



מערכת מעקב אחר דפי אינטרנט ודיוור התוכן ללקוחות

מיכל זקש | רות גורביץ | 2017

שלמי תודה

בפתח דבר נודה לבורא עולם שהביאנו עד הלום וסייע לאורך כל הדרך.

בנוסף נודה לכל אלו שנתנו רבות מכוחם, מרצם ומתבונתם במהלך העבודה.

✧ תודה - לגב' יהודית חיה שמעוני, חן רוזנבלום, שסייעו לנו לאורך כל הדרך, החל

משלב התכנון ועד לשלב הביצוע. ייעצו, בדקו ונתנו חוות דעת על מנת להגיע להישגים

המקסימליים ולפרויקט ראוי ומושלם.

✧ תודה - למרכזת גב' הדסה גרוס אשר השקיעה וטרחה רבות למעננו,

✧ תודה - לצוות המורות המסור על המקצועיות הרבה, העזרה הרבה, התמיכה

וההשקעה לטווח הרחוק.

✧ לסמינר מרכז בית יעקב, ולעומדים בראשו הרב ישעיה' ליברמן, גב' מינה זלזניק על

החינוך והמקצועיות הרבה, וההשקעה לטווח הרחוק.

✧ תודה - להורינו היקרים ולשאר בני המשפחה וידידים על תמיכה ועידוד לאורך כל

הדרך החל מתקופת הלימודים ועד להגשת פרויקט גמר.



בבואנו לבחור נושא לפרויקט גמר, צעד לפני היציאה לשוק העבודה, חיפשנו תחום שידמה ככל הניתן את עולם התכנות המעשי. תחום מאתגר, מעניין, אלגוריתמי, טכנולוגי ומתקדם.

במקביל נוכחנו כי 13% מהישראלים אינם גולשים באינטרנט, לעומתם 51% מהציבור "החרדי" אינם מחוברים לרשת האינטרנט הפתוחה אלא למייל בלבד. רובו של הציבור החרדי מותיר מאחור את רשת האינטרנט מסיבות מוסריות, ובהתאם לכך, משלם מחיר על ניתוקו מהרשת שמשמשת גם כמאגר מידע מלבד היותה אמצעי תקשורת. כמויות עצומות של מידע זורמות ברשת מכל מקום אל כל מקום בעולם. קצב זרימת המידע, כמויות המידע וכמות המכשירים המחוברים לרשת גדלים בקצב מסחרר והופכים את העולם שלנו ל"כפר גלובלי" שבו המידע נגיש יותר מאי-פעם.

כמענה לקונפליקט שנוצר בין הצורך להתעדכן בזמן אמת לבין הרצון לשמור על ניתוק החלטנו ליצר מערכת המעבירה נתונים ברשת דרך מייל בלבד. המערכת עוקבת אחר שינויים בדפי האינטרנט ומעדכנת את המשתמש על שינויים רלוונטיים.

שימושים לדוגמא: ניהול מעקב אחר תורנות בתי חולים, מעקב אחר חשבון בנק.

מערכת זו עונה על כל הדרישות בכך שמשלבת טכנולוגיות מגוונות יחד עם לוגיקה מורכבת.



תקציר

המערכת מתחלקת לתת מערכות המממשות צרכים שונים של המנגנון השלם: -

- אתר המסופק ללקוחות ומאפשר רישום, ניהול תוכן ושירותים נגשים. האתר מבוסס Asp.net ומעדכן את מסד הנתונים של המערכת.
 - מערכת Net. להפקת נתונים מאתרים, המתבססת על מסד נתונים sql ומשתמשת בגלישה רובוטית. מערכת זו מכילה אלגוריתמים מורכבים לייעול מהירות ואיכות. מערכת זו מתוזמנת ע"י מתזמן המשימות.
 - extension ל – Google Chrome מיועד לבחירת תוכן האתר ע"י הלקוח, קליטת סיסמאות, רישום פעולות וסינון מידע. התוסף נכתב ב html | JavaScript..
- הפיתוח כולל אפיון, תכנון מסד נתונים, כתיבת שאילתות, הקמת service , בניית אתר לתוכנה, כתיבת דפי html ופונקציונליות רחבה JavaScript , פיתוח תוסף ל Chrome וכן עיצוב גרפי ב CSS.



הספר

המעין בספר הזה ילמד על צורת העבודה של המערכת, חלקי המערכת, אופי הפעילות של כל חלק וההשלמה בצירופם יחד, איתור שינויים בין גרסאות באופן יעיל, גלישה רובוטית, ומבנה מסד הנתונים. בנוסף מכיל הספר מדריך בו נמצאים צילומי המסכים והסברים כיצד להתקין את התוסף וכיצד להשתמש במערכת.



תוכן עניינים

7	1 תיאור הפרויקט
7	1.1 תיאור כללי
9	2 מפרט טכני
9	2.1 מפרט לעמדת משתמש
9	2.2 מפרט לעמדת שרת
9	2.3 כלי התוכנה לפיתוח המערכת
9	2.4 שפת התכנות
10	3 יעדי המערכת
10	4 תיחום המערכת
11	5 אלגוריתם מרכזי
13	5.1 מציאת מחרוזת LCS
13	5.1.1 מציאת מחרוזת LCS
15	5.1.2 תכנות דינאמי
17	5.1.3 HIRSCHBERG
23	5.1.4 סינון
29	5.2 סימון השינויים בין הגרסאות
34	6 המודלים והקשרים ביניהם
34	6.1 Data Diagram
35	6.2 Database
38	6.3 מחלקות
47	6.4 פונקציות עזר
49	7 תזמון משימות
52	8 ממשקי המערכת ומדריך למשתמש
52	8.1 מדריך למשתמש באתר
52	8.1.1 דף כניסה
53	8.1.2 עריכת הבקשות של המשתמש
54	8.1.3 Download Extension – הורדת התוסף
55	8.2 extension ל-Google Chrome
55	8.2.1 התקנת extension ל-Google Chrome
57	8.2.2 מדריך למשתמש בתוסף
59	9 מסקנות
60	10 ביבליוגרפיה



1 תיאור הפרויקט

1.1 תיאור כללי

כפי שהוסבר במבוא, ישנם צרכנים רבים, בעיקר בקרב הציבור החרדי, שנמנעים משימוש שוטף באינטרנט מסיבות עקרוניות. מן הצד השני עומד הצורך שלהם להתעדכן על שינויים באתרים החשובים להם.

