



נקודה. dot

יצירת משחק מנקודה לנקודה מתמונה שהעלה המשתמש

שם התלמידה: שירה חדאד

מספר זהות: 212823736

מקום לימודים: סמינר נוות ישראל

שם המנחה: הגב' גילה נוסבאום

תאריך הגשה: 06\2022 \ 16

תוכן עניינים

5.....	מבוא.....
6.....	הרקע לפרויקט:
6.....	תהליך המחקר:.....
6.....	סקירה ספרותית:
6.....	הפניות בהם נעזרתי:.....
7.....	אתגרים מרכזים:.....
8.....	דרכי הפתרון שנבדקו:.....
8.....	מטרות
9.....	יעדים
9.....	אתגרים :
10.....	מדדי הצלחה:
10.....	רקע תיאורטי:.....
10.....	מצב קיים:.....
12.....	תיאור החלופה הנבחרת:
17.....	אפיון המערכת:.....
17.....	סביבת פיתוח:.....
18.....	מודול מערכת:
18.....	אפיון פונקציונאלי:.....
19.....	ביצועים עיקריים:.....
19.....	אילוצים:
19.....	תיאור הארכיטקטורה
20.....	תיאור המרכיבים בפתרון:
21.....	תיאור פרוטוקולי התקשורת:.....
22.....	ניתוח ותרשים UML / Use cases של המערכת המוצעת
31.....	תרשים המחלקות:.....
31.....	מחלקות הDAL:.....
32.....	מחלקות הDTO:
32.....	מחלקות הBL:
33.....	תיאור המחלקות המוצעות:
33.....	❖ שכבת – Data Base -
33.....	❖ שכבת הגישה לנתונים:
33.....	❖ שכבת ה BL:.....
34.....	❖ מחלקות הData Object:
34.....	❖ מחלקות הקוד:
34.....	רכיבי ממשק:.....
34.....	תיכון המערכת:

שירה חדאד-מנקודה לנקודה

34.....	ארכיטקטורת המערכת:
34.....	תיכון מפורט:
34.....	חלופות לתיכון המערכת:
34.....	תיאור התוכנה:
35.....	תרשים המסכים:
35.....	תיאור המסכים:
35.....	מסכים לאורח:
35.....	מסכים למשתמש הרגיל:
36.....	מסכים למנהל:
36.....	מסכים לאורח:
37.....	מסכים למשתמש הרגיל:
44.....	מסכים למנהל:
47.....	קוד התכנית + תיעוד:
48.....	שלב 1:
50.....	שלב 2:
51.....	שלב 3:
54.....	שלב 4:
55.....	שלב 5:
56.....	שלב 6:
58.....	שלב 7:
60.....	קודים נוספים מצד השרת:
60.....	חישוב הדירוג:
61.....	קוד העלאת התמונה אל הצד שרת:
62.....	תיאור מסד הנתונים:
63.....	תרשים טבלאות:
63.....	משחקים-Picture_TBL:
63.....	משתמשים-User_TBL:
64.....	קטגוריות-Kategory_TBL:
64.....	סוגי מספור-Counting_TBL:
64.....	מדריך למשתמש:
64.....	בדיקות והערכה:
64.....	ניתוח יעילות:
65.....	פיתוחים עתידיים:
66.....	ביבליוגרפיה:

הצעת פרויקט י"ד\הנדסת תוכנה

סמל מוסד: 141119

שם מכללה: סמינר נוות ישראל

שם הסטודנט: שירה חדאד

תעודת זהות הסטודנט: 212823736

שם הפרויקט: ייצור אוטומטי של חידות מנקודה לנקודה (העברת קוים בין מספרים)

תיאור הפרויקט: המשתמש יוכל להעלות תמונה האתר יבדוק אם התמונה מתאימה לחידת העברת

קוים, יזהה את נקודות הענין בתמונה וייצור חידת מנקודה לנקודה הניתנת להורדה כקובץ.

הגדרת הבעיה האלגוריתמית: מציאת נקודות קיצון ונקודות ענין בתמונה. נקודות אלו יהוו את הנקודות המשתתפות בחידה.

רקע תיאורטי בתחום הפרויקט: אלגוריתמים לזיהוי גבולות ונקודות ענין. קיימים אלגוריתמים רבים העוסקים בנושא זה כגון: האלגוריתם של מורוביץ, שיטת tamasi & kanade למציאת מקסימום מקומי, Stephens & Harris ועוד במהלך כתיבת הפרויקט אחקור את ביצועי האלגוריתמים השונים ואבחר איזה מהם לממש.

תהליכים עיקריים בפרויקט: העלאת תמונה, בדיקה אם התמונה מתאימה, ניקוי רעשי רקע, מציאת נקודות קריטיות בתמונה, סגירה התמונה כמשחק בקובץ pdf עם אפשרות להורדה.

תיאור הטכנולוגיה: מודל שלוש השכבות

צד שרת: web api

שפת תכנות בצד השרת: C#

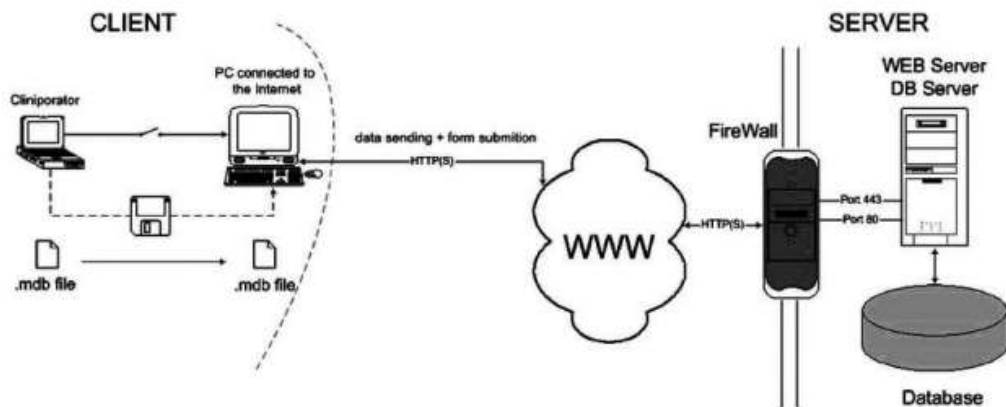
צד לקוח: react

שפת תכנות בצד הלקוח: javascript + html

מסד נתונים: sql server

פרוטוקולי תקשורת: המערכת מורכבת משרת IIS המריץ את האתר בסביבת ה-server,

מסד נתונים - database של sql-server, ממשק משתמש בצד הלקוח: דפדפן אינטרנט כלשהו: firefox, chrom, internet explorer



לוחות זמנים:

1. חקר מצב קיים: ספטמבר
2. הגדרת הדרישות: ספטמבר
3. אפיון המערכת: אוקטובר
4. אפיון בסיס הנתונים: אוקטובר
5. אלגוריתמים: אוקטובר-נובמבר
6. עיצוב המערכת: דצמבר
7. בניית התכנה: ינואר-פברואר
8. בדיקות: מרץ
9. הכנת תיק פרויקט: אפריל
10. הטמעת המערכת: מאי
11. הגשת פרויקט סופי: מאי

חתימת הסטודנט:

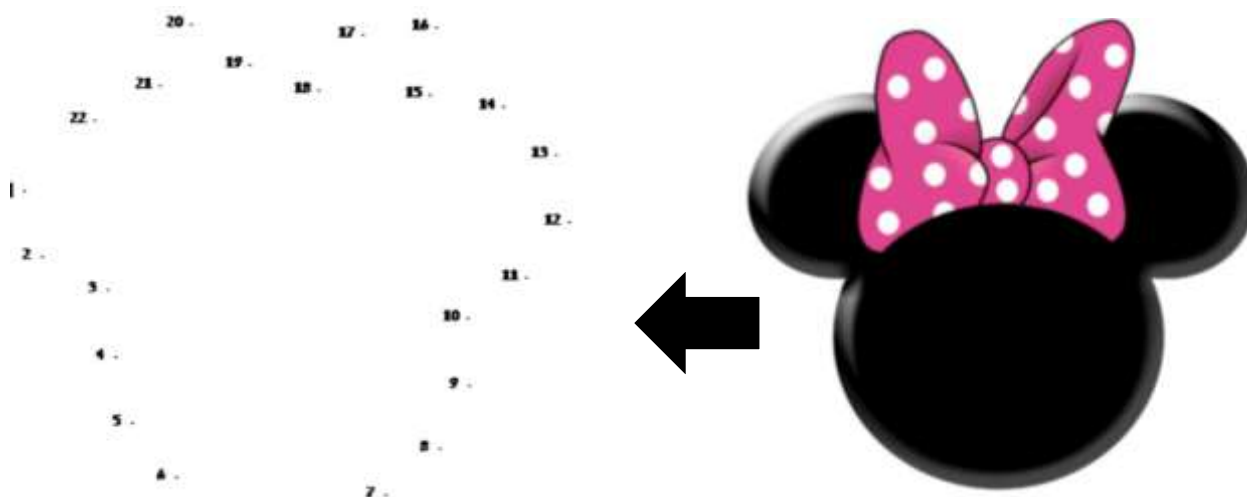
שירה חדאד- ספר פרויקט י"ד הנדסת תוכנה

מבוא

מאז ומעולם חיבבתי חידות היגיון וחשיבה. במיוחד אני אוהבת חידות שמשלבות גם הנאה מלבד המאמץ המחשבתי. ואכן הפרויקט שבחרתי עוסק בדיוק בתחומים אלו- הפרויקט בעצם יוצר משחק קו לקו מתמונה שהמשתמש בחר. החלק המחשבתי בפרויקט טמון בעיקר בכתיבת האלגוריתם והחלק המהנה שמור לצד הלקוח כמתבקש.

(משחק קו לקו הינו המעבר על נקודות החל מהנקודה הממוספרת ב1 ועד לנקודה הסופית לפי הסדר כאשר בסוף המעבר תיווצר תמונה)

להלן דוגמא לחידון על תמונת מיני מאוס:



לפני שהתחלתי לכתוב את הפרויקט ערכתי חקר קטן על המצב הקיים בשוק בתחום בו עוסק הפרויקט ומצאתי אתר שבו מאגר משחקי קו לקו מוכנים ניתנים להורדה, אולם מצאתי רק אתר אחד המאפשר למשתמש לבחור את התמונה שתיווצר במשחק...

קישור לאתר - המדריך:

<http://www.xn--7dbbqer5d.co.il>

קישור לאתר נוסף:

<https://www.ohmydots.com>

כמו"כ היו הרבה משחקים מודפסים כחוברת או קלפים אך גם שם לא ניתנה למשתמש האופציה להחליט על התמונה הסופית.

לאחר שנבחר נושא הפרויקט ניסיתי לשער מהם הקשיים הצפויים

אחד הקשיים הוא שהאלגוריתם לא יוכל להפוך כל תמונה שתעלה על לב המשתמש אלא רק תמונה המכילה עצם אחד בודד. אי לכך אני אבקש מהמשחק לבחור את העצם מתוך התמונה שהעלה במקרה כזה או שאודיע לו שהתמונה אינה ניתנת לשליחה.

במהלך פיתוח הפרויקט שמעתי שישנן גם חידונים המכילים כמה עצמים כאשר כל עצם ממוספר בנפרד, כמובן שנהניתי מהרעיון ושמחתי מאד לשכלל את הפרויקט לאפשרויות נוספות. פיתוח אופציה זו עומד בשלבי הסיום שלו וכולי תקווה שעד להגשת הפרויקט כבר אוכל להציגו

גם לאחר שוידאתי שהמשתמש שולח לנו תמונה בעלת עצם אחד עלי לנקות רעשי רקע ולמצוא אלגוריתם יעיל ביותר לזיהוי הקצוות של העצם שנבחר בתמונה, לשלוח לאלגוריתם ולהראות את התוצר למשתמש שישחק בהנאה.

והחל מכאן מתחיל המסע...

הרקע לפרויקט:

הפרויקט עוסק באלגוריתם הפיכת תמונה בעלת עצם יחיד למשחק קו לקו.

תוך מתן אפשרות למשתמש להגדיר את סוג הקפיצות בין נקודה לנקודה כך שתהיה אופציה גם ללימוד חוויתי דרך האתר. לימוד של a-b, רצף ה א-ב, סדרות חשבוניות והנדסאיות. הכול לפי בחירת המשתמש.

תהליך המחקר:

כרגע בשוק אין אף אתר המציע אופציה כזו של בחירת התמונה ע"י המשתמש אך היו הרבה אתרים המספקים מאגר מוכן של משחקים שהועלו ע"י המערכת הניתנים להורדה. נקודה זו מתחדשת בפרויקט שלי ע"י שהאתר מכיל תמונות שהועלו ע"י המערכת ובנוסף המשתמש יוכל לבחור תמונה וליצור ממנה משחק.

סקירה ספרותית:

הפניות בהם נעזרתי:

במהלך הפרויקט נעזרתי באינספור אתרי אינטרנט הן בחיפוש אחר קוד והן בלמידה אודות מה שמצאתי בקודים אלו.

בתחילה כאשר חקרתי את הנושא נעזרתי באתר המסביר על זיהוי פינות בתמונה ע"י בינה מלאכותית:

✓ https://elad.cs.technion.ac.il/wp-content/uploads/2018/02/Book_ImageProcessing.pdf

✓ https://www.openu.ac.il/lists/mediaserver_documents/academic/cs/Identifyingcornersvision.pdf

לאחר מכן כאשר החלטתי לבצע עיבוד תמונה נעזרתי בהרבה מאד אתרים להלן מספר הפניות:

✓ אלגוריתם קני לזיהוי קצוות:

<https://www.hamichlol.org.il>

✓ אלגוריתם מורבץ לזיהוי קצוות:

[Identifyingcornersvision.pdf](https://www.hamichlol.org.il/Identifyingcornersvision.pdf)

✓ אלגוריתם מציאת גבול ורק"חים בתמונה

[lecture4.pdf](#)

✓ אתר עיצוב צד הלוקח: MDB

<https://mdbootstrap.com/docs/react/>

✓ stackoverflow

✓ לעזרה באלגוריתמים השונים:

<https://stackoverflow.com/>

אתגרים מרכזים:

- זיהוי נקודות עיקריות בתמונה שהמשתמש בחר
- חלוקת התמונה לרכיבי קשירות במקרה של מספר עצמים
- התמודדות או זיהוי של תמונות שלא ניתנות להגדרה כעצם סגור.

- כתיבת אלגוריתמים יעילים
 - למידה עצמאית מתוך חומר תיאורטי ויישום החומר הנלמד
- הסיבות לבחירת הפרויקט: כפי שהוזכר לעיל חיפשתי פרויקט בעל אלגוריתם מאתגר שיגרום לי לסיפוק אמיתי בסוף השנה, שארגיש את החשיבה שלי עולה דרגה. אלגוריתם זה עונה לי על כל הדרישות שלי ומהווה אתגר רציני ומכובד לפרויקט גמר בהנדסאים.
- הפרויקט עונה על הצורך לתת למשתמש את האפשרות לבחור את התמונה וליצור ממנה חידון ברמת קושי סבירה ופתירה שפתרונה יתן תוצאה קרובה לקוי המתאר של התמונה המקורית

דרכי הפתרון שנבדקו:

- ✓ פתרון על ידי רשת נוירונים המסוגלת לזהות נקודות ענין/פינות בתמונה
- ✓ המרת תמונה ל double
- ✓ אלגוריתם למציאת פיקסלי הגבול/קונטור וההיקף ע"י מציאת נקודת ההתחלה השחורה ומעבר על שמונת שכניו החל מהשכן המערבי (עם כיוון השעון) עד למציאת פיקסל שחור.
- ✓ מציאת מרחק בין 2 נקודות ע"י פונקציית חישוב distance .
- ✓ אלגוריתם מורכב לזיהוי פינות (נקודות העניין) בתמונה ע"י מדידת השוונות בנקודה p מסוימת של התמונה. מגדירים חלון ריבועי סביב הנקודה ואז מזיזים את החלון בשמונה כיוונים שונים (אופקי אנכי ושני אלכסונים) ואז מסכמים את ריבועי ההפרשים בין הנקודות המתאימות בחלון המקורי והמוזז. השוונות המינימאלית מאלו שחושבו תהיה השוונות של נקודה p .
- ✓ אלגוריתם למציאת כל המרכיבים הקשירים בתמונה נתונה, והקניית תווית (label) בלעדית לכל הפיקסלים השייכים לאותו מרכיב קשיר. וע"י כך לקבל אזורים (regions) בתמונה שעשויים לייצג אובייקטים שונים.
- ✓ אלגוריתם דילול לקבלת קטעים קווים (lines) המהווים אומדן לקווים מרכזיים (mat) של קבוצות פיקסלים בתמונה בינארית. תוצאת האופרטור הינה סט דחוס של פיקסלים קשירים כך שנקודות הקצה נשמרות (שלד התמונה).

מטרות

- ✓ האתר יספק בניית משחק קלה ומהנה למשתמש
- ✓ חסכון בזמן ומאמץ מיותר מצד המשתמש.

- ✓ האתר ישלב אף למידת סדרות ורצפים שונים עפ"י בחירת המשחק.
- ✓ תינתן אפשרות למנהל האתר לנהל את מסד הנתונים ולעדכן לפי הצורך.
- ✓ המשחקים יהיו מוגשים לקשת רחבה של גילאים (כל משתמש יבחר את התמונה המתאימה לרמתו)
- ✓ המשתמש ירגיש שייכות לאתר ע"י העלאת התמונות שבחר

יעדים

- ✓ בניית אתר נעים ונח למשתמשים.
- ✓ האתר יכיל מסכים מעוצבים ומסודרים לרווחת המשתמש.
- ✓ האתר יחזיק ויתפעל מאגר גדול ביותר של תמונות הן מהעלאה אישית של משתמשים והן תמונות ראשוניות מהמערכת.
- ✓ המערכת תזהה את הנקודות מתוך תמונה לאחר ניקוי רעשים בצורה אופטימלית.
- ✓ חקירת הנושא של גילוי קצוות בתמונה בצורה הטובה ביותר.
- ✓ אצליח לקבל ולפענח גם תמונות בעלות יותר מעצם אחד
- ✓ אשלב למידת צירופים וסדרות עפ"י בחירת המשתמש
- ✓ ניתן יהיה לצרף דירוג לכל תמונה כדי להגביר את תחושת השייכות של המשתמש לאתר
- ✓ דרך נוספת ליצירת שייכות למשחקים היא שעל כל תמונה יופיע שם המשתמש שהעלה אותה.

אתגרים :

במהלך בניית האתר ניצבו בפני אתגרים שונים אשר אתגרו אותי לא מעט אבל כמובן

רכשתי בעקבתן ניסיון רב ומיומנויות חדשות. כמו:

- לימוד חומר חדש-עיבוד תמונה.
- כתיבת ממשק משתמש ברור ונוח.
- כתיבת אלגוריתם מהיר ויעיל ככל הניתן גם בתמונות מורכבות.
- חקירת נושא הפרויקט.
- פתירת באגים.
- לימוד והבנה של קודים קיימים יחד עם שימוש בהם.
- מילוי המספור עפ"י כל בקשה שהמשתמש יכניס. (אוניברסלי לגמרי).

- חישוב דירוג התמונה.
- כתיבת מספרים לתוך מסמך קיים.

