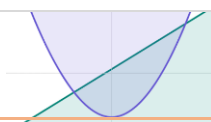


פרויקט גמר 2022

רינת אביטל | 212735427

## תוכן עניינים

1.	הצעת פרויקט	4
2.	מבוא / תקציר	6
2.1.	הרקע לפרויקט	6
2.2.	תהליך המחקר	8
2.3.	סקירת ספרות	9
3.	מטרות ויעדים	10
4.	אתגרים	11
5.	מדדי הצלחה	12
6.	תיאור המצב הקיים	12
7.	רקע תאורטי	13
8.	ניתוח חלופות מערכת	21
9.	תיאור החלופה הנבחרת והנימוקים לבחירה	22
10.	אפיון המערכת	23
10.1.	ניתוח דרישות המערכת	24
10.2.	מודול המערכת	24
10.3.	אפיון פונקציונאלי	25
10.4.	ביצועים עיקריים	26
10.5.	אילוצים	27
11.	תיאור הארכיטקטורה	28
11.1.	הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top	28
11.2.	תיאור הרכיבים בפתרון:	29
11.3.	ארכיטקטורת רשת (לא רלוונטי) <b>שגיאה! הסימניה אינה מוגדרת.</b>	31
11.4.	תיאור פרוטוקולי התקשורת	31
11.5.	שרת – לקוח	31
11.6.	תיאור הצפנות (לא רלוונטי) ..... <b>שגיאה! הסימניה אינה מוגדרת.</b>	32
12.	ניתוח ותרשים use case של המערכת המוצעת	32
12.1.	רשימת use case	32
12.2.	תיאור ה-use case העיקריים של המערכת	33
12.3.	מבני נתונים בהם משתמשים בפרויקט	36
12.4.	תרשים מחלקות	37
12.5.	תיאור המחלקות	38



13.	תיאור התוכנה .....	40
14.	אלגוריתמים מרכזיים .....	41
14.1.	חלק מהאלגוריתם... הפיכת התמונה לשחור לבן <b>שגיאה! הסימניה אינה מוגדרת.</b>	
14.2.	חלק אחר מהאלגוריתם... פירוק התמונה לאותיות .....	41
14.3.	החלק העיקרי באלגוריתם למידת מכונה – זיהוי האות .....	41
15.	קוד האלגוריתם .....	42
16.	תיאור מסד הנתונים .....	48
16.1.	פירוט הטבלאות ב- Data Base .....	48
17.	מדריך למשתמש .....	50
17.1.	תיאור המסכים .....	50
17.2.	מדריך למשתמש .....	<b>שגיאה! הסימניה אינה מוגדרת.</b>
17.3.	צילומי מסכים .....	54
18.	בדיקות והערכה .....	57
19.	ניתוח יעילות .....	58
20.	אבטחת מידע .....	59
21.	מסקנות .....	60
22.	פיתוח עתידי .....	61
23.	ביבליוגרפיה .....	62



**סמל מוסד:****שם מכללה:****הצעת פרויקט****שם הסטודנט:** רינת אביטל**ת.ז. הסטודנט:** 212735427**שם הפרויקט:** Visual function**תיאור הפרויקט:**

האפליקציה תאפשר כניסה למשתמש חדש וקיים כאשר ישנה אפשרות להכניס פונקציה והאפליקציה תשרטט בצורה ויזואלית ותחשב עבור הפונקציה את נקודות הקיצון, נגזרת הפונקציה, תחום חיוביות ושליליות... וכן תתאפשר למידה ממוקדת על נושאי הפונקציות. וכן הצגת היסטוריה על כל פעולותיו של המשתמש, במידה והמשתמש רשום.

**הגדרת הבעיה האלגוריתמית:** למידת חשבון דיפרנציאלי ואינטגרלי, נגזרות למשוואות ומימושם הן מבחינת קוד והן מבחינת הצגה ויזואלית. תרגום מחרוזת למשוואה. פתירת משוואות.

**רקע תאורטי בתחום הפרויקט:**

שרטוט המשוואה דבר מורכב. וגם אחרי מציאת גרף המשוואה הוא אינו מדויק וברור. אפליקציה זו תעזור לסטודנטים בשרטוט הגרף ומציאת נקודות מיוחדות ונגזרת וכן תסייע למורים ומרצים במהלך השיעור להסבר התרגילים ורמת הדיוק בנוסף עוזרת במיוחד ליצירת מבחנים שנדרש שרטוט מוכן ומדויק של שרטוט הגרף.

**תהליכים עיקריים בפרויקט:**

1. הכנסת משוואה לאפליקציה
2. המרת המשוואה ממחרוזת למשוואה הניתנת לפרוק
3. פתרון ניתוח משוואות
4. הצגה ויזואלית של המשוואה ונגזרת, נקודות קיצון...
5. שמירת התרשים עם כל פרטיו (במקרה של משתמש רשום)



## תיאור הטכנולוגיה:

### צד שרת:

שפת תכנות בצד השרת: C#

### צד לקוח:

שפת תכנות בצד לקוח: angular

מסד נתונים: SQL

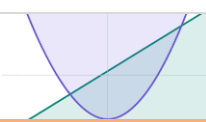
פרוטוקולי תקשורת: <http?>

### לוחות זמנים:

1. חקר המצב הקיים – ספטמבר
2. הגדרת הדרישות – ספטמבר
3. אפיון המערכת – אוקטובר
4. אפיון בסיס הנתונים – נובמבר
5. עיצוב המערכת – דצמבר
6. בנית התוכנה – ינואר, פברואר
7. בדיקות – מרץ
8. הכנת תיק פרויקט – אפריל
9. הטמעת המערכת – מאי
10. הגשת פרויקט סופי - מאי

### חתימת הרכז המגמה:

### אישור משרד החינוך:

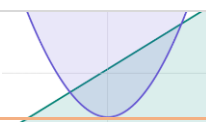


כשהתחלנו לחשוב על רעיון לפרויקט רציתי רעיון שיהיה לי מעניין לעשות לא רציתי לעשות פרויקט שלא יעניין אותי, העדפתי להשקיע בפרויקט שיכול להיות שיהיה קשה יותר אבל גם מהנה.

אחת ממטרות הפרויקט היא פיתוח מיומנויות חשיבה, להביא אותנו להתנסות עצמית. כשאני יושבת על פרויקט משלי שאני אמורה לעשות לבד מתחילתו ועד סופו אני אמורה ללמוד לבד חומרים שלא למדנו בכיתה, לנתח פרויקט ולהבין מה אני צריכה לבצע, להכיר קודים ואלגוריתמים שונים, לנסות ולהתנסות. וזה בעצם חלק ממטרות הפרויקט – להביא אותי בסופו של דבר למוכנות לעבודה כי הרי לימודים ועבודה זה עולמות שונים.

רציתי לנצל את הפרויקט כדי ללמוד עוד על תחומים שלא יצא לי להכיר ואכן זה מה שגם גרם לבחור דווקא את התחום של פתרון משוואות – התחום נשמע לי מעניין, מרתק ושונה. בלימודי בתיכון התעסקתי רבות בחקירת משוואות פונקציות ושרטוטם ויזואלית ואהבתי את המתמטיקה ומאוד רציתי להכיר את עולם הפונקציות והמתמטיקה גם דרך קוד התיכנות וראות כיצד אפשר לפתור ולשרטט את הפונקציות דרך המחשב. ואכן היה לי הרבה מה ללמוד. אני מרגישה שהפרויקט הרחיב לי אופקים התחום הצריך הרבה למידה עצמית, התנסות וכישלון וזה הביא אותי לרמה הרבה יותר גבוהה.

לאחר שהחלטתי על נושא לפרויקט וחקרתי אותו קצת הבנתי עד כמה הוא חשוב ונצרך. פתירת משוואות פונקציה היא משהוא שנצרך ביותר לסטודנטים הלומדים חדו"א חשבון דיפרנציאלי ואינטגרלי, שעד היום לא היה בטוח בתשובותיו ובשרטוט הפונקציה הנתונה לו ועתה תהיה לו אפשרות לבדוק את עצמו ותרגל ביתר קלות ונוחות מרבית באתר ששרטט את הפונקציה ומציג נקודות מיוחדות נוסף על כך הסטודנט יוכל לקבל את נגזרת הפונקציה שלו. וכן שרטוט המשוואה דבר מורכב. וגם אחרי מציאת גרף המשוואה הוא אינו מדויק וברור. אפליקציה זו תעזור לסטודנטים בשרטוט הגרף ומציאת נקודות מיוחדות ונגזרת וכן תסייע למורים ומרצים במהלך השיעור להסבר התרגילים ורמת הדיוק בנוסף עוזרת במיוחד ליצירת מבחנים שנדרש שרטוט מוכן ומדויק של שרטוט הגרף.



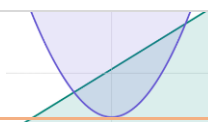
החלטתי שאני מאוד רוצה לקחת דווקא את הנושא הזה כי אני רואה את חשיבותו הרבה ונחיצותו.

הדבר שהכי חיזק בי את ההחלטה לבחור ברעיון של פתירת משוואות ושרטוטן כרעיון לפרויקט גמר הוא מכיוון שגם אני הייתי סטודנטית שלמדה על משוואות פונקציות ולא היה לי אפשרות לבדוק את עצמי, האם פתרתי ושרטטתי נכון והיה מאוד מקל עלי אם היה לי אתר שבו הייתי יכולה לבדוק את עצמי וללמוד דברים חדשים אודות הפונקציות המעניינות.

האתר מתאים גם לכל הקהל שמתעניין במתמטיקה שנהנה לראות איך פונקציות נראות ואת הנגזרת.

השם שבחרתי לפרויקט הינו "visual function" - משמעותה פונקציה ויזואלית, שרטוט כל פונקציה שמזינים למערכת.

לסיכום אני מאוד מקווה שהפרויקט יעיל ויעזור לשיפור נוחות הסטודנטים, אני חולמת שסטודנטים באמת יוכלו להשתמש באתר והוא באמת יהיה יעיל ושימושי. בנוסף הקוד יהיה קוד פתוח ואנשים ומרצים רבים יוכלו להיעזר בקוד ולהשתמש בו לצרכיהם הפרטיים.

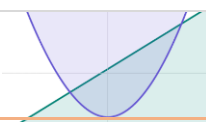


כשהתחלתי לפתח את הפרויקט דבר ראשון הלכתי לחפש חומרים קראתי הרבה מאמרים בנושאים שקשורים לפרויקט. בדקתי אלגוריתמים שקשורים בנושא פתירת משוואות אך לא מצאתי אלגוריתם העונה לדרישות שלי - פירוק משוואה לאיברים. לכן החלטתי לכתוב את האלגוריתם בעצמי - פירוק משוואה לאיברים.

בנוסף חקרתי רבות אודות שרטוט המשוואה אחרי הפרוק. חיפשתי ספריה באנגלר שתוכל לסייע לי בשרטוט המשוואה.

בהמשך שראיתי שאני יודעת הרבה על הנושא התחלתי לעבוד על הפרויקט בפועל וליישם את כל מה שקראתי ולמדתי.

את החומר הרב שקראתי מצאתי בגוגל באתרי למידה ומידע שונים כמו: Internet Israel, Stack Overflow וכד'. כמו כן ראיתי סרטונים רבים בנושאים של פתירת משוואות. לאחר החומר הרב שקראתי החלטתי להשתמש בספריה באנגלר echarts.





האתרים בהם השתמשתי לביצוע הפרויקט הם:

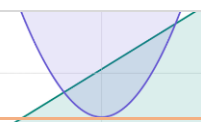
חקר הפרויקט –

### סקירת ספרות

- ויקיפדיה - [/https://www.hamichlol.org.il](https://www.hamichlol.org.il)
- angular - [/https://angular.io](https://angular.io)

אלגוריתם –

- [/https://echarts.apache.org](https://echarts.apache.org) - echarts.apache
- [/https://hodgef.com](https://hodgef.com)
- [/https://stackoverflow.com](https://stackoverflow.com) - Stack OverFlow
- [/https://github.com](https://github.com) - GitHub
- [/https://www.m-math.co.il](https://www.m-math.co.il)
- [/https://getbootstrap.com](https://getbootstrap.com) -Bootstrap
- [/https://www.w3schools.com](https://www.w3schools.com) -W3School



סטודנטים ומרצים רבים בתחום המתמטיקה נצרכים לחקור, לפתור ולשרטט פונקציות וכו' דבר זה דורש זמן, חשיבה וידע. בנוסף ישנם סיכויים שהפתרון לא יהיה נכון באופן מלא וכן שרטוט הפונקציה לא יהיה מושלם ומדויק. **וטרת ויעדים**

חשבתי לעזור ולקדם את תחום חקירת הפונקציות ולמטרה זו נועדה "visual function" - לעזור לסטודנטים ומרצים שרוצים לבדוק את נכונות פתרונם, לשרטוט מדויק של הפונקציה, לבדיקת משוואת הנגזרת ובדיקת נקודות מיוחדות.

