות הספורטאים, המשחקים, המדליות ואירועים הקשורים לאולימפיאדה.
קישור לדאטה סט המרכזי

עבור איכלוס הדאטה בייס שלנו במשתמשים לקחנו דאטה סט של משתמשים וסיסמאות כך יצרנו את טבלת היוזרים שלנו.
קישור לדאטה סט - משתמשים

עבור טבלת המדינות נעזרנו בדאטה סט הבא:
קישור לדאטה סט - מדינות

Code Structure:

בפרויקט זה בחרנו ליצור אפליקציית WEB ע"י שימוש בעקרונות REST API.
החלוקה הלוגית מתבצעת בצורה הבאה:

- שכבת ה UI
- שכבת הלוגיקה (Model, Controllers)
- שכבת ה DB

שכבת ה UI :

מתחלקת ל3 חלקים ונמצאת בתוך תיקיית wwwroot:

- קבצי HTML.
- קבצי js, שימוש בספריית jQuery ובקשות HTTP.
- קבצי CSS ובנוסף שימוש ב bootstrap.

שכבת הלוגיקה :

Controllers -

קבצי הקונטרולרים מקבלים בקשות HTTP המגיעות מצד הלקוח (ע"י JS) ומעבירות את הבקשות ל App Manager.
חלוקת הקונטרולרים היא לפי הלוגיקה של האפליקציה:

- Feed
- Quiz
- Search
- Users

Model -

ה App Manager יוצר חיבור יחיד מול הדאטה בייס, מקבל בקשות מהקונטרולרים השונים ומנתב אותן למחלקות הDB המתאימות.
בנוסף קיימות המחלקות Post, User, Question המגדירות אובייקטים עבור המידע שעובר באפליקציה.

שכבת ה DB:

DBConnect - מחלקה זו אחראית על פתיחת חיבור מול ה SQL Server וסגירתו.
המחלקות המפורטות מטה הן המחלקות היחידות בקוד אשר מבצעות שאילתות על הדאטה בייס.
מחלקות אלה מקבלות את החיבור היחיד ומבצעות את השאילתות דרכו.
DBGeneral - מחלקה זו אחראית על השאילתות הכלליות המשמשות מחלקות נוספות שעובדות מול הדאטה בייס.
DBFeed - מחלקה זו אחראית על השאילתות הקשורות לפיד המוצג באתר.
DBQuiz - מחלקה זו אחראית על השאילתות הקשורות למבחנים בנושאי הספורט השונים המוצגים באתר בלשונית Tests.
DBSearch - מחלקה זו אחראית על השאילתות הקשורות לדף החיפוש המוצג באתר בלשונית Search.
DBUsers - מחלקה זו אחראית על השאילתות הקשורות למשתמשים, החל מדף הרישום/ההתחברות ובדף הפרופיל האישי ובכל הקשור למשתמשים.

General Flow Of The Application:

בכניסה לאתר יש לבצע הרשמה לאתר או התחברות למשתמש קיים. לאחר ההתחברות לאתר מוצג דף הבית שבו מופיעות ידיעות שונות על האולימפיאדות ועל האתלטים השונים.
בסרגל הניווט ניתן להגיע לעמוד הפרופיל האישי בו ניתן למחוק את המשתמש, להתנתק, לשנות סיסמא ולראות אילו בחנים המשתמש צלח באתר. בעמוד החיפוש, המשתמש יכול לבחור פילטרים שונים ולקבל תוצאות. בעמוד המבחנים המשתמש יכול לבחור ענף ספורט ולבצע מבחן בנושא, במידה וצלח 3 שאלות מתוך 5 יופיע ענף זה בפרופיל האישי באזור המבחנים.