

Taki-Online

TAKI Online

עבודת גמר

בתכנון ותכנות מערכות

במסלול שירותי רשת אינטרנט

שנת הלימודים

שם התלמיד: איתי לוסטיג

בית ספר : תיכון ע"ש אילן רמון

ת.ז: 213915937

שם המורה: סמדר וכטר

הקדשה/תודה:

עבודה זו נתנה לי את האפשרות להתמודד עם שלל אתגרים, הן בתחום הטכנולוגי, חיפוש וחשיבה על הטכנולוגיות המתאימות לפיתוח המשחק ויישומם של טכנולוגיות אלו (חלקם למדתי במגמה וחלקם למדתי בעצמי), והן בחשיבה הכוללת של פיתוח משחק מקוון, כמו יצירת עניין למשתמש (התקדמות בליגות), הוספת חברים, ותחרות ממושכת מול עצמך ומול שחקנים אחרים ברשת. רציתי להודות למגמה ובראשה סמדר על ההזדמנות והתמיכה שהם נתנו לי, לפתח את הפרוייקט הזה.

במהלך העבודה מאוד נהייתי משלבי התכנון והחקירה, שמורכבים מכל העולמות

3.....	הקדשה/תודה:
4.....	תוכן עניינים
5.....	מבוא
5.....	מטרות האתר
5.....	מטרת האתר:
5.....	קהל יעד
5.....	תיאור האתר
8.....	מבנה בסיס הנתונים
51.....	שרותי הרשת באתר
51.....	שירותי רשת שהאפליקציה מספקת
51.....	שירותי רשת נצרכים ממקור חיצוני
55.....	מדריך האפליקציה לשחקן
64.....	מדריך האפליקציה למנהל
68.....	נספחים
69.....	ReadMe

מבוא

מטרות האתר

מטרת האתר:

הפרויקט הוא מערכת המאפשרת לשחקנים לשחק במשחק טאקי באופן מקוון ולהיות חלק מליגת שחקנים. האתר מאפשר לשחקנים לצפות בליגות השונות ומיקומיהם בהן. כמו כן יש אפשרות דרך האתר לחפש ולבחור שחקן אחר ולהציע לו הצעת חברות. ניתן לפתוח חדר משחק שבו כמה שחקנים יכולים להשתתף במשחק או להציע הצעת משחק לשחקן אחר. המשחק וחוקיו הם לפי חוקי הטאקי הקלאסי, על פיו שחקן יכול לבצע מהלכים מותרים ונימנע מפעולות לא חוקיות.

ניתן להזמין חבר ברשת שרשום באתר המשחק לצורך משחק. והיא אינה מוגבלת למספר שחקנים.

המערכת מאפשרת לצפות בכל נתוני הליגה והשחקנים בעונה הנוכחית.

המערכת מנהלת את מיקומם של השחקנים בליגה על פי הישגיהם במשחקי הטאקי והנסיון אותו הם רכשו המיוצג בדרגות. מודל החישוב של מיקומך בליגה הוא מודל המבוסס על האלגוריתם של ליגות שחמט בעולם

לאחר יש גם משתמש מסוג מנהל, שיכול לצפות בנתונים וסטטיסטיקות נוספות על האתר. בסיום כל משחק השחקנים מקבלים את תוצאת המשחק ואת הישגיהם

המערכת מאפשרת לשחקן לשנות את שם המשתמש ותמונת הפרופיל שלו.

קהל יעד

קהל היעד הוא אנשים שאוהבים לשחק טאקי (או חדשים למשחק ורוצים ללמוד), ורוצים לשחק עם אנשים אחרים ברשת, או אנשים שמחפשים פלטפורמה תחרותית לטאקי. המשחק מיועד לגילאים 13-100

תיאור האתר

אורח:

- הרשמה לאתר בו הנרשם מתבקש למלא את הנתונים הבאים

- שם פרטי
- שם משפחה
- שם משתמש
- דואר אלקטרוני
- תאריך לידה
- תמונת פרופיל (לא חובה)

חשבון שחקן:

- בחשבון השחקן ניתן לראות את הנתונים הבאים

- שם פרטי
- שם משפחה

- חברים של השחקן
- לראות חדרי משחק פתוחים
- בקשות לחברות
- בקשה לשחק איתך משחקן אחר
- רמת השחקן
- ניסיון של השחקן
- דירוג ליגה של השחקן
- פאי תוצאות המשחקים של השחקן (כמה ניצח וכמה הפסיד)
- את כל השחקנים שנמצאים הליגות השונות

- בחשבון השחקן ניתן לבצע את הפעולות הבאות:

- חיפוש שחקנים אחרים שחברים במשחק
- הצעת חברות לשחקן
- קבלת הצעת חברות משחקנים אחרים
- שינוי תמונת פרופיל
- שינוי שם משתמש
- פתיחת חדר משחק לשחקנים חדשים
- חיפוש חדרי משחק פתוחים
- כניסה לחדר משחק פתוח כדי להשתתף במשחק
- הצעה לשחקן אחר להשתתף במשחק
- הפעלה של משחק חדש

המשחק:

- המשחק מבוסס על חוקי הטאקי הקלאסיים

- כל שחקן משחק בתורו
- ניתן לראות בזמן אמת תור מי מהשחקנים לשחק כעת
- במידה והתבצע פעולה לא חוקית המערכת נותנת הודעה על מהלך לא חוקי והסיבה לחוסר החוקיות

- בסיום המשחק המערכת מכריזה על המנצח לכל השחקנים
 - המערכת מעדכנת את הנתונים הבאים של כל שחקן בסוף המשחק
-
- ניסיון החדש של השחקן XP – מבוסס על תוצאות המשחק ורמת השחקן בו (כמה קלפים נשארו לו)
 - כמה נקודות חסרות לו על מנת לעבור שלב נוסף בניסיון
 - הרמה החדשה של השחקן ELO (יכול להישאר או להשתנות) – מבוסס על לוגיקת חישוב מבוסס על שיטת SME המבוססת על השיטה שבה גם משתמשים לדירוג שחקני שחמט וטניסאים בעולם
 - הליגה החדשה אליה הוא שייך (במידה והשתנת)
 - כמות המשחקים אותם הפסדת או ניצחת

מבנה בסיס הנתונים

Users-טבלת המשתמשים

XP	Level	First Name	Last Name	Date Of Birth	Username	Email	Password	Type	Picture	לחץ כדי להוסיף
1	1	Alon	Metuky	6/22/2003	Alonzo nZ@gmail.com	onisama	0			
0	1	Gerisha	Jeager	4/30/2017	AttackTitan lel@gmail.com	adon	0			
14	1	Ohad	Hertz	4/30/2017	Avi lar@gmail.com	aviKing	0		123.png	
77	4	Eren	Jeager	4/30/2017	Founder lel@gmail.com	lovemikasa	0			
84	5	itai192	lustig	9/30/2020	itai192 tig@gmail.com	bruhr	1		itai192.png	
6	1	itai	lustig	6/6/2016	JoJo tig@gmail.com	omen5	0		amen.png	

שם שדה	סוג נתונים	
XP	מספר	נקודות הניסיון של השחקן
Level	מספר	הרמה של השחקן
First Name	טקסט קצר	שם השחקן
Last Name	טקסט קצר	שם המשפחה של השחקן
Date Of Birth	תאריך/שעה	תאריך הלידה של השחקן
Username	טקסט קצר	שם המשתמש של השחקן
Email	טקסט קצר	אימייל של המשתמש
Password	טקסט קצר	סיסמה של המשתמש
Type	מספר	מספר הסוג משתמש של המשתמש (0 לשחקן רגיל 1 למנהל)
Picture	טקסט קצר	השם של תמונת הפרופיל של המשתמש

Friend Request Statuses- טבלת הצעות חברות ומצבן

Sender	Recipiant	Request stat	Time Sent	לחץ כדי להוסיף
Alonzo	itai192	1	#####	
Avi	Alonzo	0	#####	
Avi	itai192	1	#####	
itai192	AttackTitan	0	#####	
itai192	Founder	1	#####	
shira9	itai192	1	#####	
tamar9	itai192	1	#####	

שם שדה	סוג נתונים	
Sender	טקסט קצר	שולח של ההצעת חברות
Recipiant	טקסט קצר	נמען הצעת החברות
Request status	מספר	סטטוס ההזמנה
Time Sent	תאריך/שעה	הזמן בו נשלחה ההזמנה

Game Invitations - טבלת הזמנות למשחקים

Game ID	Recipient	Sender
38	itai192	Founder
39	itai192	Founder
88	itai192	shira9
83	itai192	tamar9
82	itai192	yana
81	itai192	yossi

שם שדה	סוג נתונים	
Sender	טקסט קצר	השולח של ההזמנה
Recipient	טקסט קצר	הנמען של ההזמנה
Game ID	מספר	מספר המשחק

Games - טבלת המשחקים

Game ID	Time Played	Activity	Game Name	Host
2	1/4/2021	2	game 1	itai192
3	1/5/2021	2	game 2	itai192
4	1/15/2021	2	game 3	itai192
5	12/15/2020	2	game 4	itai192
7	3/8/2021 4:52:03 PM	2	game 5	itai192
8	3/5/2021 4:54:36 PM	2	game 6	itai192
9	3/6/2021 4:54:38 PM	2	game 7	itai192
10	3/6/2021 4:54:39 PM	2	game 8	itai192
11	3/18/2021 4:54:41 PM	2	itai's game 9	itai192
38	4/7/2021 5:43:22 PM	2	Try my game	Founder
39	4/7/2021 6:16:42 PM	1	Try my game	Founder
79	4/8/2021 11:45:34 PM	2	hell yeha	Founder

שם שדה	סוג נתונים	
Game ID	מספור אוטומטי	מספר המשחק
Time Played	תאריך/שעה	הזמן בו המשחק נפתח
Activity	מספר	מספר הפעילות של המשחק (0 עדיין לא התחיל, 1 התחיל, 2 נגמר)
Game Name	טקסט קצר	שם המשחק (אם המשחק בהתאמה אישית)
Host	טקסט קצר	מי שיצר את המשחק

Ranks - טבלת הליגות

Rank ID	Rank Name	Lowest Elo
1	Noob	0
2	Beginer	151
3	Intermidate	251
4	Master	351
5	Grand Master	451
6	Card Wizard	551

שם שדה	סוג נתונים	
Rank ID	מספור אוטומטי	מספר הסולם (מעין ליגה בה השחקן מתחרה מדירוג מסויים עד דירוג מסויים)
Rank Name	טקסט קצר	שם הסולם
Lowest Elo	מספר	הדירוג הנמוך ביותר בו אפשר להיות בסולם

Ranking History - טבלת הדירוג בתחילת כל עונה לכל משתמש

User	Season	Season Start
Alonzo	2	100
AttackTitan	2	100
Avi	2	100
Founder	2	100
itai192	2	100
JoJo	1	100
JoJo	2	100
MidgetSpinner	2	100

שם שדה	סוג נתונים	
User	טקסט קצר	שם המשתמש
Season	מספר	מספר העונה
Season Start Elo	מספר	מספר הדירוג