בבניית מערכת ובמיוחד מערכת כזו שמטרתה פתירת משוואות ושרטוטם - אחד הדברים החשובים הוא להקפיד על מערכת ברורה ונוחה למשתמש. לרוב האנשים אין את הזמן והכוח לנסות להבין איך המערכת עובדת וכיצד להשתמש בה ולכן על המתכנת לבנות את המערכת בצורה שיהיה למשתמש קל ונוח להתעסק בה כמובן שחשוב שהמערכת תעבוד בצורה יעילה ונכונה שהרי היא נועדה לפתירת משוואות ושרטוטם.

לשם הקמת המערכת וכדי שהיא תתפקד בצורה המיטבית הצבתי לעצמי כמה מטרות ובשביל כך כמה יעדים:

#### **מטרות המערכת:**

1. אפשרות לפתירת משוואת בצורה מהירה קלה ויעילה.
2. נוחות וידידותיות למשתמש.
3. המרת המשוואה ממחרזת למשוואה הניתנת לפרוק
4. הצגה ויזואלית של המשוואה ונגזרת, נקודות קיצון...
5. לאפשר שמירת היסטוריית משתמש עם פרטי פתרון משוואותיו

#### **יעדי המערכת:**

1. בניית מערכת חכמה לפתרון משוואות ושרטוטם.
2. הוספת אפשרות ליצירת חשבון אישי ושמירת היסטוריית פתירת משוואותיו.
3. יצירת המערכת כאתר המפותחת בצורה נוחה וקלה למשתמש.
4. הצגה ויזואלית של המשוואה ונגזרת, נקודות קיצון...

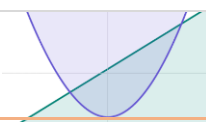


כשניגשתי להתחיל את הפרויקט ידעתי שהוא לא הולך להיות קל אבל לקחתי את זה בחשבון. ידעתי שאם אני רוצה להצליח אני חייבת להתאמץ ולהשקיע אפילו שיהיה קשה.

### אתגרים

בתחילת תכנון הפרויקט ושלבי העבודה לא ידעתי בכלל מאיפה להתחיל, הפרויקט היה נראה לי אתגר ענק ובלתי אפשרי, היו המון נושאים שלא הכרתי ומעולם התעסקתי בהם. עכשיו לאחר בניית הפרויקט אני מבינה שהכול אפשרי וגם דברים שנראים בלתי אפשריים בסוף עבירים.

לצורך פתרון המשוואות חיפשתי אלגוריתם שיעזור לי בפתירת משוואות אך לא מצאתי אלגוריתם שיכול לעזור לי, לכן החלטתי לכתוב בעצמי את האלגוריתם שיתאים בדיוק לצרכי פתירת המשוואות. כתיבת האלגוריתם בעצמי דרשה ממני חקר עמוק ויסודי בכדי שהגיע לתוצאות טובות וביתר קלות וכן חיפשתי חומרי עזר רבים אשר יוכלו לסייע לי במהלך פיתוח האלגוריתם החדש של פרוק המשוואה לגורמים – איברים.



האפליקציה אכן הצליחה והוכיחה את נכונותה כאשר משתמש מכניס פונקציה יש הצלחה של 100% בשרטוט גרף הפונקציה ושרטוט גרף הנגזרת וכן נקודות מיוחדות.

### מדדי הצלחה

כיום סטודנטים רבים ומרצים עוסקים רבות בתחום המתמטיקה. סטודנטים ומרצים או סתם חובבי ידע פותרים משוואות רבות מוצאים נקודות מיוחדות למשוואת הפונקציה אותם הם חוקרים וכן לבסוף אף משרטטים ויזואלית את הפונקציה. תהליך זה דורש ריכוז וידע בתחום ונוסחאות. שרטוט המשוואה דבר מורכב לא תמיד מגיעים לתוצאות ושרטוט כמו שציפינו וגם אחרי מציאת שרטוט גרף המשוואה הוא אינו מדויק וברור.

אתר Visual Function יוכל לסייע ולעזור לסטודנטים בשרטוט הגרף ומציאת נקודות מיוחדות ונגזרת וכן תסייע למורים ומרצים במהלך השיעור להסבר התרגילים ורמת הדיוק בנוסף עוזרת במיוחד ליצירת מבחנים שנדרש שרטוט מוכן ומדויק של שרטוט הגרף. וכן יצור חוויה נעימה וברורה הרבה יותר למשתמש וייתן לו חשק ורצון להמשיך ללמוד ולתרגל.

אני מאמינה שסטודנטים רבים ייהנו מחוויית הלימוד ויגיעו לתוצאות איכותיות ומדויקות.



תיאור המשימה ודרכי היישום:**רקע תאורטי**

בשלב תכנון המערכת ובניית מסדי הנתונים התבקשתי לבנות זאת בצורה הנכונה, המדויקת והממוקדת ביותר. לכן חיפשתי את האלגוריתם היעיל ביותר לניווט בשטח מקורה, אלגוריתמים ושיטות שונות למיפוי ולשמירת השטח.

**המשימה:** ייצוג המשוואה באופן שנוכל לבצע בה חישובים מתמטיים ופתירתה והצגתה באופן ויזואלי.

חקירת נושא פתירת משוואות פונקציונליות ושרטוטם הביאה אותי למסקנה שהדרך הטובה והיעילה ביותר היא ייצוג הפונקציה ע"י מבנה נתונים של רשימת פרמטרים לפונקציה.

חיפשתי סוגים שונים של אלגוריתמים שקשורים במתמטיקה ופתרון משוואות פונקציונליות אך לא מצאתי משהו שתואם את הבעיה שלי ולכן החלטתי ליצור בעצמי את האלגוריתם שמתאים כדי שאוכל לייצג את משוואת הפונקציה בצורה קלה ונכונה וכך אוכל ביתר קלות לגשת לכל אברי הפונקציה ולחקור אותה ולקבל תוצאות מדויקות.

האלגוריתם המרכזי בפרויקט הוא פירוק המשוואה לגורמים -איברים. הווה אומר לקחת מחרוזת של משוואת פונקציה פשוטה ולפרק אותה בצורה שתהיה ניתנת לייצוג נוח ולפתירה של הפונקציה. מכיוון שאין אפשרות לטפל במחרוזת פשוטה כדי לפתור את המשוואה חשבתי על רעיון שבו תיוצג המשוואה ע"י אובייקט שאליו נכניס את פרוק המשוואה ע"י האלגוריתם העיקרי שהוא פרוק המשוואה. הכנת האובייקט של המשוואה הוא רק הקדמה והכנה לאלגוריתם שיגיע בו הפרוק נעשה בפועל והאלגוריתם אחראי להכניס כל תו ותו במחרוזת למקומו המתאים באובייקט שיצרתי (יפורט בהמשך).

לאחר הפרוק והאובייקט המוכן לשימוש ניגש לפתור את המשוואה ולמציאת נקודות מיוחדות, נקודות אפס, נקודות קיצון, חישוב נגזרת הפונקציה והצגה ויזואלית של משוואת הפונקציה שהוכנסה למערכת.

ערכתי עבודת חקר וקראנו חומרים רבים בנושא פתירת משוואות, התדיינתי רבות באשר לדרך המימוש היעילה, התייעצתי עם מומחים היאך לחשב את פתרון המשוואות בדרך הטובה ביותר כדי לשמור על יעילות, דיוק ומהירות.



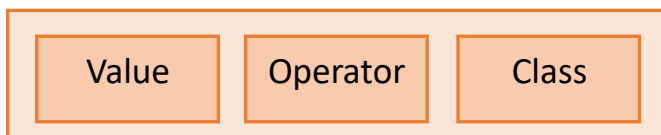
להלן מוצגות חלק מבעיות נוספות שהתעוררו בעת תכנון מבנה המערכת באופן זה, וכן הצעות שעלו ופתרונות שנקטנו בפועל.

### בעיות ופתרונן:

א. שמירת מחרוזת של גרף הפונקציה לכדי מבני נתונים שישמר בתוכו נתוני הפונקציה והפיכתה לשימושית ופתירה.

1. לשם כך צריך ליצור אובייקט "פרמטר" שבו מאוכסן כל איבר בפונקציה:

### Parameter

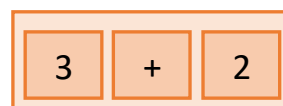


Value – ערך האיבר

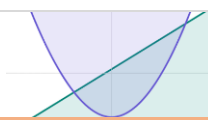
Operator – אופרטור (+/-)

Class – מעלה של האיבר (דרגה)

Parameter

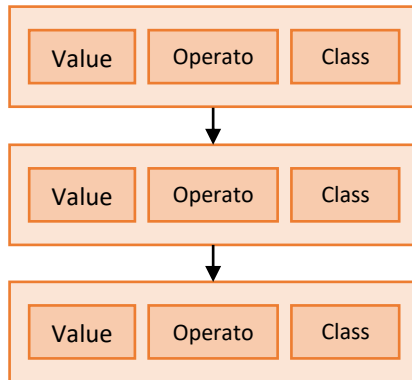


לדוגמא:  $3x^2$



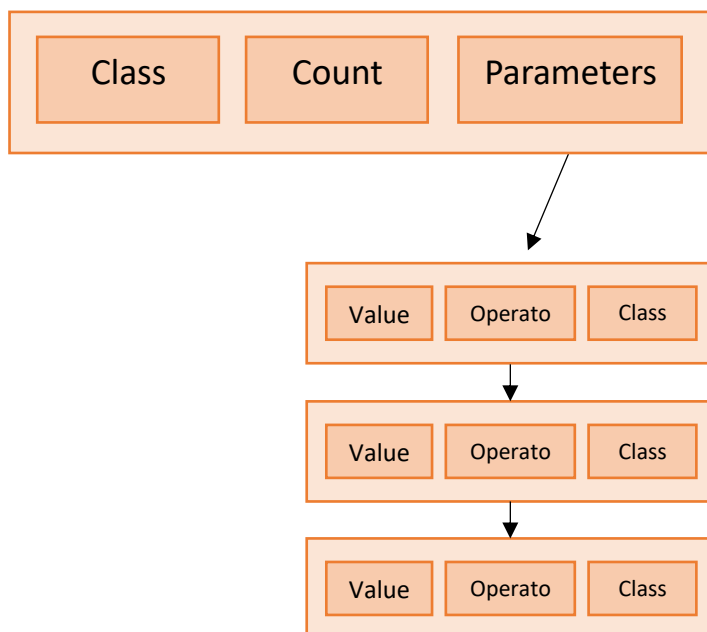
2. על מנת שאוכל לאגד את כל רצף האיברים הקיימים לי אצטרך ליצור רשימת פרמטרים שתכיל את כל רצף האיברים הקיימים בפונקציה.

List< Parameter>



3. אך מכיוון שאני צריכה ליצור אובייקט משוואת פונקציה ולדעת פרטים נוספים חשובים אודות הפונקציה החדשה שנוצרת על כן ניצור אובייקט משוואה שבה נשמור את כל המידע הדרוש אודות הפונקציה.