מדדי הצלחה:

- התוצר הסופי - האתר יחזיר למשתמש משחק קו לקו מהתמונה שהכניס. כאשר נראה שהמשתמש קיבל תוצאת משחק משביעת רצון אדע שהאתר מוצלח.
- בנוסף לכל תמונה יש אופציית דירוג בין 1-5 ניתן יהיה לעקוב גם על הדירוגים כמדד.

רקע תיאורטי:

בין התחומים של מדעי המחשב המתפתחים בצעדי ענק בשנים אחרונות הינם התחום של עיבוד התמונה ותחום בעל קשר הדוק אליו של ראייה ממוחשבת. כאשר מדברים על עיבוד התמונה, בדרך כלל מתכוונים לפעולות חישוב אשר הקלט שלהן הוא תמונה ספרתית והפלט הוא תמונה ספרתית. להבדיל מעיבוד התמונה, תחום הראיה הממוחשבת עוסק בעיבוד הקלט שהוא גם תמונה (או אוסף תמונות) על מנת לזהות תכונות ופרמטרים שונים לצורך הסקת מסקנות וקבלת החלטות על בסיסם.

לאחר עיבוד התמונה מגיע שלב חילוץ המאפיינים (extraction feature) הינו מושג המתייחס לשיטת הורדת הממד של המידע המועבד. כאשר הקלט כולל כמות רבה של נתונים יתירים (redundant), רצוי מאוד להמיר אותו לאוסף מצומצם של תכונות או מאפיינים המספקים את כל המידע המעניין ללא יתירות.

קיימות הרבה שיטות לחילוץ המאפיינים השונים בתמונות ספרתיות. בעבודה זו אתמקד בשיטה לזיהוי פינות בתמונה.

מצב קיים:

הגישה הכללית בה משתמשים רוב האלגוריתמים לחילוץ מאפיינים ולזיהוי פינות בפרט היא בחלוקת העבודה לשלבים הבאים:

1. חישוב מפת העניין של התמונה, בה לכל נקודה משויך מספר המציין את מידת ההתאמה של הנקודה לנקודות העניין. במקרה של זיהוי פינות מדברים על מפת הפינתיות (map cornerness), המציינת את מידות ההסתברות שהנקודות הן פינות.

2. מציאת נקודות בעלות ערכי הפינתיות הגבוהים ביותר בסביבתן, או במילים אחרות - מקסימום מקומי (local maxima).

ניתוח חלופות מערכתי:

עיבוד תמונה מול ראייה ממוחשבת:

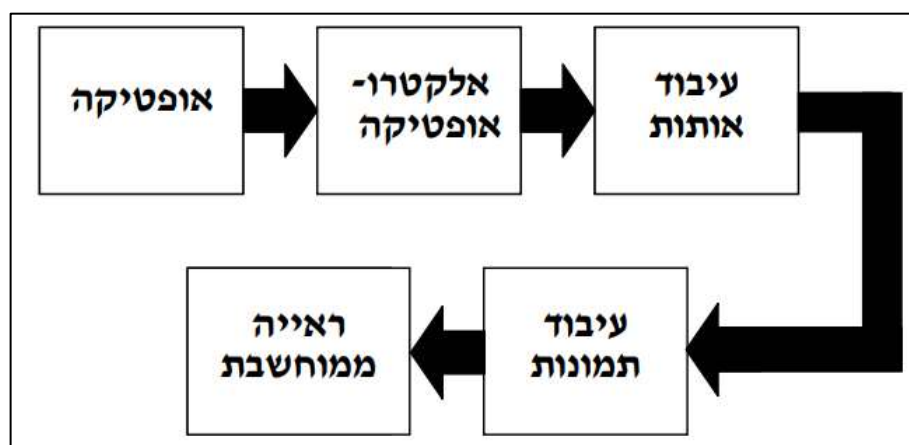
קיימים יישומים רבים לראיה ממוחשבת ולזיהוי צורות, וניכרת חדירה חזקה של תחום זה לתעשייה ולמוצרים בעשור האחרון.

"עיבוד תמונה" מתמקדת במשימות בהן הן הכניסה והן היציאה הם תמונות. "ראיה ממוחשבת" לעומת זאת דנה במשימות בהן הכניסה היא תמונה והמוצא אינו תמונה.

ניתן גם להציע חלוקה אחרת של פרקים הקשורים בעיבוד תמונות, המבוססת על תהליך עיבודה של תמונה מרגע יצירתה ועד סיום הטיפול בה במערכת דמוית ראיית אנוש. בשלב ראשון מצלמה (העין) אמורה לרכוש את התמונה, וזאת ע"י אופטיקה מתאימה (העדשה בעין), לאחר מכן התקנים אלקטרו-אופטיים (אותם גלאי CCD למשל במצלמה, או החיישנים בעין הקרויים Rods ו- Cones, ובסיום עיבוד אותות אלמנטרי של תיקון תחום דינאמי, דגימה וקוונטיזציה. ואמנם, לאופטיקה, אלקטרו-אופטיקה ועיבוד אותות קשר ישיר וחשוב לעיבוד תמונות (שלבים אלו מתבצעים ברטינה בעין).

לאחר בנייתה של התמונה, יבואו אלגוריתמי דחיסה ושיפור איכות השייכים ל- Stream Main בעיבוד תמונות (פעולות אלו מבוצעים בדרך מהעין למוח, ובמוח הראיה עצמו). אחריהם תתחיל פעולת ניתוח (ראייה ממוחשבת) לשם הפקת מידע מהתמונות, כגון זיהוי האנשים, אבחנה בתנועה וכו'.

כך אנו רואים מעבר מאופטיקה ועד ראייה ממוחשבת ולמעשה בינה מלאכותית. הציור הבא סוקר את המסלול הנ"ל על הפרקים אותם הוא רותם.



בפרויקט שלי עסקתי בחילוץ מאפיינים מתוך התמונה אך החזרת התמונה ולא הסקת מסקנות ממנה, כפי שנעשה בראייה ממוחשבת לכן **בחרתי בעיבוד תמונה** ולא בראיה ממוחשבת המבוססת על מודל. כמו כן רציתי לכתוב אלגוריתם שייתן לי סיפוק בו אני אכתוב את הקודים ואשתפשף כמה שיותר בכך. עצם הבחירה בכתיבת אלגוריתם ולא בלקיחת מודל בנוי שיפרה מאד את היכולת שלי להתמודד עם כתיבת קודים ארוכים ומסובכים.

מימוש ע"י עיבוד תמונה:

בעיבוד תמונה קיימים אלגוריתמים שונים למימוש השלבים לחילוץ מאפיינים ולזיהוי פינות. ביניהם:

מציאת מקסימום מקומי בשיטת K -Tomas, מציאת מקסימום מקומי בשיטה מקבילית, אלגוריתם NMS מקבילי ועוד רבים נוספים.

האלגוריתמים השונים נבדלים בעיקר בשיטת ביצוע השלב הראשון (חישוב מפת הפינתיות). השלב השני מתבצע לרוב בעזרת שיטת NMS (on maxima-non) (בה באופן הדרגתי הנקודות בעלות ערך לא מכסימלי בסביבתן מוסרות מהפתרון). (השלבים מה "מצב קיים" הנ"ל).

תיאור החלופה הנבחרת:

מהלך האלגוריתם הוא כמתואר לעיל במצב הקיים.

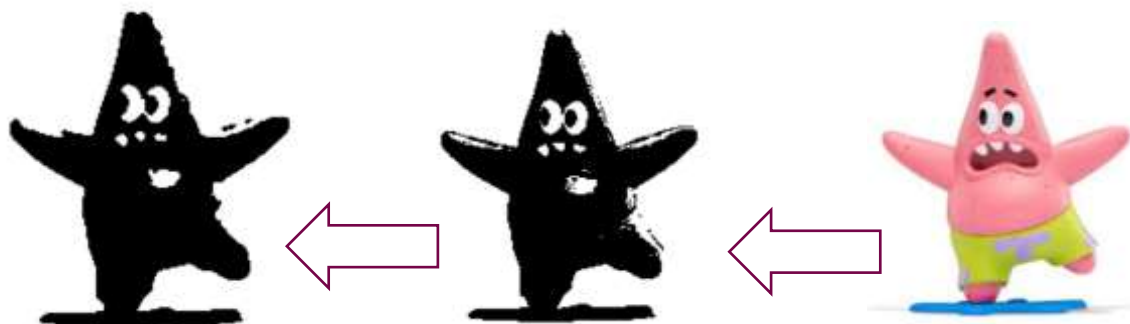
בפרויקט הנוכחי התמקדתי באלגוריתם לזיהוי פינות שהוצג על ידי האנס מורבץ (Moravec) (P Hans) מאוניברסיטת קרנגי-מלון בארה"ב. מורבץ מגדיר את מושג "נקודות העניין" כאזורים בתמונה הניתנים להבחנה, ומסיק כי בעזרתן ניתן למצוא אזורים מתאימים ברצפים של תמונות (למשל לצורך מעקב אחרי עצמים שונים). האלגוריתם נחשב לאלגוריתם המזהה פינות כי נקודות העניין אותן הוא מזהה הן הנקודות עם שונות גבוהה בבהירות בכיוונים שונים, שזה המצב בעיקר בנקודות שהן פינות.

האלגוריתם שלי פועל כך:

בתחילה **המרתי את התמונה לגוני אפור**

לאחר מכן **ניקיתי אותה מרעשי רקע** כך שזיהוי העצמים הרלוונטיים יהיה קל יותר ואז **המרתי את התמונה למטריצה של double** כדי שאוכל לבצע עליה את החישובים.

דוגמא לתמונה שהרצתי עליה הפיכה לשחור לבן ולאחר מכן ניקיתי מממנה רעשים:



השלב הבא הוא מציאת מספר הרכיבים הקשירים בתמונה ע"י אלגוריתם **למציאת מרכיבים קשירים** (connected component labeling)

תאור האלגוריתם:

בתחילה סרקתי את התמונה ולכל פיקסל נתתי תווית עפ"י השכנים שלו:

אם הפיקסל הוא 1 (לוגי) אזי:

א. אם רק לשכן השמאלי או לשכן שמעל יש תווית, העתק אותה.

ב. אם לשני השכנים הנ"ל אותה תווית, העתק אותה.

ג. אם לשניהם תוויות שונות, העתק את זו של השכן מעל וסמן את התוויות כשקולות בטבלת השקילות.

ד. אחרת, תן תווית חדשה לפיקסל והכנס אותה לטבלת השקילות

לאחר מכן עבור כל קבוצת שקילות מצאתי את התווית בעלת הערך הקטן ביותר.

סרקתי שוב והחלפתי כל תווית בתווית השקולה שמצאתי קודם.

אחרי שלבים התחלתיים אלו יכולתי להתחיל בביצוע הקטע המרכזי של האלגוריתם:

בתחילה מצאתי את נקודות הקצה בתמונה באמצעות **אלגוריתם מורבץ**:

אלגוריתם מורבץ מזהה את הנקודות עם השונות הגבוהה בבהירות בכיוונים שונים. (שזה המצב בעיקר בנקודות שהן פינות). עבור כל נקודה p בתמונה מגדירים חלון ריבועי קטן סביבה. ואז מזיזים את החלון בשמונה כיוונים שונים (אופקי אנכי ושני אלכסונים) בצעד של נקודה אחת ואז מסכמים את ריבועי ההפרשים בין הנקודות המתאימות בחלון המקורי והמוזז. השונות של הנקודה P היא השונות המינימלית בין המחושבות עבור כל אחד מהכיוונים. נתבונן באיור הבא. באיור מוצגת דוגמא להפעלת האלגוריתם על נקודה P עם חלון של 3×3 .

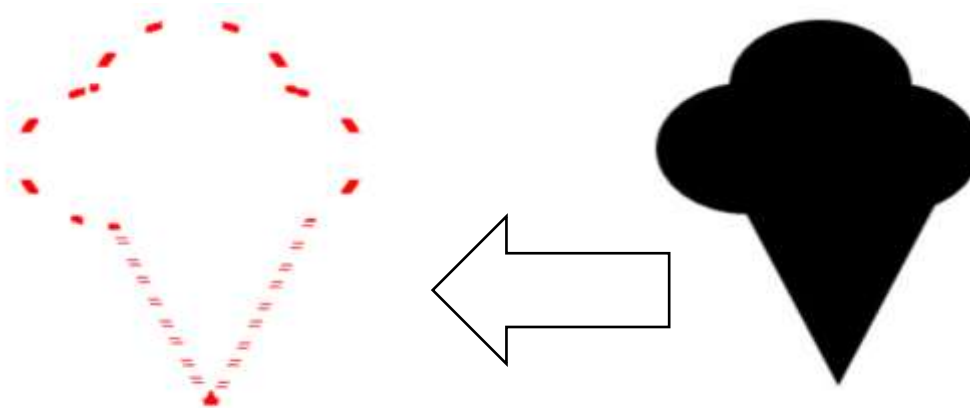
a_i מסומנות רמות האפור של הנקודות בחלון שסביב P ובי- b_i רמות האפור המתאימות בחלון המוזז בכוון אלכסוני.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	0	X	X	X	X
X	X	0	0	0	0	0	1	1	1	0	0	X	X	X	X
X	X	0	0	0	0	0	1	2	1	0	0	X	X	X	X
X	X	0	0	1	1	0	1	1	1	1	0	0	X	X	X
X	X	0	0	2	1	0	0	0	0	0	0	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	0	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

איור 5: מפת פינתיות

כפי שניתן לראות, במפת הפינתיות המתקבלת לאחר הפעלת אופרטור מורבץ, הפינות מסומנות בערך מכסימום מקומי (local maximum). (נקודות מבודדות גם הן מסומנות במכסימום מקומי, ולכן אם אנחנו לא מעוניינים בגילוי נקודות כאלה, נשתמש בחלונות גדולים יותר מ- 3×3 , כך שערך הפינתיות שלהן יהיה קטן יותר מזה של פינות אמיתיות, ונוכל לסנן אותן על ידי הצבת סף (threshold) על ערך הפינתיות.

דוגמא להרצת האלגוריתם על תמונה וצביעת נקודות הפינה שהאלגוריתם זיהה באדום:



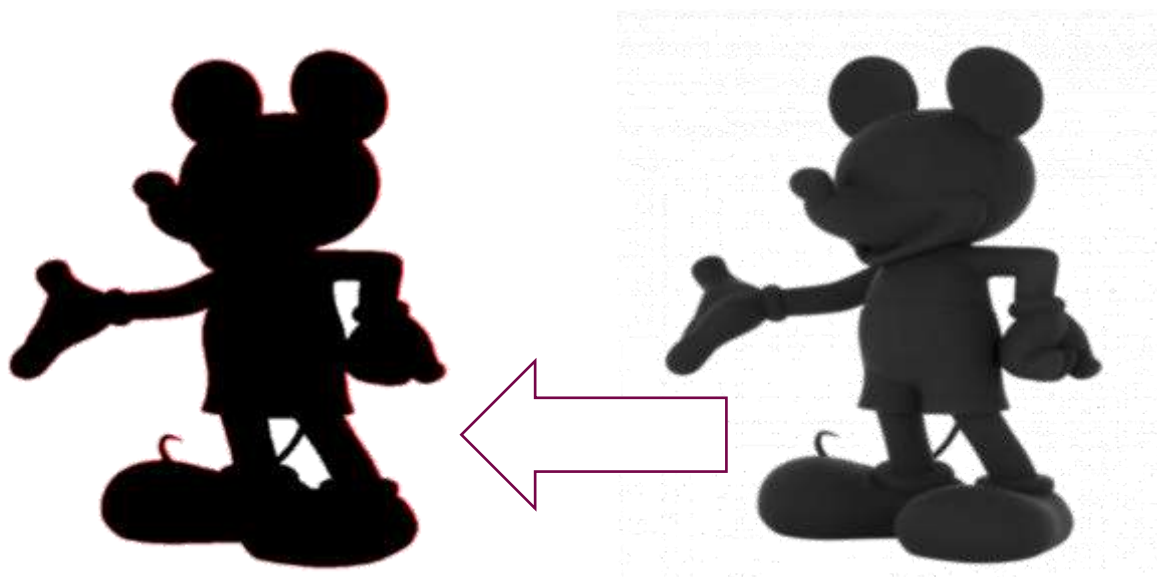
על מנת לזהות במין נקודות הענין שזוהו אילו נקודות הן פינות, הרצתי על התמונה את האלגוריתם למציאת פיקסלי הגבול/קונטור וההיקף.

האלגוריתם פועל כך:

בתחילה מצאתי את נקודת ההתחלה השחורה ע"י סריקה שיטתית של התמונה. לאחר מכן עבור כל נקודה עברתי על כל השכנים החל מהשכן המערבי עד למציאת פיקסל שחור.

כך נמצאו כל הפיקסלים המרכיבים את הגבול של התמונה.

דוגמא להפעלת האלגוריתם על תמונה וצביעת התוצאה באדום:



הרצתי את שתי האלגוריתמים האלו ולא הסתפקתי רק באחד המזהה פינות כי הוא לפעמים מוצא גם נקודות אחרות בתמונה כבעלות עניין אך הן לא במסגרת. כמו כן כאשר הרצתי רק את הראשון של פינות הוא קלט את הנקודות לפי הסריקה של השורות לרוחב ולכן אם המשתמש היה מעביר לפי זה היה יוצא לו זיגזגים בתוך התמונה ולא העברה מסודרת מסביב. ואילו לאחר ההרצה של זיהוי נקודות הגבול הנקודות יצאו ממוינות ונשאר רק לחבר ביניהם:

לאחר הרצת 2 האלגוריתם הללו נוצרו לנו 2 רשימות המכילות את הנקודות שנמצאו (1stBorder – עבור הרצת אלגוריתם מציאת גבול ו1stCorners – עבור אלגוריתם מורבץ לנקודות שהן פינות).

עברתי על כל רשימה ודיללתי אותה מאחר והיא מכילה אלפי פיקסלים שנמצאו. ולכן מחקתי בקירוב עבור כל נקודה את כל הנקודות שפונקציית חישוב מרחק (distance) החזירה עבורה מרחק הקטן מ45. –פונקציית דילול

לאחר מכן ביצעתי **חיתוך של שני הרשימות** שנמצאו ע"י שליחה לפונקציה אוניברסלית שיוצרת רשימה חדשה המכילה רק את הנקודות המשותפות לשתי הרשימות (שפונקציית חישוב המרחק החזירה עבורם מרחק הקטן מ150).

כרגע יש לי את הנקודות הסופיות שיהיו ממספרות בציר.

הכנתי רשימה נוספת המכילה את אותה כמות איברים כמו הרשימה הנ"ל ואני ממלאת אותה לפי מה שביקש המשתמש **למספר את הנקודות**:

אם ביקש מספור רגיל אני אשלח לפונקציית הממלאת מיספור עוקב רגיל (סדרה חשבונית המתחילה ב1 וההפרש יהיה 1 גם כן).

אם הוא ביקש אותיות עבריות נשלח אותו לפונקציית מילוי עם האות ההתחלתית 'א'. כנ"ל באנגלית עם האות 'A'.

אם הוא ביקש למספר בסדרה חשבונית או הנדסית קיבלתי מממנו את האיבר הראשון וההפרש ומילאתי לפיהם. כמו כן דאגתי שהמספור יהיה מותאם לגילאי המשחקים בו ולכן המספור לא יעלה על 100 ואם הגיע כבר ל100 יתחיל המספור שוב מהתחלה.