Seasons - טבלת העונות

Season ID	Start Date	End Date
1	10/3/2020	1/1/2021
2	1/2/2021	6/20/2021

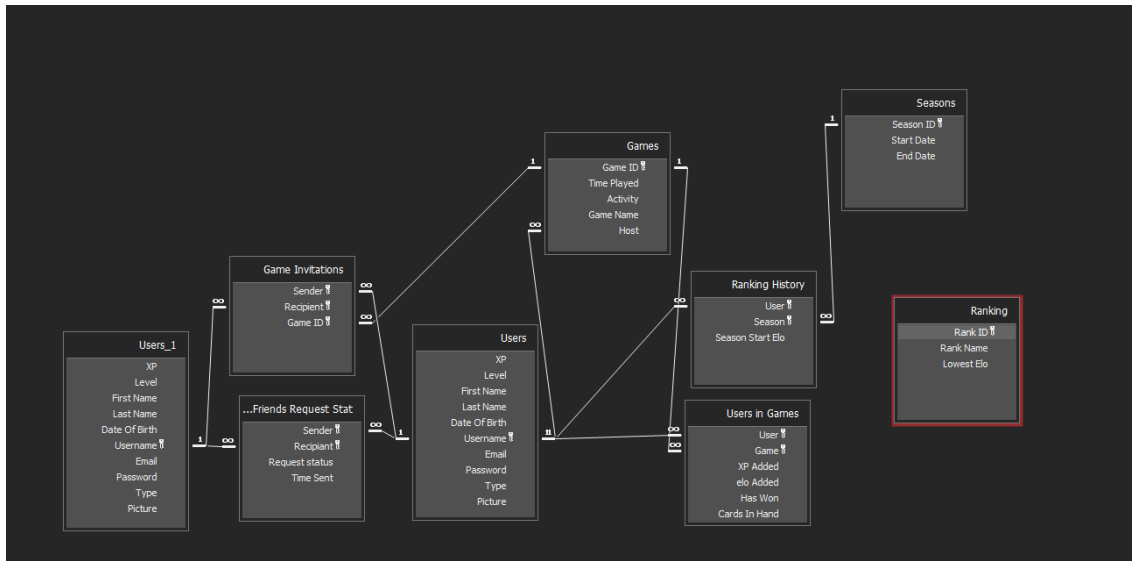
שם שדה	סוג נתונים	
Season ID	מספור אוטומטי	מספר העונה (העונות מתחלפות ואיתן מתאפס הדירוג של השחקנים)
Start Date	תאריך/שעה	תאריך ההתחלה של העונה
End Date	תאריך/שעה	תאריך הסיום של העונה

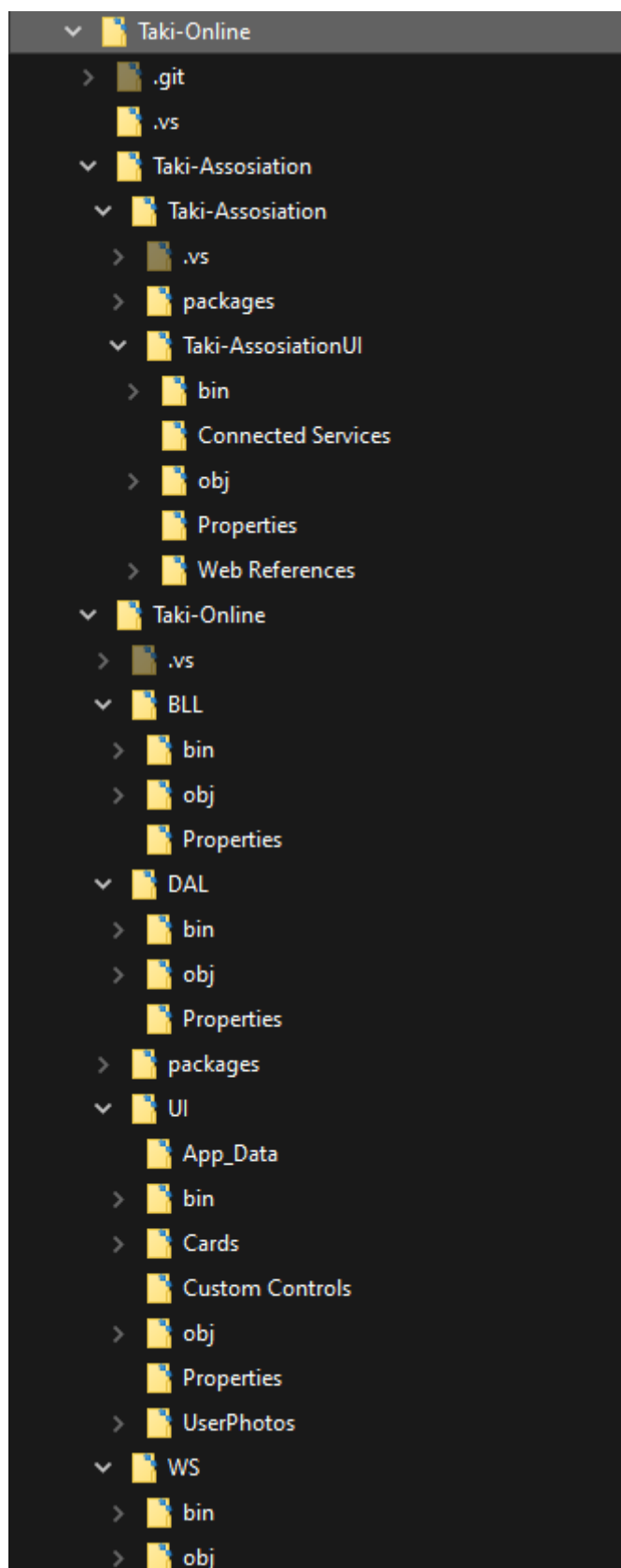
Users in games - טבלת שחקנים במשחקים

	User ▼	Game ▼	XP Added ▼	elo Added ▼	Has Won ▼	Cards In Han ▼
	Alonzo	10	0	-20	<input type="checkbox"/>	5
	AttackTitan	9	0	-40	<input type="checkbox"/>	6
	Avi	11	0	-10	<input type="checkbox"/>	3
	Founder	38	0	0	<input checked="" type="checkbox"/>	0
	Founder	39	0	0	<input checked="" type="checkbox"/>	0
	Founder	79	50	15	<input checked="" type="checkbox"/>	0
	Founder	80	50	13	<input checked="" type="checkbox"/>	0
	Founder	81	12	-1	<input type="checkbox"/>	3
	itai192	9	0	-30	<input checked="" type="checkbox"/>	0
	itai192	38	0	0	<input checked="" type="checkbox"/>	0
	itai192	39	0	0	<input checked="" type="checkbox"/>	0
	itai192	79	12	-15	<input type="checkbox"/>	3
	itai192	80	10	-13	<input type="checkbox"/>	4

	שם שדה	סוג נתונים
User	שם המשתמש של השחקן	טקסט קצר
Game	מספר המשחק	מספר
XP Added	כמות נקודות הניסיון שהשחקן צבר	מספר
elo Added	השינוי בדיחוג של השחקן	מספר
Has Won	האם השחקן ניצח במשחק	כן/לא
Cards In Hand	כמות הקלפים ביד של השחקן	מספר

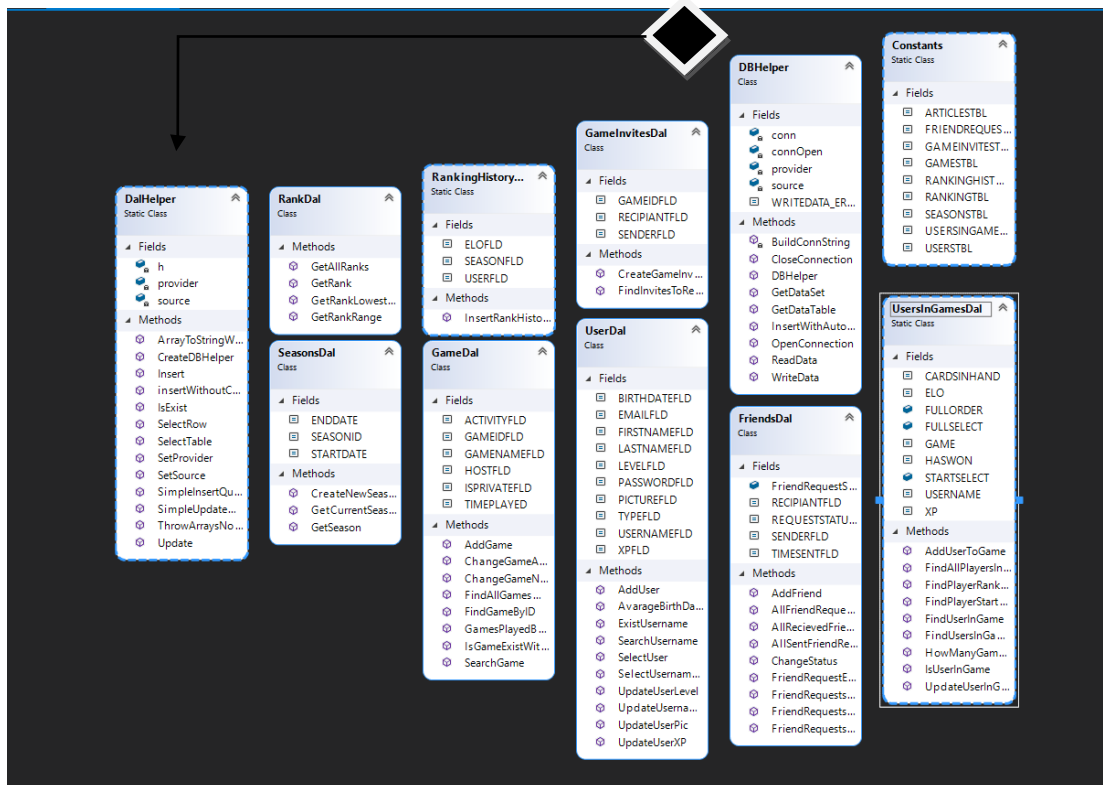
תמונת/תמונות קשרי הגומלין מתוך מסד הנתונים (DSD)





DAL •

○



○ Constants-מחלקת קבועים, מכילה שמות של טבלאות

```

public static class Constants
{
    public const string USERSTBL = "Users";
    public const string ARTICLESTBL = "Articles";
    public const string GAMEINVITESTBL = "[Game Invitations]";
    public const string FRIENDREQUESTSTBL = "[Friends Request Statuses]";
    public const string GAMESTBL = "Games";
    public const string RANKINGHISTORYTBL = "[Ranking History]";
    public const string SEASONSTBL = "Seasons";
    public const string USERSINGAMESTBL = "[Users in Games]";
    public const string RANKINGTBL = "Ranking";
}
    
```

○

○ DBHelper המחלקה שניגשת לבסיס הנתונים

```

public class DBHelper
{
    //Constants
    public const int WRITEDATA_ERROR = -1;
    //class member variables
    private OleDbConnection conn;
    private string provider;
    private string source;
    private bool connOpen;

    //Construct the object with a provided provider and source
    1 reference
    public DBHelper(string provider, string source){...}

    //This function builds the connection string to be used to open connection.
    1 reference
    private string BuildConnString(){...}
    //This function is closing the connection unless it is already closed!
    7 references
    public void CloseConnection(){...}
    //Open connection to the database.
    4 references
    public void OpenConnection(){...}
    //Execute UPDATE or INSERT sql commands and return number of rows affected.
    //return WRITEDATA_ERROR on failure
    3 references
    public int WriteData(string sql){...}
    //Execute SELECT sql commands and return a reference to an OleDbDataReader
    //if execution fails return null
    2 references
    public OleDbDataReader ReadData(string sql){...}
    //This function should be used for inserting a single record into a table in the database with an autonumber key. the format of the sql must be
    //INSERT INTO <TableName> (Fields...) VALUES (values...)
    //the function return the autonumber key generated for the new record or WRITEDATA_ERROR if fail
    1 reference
    public int InsertWithAutoNumKey(string sql){...}
    //This function reads from the database a data table fully cached in memory using a standard SQL SELECT statement.
    //The function returns the data table or null on failure.
    3 references
    public DataTable GetDataTable(string sql){...}

    //This function reads from the database a data set fully cached in memory using an array of standard SQL SELECT statements.
    //The function returns the data set or null on failure. The table names inside the dataset are sql1, sql2,...
    0 references
    public DataSet GetDataSet(string[] sql){...}
}