## Equation



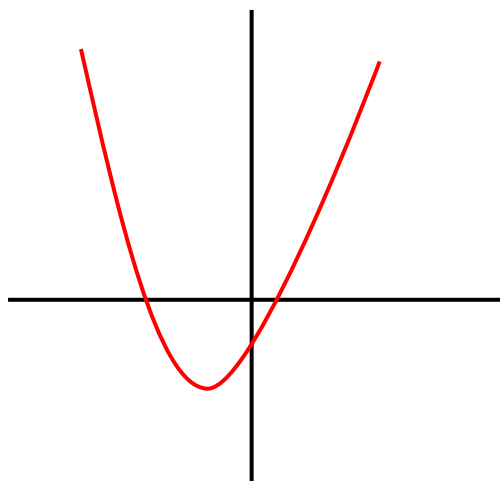
Class – מעלת האיבר הגדול ביותר מבין הפרמטרים

Count – מספר האיברים בפונקציה

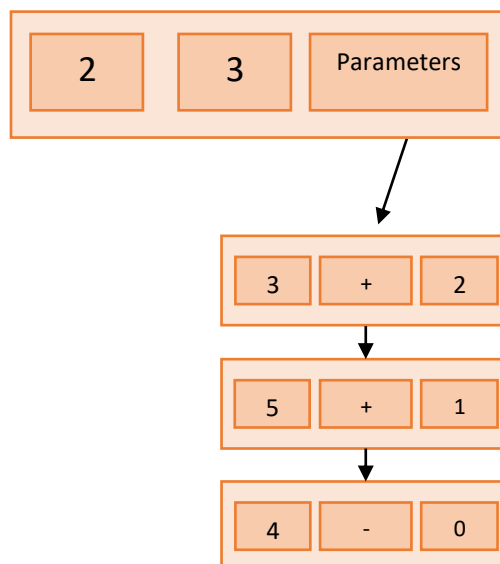
Parameters – רשימת כל אברי הפונקציה הממוינים בסדר יורד



לדוגמא: עבור הפונקציה:  $3x^2 + 5x - 4$



Equation

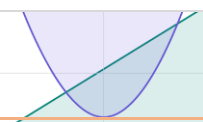


גם חישוב פונקציית הנגזרת מיוצגת ע"י האובייקט Equation שכמובן יכנס לתוכו ע"י חישוב אובייקט המשוואה שאפרט בהמשך.

אחרי שאני יודעת איכן יאוכסן כל המידע אודות הפונקציה נשאלת השאלה איך ממחרזת פשוטה שהמשתמש הכניס למערכת אוכל לפרק את המשוואה לכדי מבני נתונים של אובייקט משוואה.

וכאן האלגוריתם העיקרי מגיע ותפקידו לעבור באופן שיטתי על כל איבר בפונקציה ובו לטפל באופן אישי.

הנה דוגמת קוד להסבר האלגוריתם המרכזי שמפרק את משוואת הפונקציה לאיברים ומכניס אותם לתוך מבנה הנתונים.





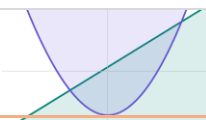
```

//פונקציה המקבלת מחרוזת של פונקציה ומכניסה אותה לתוך אובייקט משוואה
public static Equation getEquationFromStr(string str)
{
    string strCopy = str;
    // מסיר רווחים לא תקינים שנוצרים ויוצר רווח לפני כל איבר חדש
    string strGraph = string.Join(" ", strCopy
        .Replace(" ", "")) // remove all spaces because they kind of random
    now
        .Replace("+", " +") // add space to signs to keep them with their
    value
        .Replace("-", " -")
        .Split(' ');
    //באשר המחרוזת מתחילה באיבר שהוא שלילי (-) יש להסיר את הרווח המיותר
    //ע"י העברת המחרוזת לרשימה והחזרתה שוב למחרוזת רגילה
    if (strGraph[0] == '-')
    {
        //strGraph = strGraph.Skip(0).ToString();
        List<char> strGraphList = new List<char>(strGraph);
        strGraphList.RemoveAt(0);
        //strGraph = strGraphList.ToArray().ToString();
        strGraph = "";
        foreach (var i in strGraphList)
        {
            strGraph += i;
        }
    }

    //split- ע"י איברים מערך של איברים ע"י
    //עבור כל רווח במחרוזת מפרק לאיבר חדש
    string[] s = strGraph.Split(' ');
    //בדיקת נמוח האיברים בפונקציה
    int count = strGraph.Count(f => f == ' ') + 1;

    //יצירת פונקציה חדשה ורשימת פרמטרים אליה יוכנסו נתוני המשוואה
    Equation equation = new Equation();
    equation.Parameters = new List<Parameter>();
    //עובר על כל איבר במשוואה
    for (int i = 0; i < s.Length && s.Length != 0; i++)
    {
        //יצירת איבר - פרמטר לאיבר יחיד שיוכנס לרשימת הפרמטרים- האיברים
        Parameter p = new Parameter();
        //בדיקת מעלת הפרמטר
        switch (s[i][s[i].Length - 1])//הראשון באיבר הראשון
        {
            //1 היא שלו
            case 'x':
            {
                p.Class = 1; //מכניס את מעלת האיבר
                break;
            }
            default:
            {
                //אם המקום השני מהסוף הוא ג (חזקה) נכניס את המעלה
                שנמצאת במקום האחרון באיבר
                if (s[i][s[i].Length - 2] == '^')//גבוה מעלה גבוהה
                {
                    p.Class = Convert.ToInt32(s[i][s[i].Length - 1])
                    - 48; //מכניס את מעלת האיבר
                }
                //0 היא
                else
                {
                    p.Class = 0;
                }
            }
        }
    }
}

```



```

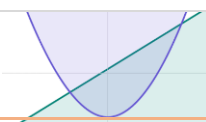
        break;
    }
}
//שולח את האיבר ואת אובייקט הפרמטר ומסיים לאכנס את האיבר...
calcOperator(p, s[i]);
//מוסיף את האיבר הבודד לרימת האיברים - פרמטרים
equation.Parameters.Add(p);
}
//ממין ע"פ מעלת האיברים מהגדול לקטן
equation.Parameters = equation.Parameters.OrderByDescending(c =>
c.Class).ToList();
//מעדכן את מעלת הפונקציה הגבוה ביותר וכן את מספר האיברים בפונקציה
equation.Class = equation.Parameters.Max(m => m.Class);
equation.Count = count;
return equation;
}

```

```

//פונקציה המקבלת את האיבר ואת אובייקט הפרמטר ומסיים לאכנס את האיבר
public static void calcOperator(Parameter p, string v)
{
    int i = 0;
    //מכניס בהתאמה את האופרטור האיבר
    p.Operator = '+';
    if (v[i] == '-')
    {
        p.Operator = '-';
        //אם האופרטור הוא '-' אז צריך לקדם לתו הבא
        i++;
    }
    //אם לא קיים ערך
    if (v[i] == 'x') p.Value = 1;
    //באשר קיים ערך
    else
    {
        //מסירים מהאיבר את כל התווים השונים מהערך וממיינים ע"פ אורך הגדול
        string s = v.Split('+', '-', '^', 'x').OrderByDescending(x =>
x.Length).ToArray()[0];
        if (s == "")
            p.Value = Convert.ToDouble(1);
        else
            p.Value = Convert.ToDouble(s);
    }
}
}

```



ע"י שתי פונקציות אלו נפרק את המחרוזת הפשוטה של גרף הפונקציה ל אובייקט משוואה שבו מאוכסנים נתוני הפונקציה.

הפונקציה `getEquationFromStr` מקבלת מחרוזת משוואה ומסירה ממנה רווחים מיותרים ויוצרת רווח לפני כל איבר חדש במשוואה שע"י כך נוכל לדעת איכן מתחיל איבר חדש ולפרק אותם למערך של איברים.

הפונקציה עוברת על כל איבר ואיבר במערך האיברים ובודקת אותו באופן אישי. וכן יוצרת אובייקט משוואה ובתוכה מוכלת רשימת פרמטרים – איברים שלתוכה נכניס לכל איבר חדש. עוברים על כל איבר מהתו האחרון שלו ובודקים אם התו הוא 'X', אם כן סימן שהמעלה שלו היא 1 ולכן נכניס 1 למעלת האיבר. אם לא נבדוק אם המעלה של האיבר גדולה מ-1 או שהיא 0 שאז זה אומר שהאיבר הוא מספר בודד.

אם המקום השני מהסוף הוא גג (סימן מעלה – דרגה) אזי נדע שהאיבר האחרון באיבר הוא המעלה של האיבר. אחרת אם לא קיים גג סימן שאין מעלה באיבר והוא רק מספר בודד.

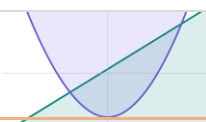
אחרי שחישבנו את מעלת האיבר נעבור לחשב את האופרטור ואת הערך של האיבר ע"י הפונקציה `calcOperator`. הפונקציה מקבלת את מחרוזת של האיבר ואת אובייקט האופרטור ומכניסה לאובייקט האיבר את האופרטור (+/-) בהתאמה.

אם מחרוזת האיבר במקום הראשון הוא 'X' אזי ערך האיבר הוא -1 אחרת נוריד מהמחרוזת ע"י `split` את כל התווים חוץ מתו מספרי ואז ממינים ולוקחים את המקום הראשון שהוא ערך האיבר. וכאן סיימנו להכניס לאובייקט האיבר את כל נתוני האיבר. וחוזרים בחזרה לפונקציה `getEquationFromStr` ששם מכניסים את אובייקט האיבר לתוך רשימת הפרמטרים של המשוואה. וכך עוברים על כל אברי המשוואה.

לבסוף כאשר רשימת הפרמטרים מלאה, ברצוני לבדוק את המעלה בגבוה ביותר ואת כמות האיברים משוואה. לכן נמיין את רשימת הפרמטרים ע"פ המעלה של האיברים מהגבוה לנמוך, וניקח את המעלה של האיבר הראשון אחרי המיון. וכן נכניס את כמות האיברים.

וכך יצרנו אובייקט משוואה המכיל מידע על המשוואה: מעלה הגבוה ביותר, כמות האיברים ורשימת פרמטרים – איברים.

לאחר שיצרנו אובייקט משוואה נוכל לחקור ולפתור את המשוואה ולהגיעה לכל איבר בפונקציה. ע"י נוסחאות מתמטיות נתפור ונמצא נקודות מיוחדות לפונקציות, נקודות קיצון, נגזרת לפונקציה...

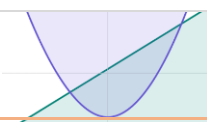


נוסחת שורשים: נוסחה זו מוצאת נקודות אפס בציר האיקס.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \text{ הם } ax^2 + bx + c = 0$$

חישוב טרינום - נקודות חיתוך עם ציר האיקס //

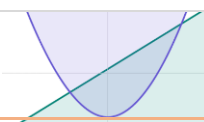
```
double x1 = ((-b) + Math.Sqrt(Math.Pow(b, 2) - 4 * a * c)) / (2 * a);
double x2 = ((-b) - Math.Sqrt(Math.Pow(b, 2) - 4 * a * c)) / (2 * a);
```



ישנם אפשרויות שונות לפתרון הבעיה:

1. המשתמש יוכל להכניס למערכת נקודות מיוחדות של משוואת הפונקציה שלו והמחשב ישרטט בעצמו את הפונקציה על גבי מערכת צירים.
2. אפשרות נוספת שהמשתמש משרטט בעצמו את משוואת הפונקציה על גבי מערכת צירים והמחשב יחשב את מחרוזת המשוואה של הפונקציה ויפתור לו את המשוואה וייתן לו את נגזרת הפונקציה ונקודות מיוחדות.

אני בחרתי את האפשרות שהמשתמש יכניס בעצמו את מחרוזת הפונקציה למערכת והיא תפתור את משוואת הפונקציה ותציג באופן ויזואלי את הפונקציה על גבי מערכת צירים ותוכל להציג בנוסף את משוואת הנגזרת של המשוואה הנתונה כמו כן המערכת תציג את הנקודות המיוחדות של הפונקציה, נקודות אפס, נקודות קיצון.



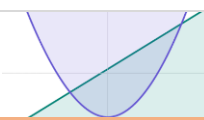
האתר שאני מפתחת עוסק בפתירת משוואות רציונליות והצגתם בצורה ויזואלית על גבי מערכת צירים. המשתמש מכניס את משוואת פונקציה והאתר פותר את המשוואה ומשרטט ויזואלית את משוואת הנגזרת שהוכנסה וכן מציג ויזואלית את משוואת הנגזרת של הפונקציה בנוסף האתר מציג את הנקודות המיוחדות של הפונקציה, נקודות אפס, נקודות קיצון.

הפתרונות הנוספים שקיימים אינם אפקטיביים מכיוון שהפתרון הראשון, שהמשתמש יכניס את הנקודות, אינו מנצל את יכולת המחשב לפתור לבד את המשוואה וכן מדדי ההצלחה לא יהיו מושלמים.

הפתרון השני פחות נח למשתמש שהוא בעצמו משרטט את המשוואה (שהרי זה מה שהאתר שלי עושה) פתרון זה אינו יעיל למשתמש אלא אם כן רק קיימת לו רק שרטוט הפונקציה, מה שפחות נפוץ.

האלגוריתם שפתחתי כולל מבנה נתונים לאליו מוכנס פירוק המשוואה לאיברים ע"י האלגוריתם. האלגוריתם עובר על כל איבר ואיבר במחרוזת המשוואה ומכניס לרשימת הפרמטרים במבנה הנתונים. אחרי שמבנה הנתונים אכן מלא נוכל לגשת לפתירת המשוואה והצגת נקודות, הצגת הנגזרת ושרטוט המשוואה.