מעקב אחר שינויים ברשת הוא צורך מורכב. על מנת לבנות תוצר יעיל ונוח לשימוש, עלתה הדרישה לחלק את המערכת למספר רכיבים שמתפקדים יחד כמערכת.

- אתר אינטרנט
- מסד נתונים
- מערכת להפקת נתונים מאתרים, איתור שינויים ועדכון המשתמשים.
- extension ל – Google Chrome

משתמש המעוניין להצטרף לשרות נרשם באתר האינטרנט מגדיר פרטים, כתובת מייל ומוריד משם extension ל – Google Chrome המבצע הקלטה של פעולות הגלישה בדפדפן.

כאשר המשתמש מוצא מידע עליו מעוניין לעקוב הוא מעדכן את התוסף על כך וההקלטה נשמרת.

המשתמש לא יסתפק בכתובת URL בלבד וידרוש הקלטה מכיוון שמעוניין במידע ספציפי שלא מגיע מידית בטעינת האתר אלא לאחר מספר פעולות.

באתר האינטרנט המשתמש מגדיר באילו תדירויות הוא מעוניין להתעדכן והאם מעוניין להתעדכן רק במקרה בו השתנו נתונים או בכל מקרה ועוד.

נתוני המשתמש והקלטות הגלישה נשמרים במסד נתונים SQL בשרת.

בשרת מתוזמנת בתדירות קבועה מערכת שתפקידה לבצע בפועל מעקב אחר דפים ברשת.

המערכת שולפת את נתוני הגלישה של הבקשות שמוגדרות לתזמון הנוכחי, מבצעת גלישה רובוטית ומקבלת את דף הHTML המעודכן.

במידה והמשתמש מעוניין בידע חדש בלבד המערכת משווה בין הגרסאות בודקת



אם היו שינויים רלוונטיים ומדגישה אותם בדף הHTML כדי לחסוך קריאה של הדף כולו.

המערכת שולחת את דפי הHTML הנ"ל לחשבון המייל של המשתמש.



2 מפרט טכני

2.1 מפרט לעמדת משתמש

- דפדפן Google Chrome
- חיבור לרשת
- חשבון מייל

2.2 מפרט לעמדת שרת

- שכבת מסד הנתונים - SQL Server 2014
- Imacros
- חיבור לרשת

2.3 כלי התוכנה לפיתוח המערכת

Visual Studio 2015

2.4 שפת התכנות

- C#
- JavaScript



3 יעדי המערכת

מטרת המערכת לבצע מעקב אחר דפי אינטרנט ודיוור התוכן ללקוחות באופן

יעיל ונגיש. לצורך כך הוגדרו היעדים הבאים:

- ניהול נתוני המשתמשים
- הקלטת פעולות הגלישה בדפדפן
- גלישה רובוטית לקבלת נתונים מעודכנים
- השוואת גרסאות בצורה יעילה אמינה ואיכותית
- הדגשת שינויים בדפי HTML
- שליחת נתונים עדכניים ללקוח

4 תיחום המערכת

- הקלטת פעולות הגלישה תתבצע בדפדפן Google Chrome בלבד.
- המערכת לא מטפלת בדפי HTML שאינם תקינים.
- המערכת אינה מתחייבת ל-100 אחוזי הצלחה, בדיוק זיהוי השינויים.



5 אלגוריתם מרכזי

האלגוריתם המרכזי בפרויקט הוא מציאת הבדלים בין הגרסאות השונות של דפי HTML וסימונם.

לאחר ביצוע "גלישה רובוטית" ברשת וקבלת המידע המעודכן ביותר, יש להחליט האם המידע מצדיק שליחה מחדש של הדף ללקוח- האם נערכו בדף HTML שינויים הרלוונטיים ללקוח.

ההבחנה האם קיים הבדל בין הגרסאות פשוטה (ניתן לרוץ בצורה סדרתית ולהשוות כל תו). אך אין אפשרות להסתפק בה אלא יש לזהות את השינויים מכיוון שהם נדרשים לתהליכים הבאים:

- החלטה האם השינויים שזוהו מצדיקים שליחה מחדש של הדף – האם הם רלוונטיים ללקוח.
- הדגשה של השינויים כדי לחסוך ללקוח קריאה מחדש של הטקסט כולו העשוי להיות ארוך ומיגע.

בניגוד להבחנה שבוצע שינוי, מציאת השינויים בין הגרסאות מאתגרת.

ניקח לדוגמא את המחרוזות הבאות:

First: P O N Y

Second: P Y T H O N

קל לזהות כי המחרוזת השנייה התקבלה לאחר מספר שינויים על המחרוזת המקורית, אך קשה לקבוע מהם השינויים בין המחרוזות- אלו אותיות התווספו למחרוזת החדשה ביחס למחרוזת המקורית ואלו נשמטו.

יתכן כי מן המחרוזת המקורית נמחקו האותיות O,N והתווספו האותיות T,H,O,N.

First: P O N Y

Second: P Y T H O N



יתכן כי מן המחרוזת המקורית נמחקה האות Y והתווספו האותיות Y,T,H.

First: P O N Y

Second: P Y T H O N

ויתכן כי השתנה סדר התווים במחרוזת המקורית והתווספו האותיות T,H.

First: P O N Y

Second: P Y T H O N

נצטרך להחליט על דרך פעולה מוגדרת ואופטימלית.

נגדיר **LCS**-Longest common subsequence

תת מחרוזת של תווים הארוכה ביותר המשותפת ל2 הגרסאות.

תת המחרוזת לא חייבת להופיע באופן רציף אך צריכה לשמור על הסדר של התווים.

בדוגמא: מחרוזת LCS היא: P O N .

