



**פרויקט גמר 2022**

**סמל מוסד:**

**שם מכללה:** סמינר תורני בית יעקב.

**שם הסטודנט:** תהל לוי.

**ת.ז. הסטודנט:** 212697676

**שם הפרויקט:** EazyShop

תוכן

[1. הצעת פרויקט 3](#_Toc103169958)

[2. מבוא / תקציר 5](#_Toc103169959)

[2.1. הרקע לפרויקט 5](#_Toc103169960)

[2.2 תהליך המחקר 5](#_Toc103169961)

[2.3 סקירת ספרות 5](#_Toc103169962)

[3. מטרות ויעדים 6](#_Toc103169963)

[4. אתגרים 6](#_Toc103169964)

[5. מדדי הצלחה 6](#_Toc103169965)

[6. תיאור המצב הקיים 6](#_Toc103169966)

[**7.** **רקע תאורטי** 7](#_Toc103169967)

[8. ניתוח חלופות מערכתי 7](#_Toc103169968)

[9. תיאור החלופה הנבחרת והנימוקים לבחירה 7](#_Toc103169969)

[10. אפיון המערכת 9](#_Toc103169970)

[10.1. ניתוח דרישות המערכת 9](#_Toc103169971)

[10.2. מודול המערכת 9](#_Toc103169972)

[10.3. אפיון פונקציונאלי 10](#_Toc103169973)

[10.4. ביצועים עיקריים 10](#_Toc103169974)

[10.5. אילוצים 10](#_Toc103169975)

[11. תיאור הארכיטקטורה 10](#_Toc103169976)

[11.1. הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top 10](#_Toc103169977)

[11.2. תיאור הרכיבים בפיתרון 11](#_Toc103169978)

[11.3. תיאור פרוטוקולי התקשורת 12](#_Toc103169979)

[11.4. שרת – לקוח 12](#_Toc103169980)

[12. ניתוח ותרשים use case של המערכת המוצעת 12](#_Toc103169981)

[12.1. רשימת use case 13](#_Toc103169982)

[12.2. תיאור ה-use case העיקריים של המערכת 13](#_Toc103169983)

[12.3. מבני נתונים בהם משתמשים בפרויקט 13](#_Toc103169984)

[12.4. תרשים מחלקות 13](#_Toc103169985)

[12.5. תיאור המחלקות 14](#_Toc103169986)

[**13.** **תיאור התוכנה** 14](#_Toc103169987)

[**14.** **אלגוריתמים מרכזיים** 15](#_Toc103169988)

[**15.** **קוד האלגוריתם** 15](#_Toc103169989)

[**16.** **תיאור מסד הנתונים** 15](#_Toc103169990)

[16.1. פירוט הטבלאות ב- Data Base 16](#_Toc103169991)

[**17.** **מדריך למשתמש** 19](#_Toc103169992)

[17.1. תיאור המסכים 19](#_Toc103169993)

[17.2. מדריך למשתמש 19](#_Toc103169994)

[17.3. צילומי מסכים 19](#_Toc103169995)

[18. בדיקות והערכה 19](#_Toc103169996)

[19. ניתוח יעילות 19](#_Toc103169997)

[20. אבטחת מידע 20](#_Toc103169998)

[21. מסקנות 20](#_Toc103169999)

[22. פיתוח עתידי 20](#_Toc103170000)

[23. ביבליוגרפיה 20](#_Toc103170001)

# **הצעת פרויקט**

**תיאור הפרויקט:**

האפליקציה תאפשר כניסה למשתמש חדש, משתמש קיים, מלקטים ומנהל.

המשתמש החדש יתבקש להכניס פרטים אישיים כגון: שם וטלפון ולאחר שירשם במערכת יוכל לבצע רשימת קניות.

ביצוע הרשימה יתבצע בשני אופנים או על ידי חיפוש מוצרים חופשי או על ידי כניסה לקטגוריות. לאחר שהמשתמש יסיים לערוך את רשימת הקניות האפליקציה תחשב לו את המסלול הקצר ביותר בתוך הסופר .

בסיום הקניה המשתמש יוכל לבחור אם לשמור את הרשימה שערך או לא במידה וזו קניה חד פעמית ולא נשנית.

המשתמש הקיים יוכל לערוך רשימת קניות חדשה או לבחור מבין רשימות קודמות שבחר לשמור במערכת וכן לערוך בהם שינויים.

המלקטים עבודתם היא לאסוף מוצרים מהמדפים, לפי ההזמנה של לקוח מסוים. ולארוז אותם בצורה טובה. זה יסייע להם בכך שהם יוכלו להכניס את הרשימות שעליהם ללקוט ואפליקציה תאחד להם את הרשימות הדומות למסלול אחד וקצר. ובכך תחסוך להם זמן רב בעבודתם.

המנהל יכנס למערכת ויוכל לעדכן שינויים שנעשו בסופר.

**הגדרת הבעיה האלגוריתמית:**

חישוב מסלול יעיל וקצר תוך התחשבות באילוצים השונים.

בתחילה הוגדרה הבעיה כבעיית "הסוכן הנוסע" אך לאחר הבנה של תוכן האלגוריתם הגענו למסקנה שבעיה זו אינה פתירה משום שיעילות זמן הריצה שלה היא אין סופית. ולכן האלגוריתם המקורב ביותר לפתרון זה הוא **greedy** **Dijkstra**.

אלגוריתם זה פותר את הבעיה של מסלולים קצרים ביותר ממקור יחיד עבור גרף מכוון וממושקל, עבור משקולות אי שליליים בלבד.

תוצאת האלגוריתם של דייקסטרה זהה לתוצאת [אלגוריתם בלמן-פורד](https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_%D7%91%D7%9C%D7%9E%D7%9F-%D7%A4%D7%95%D7%A8%D7%93) אך אלגוריתם בלמן-פורד פועל גם על גרפים הכוללים קשתות שמשקלן שלילי )שאצלי אין צורך בכך משום שזו מפה ואין משקולות שליליים(. כמו כן זמן הריצה של אלגוריתם דייקסטרה מהיר יותר.

הבעיה העיקרית היא מציאת הצמתים וחישוב הקשתות מתוך המפה של המבנה ותרגומה לקוד אך בעיה זו אכן פתירה.