את פרוט האלגוריתם כתבתי תחת הכותרת "רקע תאורתי".



חומרה: מעבד i7 RAM 32GB

עמדת פיתוח: מחשב dell

מערכת ההפעלה: Window 10

### סביבת פיתוח :

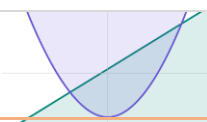
שפות תוכנה: C# , תוך שימוש בטכנולוגיית WebApi , אנגולר .

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

מסד נתונים: SQL SERVER

עמדת משתמש מינימאלית :

- חומרה: מעבד i7 RAM 4GB .
- מערכת ההפעלה: Windows 10 ומעלה.
- חיבור לרשת: נדרש .
- Chrom: תוכנות



## דרישות בהן המערכת צריכה לעמוד:

- כתיבה בסטנדרטים מקצועיים.
- מחשוב השרות ללקוח.
- **3.1 ניתוח דרישות המערכת**
- כתיבת הקוד בסיבוכיות היעילה ביותר.
- ממשק נוח וידידותי למשתמש.
- תגובה מהירה ככל שניתן למשתמש.

## 3.2

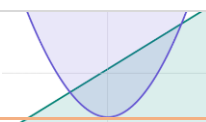
- הזנת פרטי המשתמש והמשוואה בדאטה בייס.
- פתירת המשוואה והצגתה וזאליית ע"י האלגוריתם.
- **מידול המערכת**
- אפשרות לפעולות נוספות הניתנות על המשוואה.
- הכנסת המשוואה לרשימת ההיסטוריה של המשתמש.





פירוט פונקציות עיקריות ותפקידן

1. **getEquationFromStr()** - פונקציה המקבלת מחרוזת של פונקציה ומכניסה אותה לתוך אובייקט משוואה.  
**3.3. אפיון פונקציונאלי**
  2. **calcOperator()** - פונקציה המקבלת את האיבר ואת אובייקט הפרמטר ומסיים לאכסן את האיבר.
  3. **culc\_points()** - חישוב נקודות מיוחדות למשוואת הפונקציה.
  4. **calc\_nigzeret()** - פונקציה מחשבת את נגזרת הפונקציה.
- פונקציות אלו מצד לקוח – אנגולר והם אחראיות להצגת משוואת הפונקציה בצורה ויזואלית.
5. **drawFunc()** - פונקציה מקבלת אובייקט משוואה והופכת אותו בחזרה למחרוזת (string) של משוואת הפונקציה ומחזירה את ה eval של המחרוזת שפותרת את המחרוזת.  
 זאת לאחר שגרף הפונקציה נשלחה לשרת ושם פתרה את הפונקציה ומצאה נקודות מיוחדות, נגזרת... ולאחר מכן מחזירה בחזרה את מבנה ה Equation של הפונקציה למחרוזת.
  6. **generateDataEquation()** - הפונקציה עוברת ב for על מערכת הצירים ובודקת את קימות הפונקציה במקום הנוכחי. ומפעילה את הפונקציה func על כל נקודה במערכת הצירים.
  7. **Func()** - הפונקציית func מפעילה את הפונקציה drawFunc ושולחת לה את ערך הפונקציה ואת אובייקט הפונקציה אותו קיבלנו ואותו צריך להפוך למחרוזת.



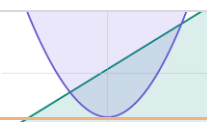
המשתמש יוצר לעצמו חשבון חדש או נכנס עם חשבון קיים וכך היסטוריית הזנת משוואות הפונקציות.

לאחר יצירת החשבון המשתמש יוכל להזין את מחרוזת הפונקציה ולראות את הפונקציה בצורה ויזואלית על גבי מערכת צירים.

בנוסף קיימת אפשרות להצגת נקודות מיוחדות, נקודות אפס, נקודות קיצון וכו'. קיימת אפשרות גם למציאת פונקציית הנגזרת.

המשתמש יוכל לגשת להיסטוריית החיפושים של הפונקציות שלו.

המשתמש מזין את פרטיו האישיים כנדרש ומעלה תמונה עדכנית שלו למקרה הצורך.

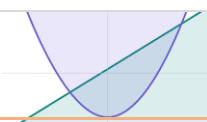


### 3.5.

המערכת משרטטת כל פונקציה שתוכנס למערכת בתנאי שהיא פונקציית פולינום (לא מאפשר פונקציות רציונליות וטריגונומטריות).

כמו כן המערכת תמצא נקודות מיוחדות לפונקציות עד מעלה שלישית.

אך המערכת כן תטפל ותמצא את פונקציית הנגזרת של כל פונקציה פולינומית.



## 4.1.

### 4. תיאור הארכיטקטורה

צד השרת - server side פותח במודל 3 השכבות ומתחלק ל-4 פרויקטים  
**הארכיטקטורה של הפתרון המוצע בפורמט של Design level**  
 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.

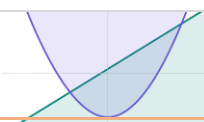
API – שכבת ה Controller – חיבור בין צד השרת והלקוח.

BL – הלוגיקה של המערכת.

DAL – מכיל את הפונקציונאליות הנדרשת לכל התקשורת עם ה - Data Base.

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

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



הפרויקט מחולק ל-2 חלקים:

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

בחרתי לכתוב צד לקוח ב- אנגולר שהינה שפה מתקדמת ועדכנית בעלת מאפייני Angular8 חדשניים ופונקציונאלית ביותר.

אנגולר הינה סביבת עבודה שפותחה על ידי גוגל. מאפשרת לפתח אפליקציות Framework אינטרנט בקלות ומהירות. במקור היא באה לתת מענה לבניית Applications 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 ב-C#. C# היא שפת תכנות עילית מרובת-פרדיגמות, מונחית עצמים בעיקרה המשלבת רעיונות כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיות, פונקציונאליות פרוצדוראליות וגנריות.

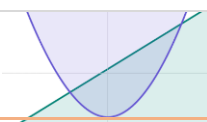
#C היא שפה מעניינת, נוחה ומלאה פונקציונאליות למתכנת. שימוש בשפה זו נפוץ כיום, וכתוצאה מכך, ניתן היה למצוא בה קודים שונים שנדרשו לפיתוח.

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

ה- EntityFramework מאפשר לטעון את הנתונים מה DB-ולעשות להם השמה בצורה ישירה ואוטומטית לתוך אובייקטים בקוד הממפים את מאגר הנתונים בצורה מידית.

ה - EntityFramework -

קורא נתונים מה DataBase שנכתב בשפת SQL. למסדר נתונים של 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.

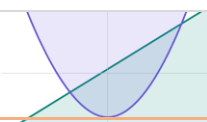


## דוגמא לזרימת מידע במערכת

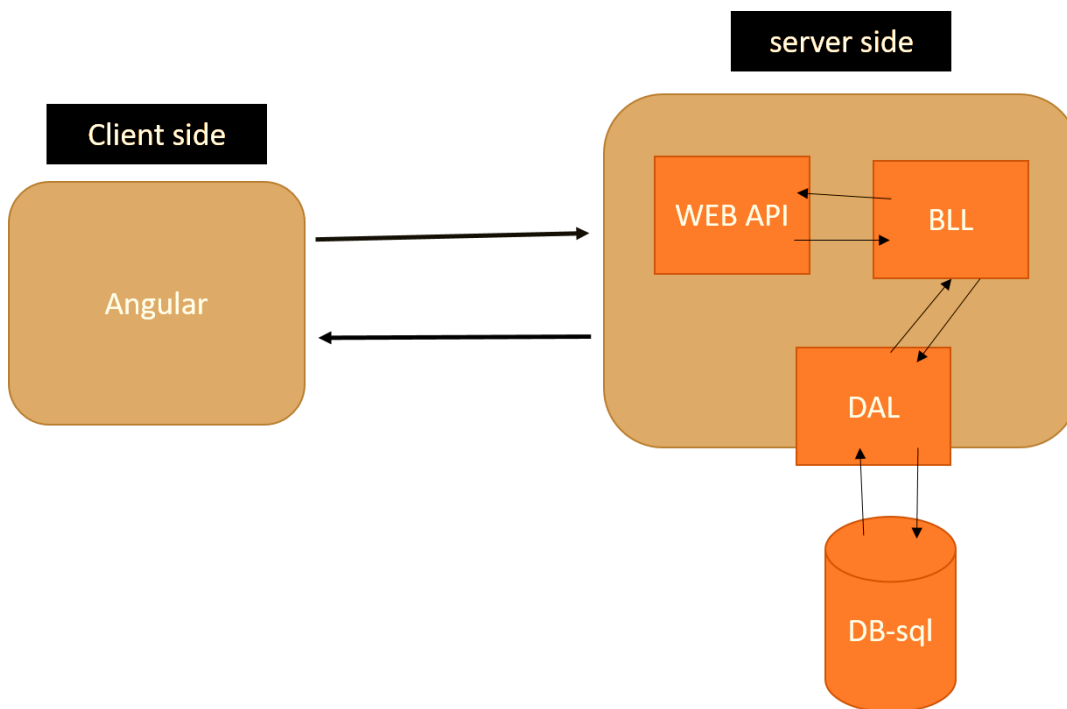
שליפת כל המשתמשים:

ברצוננו לקבל את כל המשתמשים הקיימים במערכת מה - DB ולכן יתבצעו השלבים הנ"ל:

- בכניסה בתור מנהל תהיה אפשרות לראות את כל המשתמשים הקיימים במערכת. הוא ילחץ על כפתור הצג משתמשים בתצוגה (ב-html) ובקשתו תפנה ל-TypeScript.
- תתבצע קריאה לפונקציה ShowPoint ב-TypeScript אשר תפנה לשרת url ותתבצע בקשת services.
- השרת מקבל את הבקשה ומנווט ל Controller שנמצא ב-API.
- ה Controller יזמן את הפונקציה GetPoint שנמצאת ב- GraphManager ב- BL הוא מעוניין לקבל נתונים מה - DB ולכן הוא פונה ל-DAL- דרך ה framework Entity
- ה- DAL שואב את הנתונים הרצויים ממסד הנתונים וכעת מתבצע שלב החזרה.
- ה - DAL מחזיר את רשימת המשתמשים לשכבת הBL בה מתבצעת פונקציית הסינון של הבאת המשתמשים הקיימים במערכת.
- הפונקצייה GetUsers מחזירה את הנתונים מה- controller ל- BL.
- הנתונים מוחזרים ל- controller מה- services.
- מה service חוזרת הרשימה ל - TypeScript.
- הרשימה מוצגת בHTML.



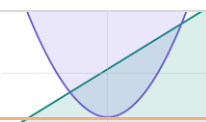
## איור:



1. מסד נתונים SQL SERVER הבנוי מישויות - מטבלאות.
  2. שכבת הגישה לנתונים באמצעות Entity Framework.
  3. שכבת הישויות.
  4. שכבת ה - BL בה כתובים האלגוריתמים.
  5. Web Api פרוטוקול התקשורת בין צד הלקוח וצד השרת.
  6. angular, TypeScript צד לקוח.
- 4.3. תיאור פרוטוקולי התקשורת

**http** – ראשי תיבות של המילים - Hypertext Transfer Protocol הוא פרוטוקול תקשורת שנועד להעברת דפי HTML ואובייקטים ברשת האינטרנט וברשתות האינטראנט. הפרוטוקול פועל בשכבת היישום של מודל ה - OSI ובשכבת **4.4 שרת - לקוח** היישום של מודל TCP/IP.

צד השרת נכתב בטכנולוגיית WebApi ובשפת c#. צד הלקוח נכתב בשפות: TypeScript, HTML, Css בטכנולוגיית Angular.