המחרוזת P Y אינה מחרוזת LCS מכיוון שאינה הארוכה ביותר.

המחרוזת P Y O N אינה מחרוזת LCS מכיוון שאינה שומרת על סדר התווים.

השוואת גרסאות מתבצעת ע"י מציאת LCS כבסיס.

תו במחרוזת המקורית שאיננו שייך למחרוזת LCS הינו תו שנשמט בגרסה החדשה.

בדוגמא: התו Y נשמט בגרסה החדשה.

תו במחרוזת החדשה שאיננו שייך למחרוזת LCS הינו תו שהתווסף בגרסה זו.

בדוגמא: התווים H,T התווספו בגרסה החדשה.

התווים שזוהו כשונים נבדקים האם הם רלוונטיים ללקוח ומצדיקים שליחה מחדש של הדף, באם נמצאו משמעותיים מודגשים בדף שנשלח.



5.1 מציאת מחרוזת LCS

5.1.1 מציאת מחרוזת LCS

נתונות שתי מחרוזות $X_n = \{x_1, x_2 \dots x_n\}, Y_m = \{y_1, y_2 \dots y_m\}$

נגדיר $LCS(X_n, Y_m)$:

$$LCS(X_n, Y_m) = \begin{cases} \emptyset & \text{if } n = 0 \text{ or } m = 0 \\ LCS(X_{n-1}, Y_{m-1}) + x_n & \text{if } x_n = y_m \\ \text{longest}(LCS(X_{n-1}, Y_m), LCS(X_n, Y_{m-1})) & \text{if } x_n \neq y_m \end{cases}$$

כדי למצוא את המחרוזת הארוכה ביותר המשותפת ל X_n, Y_m יש להשוות את התווים x_n, y_m .

אם הם שווים המחרוזת הארוכה ביותר היא $LCS(X_{n-1}, Y_{m-1}) + x_n$ וודאי יופיע ב LCS .

אם אינם שווים LCS היא הארוכה מבין $LCS(X_{n-1}, Y_m), LCS(X_n, Y_{m-1})$ - לא יתכן שגם x_n וגם y_m יופיעו ב LCS .

לצורך מציאת מחרוזת LCS ניצור 2 טבלאות עם מספר שורות כמספר התווים במחרוזת המקורית, ומספר עמודות כמספר התווים במחרוזת החדשה. בדוגמא טבלאות 6×4 .

כל שורה מייצגת תו במחרוזת המקורית וכל עמודה מייצגת תו במחרוזת החדשה, כך שכל תא בטבלאות מיצג תו במחרוזת המקורית ותו במחרוזת החדשה. טבלה ראשונה תמנה עבור כל תא את מספר התווים המופיעים ב LCS של התווים שהאינדקס שלהם קטן מערכי האינדקס של התא. טבלה שניה תסמן ע"י חיצים לאיזו מחרוזת התא מצטרף (התא שמעל, משמאל או באלכסון).

בטבלת הניקוד נוסיף שורה ראשונה ועמודה ראשונה שמאותחלות ב0, מכיוון שכאשר רצף ריק מושווה עם רצף לא ריק LCS הוא תמיד ריק. מכיל 0 תווים.

דוגמא עבור המחרוזות **P Y T H O N - P O N Y**



	P	O	N	Y
P	0	0	0	0
Y	0			
T	0			
H	0			
O	0			
N	0			

	P	O	N	Y
P				
Y				
T				
H				
O				
N				

$LCS(R_1, C_1)$ נקבע ע"י השוואת התו הראשון בכל אחת מהמחרוזות, P ו P זהים, לכן P מצטרפת למחרוזת LCS והחץ מכוון באלכסון. מחרוזת LCS תכיל כעת תו בודד ולכן ערכו יקבע ל-1.

	P	O	N	Y
P	0	0	0	0
Y	0	1		
T	0			
H	0			
O	0			
N	0			

	P	O	N	Y
P	↖			
Y				
T				
H				
O				
N				

$LCS(R_1, C_2)$ נקבע ע"י השוואת התווים P ו O שאינם זהים, לכן LCS הוא הארוך מבין $LCS(R_1, C_1) = 1$, $LCS(R_0, C_2) = 0$.

ערכו יקבע ע"פ התא שמשמאלו – מצטרף למחרוזת שמשמאלו המכילה P , במקביל החץ יכוון לשמאל.

	P	O	N	Y
P	0	0	0	0
Y	0	1	1	
T	0			
H	0			
O	0			
N	0			

	P	O	N	Y
P	↖	←		
Y				
T				
H				
O				
N				



באותה הדרך נמלא את המשך הטבלה:

		P	O	N	Y
	∅	∅	∅	∅	∅
P	∅	1	1	1	1
Y	∅	1	1	1	2
T	∅	1	1	1	2
H	∅	1	1	1	2
O	∅	1	2	2	2
N	∅	1	2	3	3

		P	O	N	Y
P	∅	↖	←	←	←
Y	∅	↑	↖	↖	↖
T	∅	↑	↖	↖	↖
H	∅	↑	↖	↖	↖
O	∅	↑	↖	←	↖
N	∅	↑	↑	↖	←

החל מהתא התחתון הימני יש לעבור בין התאים ע"פ החיצים עד לתא השמאלי העליון, עבור חץ אלכסון שעוברים דרכו יש לצרף את התו של העמודה שלו למחרוזת LCS.

		P	O	N	Y
P	∅	↖	←	←	←
Y	∅	↑	↖	↖	↖
T	∅	↑	↖	↖	↖
H	∅	↑	↖	↖	↖
O	∅	↑	↖	←	↖
N	∅	↑	↑	↖	←
		P	O	N	

מחרוזת LCS היא P O N

סיבוכיות מקום של האלגוריתם עבור מחרוזות באורכים של m , היא $O(m * n)$

וסיבוכיות זמן הריצה $O(m * n)$

מאחר ודפי ה HTML יכולים להיות גדולים מאד, זמן התגובה ארוך מאד וכן מחשב רגיל לא יוכל להריץ את האלגוריתם עקב מחסור במקום בזיכרון.