בסופו של כל התהליך הזה הגיע השלב הסופי של **מספור התמונה**. שלחתי לפונקציה את הרשימה הסופית המכילה את הנקודות למספור כנ"ל וכעת עברתי בלולאה בגודל אותה הרשימה ועבור כל פיקסל ציירתי את המספר שנמצא ברשימה שמילאתי קודם(לפי דרישת המשתמש) באותו המיקום. במרחק 10 פיקסלים מכל אחת מהנקודות ציירתי גם נקודה שהמשתמש יוכל להעביר את הקווים בצורה יפה ומהנה.

א פ י ו ן המערכת:

נ י ת ו ח ד ר י ש ו ת המערכת:

ד ר י ש ו ת ב ה מ המערכת צ ר י כ ה ל ע מ ו ד :

- ✓ כתיבה בסטנדרטים מקצועיים.
- ✓ מחשוב השרות ללקוח.
- ✓ כתיבת הקוד בסיבוכיות היעילה ביותר.
- ✓ ממשק נוח וידידותי למשתמש.
- ✓ תגובה מהירה ככל שניתן למשתמש

ס ב י ב ת פ י ת ו ח :

חומרה: 8GB I5

עמדת פיתוח: מחשב DELL

מערכת ההפעלה: Windows 10

שפות תוכנה: C#, תוך שימוש בטכנולוגית WebApi וריאקט.

כלי תוכנה לפיתוח המערכת: Studio Visual Microsoft 2019

מסד נתונים: Server SQL

עמדת משתמש מינימאלית:

- חומרה: 8GB I5
- מערכת ההפעלה: Windows 7 ומעלה
- חיבור לרשת: נדרש
- תוכנות: Internet Explorer 7 or higher or chrome

מודול מערכת:

- ☒ העלאת תמונה מהלקוח לשם יצירת משחק
- ☒ קריאת הקובץ שהועלה והזנת הפרטים בDB ע"י המערכת.
- ☒ קריאה למערכת לביצוע פונקציית הפיכה לגווני אפור
- ☒ הפיכה לשחור לבן
- ☒ ניקוי רעשי רקע
- ☒ מציאת קווי מתאר של התמונה
- ☒ מציאת פינות בתמונה
- ☒ מספור הפינות
- ☒ חלוקה לעצמים
- ☒ הזנת הפרטים בDB
- ☒ צפייה בתוצאה הסופית עם אפשרות להוספת פרטים לתמונה.

אפיון פונקציונאלי:

CreatePoints - פונקציה ראשית המזמנת את הפונקציות לפי הסדר.

ToGray - פונקציית ההופכת את התמונה לגווני אפור

ToBlackWhite - פונקציית ההופכת את התמונה לצבעי שחור לבן

PerformClean - פונקציה לניקוי רעשי רקע

ToDouble - פונקציית הממירה את התמונה לסוג double

find_corners - אלגוריתם המזהה פינות בתמונה. נעזר בשתי הפונקציות הבאות:

calculate_point - עובר כל נקודה עובר על המסגרת ומזמן את:

Calculate - פונקציה המקבלת קורדינציות של פינה שמאלית עליונה של שתי מטריצות ומחזירה את סכום ריבוע הפרשי הערכים בין נקודות תואמות במטריצה, הפונקציה מקבלת גם את גודל החלון.

FindBorder - מציאת נקודות הגבול של התמונה (קוי המתאר).

ReduceList - פונקציה המקבלת את רשימת הנקודות שנמצאו בתמונה ומחזירה את אותה הרשימה רק מצומצמת בכמות האיברים. מדללת לכל נקודה את אלו שהמרחק בינה לבניהן קטן מ45.

FindDistance - מציאת מרחק בין שתי נקודות. לפי חישוב מתמטי של distance

intersectLists-פונקציה המקבלת את רשימת הנקודות שחזרו מהאלגוריתם הראשון (מציאת פינות) ואת רשימת הנקודות שחזרו מהאלגוריתם השני (מציאת גבול) ומשלבת את הנקודות המשותפות לרשימה ממוינת חדשה (משותפות= שהמרחק בניהן קטן מ150).

פונקציות מילוי רשימת הנקודות למספור עפ"י בקשת השתמש:

- ✓ fillLetters - מילוי אותיות עברית או אנגלית.
- ✓ fillSeries – סדרה חשבונית
- ✓ fillEngineeringSeries – סדרה הנדסית (כפל או חילוק)

countingMap-פונקצייה מספור הנקודות (לפי הרשימה שהתמלאה באחת מפונקציות המיספור הנ"ל).

find_next_point-מציאת הנקודה הבאה למספור.

Region – פונקציה למציאת רק"חים בתמונה.

ביצועים עיקריים:

- מציאת רק"ח בתמונה
- עבור כל רק"ח מציאת נקודות ענין
- עבור כל רק"ח – מציאת נקודות הגבול
- צמצום רשימות הנקודות - על ידי איתור נקודות קרובות
- ביצוע חיתוך בין נקודות העניין לנקודות הגבול.
- סימון הנקודות על המפה
- שמירת התוצר כקובץ הניתן להורדה
- בשלבי פיתוח: מתן אפשרות להוספת קווים פנימיים מהתמונה המקורית

אילוצים:

המערכת מקבלת תמונות מסוג jpg בלבד על מנת שאלגוריתם ההמרה לשחור לבן יעבוד כראוי. (בנוסף ישנה הגבלה על כמות העצמים בתמונה).

תיאור הארכיטקטורה

הארכיטקטורה של הפתרון המוצע בפורמט של top-Down level Design:

בניית התוכנה נעשתה תוך שימוש בהפרדת רשויות מלאה ובניית היררכיה ברורה, על מנת למנוע שיבושים בנתונים ובצורת התצוגה. מה גם שרציתי לאפשר שינוי תכנותי בעתיד, הן בצורת הגישה לנתונים, והן בצורת התצוגה ללא הצורך במגע עם רשויות נוספות.

המערכת כוללת שני תחומים עיקריים:

- צד הלקוח המקבל תמונה שהמשתמש מעלה.
- פעולות מצד השרת ההופך את התמונה למשחק ע"י הפעולות הנדרשות תוך גישה למסד הנתונים.

תיאור המרכיבים בפתרון:

הארכיטקטורה של הפתרון בנויה במודל 3 השכבות:

שכבת מסד הנתונים – SQL.

שכבת הישויות – (Framework Entity) (והגישה למסד הנתונים – (Dal).

שכבת הקוד – (BLL).

שכבת הקישור – (WebApi).

ממשק משתמש – (UI).

שכבת הקוד נעזרת בספרייה שמכילה את מבנה הנתונים כדי לתקשר עם שכבת ה-Dal וה-WebApi.

בשיטה זו קיימת הפרדת רשויות מוחלטת וכל שכבה עומדת באופן עצמאי לחלוטין, ומתקשרת רק עם השכבה הסמוכה לה.

1. Base Data – מסד הנתונים, מורכב מטבלאות.

2. שכבת גישה לנתונים-DB

3. צד שרת c# בו כתוב האלגוריתם API

4. WebApi פרוטוקול התקשורת בין צד הלקוח וצד השרת.

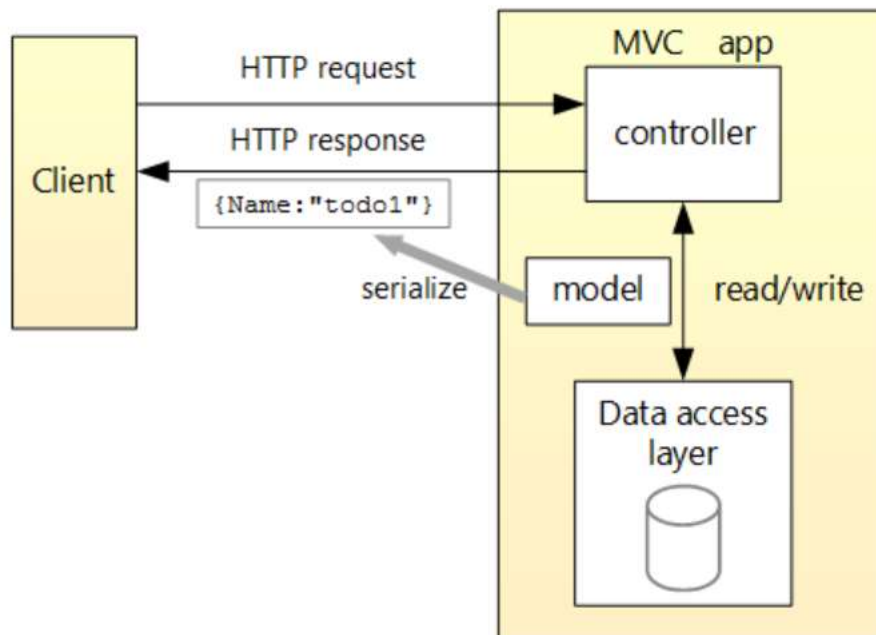
5. צד לקוח react

ארכיטקטורת רשת

המערכת מורכבת משרת IIS המריץ את האתר בסביבת ה-server,

מסד נתונים – database של sql-server , ממשק משתמש בצד הלקוח: דפדפן אינטרנט

כלשהו: internet explorer ,chrome,firefox



תיאור פרוטוקולי התקשורת:

http

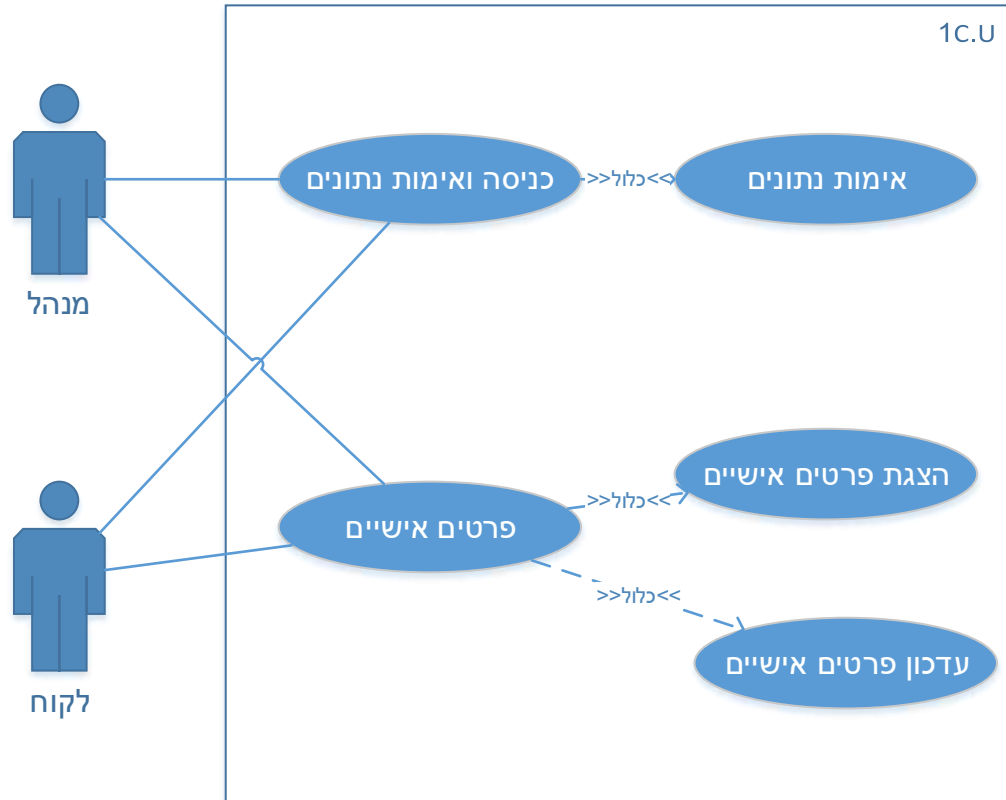
שרת - לקוח

צד השרת נכתב בטכנולוגיית WebApi ובשפת C#.

צד הלקוח נכתב בשפות react

ניתוח ותרשים UML / Use cases של המערכת המוצעת

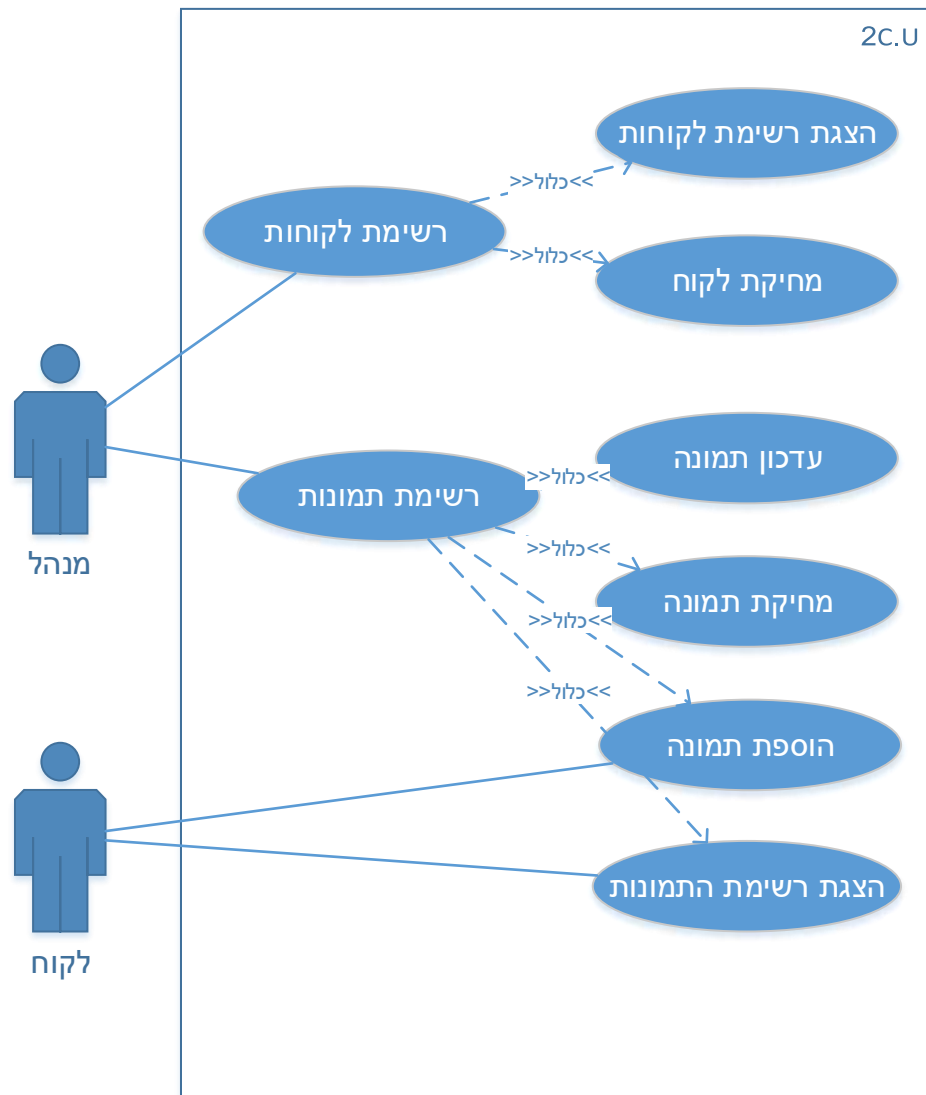
תיאור ה-UC העיקריים של המערכת:



U.C1	כניסה ואימות נתונים
הלקוח או המנהל נכנס למערכת אחרי הזנת פרטים אישיים	תיאור קצר:
הלקוח או המנהל רשום במערכת	תנאי קודם:
הפרטים שנכנסו שגויים או שהלקוח לא רשום במערכת	חריגים:
לקוח / מנהל	שחקנים מעורבים:
הלקוח או המנהל מעוניין להירשם במערכת כדי להיחשף לאופציות נוספות.	הזנק:
המערכת מאמתת את השם והסיסמא אם הם נכונים יוצג הדף הראשי עם שם המשתמש והודעה אם לא.	מהלך:
העמוד הראשון נפתח	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C1	הצגת פרטים אישיים
הלקוח או המנהל צופים בפרטיהם המופיעים במערכת.	תיאור קצר:
הלקוח או המנהל רשום במערכת	תנאי קודם:
הלקוח/המנהל לא רשום במערכת	חריגים:
לקוח / מנהל	שחקנים מעורבים:
הלקוח או המנהל מעוניינים לצפות בפרטיהם המופיעים במערכת.	הזנק:
המערכת שולפת את פרטי הלקוח או המנהל ומציגה אותם.	מהלך:
פרטי הלקוח או המנהל מוצגים במסך	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C1	עדכון פרטים אישיים
הלקוח או המנהל מעדכנים את פרטיהם המופיעים במערכת.	תיאור קצר:
הלקוח או המנהל רשום במערכת	תנאי קודם:
הלקוח/המנהל לא רשום במערכת או הוכנסו ערכים שגויים	חריגים:
לקוח / מנהל	שחקנים מעורבים:
הלקוח או המנהל מעוניינים לעדכן את פרטיהם המופיעים במערכת.	הזנק:
המערכת שולפת את פרטי הלקוח או המנהל ומציגה אותם באופן הניתן לעריכה.	מהלך:
לאחר לחיצה על "עדכן" השינויים יישמרו במערכת.	
מוצג דף הכניסה לאתר	תנאי סיום:
מספר פעמים ביום	תדירות:



U.C2	הצגת רשימת לקוחות
המנהל רואה את פרטי הלקוחות	תיאור קצר:
הלקוחות רשומים במערכת	תנאי קודם:
מנהל	שחקנים מעורבים:
המנהל מעוניין לצפות בפרטי הלקוחות	הזנק:
המערכת מציגה למנהל את רשימת הלקוחות עם אופציה למחיקת לקוח ספציפי.	מהלך:
ידיעת פרטי הלקוחות ויכולת מחיקה	תנאי סיום:
מספר פעמים ביום	תדירות:

שירה חדאד-מנקודה לנקודה

U.C2	מחיקת לקוח
המנהל מסיר לקוח מרשימת הלקוחות	תיאור קצר:
הלקוח או המנהל רשום במערכת	תנאי קודם:
מנהל	שחקנים מעורבים:
הלקוח קיים במערכת ומעוניינים להסירו	הזנק:
המנהל רואה את רשימת הלקוחות, בוחר את הלקוח שברצונו להסיר ולוחץ על מחיקה. לפני מחיקה סופית תוצג מודעת אזהרה ליוזם סופי.	מהלך:
הלקוח נמחק	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C2	הצגת רשימת תמונות
מאפשר ללקוח או למנהל לצפות ברשימת התמונות הקיימות במאגר	תיאור קצר:
יש תמונות במאגר	תנאי קודם:
לקוח / מנהל	שחקנים מעורבים:
הלקוח או המנהל מעוניין לצפות ברשימת התמונות	הזנק:
המערכת עוברת על רשימת התמונות ומציגה את פרטיהן.	מהלך:
המערכת מציגה את רשימת התמונות (משחקים).	תנאי סיום:
מספר פעמים ביום	תדירות:

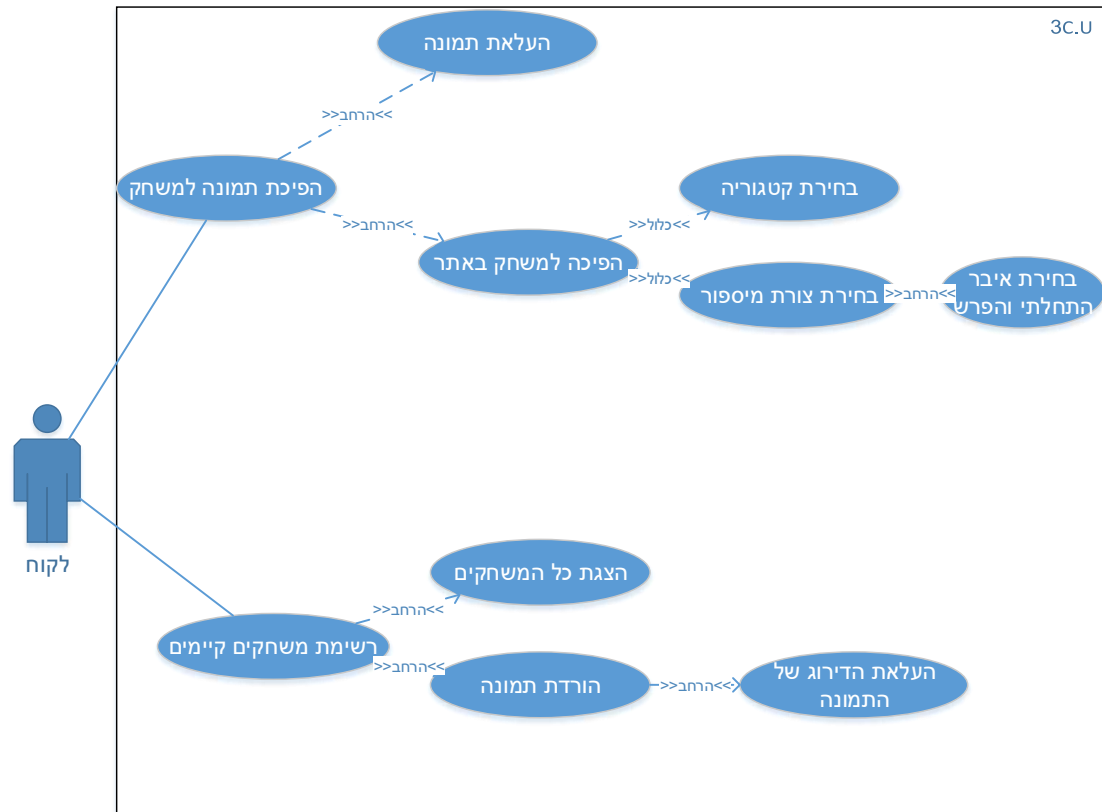
U.C2	הוספת תמונה
הוספת תמונת משחק לרשימת התמונות.	תיאור קצר:
המשחק לא קיים במערכת	תנאי קודם:
לקוח / מנהל	שחקנים מעורבים:
הלקוח או המנהל מעוניין להעלות תמונת חדשה למערכת (שתהפוך למשחק).	הזנק:
לחיצה על משחק חדש, העלאת תמונה, לחיצה על הפוך למשחק.	מהלך:
הוזנה תמונה שאינה עונה על הקריטריונים הדרושים. תוצג אזהרה במקרה הניתן לשינוי והסבר במקרה שאין.	חריגים:
נוספה תמונה חדשה לרשימת התמונות.	תנאי סיום:
מספר פעמים ביום	תדירות:

שירה חדאד-מנקודה לנקודה

U.C2	עדכון תמונה
המנהל נכנס למערכת אחרי הזנת פרטים אישיים	תיאור קצר:
המנהל רשום במערכת	תנאי קודם:
מנהל	שחקנים מעורבים:
המנהל מעוניין להירשם במערכת כדי להיחשף לאופציות נוספות.	הזנק:
המנהל עובר על פרטי המשחק ומעדכן את הפרטים הרצויים.	מהלך:
הוזנו נתונים לא תקינים, תוצג תיבת אזהרה ותינתן אפשרות להזנה מחדש.	חריגים:
העמוד הראשון נפתח	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C2	מחיקת תמונה
מחיקת תמונה מתוך מאגר המשחקים הקיימים	תיאור קצר:
הצגת רשימת התמונות	תנאי קודם:
מנהל	שחקנים מעורבים:
מנהל האתר מעוניין למחוק את התמונה הזו.	הזנק:
המערכת מציגה את רשימת המשחקים, המנהל מכניס את הקוד של התמונה שברצונו להסיר, ולוחץ על אישור.	מהלך:
התמונה נמחקה ממאגר המשחקים	תנאי סיום:
מספר פעמים ביום	תדירות:

שירה חדאד-מנקודה לנקודה



U.C3	העלאת תמונה
מאפשר ללקוח להעלות תמונה למאגר	תיאור קצר:
התמונה עומדת בסיומת המתאימה	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין להעלות תמונה	הזנק:
הלקוח בוחר קובץ ממכשירו ולוחץ על "בחר"	מהלך:
התמונה הועלתה בהצלחה	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C3	הפיכת תמונה למשחק באתר
המשתמש הופך את התמונה שבחר למשחק קו לקו חדש	תיאור קצר:
המשתמש העלה תמונה בהצלחה	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין לקבל משחק מהתמונה שהעלה	הזנק:
המערכת שולפת את התמונה שהעלה המשתמש ומפעילה עליה את האלגוריתם.	מהלך:
המשתמש רואה את המשחק הסופי	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C3	בחירת קטגוריה
המשתמש בוחר את הקטגוריה המתאימה לתמונה שבחר	תיאור קצר:
המשתמש בחר תמונה למשחק	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין לקבל משחק מהתמונה שהעלה	הזנק:
המערכת מגדירה את הקטגוריה המתאימה בתמונה החדשה.	מהלך:
הקטגוריה נבחרה בהצלחה	תנאי סיום:
כמספר הפעמים של יצירת משחק חדש.	תדירות:

U.C3	בחירת צורת מספור
המשתמש בוחר את המספור המסוים שהוא מעוניין למספר את התמונה שבחר	תיאור קצר:
המשתמש העלה תמונה בהצלחה	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין לקבל משחק מהתמונה שהעלה	הזנק:
המערכת מגדירה את המספור המתאים בתמונה החדשה.	מהלך:
המספור נבחר בהצלחה	תנאי סיום:
כמספר הפעמים של יצירת משחק חדש.	תדירות:

U.C3	בחירת איבר התחלתי והפרש
המשתמש בוחר את המספר ההתחלתי למספור ואת הפרש הקפיצות	תיאור קצר:
המשתמש בחר שצורת המספור תהיה סדרה (הנדסית או חשבונית)	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין למספר את התמונה בסדרה.	הזנק:
המספרים נקלטו בהצלחה	תנאי סיום:
כמספר הפעמים של בחירת מספור - <סדרה הנדסית \ חשבונית	תדירות:

U.C3	הצגת כל המשחקים
המשתמש רואה את כל המשחקים הקיימים במאגר	תיאור קצר:
במערכת קיימים משחקים	תנאי קודם:
לקוח	שחקנים מעורבים:
הלקוח מעוניין לצפות במאגר התמונות	הזנק:
המערכת שולפת את כל המשחקים הקיימים ומציגה אותם	מהלך:
מתווספת תמונה חדשה למאגר	תנאי סיום:
מספר פעמים ביום	תדירות:

U.C3	הורדת תמונה+ העלאת הדירוג
הלקוח מעוניין להוריד את המשחק אל המחשב האישי שלו	תיאור קצר:
התמונה קיימת במערכת	תנאי קודם:
לקוח	שחקנים מעורבים:
הורדת התמונה.	הזנק:
לחיצה על כפתור הורדה תוריד את התמונה למחשב האישי של הלקוח ותעלה אוטומטית את הדירוג שלה.	מהלך:
התמונה הורדה בהצלחה אל מחשב הלקוח	תנאי סיום:
מספר פעמים ביום	תדירות:

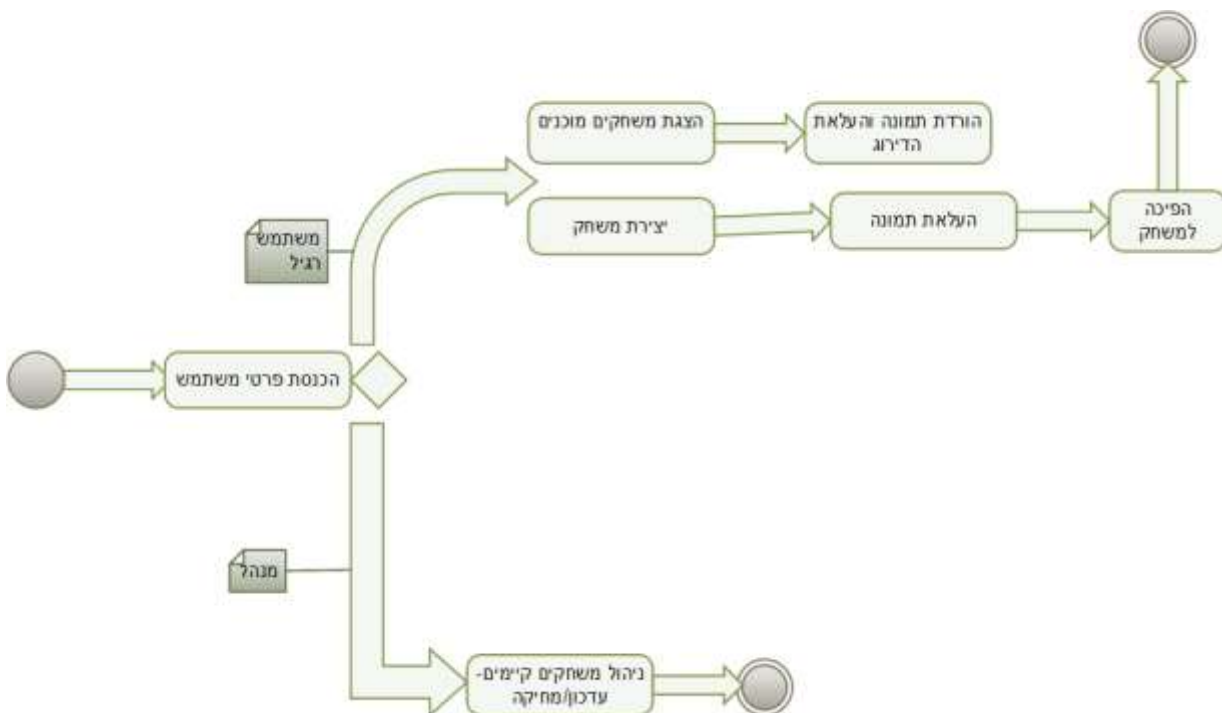
מבני הנתונים בפרויקט:

מטריצה - כל תמונה שנסרקה מיוצגת בהתחלה ע"י מטריצת ביטים. בחירה במבנה זה נעשתה באופן טבעי כי תואמת לפורמט התמונה המקורי וכן למבנה הדרוש לאלגוריתם.

רשימה מקושרת (List) – נקודות העניין שיאותרו במהלך ריצת האלגוריתם וכן נקודות הכבול שיאותרו על ידי האלגוריתם למציאת פיקסלי גבול יישמרו ברשימה מקושרת כל איבר ברשימה הוא מטיפוס point המיצג נקודה.

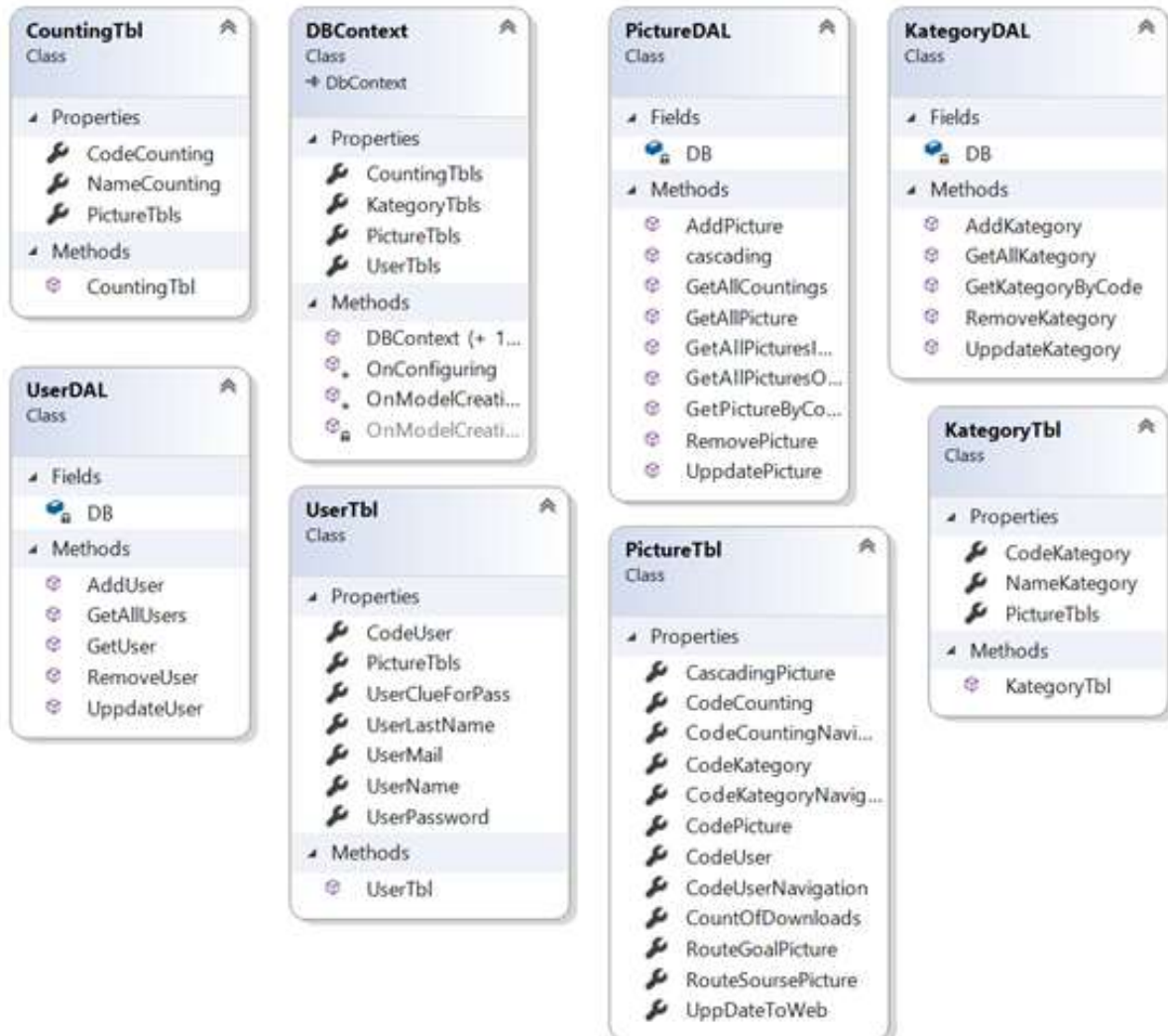
בחרתי לשמור את הנתונים במבנה נתונים זה הממומש בסביבת העבודה כמערך דינמי משום שהוא משלב את הנוחות של גישה אקראית וריצה באמצעות אינדקס, עם אפשרות להגדלה דינמית של מספר האיברים בסיבוכיות ממוצעת של $O(1)$ – המערך מכפיל את גודלו פי שניים בכל מצב של הוספה לאוסף מלא.

תרשים תהליכים:

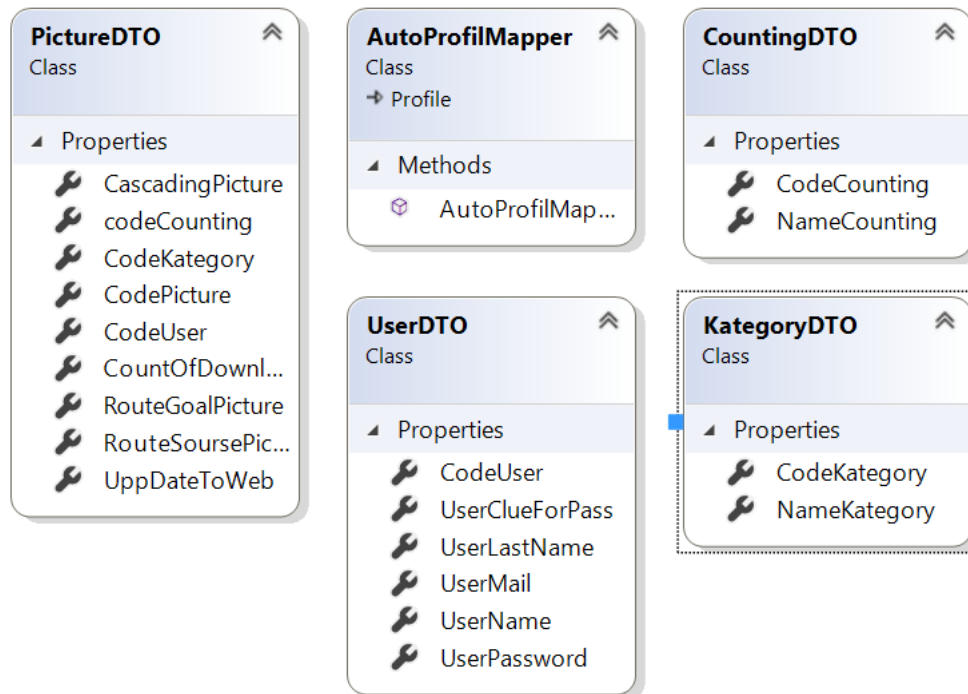


תרשים המחלקות:

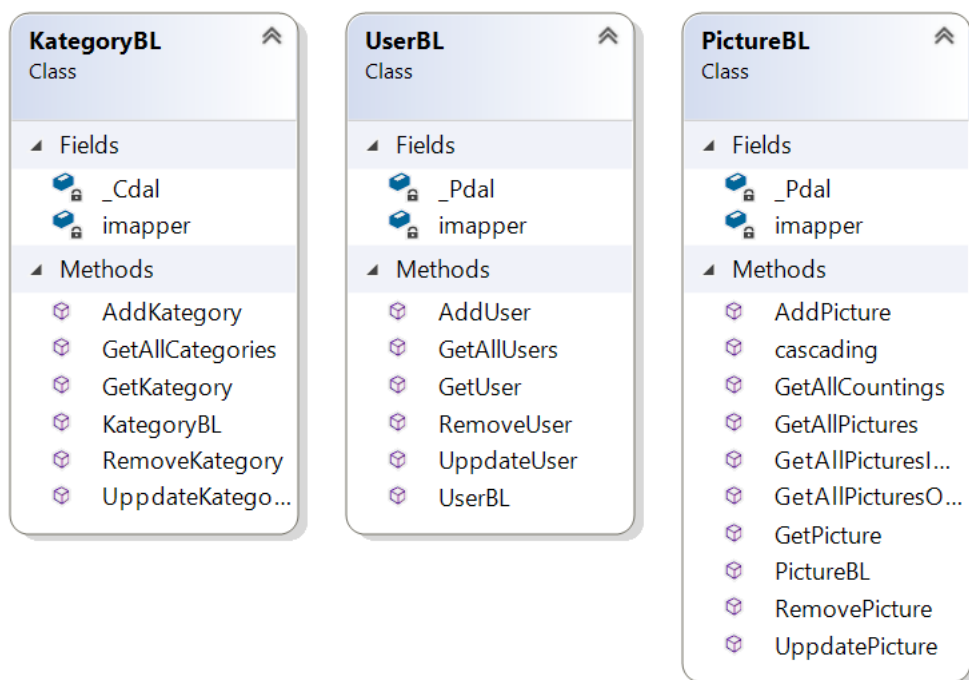
מחלקות הDAL:



מחלקות DTO:



מחלקות הבל:



תיאור המחלקות המוצעות:

- ❖ **שכבת – Data Base** - שכבה זו היא בסיס נתונים שישמור את הנתונים עבור הפרויקט, בנוי כקובץ של SQL DATABASE ומכיל את כל הטבלאות הנחוצות עבור הנתונים שימשו עבור עיבוד התמונות. ב. DB נשמרים הנתונים הבאים לפי טבלה:
- טבלת קטגוריות: בטבלה זו נשמרים סוגי הקטגוריות הקיימים במערכת.
 - טבלת משתמשים: עבור כל משתמש נשמר המידע הרלוונטי כמו: קוד, שם, משפחה, מייל, סיסמא, רמז לסיסמא.
 - טבלת תמונות(משחקים): עבור כל משחק נשמר מידע רלוונטי עבורו כמו: קוד תמונה, קוד משתמש שהעלה את התמונה, קוד קטגוריה, תאריך העלאה, דירוג התמונה, ניתוב תמונת מקור, ניתוב תמונת משחק, מספר הורדות של התמונה.
 - טבלת צורת מיספור: בטבלה זו נשמרים סוגי המיספורים הקיימים במערכת. (רגיל, אותיות בעברית, אותיות באנגלית, סדרות וכו..).