```

DalHelper-מחלקה המשתמשת בdbhelper ועוזרת בבניית וביצוע שאילתות
 למסד הנתונים

```

    /// <summary>
    /// Uses DBHelper and makes it easier to execute queries
    /// </summary>
    65 references
    public static class DalHelper
    {

        ///a string of the provider of the dal helper
        private static string provider;
        ///a string of the source of the dal helper
        private static string source;
        ///the object of DBHelper dal helper is using
        private static DBHelper h;
        ///sets the provider of the DBHelper
        1 reference
        public static void SetSource(string path)...
        ///sets the provider of the DBHelper
        1 reference
        public static void SetProvider(string prov)...
        ///creates an instance of the DBHelper
        1 reference
        public static void CreateDBHelper()...
        ///works for all insert sql that also needs to create an int key and returns that key
        3 references
        public static int Insert(string sql) ...
        13 references
        public static DataRow SelectRow(string sql) //works for all select sql that selects a single row and returns that row...
        /// <summary>
        /// a method that takes a string of arrays and returns a string of them separated by commas
        /// </summary>
        /// <param name="arr">the array</param>
        /// <returns>a string of the contents of the array separated by commas</returns>
        3 references
        public static string ArrayToStringWithCommas(string[] arr)...
        /// <summary>
        /// throws an arrays not in same length exception
        /// </summary>
        /// <param name="arr1">an array expected to be the same length as the other array</param>
        /// <param name="arr2">an array expected to be the same length as the other array</param>
        2 references
        public static void ThrowArraysNotInSameLengthExeption(Array arr1,Array arr2)...

        /// <summary>
        /// a method that returns a simple insert query based over the parameters
        /// </summary>
        /// <param name="table">table name</param>
        /// <param name="columns">the columns you want to insert in order</param>
        /// <param name="values">the values to be inserted into the columns in order</param>
        /// <returns>a simple insert sentence based over the parameters</returns>
        7 references
        public static string SimpleInsertQuery(string table, string[] columns, string[] values)...
        /// <summary>
        /// a method that returns a simple update query based over the parameters
        /// </summary>
        /// <param name="table">the table to update</param>
        /// <param name="columns">the columns you want to update in order</param>
        /// <param name="values">the values to be updated into the columns in order</param>
        /// <param name="whereCondition">the where condition (without the word WHERE)</param>
        /// <returns>a simple update sentence based over the parameters</returns>
        4 references
        public static string SimpleUpdateQuery(string table, string[] columns, string[] values, string whereCondition)...
        /// <summary>
        /// a method that returns a simple update query based over the parameters
        /// </summary>
        /// <param name="table">the table to update</param>
        /// <param name="setVals">a string of the set values</param>
        /// <param name="whereCondition">the where condition (without the word WHERE)</param>
        5 references
        public static string SimpleUpdateQuery(string table, string setVals, string whereCondition)...
        /// <summary>
        /// executes an update query
        /// </summary>
        /// <param name="sql">an sql query</param>
        /// <returns>returns how many columns have been updated</returns>
        8 references
        public static int Update(string sql)//works for all update sql and checks if the db changed, if it did than the update worked and returns true, else false...
        13 references
        public static DataTable SelectTable(string sql)//works for all select sql that selects a table and returns that table...
        4 references
        public static bool insertWithoutCreatingID(string sql)//works for all insert sql that doesnt need to create an auto key...
        6 references
        public static bool IsExist(string sql)//gets a select sql for a row and checks if such row exists...
    }

```

FriendsDal-מחלקה האחראית על טבלת החברים


```

public class FriendsDal
{
    //field names so it is easier to make queries
    public const string SENDERFLD = "Sender", RECIPIANTFLD = "Recipient", REQUESTSTATUSFLD = "[Request status]", TIMESENTFLD = "[Time Sent]";
    //a friend request query beginning that was useful to make a constant (readonly)
    public static readonly string FriendRequestSQL = $"SELECT * FROM " + Constants.FRIENDREQUESTSTBL;
    /// <summary>
    /// return's whether a friend request between two users exists
    /// </summary>
    1 reference
    public static bool FriendRequestExists(string username1, string username2)[...]
    /// <summary>
    /// Creates a new friend request from sender to recipient
    /// </summary>
    1 reference
    public static void AddFriend(string sender, string recipient)[...]
    /// <summary>
    /// changes the status of a friend request between sender and recipient
    /// </summary>
    3 references
    public static void ChangeStatus(string sender, string recipient, int status)[...]
    /// <summary>
    /// returns all friend request records associated with a user
    /// </summary>
    0 references
    public static DataTable AllFriendRequestsOfUser(string username)[...]
    /// <summary>
    /// returns all records of friend requests that a user has sent
    /// </summary>
    0 references
    public static DataTable AllSentFriendRequestsOfUser(string username)[...]
    /// <summary>
    /// returns all records of friend requests that a user has recieved
    /// </summary>
    0 references
    public static DataTable AllRecievedFriendRequestsOfUser(string username)[...]
    /// <summary>
    /// returns all records of friend requests with specific status a user is associated with
    /// </summary>
    0 references
    public static DataTable FriendRequestsWithStatus(string username, int status)[...]
    /// <summary>
    /// returns all records of friend requests with specific status a user has sent
    /// </summary>
    2 references
    public static DataTable FriendRequestsWithStatusSent(string username, int status)[...]
    /// <summary>
    /// returns all records of friend requests with specific status a user has recieved
    /// </summary>
    3 references
    public static DataTable FriendRequestsWithStatusRecieved(string username, int status)[...]
}

```

GameDal-מחלקה האחראית על טבלת המשחקים

```

public class GameDal
{
    //field names so it is easier to make queries
    public const string GAMEIDFLD = "[Game ID]", TIMEPLAYED = "[Time Played]", ACTIVITYFLD = "Activity", GAMENAMEFLD = "[Game Name]", ISPRIVATEFLD="[Is Private]", HOSTFLD = "Host";
    /// <summary>
    /// Inserts a game and returns it's id
    /// </summary>
    1 reference
    public static int AddGame(string gameName, string host,int activity)[...]
    /// <summary>
    /// changes a game's activity status
    /// </summary>
    1 reference
    public static void ChangeGameActivity(int activity, int gameId)[...]
    /// <summary>
    /// Changes a game's name
    /// </summary>
    1 reference
    public static void ChangeGameName(string gameName, int gameId)[...]
    /// <summary>
    /// finds all games with specific activity
    /// </summary>
    0 references
    public static DataTable FindAllGamesWithActivity(int activity)[...]
    /// <summary>
    /// finds if a game with given name exists with a given activity
    /// </summary>
    1 reference
    public static bool IsGameExistWithNameAndActivity(string name, int activity)[...]
    /// <summary>
    /// Finds a game with specific ID and returns it's record
    /// </summary>
    1 reference
    public static DataRow FindGameByID(int ID)[...]
    /// <summary>
    /// searches a game with name that contains search term and also has given activity status
    /// </summary>
    1 reference
    public static DataTable SearchGame(string searchTerm,int activity)[...]
    /// <summary>
    /// returns how many games were played in a certin date range
    /// </summary>
    1 reference
    public static int GamesPlayedBetweenDates(DateTime start, DateTime end)[...]
}

```

GameInvitesDal-מחלקה האחראית על טבלת הזמנות למשחקים

```

public class GameInvitesDal
{
    //field names so it is easier to make queries
    public const string RECIPIANTFLD = "Recipient", SENDERFLD = "Sender", GAMEIDFLD="Game ID";
    /// <summary>
    /// inserts a new game invitation to recipient from sender to game with game id
    /// </summary>
    1 reference
    public static void CreateGameInvite(string sender, string recipient, int gameID)...
    /// <summary>
    /// finds game invitations to recipient where game has specific activity status
    /// </summary>
    1 reference
    public static DataTable FindInvitesToRecipientWhereGameWithActivity(string recipient, int activity )...
}

```

RankDal-מחלקה האחראית על טבלת הליגות

```

public class RankDal
{
    /// <summary>
    /// gets all records that describe ranks
    /// </summary>
    1 reference
    public static DataTable GetAllRanks()...
    /// <summary>
    /// gets the range of elos of a certin rank
    /// </summary>
    1 reference
    public static string GetRankRange(int rankID)...
    /// <summary>
    /// gets the lowest elo of a rank
    /// </summary>
    1 reference
    public static int GetRankLowestValue(int rankID)...
    /// <summary>
    /// returns a record that describes the rank
    /// </summary>
    2 references
    public static DataRow GetRank(int rankID)...
}

```

RankingHistoryDal-מחלקה האחראית על טבלת הדירוגים לאורך ההיסטוריה

```
public static class RankingHistoryDal
{
    //field names so it is easier to make queries
    public const string USERFLD = "[User]", SEASONFLD = "Season", ELOFLD = "[Season Start Elo]";
    /// <summary>
    /// inserts a record of user start elo in seasons to rankingHistory table
    /// </summary>
    1 reference
    public static void InsertRankHistory(string username, int season, int elo)...
```

SeasonsDal-המחלקה האחראית על טבלת העונות

```
public class SeasonsDal
{
    //field names so it is easier to make queries
    public const string SEASONID = "[Season ID]", STARTDATE = "[Start Date]", ENDDATE = "[End Date]";
    /// <summary>
    /// inserts a new season and returns it's ID
    /// </summary>
    1 reference
    public static int CreateNewSeason(DateTime startDate, DateTime endDate)...
```

UserDal-מחלקה האחראית על טבלת המשתמשים

```
public class UserDal
{
    //field names so it is easier to make queries
    public const string APFLD = "xp", LEVELFLD = "[level]", FIRSTNAMEFLD = "[First Name]", LASTNAMEFLD = "[Last Name]", BIRTHDATEFLD = "[Date of Birth]", USERNAMEFLD = "Username", EMAILFLD = "Email", PASSWORDFLD = "[Password]", TYPEFLD = "Type", PICTUREFLD = "Picture";
    /// <summary>
    /// adds a user
    /// </summary>
    1 reference
    public static void AddUser(string email, string password, int type, string fname, string lname, DateTime bdate, string username)...
```

UsersInGamesDal-מחלקה האחראית על טבלת המשתמשים במשחקים

```

public static class UsersInGamesDal
{
    //a query that selects every player in each season and it's elo score, and rank,
    public static string FULLSELECT = "SELECT R2.[Rank ID], R2.[Rank Name], T2.Lowest, T2.[User], T2.Season, T2.ELO FROM Ranking AS R2 INNER JOIN"
    +"(SELECT T.ELO, T.Season, T.[User], Max(R1.[Lowest Elo]) AS Lowest FROM Ranking AS R1 INNER JOIN (SELECT ((IIF(ISNULL(Sum([Users in Games].[elo Added])),0,"
    +" Sum([Users in Games].[elo Added]))+[Ranking History].[Season Start Elo]) AS ELO, [Ranking History].Season, [Ranking History].User FROM (Seasons INNER JOIN"+
    "[Ranking History] ON Seasons.[Season ID] = [Ranking History].Season) LEFT JOIN (Games RIGHT JOIN [Users in Games] ON Games.[Game ID] = [Users in Games].Game) ON "+
    "[Ranking History].User = [Users in Games].User GROUP BY [Ranking History].Season, [Ranking History].User, [Ranking History].[Season Start Elo]) AS T ON"+
    " R1.[Lowest Elo]< T.ELO GROUP BY T.ELO, T.Season, T.[User]) AS T2 ON T2.Lowest = R2.[Lowest Elo]";
    //part of the query that orders it by elo score
    public static string FULLORDER = "ORDER BY T2.ELO DESC";
    //a query that selects the rank and elo from season start
    public static string STARTSELECT = "SELECT R2.[Rank ID], R2.[Rank Name], T2.Lowest, T2.[User], T2.Season, T2.ELO FROM Ranking AS R2 "+
    "INNER JOIN (SELECT T.ELO, T.Season, T.[User], Max(R1.[Lowest Elo]) AS Lowest FROM Ranking AS R1 INNER JOIN "+
    "(SELECT[Ranking History].[Season Start Elo] AS ELO, [Ranking History].Season, [Ranking History].User FROM[Ranking History]) AS "+
    "T ON R1.[Lowest Elo] < T.ELO GROUP BY T.ELO, T.Season, T.[User]) AS T2 ON T2.Lowest = R2.[Lowest Elo]";
    //names of fields as constants so it would be easier to make queries
    public const string USERNAME = "[User]", GAME = "Game", XP = "[XP Added]", ELO = "[Elo Added]", HASWON = "[Has Won]", CARDSINHAND = "[Cards In Hand]";
    /// <summary>
    /// adds a user to users in games table
    /// </summary>
    1 reference
    public static void AddUserToGame(string username, int GameID)[...]
    /// <summary>
    /// updates a record of a user in a game
    /// </summary>
    1 reference
    public static void UpdateUserInGame(string username, int GameID, int xpAdded,int eloAdded, bool HasWon,int CardsInHand)[...]
    /// <summary>
    /// a method finds a player's rank(and elo) in a given season
    /// </summary>
    2 references
    public static DataRow FindPlayerRankInSeason(int season, string username)[...]
    /// <summary>
    /// a method that find's all player's in a given rank in a given season
    /// </summary>
    1 reference
    public static DataTable FindAllPlayersInRankInSeason(int rankID, int season)[...]
    /// <summary>
    /// finds the start rank (and elo) of a player in a season
    /// </summary>
    2 references
    public static DataRow FindPlayerStartRankInSeason(int season, string username)[...]
    /// <summary>
    /// finds the records of users in a specific game in order of winners.
    /// </summary>
    2 references
    public static DataTable FindUsersInGame(int gameID)[...]
    /// <summary>
    /// returns whether a user is (or was) in a game
    /// </summary>
    1 reference
    public static bool IsUserInGame(string user, int gameID)[...]
    /// <summary>
    /// a user in game record of user in game
    /// </summary>
    1 reference
    public static DataRow FindUserInGame(string user, int gameID)[...]
    2 references
    public static int HowManyGamesUserWonOrLost(string username, bool won)[...]
}