להלן פעולות לייעול ושיפור האלגוריתם.

5.1.2 תכנות דינאמי

הניקוד של כל שורה בטבלה נקבע בהתחשב בניקוד של התא שמעליו, התא שבאלכסון והתא שלשמאלו בלבד.



לפי זה ניתן לקבל את הניקוד של שורה מסוימת בטבלה בלי להחזיק בזיכרון את הטבלה כולה אלא רק את השורה המבוקשת והשורה שמעליה.

נחשב את הניקוד של השורה הראשונה, ולפי חישוב זה נחשב את הניקוד של השורה השנייה. לאחר שהשורה השנייה מחושבת נחשב את הניקוד של השורה השלישית תוך דריסה של השורה הראשונה על סמך השורה השנייה בלבד וכן הלאה.

במימוש זה מקום למלא את השורה הראשונה והעמודה הראשונה ב-0, בתא הראשון נזין 0, וכל תא אחר בערך הקטן בשתיים מקודמו- התוצאה תהיה זהה אם נכתוב אפסים מאחר ובשני הצורות נעדיף תמיד לא להשתמש בערכים של השורה והעמודה ראשונה- הערך שלהם הקטן ביותר.

סיבוכיות המקום למציאת LCS ע"י תכנות דינאמי היא $O(m)$.

זמן הריצה למציאת LCS ע"י תכנות דינאמי הוא $O(m^2n)$.

למרות שזמן הריצה של האלגוריתם הבסיסי קטן מזמן הריצה של תכנות דינאמי- $O(m * n)$, תכנות דינאמי יעיל עבור מחרוזות גדולות שאין מספיק מקום בזיכרון להריץ עבורן את האלגוריתם הבסיסי.

מימוש:

```
public int[] lastLineAlign(List<T> x, List<T> y)
{
    List<T> row = y;
    List<T> column = x;
    int minLen = y.Count;
    int[] prev = new int[minLen + 1];
    int[] current = new int[minLen + 1];
    //initialization of the first row
    for (int i = 1; i <= minLen; i++)
    {
        prev[i] = prev[i - 1] + insertion;
    }
    current[0] = 0;
    for (int j = 1; j < column.Count + 1; j++)
    {
        current[0] += deletion;
        //points
    }
}
```




```

for (int i = 1; i < minLen + 1; i++)
{
    if (row[i - 1].ToString() == column[j - 1].ToString())
    {
        current[i] = Max(current[i - 1] + insertion, prev[i - 1] + match,
prev[i] + deletion);
    }
    else
    {
        current[i] = Max(current[i - 1] + insertion, prev[i - 1] +
substitution, prev[i] + deletion);
    }
}
//ready to next row
current.CopyTo(prev, 0);
}
return current;
}

```

HIRSCHBERG 5.1.3

תכנות דינאמי יעיל לשיפור סיבוכיות מקום אך גרוע מבחינת סיבוכיו זמן הריצה שלו.

Dan HIRSCHBERG כתב אלגוריתם למציאת מחרוזת lcs בעל סיבוכיות מקום של $O(mn)$ (לצורך כך השתמש בתכנות דינאמי) אך עם סיבוכיות זמן ריצה של $O(mn)$. האלגוריתם מחשב את הרצף האופטימלי בין 2 מחרוזות. אופטימליות נמדדת כסכום עליות הוספה, מחיקה והחלפה בין תאים.

לשם כך הירשברג הגדיר את הפונקציות $Del(x)$ ו $Ins(y)$ Sub(x, y)

מימוש לדוגמא:

$$Del(x) = -2$$

$$Ins(y) = -2$$

$$Sub(x, y) = \begin{cases} +2 & \text{if } x = y \\ -1 & \text{if } x \neq y \end{cases}$$

כאשר הציון של התא נקבע ע"פ התא שלשמאלו יש להוסיף לציון שלו את הערך $Ins(y)$.

כאשר הציון של התא נקבע ע"פ התא שמעליו יש להוסיף לציון שלו את הערך $Del(x)$.



כאשר הציון של התא נקבע ע"פ התא שבאלכסון יש להוסיף לציון שלו את הערך $Sub(x,y)$.

למרות השינויים הנ"ל קל לראות כי התוצאה תהיה זה לתוצאה שראינו לעיל מכיוון שהיחס בין הציונים נשמר והוא השיקול בקביעת כיוון החץ.

נגדיר $X_{i:j} = X\{x_i, x_{i+1} \dots x_j\}$

הירשברג הניח כי עבור כל i, Y, X קיים j כך שיתקיים:

$$LCS(X_n, Y_m) = LCS(X_{0:i}, Y_{0:j}) + LCS(X_{i:n}, Y_{j:m})$$

ההנחה של הירשברג שקולה להנחה כי עבור כל חלוקה של אחד הרצפים ל-2, ניתן לחלק גם את הרצף השני ל-2 חלקים עבורם חיבור LCS של כל זוג חלקים של Y, X ייתן $LCS(X, Y)$.

הנחה זו ניתנת להוכחה כי עבור כל חלוקה שרירותית של X נוכל להתאים לכל חלק, חלק מתוך LCS שמתאים לו ולכל חלק של LCS נוכל להתאים חלק מתוך Y המתאים לו.

יש לחלק את הטבלה ל-2 חלקים לרוחב באמצע (-אופטימלי), ולחשב ניקוד עבור 2 השורות האמצעיות כך שחישוב הנקודות של השורה העליונה מבניהם יחל מהפינה השמאלית העליונה וחישוב הנקודות של התחתונה יחל מהפינה הימנית התחתונה.

	P	O	N	Y	
P	0	-2	-4	-6	-8
Y	-2	2	0	-2	-4
T	-4	0	1	-1	0
H	-6	-2	-1	0	-2
O		1	0	-4	-5
N		0	2	-2	-3
		-4	-2	0	-1
		-8	-6	-4	-2
					0

יש לסכום את 2 השורות האמצעיות ולמצוא את העמודה הראשונה בעלת הסכום הגבוה.