❖ שכבת הגישה לנתונים:

שכבה זו היא השכבה האחראית על הגישה לנתונים השמורים במסד הנתונים ולהעבירם לשכבה המבצעת את האלגוריתם. מגיעות אליהן בקשות ממחלקת ה BL שהן מקבלות בקשות מהלקוח. מחלקות אלו בנויות כל אחת- ממחלקה המממשת ממשק, בממשק כתובות כל פעולות המחלקה. ברגע שמגיעה בקשה ממחלקת ה BL אנו פונים אל המחלקה המממשת את הממשק ומבצעים את פעולת השליפה ממסד הנתונים. יוחזר לנו אובייקט מסוג שהלקוח אינו מכיר לכן נחזיר את האובייקט למחלקת BL והוא כבר יבצע את ההמרה לאובייקט שהלקוח מכיר.

ע"י הפקודה DB-Scaffold נוצרות ישויות -מחלקות עבור כל אחת מהטבלאות שדרכם, עם טכנולוגיית ה Entity-Framework – ניתן לגשת לנתוני מסד הנתונים.

המחלקות:

KategoryDAL,PictureDAL,UserDAL,countingDAL

❖ שכבת ה BL:

מחלקות אלו מקשרות בין הבקשות הזורמות מהלקוח, לבין השליפות ממסד הנתונים- הנמצא במחלקות נ. DAL-מחלקות אלו בנויות כל אחת- ממחלקה המממשת ממשק, בממשק כתובות כל פעולות המחלקה. ברגע שמגיעה בקשה מהקונטרולר, הבקשה קוראת לפעולה מהממשק והיא פונה אל המחלקה לצורך ביצוע הפעולה. הפעולות הממומשות במחלקה זו הן פעולות שקוראות לשכבה התחתונה יותר וממירות את הערך המוחזר מהן לאובייקט שיכול לחזור אל הלקוח.

בשכבה זו מתבצעים האלגוריתמים וקטעי הקוד הקריטיים והמשמעותיים בפרויקט.

שכבה זו מחולקת גם היא למחלקות.

המחלקות:

KategoryBL,PictureBL,UserBL,CountingBL

❖ מחלקות Data Object :

מחלקות אלו מכילות את שדות מסד הנתונים. האובייקטים היוצאים ממחלקות אלו , יכולים לחזור אל הלקוח. ברגע שמתקבלת קריאה מהלקוח נחזיר אובייקט מסוג זה

❖ מחלקות הקוד :

`Imageprosses`: מחלקה זו היא זו שמבצעת את הפעולות העיקריות בפרויקט
`DisjointSet`-מחלקה המשמשת למימוש הפונקציות של קבוצות זרות ורקח"ים.
`UnsafeBitmap` השתמשתי בפונקציות ממחלקה זו כאשר ניקיתי רעשי רקע.

רכיבי ממשק:

דפדפן chrome ,edge ,firefox וכו'.

תיכון המערכת:

שרת לקוח

ארכיטקטורת המערכת:

WebAPI

תיכון מפורט:

צד שרת-C#

צד לקוח- react

חלופות לתיכון המערכת:

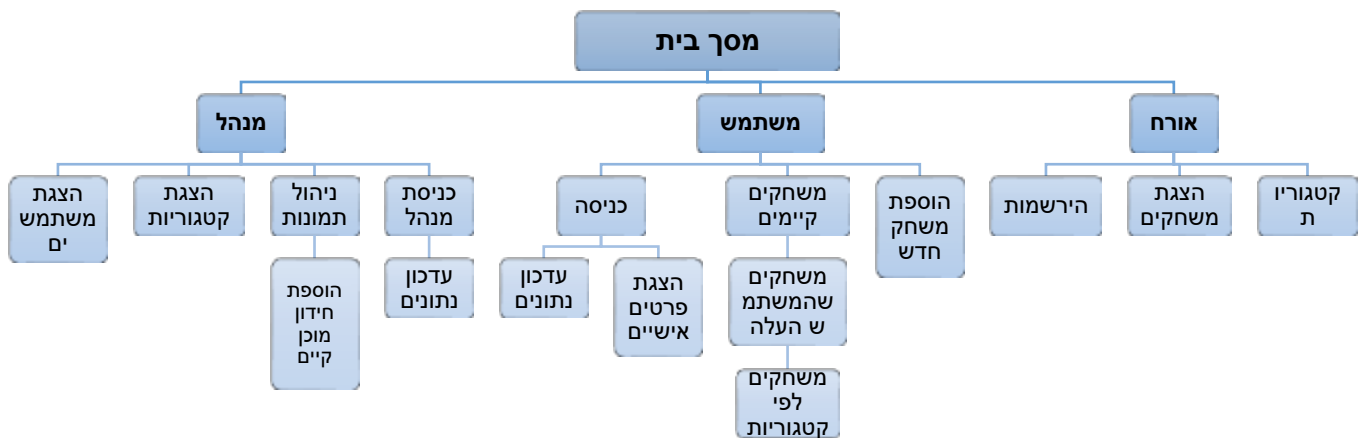
Java, Python

תיאור התוכנה:

סביבת עבודה: visual studio 2019 ,visual studio code

שפות תכנות: שימוש ב- HTML וב- java script . כמו כן, שימוש בטכנולוגית WebAPI .

תרשים המסכים:



תיאור המסכים:

מסכים לאורח:

- ❖ מסך הירשמות.
- ❖ מסך כניסה שונה.
- ❖ משחקים קיימים
- ❖ משחקים לפי קטגוריות

מסכים למשתמש הרגיל:

- ❖ דף הבית
- ❖ כניסה.
- ❖ עדכון נתונים.
- ❖ משחקים קיימים
- ❖ הוספת משחק חדש
- ❖ קטגוריות משחקים
- ❖ משחקים שאני(המשתמש) העלתי

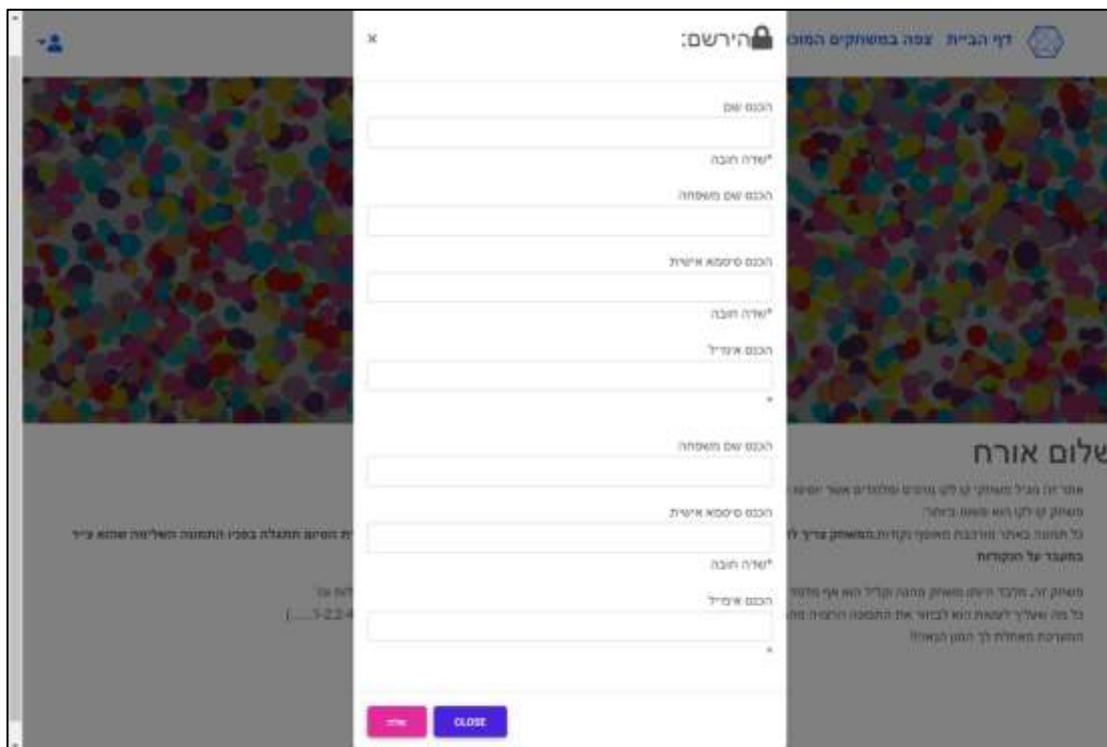
מסכים למנהל:

- ❖ כניסת מנהל
- ❖ ניהול המשחקים
- ❖ רשימת משתמשים
- ❖ הוספה והסרת קטגוריות

מסכים לאורח

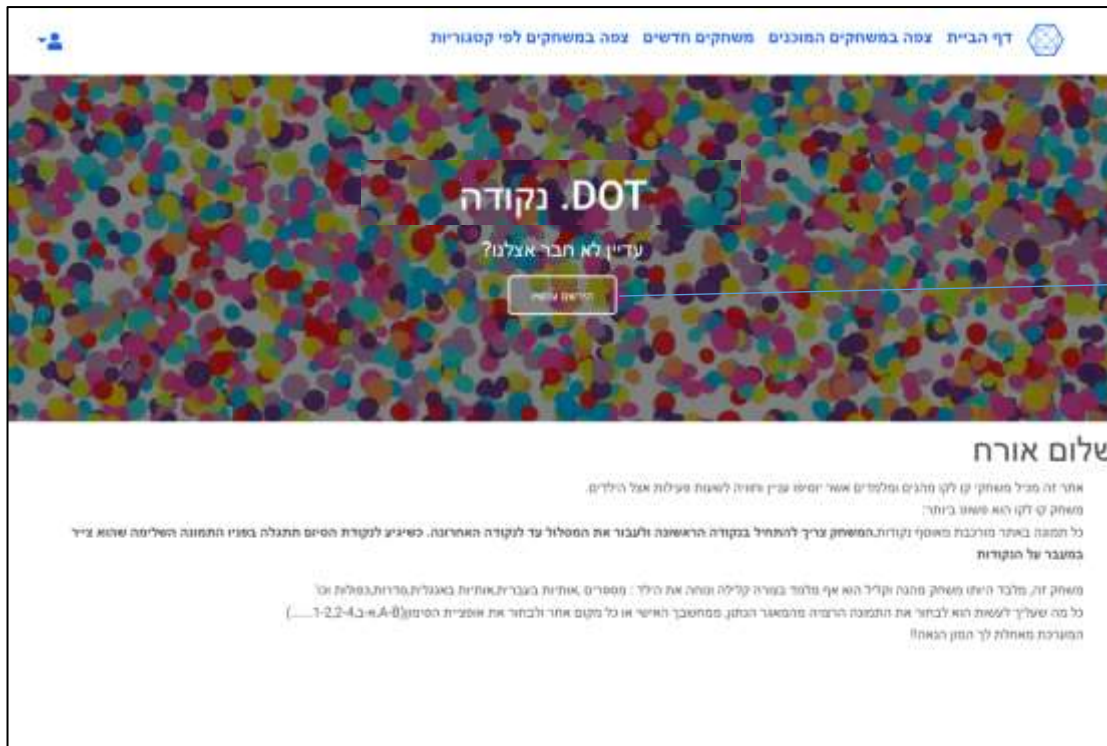
- ❖ מסך הירשמות.

האורח יוכל להירשם למשתמש באתר.



- ❖ מסך כניסה שונה: במקום שמו יופיע כפתור הירשמות

שירה חדאד-מנקודה לנקודה



❖ משחקים קיימים

כמו אצל משתמש רגיל בהמשך.

❖ משחקים לפי קטגוריות

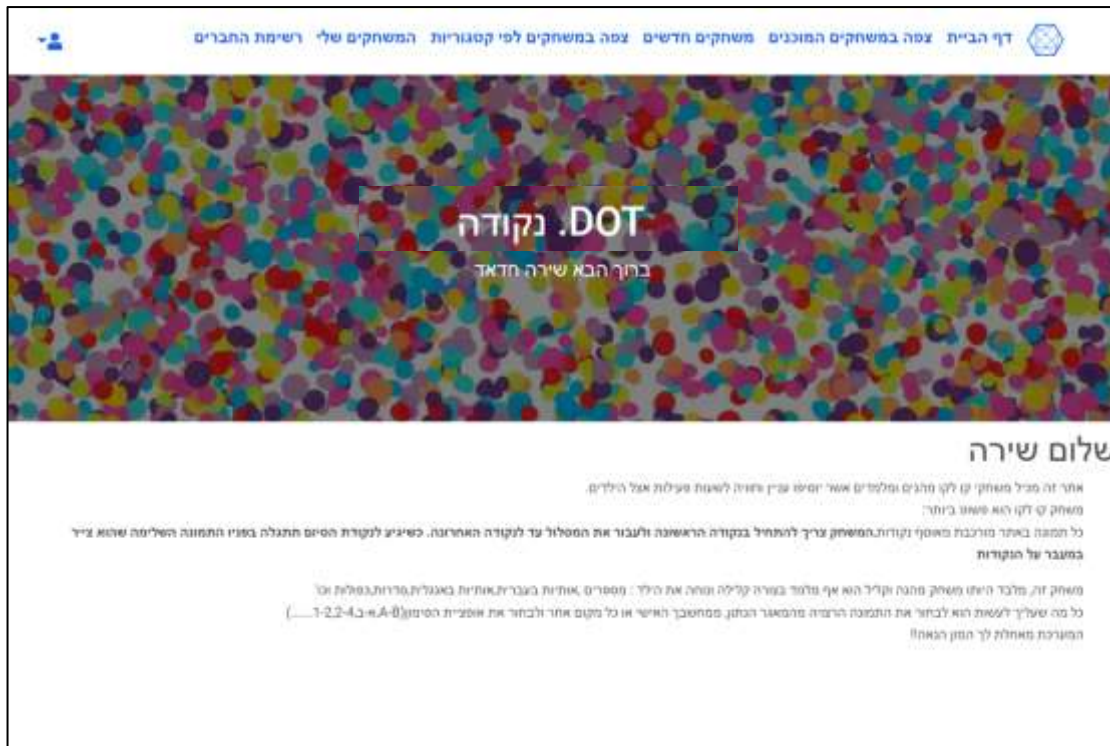
כמו אצל משתמש רגיל בהמשך.

מסכים למשתמש הרגיל:

❖ דף הבית

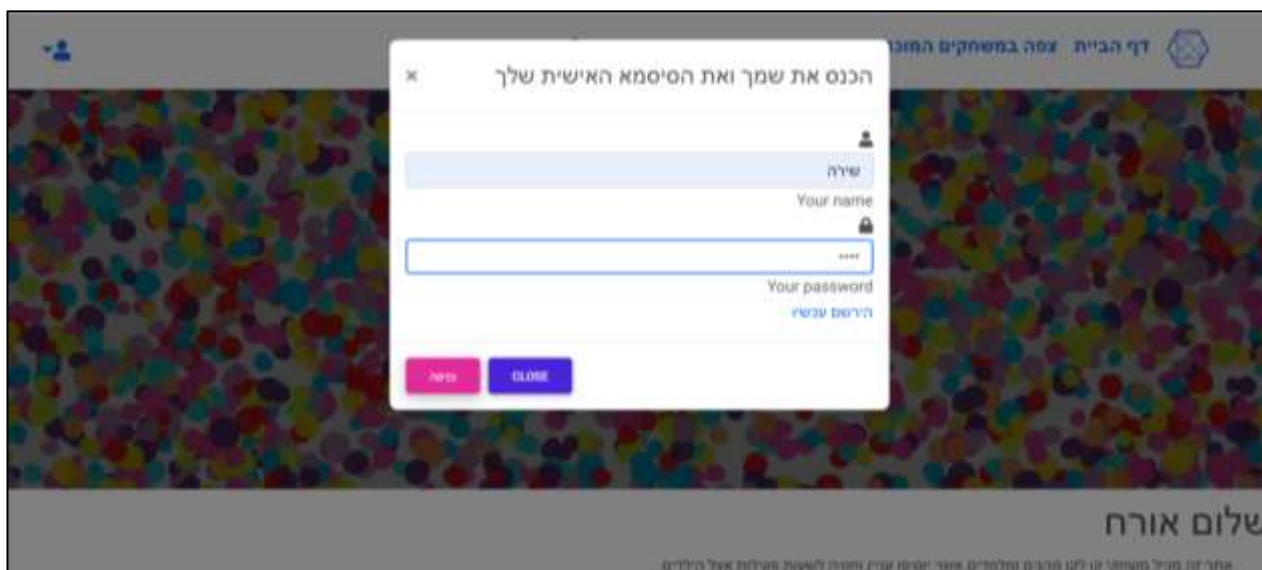
כאשר משתמש נכנס לאתר מוצג לפניו שמו באופן ידידותי, נעים ומזמין. כמו"כ מוצג תפריט כך הוא יכל לעבור בין החלוניות השונות:

שירה חדאד-מנקודה לנקודה



❖ כניסה

המשתמש מזין שם וסיסמא על מנת להיכנס.



❖ עדכון נתונים:

לאחר שהמשתמש נכנס למערכת הוא יוכל לעדכן את הפרטים שלו.

שירה חדאד-מנקודה לנקודה

טופס עדכון:

הכנס שם:

שירה:

ישדה חובה:

הכנס שם משפחה:

חדאד:

הכנס סיסמא אישית:

ישדה חובה:

הכנס אימייל:

תחום: @gmail.com

הכנס שם משפחה:

חדאד:

הכנס סיסמא אישית:

ישדה חובה:

הכנס אימייל:

תחום: @gmail.com

שלום שירה

אנחנו זוכים לנכדך המיוחד, נגדך נוספים פילמונים אשר יסייעו
 במחקר של ד"ר שירה חדאד ובחזרה
 כל תמונה באיור וזיהוי מאפשרת נקודות במחקר שלך על
 בחינה של הנקודות

מחקר זה מלווה היסטוריה משותפת חזקה וקדירה היא אף חדשה
 כל מה שעליך לעשות הוא לבדוק את התמונה הרגילה מה
 המדעית מאחוריך לך המון תשובות!

❖ משחקים קיימים:

המשתמש יוכל לצפות במשחקים הקיימים ועבור כל אחד ללחוץ על פרטים נוספים. בפרטים הנוספים נוכל לראות את המשתמש שהעלה את התמונה, את הדירוג שלה וכן להוריד אותה.