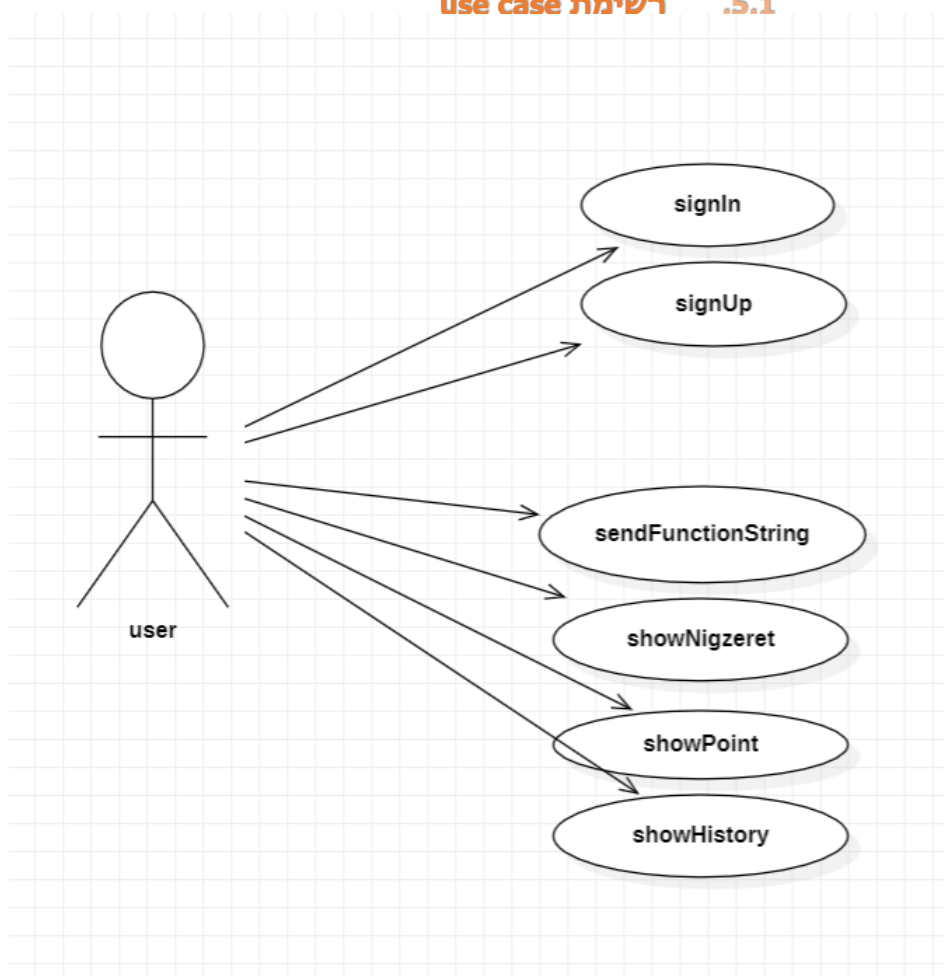


לא רלוונטי

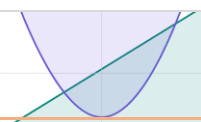
#### 4.5. תיאור הצפנות

### 5. ניתוח ותרשים use case של המערכת המוצעת

#### 5.1. רשימת use case



- המשתמש מכניס את עצמו למערכת כמשתמש חדש או קיים.
- המשתמש מכניס את הפונקציה לאתר שולח אותה ומקבל סרטוט ויזואלי של הפונקציה.
- יכול להציג ויזואלית את גרף הנגזרת
- מציג את הנקודות המיוחדות.
- בנוסף כל משתמש יוכל לראות את היסטוריית חיפשי משוואות הפונקציה שלו.



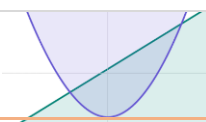


## UC1

- **Identifier UC1:**
- **Name:** הרשמה לאפליקציה. **תיאור ה-use case העיקריים של המערכת**
- **Description:** משתמש מזין את פרטיו האישיים.
- **Actors:** משתמש שלא קיים במערכת.
- **Frequency:** בכניסה לאפליקציה.
- **Condition Pre -:** לא קיים במערכת.
- **Condition – Post:** נתוני המשתמש נקלטים במערכת. המערכת מוסיפה אותם למאגר (ב – Data base).
- **Extended use cases:** בהרשמה לאפליקציה.
- **Assumptions:** הנתונים שהוזנו נכונים ותקינים.
- **Basic course of action:** המערכת שומרת את פרטי המשתמש. כעת המשתמש יכול להיכנס לאזור האישי שלו.
- **Altemate course of action:** אם מספר הזהות כבר קיים במערכת תוצג שגיאה וכן אם המשתמש לא מילא את כל השדות.
- **Change history:** גרסה ראשונה, רינת אביטל.
- **Issues:** הכנסת פרטי משתמש חדש
- **Decisions:** אנו מסתמכים על נכונות הפרטים.

## UC2

- **Identifier UC2:**
- **Name:** התחברות לאפליקציה.
- **Description:** משתמש מכניס מספר זהות וסיסמה.
- **Actors:** משתמש הקיים במערכת.
- **Frequency:** בכניסה לאפליקציה.
- **Condition Pre -:** פרטי המשתמש קיימים במערכת.
- **Condition – Post:** הפרטים נבדקים במידה ואכן קיימים במערכת מתבצעת התחברות – ניתן לגשת לאזור האישי.
- **Extended use cases:** בהתחברות לאפליקציה.
- **Basic course of action:** המערכת קולטת את פרטי המשתמש. על מנת להתחבר המשתמש חייב להיות קיים במערכת.



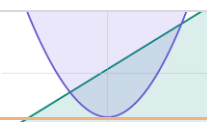
- **Altemate course of action:** אם המשתמש לא זוהה תוצג הודעה "משתמש לא קיים" ואפשרות לנסות שוב.
- **Change history:** גרסה ראשונה, רינת אביטל.
- **Issues:**
  - הכנסת מספר זהות.
  - הכנסת סיסמה.
- **Decisions:** אנו מסתמכים על נכונות הפרטים.

## UC3

- **UC3 :Identifie**
- **Name:** הכנסת מחרוזת פונקציה והצגתה ויזואלית
- **Description:** משתמש מזין מחרוזת של משוואת פונקציה בכדי לראות אותה ויזואלית על גבי מערכת צירים.
- **Actors:** משתמש.
- **Frequency:** כשמשתמש מחובר למערכת.
- **Condition Pre -:** התחברות/ הרשמה.
- **Condition – Post:** המערכת מוסיפה את משוואת הפונקציה למאגר (ב – Data base) ומשרטטת את הפונקציה על גבי מערכת צירים.
- **Assumptions:** הנתונים שהוזנו תקינים, הפונקציה הועלתה בהצלחה ושורטטה מדויק.
- **Basic course of action:** המערכת שומרת את פרטי הפונקציה וניתן לחזור אליה דרך ההיסטוריה.
- **Altemate course of action:** אם הפונקציה לא הועלתה כמו שצריך תוצג הודעה מתאימה.
- **Change history:** גרסה ראשונה, רינת אביטל.
- **Issues:** הכנסת משוואת פונקציה בצורה המקובלת מתמטית.
- **Decisions:** אנו מסתמכים על נכונות הפרטים.

## UC4

- **UC4 :Identifier**
- **Name:** צפיה ויזואלית בפונקציית הנגזרת של המשוואה.
- **Description:** לאחר הכנסת משוואת הפונקציה למערכת המשתמש יוכל לצפות בפונקציית הנגזרת של הפונקציה שהזין.
- **Actors:** משתמש רשום.



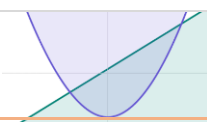
- **Frequency:** לאחר הכנסת משוואת פונקציה.
- **Condition Pre -:** המשתמש הכניס משוואת פונקציה למערכת.
- **Condition – Post:** המשתמש צופה בפונקציית הנגזרת של המשוואה.
- **Assumptions:** הפונקציה שהוכנסה תקינה.
- **Change history:** גרסה ראשונה, רינת אביטל.

## UC5

- **Identifier:** UC5
- **Name:** צפיה בנקודות מיוחדות של הפונקציה.
- **Description:** לאחר הכנסת משוואת הפונקציה למערכת המשתמש יוכל לצפות בנקודות מיוחדות של הפונקציה, נקודות קיצון, נקודות אפס.
- **Actors:** משתמש רשום.
- **Frequency:** לאחר הכנסת משוואת פונקציה.
- **Condition Pre -:** המשתמש הכניס משוואת פונקציה למערכת.
- **Condition – Post:** המשתמש צופה בנקודות מיוחדות של הפונקציה אותה הכניס.
- **Assumptions:** הפונקציה שהוכנסה תקינה והיא ממעלה שניה ומטה.
- **Altemate course of action:** אם לא נמצאה התאמה תוצג הודעה על כך.
- **Change history:** גרסה ראשונה, רינת אביטל.

## UC6

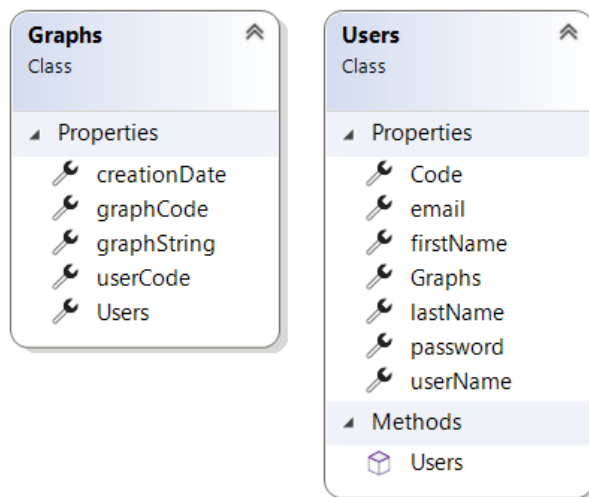
- **Identifier:** UC5
- **Name:** צפיה בהיסטוריית חיפושים של משתמש.
- **Description:** משתמש רשום במערכת יוכל לצפות בכל הפונקציות שכבר הכניס למערכת ולחזור לפתור אותן ולהפעיל עליה פעולות נוספות
- **Actors:** משתמש רשום.
- **Frequency:** לאחר כניסתו למערכת.
- **Condition Pre -:** המשתמש נכנס למערכת.
- **Condition – Post:** המשתמש צופה בהיסטוריית החיפושים של המשוואות שלו.
- **Assumptions:** קיימים חיפושים קודמים של פונקציות למשתמש זה.
- **Altemate course of action:** אם המשתמש לא קיים או שלא קיימים חיפושים קודמים התאמה תוצג הודעה על כך.
- **Change history:** גרסה ראשונה, רינת אביטל.



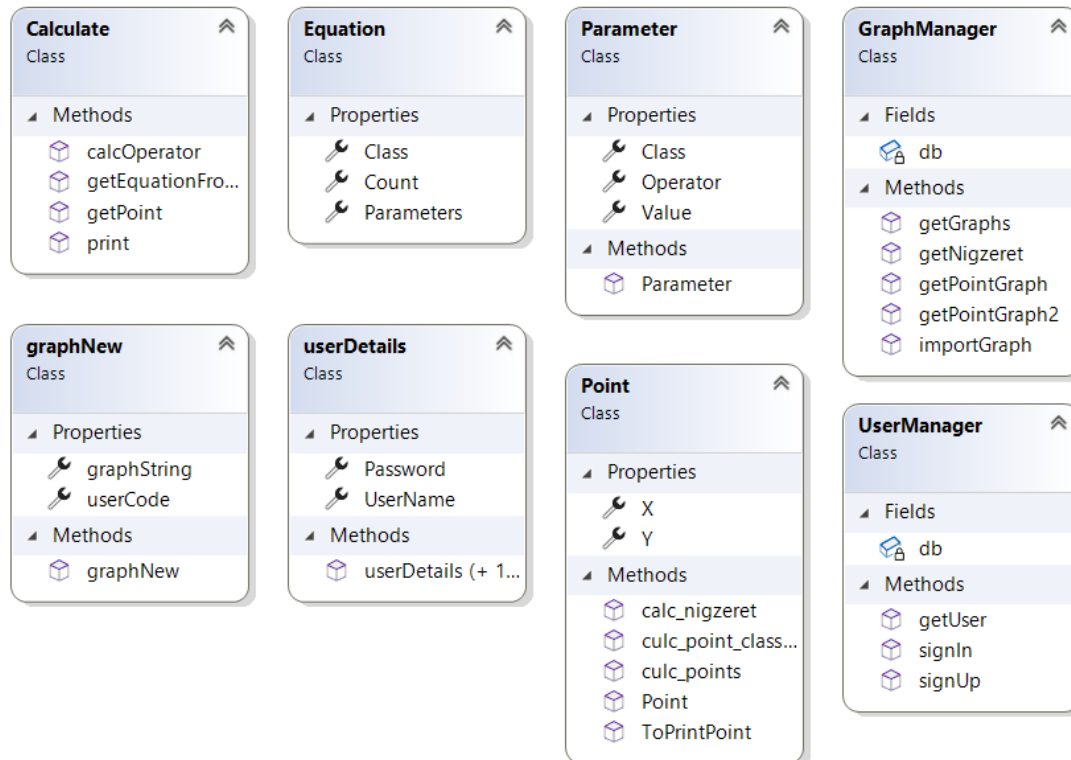
בפרויקט השתמשתי במבנה הנתונים שייצרתי בעצמי לצורך האלגוריתם.  
 מבנה הנתונים הוא Equation שהוא מכיל את כל הפרטים אודות משוואת הפונקציה.  
 מבנה זה מכיל רשימת פרמטרים List מסוג Parameters המכיל את אברי הפונקציה  
 המחולקת כמו שפורט למעלה.  
**5.3 מבני נתונים בהם משתמשים בפרויקט**  
 מבנה נתונים זה עזר ביותר לכתיבת האלגוריתם.