-2	-1	0	-2
1	0	-4	-5
-1	-1	-4	-7

לאחר חיבור 2 מחרוזות LCS נקבל מחרוזת שציונה הוא סכום הציונים של 2 המחרוזות המרכיבות אותה, כדי לקבל מחרוזת אופטימלית נבחר בחלוקה שתיתן מחרוזת עם ציון גבוה ביותר.

מימוש:

```
public int PartitionY(int[] scoreL, int[] scoreR)
{
    int max_index = 0;
    int max_sum = int.MinValue;
    for (int i = 0; i < scoreL.Length - 1; i++)
    {
        if (scoreL[i] + scoreR[scoreR.Length - i - 1] > max_sum)
        {
            max_index = i;
            max_sum = scoreL[i] + scoreR[scoreR.Length - i - 1];
        }
    }
    return max_index - 1;
}
```

ע"פ עמודה זו והשורה האמצעית הנ"ל נחלק את במחרוזות ונקבל 2 טבלאות שונות.

	P			
P				
Y				
T		O	N	Y
	H			
	O			
	N			

LCS יורכב מצרופי LCS של 2 החלקים שימצאו גם הם באותה הדרך בצורה רקורסיבית.




```

public List<T> Hirschberge(List<T> x, List<T> y)
{
    List<T> resultl = new List<T> (), resultR = new List<T> (), result = new
List<T> ();
    //simple calculation
    if (y.Count <=2 || y.Count <=2)
    {
        result = Lcs (x, y);
    }
    else if (y.Count > 0 && y.Count > 0)
    {
        int xlen = x.Count;
        //X position to cut
        int xmid = xlen / 2;
        int ylen = y.Count;
        //first middle row points
        int[] scoreL = lastLineAlign(CutArr(x, 0, xmid - 1, false), y);
        //second middle row points
        int[] scoreR = lastLineAlign(CutArr(x, xmid, xlen - 1, true), CutArr(y, 0,
ylen - 1, true));
        //Y position to cut
        int ymid = PartitionY(scoreL, scoreR);
        //calculate each pary of the subsequences
        resultl = Hirschberge(CutArr(x, 0, xmid - 1, false), CutArr(y, 0, ymid,
false));
        resultR = Hirschberge(CutArr(x, xmid, xlen - 1, false), CutArr(y, ymid
+ 1, ylen - 1, false));
        result = MyAddRange(resultl.ToList(), resultR.ToList(), resultR.Count);
    }
    return result;
}

```

הפונקציה LCS מוצאת LCS בצורה הפשוטה כך שהמטריצה Path מכילה החיצים ,
המטריצה M מכילה את הניקוד.

```

public List<T> Lcs(List<T> x, List<T> y)
{
    int i, j;
    int[,] M = new int[x.Count + 1, y.Count + 1];
    char[,] Path = new char[x.Count + 1, y.Count + 1];
    for (i = 1; i <= y.Count; i++)
    {

```



```

    M[0, i] = M[0, i - 1] + insertion;
    Path[0, i] = 'I';
}
for (j = 1; j <= x.Count; j++)
{
    M[j, 0] = M[j - 1, 0] + deletion;
    Path[j, 0] = 'u';
}

for (i = 1; i < x.Count + 1; i++)
{
    for (j = 1; j < y.Count + 1; j++)
    {
        if (x[i - 1].Equals(y[j - 1]))
        {
            M[i, j] = Max(M[i - 1, j - 1] + match, M[i - 1, j] + insertion, M[i, j
- 1] + deletion);
            if (M[i, j] == M[i - 1, j - 1] + match)
            { Path[i, j] = 'd'; }
            else if (M[i, j] == M[i - 1, j] + insertion)
            { Path[i, j] = 'u'; }
            else
            { Path[i, j] = 'I'; }
        }
        else
        {
            M[i, j] = Max(M[i - 1, j - 1] + substitution, M[i - 1, j] + insertion,
M[i, j - 1] + deletion);
            if (M[i, j] == M[i - 1, j - 1] + substitution)
            { Path[i, j] = 'd'; }
            else if (M[i, j] == M[i - 1, j] + insertion)
            { Path[i, j] = 'u'; }
            else
            { Path[i, j] = 'I'; }
        }
    }
}
i = x.Count;
j = y.Count;
List<T> res = new List<T>();
while (i > 0 && j > 0)
{
    if (Path[i, j] == 'd')

```



```

{
    if (x[i - 1].Equals(y[j - 1]))
    {
        res.Add(x[i - 1]);
    }
    i -= 1;
    j -= 1;
}
else if (Path[i, j] == 'u')
{
    i -= 1;

}
else if (Path[i, j] == 'l')
{
    j -= 1;
}
}
return res;    }

```

5.1.4 סינון

דפי HTML עשויים להיות ארוכים, אך חלקים גדולים מתוכם אינם משמעותיים ללקוח- גם אם השתנו אינם מצדיקים שליחה מחדש של הדף או הדגשה של השינוי. כדי לשפר את זמני הריצה של האלגוריתם ייערך סינון בתוכן הדפים ויוסרו חלקים ששינוי בהם אינו רלוונטי כך ההשוואה תהייה בין מחרוזות קצרות יותר וממילא מהירה יותר.

נגדיר תווים מיותרים: '\n', '\r', ' '.

תגיות מיותרות: `<script>`, `</script>`, `<!-- -->`, `
`, `<style>`, `</style>`.

דוגמא:

First: `<div>hello</div>
<div>world</div>`

Second: `<div>hello to</div>
<div>my world</div>`

לאחר הסרת התוויות `
` שאינה רלוונטית ללקוח תתקבלנה המחרוזות:

First: `<div>hello</div><div>world</div>`

Second: `<div>hello to</div><div>my world</div>`



LCS: <div>hello</div> <div>world</div>

לאחר ההשוואה בין המחרוזת החדשה לLCS יתקבלו שינויים בתוויות div.

ללקוח יש לשלוח את הדף המקורי כולל התווים המיותרים שהוסרו בזמן ההשוואה שכן למרות ששינוי בהם אינו משמעותי אין הצדקה להסיר אותן לגמרי. מאחר והשינויים אותרו בדף HTML ערוך ללא תגיות יש ליצור דרך למצוא את מיקום התוויות בדף המקורי.