```

○

BLL •


```

public class Action
{
    //the action type
    11 references
    public ActionType type { get; }
    //card associated with the action
    6 references
    public Card card { get; }
    //Player who did the action
    7 references
    public Player player { get; }
    /// <summary>
    /// action constructor
    /// </summary>
    4 references
    public Action(ActionType type, Card card, Player player)
    {
        this.type = type;
        this.card = card;
        this.player = player;
    }
}

```

○ BLL_Helper-מחלקת פעולות סטטיות ○

```

27 references
public static class BLL_Helper
{
    /// <summary>
    /// returns a list of how many days were played in the last 30 days, where index 0 is today and 29 is 29 days ago
    /// </summary>
    1 reference
    public static List<int> GamesPlayedInMonth()...
    /// <summary>
    /// gets a season object describing current season
    /// </summary>
    1 reference
    public static Season GetCurrentSeason()...
    /// <summary>
    /// Gets a datatable of all players whose rank this season is corresponding to the rank id
    /// </summary>
    2 references
    public static DataTable GetAllPlayersInRankThisSeasonDataTable(int rankID)...
    /// <summary>
    /// gets all rank objects
    /// </summary>
    3 references
    public static List<Rank> GetAllRanks()...
    /// <summary>
    /// Sets the source and provider of the dal helper
    /// </summary>
    1 reference
    public static void SetSourceAndProvider(string source, string provider)...
    /// <summary>
    /// Creates the db helper needed in dal helper
    /// </summary>
    2 references
    public static void CreatedBHelperInDalHelper(string source, string provider)...
    /// <summary>
    /// a method which takes a data table and field in that table, and turns all values in that field into a list of type T
    /// </summary>
    8 references
    public static List<T> DataTableToList<T>(DataTable dt, string field)...
    /// <summary>
    /// a method which takes two lists of type T and unites them into one list
    /// </summary>
    1 reference
    public static List<T> UniteLists<T>(List<T> l1, List<T> l2)...
    /// <summary>
    /// a method which checks and returns whether a user exists
    /// </summary>
    2 references
    public static bool UserExists(string username)...
    /// <summary>
    /// A method which turns a data table of user details and turns it into a list of user objects
    /// </summary>
    1 reference
    public static List<User> UserListFromDataTable(DataTable dt)...
    /// <summary>
    /// A methods which creates and returns a user list from a username list
    /// </summary>
    6 references
    public static List<User> UserListFromUsernameList(List<string> usernames)...
}

```

○ IPlayerBroadcast-ממשק המתאר תשדירים לשחקנים

```

/// <summary>
/// an interface that defines a broadcast sent from game to
/// players that tell the player object what happened in the game
/// used to help make the game asynchronous
/// </summary>
11 references
public interface IPlayerBroadcast
{
    /// <summary>
    /// a method which executes the broadcast over the player
    /// </summary>
    5 references
    void DoBroadcast(Player player);
}

```

○ ChangeTurnBroadcast-מחלקה המתארת תשדיר שאחראי לשנות את התור
○ אצל שחקן

```
/// <summary>
/// a broadcast which changes the turn of the taki game
/// implements the IPlayerBroadcast interface
/// </summary>
2 references
public class ChangeTurnBroadcast : IPlayerBroadcast
{
    //describes the turn to go to
    private int turn;
    /// <summary>
    /// changes player's turn to specified turn
    /// </summary>
    5 references
    public void DoBroadcast(Player player)...
    /// <summary>
    /// a simple constructor for the changeturn broadcast
    /// </summary>
    1 reference
    public ChangeTurnBroadcast(int turn)...
}
```

○ ChangeOrderBroadcast-מחלקה המתארת תשדיר שמשנה את סדר המשחק

```
/// <summary>
/// a broadcast which changes the turn order of the taki game
/// implements the IPlayerBroadcast interface
/// </summary>
1 reference
public class ChangeOrderBroadcast : IPlayerBroadcast
{
    //the order of the game(true means to add to the list and false to subtract)
    private bool order;
    /// <summary>
    /// a simple constructor for the changeorder broadcast
    /// </summary>
    0 references
    public ChangeOrderBroadcast(bool order)...
    /// <summary>
    /// changes player's order to specified order
    /// </summary>
    5 references
    public void DoBroadcast(Player player)...
}
```

○ ActionBroadcast-מחלקה המתארת תשדיר של פעולה של שחקן במשחק


```

/// <summary>
/// a broadcast which describes a player's action
/// implements the IPlayerBroadcast interface
/// </summary>
5 references
public class ActionBroadcast : IPlayerBroadcast
{
    // the action describing the action a player has done
    public Action action;
    //a simple player object describing the player who did the action
    2 references
    public SimplePlayer player...
    //the type of the action done
    0 references
    public ActionType type...
    /// <summary>
    /// a simple constructor for the action broadcast
    /// </summary>
    2 references
    public ActionBroadcast(Action action)...
    /// <summary>
    /// does the action described in the action object
    /// </summary>
    5 references
    public void DoBroadcast(Player player)...
}

```

EndGameBroadcast-מחלקה המתארת תשדיר של סוף המשחק ○

```

/// <summary>
/// a broadcast which tells the player the game has ended
/// implements the IPlayerBroadcast interface
/// </summary>
2 references
public class EndGameBroadcast:IPlayerBroadcast
{
    /// <summary>
    /// an empty constructor for the end game broadcast
    /// </summary>
    1 reference
    public EndGameBroadcast()...
    /// <summary>
    /// changes the players gameended property to true
    /// </summary>
    5 references
    public void DoBroadcast(Player player)...)
}

```

○ Card-מחלקה אבסטרקטית המיצגת קלף טאקי

```

/// <summary>
/// an abstract class describing a taki card
/// </summary>
44 references
public abstract class Card
{
    ///the card's color
    11 references
    public Color color { get; protected set; }
    /// <summary>
    /// an empty constructor for a card,
    /// for inheritance purposes
    /// </summary>
    0 references
    public Card()...
    /// <summary>
    /// an empty constructor for a card which takes the color of the card,
    /// for inheritance purposes
    /// </summary>
    2 references
    public Card(Color color)...
    /// <summary>
    /// a method which checks if a card can be put on this card
    /// </summary>
    7 references
    public virtual bool CanPutOn(Card card)...
    /// <summary>
    /// a method which checks if this card can be put on a card
    /// </summary>
    1 reference
    public virtual bool CanBePutOn(Card card)...
    /// <summary>
    /// a virtual method that describes the end ability of a card
    /// </summary>
    6 references
    internal protected virtual void EndAbility(Game game)...
    /// <summary>
    /// a virtual method that describes the start ability of a card
    /// </summary>
    1 reference
    internal protected virtual void StartAbility(Game game)...
    /// <summary>
    /// an exception describing where this card can be put
    /// </summary>
    2 references
    internal protected virtual Exception WhereCanPutCard()...
    /// <summary>
    /// a method which processes player action
    /// </summary>
    1 reference
    internal protected virtual void ProcessPlayerAction(Game game, Action a)...
}

```

○ IGetCardText-ממשק המתאר קלף שאפשר להציג באמצעות טקסט ○

```

/// <summary>
/// an interface that describes cards which you can get their text
/// </summary>
4 references
public interface IGetCardText
{
    /// <summary>
    /// a method that returns the card's text
    /// </summary>
    3 references
    string GetCardText();
}

```

Reverse-מחלקה המתארת קלף שנה כיוון

```

/// <summary>
/// a card class describing the change direction or reverse card.
/// </summary>
2 references
public class Reverse : Card
{
    /// <summary> a constructor for the reverse class
    0 references
    public Reverse(Color color) ...
    /// <summary>
    /// a method which returns whether you can put a card on this reverse card
    /// </summary>
    7 references
    public override bool CanPutOn(Card card) ...
    /// <summary>
    /// the end ability of reverse card, which changes direction of the game
    /// </summary>
    6 references
    internal protected override void EndAbility(Game game) ...
}

```

Stop- מחלקה המתארת קלף עצור

```

/// <summary>
/// a class describing the stop card
/// </summary>
1 reference
public class Stop : Card
{
    /// <summary>
    /// a method which returns whether you can put a card on this stop card
    /// </summary>
    7 references
    public override bool CanPutOn(Card card) ...
    /// <summary>
    /// the end ability of stop card, which changes skips a player's turn
    /// </summary>
    6 references
    internal protected override void EndAbility(Game game) ...
}

```

○ NumberCard - מחלקה המתארת קלף מספר

```

/// <summary>
/// a class describing the number card
/// </summary>
5 references
public class NumberCard : Card, IGetCardText
{
    //the maximum number of a number card
    const int MAXNUMBERCARD = 9;
    /// <summary>
    /// the get card text method, implements the IGetCardText
    /// </summary>
    3 references
    public string GetCardText() ...
    // the number on the card
    private int value;
    /// <summary>
    /// constructor for the number card
    /// </summary>
    2 references
    public NumberCard(Color color, int value) ...
    /// <summary>
    /// a method which returns whether you can put a card on this number card, same number or same color
    /// </summary>
    7 references
    public override bool CanPutOn(Card card) ...
    /// <summary>
    /// an exception describing where this card can be put in it's messege
    /// </summary>
    2 references
    protected internal override Exception WhereCanPutCard() ...
    /// <summary>
    /// to string method returns the color and value of the card
    /// </summary>
    2 references
    public override string ToString() ...
}

```

○ Elo - מחלקה סטטית שבעזרת הפעולות בה ניתן לשנות את הדירוג לפי שיטת Elo

```

public static class Elo
{
    /// <summary>
    /// score in this context means probability to win
    /// where 1 is an expected win
    /// 0.5 is an expected draw
    /// and 0 is an expected loss
    /// </summary>
    const int KFACTOR = 32;
    const int EXPONENTDIVISOR = 400;
    const int BASEOFEXPONENT = 10;
    /// <summary>
    /// Calculates and returns the expected score of a player against an opponent using their elo rating
    /// </summary>
    /// <param name="PlayerElo">The elo rating of the player who we calculate his expected score</param>
    /// <param name="OponentElo">The elo rating of the opponent of the player who we calculate his expected score</param>
    /// <returns>the expected score of the player</returns>
    1 reference
    public static double ExpectedScore(int PlayerElo, int OponentElo)[...]
    /// <summary>
    /// Calculates the change in elo rating of a player against an oppenent using their elos and the player's score
    /// </summary>
    /// <param name="PlayerElo">The elo rating of the player who we calculate his change in elo rating</param>
    /// <param name="OponentElo">The elo rating of the opponent of the player who we calculate his change in elo ratings</param>
    /// <param name="score">the score of the player in a game between the two players</param>
    /// <returns>the change in elo rating of the player</returns>
    4 references
    public static int ChangeInRating(int PlayerElo, int OponentElo, double score)[...]
}