האלגוריתם יופעל לאחר שנמצא עבורו 2 מסלולים דומים ברשימותיהם. ויאלץ לסנכרן בין 2 המסלולים ומציאת המסלול הקצר ביותר ובכך יסייע גם למלקטים בעבודתם וגם לצרכים הרוצים לבצע שני קניות במקביל.

הבעיה הנוספת שעומדת לרשותנו במידה והזמן יאפשר הינה מציאת מיקום בתוך מבנה. תכולת המבנים הגדולים כיום מכילה ברזל שיוצר עיוותים במציאת המיקום.

ובכן קיימות מס' גישות לפתרון הניווט במבנה סגור:

**הגישה הראשונה** מדברת על שימוש באותות רדיו קיימים, כגון אותות סלולריים או אותות Wi-Fiבכדי לחשב את המיקום הנוכחי.

**הגישה השנייה** היא להתחיל עם מיקום ידוע המבוסס על לווייני ה -GPS-ובעת שהמכשיר נכנס לתוך הבניין, להתחיל לעקוב אחרי מיקומו של המכשיר באמצעות חיישן התנועה המובנה בו.

**הגישה השלישית** היא הוספת מכשירים חדשים, המבוססים על גלי רדיו אשר ניתן להשתמש בהם בקלות וביעילות בכדי לקבוע את המיקום שלכם באזורים מסוימים אשר ברצונכם לכסות.

**הגישה הרביעית** היא השימוש בגישות אזוטריות אחרות בכדי לקבוע את המיקום הגישות הללו עדיין לא נפרסו ביישומים מעשיים והם עדיין לא פתירות.

בניגוד לגישות האחרות המסתמכות על קישוריות בלוטות Wi-Fi ,או סממנים פיזיים הפזורים בבית העסק הגישה השנייה הינה הפתירה ביותר והזולה ביותר לפיתוח, משום שחיישן זה נמצא כיום בכל סמארטפון .על המשתמש בסך הכול להכניס את מיקומו הראשוני .

בנוסף גישה זו הינה המדויקת ביותר ברמת דיוק של מטר אחד בלבד ולכן נפתור על פי הגישה הזו.

**רקע תאורטי בתחום הפרויקט**:

לכולנו יצא יותר מפעם אחת להיכנס לסופר שאנחנו לא מכירים ולערוך בו קניות. דבר זה עולה לנו זמן רב תוך חזרה על מקומות שביקרנו בהם וחוסר מציאת המוצרים המבוקשים.

מחקרים שונים מראים שכ-%30 מהקונים יוצאים מן החנות מבלי שרכשו לפחות מוצר אחד שהתכוונו לקנות, בין אם לא מצאו אותו או בין בגלל שמיהרו.

הפתרון לכך הוא חישוב המסלול הקצר ביותר תוך התחשבות בכל האילוצים של המשתמש.

**תהליכים עיקריים בפרויקט:**

1. המערכת תחשב את מפת הסופר.
2. המערכת תאפשר ללקוח לקבל את המסלול הקצר על פי רשימת הקניות הנתונה.
3. המערכת תחשב ללקוח את סכום הקניה שלו בכל שלב של הקניה.
4. המערכת תאחד בין שני מסלולים למסלול אחד קצר ביותר על פי דמיון בין הרשימות.

# **מבוא / תקציר**

## הרקע לפרויקט

בפרויקט הגמר מטרת העל היא התנסות ולמידה עצמית אך לצד זה עבודה על נושא מעניין שיסיע לך ליהנות מהעבודה ומהתהליך.

בבחירת הרעיון לפרויקט התייעצתי עם אנשים רבים לגבי רעיון שיסייע לכלל הציבור ולאחר מחשבות רבות הגעתי להחלטה לעשות מסלול מקוצר בסופר. הגעתי לרעיון הזה משום שרציתי לפתח דבר דומה לפני כמה שנים ולא עלה בידי אבל השנה החלטתי לעשות זאת.

בחרתי דווקא ברעיון זה לפרויקט גמר משום שקיים בו אלגוריתם מאתגר וחכם ובנוסף למדנו על  **Dijkstra** במבני נתונים והתחברתי מאוד לרעיון.

האפליקציה שלי מסייעת לצרכן בכך שחוסכת לו זמן רב ובזבוז כסף אדיר. שהרי כשהלקוח מזין את רשימת הקניות שלנו מוצג לו מסלול מדויק שמאפשר לו לבצע את הקנייה שלו במינימום של זמן כמו כן מטרת בעלי הסופרים היא שהלקוחות יוסיפו לעגלה שלהם מבצעים או מוצרים נוספים שלא בהכרח זקוקים להם ולכן האפליקציה שלנו מסייעת ללקוח בהיצמדות לרשימת הקניות שהוכנה לו מראש.

את האפליקציה שייצרתי יצרו רבים לפניי אך קיים פן נוסף האפליקציה שלי מסייעת למלקטים שעובדים בסופר לאחד מספר רשימות ולעבוד על כמה הזמנות בו זמנית ולחסוך להם זמן רב בעבודתם והספק יעיל.

האלגוריתם שעזר לי ביותר הוא diextara משום שעובד על גרף ממושקל וחיובי והמפה של הסופר בנויה רק על משקלים חיובים ולכן נעזרתי בו.

לאפליקציה קוראים EazyShop השם שלה זה המטרה שלה, פשוט לסייע למשתמש בקניה מהירה.

## תהליך המחקר

לאחר הבחירה בנושא חשבתי על איזה סופר לעשות את הפרויקט. הבחירה נבעה ממספר שיקולים ראשית העדפתי לעבוד על סופר שאני מכירה ויודעת היכן נמצאים הדברים כמו כן בסופר שיסיעה לי בהשגת מפות וכדומה ולכן נעזרתי בחברה שלי שעובדת בסופר הנוכחי ועזרה לי להשיג את המפה.

לאחר מכן קראתי חומרים על אנשים שפיתחו דברים דומים וכך בניתי תבנית לאיך שאני רוצה שהפרויקט יעבוד ומה הוא יכיל.