לצורך כך נגדיר מערך ChangeChars מסוג Boolean באורך המחרוזת, בו נסמן את החלקים שהושמטו מן הדף המקורי.

לאחר שסומנו כל החלקים שנשמטו ניצור מערך Arr בגודל הדף הערוך שישמש כאינדקס המתאים לכל תו בדף הערוך את מיקומו בדף המקורי. נמלא את המערך הזה על סמך המערך ChangeChars.

בדוגמא (Second):

:<div>hello to</div>
 <div>my world</div>

ChangeChars :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42		
<	D	I	V	>	H	E	L	L	O		T	O		<	/	D	I	V	>	<	B	R	/	>	<	D	I	V	>	M	Y		W	O	R	L	D		<	/	D	I	V	>

Arr:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42					

רוב השינויים בדפי HTML מתבצעים באזור מרכז הדף ולכן יהיה יעיל מאד להסיר את החלק הזהה בין המחרוזות בתחילתן ובסופן.

בדוגמא הנ"ל:

First: <div>hello</div>
 <div>world</div>

Second: <div>hello to</div>
 <div>my world</div>

לאחר הסרת החלק הזהה בין המחרוזות בתחילתן ובסופן:

First: </div>
 <div>

Second: to</div>
 <div>my



מימוש מחיקת תווים מיותרים:

הפונקציה CharEdit מוחקת מן המחרוזת את התווים מתוך רשימת התווים המיותרים clist. עבור המחרוזת החדשה, שאליה נרצה לגשת עם אינדקסים רלוונטיים, מסמנת במערך ChangeChars את התווים שמחקה.

```
private string CharEdit (String str, Boolean IsNewHtml)
{
    string res = null;
    bool flag;
    for (int i = 0; i < str.Length; i++)
    {
        flag = false;
        foreach (char c in clist)
        {
            if (c.Equals(str[i]))
            {
                flag = true;
                if (IsNewHtml) ChangeChars[i] = true;
                break;
            }
        }
        if (!flag)
        {
            res += str[i];
        }
    }
    return res;}
```

הפונקציה UpdateArr מעדכנת את המערך Arr מתוך המערך ChangeChars.

```
private void UpdateArr()
{
    int ind = 0;
    for (int i = _preSkip; i < NewHtml.Length - _postSkip; i++)
    {
        if (!tmp[i])
            Arr[ind++] = i;
    }
}
```



מימוש מחיקת התחלה וסוף מחרוזת:

הפונקציה CalculatePreSkip מחשבת את מספר התווים למחיקה מתחילת המחרוזת ומעדכנת את המשתנה _preSkip.

```
private void CalculatePreSkip()
{
    int leftLen = EditeSource.Length;
    int rightLen = EditeNewHtml.Length;
    while (_preSkip < leftLen - _postSkip && _preSkip < rightLen - _postSkip
&&
        EditeSource[_preSkip].Equals(EditeNewHtml[_preSkip]))
    {
        _preSkip++;
    }
}
```

הפונקציה CalculatePostSkip מחשבת את מספר התווים למחיקה מסוף המחרוזת ומעדכנת את המשתנה _postSkip.

```
private void CalculatePostSkip()
{
    int leftLen = EditeSource.Length;
    int rightLen = EditeNewHtml.Length;
    int ind = rightLen;
    while (_postSkip < (leftLen) && _postSkip < (rightLen) &&
        EditeSource[leftLen - _postSkip - 1].Equals(EditeNewHtml[rightLen -
_postSkip - 1]))
    {
        SavePostSkip += tmp[(ind--) - 1] ? 1 : 0;
        _postSkip++;
    }
}
```

הפונקציה CalculateSkip קוראת לפונקציות CalculatePreSkip ו-

CalculatePostSkip ובודקת אם נוצרה בעיה בעקבות ריבוי התווים '>','<','.

```
private void CalculateSkip()
{
    CalculatePostSkip();
    CalculatePreSkip();
}
```



```

        if (_preSkip > 0 && EditeNewHtml[_preSkip - 1] == '<' &&
            EditeNewHtml[EditeNewHtml.Length - _postSkip - 1] == '>')
        {
            _postSkip++;
            _preSkip--;
        }
    }
}

```

הפונקציה DelSkip מוחקת מן המחרוזות את _preSkip התווים הראשונים, וכן את _postSkip התווים האחרונים.

```

private string DelSkip(string str)
{
    str = str.Substring(_preSkip);
    str = str.Substring(0, str.Length - _postSkip);
    return str;
}

```

מימוש מחיקת תגיות מיותרות:

הפונקציה TagEdit מוחקת מן המחרוזת את התגיות מתוך רשימת התגיות המיותרות tlist. עבור המחרוזת החדשה, שאליה נרצה לגשת עם אינדקסים רלוונטיים, מסמנת במערך ChangeChars את התווים שמחקה.

```

private string TagEdit (String str, Boolean IsNewHtml)
{
    if (Source.Equals(NewHtml))
    {
        return "";
    }
    int ind = 0;
    string res = null;
    int i;
    bool flag;
    ind = _preSkip;
    while ((i = str.IndexOf("<")) != -1) //begin of tag
    {
        flag = false;
        foreach (var item in list) //tags to remove
        {
            if (str.Length - i - item.Key.Length > 0 && str.Substring(i + 1,
                item.Key.Length).ToLower().Equals(item.Key)) //check if this is tag to remove
            {

```



```

int j = str.IndexOf(item.Value, i);
res = res + str.Substring(0, i);
if (str.Length >= j + item.Value.Length + 1)
{
    str = str.Substring(j + item.Value.Length + 1);
    if (IsNewHtml)//update Arr in case it the new string
    {
        for (int ii = i; ii <= j + item.Value.Length; ii++)
        {
            tmp[Arr[ind + ii]] = true;
        }
    }
    ind += j + item.Value.Length + 1;
    flag = true;
    break;
}
}
if (!flag)
{
    res = res + str.Substring(0, i + 1);
    str = str.Substring(i + 1);
    ind += i + 1;
}
}
res += str;
return res;
}

```

הפונקציה UpdateArr2 מעדכנת את המערך Arr מתוך המערך ChangeChars.

```

private void UpdateArr2()
{
    int ind = 0;
    for (int i = Arr[_preSkip]; i < NewHtml.Length - (_postSkip +
SavePostSkip); i++)
    {
        if (!tmp[i])
            Arr[ind++] = i;
    }
}

```



5.2 סימון השינויים בין הגרסאות

לאחר מציאת מחרוזת ה LCS המכילה את התווים השייכים למחרוזת המקורית ולמחרוזת החדשה יש לאתר במחרוזת החדשה את התווים שהשתנו.