```

Game-מחלקה המתארת משחק טאקי

```

public partial class Game
{
    //a boolean describing whether this game has ended
    0 references
    public bool GameEnded[...]
    //the game room object containg this game
    5 references
    public GameRoom gameRoom[...]
    //number of cards in hand
    const int NUMCARDS=8;
    //a stack of cards describing the deck
    private Stack<Card> deck;
    //a stack of cards describing the pile
    private Stack<Card> pile;
    //the list of players in order of turns when order is true
    private List<Player> players;
    //the current turn of the game
    6 references
    public int turn { get; private set; }
    //the active card, can be null, if not null, processes player's actions
    private Card activeCard;
    //a boolean which describes whether the last player has put a card yet
    3 references
    public bool hasLastPlayerPutCard
    { get; internal set; }
    //the game's leading card, top of the pile
    3 references
    public Card leadingCard[...]
    //the order of game, true means the next player is the next in the list and false, previous
    5 references
    public bool order {get; internal set; }
    //extra cards when you draw
    2 references
    public int penelty {get; internal set;}
}

```

```

/// <summary>
/// changes the active card of the game to the leading card and trigger's it's start ability
/// </summary>
1 reference
internal void ChangeActiveCard()...
/// <summary>
/// clears the active card, and if game has ended, does the end progression
/// </summary>
1 reference
internal void ClearActiveCard()...
/// <summary>
/// returns whether there is a winner to this game
/// </summary>
1 reference
private bool IsEnded()...
/// <summary>
/// a method which ends the game
/// </summary>
1 reference
private void EndProgression()...
/// <summary>
/// returns the player list sorted by card amount
/// </summary>
1 reference
private List<Player> SortedPlayerList()...
/// <summary>
/// compares two player's card amount, used for sorting
/// </summary>
1 reference
private int ComparePlayerCards(Player x, Player y)...
```

```

private int ComparePlayerCards(Player x, Player y)...
/// <summary>
/// returns the player whose it is it's turn now
/// </summary>
2 references
internal Player GetPlayerTurn()
{
    return players[turn];
}
/// <summary>
/// a game constructor, gets the game room which contains it
/// </summary>
2 references
public Game(GameRoom gameRoom)...
//takes cards from deck to player
1 reference
internal void TakeCardsFromDeck(Player p)...
/// <summary>
/// tries to use end ability using player
/// </summary>
0 references
public void UseEndAbility(Player player)...
/// <summary>
/// goes to next turn according to game order
/// </summary>
3 references
internal void NextTurn()...
/// <summary>
/// a method that broadcasts broadcasts to all players in game
/// </summary>
2 references
private void Broadcast(IPlayerBroadcast broadcast)...

```



```

/// <summary>
/// a method that broadcasts action broadcasts to all players in game
/// </summary>
3 references
private void BroadcastAction(Action action)...
/// <summary>
/// method that does action if leagal
/// </summary>
2 references
public void TryDoAction(Action action)...
/// <summary>
/// puts a card using action
/// </summary>
1 reference
private void PutCard(Action action)...
//reshuffles deck using cards in pile
2 references
private void Reshuffle()...
/// <summary>
/// shuffles deck
/// </summary>
2 references
private void ShuffleDeck()...
/// <summary>
/// returns whether a user is in game
/// </summary>
1 reference
private bool IsUserInGame(User user)...
/// <summary>
/// adds player to game, and returns it
/// </summary>
1 reference
public Player AddPlayer(User user)...
    /// <summary>
    /// returns player list as simple player list
    /// </summary>
    1 reference
    public List<SimplePlayer> GetSimplePlayerList()...
    /// <summary>
    /// updates userlists in all players
    /// </summary>
    1 reference
    public void UpdatePlayerListsInAllPlayers()...
}

```

GameInvite - מחלקה המתארת הזמנה למשחק ○

```

public class GameInvite
{
    //game id
    2 references
    public int gameId[...]
    //sender of game invite
    1 reference
    public string sender[...]
    //recipient of game invite
    1 reference
    public string recipient[...]
    //game room associated with game invitation
    0 references
    public GameRoom gameRoom[...]
    /// <summary>
    /// a constructor for game invite object
    /// </summary>
    1 reference
    public GameInvite(int gameId, string sender, string recipient) [...]
    /// <summary>
    /// a constructor for game invite object from data row in game invites table
    /// </summary>
    1 reference
    public GameInvite(DataRow dr) [...]
}

```

○ GameRoom-מחלקה המתארת חדר משחק ○

```

18 references
public class GameRoom
{
    //dictionary containing all games in the key of their game id
    private static Dictionary<int,Game> games = new Dictionary<int,Game>();
    //the name of the host
    private string _host;
    //object of host
    1 reference
    public User host...
    //the name of the game
    private string _gameName;
    //the status of the game
    private GameStatus _status;
    //users in the game
    3 references
    public List<string> users...
    //the status of the game
    7 references
    public GameStatus status...
    //the name of the game
    0 references
    public string gameName...
    //id of the game
    18 references
    public int GameID...
    //the game associated with the game room object
    2 references
    private Game game...
    /// <summary>
    /// adds a user to the game
    /// </summary>
    1 reference
    public Player AddUserToGame(User user)...
}

```

```

    /// <summary>
    /// validates the name of the game
    /// </summary>
    2 references
    public void ValidateName(string name)...
    /// <summary>
    /// gameroom constructor
    /// </summary>
    1 reference
    public GameRoom(User host, string gameName)...
    /// <summary>
    /// game room constructor which returns an already existing game room by game ID
    /// </summary>
    3 references
    public GameRoom(int ID)...
    /// <summary>
    /// constructs a game room using a data row
    /// </summary>
    1 reference
    public GameRoom(DataRow dr)...
    /// <summary>
    /// updates the game room information
    /// </summary>
    2 references
    public void UpdateRoom()...
    /// <summary>
    /// updates the game room information using a datarow
    /// </summary>
    2 references
    public void UpdateRoom(DataRow dr)...
}

```

○ Player-מחלקה המתארת שחקן

```

public class Player
{
    //game id of associated game
    1 reference
    public int GameID[...]
    //whether game has ended
    3 references
    public bool GameEnded[...]
    //user asociated with a game
    17 references
    public User user[...]
    //number of cards in player hand
    10 references
    public int numberOfCards[...]
    //the list of cards describing the hand
    internal List<Card> hand;
    //the leading card in the game, updates asynchronously to the game
    4 references
    public Card leadingCard[...]
    //the order of the game, updates asynchronously to the game
    2 references
    public bool order[...]
    //the turn of the game, updates asynchronously to the game
    2 references
    public int turn[...]
    //a list of all other players in order
    private List<SimplePlayer> _players;
    //a list of all other players in order
    4 references
    public List<SimplePlayer> players[...]
    //a queue of broadcasts to do
    private Queue<Game.IPlayerBroadcast> broadcastsToDo;
    //the game associated with the game
    private Game game;
}

```

```

/// <summary>
/// finds and returns the simple player in the player list, equal to player
/// </summary>
2 references
private SimplePlayer FindSimplePlayer(SimplePlayer player)...
/// <summary>
/// gets the hand of the player
/// </summary>
1 reference
public List<Card> GetHand()...
/// <summary>
/// adds a card to a simple player
/// </summary>
1 reference
internal void AddACardToSimplePlayer(SimplePlayer player)...
/// <summary>
/// subtracts a card to a simple player
/// </summary>
1 reference
internal void SubtractACardFromSimplePlayer(SimplePlayer player)...
/// <summary>
/// updates the simple player list
/// </summary>
1 reference
internal void UpdatePlayerList(List<SimplePlayer> players)...
/// <summary>
/// checks whether player has a card
/// </summary>
1 reference
public bool HasCard(Card card)...
/// <summary>
/// returns whether player has broadcasts to do
/// </summary>
3 references
public bool HasUndoneBroadcasts()...

```

```

/// <summary>
/// returns the broadcast that will execute next
/// </summary>
1 reference
public Game.IPlayerBroadcast NextBroadcast()...
/// <summary>
/// does broadcast and removes it from broadcast queue
/// </summary>
1 reference
public void DoBroadcast()...
/// <summary>
/// Tries to take a card
/// </summary>
1 reference
internal Card TakeCard(Card card)...
/// <summary>
/// adds a card to player's hand
/// </summary>
3 references
internal void AddCardToHand(Card c)...
/// <summary>
/// adds a broadcast to broadcast queue
/// </summary>
3 references
internal void AddBroadcast(Game.IPlayerBroadcast broadcast)...
/// <summary>
/// create a simple player based of this player
/// </summary>
2 references
public SimplePlayer ToSimplePlayer()...
/// <summary>
/// player constructor
/// </summary>
1 reference
internal Player(Game game, User user)...
```

```

    /// <summary>
    /// tries to draw cards from deck
    /// </summary>
    1 reference
    public void DrawCards()...
    /// <summary>
    /// tries to put card
    /// </summary>
    1 reference
    public void putCard(Card card)...
    /// <summary>
    /// invites another user to the game
    /// </summary>
    1 reference
    public void invitePlayer(string recipiant)...
    /// <summary>
    /// equals method, checks if users are equal
    /// </summary>
    11 references
    public override bool Equals(object obj)...
}

```

○ SimplePlayer-מחלקה המייצגת שחקן עם פחות גישה לנתונים


```

/// <summary>
/// a simpler player class, to encapsulate player
/// </summary>
22 references
public class SimplePlayer
{
    //the player object associated with this simple player
    private Player player;
    //user associated with simple player;
    3 references
    public User user...
    //checks whether the player is described in this simple player
    0 references
    public bool IsPlayer(Player player)...
    /// <summary>
    /// simple player constructor
    /// </summary>
    1 reference
    public SimplePlayer(Player p, User user)...
    /// <summary>
    /// equals override, checks if players are equal
    /// </summary>
    11 references
    public override bool Equals(object obj)...
    /// <summary>
    /// the number of cards the simple player has
    /// </summary>
    4 references
    public int NumberOfCards...
}

```

Rank-מחלקה המתארת ליגה

```

/// <summary>
/// class used to describe ranks
/// </summary>
16 references
public class Rank
{
    ///id of the rank
    3 references
    public int ID { get; private set; }
    ///name of the rank
    6 references
    public string name { get; private set; }
    ///lowest elo in the rank
    2 references
    public int lowestElo { get; private set; }
    /// <summary>
    /// returns a string describing the range of elo ratings in the rank
    /// </summary>
    1 reference
    public string GetRange()...
    /// <summary>
    /// rank constructor by rank ID
    /// </summary>
    3 references
    public Rank(int ID)...
    /// <summary>
    /// rank constructor from datarow from ranks table
    /// </summary>
    2 references
    public Rank(DataRow dr)...
    /// <summary>
    /// to string method returns names
    /// </summary>
    2 references
    public override string ToString()...
}

```

Season-מחלקה המתארת עונה ○

```

/// <summary>
/// a class that describes a game season
/// </summary>
6 references
public class Season
{
    // season id
    2 references
    public int SeasonID[...]
    // start date of season
    2 references
    public DateTime StartDate[...]
    // end date of season
    2 references
    public DateTime EndDate[...]
    /// <summary>
    /// season constructor, adds a new season
    /// </summary>
    0 references
    public Season(DateTime StartDate, DateTime EndDate)[...]
    /// <summary>
    /// season constructor, gets season information from ID
    /// </summary>
    1 reference
    public Season(int ID)[...]
    /// <summary>
    /// season constructor, gets season information from data row
    /// </summary>
    1 reference
    public Season(DataRow dr)[...]
}

```

○ User-מחלקה המתארת משתמש

```

/// <summary>
/// a class describing a user
/// </summary>
68 references
public class User
{
    //xp needed to pass level
    const int InitailXpNeeded=50;
    //initial elo of a user
    const int InitailElo = 200;
    //username
    private string _username;
    //username property
    41 references
    public string username { ... }
    //User's rank this season
    3 references
    public Rank rank { ... }
    //User's elo this season
    10 references
    public int elo { ... }
    //the user's type
    4 references
    public UserType type { get; }
    //user's email
    2 references
    public string email { get; }
    //user's birthdate
    3 references
    public DateTime BirthDate { get; }
    //user's level
    private int _level;
    //user's picture
    private string _picture;
}

```

```

//user's level
6 references
public int level [...]
//user's expiriance level
private int _xp;
//user's expiriance level
2 references
public int xp [...]
//user's first name
3 references
public string fName { get; }
//user's last name
3 references
public string lName {get;}
//user's picture profile path
6 references
public string picture [...]
//user's accepted friends list
4 references
public List<string> AcceptedFriends[...]
/// <summary>
/// levels user up if he can level up
/// </summary>
1 reference
public void LevelUpIfCan() [...]
/// <summary>
/// xp needed to reach next level, doubles every level
/// </summary>
2 references
public int XPUntilNextLevel() [...]
// users that this user has declined friend requests from
2 references
public List<string> DeclinedFriends[...]
//names of users which have requested to be friends and not answered yet
4 references
public List<string> UnopenedFriendRequests[...]