חיפשתי אלגוריתמים על חיפוש מסלול מקוצר וקראתי על הסוגים וההבדלים ביניהם והבנתי שאני צריכה אלגוריתם שיעבוד רק על משקל חיובי כמו כן בתחילה הגדרתי את הבעיה כבעיית הסוכן הנוסע אך הסיבוכיות זמן ריצה שלו היא אין סופית ולכן בחרתי ב **. Dijkstra**

## סקירת ספרות

לצורך הפרויקט נעזרתי באתרי מידע רבים כדי לקרוא חומר בנוסף נעזרתי באתרים של תיקוני שגיאות וכדומה.

* האפליקציה פותחה בשפת c# משום שהיא נוחה וגמישה לעבודה.
* את הצד לקוח עשיתי באנגולר משום שהוא נוח למשתמש ויעיל למתכנת כמו כן אנגולר זו שפה שימושית מאוד בשוק העבודה ורציתי להכיר אותה לעומק ואכן הפרויקט סייע לי רבות בכך.

# **מטרות ויעדים**

לצורך עבודה רצופה על הפרויקט הצבתי לעצמי מטרות ויעדים שאני רוצה להגיע אליהם לצד מטרות אישיות שרציתי לקדם:

המטרות:

* לחסוך לצרכנים זמן יקר
* לחסוך להם כסף בכך שייצמדו לרשימה ברורה ומתכוננת מראש
* לסייע למלקטים בעבודתם
* משיכת לקוחות לסופר על ידי שימוש בטכנולוגיה חדשנית ומתקדמת
* ליצור ממשק נוח וקל ולמשתמש.

היעדים:

* המערכת תשמור עבור המשתמש רשימות שהשתמש שהם בעבר
* המערכת תבנה מפה על פי רשימת הלקוח
* המערכת תאחד מספר רשימות לרשימה אחת ותסייע בכך רבות למלקים.

מטרת העל שלי הייתה להגיע לתוצאה סופית מוכנה וליצור אפליקציה שאכן יכולה לצאת לשיווק.

לצד מטרות לצורך קידום הפרויקט היו לי מספר מטרות אישיות כגון צבירת ידע וניסיון באנגולר ובc#

כמו כן התנסות עם טעויות שלא הכרתי קודם ולמידה עצמית.

# **אתגרים**

במהלך הפרויקט נתקלתי באתגרים רבים שסייעו לי רבות בלמידה על כיווני חשיבה שונים והתמודדות עם אתגרים שונים שלא הכרתי קודם. כגון: יצירת המפה בעצמה. לצורך יצירת המפה הייתי צריכה להליט מי הצמתים וכיצד אני מחשבת את הקשתות ולמדתי שצומת חייב להיות דבר שלא חוזר על עצמו בשום דבר אחר ולכן שללתי מספר אופציות והחלטתי שהצמתים יהיו העמודות של הסופר והקשתות יהוו את המרחק בין עמודה לעמודה. כמו כן ההצעת פרויקט שלי לא קיבלה אישור בהתחלה והייתי צריכה לחשוב על כיוון שונה ולכן הוספתי את המלקטים מה שהוסיף צבע ואתגר נוסף.

# **מדדי הצלחה**

מדדי ההצלחה שלי היו שברגע שהלקוח יכניס רשימה תיווצר לו מפה ברורה של המסלול שלו כמו כן שהמלקט יכניס מספר רשימות והמערכת תאחד לו אותם ולאחר מכן תיצור מפה מסודרת.

# **תיאור המצב הקיים**

כיום המצב הוא שהאפליקציה קיימת בשוק במגוון מקומות בחלקם שימושית יותר ובחלק פחות.

לאחר בירור אצל אנשים שונים הבנתי מה היתרונות של כל אפליקציה ומה החסרונות ולכן בפרויקט שלי יש סיוע גם למלקט שזה חידוש על פני האפליקציות האחרות שיש סיוע גם לעובד ולא רק ללקוח במיוחד שבשנים האחרונות מתפתח מאוד העניין של קניות אונליין ועבודת המלקטים תופסת נפח רב בשוק העבודה.

# **רקע תאורטי**

האלגוריתם המרכזי בפרוייקט הוא הדייקסטרה המשמש למציאת מסלול מקוצר ביותר ממקור יחיד עבור גרף מכוון וממושקל, עבור משקולות אי שליליים בלבד.

הבעיה העיקרית בפרויקט שלי הייתה לבחור מה יהווה את הצמתים וכיצד יחושבו הקשתות.

את קביעת הצמתים בחרתי לקבוע כנקודות מסוימות בכניסה לטור באמצע טור ובסוף טור כאשר המשקל הוא המרחק בין צומת לצומת. תיאור זה יתבצע על ידי כך שמספרתי את כל העמודות בסופר וכל טור הכיל מספר מסוים של עמודות, בנוסף לכל עמודה שמור מידע האם היא בתחילת הטור או בסוף הטור. לאחר שמירת מידע זה החלטתי להציב את הצמתים בכניסה באמצע ובסוף. כך הלקוח יוכל להגיע לטור המבוקש או לתחילתו במידה והמוצר בהתחלה או לסופו במידה והמוצר בסוף וכן הלאה. כעת האלגוריתם המתבקש הוא אלגוריתם שיחשב את המסלול הקצר בין מספר הסלולים האפשריים.

1. מה האלגוריתם (החלק הכי מורכב) העיקרי בפרויקט? (3 שורות)
2. תזכירי, מה הייתה הבעיה שהאלגוריתם פותר? (6 שורות)
3. איזה סוג של אלגוריתם הכי מתאים לפירוט הבעיה שתיארת?  
   אל תתנו שם של אלגוריתם אלא תסבירו איזה סוג של אלגוריתם צריך.

# **ניתוח חלופות מערכתי**

היו אפשרויות שונות ליישום רעיון זה למשל להפוך את הסופר למטריצת סמיכויות ועליה להפעיל את האלגוריתם או שימוש באלגוריתם "הסוכן הנוסע" אך אני בחרתי באלגוריתם הדייקסטרה גם בעקבות זמן הריצה הנמוך וגם משום התאמה למערכת.

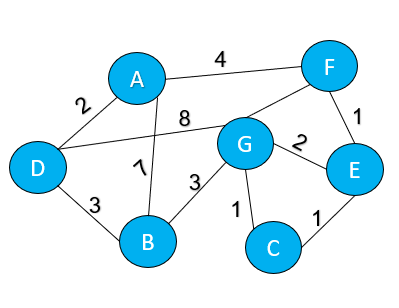
# **תיאור החלופה הנבחרת והנימוקים לבחירה**

1. שוב, מה המערכת שפתחת?