דוגמא א':

First: `<h1> hello </ h1><h3>world</h3>`

Second: `<h1> hello my </h1><h3>world</h3>`

LCS: `<h1> hello </h1><h3>world</h3>`

כאשר נשווה את ה LCS בסריקה סדרתית מסוף המחרוזת נקבל:

`<h1> hello my </h1><h3>world</h3>`

מובן כי השינויים התבצעו בתגית h1 ולכן נדגיש אותה.

פלט: `<h1 style='background-color: yellow'> hello my </h1><h3>world</h3>`

דוגמא ב'

First: `<h1> hello </ h1><h3>world</h3>`

Second: `<h1> hello </h1><div> my </div><h3>world</h3>`

LCS: `<h1> hello </h1><h3>world</h3>`

כאשר נשווה את ה LCS בסריקה סדרתית מסוף המחרוזת נקבל:

`<h1> hello </h1><div> my </div><h3>world</h3>`

נראה לכאורה כי התבצעו שינויים ב-div וב-h1.

מדוע זה קרה?

דף HTML בנוי מתגיות המוקפות בתווים ">" , "<". עקב ריבוי בתווים ">" , "<" יתכנו שיבושים בזיהוי אלו מהם השתנו ואלו נשארו. צריך להתחשב במקרים אלו.

מימוש הדגשת השינויים מתבצע בפונקציה PaintYellow.



הפונקציה עוברת בלולאה על כל התווים שהתווספו לדף החדש החל מהתו האחרון (pos) ומאתרת את התגיות שהשתנו, לאחר שאיתרה תגית שולחת אותה לפונקציה addColor שצובעת אותה.

דף HTML סטנדרטי בנוי מ-2 סוגים של תגיות: תגית בודדת נסמן ב-T1, תגית פותחת ותגית סוגרת- נסמן ב-T2.

T1 תגית פותחת ותגית סוגרת- מסתיימות ב>, תגית פותחת מתחילה ב< ותגית סוגרת ב/>.

T2 תגית בודדת- תגית המתחילה ב< ומסתיימת ב/>.

תווים שניתן לקבל כתו אחרון שהשתנה בתגית:

New T1: <div>hello</div>

1.1 <div>hello</div>

1.2 ><div>hello</div

New T2: <input type="radio" />

2.1 <input type="radio" />

2.2 ><input type="radio" /

Update T1: <div>hello</div>

3.1 <div>hello world</div>

Update T2: <input type="number" value="1"/>

4.1 <input type="number" value="2"/>

חימוש:

```
public void PaintYellow()
{int beginChar = 0, endChar = 0, pos, tmpPos;
bool cont = false;
```

```
for (int i = size - 1; i >= 0; i--)
{
    pos = result[i];
```

```
switch (Html[pos])
{
```



אם התו האחרון שהתווסף הוא > משמע שהתווספה תגית חדשה (בדוגמא 1.1 1.2). יש לבדוק את סוג התווית (2T/1T) ולמצוא את תחילת התגית בהתאם לכך.

```
case '>':
{
    endChar = pos;
    beginChar = endChar;
    while ((beginChar > 0) && (!Html[beginChar].Equals('<'))
    { beginChar--; }

    if (Html[beginChar + 1].Equals('/'))//T1
    {
        endChar = beginChar;
        while ((endChar > 0) && (!Html[endChar].Equals('>'))
        { endChar--; }
        beginChar = endChar;
        while ((beginChar > 0) && (!Html[beginChar].Equals('<'))
        { beginChar--; }
    }
}
break;
```

אם התו האחרון שהתווסף הוא /, משמע שהתווסף T2 (2.2). נמצא את תחילת התגית, ונסמן שאת התו אחרי התגית (> שעבר לסוף) אין לסמן-) המשתנה cont דלוק)

```
case '/':
{
    endChar = pos + 1;
    beginChar = endChar;
    while ((beginChar > 0) && (!Html[beginChar].Equals('<'))
    { beginChar--; }
    cont = true;
}
break;
```

אם התקבל תו אחר, יש לרוץ בלולאה על המחרוזת אחורה עד למציאת אחד מן התווים <, >, /,

```
default:
{
    tmpPos = pos;
    while ((tmpPos > 0) && (!Html[tmpPos].Equals('<')) &&
    (!Html[tmpPos].Equals('>')) && (!Html[tmpPos].Equals('/')))
    { tmpPos--; }
    switch (Html[tmpPos])
```



{
אם התקבל התו <, משמע שמצאנו את התו הפותח (בדוגמא 4.1) ויש למצוא את
התו הסוגר

```
case '<':  
{  
    beginChar = tmpPos;  
    endChar = beginChar;  
    while ((endChar < Html.Length) &&  
(!Html[endChar].Equals('>'))  
        { endChar++; }  
    }  
    break;
```

אם התקבל התו >, השינוי בין התוויות (בדוגמא 3.1) המצביע מצביע לסיום התגית
הפותחת ויש למצוא את תחילה

```
case '>':  
{  
    endChar = tmpPos;  
    beginChar = endChar;  
    while ((beginChar > 0) && (!Html[beginChar].Equals('<'))  
        { beginChar--; }  
    }  
    break;
```