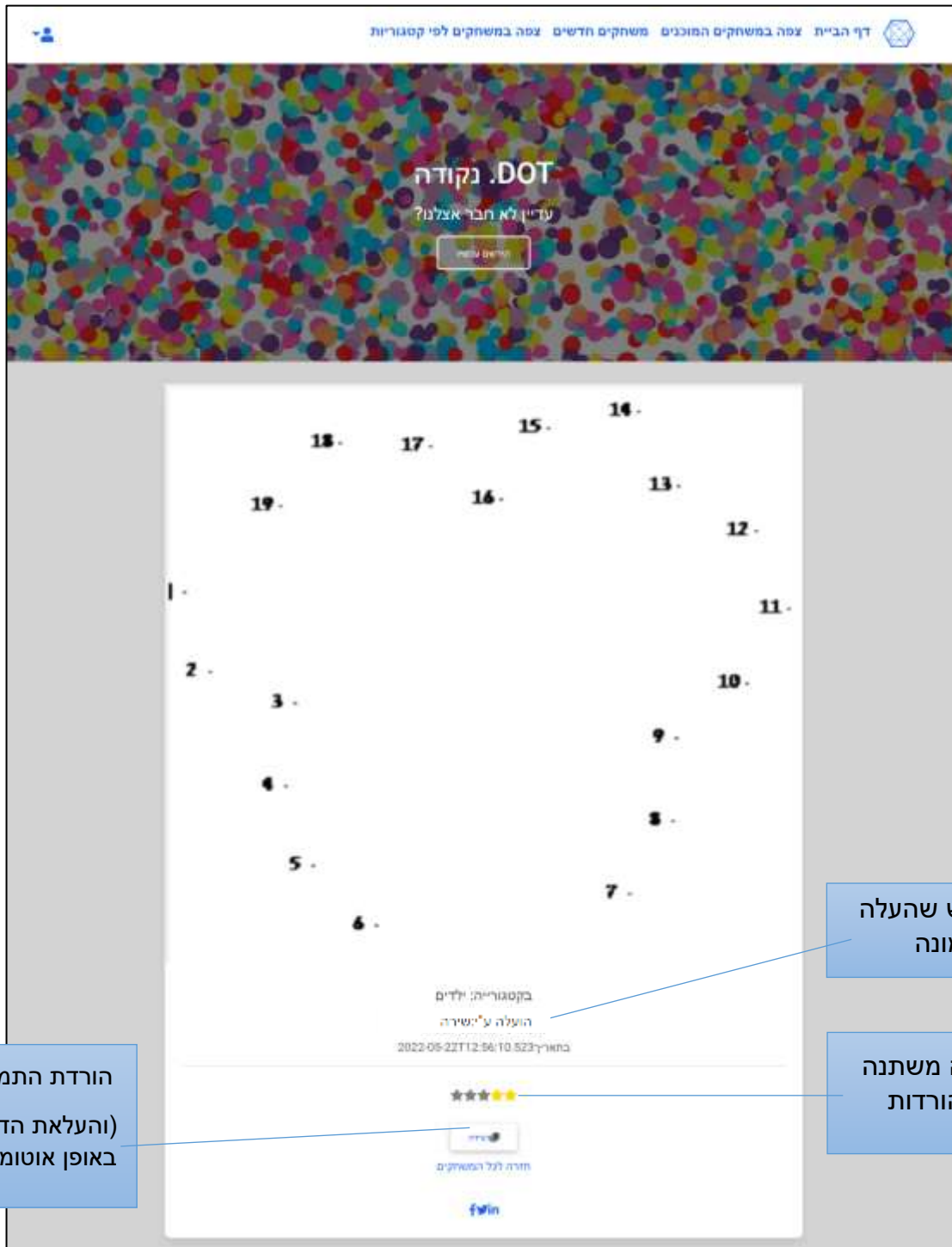
דף הבית צפה במשחקים המוכנים צפה במשחקים חדשים צפה במשחקים לפי קטגוריות המשחקים שלי

DOT. נקודה
 בחר הבא שירה חדאד

דירוג התמונה

פרטי תמונה נוספים

שירה חדאד-מנקודה לנקודה



שם המשתמש שהעלה את התמונה

דירוג התמונה משתנה לפי כמות הורדות

הורדת התמונה (והעלאת הדירוג באופן אוטומטי).

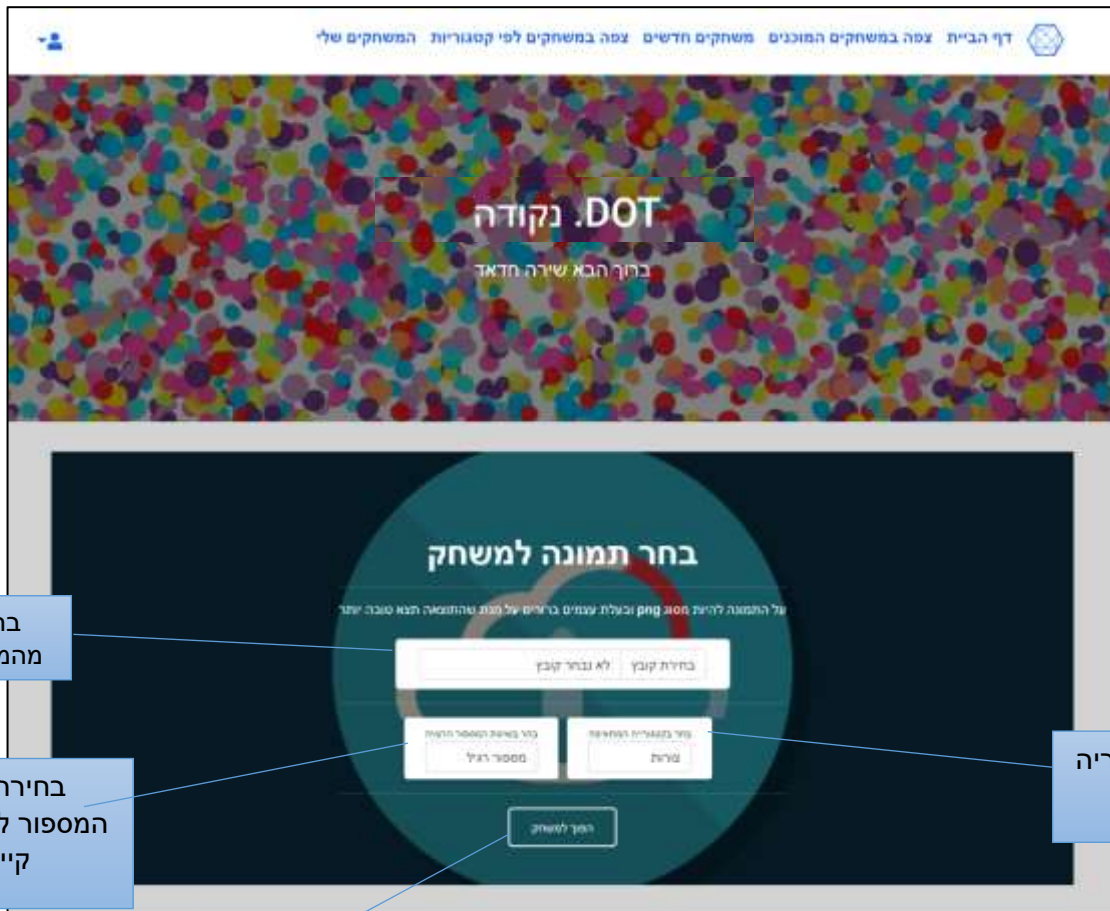
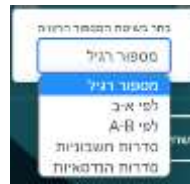
שים לב!

לא תוכל להעלות את הדירוג של התמונה שלך! הורדה...

משתמש שהעלה את התמונה לא יכל להעלות את הדירוג ותופיע ההערה:

❖ הוספת משחק חדש:

המשתמש יוכל לבחור תמונה ולהעלות למשחק. כאשר יהיה עליו לבחור את הקטגוריה המתאימה ואת סוג המספור שירצה:



בחירת קובץ
מהמחשב האישי

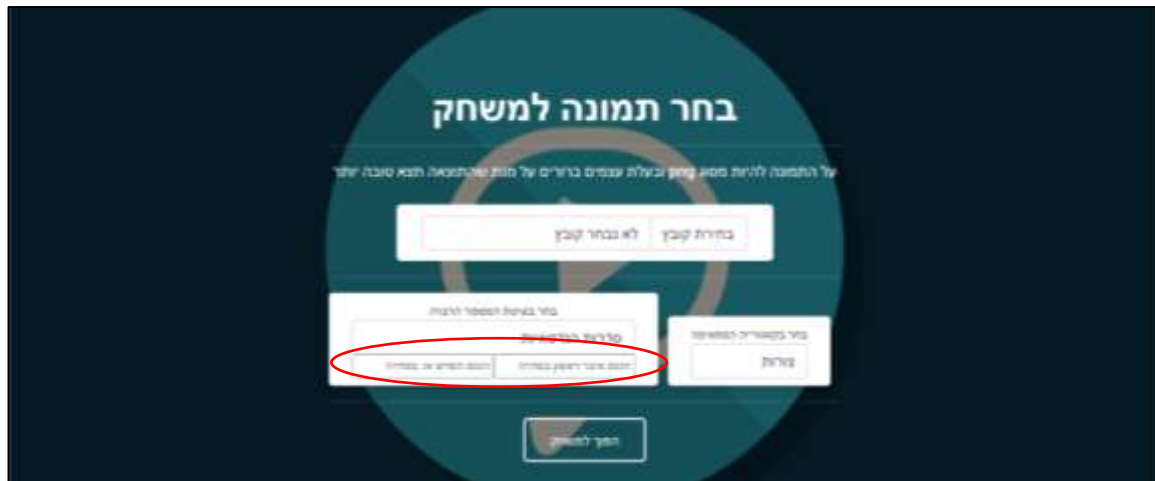
בחירת שיטת
המספור לפי אופציות
קיימות

בחירת הקטגוריה
המתאימה

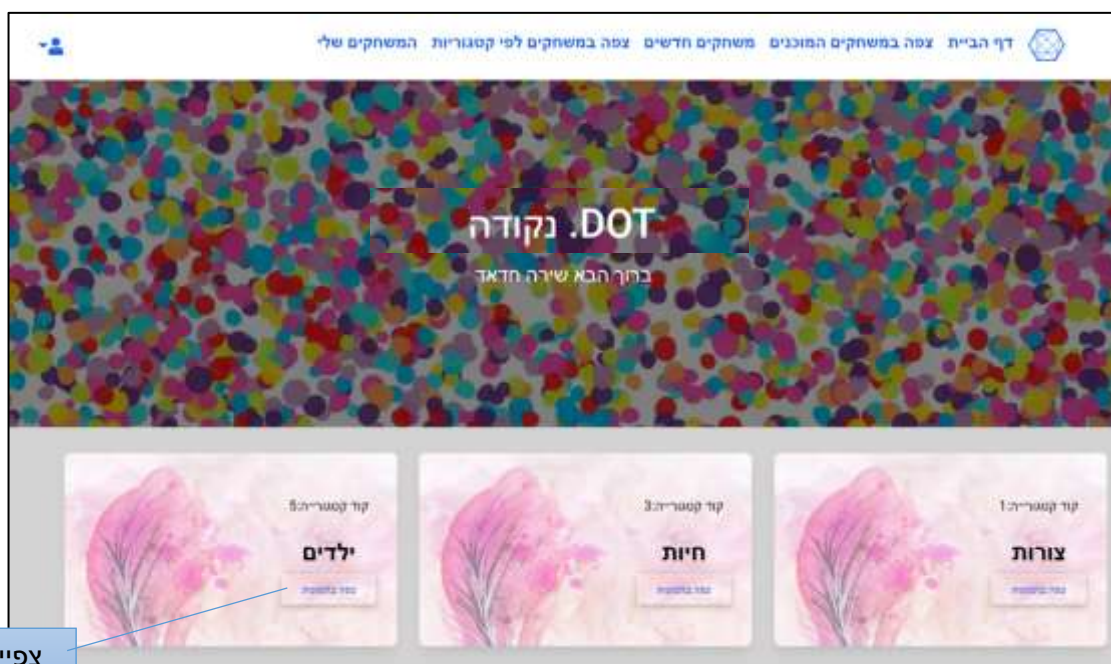
הפיכת תמונה
למשחק

כאשר 'בחר בשיטת מספור סדרה הנדסית או חשבונית הוא יידרש להכניס את האיבר הראשון בסדרה ואת ההפרש כפי שניתן לראות בתמונה הבאה.

שירה חדאד-מנקודה לנקודה



❖ קטגוריות משחקים



צפייה במשחקים
מאותה קטגורייה

❖ משחקים שהמשתמש העלה

המשתמש יראה את רשימת התמונות שהוא העלה לאתר.

שירה חדאד-מנקודה לנקודה

דף הבית צפה במשחקים המוכנים משחקים חדשים צפה במשחקים לפי קטגוריות המשחקים שלי

DOT נקודה

בחור הבא שירה חדאד

הורדת התמונה

מיקי שלם חידון.png

תמונת מקרי

שם משחק

הורד את המשחק כקובץ

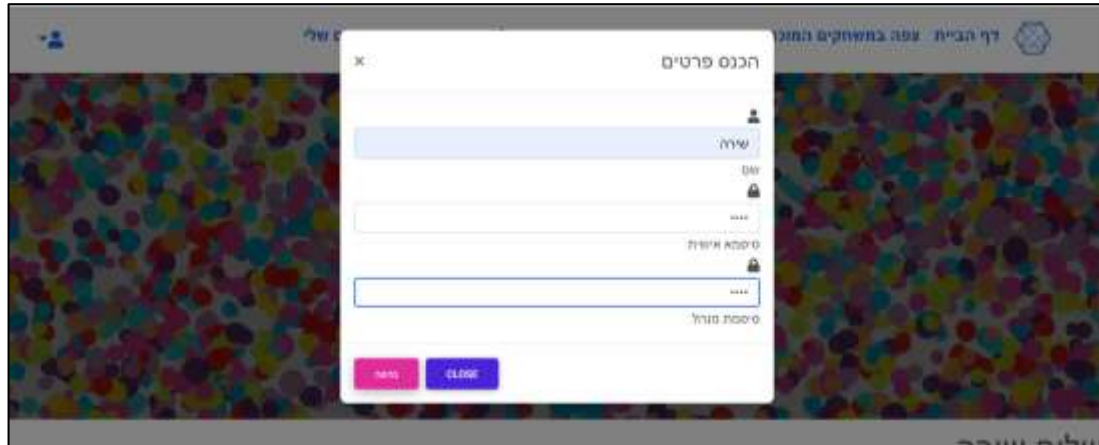
מיקי ראש חידון.png

מיקי ראש חידון.png

מסכים למנהל:

❖ כניסת מנהל

כמו משתמש רגיל בתוספת סיסמת מנהל מיוחדת המאוחסנת בקובץ xml ורק למנהל יש גישה אליו ע"מ לשנותה.



❖ ניהול המשחקים

למנהל מתווספות היכולות של עדכון או מחיקת משחק. ביכולתו להעלות תמונת משחק מוכנה כבר אל האתר כן לעדכן או למחק תמונה קיימת.

שירה חדאד-מנקודה לנקודה

The screenshot shows the DOT.נקודה website. At the top, there is a navigation bar with links: דף הבית, צגה במחשבים, מחשבים ודפוסים, צגה במחשבים לפי קטגוריות, מחשבים שלי, and רשימת התכנים. The main header features the text "DOT.נקודה" and "ברוך הבא שירה חדאד". Below this, there are two main panels. The left panel, titled "העלאת תמונה", has a "לפי קוד" button. The right panel, titled "משחק מוכן (ללא הגבלה בסיומת)", has a "עדכון פרטי התמונה" button. Below these panels, there are three cards showing different patterns of dots forming shapes, each with a "קוד" and "תמונה" button.

מחיקת התמונה לפי קוד

העלאת תמונה של משחק מוכן (ללא הגבלה בסיומת)

עדכון פרטי התמונה

❖ רשימת משתמשים

כולל יכולת למחוק משתמש. כאשר מנהל ינסה למחוק תוצג לו הודעת אזהרה ויצטרך להקיש לאישור שנית.

שירה חדאד-מנקודה לנקודה

דף הבית צפה במשחקים המוכנים משחקים חדשים צפה במשחקים לפי קטגוריות המשחקים שלי רשימת החברים

נקודה DOT

ברוך הבא שירה חדאד

שם משתמש	שם משתמש	קוד משתמש	שם משתמש	שם משתמש	שם משתמש
מנהל	מנהל	1	11111111	11111111	11111111
שירה	שירה	3	1234	1234	1234
ר	ר	5	1234567	1234567	1234567
וי	וי	7	1234567	1234567	1234567
string	string	8	string	string	string
string	string	9	string	string	string
dana	dana	12	string	string	string
שיר	שיר	13	y1234567	y1234567	y1234567
ליאת	ליאת	14	y1234567	y1234567	y1234567
אפרת	אפרת	16	e1234567	e1234567	e1234567
אנדרית	אנדרית	17	w1234567	w1234567	w1234567

מחיקת המשתמש

❖ הוספה והסרת קטגוריות

מתווספות היכולות של עדכון שם קטגוריה ומחיקה.

דף הבית צפה במשחקים המוכנים משחקים חדשים צפה במשחקים לפי קטגוריות המשחקים שלי רשימת החברים

נקודה DOT

ברוך הבא שירה חדאד

קוד קטגוריה: 5

ילדים

הוסף קטגוריה

קוד קטגוריה: 3

חיות

הוסף קטגוריה

קוד קטגוריה: 1

צורות

הוסף קטגוריה

הוסף קטגוריה

הוסף קטגוריה

הוסף קטגוריה

הוסף קטגוריה

הוסף קטגוריה

הוסף קטגוריה

עדכון שם קטגוריה

הוספת קטגוריה חדשה

מחיקת קטגוריה מסוימת עפ"י קוד

קוד התכנית + תיעוד

האלגוריתמים בחישוב זה כתובים במחלקת `ImageProcess` והזימון של כל הפונקציות מתבצע מהפונקציה `CreatePoints`.