```

```

//names of users which ypu have requested to be friends with and not answered yet
2 references
public List<string> UnopenedSentFriendRequests[...]
//a list of game invites
1 reference
public List<GameInvite> activeGameInvites[...]
/// <summary>
/// user constructor, using a user's username
/// </summary>
5 references
public User(string username) [...]

/// <summary>
/// user constructor, using a user's username and password
/// </summary>
2 references
public User(string username, string password) [...]
/// <summary>
/// returns whether a user is or was in a game with game id
/// </summary>
1 reference
public bool IsUserInGame(int gameId) [...]
/// <summary>
/// tries to send a friend request and returns a textual response
/// </summary>
1 reference
public string AddFriend(string username) [...]
/// <summary>
/// tries to accept a friend request from user with username
/// </summary>
1 reference
public void AcceptFriendRequestFrom(string username) [...]
/// <summary>
/// tries to decline a friend request from user with username
/// </summary>
1 reference
public void DeclineFriendRequestFrom(string username) [...]
/// <summary>
/// user constructor which inserts a new user into user table using details
/// </summary>
2 references
public User(string username, UserType type, string email, DateTime BirthDate, string fName, string lName, string password)
/// <summary>
/// user constructor that uses data row as data source
/// </summary>
2 references
public User(DataRow dr) [...]
/// <summary>
/// statistics of user in game
/// </summary>
1 reference
public UserStatsInGame statsInGame(int gameId) [...]
/// <summary>
/// returns whether user equals to other user, using users' usernames
/// </summary>
11 references
public override bool Equals(object obj) [...]
/// <summary>
/// to string method, returns username
/// </summary>
2 references
public override string ToString() [...]
/// <summary>
/// returns how many games this user has won
/// </summary>
1 reference
public int HowManyGamesWon() [...]

```

```

    /// <summary>
    /// returns how many games this user has lost
    /// </summary>
    1 reference
    public int HowManyGamesLost()...
}

```

○ UsersInGameStats-מחלקה המתארת תוצאות של שחקן במשחק

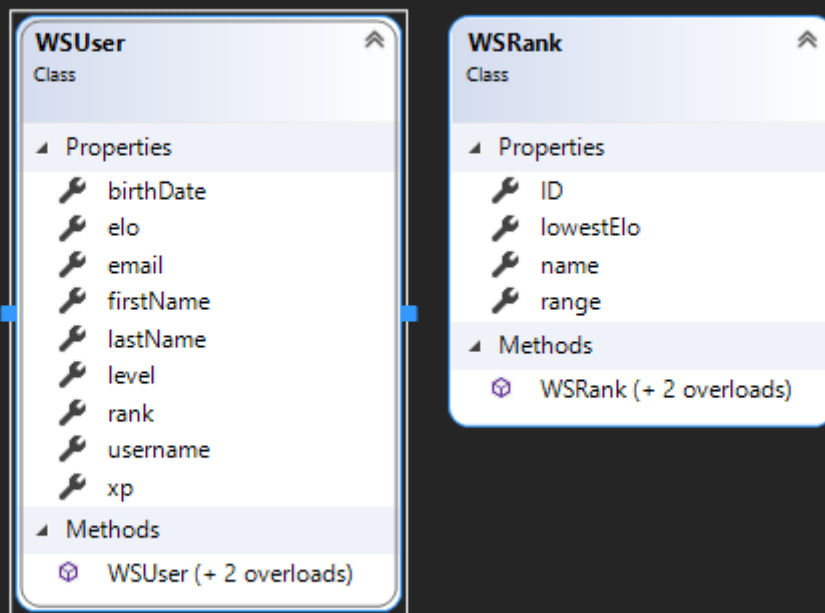
○

```

/// <summary>
/// describes game information about user
/// </summary>
11 references
public class UserStatsInGame
{
    //username of statistic
    2 references
    public string username...
    //how many cards user has in hand
    1 reference
    public int cardsInHand...
    //has user won
    1 reference
    public bool hasWon...
    //game ID of game
    1 reference
    public int GameID...
    //the change in elo of user
    2 references
    public int eloChanged...
    //the change in xp of user
    2 references
    public int xpChanged...
    /// <summary>
    /// a constructor that finds user in game statistics
    /// </summary>
    1 reference
    public UserStatsInGame(string user, int gameID):this(UsersInGamesDal.FindUserInGame(user,gameID))
    {
        ...
    }
    /// <summary>
    /// a constructor that uses data row as data source
    /// </summary>
    2 references
    public UserStatsInGame(DataRow dr)...
}

```

WS •



WSUser - מחלקה המתארת משתמש בשירות הרשת


```

public class WSUser
{
    //first name of user
    1 reference
    public string firstName{get; set;}
    //last name of user
    1 reference
    public string lastName{get; set;}
    //username of user
    1 reference
    public string username { get; set; }
    //email of user
    1 reference
    public string email { get; set; }
    //rank name of user
    1 reference
    public string rank{get; set;}
    //level of user
    1 reference
    public int level{get; set;}
    //xp of user
    1 reference
    public int xp{get; set;}
    //elo of user
    1 reference
    public int elo { get; set; }
    //birth date of users
    1 reference
    public DateTime birthDate{get; set;}
    /// <summary>
    /// empty constructor
    /// </summary>
    0 references
    public WSUser()...
    /// <summary>
    /// constructor using user
    /// </summary>
    3 references
    public WSUser(User user) ...
    /// <summary>
    /// constructor using username
    /// </summary>
    1 reference
    public WSUser(string username) : this(new User(username))
    { }
}

```

WSRank-מחלקה המתארת ליגה ○

```

public class WSRank
{
    //rank id
    2 references
    public int ID { get; set; }
    //rank name
    1 reference
    public string name { get; set; }
    //rank lowest elo
    1 reference
    public int lowestElo { get; set; }
    //rank range
    1 reference
    public string range { get; set; }

    /// <summary>
    /// empty constructor
    /// </summary>
    0 references
    public WSRank()...
    /// <summary>
    /// constructs wsrank using rank
    /// </summary>
    3 references
    public WSRank(Rank rank)...
    /// <summary>
    /// constructs wsrank using rank ID
    /// </summary>
    0 references
    public WSRank(int ID) ...
}

```

שירותי הרשת באתר

שירותי רשת שהאפליקציה מספקת

האתר מספק שירותי הרשת המקנה לצורך חלק מהפונקציונליות של האתר, כגון הרשמה, התחברות, החזרה של חברים, מציאת כל הליגות, מציאת השחקנים בליגה, פרטי משתמש למשתמשים שונים. בתמונה ניתן לראות את כל שירותי הרשת שהאתר מספק

```
public class TakiWebService : System.Web.Services.WebService
{
    /// <summary>
    /// a method that signs a user in
    /// </summary>
    [WebMethod(enableSession:true)]
    0 references
    public bool SignIn(string username, string password)...
    /// <summary>
    /// a method that signs up a user
    /// </summary>
    [WebMethod]
    0 references
    public bool SignUp(string username, string password, string email, DateTime birthDate, string fName, string lName)...
    /// <summary>
    /// a method that returns a list of a connected user's friends
    /// </summary>
    [WebMethod(enableSession: true)]
    0 references
    public List<WSUser> GetFriends()...
    /// <summary>
    /// a method that gets user data using a user's name
    /// </summary>
    [WebMethod]
    0 references
    public WSUser GetUserDetails(string username)...
    /// <summary>
    /// a method which returns all ranks
    /// </summary>
    [WebMethod]
    0 references
    public List<WSRank> GetAllRanks()...
    /// <summary>
    /// a method which returns a list of all users in a rank
    /// </summary>
    [WebMethod]
    0 references
    public List<WSUser> GetAllUsersInRank(WSRank rank)...
    /// <summary>
    /// a method which returns whether the session is signed in
    /// </summary>
    [WebMethod(enableSession:true)]
    0 references
    public bool IsSessionConnected()...
    /// <summary>
    /// a method that returns the signed in user
    /// </summary>
    [WebMethod(enableSession:true)]
    0 references
    public WSUser GetConnectedUser()...
    /// <summary>
    /// a private static method to turn lists of users to WSUsers
    /// </summary>
    2 references
    private static List<WSUser> UserListToWSUserList(List<User> users)...
    /// <summary>
    /// a private static method to turn lists of Ranks to WSRanks
    /// </summary>
    1 reference
    private static List<WSRank> RankListToWSRankList(List<Rank> ranks)...
```

שירותי רשת נצרכים ממקור חיצוני

צריכת שירותי הרשת שהאתר מספק מתבצעת באתר Taki Assosiation, שהוא אתר של נתונים על הפרוייקט הראשי. האתר המדובר הוא מערכת קטנה במסגרת עבודת הגמר, המדגים כיצד ניתן

לצורך את שירותי הרשת שמספקת המערכת הראשית.

Taki Assosiation

[Home](#) [Sign up](#) [Sign in](#)

כאשר לוחצים על sign in בא לידי ביטוי שירות הרשת המאפשר למשתמש להתחבר:

Username:

Password:

new user?