המערכת שלי זוהי מערכת שיוצרת מסלול קצר ביותר בסופר מתוך רשימת קניות של משתמש. בדאטה בייס שלי ישנה טבלה של צמתים שכנים Neighboring\_Nodes טבלה זו הינה טבלה בקשר של רבים לרבים ושני השדות בה הינם מפתחות ראשיים, תפקידה היא לשמור נתונים לכל צומת איזה צמתים שכנים יש לה. ונוצרה על מנת לממש את האלגוריתם שלי. וכיצד?

אלגוריתם הדייקסטרה עובד על מציאת מסלול קצר ביותר **ממקור יחיד** עבור גרף מכוון.

ונסביר: כשלקוח מכניס רשימת מוצרים יש לו הרבה צמתים והרבה דרכים, האלגוריתם לא עובד על הרבה צמתים והרבה דרכים, אלא ממקור יחיד להרבה צמתים ניתן לראות דוגמא באיור משמאל. על גרף כזה:



 הדייקסטרה אינו יכול לעבוד ולכן על מנת להגיע לגרף המבוקש כתבתי פונקציה שמחשבת מרחק בעזרת נוסחת מרחק -

פונקציה זו מטרתה היא לחשב את המרחק בין כל שתי נקודות בטבלת השכנים שהוזכרה לעיל.

לאחר שמצאנו מרחק קבוע בין כל צומת לצומת:

המרחק מצומת לצומת סמוכה הוא ידוע.   
מרחק בין צמתים רחוקות לא ידוע וקיימות אפשרויות רבות למציאתו.

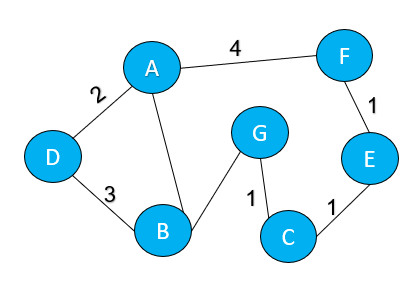
לכן:

נשתמש בדייקסטרה פעמים רבות כדי לחשב מרחק קצר ביותר בין צמתים רחוקות.

ורק אחר כך:

נתבסס על הקשתות שחישבנו כדי להפעיל דייקסטרה שמוצא את המסלול הקצר ביותר בין הצמתים הרצויות.

הגרף הרצוי הוא הגרף הזה:



על גרף כזה הדייקסטרה יודע לעבוד ולכן עשיתי פונקציה שבה יש לולאה מקוננת. הלולאה החיצונית עוברת על כל הצמתים שהתקבלו. צמתים אלה התקבלו מהרשימה של הלקוח בעזרת פונקציה נוספת שמקבלת רשימת מוצרים והופכת אותם לרשימת צמתים. והלולאה הפנימית עוברת על כל השכנים של כל צומת שהתקבל. כעת הדייקסטה מופעל. אלגוריתם זה מקבל צומת מקור וצומת יעד ויוצר לי גרף זהה לגרף שהוזכר לעיל בצורה כזו דייקסטרה על:

A->B והמרחק שחוזר הוא המשקל של הקשת.

דייקסטרה על A->D

דיקסטרה על A->F

לאחר שיתקבל הגרף יוצג למשתמש המסלול הקצר ביותר שעליו לעבור.

סיבוכיות זמן הריצה של אלגוריתם זה הינה O(n\*n)(E+V) זוהי סיבוכיות גדולה יחסית אך בהתחשב בכך שהאלגוריתם השני שעושה את הפעולה הזו הינו הסוכן הנוסע וסיבוכיות הזמן שלו היא אין עצרת O(n!) ולכן וודאי שהסיבוכיות של האלגוריתם שלי נמוכה בהרבה.

# **אפיון המערכת**

**סביבת פיתוח :**

חומרה: מעבד RAM 32GB i7 עמדת פיתוח: מחשב Lenovo מערכת ההפעלה: 10 Windows

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

כלי תוכנה לפיתוח המערכת: .Microsoft Visual Studio2019, vs code:

מסד נתונים:.SQL Server עמדת משתמש מינימאלי ת :

* + חומרה: מעבדRAM 4GB i5 .

o מערכת ההפעלה: Windows 7ומעלה.

* + חיבור לרש ת: נדרש .
  + תוכנת: chrome

## ניתוח דרישות המערכת

**דרישות בהן המערכת צריכה לעמוד:**

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

## **מודול המערכת**

* העלאת רשימת מוצרים של משתמש.

* קריאת הקובץ שהועלה וקריאת הנתונים מהדאטה בייס,
* הזנת נתונים נוספים לתלמידים ע"י המשתמש.
* קריאה למערכת לביצוע שיבוץ.
* מציאת השיבוץ האופטימלי ככל שניתן ע"י האלגוריתם.
* הזנת פרטי השיבוץ בדאטה ביי ס
* הצגת פרטי השיבוץ ללקוח.
* שליחת מייל ללקוח עם השיבוץ הסופי

## אפיון פונקציונאלי

(פירוט פונקציות עיקריות ותפקידן)

public static List<Route> CreateRoute(List<Nodes> ln) – הפונקציה מחשבת את אורך הקשתות בין צומת לצומת באופן כזה:

הפונקציה מקבלת רשימה של צמתים ועוברת בforeach על כל צומת בנפרד.

לכל צומת היא בודקת את כל שכניו ומחשבת מרחק ביניהם על ידי משוואת distance. את התוצאה היא שומרת בתוך רשימה של קשתות שתכיל את כל המרחקים מצומת נתונה לצמתים השכנים שלו.

public static List<Nodes> ConvertProductToNodes(List<Products> lp)

הפונקציה ממירה רשימת מוצרים לרשימת צמתים באופן הבא:

הפונקציה מקבלת רשימת מוצרים ובודקת לכל מוצר מה העמודה שלו ומתוך טבלת עמודות האם העמודה עומדת בתחילה או בסוף טור. על פי המידע הזה אני פונה לתחילת הטור ומשם לצומת שמציינת את תחילת הטור הנוכחי. כך לכל המוצרים עד שנוצרת לי רשימת צמתים על פי המוצרים שהתקבלו.