לצורך כלל החישובים יצרתי מספר משתנים:

☒ משתנים המייצגים צבעים:

```
const Int32 avg = 200;
const Int32 FF = 255;
const Int32 zero = 0;
const double Red = .299;
const double Green = .587;
const double Blue = .114;
```

☒ משתני רשימות שיאחסנו את הנקודות החוזרות מהרצת האלגוריתמים.

```
List<Point> lstCorners = new List<Point>(); //רשימה של נקודות הפינות
```

```
List<Point> lstBorder = new List<Point>(); //רשימה של נקודות גבול
List<Point> lstPoints = new List<Point>(); //רשימה של הנקודות הסופיות
למיספור
```

☒ משתנה תמונה שיכיל את התמונה המתקבלת ועליה אנחנו עובדים:

```
public Bitmap Picture { get; set; }
```

☒ רשימה שתכיל את כל הסימונים (אותיות/מספרים) שיהיו על הנקודות למספור:

```
List<string> lstSign = new List<string>(); //למספור הנקודות על הסימונים רשימת
```

תיאור האלגוריתם שלב אחרי שלב:

האלגוריתם מחולק למספר שלבים המבוצעים שלב שלב בפונקציה הראשית: `CreatePoints`:

```
public void CreatePoints()
{
    Region();
    find_corners(); //קצה נקודות
    FindBorder(); //תמונה גבול מציאת
    lstCorners = ReduceList(lstCorners); //דילול
    lstBorder = ReduceList(lstBorder);
    lstPoints = intersectLists(lstBorder, lstCorners); //איחוד
    fillSeries(1, 1);
    אופציות מילוי המיספור: (הסבר בהמשך)
    //fillLetters('a');
```



```
//fillLetters('א');
// fillEngineeringSeries(2, 2, '*');
// fillEngineeringSeries(100, 10, '/');
Picture = countingMap(1stSign);
```

}

שלב 1:

בתחילה הפכתי את התמונה המתקבלת לגווני אפור ואח"כ הפכתי אותה לשחור לבן כדי שיהיה אפשר לעבד אותה.

```
public void ToGray()//הופך תמונה לאפור
{
    Bitmap picMap = (Bitmap)Picture.Clone();
    Color c;

    for (int i = 0; i < picMap.Width; i++)
    {
        for (int j = 0; j < picMap.Height; j++)
        {
            c = picMap.GetPixel(i, j);
            byte gray = (byte)(Red * c.R + Green * c.G + Blue * c.B);
            picMap.SetPixel(i, j, Color.FromArgb(gray, gray, gray));
        }
    }
    Picture = (Bitmap)picMap.Clone();
}
```

```
public void ToBlackWhite()//הופך תמונה לשחור לבן
{
    Bitmap picMap = Picture;
    Color newColor;

    for (int i = 0; i < picMap.Width; i++)
    {
        for (int j = 0; j < picMap.Height; j++)
        {
            Color pixelColor = picMap.GetPixel(i, j);
            if (pixelColor.R <= avg && pixelColor.G <= avg && pixelColor.B <= avg)
                newColor = Color.FromArgb(zero, zero, zero);//שחור
            else
                newColor = Color.FromArgb(FF, FF, FF);//לבן
            picMap.SetPixel(i, j, newColor);
        }
    }
    Picture = picMap;
}
```

בשלב זה גם ביצעתי ניקוי רעשי רקע מהתמונה:

```
public Bitmap PerformClean(Bitmap bitmap)
{
    var newBitmap = new UnsafeBitmap(bitmap);
    newBitmap.LockBitmap();

    PixelData color;
    var mask = new int[9];
    for (var i = 0; i < 9; i++)
        mask[i] = 1;

    for (var i = 0; i < bitmap.Width; i++)
    {
        for (var j = 0; j < bitmap.Height; j++)
        {
            if (i - 1 >= 0 && j - 1 >= 0)
            {
                color = newBitmap.GetPixel(i - 1, j - 1);
                mask[0] = Convert.ToInt16(color.GetBrightness() * 255);
            }
            else
                mask[0] = 255;

            if (j - 1 >= 0 && i + 1 < bitmap.Width)
            {
                color = newBitmap.GetPixel(i + 1, j - 1);
                mask[1] = Convert.ToInt16(color.GetBrightness() * 255);
            }
            else
                mask[1] = 255;

            if (j - 1 >= 0)
            {
                color = newBitmap.GetPixel(i, j - 1);
                mask[2] = Convert.ToInt16(color.GetBrightness() * 255);
            }
            else
                mask[2] = 255;

            if (i + 1 < bitmap.Width)
            {
                color = newBitmap.GetPixel(i + 1, j);
                mask[3] = Convert.ToInt16(color.GetBrightness() * 255);
            }
            else
                mask[3] = 255;

            if (i - 1 >= 0)
            {
                color = newBitmap.GetPixel(i - 1, j);
                mask[4] = Convert.ToInt16(color.GetBrightness() * 255);
            }
            else
                mask[4] = 255;

            if (i - 1 >= 0 && j + 1 < bitmap.Height)
            {
                color = newBitmap.GetPixel(i - 1, j + 1);
```

שירה חדאד-מנקודה לנקודה

```
mask[5] = Convert.ToInt16(color.GetBrightness() * 255);
}
else
    mask[5] = 255;

if (j + 1 < bitmap.Height)
{
    color = newBitmap.GetPixel(i, j + 1);
    mask[6] = Convert.ToInt16(color.GetBrightness() * 255);
}
else
    mask[6] = 255;

if (i + 1 < bitmap.Width && j + 1 < bitmap.Height)
{
    color = newBitmap.GetPixel(i + 1, j + 1);
    mask[7] = Convert.ToInt16(color.GetBrightness() * 255);
}
else
    mask[7] = 255;

color = newBitmap.GetPixel(i, j);
mask[8] = Convert.ToInt16(color.GetBrightness() * 255);

Array.Sort(mask);
var mid = mask[4];
newBitmap.SetPixel(i, j, new PixelData { B = (byte)mid, R =
(byte)mid, G = (byte)mid });
}
}
newBitmap.UnlockBitmap();
return newBitmap.Bitmap;
}
```

שלב 2:

כעת מצאתי את מספר הרכיבים הקשירים בתמונה ע"י אלגוריתם למציאת רכיבים קשירים:
סרקתי את התמונה ולכל פיקסל נתתי תווית עפ"י השכנים שלו.

```
public int Region()
{
    int y = 0;
    tempPicture = new int[Picture.Width, Picture.Height];

    DisjointSet ds = new DisjointSet(100); // מופע מחלקת חישוב קבוצות זרות

    int tav = 0;
    for (int x = 0; x < tempPicture.GetLength(0); x++)
    {
        for (y = 0; y < tempPicture.GetLength(1); y++)
        {
            // הפיקסל בתמונה המקורית הוא שחור
            if (Picture.GetPixel(x, y) == Color.FromArgb(zero, zero, zero))
            {
                int befor = 0, on = 0;
                if (x > 0)
                    on = tempPicture[x - 1, y];
                if (y > 0)
                    befor = tempPicture[x, y - 1];
            }
        }
    }
}
```

שירה חדאד-מנקודה לנקודה

```

        if (befor == 0 && on == 0) // לשני השכנים
        {
            tempPicture[x, y] = ++tav; // לפיקסל תווית חדשה
        }
        else
        {
            // אם יש לשני השכנים את אותה תווית העתקתי אותה
            if (befor == on)
                tempPicture[x, y] = on;
            // אם רק לשכן מעל או לפניו יש תווית העתק אותה
            else
                if ((befor != 0 && on == 0) || (on != 0 && befor == 0))
                {
                    if (befor != 0)
                        tempPicture[x, y] = befor;
                    else
                        tempPicture[x, y] = on;
                }
            // אם לשניהם תוויות שונות קח את זו של השכן מעל וסמן אותם כשקולות בטבלת השקילות
            else
                if (befor != on)
                {
                    tempPicture[x, y] = on;
                    ds.union(befor, on);
                }
        }
    }
}
}

```

עכשיו אני עוברת על כל קבוצת שקילות ומוצאת את התווית בעלת הערך הקטן ביותר.

```
//מצא את התווית בעלת הערך הקטן ביותר עבור כל קבוצת שקילות
for (int i = 0; i < tempPicture.GetLength(0); i++)
    for (int j = 0; j < tempPicture.GetLength(1); j++)
    {
        if (tempPicture[i, j] != 0)
            tempPicture[i, j] = ds.find(tempPicture[i, j]);
    }
return tav;
```

שלב 3:

אם התמונה חזרה מתאימה במספר הרכיבים בה הפעלתי עליה המרה נוספת ל double כדי שאוכל לבצע עליה את החישובים בהמשך:

שירה חדאד-מנקודה לנקודה

```
public double[,] ToDouble(Bitmap image)
{
    //double ממיר תמונה ל
    var r = new double[image.Width, image.Height];
    var g = new double[image.Width, image.Height];
    var b = new double[image.Width, image.Height];
    var a = new double[image.Width, image.Height];
    double[,] res = new double[image.Width, image.Height];

    unsafe
    {
        BitmapData bitmapData = image.LockBits(
            new Rectangle(0, 0, image.Width, image.Height), ImageLockMode.ReadWrite, image.PixelFormat);

        int bytesPerPixel = Bitmap.GetPixelFormatSize(image.PixelFormat) / 8;
        int heightInPixels = bitmapData.Height;
        int widthInBytes = bitmapData.Width * bytesPerPixel;
        byte* PtrFirstPixel = (byte*)bitmapData.Scan0;

        if (bytesPerPixel == 3)
        {
            for (int y = 0; y < heightInPixels; y++)
            {
                byte* currentLine = PtrFirstPixel + (y * bitmapData.Stride);
                for (int x = 1; x <= widthInBytes; x = x + bytesPerPixel)
                {
                    r[(x - 1) / bytesPerPixel, y] = (double)currentLine[x + 1] / 255;
                    g[(x - 1) / bytesPerPixel, y] = (double)currentLine[x] / 255;
                    b[(x - 1) / bytesPerPixel, y] = (double)currentLine[x - 1] / 255;
                    a[(x - 1) / bytesPerPixel, y] = 0.0d;
                }
            }
        }
        else
        {
            for (int y = 0; y < heightInPixels; y++)
            {
                byte* currentLine = PtrFirstPixel + (y * bitmapData.Stride);
                for (int x = 1; x <= widthInBytes; x = x + bytesPerPixel)
                {
                    r[(x - 1) / bytesPerPixel, y] = (double)currentLine[x + 1] / 255;
                    g[(x - 1) / bytesPerPixel, y] = (double)currentLine[x] / 255;
                    b[(x - 1) / bytesPerPixel, y] = (double)currentLine[x - 1] / 255;
                    a[(x - 1) / bytesPerPixel, y] = (double)currentLine[x + 2] / 255;
                }
            }
            image.UnlockBits(bitmapData);
        }

        for (int i = 0; i < res.GetLength(0); i++)
            for (int j = 0; j < res.GetLength(1); j++)
            {
                res[i, j] = Red * r[i, j] + Green * g[i, j] + Blue * b[i, j];
            }

        return res;
    }
}
```

לאחר מכן הפעלתי עליה את פונקציית find_corners שתפקידה למצוא את נקודות הקצה בתמונה. הפונקציה מבוססת על האלגוריתם של מורבץ למציאת נקודות הקצה בתמונה.

אלגוריתם מורבץ מזהה את הנקודות עם השונות הגבוהה בבירות בכיוונים שונים. (שזה המצב בעיקר בנקודות שהן פינות). עבור כל נקודה p בתמונה מגדירים חלון ריבועי קטן

סביבה. ואז מזיזים את החלון בשמונה כיוונים שונים (אופקי אנכי ושני אלכסונים) בצעד של נקודה אחת אח"כ מסכמים את ריבועי ההפרשים בין הנקודות המתאימות בחלון המקורי והמוזז. השונות של הנקודה P היא השונות המינימלית בין המחושבות עבור כל אחד מהכיוונים.

הפונקציה שלי עובדת כך:

עבור כל פיקסל שלחתי לפונקציית החישוב calculate_point שמוצאת את ההפרש המינימאלי בין הנקודה לאלו הסובבות לה. תוך כדי היעזרות בפונקציה calculate המקבלת קורדינציות של פינה שמאלית עליונה של שתי מטריצות ומחזירה את סכום ריבוע הפרשי הערכים בין נקודות תואמות במטריצה, הפונקציה מקבלת גם את גודל החלון(7).

```
public void find_corners()
{
    //שליחה לפונקציה שאני כתבתי הממירה את התמונה ל/
    double[,] mat = ToDouble(Picture);
    double max = 0;
    double[,] mat2 = new double[mat.GetLength(0), mat.GetLength(1)];
    for (int i = 2; i < mat.GetLength(0) - 8; i++)
        for (int j = 2; j < mat.GetLength(1) - 8; j++)
            mat2[i, j] = calculate_point(mat, i, j, 7);
    //מציאת מקסימום
    for (int i = 2; i < mat2.GetLength(0) - 8; i++)
        for (int j = 2; j < mat2.GetLength(1) - 8; j++)
            if (mat2[i, j] > max)
                max = mat2[i, j];
    int cnt = 0;

    //יצרתי מערך מונים כדי לשכן בתוכו את השונויות
    int[] arr = new int[(int)max + 1];
    for (int i = 2; i < mat2.GetLength(0) - 10; i++)
        for (int j = 2; j < mat2.GetLength(1) - 10; j++)
        {
            arr[(int)(mat2[i, j])]++;
        }
}
```

עכשיו אני רוצה לקחת רק מאית מתוך כל הנקודות שיצאו ולשים אותם ברשימת נקודות העניין.

```
double sum = 0, border=max;
for (int i=arr.Length-1;i>=0;i--)
{
    sum += arr[i];
    if(sum/ (mat2.GetLength(0)* mat2.GetLength(1))>0.001)
    {
        border = i;
        break;
    }
}
for (int i = 2; i < mat2.GetLength(0) - 10; i++)
    for (int j = 2; j < mat2.GetLength(1) - 10; j++)
    {
        if (mat2[i, j] >= border)// אם ערך השונויות הוא אכן חלק מהמאית
        {
            lstCorners.Add(new Point(i, j)); //תוסיף את הנקודה לרשימה
        }
    }
}
```

```

    }

}

}
```

לאחר פונקצייה זו יש לנו רשימה המכילה את הנקודות קצה בתמונה-lstCorners .

שלב 4 :

כעת מצאתי בתמונה המקורית את נקודות הגבול-קווי המתאר של התמונה לצורך החישוב בהמשך.

חישוב זה נעשה בעזרת פונקצייה FindBorder שלאחריה נקודות הגבול יאוחסנו ברשימה lstBorder.

בפונקצייה מצאתי תחילה את הנקודה השחורה הראשונה בתמונה ולאחר מכן עבור כל פיקסל עברתי על כל השכנים החל מהשכן המערבי עד למציאת פיקסל שחור שיוכנס לרשימה lstBorder.

```

public void FindBorder()
{
    int x, y = 0, b = 0;
    Point start = new Point(0, 0); //התחלה
    int step = 1;
    lstBorder.Clear(); //האיברים כל האיברים
    int[,] mat = new int[2, 8];
```

לצורך מעבר על כל שכניו של הפיקסל יצרתי מטריצה שתכיל את הפרש הקורדינציות בין הנקודה הנוכחית לכל אחד מהשכנים (8 שכנים מסביב).
המטריצה מאותחלת כך:

-1	-1	0	-1	0	1	1	0
0	-1	-1	1	1	1	0	-1

```

    //מציאת נקודת ההתחלה- הפיקסל השחור הראשון
    for (x = 0; x < Picture.Width; x++)
    {
        for (y = 0; y < Picture.Height; y++)
            if (Picture.GetPixel(x, y) == Color.FromArgb(zero, zero, zero)) //
                אם הנקודה שחורה
                {
                    start = new Point(x, y); //אז תתחיל מהנקודה הזו
                    break; //תעצור
                }
            if (y < Picture.Height) //אם נעצרת באמצע
                continue;
```

```

        break; // תעצור
    }

    // מעבר על כל השכנים החל מהשכן המערבי עד למציאת פיקסל שחור
    do
    {
        int i = 0;
        for (i = 0; i < 8; i++)
        {
            // תקינות- שאין יציאה מהגבולות
            if (x + mat[0, b]*step >= 0 && x + mat[0, b]*step < Picture.Width &&
                y + mat[1, b]*step >= 0 && y + mat[1, b]*step < Picture.Height)
                if (Picture.GetPixel(x + mat[0, b] * step, y + mat[1, b] * step) ==
                    Color.FromArgb(zero, zero, zero)) // אם גם אתה שחור
                {
                    lstBorder.Add(new
                    Point(x+mat[0,b]*step,y+mat[1,b]*step)); // תוסיף לי הנקודה
                    x = x + mat[0, b] * step;
                    y = y + mat[1, b] * step;
                    b = b - 1;

                    if (b < 0)
                    {
                        b = 7;
                        step = 1;
                        break;
                    }
                }
            else
            {
                b++;
                if (b == 8)
                    b = 0;
            }
        }
        if (i == 8)
        {
            step++;
        }
    }

    while (x != start.X || y != start.Y);
}

```

שלב 5:

מאחר וכל תמונה מורכבת ממספר עצום של פיקסלים היה עלי לצמצם את הפיקסלים שברשימות. על כן הפונקציה הבאה תצמצם באופן יחסי לפי מיקום(בעזרת פונקציית חישוב מרחק: FindDistance) ותיקח נקודות עיקריות מהרשימה. לפונקצייה קוראים ReduceList שמקבלת רשימה ומחזירה אותה מעודכנת.

לפונקצייה הזו אנו שלחתי את שתי הרשימות:


```
lstCorners = ReduceList(lstCorners);
lstBorder = ReduceList(lstBorder);
```

```
public List<Point> ReduceList(List<Point> list)//דילול פונקציה
```

```
{
    List<Point> lst = new List<Point>();
    while (list.Count > 0)
    {
        Point p = new Point(list[0].X, list[0].Y);
        עבור כל נקודה אני מוחקת את כל הנקודות שהמרחק ממנה אליהן קטן מ 30 לפי פונקציה מרחק
        שמופיעה למטה.
        list.RemoveAll(p1 => FindDistance(p, p1) < 30);
        lst.Add(p);
    }
    return lst;
}
```

פונקציה זו משתמשת בפונקציית חישוב מרחק בין 2 נקודות:

```
public double FindDistance(Point p1, Point p2)
{
    return Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y - p2.Y, 2));
}
```

שלב 6:

לאחר מכן נצמצם את שתי הרשימות לרשימה אחת בה יופיעו רק הנקודות המשותפות. ע"י קריאה לפונקציה שמקבלת את שתי הרשימות.

```
intersectLists(lstBorder, lstCorners)
```

הנקודות המשותפות בין נקודות הענין לבין פיקסלי הגבול הן אלה שהמרחק ביניהן קטן מ 150

```
public List<Point> intersectLists(List<Point> lst1, List<Point> lst2)//מאחד את
הפיקסלים המשותפים בין פיקסלי הגבול והפינות
```

```
{
    List<Point> lst = new List<Point>();
    foreach (Point point in lst1)
    {
        foreach (Point point1 in lst2)
        {
            if (FindDistance(point, point1) < 150)
            {
                lst.Add(point);
                break;
            }
        }
    }
    return lst;
}
```

לאחר השלב הזה נגיע לשלב הסופי של מספור הנקודות המשותפות שחזרו מהפונקציה הקודמת.

לפני כן אני קודם כל ממלאה רשימה בגודל רשימת הנקודות שנמצאו ובה יהיה את המספור המתאים לפי מה שבחר המשתמש(מספור רגיל, כפולות, חילוק וכו). הנקודות הסופיות יאוחסנו ברשימה:

lstSign.