כאשר מתחברים לאתר האתר מעביר את המשתמש לדף המשתמש בו מגיעים לידי ביטוי שירות הרשת המאפשר לראות פרטי משתמש מחובר, ושירות הרשת המחזיר את רשימת החברים של המשתמש המחובר:

firstName: itai192
lastName: lustig
username: itai192
email: itailustig@gmail.com
rank: Noob
level: 5
xp: 5
elo: 34
birthDate: 9/30/2020

Friends:

Username	
Avi	<input type="button" value="See User Details"/>
Alonzo	<input type="button" value="See User Details"/>
yossi	<input type="button" value="See User Details"/>
yana	<input type="button" value="See User Details"/>
tamar9	<input type="button" value="See User Details"/>
shira9	<input type="button" value="See User Details"/>
Founder	<input type="button" value="See User Details"/>

בלחיצה על אחד המשתמשים מרשימת החברים, המשתמש יועבר לדף פרטי משתמשים אחרים בו מתממש שירות הרשת המספק מידע על כל משתמש:

firstName: yana
lastName: lustig
username: yana
email: yanalustig@gmail.com
rank: Beginner
level: 1
xp: 1
elo: 210
birthDate: 1/1/1973

[Home](#) [Leaderboard](#) [Log Out](#) [Account](#)

בלחיצה על כפתור הleaderboard מתממשים שירות הרשת המאפשר לראות את כל הליגות ואת כל השחקנים בליגה:

Username	Elo Raiting	
TheLegend27	130	See User Details
Founder	127	See User Details
TheBeast3	120	See User Details
Avi	90	See User Details
RedScarf	80	See User Details
Alonzo	80	See User Details
AttackTitan	60	See User Details
itai192	34	See User Details

Noob

Noob

Beginer

Intermidate

Master

Grand Master

Card Wizard

ומשם גם אפשר להגיע לדף פרטי המשתמש שכבר ראינו

בנוסף לכך, משתמש לא מחובר יכול להגיע לדף ההרשמה ולהירשם לאתר בעזרת שירות הרשת המאפשר זאת:

Email

Username:

Password:

First Name:

Last Name:

Birth Date:

≤

April 2021

≥

Sun

Mon

Tue

Wed

Thu

Fri

Sat

28

29

30

31

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

1

2

3

4

5

6

7

8

Select Year:

2021

SIGN UP

מפת האתר

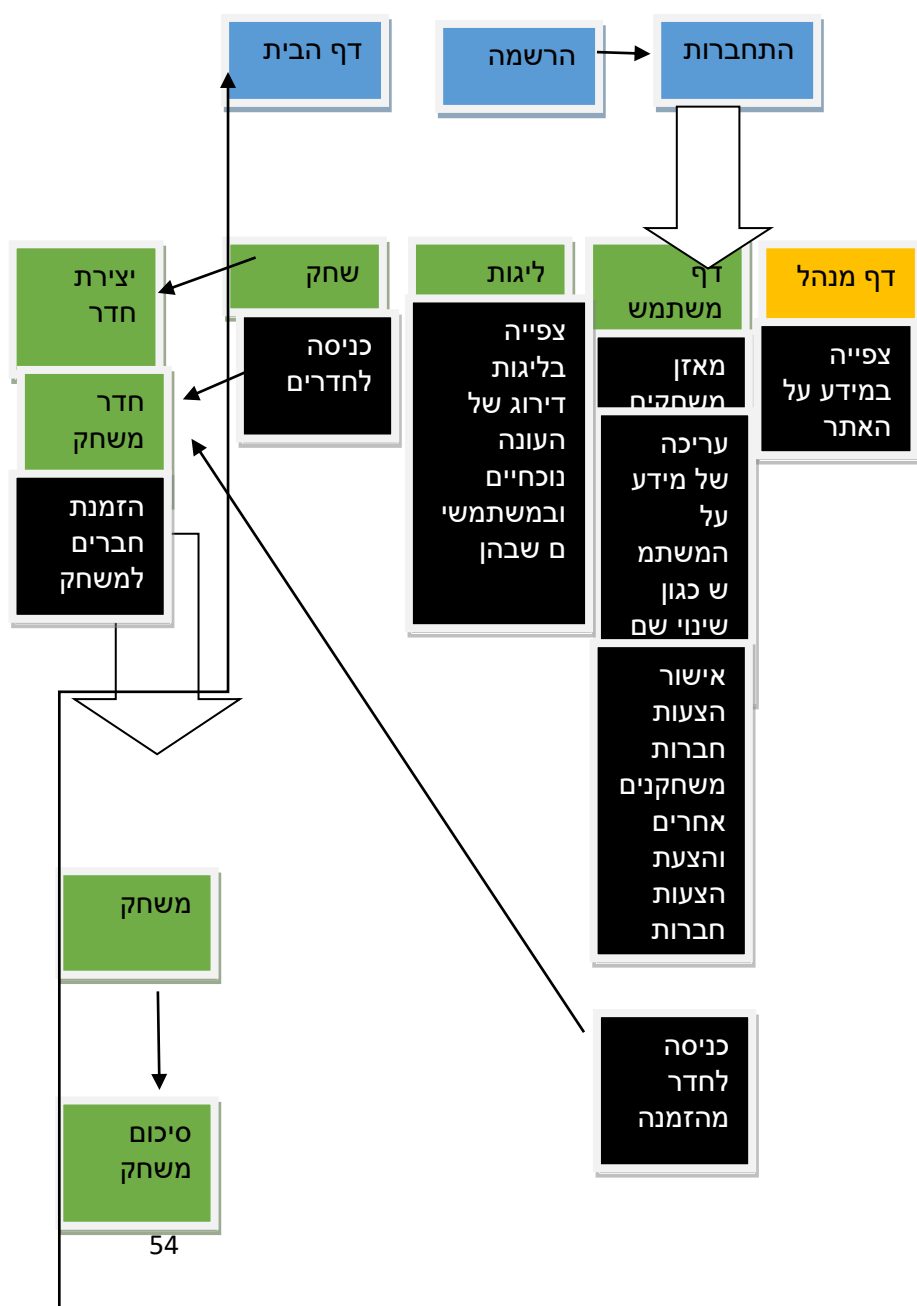
להלן מוצג תרשים הזרימה האפשרי באתר:

עמוד שכל משתמש יכול לראות

מידע או פעולה שאפשר לעשות ב עמוד מעל

עמוד שרק משתמשים מחוברים יכולים להיכנס אליו

עמוד שרק מנהל יכול להיכנס אליו



מדריך האפליקציה לשחקן

תאור האפשרויות למשתמש באפליקציה. יש ללוות את ההסברים בצילומי המסכים המתאימים. יש צורך להיות ענייניים ולא להאריך יתר על המידה (הציון אינו נמדד בכמות הדפים וגודל האותיות)

יש להתחיל באפשרויות הפתוחות בפני משתמש מזדמן, לתאר את דרך הרישום ולהמשיך באפשרויות למשתמשים הרשומים.

אם יש יותר מ-2 סוגי משתמשים, יש לכלול מדריך משתמש לכל אחד מהסוגים.

מדריך למשתמש - שחקן

דף כניסה והרשמות

כאשר שחקן נכנס לאתר, ניתנת לו האפשרות להירשם כשחקן חדש ע"י לחיצה על הכפתור

Sign up



לאחר לחיצה על הכפתור Sign up, ייפתח לשחקן עמוד ההרשמה בו הוא יידרש למלא את פרטיו.

Email: Please insert your email address
 Username: Please insert your username
 Password: Please insert your password
 Confirm Password: Please insert your password again
 First Name: Please insert your first name
 Last Name: Please insert your last name
 Profile picture: No file chosen
 Birth Date:

April 2021						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

 Please insert a valid birth date
 Select Year:

אחד הפרטים אותם יתבקש למלא השחקן החדש יהיה password ויתבקש לוודא את ה password פעם שניה במידה ויטעה יוצג לו הודעת שגיאה והוא יצטרך למלא שוב את ה password

Password:
 Confirm Password: Your passwords don't match

תינתן לשחקן החדש גם אפשרות להוסיף תמונת פרופיל מהמחשב האישי שלו

Profile picture: No file chosen

בסוף התהליך יש ללחוץ על הכפתור Sign up

על מנת להיכנס לדף החשבון שלו בפעם הראשונה השחקן יצטרך ללחוץ על כפתור Account

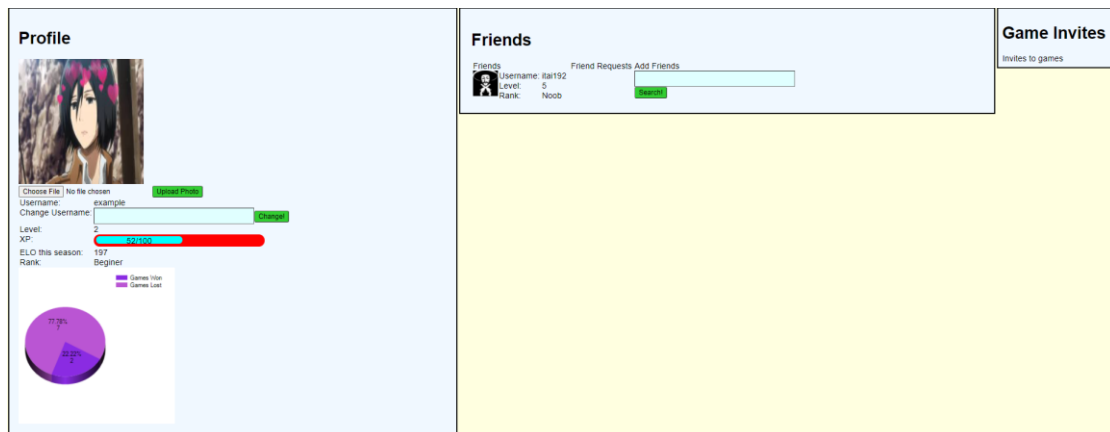


שחקן שכבר נרשם יכול להיכנס לדף החשבון שלו דרך כפתור ה SIGN IN, מילוי פרטיו ולחיצה על כפתור ה SIGN IN

A light blue rectangular box containing a login form. It has two input fields: 'Username:' with the text 'example' and 'Password:' with masked characters '.....'. Below the password field is a green 'SIGN IN' button, which is circled in red. Below the 'SIGN IN' button is the text 'new user?' followed by a green 'SIGN UP' button.

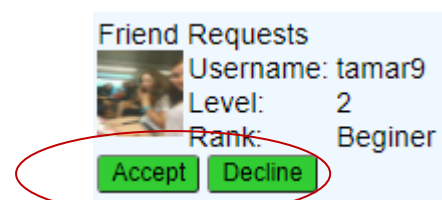
דף חשבון

בדף החשבון השחקן יכול לראות את פרטיו כשחקן, כולל הרמה, הניסיון, הדרוג, מספר המשחקים אותם ניצח או הפסיד והליגה שלו. כמו כן השחקן יכול לראות את חבריו, בקשות חברות חדשות וכן הזמנות להשתתף במשחקים

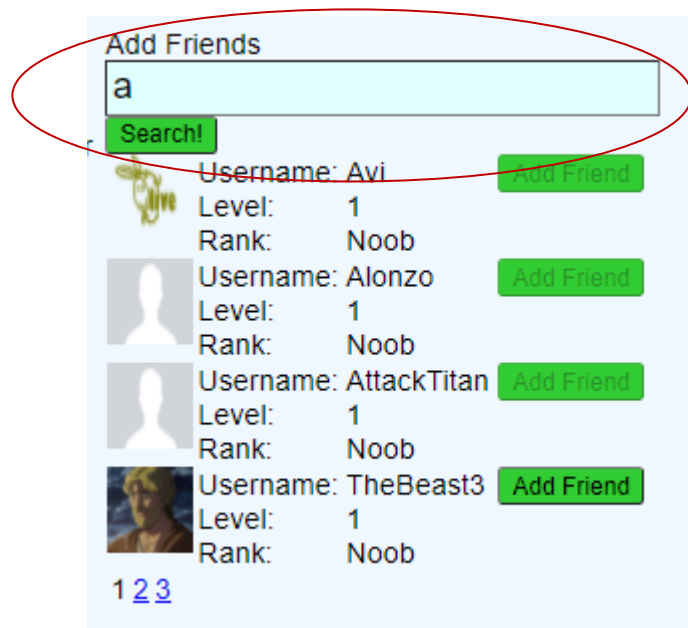


בעמוד החשבון, השחקן יכול לבצע את הפעולות הבאות:

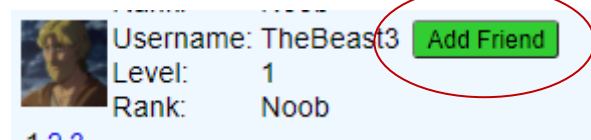
אישור ודחייה של בקשות חברות



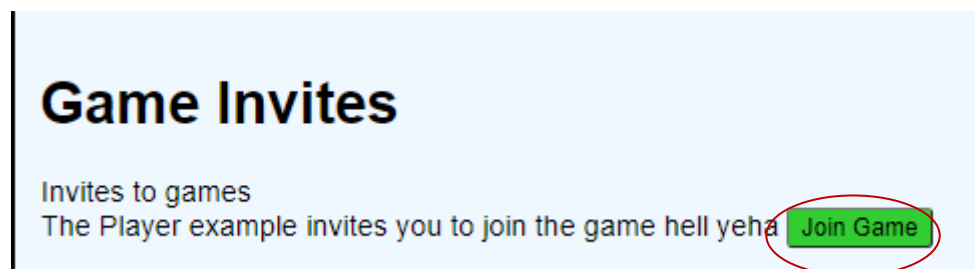
חיפוש משתמשים להצעת חברות



בקשת חברות



כניסה למשחקים דרך הצעה להזמנה למשחק



השתתפות במשחק

ניתן להשתתף משחק בכמה דרכים:

1. יצירת חדר משחק חדש והתחלת המשחק דרכו:

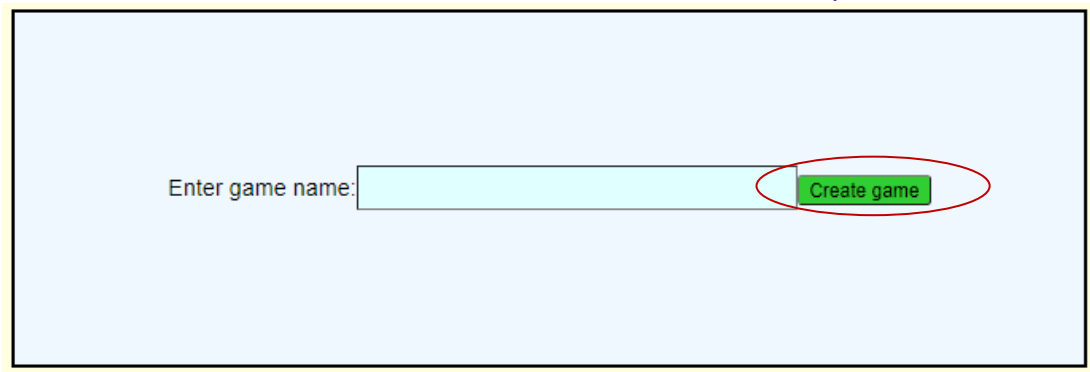
a. לחיצה על כפתור ה PLAY



b. לחיצה על הכפתור CREATE GAME



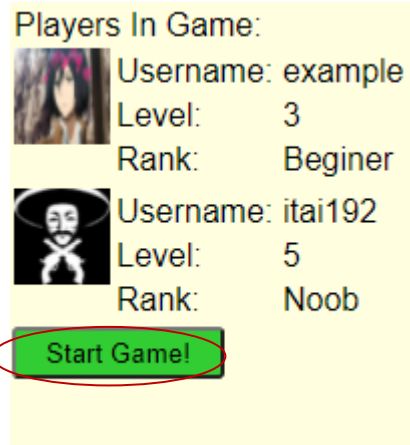
c. נתינת שם לחדר המשחק



d. הזמנת חבר למשחק או לחכות לשחקן אחר שייכנס לחדר המשחק

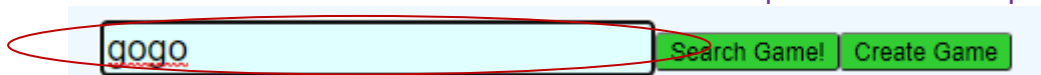


e. לחיצה על כפתור START GAME לצורך התחלת המשחק

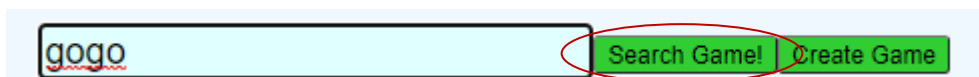


2. חיפוש חדר משחק לפי שם או לצפות בכל החדרים:

a. מזין את שם חדר המשחק



b. לחיצת על כפתור search

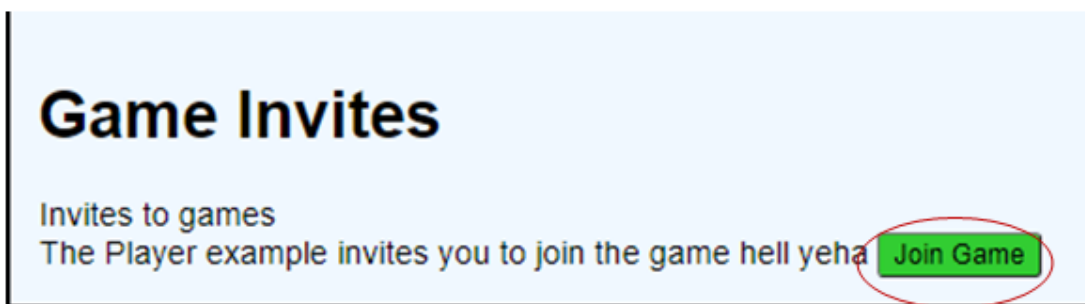


c. בחירת חדר משחק ע"י לחיצה על כפתור join game

3. כניסה לחדר משחק ששחקן אחר הזמין אותך

a. כניסה לדף החשבון

b. לחיצה על כפתור ה join game למשחק אליו השחקן מעוניין להצטרף



לוח ליגות

האתר מאפשר לשחקנים אף לצפות בליגות השונות אם אפשרות לסנן את הליגה בה השחקן מעוניין לצפות בהם הנתונים על השחקנים בכל ליגה והדירוג התחרותי שלהם.

Noob	
Username	Elo
TheLegend27	130
Founder	127
TheBeast3	120
Avi	90
RedScarf	80
Alonzo	80
AttackTitan	60
itai192	29

Beginer	
Username	Elo
tamar9	235
MidgetSpinner	220
yana	210
shira9	209
example	197
yossi	181

תחילת משחק

המשחק מתחיל לאחר שהשחקן שפתח את חדר המשחק על כפתור ה START GAME

קופה, לחיצה עליה
תוביל לניסיון לקיחת
קלף מהקופה

הקלף
המוביל

8

הקלפים ביד של השחקן,
לחיצה על קלף תוביל
לניסיון שימוש בו

נתוני השחקנים
והשחקן שתורו
מסומן באדום

1

4

5

3



4

7

4

4



Players

	Username: example
	Amount of Cards: 8

כל שחקן מקבל 8 קלפים מהקופה. כל שחקן בתורו יכול לשים קלף ע"פ חוקי משחק הטאקי המקובלים או לשלוף קלף חדש מהקופה. המשחק מראה בכל רגע נתון תור איזה שחקן לבצע פעולה.

במידה והשחקן מנסה לבצע פעולה לא חוקית המשחק יתריע לו על כך (ולא יבצע את הפעולה)



Players

	Username: example
	Amount of Cards: 8

You can only put blue 4 on cards that are either blue or 4

סיום משחק

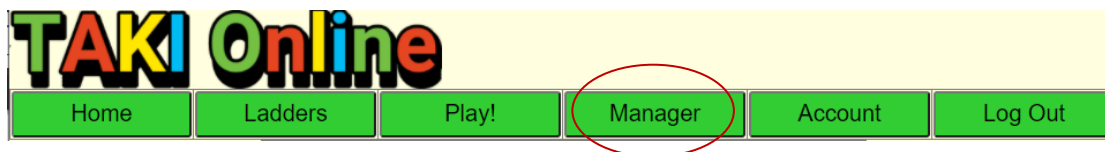
בסיום המשחק מוצג לשחקנים דף סיכום המשחק הכולל את מקום כל שחקן, כמות הקלפים שנותרו ביד, התוספת לניסיון והשנוי בנקודות הדירוג

Place		User	Cards in hand
#1		Username: example Level: 3 Rank: Beginner	0
#2		Username: itai192 Level: 5 Rank: Noob	5

Elo Change: +9
XP Change: +50

מדריך האפליקציה למנהל

המנהל נכנס לאתר בעזרת כפתור ה SIGN IN ובסרגל הניווט הוא יראה כפתור נוסף בשם .MANAGER



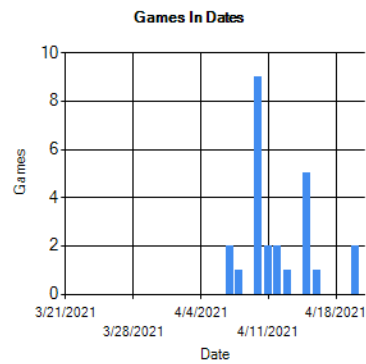
המנהל יוכל לצפות בנתונים נוספים אודות האתר והסטיסטיקות שבו. לדוגמא גרף המשחקים ששחקו בחודש הנוכחי

TAKI Online

[Home](#)[Ladders](#)[Play!](#)

Online:
2

Today:
2



רשימה ביבליוגרפית/ רשימת מקורות

אתר שבו נעזרתי לשם דיאגרמות UML

<https://docs.microsoft.com/en-us/visualstudio/ide/class-designer/how-to-add-class-diagrams-to-projects?view=vs-2019>

אתרים שבהם נעזרתי בכדי לממש את שיטת SME

http://www.tckerrigan.com/Misc/Multiplayer_Elo/

https://en.wikipedia.org/wiki/Elo_rating_system

<https://gobase.org/studying/articles/elo/>

אתר שבו נעזרתי בכל פעם שהיתה לי בעיה בקוד

<https://stackoverflow.com/>

אתרים שבהם נעזרתי בשביל עיצוב הקלפים

<https://www.figma.com/>

<https://css-tricks.com/>

<https://developer.mozilla.org/he/>

אתר בו השתמשתי בכדי ללמוד את AJAX

<https://www.c-sharpcorner.com/article/ajax-in-asp-net/>

אתר בו השתמשתי בכדי להיזכר בSQL

<https://www.w3schools.com/sql/default.asp>

במהלך הפרויקט חקרתי טכנולוגיות חדשות כמו AJAX בהם השתמשתי לצורך בניית המשחק, ולאפשר עדכון בזמן אמת לכל השחקנים שמשתתפים במשחק.

כמו כן חקרתי והשתמשתי בשיטת חישוב לדירוג המבוססת על שיטת ELO (שיטה שבה משתשים כדי לדרג שחקני שחמט בליגות בעולם) על מנת לחשב את הדירוג של השחקנים והתאמתם לליגות השונות.

במהלך הפרויקט נתקלתי באתגר של שליחת עדכונים לשחקנים באופן א-סינכרוני מהמשחק מה שאפשר לשחקנים לשחק בהתאמה גם אם תהיה איטיות ברשת. כדי לפתור את הבעיה יצרתי מחלקות ייחודיות לתשדירים לשחקנים כך שהשחקנים מתעדכנים באופן א-סינכרוני מהמשחק (שחקן חווה רק מה שקורה אצלו במחשב למעט פעולות שהוא עושה ללא תלות במה שהשחקנים האחרים רואים)

פתיחת הפרויקט הראשי:

בכדי לפתוח ולהריץ את הפרויקט יש לפתוח את התיקייה TakiOnline לאחר מכן יש לפתוח את הקובץ TakiOnline.sln בתוכנה Visual Studio .

לאחר שהתוכנה נפתחה ב Visual Studio , יש ללחוץ על הכפתור ISS Express שבראש

העמוד (לידו משולש ירוק) בכדי להריץ אותה.

לצורך כניסה בתור שחקן רשום באתר יש למלא את הפרטים הבאים בדף ה-Sign in :

שם משתמש: Founder

סיסמה: lovemikasa

לצורך כניסה בתור מנהל רשום לאתר יש למלא את הפרטים הבאים בדף ה-Sign in :

תעודת זהות : itai192

סיסמה : bruhr

פתיחת המיני פרויקט (שצורך את שירותי הרשת של הפרויקט הראשי):

בכדי לפתוח ולהריץ את המיני פרויקט יש ראשית לפתוח את הפרויקט הראשי לפי

ההוראות הכתובות מעל, אך במקום ללחוץ על Start, יש לפתוח את הפרויקט

WS המופיע בצד ימין של המסך.

לאחר מכן יש ללחוץ עם המקש הימני על הקובץ TakiWebService.asmx וללחוץ על View

In Browser.

כעת ניתן לפתוח את המיני פרויקט כך:

יש לפתוח את התיקייה Taki-Assosiation, ולאחר מכן שוב לפתוח תיקיה באותו שם לאחר מכן יש

לפתוח את הקובץ Taki-Assosiation.sln

בתוכנה Visual Studio.

לבסוף, יש ללחוץ על הכפתור ISS Express שבראש העמוד (לידו משולש ירוק) בכדי להריץ

את המיני פרויקט.