## ביצועים עיקריים

המשמש מכניס את פרטיו ולאחר מכן יוצר רשימת קניות

למשתמש מוצגת מפה עם המסלול הקצר ביותר שעליו לעבור

המלקט מכניס מספר רשימות והמערכת מאחדת לו אותן לרשימה אחת ויוצרת לו מפה גם כן.

## אילוצים

* המערכת יכולה לפעול רק על סופר אחד.(אושר עד שמגר)
* המערכת חייבת עדכון במידה ומשנים מיקום של מוצרים בסופר.

# **תיאור הארכיטקטורה**

## הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top

צד השרת - server side פותח במודל 3 השכבות ומתחלק ל-4 פרויקטים

החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין הנתונים עצמם. הפרדה זו מאפשרת לבצע שינויים בכל אחת מהשכבות בלי תלות ובלי זעזועים בשכבות האחרות.

API – שכבת ה Controller – חיבור בין צד השרת והלקוח.

BL – הלוגיקה של המערכת.

DAL – מכיל את הפונקציונאליות הנדרשת לכל התקשורת עם הData Base .

Models – מכילה מחלקות המתארות את הנתונים ובמבנה זה מעבירים את הנתונים בין השכבות.

מטרת שכבה זו היא למנוע תלות של שכבת הBL במבנה בסיס הנתונים. שכבת הBL מכילה פונקציות המרה מטיפוס הנתונים של בסיס הנתונים לטיפוס הנתונים של שכבת הModels ולהיפך, וכך מיוצגים הנתונים בכל הפרויקט.

## תיאור הרכיבים בפתרון

הפרויקט מחולק ל-2 חלקים:

* + - צד שרת - הנכתב בשפת #C ובטכנולוגיית WebApi.
    - צד לקוח - נכתב בשפת Angular ובטכנולוגיית Html, TypeScript.

בחרתי לכתוב צד לקוח ב - אנגולר שהינה שפה מתקדמת ועדכנית בעלת מאפייניAngular8 חדשניים ופונקציונאלית ביותר.

אנגולר הינה סביבת עבודה שפותחה על ידי גוגל. מאפשרת לפתח אפליקציותFramework אינטרנט בקלות ומהירות. במקור היא באה לתת מענה לבניי תApplications Page Single בצורה מושלמת ומהירה. מהיתרונות הבולטים והעיקריים של אנגולר אפשר למנות: חיסכון במשאבים, מהירות ביצוע, קוד קצר יותר, רוב העבודה מתבצעת בצד הלקוח ופחות בשרת ויכולת התמודדות טובה )סינון מהיר ופשוט לביצוע( של תוכן המתקבל מהשרת לפי מספר רב של פרמטרים.

צד שרת בחרתי לכתוב בC# .C# היא שפת תכנות עילית מרוב ת-פרדיגמות, מונחית עצמים בעיקרה המשלבת רעיונו ת כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיות, פונקציונאליות פרוצדוראליות וגנריות .

C# היא שפה מעניינת, נוחה ומלאה פונקציונאליות למתכנת. שימוש בשפה זו נפוץ כיום , וכתוצאה מכך, ניתן היה למצוא בה קודים שונים שנדרשו לפיתוח.

בנוסף ,בחרתי להשתמש ב - EntityFramework טכנולוגית עבודה מתקדמת של מיקרוסופט.

הEntityFramework מאפשר לטעון את הנתונים מהDB -ולעשות להם השמה בצורה ישירה ואוטומטית לתוך אובייקטים בקוד הממפים את מאגר הנתונים בצורה מידית.

.Sql Server שנכתב בשפת DataBaseקורא נתונים מ ה EntityFrameworkה

למסד הנתונים של ה-SQL Server יש כלים נרחבים לגיבוי כל המידע של המערכת ,כולל מערכת ההפעלה, חשבונות המשתמשים והרשאותיהם, הגדרות ההתקנים, תוכניות וכן של שאר הרכיבים המסופקים עם השרת ואובייקטי המשתמש.

***דוגמא לזרימת מידע במערכת***

שליפת כל המשתמשים

ברצוננו לקבל את כל המשתמשים הDB ולכן יתבצעו השלבים הנ"ל:

* המנהל יחפוץ לראות את כל המשתמשים, הוא ילחץ על כפתור מסוים בתצוגה) html( ובקשתו תפנה לTypeScript.
* . servicesאשר תפנה ל GetStudentByiD תתבצע קריאה לפונקציה- script Type . לשרת URL תתבצע בקשת services -ב o
* השרת מקבל את הבקשה ומנווט ל Controller שנמצא בAPI.
* הController יזמן את הפונקציה GetStudentByiD שנמצאת studentBL. o BL מעוניין לקבל נתונים מהDB ולכן הוא פונה לDAL- דרך הframework Entity o ה-DAL שואב את הנתונים הרצויים ממסד הנתונים וכעת מתבצע שלב החזרה.
* ה DAL מחזיר את רשימת המשתמשים לשכבת הBL.
* . BLמה controllerמחזירה את הנתונים ל **GetUser**הפונקציה
* . serviceהנתונים מוחזרים ל controller -מה
* מהservice חוזרת הרשימה לtypeScript. o הרשימה מוצגת בHTML.

*איור:*

DB

Sql Server

Dal

Entity Framework

Bl

L

ayer

Controllers

Web Api

Client

Models

1. מסד הנתונים הבנוי מטבלאות וקשרי גומלין ביניהם.
2. שכבת הגישה לנתונים באמצעות Entity Framework.
3. שכבת הישויות.
4. שכבת ה - BL בה כתובים האלגוריתמים.
5. פרוטוקול התקשורת בי ן צד הלקוח וצד השרת. Web Api
6. צד לקוח: angular, TypeScript

## תיאור פרוטוקולי התקשורת

http – הוא [פרוטוקול תקשורת](https://www.hamichlol.org.il/%D7%A4%D7%A8%D7%95%D7%98%D7%95%D7%A7%D7%95%D7%9C_%D7%AA%D7%A7%D7%A9%D7%95%D7%A8%D7%AA) שנועד להעברת דפי [HTML](https://www.hamichlol.org.il/HTML) ואובייקטים ברשת ה[אינטרנט](https://www.hamichlol.org.il/%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98).