אלו הפונקציות מילוי:

פונקצית מילוי עבור אותיות עבריות או אנגליות

```
public void fillLetters(char a1) //מילוי אותיות עבריות או אנגליות
{
    char cnt = a1;
    משתנה שמכיל האם ממלאים אותיות עבריות או אנגליות.
    bool isEnglish = a1 == 'א' ? true : false;
    for (int i = 0; i < lstPoints.Count; i++)
    {
```

כאן דאגתי שהאות יהיה תקין ולכן אם הגענו לאות האחרונה (א או ז) יתחיל האיות שוב מהתחלה.

```
        if (isEnglish && cnt > 'z')
            cnt = a1;
        else if (!isEnglish && cnt > 'ת')
            cnt = a1;

        lstSign.Add(cnt.ToString());
        cnt++;
    }
}
```

פונקצית מילוי עבור סדרות חשבוניות

```
public void fillSeries(int a1, int d) //סדרות חשבוניות
{
    int cnt = a1;
    for (int i = 0; i < lstPoints.Count; i++)
    {
        lstSign.Add(cnt.ToString());

        cnt += d;
    }
}
```

פונקצית מילוי עבור סדרות הנדסאיות

```
public void fillEngineeringSeries(int a1, int d, char op) //סדרות כפל או חילוק
{
    int cnt = a1;
    for (int i = 0; i < lstPoints.Count; i++)
    {
        lstSign.Add( cnt.ToString());
```

גם כאן דאגתי שהמספור יהיה מותאם לגילאי המשחקים בו ולכן המספור לא יעלה על 100 ואם הגיע כבר ל-100 יתחיל המספור שוב מהתחלה.

```
        if (cnt >= 100 || cnt <= 0)
            cnt = a1;
```

```

switch (op)
{
    case '*':
        cnt *= d; break;
    case '/':
        cnt /= d; break;
}
}
}

```

שלב 7:

לאחר מכן שלחתי לפונקציית המספור שממספרת לפי רשימה זו שמילאנו קודם באחת הפונקציות (כנ"ל). לפונקציה זו קוראים:

countingMap

שמקבלת את הרשימה שמילאנו ותעבור על הנקודות ותמספר אותם.

פונקציה זו משתמשת בפונקצייה למציאת האיבר הבא-

find_next_point.

// מספור תמונה

```

public Bitmap countingMap(List<string> lstSign)
{
    אתחול משתנים להמשך:
    Bitmap bitmap = new Bitmap(Picture.Width, Picture.Height, Picture.PixelFormat);

    Graphics graphicsImage = Graphics.FromImage(bitmap);

    StringFormat stringformat2 = new StringFormat();
    stringformat2.Alignment = StringAlignment.Center;
    stringformat2.LineAlignment = StringAlignment.Center;
    Color StringColor2 = System.Drawing.ColorTranslator.FromHtml("#000000");
    string Str_TextOnImage2 = lstSign[0]; // המספר(או האות) ההתחלתי
}

```

עכשיו אני מציירת את הראשון על גבי

graphicsImage

שאתחלתי מראש. המספר הראשון יהיה מודגש להקל על הילד המשחק.

```

graphicsImage.DrawString(Str_TextOnImage2,
    new Font("Arial", 10, FontStyle.Bold),
    new SolidBrush(StringColor2),
    new Point(lstPoints[0].X - 10, lstPoints[0].Y), stringformat2);

    ציור הנקודה שתהיה ליד המספר
    graphicsImage.DrawString(((char)183).ToString(),
        new Font("Arial", 8, FontStyle.Regular),
        new SolidBrush(StringColor2),
        new Point(lstPoints[0].X, lstPoints[0].Y), stringformat2);
}

```

```

int index = 0;
int x = find_next_point(0);

```

עוברים על כל הנקודות ברשימה

```
while (x != 0)
```

שירה חדאד-מנקודה לנקודה

```
{
    index++;
    for (int i = lstPoints[x].X; i < bitmap.Height && i < lstPoints[x].X + 5; i++)
        for (int j = lstPoints[x].Y; j < lstPoints[x].Y + 5 && j < bitmap.Width; j++)
            graphicsImage = Graphics.FromImage(bitmap);
            stringformat2 = new StringFormat();
            stringformat2.Alignment = StringAlignment.Center;
            stringformat2.LineAlignment = StringAlignment.Center;
            StringColor2 = System.Drawing.ColorTranslator.FromHtml("#000000");
            Str_TextOnImage2 = lstSign[index];
            graphicsImage.DrawString(Str_TextOnImage2, new Font("david", 8,
                FontStyle.Regular),
                new SolidBrush(StringColor2),
                new Point(lstPoints[x].X - 10,
                    lstPoints[x].Y), stringformat2);

    ציור הנקודה שתהיה ליד המספר
        graphicsImage.DrawString(((char)183).ToString(),
            new Font("Arial", 6, FontStyle.Regular),
            new SolidBrush(StringColor2),
            new Point(lstPoints[x].X, lstPoints[x].Y),
            stringformat2);

    אמצאות הנקודה הבאה לתוך המשתנה.
        x = find_next_point(x);
    }
    lstSign.Clear();
    return (Bitmap)bitmap.Clone();
}

public int find_next_point(int index)
{
    if (index == lstPoints.Count - 1)
        return 0;
    return index + 1;
}
```

קודים נוספים מצד השרת:

חישוב הדירוג:

קוד המחשב את הדירוג של התמונה באופן יחסי בין מספר ההורדות שלה למספר ההורדות של כל התמונות:

```
public List<PictureTbl> cascading(PictureTbl p,int id)
{
    int countAllDownlods = 0;
    if (p.CodeUser != id)
    {
        try
        {
            foreach (var item in DB.PictureTbls)//סכימת ההורדות של כל התמונות
            {
                countAllDownlods += Convert.ToInt32(item.CountOfDownloads);
            }

            if (p.CountOfDownloads ==null)
                p.CountOfDownloads = 0;
            p.CountOfDownloads += 1;//עדכון ההורדות של התמונה הנוכחית
            UpddatePicture(p, p.CodePicture);
            int max = 0;
            List<PictureTbl> l = new List<PictureTbl>();
            l = GetAllPicture();
            foreach (var item in l)
            {
                if (item.CountOfDownloads > max)
                    max = Convert.ToInt32(item.CountOfDownloads);
            }

            //חישוב כמה הורדות שווה כל כוכב - המרה מאחוזים לכמות הורדות

            double everyStar = Convert.ToDouble((20 * max) / 100);
            //חמישית ממספר ההורדות המקסימאלי
            if(everyStar!=0)
                p.CascadingPicture =Convert.ToInt32( p.CountOfDownloads / everyStar);

            return UpddatePicture(p, p.CodePicture);
        }
        catch (Exception e)
        {
            throw e;
        }
    }
    else return UpddatePicture(p, p.CodePicture);
}
```

קוד העלאת התמונה אל הצד שרת:

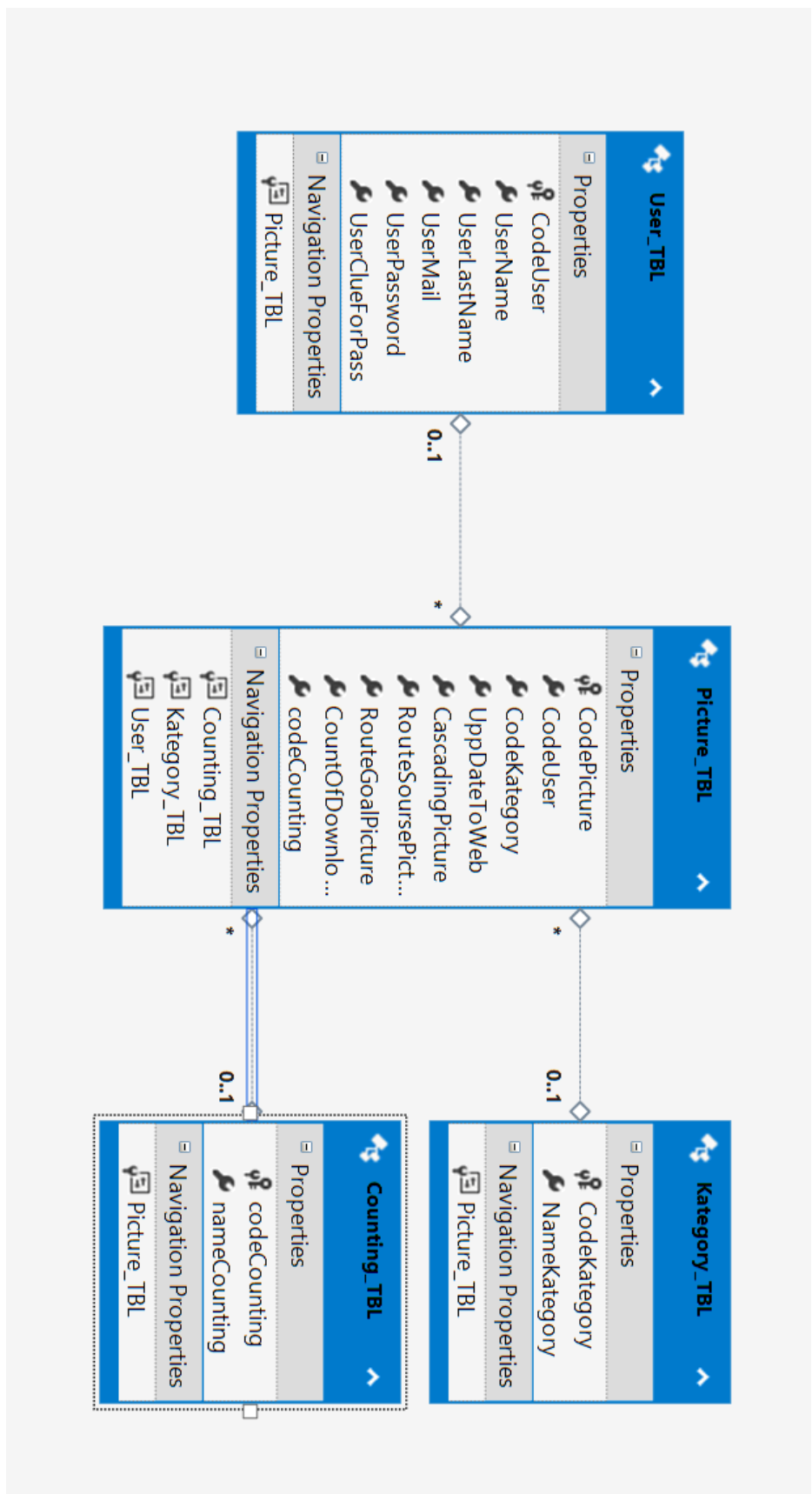
```

public static string fName { get; set; } //משתנה שיכיל את שם הקובץ שיעלה
[HttpPost("upload"), DisableRequestSizeLimit]
0 references
public IActionResult upload()
{
    try
    {
        var file = Request.Form.Files[0];
        var folderName = Path.Combine("wwwroot", "images");//wwwroot(מראש) שהכנתי בתיקייה
        var pathToSave = Path.Combine(Directory.GetCurrentDirectory(), folderName);

        if (file.Length > 0)
        {
            var fileName = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('"');
            fName = fileName.ToString();
            var fullPath = Path.Combine(pathToSave, fileName);
            var dbPath = Path.Combine(folderName, fileName);
            using (var stream = new FileStream(fullPath, FileMode.Create))
            {
                file.CopyTo(stream);
            }
            return Ok(new { dbPath });
            return Ok();
        }
        else
        {
            return BadRequest();
        }
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error: {ex}");//שגיאה שתיזרק אם ההעלאה תיכשל/
    }
}

```

תיאור מסד הנתונים



תרשים טבלאות

מסד הנתונים מכיל מספר טבלאות:

1. תמונות המשחקים המוכנים.
2. משתמשים.
3. קטגוריות.
4. סוגי מספור

תרשים טבלאות וקשרי גומלין:

משחקים - Picture_TBL

השדה בטבלה	האם הוא שדה מפתח?	טיפוס הנתונים	האם הוא שדה חובה?	האם קיים ערך ברירת מחדל?
CodePicture	כן-מפתח ראשי	מספר	כן	כן-ערך אוטומטי בסדר עולה.
CodeUser	כן-מפתח זר	מספר	לא	לא
CodeCategory	כן-מפתח זר	מספר	לא	לא
UppDateToWeb	לא	תאריך	לא	לא
CascadingPicture	לא	מספר	לא	לא
RouteSourcePicture	לא	מחרוזת	לא	לא
RouteGoalPicture	לא	מחרוזת	לא	לא
CountOfDownloads	לא	מספר	לא	לא
codeCounting	כן-מפתח זר	מספר	לא	לא

משתמשים : User_TBL

השדה בטבלה	האם הוא שדה מפתח?	טיפוס הנתונים	האם הוא שדה חובה?	האם קיים ערך ברירת מחדל?
CodeUser	כן-מפתח ראשי	מספר	כן	כן-ערך אוטומטי בסדר עולה.
UserName	כן-מפתח זר	מחרוזת	כן-בכניסה	לא
UserLastName	כן-מפתח זר	מחרוזת	לא	לא
UserMail	לא	מחרוזת	לא	לא
UserPassword	לא	מחרוזת	כן-בכניסה	לא
UserClueForPass	לא	מחרוזת	לא	לא

קטגורי ות: Category_TBL

השדה בטבלה	האם הוא שדה מפתח?	טיפוס הנתונים	האם הוא שדה חובה?	האם קיים ערך ברירת מחדל?
CodeCategory	כן-ראשי	מפתח מספר	כן	כן-ערך אוטומטי בסדר עולה.
NameCategory	לא	מחרוזת	לא	לא

סוגי מספור: Counting_TBL

השדה בטבלה	האם הוא שדה מפתח?	טיפוס הנתונים	האם הוא שדה חובה?	האם קיים ערך ברירת מחדל?
CodeCounting	כן-ראשי	מפתח מספר	כן	כן-ערך אוטומטי בסדר עולה.
NameCounting	לא	מחרוזת	לא	לא

מדריך למשתמש:

בכניסה לאתר על המשתמש להזדהות עם שם וסיסמא על מנת לארגן את המידע השמור במערכת. במידה והמשתמש אינו רשום במערכת, עליו להרשם ע"י הכנסת פרטיו האישיים.

לאחר ההזדהות נפתח למשתמש הדף הראשי ובו יש מספר אפשרויות כמו צפייה במשחקים קיימים והוספת משחק חדש.

בהוספת משחק חדש המשתמש יוכל לבחור את התמונה ואת סוג המספור שלה ע"י בחירת הסוג המתאים(מספור ע"י מספרים,אותיות עבריות,לועזיות,סדרות חשבוניות והנדסאיות..)

בדיקות והערכה:

לאחר הרצת האלגוריתם נבחנו כל התוצאות של כל התמונות שהכנסתי כאשר הופיעו טעויות או אי דיוקים בביצוע של האלגוריתם, נבדק הקוד שוב עד שתוקנו הבעיות.

לאחר בדיקות רבות אחר כל מקרי הקצה שעלו בדעתי,והרצת האלגוריתם מספר פעמים על נתונים שונים, האלגוריתם הגיע לקירוב האפשרי ביותר בכלים העומדים לרשותי.

ניתוח יעילות:

בבואי לתכנן את האלגוריתם בפרויקט נתקלתי רבות בשאלות על היעילות. ביצועי הפרויקט חייבים להיות יעילים שכן זה קשור באופן ישיר לזמן התגובה למשתמש: ברגע שהביצועים גרועים, זמן התגובה למשתמש מתארך והאתר "חושב" הרבה זמן-דבר המעצבן

את המשתמשים ומוריד את האחוזים בשימוש באתר. על כן נתתי חשיבות רבה ליעילות בביצועים ואף לעיתים העדפתי את היעילות על פני דברים אחרים.

ניתוח הסיבוכיות:

הסיבוכיות של הפרויקט שלי היא $O(n^2)$.

פירוט:

פונקציית הפיכה לגונוני אפור, ניקוי רעשים ומציאת רק"חים בתמונה- n^2 . כמעבר רגיל על מטריצה (שורות*עמודות).

בשאר הפונקציות מספר האיטרציות זהה לכמות האיברים החוזרים מהאלגוריתמים השונים (זיהוי נקודות בתמונה): מספר האיטרציות המירבי בהן הוא בפונקציה לאיחוד הרשימות שפועלת בסיבוכיות של $m*k$ כאשר m מייצגת את מספר הנקודות ברשימה של נקודות הגבול ו k מיצג את כמות נקודות העניין בתמונה.

מובן ש n^2 היא הסיבוכיות הגבוהה ביותר.

אבטחת מידע:

המידע של המשתמשים מוגן ע"י סיסמא השמורה במערכת, הכניסה היא רק באמצעות מ.ז. וסיסמא (השילוב של השניים יוצר אבטחה רבה יותר). למרות שזהו אתר שלא דורש כ"כ אבטחה עדיין רשומים במערכת פרטיהם האישיים וכן עצם מציאות הסיסמא האישית יוצרת הרגשת שייכות לאתר.

המידע על המשתמשים מוגן ומוצג רק למנהל שהזדהה בכניסה.

פיתוחים עתידיים

- לאחר הסריקה המשתמש יוכל לבחור חלקים מהתמונה המקורית ולהדביק אותם בתמונת המשחק (כמו עיניים אף וכו'..).
- משתמש שהתמונה שהעלה צברה את הדירוג הגבוה ביותר יקבל הודעה על כך במייל.

מסקנות

ביצוע הפרויקט, הוא יותר מועיל מכל מבחן, והיווה מעין סטאז' עבורי. כשניגשתי לתכנון פרויקט הגמר, חשתי כעומדת בפני דבר גדול מאוד. היקף הפרויקט, הדרישות, תכנון הספר נראה כמשהו אינסופי, אך לאחר תכנון מעמיק ומידור לחלקים הוא נהיה פשוט יותר, והפך לאתגר מהנה.

התכנות עצמו, מלבד המיומנות שהעניק, גם חשף אותי לפונקציות מיוחדות בהן לא דווקא נתקלים בדברים הפשוטים. ניתן לומר, כי הפרויקט תרם לי כסטודנטית וכמתכנתת, הוא הכניס אותי עמוק לתוך המבנה הפנימי היוצר כל תוכנה באשר היא, ניתוח ואפיון של המערכת, וכך אוכל לתכנת פרויקטים נוספים.

נקודה נוספת שקניתי במהלך בניית הפרויקט, היא העבודה היחידנית. עבודה יחידנית דורשת הרבה מאד, הצורך להתמודד לבד בכל המישורים בדברים שמוצלחים בהם יותר ובכאלה שפחות. ההתמודדות מול אתגרים, קשיים ונפילות במהלך כל העבודה נעשים מתעצמים. בעבודה בצורה זו אני מרגישה שרכשתי את הניסיון להתמודד עם כל סוגי העבודות והשלבים במהלך כתיבת פרויקט שלם מה שיקנה לי הרבה לקראת העבודה בעתיד.

לסיכום : לאחר שעות רבות של עמל ויזע, אני מרגישה כי הפקתי תועלת רבה מפרויקט הגמר שיסייע לי בעז"ה בעתיד.

ביבליוגרפיה

לימוד האלגוריתם מורבץ-

ויקיפדיה √

Identifyingcornersvision.pdf √

חקר האלגוריתם ע"י בינה מלאכותית-

https://www.openu.ac.il/lists/mediaserver_documents/academic/cs/Identifyingcornersvision.pdf √

https://elad.cs.technion.ac.il/wp-content/uploads/2018/02/Book_ImageProcessing.pdf √

כתיבת האלגוריתם-

GitHub √

כתיבת צד לקוח react-

StackOverFlow √

MDB √

GitHub √

שלמי תודה

בסיום פרויקט זה ולאחר שעות אינסופיות של חשיבה תכנון וביצוע, אינספור מחיקות, תקלות ושינויים אני רוצה **מאד** להודות לכל צוות המורות המסורות:

לרכזת המורה רחל, למנחה המורה גילה, למורה מירי ולכל שאר המתרגלות

על העזרה המסורה והאכפתית. הדאגה שכל אחת תצא עם קוד עובד שכתבה בעצמה. כל זה כלל לא מובן מאליו ונתן לי תחושה עצומה של סיפוק!!!

תודה רבה רבה.