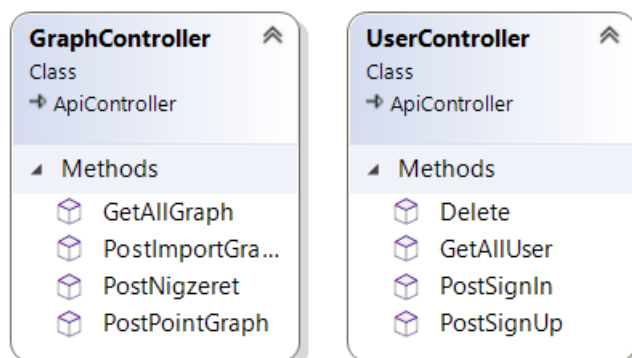


צילום של פירוט המחלקות:



### שכבת ה-BL:



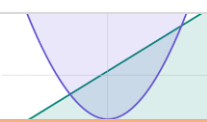


הפרויקט מחולק ל- 3 שכבות. כל שכבה אחראית על תחום מסוים בפרויקט.

השכבה הנמוכה ביותר. שכבה זו אחראית על התקשורת עם ה-Data Base. בשכבה זו פונקציות שונות המפעילות את ההתקשורת.

### המחלקות הקיימות בשכבה:

- **DBSetting** – מחלקה זו משמשת להתקשרות בפועל ל-Data Base. המחלקה מכילה את הניתוב ל-Data Base, שם ה-Data Base, ושם ה-Collection.
- **User** – מחלקה זו מייצגת משתמש במערכת. מכילה את תכונות המשתמש: שם פרטי, שם משפחה, מספר זהות, מייל, פלאפון וסיסמה – כולם מסוג String.
- **UserService** – במחלקה זו קיימות פונקציות האחראיות על פעולות המשתמש:
  - GetCollection – החזרת רשימת כל המשתמשים.
  - Insert – הוספת משתמש חדש למערכת.
  - Update – עדכון נתוני משתמש.



השכבה שמעל ה - DAL היא מקשרת בין ה - DAL ל - API. השכבה אחראית על כל החלק הלוגי של המערכת.

### המחלקות הקיימות בשכבה:

- **userManager** – במחלקה קיימות פונקציות שמשמשות לניהול המשתמשים במערכת:

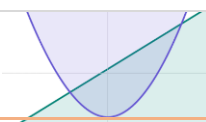
GetUsers – הפונקציה מחזירה משתמש מסוג List.

SignUp – פונקציית הרשמה למערכת.

SignIn – פונקציית התחברות למערכת. הפונקציה מוודאת שהמספר זהות והסיסמה אכן תקינים.

שכבה זו אחראית על החיבור בין צד השרת והלקוח. בשכבה זו קיימים קבצי מערכת רבים, קבצי התקנות, סקריפטים וכו'. בנוסף בשכבה זו קיימים **API** - Controlers – בקרים האחראים על ניתוב התקשורת בין השרת והלקוח:

- **UserController** – במחלקה זו קיימות הפעולות: GET, POST, הרשמה, התחברות, הוספה, מחיקה, LoadPictures – מטרתה לטעון תמונה המתקבלת מהלקוח.



- **סביבת עבודה:**

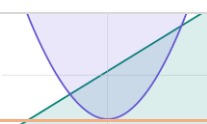
Visual Studio ו Visual Studio Code

**6. תיאור התוכנה**

- **שפות תכנות:**

צד השרת נכתב בטכנולוגיית WebApi ובשפת  $.c\#$ .

צד הלקוח נכתב בשפות: HTML, Css, TypeScript  
בטכנולוגיית Angular.





## .7.1

החלק הראשון והעיקרי של האלגוריתם הוא פרוק מחרוזת (string) גרף הפונקציה לאיברים והכנסתם לתוך מבנה הנתונים Equation שם נמצאים כל אברי הפונקציה ופרטי הפונקציה (פורט למעלה).

## .7.2

החלק השני של האלגוריתם אחרי שמבנה הפונקציה מלא הגיע הזמן לפתור את הפונקציה וליצור רשימת נקודות מיוחדות, נקודות אפס ונקודות קיצון.

## .7.3

חלק נוסף באלגוריתם הוא מציאת גרף הנגזרת של הפונקציה.



## הפונקציות העיקריות בפרויקט:

### 8. פונקציה `getEquationFromStr` - פונקציה המקבלת מחרוזת של פונקציה ומכניסה

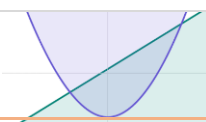
אותה לתוך אובייקט משוואה.

#### 8. קוד האלגוריתם

```
//פונקציה המקבלת מחרוזת של פונקציה ומכניסה אותה לתוך אובייקט משוואה
public static Equation getEquationFromStr(string str)
{
    string strCopy = str;
    // מסיר רווחים לא תקינים שנוצרים ויוצר רווח לפני כל איבר חדש
    string strGraph = string.Join(" ", strCopy
        .Replace(" ", "")) // remove all spaces because they kind of random
    now
        .Replace("+", " +") // add space to signs to keep them with their
    value
        .Replace("-", " -")
        .Split(' ');
    //באשר המחרוזת מתחילה באיבר שהוא שלילי (-) יש להסיר את הרווח המיותר
    //ע"י העברת המחרוזת לרשימה והחזרתה שוב למחרוזת רגילה
    if (strGraph[0] == '-')
    {
        //strGraph = strGraph.Skip(0).ToString();
        List<char> strGraphList = new List<char>(strGraph);
        strGraphList.RemoveAt(0);
        //strGraph = strGraphList.ToArray().ToString();
        strGraph = "";
        foreach (var i in strGraphList)
        {
            strGraph += i;
        }
    }

    //נפרק את המחרוזת מערך של איברים ע"י split-
    //עבור כל רווח במחרוזת מפרק לאיבר חדש
    string[] s = strGraph.Split(' ');
    //בדיקת נמות האיברים בפונקציה
    int count = strGraph.Count(f => f == ' ') + 1;

    //יצירת פונקציה חדשה ורשימת פרמטרים אליה יוכנסו נתוני המשוואה
    Equation equation = new Equation();
    equation.Parameters = new List<Parameter>();
    //עובר על כל איבר במשוואה
    for (int i = 0; i < s.Length && s.Length != 0; i++)
    {
        //יצירת איבר - פרמטר לאיבר יחיד שיוכנס לרשימת הפרמטרים- האיברים
        Parameter p = new Parameter();
        //בדיקת מעלת הפרמטר
        switch (s[i][s[i].Length - 1])//התו האחרון באיבר הראשון
        {
            //אם התו איקס הוא האחרון באיבר סימן שהמעלה שלו היא 1
            case 'x':
            {
                p.Class = 1; //מכניס את מעלת האיבר
                break;
            }
            default:
            {
                //אם המקום השני מהסוף הוא ג (חזקה) נכניס את המעלה
                שנמצאת במקום האחרון באיבר
            }
        }
    }
}
```



```

        if (s[i][s[i].Length - 2] == '^')//הגביל מעלה גבוהה
        {
            p.Class = Convert.ToInt32(s[i][s[i].Length - 1])
            - 48; //מכניס את מעלת האיבר -
        }
        //אחרת המעלה היא 0
        else
            p.Class = 0;

        break;
    }
}
//שולח את האיבר ואת אובייקט הפרמטר ומסיים לאכסן את האיבר...
calcOperator(p, s[i]);
//מוסיף את האיבר הבודד לרימת האיברים - פרמטרים
equation.Parameters.Add(p);
}
//ממין ע"פ מעלת האיברים מהגדול לקטן
equation.Parameters = equation.Parameters.OrderByDescending(c =>
c.Class).ToList();
//מעדכן את מעלת הפונקציה הגבוה ביותר וכן את מספר האיברים בפונקציה
equation.Class = equation.Parameters.Max(m => m.Class);
equation.Count = count;
return equation;
}

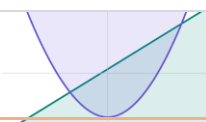
```

**9. פונקציה calcOperator -** פונקציה המקבלת את האיבר ואת אובייקט הפרמטר ומסיים לאכסן את האיבר.

```

//פונקציה המקבלת את האיבר ואת אובייקט הפרמטר ומסיים לאכסן את האיבר
public static void calcOperator(Parameter p, string v)
{
    int i = 0;
    //מכניס בהתאמה את אופרטור האיבר
    p.Operator = '+';
    if (v[i] == '-')
    {
        p.Operator = '-';
        //אם האופרטור הוא '-' אז צריך לקדם לתו הבא
        i++;
    }
    //אם לא קיים ערך
    if (v[i] == 'x') p.Value = 1;
    //נאשר קיים ערך
    else
    {
        //מסירים מהאיבר את כל התווים השונים מהערך וממיינים ע"פ אורך הגדול
        string s = v.Split('+', '-', '^', 'x').OrderByDescending(x =>
x.Length).ToArray()[0];
        if (s == "")
            p.Value = Convert.ToDouble(1);
        else
            p.Value = Convert.ToDouble(s);
    }
}
}

```



## 10. פונקציה culc\_points - חישוב נקודות מיוחדות לפונקציה.

```
//חישוב נקודות מיוחדות לפונקציה
public static List<Point> culc_points(Equation e)
{
    List<Point> Points = new List<Point>();
    double a = 0;
    double b = 0;
    double c = 0;

    //משוואה עם שתי איבריים מלאים
    if (e.Count == 2 && e.Class == 1)
    {
        double a1 = e.Parameters[0].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
        double b1 = e.Parameters[1].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
        double x = (-b1 / a1);
        Points.Add(new Point(x, 0));
        Points.Add(new Point(0, b1));
    }

    //משוואה עם שלושה איבריים מלאים
    else if (e.Count == 3 && e.Class == 2)
    {
        a = e.Parameters[0].Value * (e.Parameters[0].Operator == '-' ? -1 : 1); //להכפיל באופרטור
        b = e.Parameters[1].Value * (e.Parameters[0].Operator == '-' ? -1 : 1); //להכפיל באופרטור
        c = e.Parameters[2].Value * (e.Parameters[0].Operator == '-' ? -1 : 1); //להכפיל באופרטור
    }

    //משוואה עם שתי איבריים ממעלה שניה
    else if (e.Count == 2 && e.Class == 2 && e.Parameters[1].Class == 1)
    {
        a = e.Parameters[0].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
        b = e.Parameters[1].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
        c = 0;
    }

    //משוואה עם שתי איבריים ממעלה שניה
    else if (e.Count == 2 && e.Class == 2 && e.Parameters[1].Class == 0)
    {
        a = e.Parameters[0].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
        b = 0;
        c = e.Parameters[1].Value * (e.Parameters[0].Operator == '-' ? -1 : 1);
    }

    Console.WriteLine();
    Console.WriteLine("print a, b, c");
    Console.WriteLine("a= " + a + "\nb= " + b + "\nc= " + c);
    Console.WriteLine();

    //חישוב טרינום -נקודות חיתוך עם ציר האיקס
    double x1 = ((-b) + Math.Sqrt(Math.Pow(b, 2) - 4 * a * c)) / (2 * a);
    double x2 = ((-b) - Math.Sqrt(Math.Pow(b, 2) - 4 * a * c)) / (2 * a);
}
```



```

Points.Add(new Point(x1, 0));
Points.Add(new Point(x2, 0));
Console.WriteLine("print Cutting points");

Console.WriteLine();

//חישוב קודקוד הפונקציה - נקודת קיצון/
double Xkodkod = (-b) / (2 * a);
double Ykodkod = (a * Math.Pow(Xkodkod, e.Parameters[0].Class)
    + (b * Math.Pow(Xkodkod, e.Parameters[1].Class))
    + (c * Math.Pow(Xkodkod, e.Parameters[2].Class)));
Console.WriteLine("kodkod");
Points.Add(new Point(Xkodkod, Ykodkod));
//Console.WriteLine("(" + Xkodkod + ", " + Ykodkod + ")");

return Points;
}

```

### 11. פונקציה calc\_nigzeret - פונקציה מחשבת את נגזרת הפונקציה

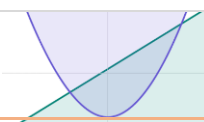
```

//חישוב נגזרת הפונקציה
public static Equation calc_nigzeret(Equation e)
{
    Equation nigzeret = new Equation();
    nigzeret.Parameters = new List<Parameter>();

    int m = 0;

    foreach (var i in e.Parameters)
    {
        if (i.Class == 0) break;
        Parameter p = new Parameter();
        p.Value = i.Value * i.Class;
        p.Operator = i.Operator;
        p.Class = i.Class - 1;
        nigzeret.Parameters.Add(p);
        m++;
    }
    nigzeret.Class = nigzeret.Parameters[0].Class;
    nigzeret.Count = m;
    return nigzeret;
}

```



פונקציות אלו מצד לקוח – אנגולר והם אחראיות להצגת משוואת הפונקציה בצורה ויזואלית.