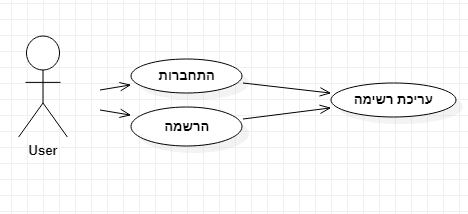
## שרת – לקוח

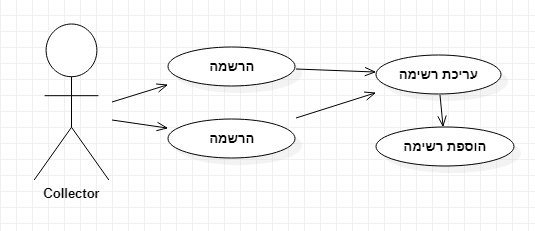
צד השרת נכת ב בטכנולוגית WebApi ובשפת c#.

צד הלקוח נכתב בשפת Angular בטכנולוגיית Html, typescript, CSS

# **ניתוח ותרשים use case של המערכת המוצעת**

תרשים כפי הנלמד בשנה שעברה. (איש לכל סוג משתמש וחץ עם הפעולות שהוא מבצע)





## רשימת use case

רשימת הפעולות המבוצעות ע''י המשתמשים:  
המשתמש מתחבר ע''י הקלדת שם משתמש וסיסמא, המשתמש מכניס מוצרים לרשימה. המשתמש בוחר האם להוסיף רשימה נוספת.(במידה והוא מלקט)

## תיאור ה-use case העיקריים של המערכת

Identifier , Name, Description , Actors, Frequency , pre-condition ,   
תחפשו כיצד מנסחים תיאור של use case.

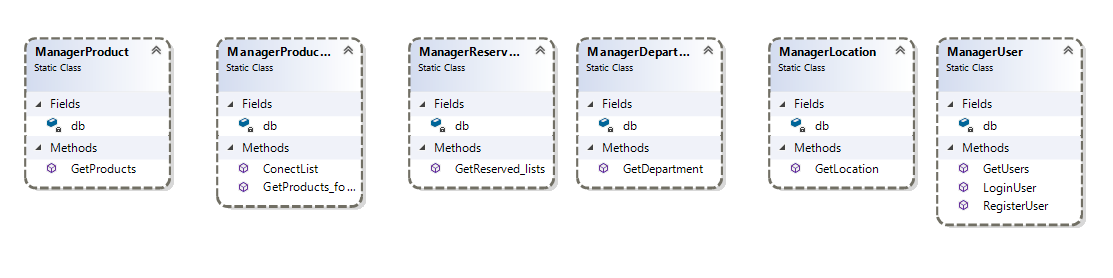
## מבני נתונים בהם משתמשים בפרויקט

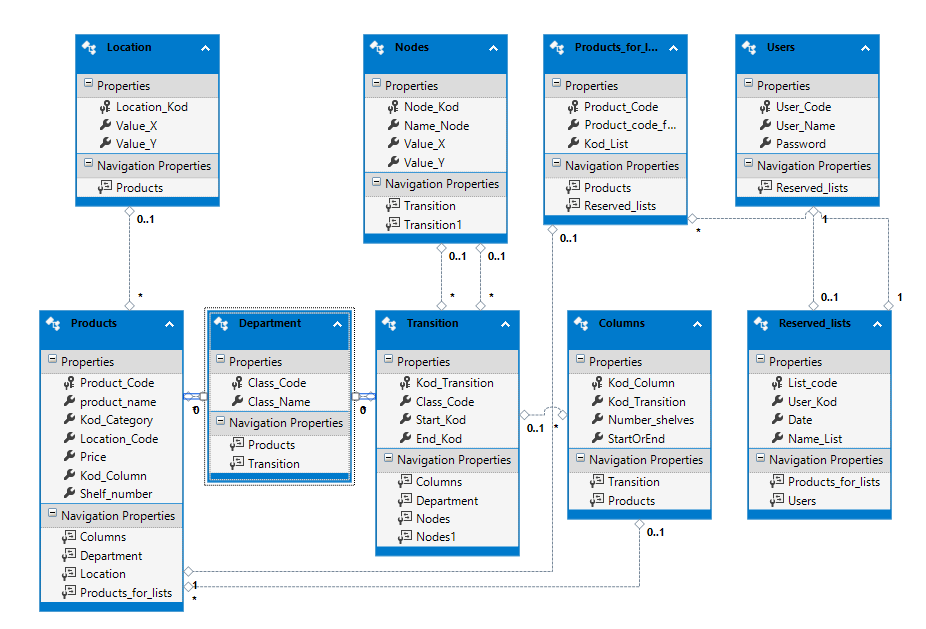
(מילון / רשימה / תור / עץ / מחסנית / ערימה והסבר איפה ולמה השתמשת במבנה זה)

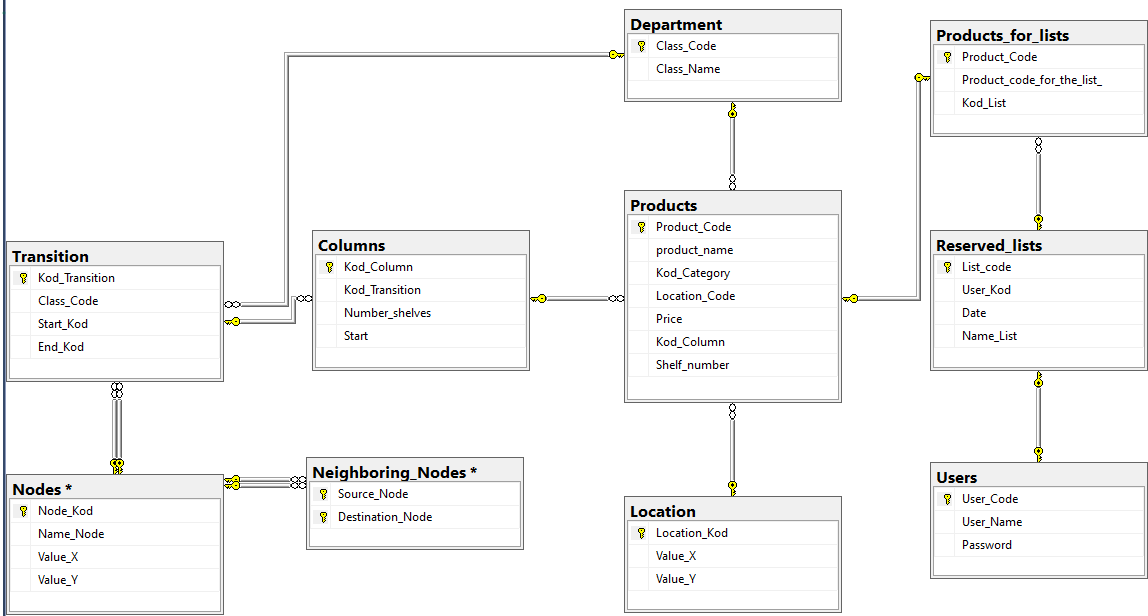
בפרויקט שלי היה שימוש במבני נתונים של רשימה השתמשתי במבנה נתונים זה במספר מקומות לדוגמא בפונקציה שהופכת בין מוצרים לצמתים פונקציה זו מקבלת רשימת מוצרים.