**12. פונקציה drawFunc** - פונקציה מקבלת אובייקט משוואה והופכת אותו בחזרה למחרוזת (string) של משוואת הפונקציה ומחזירה את ה eval של המחרוזת שפותרת את המחרוזת.  
זאת לאחר שגרף הפונקציה נשלחה לשרת ושם פתרה את הפונקציה ומצאה נקודות מיוחדות, נגזרת... ולאחר מכן מחזירה בחזרה את מבנה ה Equation של הפונקציה למחרוזת.

```
drawFunc(x: number, equation: Equation) {
  let str = "";
  let s = "";
  let i = 0;
  // עובר על כל איבר בפונקציה
  for (let p of equation.Parameters) {
    let v = p.Value;
    let c = p.Class;
    let o = p.Operator;
    let X = "";
    let xx = "";
    for (let i = 0; i < c; i++) {
      X = `*${x}`;
      xx += X;
    }
    s = `${o}${v}${xx}`;
    str += s;
    console.log(str);
    i++;
  }
  // מחזיר את הפונקציה פתורה - מספר
  return eval(str);
}
```

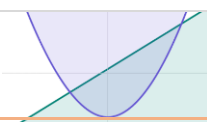
**13. פונקציה generateDataEquation** - הפונקציה עוברת ב for על מערכת הצירים ובודקת את קימות הפונקציה במקום הנוכחי. ומפעילה את הפונקציה func על כל נקודה במערכת הצירים.

```
generateDataEquation() {
  let data: any[] = [];
  for (let i = -200; i <= 200; i += 0.1) {
    data.push([i, this.func(i)]);
  }
  return data;
}
```

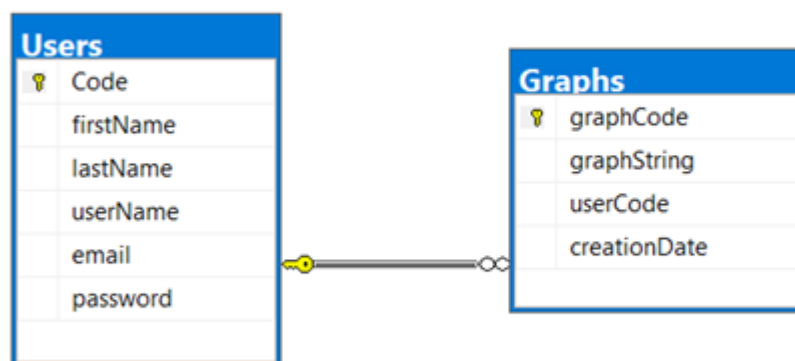


**14. פונקציה func** - הפונקציית func מפעילה את הפונקציה drawFunc ושולחת לה את ערך הפונקציה ואת אובייקט הפונקציה אותו קיבלנו ואותו צריך להפוך למחרוזת.

```
func(x: number) {  
  x /= 10;  
  return this.drawFunc(x, this.equation); // שולח לפונקציית שרטוט  
}
```



תצלמי את הדיאגרמה מה-SQL



### 9.1 פירוט הטבלאות ב-Data Base

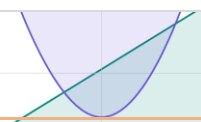
#### טבלאות

הנתונים נשמרים בתוך טבלאות, השדות נשמרים בצורה טבלאית כשלכל שדה יש שם שנשמר במערכת SqlServer, ולפי השמות של השדות נשלפים הנתונים.

#### טבלת משתמשים - Users

הטבלה מכילה את נתוני המשתמשים.

מפתחות	שם השדה	סוג השדה	תאור	שדה חובה
<b>PK</b>	Code	int	קוד	✓
	firstName	varchar(50)	שם משתמש	✓
	lastName	varchar(50)	משפחת המשתמש	✓
	userName	varchar(50)	שם משתמש	✓
	email	varchar(50)	מייל	✓
	password	varchar(50)	ססמה	✓





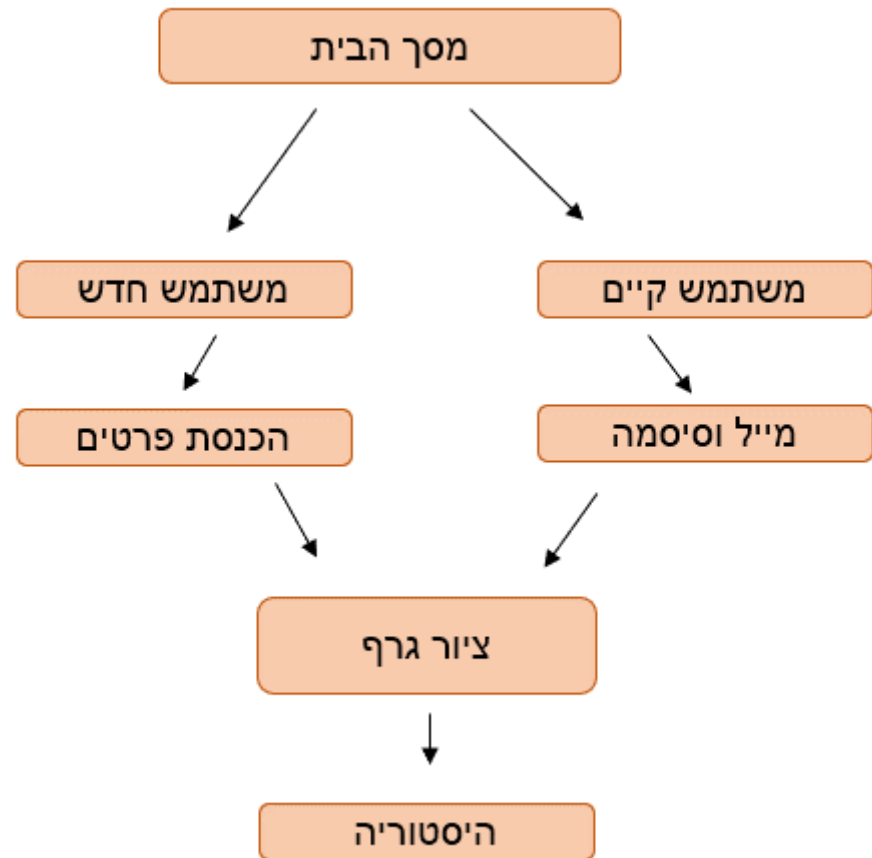
## טבלת גרפים - Graph

טבלה המכילה גרפים.

שדה חובה	תאור	סוג השדה	שם השדה	מפתחות
✓	קוד גרף	int	graphCode	<b>PK</b>
✓	מחרוזת הגרף	(50)varchar	graphString	
✓	קוד משתמש	int	userCode	
✓	תאריך יצירה	datetime	creationDate	



# 10. מסך הבית



צד הלקוח פותח באנגולר ובו כלול ממשק המשתמש עבור הצגת במשוואת הפונקציה באופן ויזואלי ופעולות נוספות.

### מדריך לממשק משתמש

צד הלקוח משרת המשתמשים (סטודנטים, מרצים...) בפתירת משוואות והצגת המשוואה בצורה ויזואלית וכן לראות את ההיסטוריה של החיפוש בייתר קלות ופשטות.

### החלקים העיקריים בפיתוח הממשק הם ארבעה:

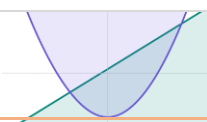
- כניסת משתמש חדש למערכת ו/או משתמש קיים ע"י שם משתמש וסיסמה.
- הכנסת משוואה למערכת והצגתה על גבי מערכת צירים.
- אפשרות הצגת נקודות מיוחדות, נקודות אפס ונקודות קיצון.
- הצגת היסטוריית החיפושים של המשוואות למשתמש שנרשם למערכת.

### המסכים:

מסך הבית

מסך פתיחה, ומסך של כניסת לקוח חדש או קיים.

מסך המאפשר הזנת משוואת פונקציה למערכת ושרטוט הפונקציה. במסך זה תהיה אפשרות לקבלת נקודות מיוחדות, שרטוט פונקציית הנגזרת.



## הצגת מערכת צירים למשתמש

אחד מיעדי הפרויקט החשובים הוא הצגת האתר בצורה יפה ומושכת.

הקדשתי זמן, מחשבה ומעוף בכדי לשרטוט משוואת פונקציה בצורה ויזואלית על גבי מערכת צירים ממוספרת ונראית טוב וברור לעין המשתמש.

במהלך הדרך חקרתי ממשקים נוספים עד שהגעתי לספריית echarts בה בחרתי את הגרף המתאים ביותר עבורי והוא Function Plot - אליו מכניסים את גרף הפונקציה אחרי שפרקנו אותו לאיברים בצד שרת ואז אנו מקבלים את שרטוט הפונקציה באופן ויזואלי. הפונקציה העיקרית עוברת נקודה נקודה במערכת הצירים ע"י for ובודרת האם הנקודה אכן מתקיימת במקום הנוכחי במערכת הצירים.

המסקנה העולה מתהליך היצירה הוא, כי בעיצוב - השמים הם הגבול, וניתן לשפר ולשפץ בלי סוף. עקב מגבלות הזמן, נסתפק בתצוגה פשוטה אך נוחה ונעימה לעין.

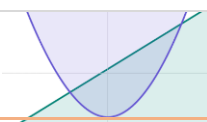
## השלבים בעיצוב:

### 1. שרטוט הפונקציה ע"י קנבס

תחילה, ייבאתי את הורדתי את הספרייה אלי למחשב והתקנתי אותה.

```
import { getInstanceByDom, connect } from 'echarts';
import * as util from 'zrender/lib/core/util';
import * as echarts from 'echarts';
```

כתבתי את קוד הפונקציות של הספרייה וכתבתי שם את האלגוריתם שלוקח אובייקט מסוג משוואה (הכולל את כלל אברי הפונקציה ופרטים נוספים על הפונקציה) ומפשט אותו למחרוזת פונקציה אחרי שבצד שרת שמרנו את פרטי הפונקציה ופתרנו אותה.



```

drawFunc(x: number, equation: Equation) {
  let str = "";
  let s = "";
  let i = 0;
  // עובר על כל איבר בפונקציה
  for (let p of equation.Parameters) {
    let v = p.Value;
    let c = p.Class;
    let o = p.Operator;
    let X = "";
    let xx = "";
    for (let i = 0; i < c; i++) {
      X = `*${x}`;
      xx += X;
    }
    s = `${o}${v}${xx}`;
    str += s;
    console.log(str);
    i++;
  }
  // מחזיר את הפונקציה פתורה - מספר
  return eval(str);
}

```

כאן הפונקציה עוברת ב for על מערכת הצירים ובודקת את קימות הפונקציה במקום הנוכחי. ומפעילה את הפונקציה func על כל נקודה במערכת הצירים

```

generateDataEquation() {
  let data: any[] = [];
  for (let i = -200; i <= 200; i += 0.1) {
    data.push([i, this.func(i)]);
  }
  return data;
}

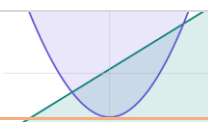
```