## תרשים מחלקות

**צילום של פירוט המחלקות:**

**שכבת ה-BL:**

**שכבת ה-DAL:**



## תיאור המחלקות

לכל שכבה בפרויקט– הסבר מילולי על השכבה. רשימת כל שמות המחלקות ולכל מחלקה הסבר מילולי. (לפי הסדר – DAL, BL, DTO, WEB-API , צד הלקוח הוא השכבה הגבוהה ביותר)

**שכבת API**-

ממשקי API הם מנגנונים המאפשרים לשני רכיבי תוכנה לתקשר זה עם זה באמצעות סט של הגדרות ופרוטוקולים.

**שכבת הלוגיקה העסקית (BL - Business Logic) –**

השכבה שאמונה על הלוגיקה של המערכת, עוסקת בעיבוד המידע, בחישובים שונים ושליחתו לשכבת התצוגה.  
בשכבה זו נממש את הפונקציונאליות של המערכת.

**שכבת ה DAL–**

שכבה זו מורכבת ממקור נתונים . (Data Source)מקור הנתונים שלי היה sql**.**

צד הלקוח היא השכבה הגבוהה ביותר משום שהיא אחראית על התצוגה ללקוח ויישום של כל האלגוריתמיקה בצורה נכונה ונוחה למשתמש.

# **תיאור התוכנה**

* + **סביבת עבודה:**

Visual Studio Codeו Visual Studio

* + **שפות תכנות:**

צד השרת נכתב בטכנולוגית WebApi ובשפת c#.

**.** angular בטכנולוגית -Html, css ,typescript צד הלקוח נכתב בשפות

# **אלגוריתמים מרכזיים**

כאן תפרטי את הפעולות העיקריות בפרויקט.

# **קוד האלגוריתם**

לכאן תעתיקי את הפונקציות העיקריות בפרויקט

# **תיאור מסד הנתונים**

## פירוט הטבלאות ב- Data Base

**טבלת עמודות** – מטרתה היא למספר את העמודות בכל טור.

Kod\_Column- זהו מספר אוטומטי שימספר את העמודות בכל הסופר.

Kod\_Transition - הינו מפתח זר מטבלת טורים על מנת שנדע באיזה טור נמצאת כל עמודה.

Number\_shelves- ממספר כמה מדפים יש בכל עמודה.

Start- זהו שדה שמכיל מספרים או 1 או 2 על פי מיקום העמודה. במידה והעמודה ממקוממת בתחילת התור שדה זה יקבל את הסיפרה 1. ובמידה והעמודה ממוקמת בסוף הטור שדה זה יקבל את הערך 2.

|  |  |  |  |
| --- | --- | --- | --- |
| מפתח | שם שדה | תיאור | טיפוס |
| Kod\_Column | Kod\_Column | קוד עמודה | int |
|  | Kod\_Transition | קוד טור | int |
|  | Number\_shelves | מספר מדפים בעמודה | int |
|  | Start | התחלה או סוף |  |

**טבלת קטגוריות**- בטבלה זו יוגדרו כל הקטגוריות שיש בסופר ולפיהם יחולקו המוצרים.

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| int | קוד קטגוריה | Class\_Code | Class\_Code |
| nchar(10) | שם קטגוריה | Class\_Name |  |

Class\_Code- זהו מספר אוטומטי שימספר את הקטגוריות.

Class\_Name- שדה זה מכיל את שם הקטגוריה.

**טבלת צמתים שכנים**- טבלה זו מגדירה לי איזה צמתים שכנים לצומת המקור.

Source\_Node- זהו צומת המקור שמהווה מפתח ראשי ומקושר לטבלת צמתים.

Destination\_Node - שדה זה יציג לי מה הצומת הקרובה לצומת המקור. מהווה גם כן מפתח ראשי ומקושר לטבלת צמתים.

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| int | קוד מקור | Source\_Node | Source\_Node |
| int | קוד יעד | Destination\_Node | Destination\_Node |

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| int | קוד צומת | Node\_Kod | Node\_Kod |
| nchar(10) | שם צומת | Name\_Node |  |
| int | ערך X | Value\_X |  |
| int | ערך Y | Value\_Y |  |

**טבלת צמתים**-טבלה זאת ממספרת לי את הצמתים של הסופר.

Node\_Kod- בשדה זה ישנו את קוד הצומת (מספור אוטומטי).

Name\_Node- בשדה זה יש את שם הצומת. על מנת לכוון את המשתמש בצורה מילולית.

Value\_X-לאחר חלוקה של מפת הסופר לערכים של x וy כאן יוכנס ערך הx.

Value\_Y- כאן יוכנס ערך הy.

**טבלת מוצרים**-בטבלה זו מאוחסנים כל הפריטים בסופר.

Product\_Code - קוד מוצר (מספור אוטומטי).

product\_name – שם מוצר.

Kod\_Category – קוד קטגוריה. זהו מפתח זר מטבלת קטגוריות. לכל מוצר משויכת הקטגוריה.

Price- מחיר של כל מוצר.

Shelf\_number- מספר מדף. באיזה מדף נמצא המוצר.

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| int | קוד מוצר | Product\_Code | Product\_Code |
| nvarchar(50) | שם מוצר | product\_name |  |
| int | קוד קטגוריה | Kod\_Category |  |
| float | מחיר מוצר | Price |  |
| int | קוד עמודה | Kod\_Column |  |
| int | מספר מדף | Shelf\_number |  |

**טבלת טורים**- בטבלה זו יצויינו הטורים שהסופר מחולק אליהם.

Kod\_Transition- זהו מספר הטור(מספור אוטומטי).

Class\_Code- זהו מפתח זר מטבלת הקטגוריות.

Start\_Kod- קוד זה משויך לטבלת צמתים כאשר שדה זה מכיל את צומת ההתחלה באותו הטור. הכוונה כניסה לטור הנוכחי.

End\_Kod- קוד זה משויך גם כן לטבלת צמתים כאשר שדה זה מכיל את צומת הסיום באותו טור הכוונה יציאה מהטור הנוכחי.

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| Int | קוד טור | Kod\_Transition | Kod\_Transition |
| Int | קוד קטגוריה | Class\_Code |  |
| Int | קוד התחלה | Start\_Kod |  |
| int | קוד סיום | End\_Kod |  |

**טבלת משתמשים**- בטבלה זו ישמרו נתוני המשתמשים במערכת.

User\_Code- זהו מספר אוטומטי על מנת למספר את המשתמשים.

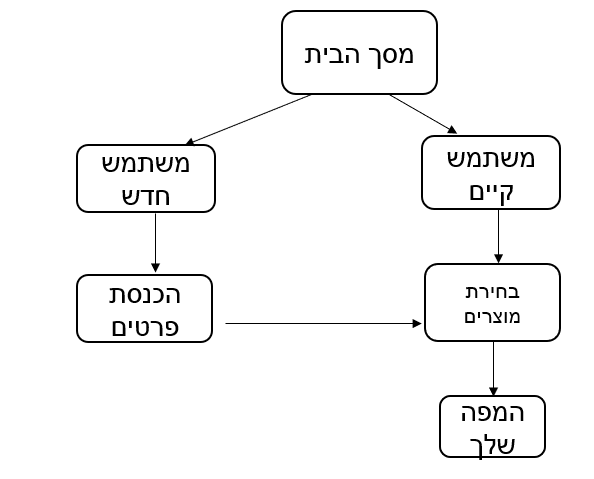
User\_Name- זה שם שהמשתמש יבחר בו. כך הוא יוכל להיכנס למערכת בקלות.

Password- סיסמא על מנת לאבטח את הכניסה.

|  |  |  |  |
| --- | --- | --- | --- |
| טיפוס | תיאור | שם שדה | מפתח |
| int | קוד משתמש | User\_Code | User\_Code |
| varchar(50) | שם משתמש | User\_Name |  |
| varchar(50) | סיסמא | Password |  |

# **מדריך למשתמש**

## תיאור המסכים

תרשים זרימה של (כל)המסכים:  


## מדריך למשתמש

בכניסה לאתר יוצג למשתמש מסך הבית. במסך זה יהיה על המשתמש או להתחבר במידה וזו לא כניסה ראשונה שלו או להירשם מחדש. הרישום למערכת הינו שלב פשוט ביותר על המשתמש להזין בסך הכול שם משתמש וסיסמא. לאחר ההתחברות למערכת המשתמש יועבר למסך של בחירת מוצרים, במסך זה המשתמש יצור לעצמו רשימה שתכיל את כל המוצרים שהוא מעוניין לרכוש. מיד בסיום הרשימה ולחיצה על אישור יוצג למשתמש מפה של כל הנקודות שעליו לבקר בהם במהלך קנייתו.

במידה והמשתמש הינו מלקט תהיה קיימת עבורו אפשרות של הכנסת מספר רשימות ולאחר לחיצה על אישור תוצג לו מפה עם כל הנקודות שעליו לבקר בהם.

## צילומי מסכים

לכל מסך: שם המסך ותמונה

# **בדיקות והערכה**

לאחר הרצת האלגוריתם נבחנו כל האילוצים שדרושים כדי להביא למפה מדויקת, למסלול קצר ביותר ובזמן ריצה הנמוך ביותר כאשר הופיעו טעויות האלגוריתם נבדק שוב ושוב עד שתוקנו כל הבעיות.

כמו כן ניסתי לחשוב על כל מיקרי הקצה האפשריים בפרוייקט שלי בתקווה שאלו היו רובם ככולם.

לאחר הרצת האלגוריתם מספר פעמים עם נתונים שונים הוא הגיע לקירוב האפשרי ביותר בכלים העומדים לרשותי ובפרק הזמן הנתון.

# **ניתוח יעילות**

(תסבירי מה זה יעילות למה היא הייתה חשובה לך וכמה הסיבוכיות של האלגוריתם שלך)

# **אבטחת מידע**

באפליקציה המשתמש חייב להיות רשום במערכת על מנת להיכנס ולכן משתמש שלא קיים יהיה חייב להכניס פרטים אישיים. כמו כן ישנה פונקציה שבודקת האם אכן המשתמש שמנסה להתחבר קיים במערכת ומאמתת את פרטיו ורק לאחר מכן תתאפשר לו כניסה בטוחה.

# **מסקנות**

הפרויקט סייע לי מאוד בהבנת הרבה דברים ושימוש בשפות וסיפריות שלא הגעתי אליהם קודם. צברתי ידע רב באנגולר שהיה לי מאוד חסר כמו כן תיקון שגיאות באופן עצמאי, למידה של חומרים רבים והכרת מושגים נוספים. לאחר כל תיקון של שגיאה והתקדמות ניכרת הרגשתי סיפוק אדיר. החשוב ביותר הוא להאמין בעצמך ולא ליפול לייאוש בכל פעם של תקלה.

# **פיתוח עתידי**

הייתי שמחה להוסיף לפרויקט סיוע מילולי למשתמש. הכוונה היא שלמשתמש יהיו הוראות ברורות כיצד לפנות ולאן(כמו בwaze) לא הספקתי מספיק לחקור על נושא זה אבל ממש שכן קראתי צריך לדעת היכן המשתמש ממוקם בכל רגע ורגע בקנייתו. אז בעתיד אשמח לגשת גם לנקודה זו.

# **ביבליוגרפיה**

* Stack overflow
* Angular material
* GitHub
* Bootstrap
* w3schools