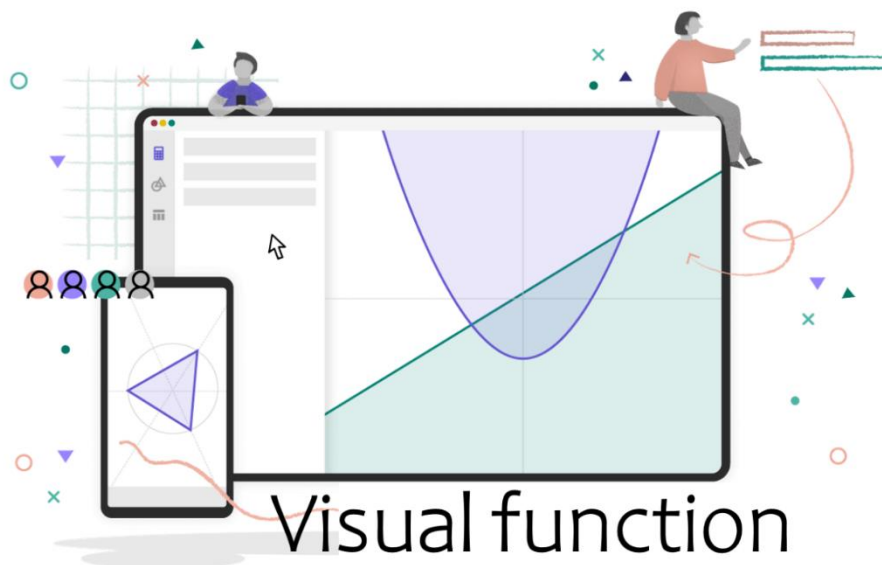
פונקציית func מפעילה את הפונקציה drawFunc ושולחת לה את ערך הפונקציה ואת אובייקט הפונקציה אותו קיבלנו ואותו צריך לפשט.

```

func(x: number) {
  x /= 10;
  // const str = "(+3)*5 ^2 (-2)*5^1 (-2)*5^0" ...
  return this.drawFunc(x, this.equation); // שולח לפונקציית שרטוט
}

```





Sign up and Sign in – התחברות והרשמה

## כניסה

שם משתמש:

סיסמה:

שלח

שם פרטי:

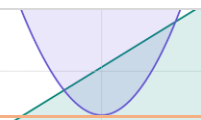
שם משפחה:

שם משתמש:

מייל:

סיסמה:

שלח



שם: Rinat  
שם משתמש: RinatAvital

גרף

שליחת הפונקציה

הצגת נגזרת

הצגת נקודות

הצגת היסטוריה

$x^4 - 2x^3 + 2x + 6$

Keyboard layout: ` ` 1 2 3 4 5 6 7 8 9 0 - = backspace  
tab q w e r t y u i o p [ ] \  
caps a s d f g h j k l ; ' < enter  
shift z x c v b n m , . / shift  
.com @

## draw – שרטוט גרף עם נקודות מיוחדות

שם: Rinat  
שם משתמש: RinatAvital

גרף

שליחת הפונקציה

הצגת נגזרת

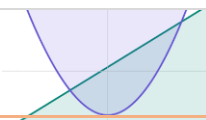
הצגת נקודות

הצגת היסטוריה

$x^2 - 6x + 9$

(3,0)  
(3,0)  
(0,9)  
(3,0)

Keyboard layout: ` ` 1 2 3 4 5 6 7 8 9 0 - = backspace  
tab q w e r t y u i o p [ ] \  
caps a s d f g h j k l ; ' < enter  
shift z x c v b n m , . / shift  
.com @



שם: Rinat
בית כניסה גרף

שם משתמש: RinatAvital

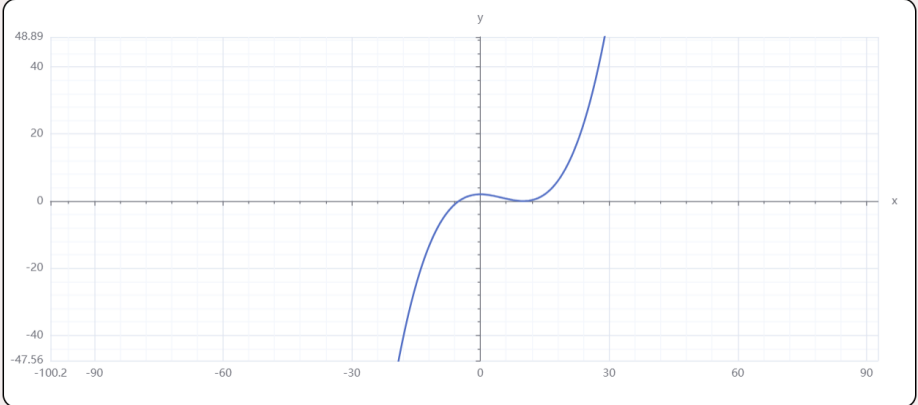
$x^4 - 2x^3 + 2x + 6$

שליחת הפונקציה

הצגת נגזרת

הצגת נקודות

הצגת היסטוריה



'	1	2	3	4	5	6	7	8	9	0	-	=	backspace
tab	q	w	e	r	t	y	u	i	o	p	[	]	\
caps	a	s	d	f	g	h	j	k	l	;	'	<	enter
shift	z	x	c	v	b	n	m	,	.	/		shift	
.com	@												

### draw – שרטוט גרף עם נקודות מיוחדות

שם: Rinat
בית כניסה גרף

שם משתמש: RinatAvital

$x^4 - 2x^3 + 2x + 6$

שליחת הפונקציה

הצגת נגזרת

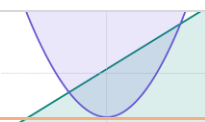
הצגת נקודות

הצגת היסטוריה

$* x^4 - 2x^3 + 2x + 6$   
 $* x^4 - 2x^3 + 2x + 6$   
 $* x^2 - 6x + 9$   
 $* x^2 - 4x - 2$   
 $* 2x^2 - 4x - 2$   
 $* x^2 - 2x - 4$   
 $* x^2 - 3x - 9$   
 $* x^2 - 3x + 9$   
 $* x^2 + 5x - 3$   
 $* x^2 + 5x - 6$   
 $* x^2 + 5x$   
 $* x^2 + 5x$   
 $* x^2 + 5x$   
 $* x^2 + 5x$   
 $* x^2 + 5x - 3$   
 $* x^2 + 5x - 6$



'	1	2	3	4	5	6	7	8	9	0	-	=	backspace
tab	q	w	e	r	t	y	u	i	o	p	[	]	\
caps	a	s	d	f	g	h	j	k	l	;	'	<	enter
shift	z	x	c	v	b	n	m	,	.	/		shift	
.com	@												





אחרי כתיבה מאומצת ומחושבת של קוד האלגוריתם והרצתו שמתאיב לב שחלוקת מחרוזת הפונקציה לאיברים והכנסתה לאובייקט המשוואה, הופיעו באגים או שהפונקציה לא הייתה מדויקת מספיק ולפעמים השתנה או חסרו נתונים, בדקתי ועקבתי רבות על קוד האלגוריתם עד שכל הבעיות תוקנו לאחר בדיקות רבות של מקרי קצה והרצת האלגוריתם מספר פעמים על נתונים שונים. לאחר תיקון הביעה האלגוריתם רץ כמו שציפיתי בצורה המקסימלית.

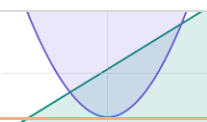


יעילות אלגוריתמית מתייחסת לכמות צריכת משאבי מערכת של אלגוריתם, ובפרט משאבי זמן וזיכרון, אך גם משאבי אנרגיה או רוחב פס יכולים להיכלל בבחינת יעילות של אלגוריתם.

ניתן להתייחס ליעילות אלגוריתמית כמקרה פרטי של יעילות הנדסית.

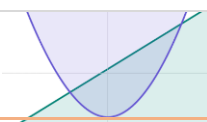
להגדלת היעילות יש לצמצם את השימוש במשאבים. עם זאת, המשאבים השונים אינם ניתנים להשוואה בצורה חד משמעית, ולכן לא תמיד ניתן לקבוע באופן חד משמעי שאלגוריתם יעיל ממשנהו, למשל, פעמים רבות אלגוריתם אחד צורך יותר זמן ופחות זיכרון ממשנהו וצריך לבחון בכל מקרה לגופו איזה משאב יותר חשוב.

ככל שהסיבוכיות האלגוריתם יותר יעילה כך הפרויקט ירוץ יותר ביעילות ובמהירות.



הכניסה לחשבון האישי חייבת להתבצע באמצעות הזנת שם משתמש וסיסמא. כך משתמש יכול להיות רגוע שהמידע אודותיו שמור במערכת וחסוי.

באופן כללי כניסה למערכת לא מחייבת התחברות באמצעות סיסמא אך כל פעילות במערכת של היסטוריית חיפוש של פונקציה לא תהיה קיימת אלא אם תהיה הרשמה או התחברות



סוף כל סוף הגעתי לזמן חתימת הספר, בו חשפתי טפח ממה שמאחורי הפרויקט וסקרתי במעט את תהליך ההתלמדות המופלא, רצוף העמל וההשקעה, עמקני ועשיר בגילויים כה מרתקים, בו הוספתי ידע והתנסות בתחומים רבים ומגוונים.

#### 14. מסקנות

תחילה, תכנון הפרויקט ושלבי העבודה, נראה בעיני כמשימה בלתי אפשרית. זאת בשל ההיקף, המורכבות ולחץ הזמן. חלקים מסוימים שתוכננו נפסלו מראש כי חששתי שלא אצליח לבצע אותם, אך בסופו של תהליך ניתן לומר שהכול אפשרי. כל שהיה נראה בלתי עביר, הפרויקט בכללותו, וכן חלקים מסוימים שנפסלו- הכול התברר כבר ביצוע. השכלתי להבין שעם הרבה מוטיבציה ומוסר עבודה, ניתן להשלים כל משימה, קשה ומורכבת ככל שתהיה.

בנוסף כמעט על כל צעד ושעל בפיתוח נתקלתי בתחומי ידע לא מוכרים, מורכבים, הרבה מעבר לרמת הידע שיש לי. היתקלויות אלה שדרשו פתרונות, גרמו לי להתנסות המון בלמידה עצמית.

ערכתי הכרות עם מגוון כלים וטכנולוגיות, גיליתי שאין בעיה חסרת פתרון וגם לסטודנטית חסרת ידע רחב ומקיף כמוני ישנה אפשרות להיכנס לנושא שהיא לא מכירה, להתאמץ להבין ולמצוא פתרון.

ניתן לומר שכסטודנטית ומתכנתת, הפרויקט הכניס אותי לעולם היוצרים בכלל והמתכנתים בפרט. למדתי לתכנן מערכת ע"י ניתוח ואפיון הצרכים והאתגרים, למדתי לחשוב על כל הפרטים ופרטי הפרטים ועל כל הבעיות שיכולות לצוץ ולמצוא את הדרך היעילה ביותר להתמודד איתן. למדתי המון על דרכי חקירה, פיתוח וכתובה, והרבה על הסיפוק העצום שביצירה.

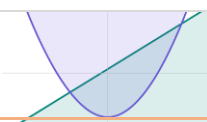
את הפרויקט אפיינתי באופן בולט בכתיבה עצמאית. את התוצר הנפלא יצרתי בעשר אצבעות, בנחישות ובאומץ. שברתי את הראש, כמעט כפשוטו: לכלכתי הרבה את הידיים בכתיבה ומחיקה וכתובה חוזרת. נאבקתי קשות במלחמת הבהירות ונטרול השגיאות. ובעצם, פיתחתי אפליקציה בעצמי, מא' ועד ת'. זו התנסות מדהימה. פיתוח עצמי הוא אולי לא הדרך הכי טובה להגיע למוצר בקלות ובלי להזיע, אולם במבט לאחור, את מטרת הלמידה השגתי גם השגתי, והכלים שרכשתי הם נכס עוצמתי ששווה את הכל ואותו קניתי לתמיד.

את **visual function** אף פעם לא אגדיר כמושלם, כי תמיד אשכללו ואפתחו עוד ועוד, אך את הדרך שעשיתי נסכם אחר אינספור שעות של עמל בחשיבה ובמעשה כי היה זה מדהים לחוות זאת, וסופר יעיל. הפקתי מפרויקט הגמר תועלת מרובה ותקוותי שניסיון זה יסייע לי בעתיד בעזרת ה'.



בהמשך כשיהיה לי את הזמן והאפשרות הייתי מאוד רוצה לשכלל את הפרויקט.  
שלכל פונקציה שהמשתמש יכניס המערכת תוכל לפתור את הפונקציה ולהציג נקודות  
מיוחדות. וכן תוכל לחשב גם אינטגרליים.

**15. פיתוח עתידי**



## 16. ביבליוגרפיה

- GitHub
- Stack Overflow
- Bootstrap
- Internet Israel
- echarts.apache
- ngdevelop.tech
- hodgef
- W3School
- GeeksForGeeks
- ויקיפדיה