אם התקבל התו /, השינוי החל מהתווית הסוגרת- תווית חדשה (בדוגמא 1.2), יש
למצוא את התגית הפותחת

```
case '/':  
{  
    endChar = pos;  
    while ((endChar > 0) && (!Html[endChar].Equals('>'))  
        { endChar--; }  
    beginChar = endChar;  
    while ((beginChar > 0) && (!Html[beginChar].Equals('<'))  
        { beginChar--; }  
    cont = true;  
    }  
    break;  
default:  
    break;  
    }  
    break;  
}  
addColor(beginChar, endChar);
```



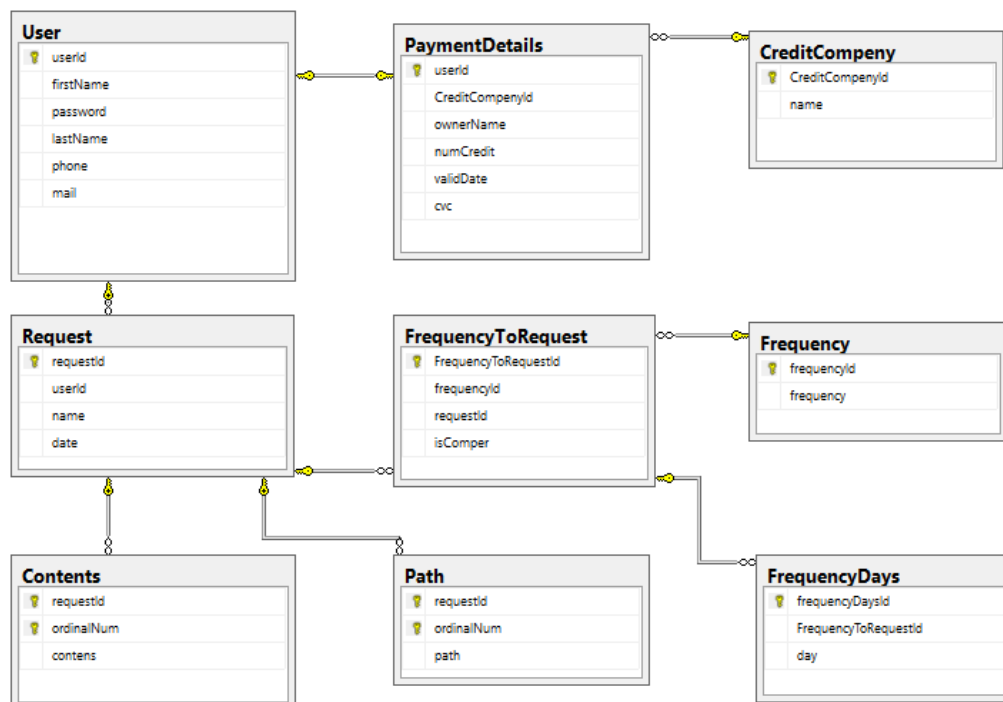
לאחר שתווית מודגשת אין אפשרות להדגיש אותה שוב בעקבות שינויים נוספים שהתבצעו בה ולכן יש להסיר את כל התווים שלה מרשימת התווים שהשתנו.

```
while ((i > 0) && (result[i - 1] >= beginChar))
{
    i--;
}
//מדלג על תווים שאינם נצרכים
if (cont)
{
    cont = false;
    i--;
} }
```




6 המודלים והקשרים ביניהם

Data Diagram 6.1




Users – טבלת משתמשים

בטבלה זו שמורים כל המשתמשים

שם	סוג	הסבר
userId 	string	תעודת זהות
firstName	string	שם פרטי
lastName	string	שם משפחה
password	string	סיסמה
phone	string	טלפון
mail	string	מייל


PaymentDetails – טבלת פרטי תשלום

בטבלה זו שמורים פרטי כרטיס אשראי של המשתמש

שם	סוג	הסבר
userId 	string	תעודת זהות משתמש
CreditCompenyId	int	קוד חברת אשראי
ownerName	string	שם בעל כרטיס
numCredit	string	מספר כרטיס אשראי
validDate	date	תוקף הכרטיס
cvc	string	מספר CVC

CreditCompeny – טבלת חברות אשראי

בטבלה זו שמורים פרטי חברות האשראי הנתמכות ע"י המערכת

שם	סוג	הסבר
CreditCompenyId 	int	קוד חברת אשראי
name	string	שם חברת אשראי

Frequency – תדירות לבקשה


בטבלה זו שמורים רמות תדירויות בהם יבדקו שינויים באתר.

שם	סוג	הסבר
frequencyId 	int	קוד תדירות
frequency	int	תיאור תדירות




Request – טבלת בקשות שרות

בטבלה זו שמורים פרטי בקשות השרות של המשתמש- כל הקלטה היא בקשת שרות.

שם	סוג	הסבר
requestId 	int	קוד בקשת שרות
userId	string	תעודת זהות משתמש
name	string	תיאור הבקשה
date	date	תאריך הבקשה

FrequencyToRequest – תדירות לבקשה

טבלה זו הינה ישות קשר בין בקשה לתדירות

שם	סוג	הסבר
FrequencyToRequestId 	int	קוד תדירות לבקשה
frequencyId	int	קוד תדירות
requestId	int	קוד בקשה
isComper	bit	האם יש לבדוק שינויים

FrequencyDays – ימים של תדירות

בטבלה זו שמורים ימים בהם יבדקו שינויים באתר.

שם	סוג	הסבר
frequencyDaysId 	int	קוד ימים
day	int	קוד תדירות

Contents – הקלטה

בטבלה זו שמורות הקלטות של המשתמשים

ההקלטות יכולות להיות גדולות מכמות המידע של SQL מאפשר לשמור בשדה של טקסט, ולכן נחלק את ההקלטה לכמה רשומות כך שאורך כל רשומה יכול להישמר בשדה מסוג טקסט.

כדי לשמור על סדר בין הרשומות נשמור לכל רשומה מספר סידורי מתאים.
מפתח ראשי יהיה מורכב מצירוף של 2 השדות קוד בקשה ומספר סידורי בהקלטה.



שם	סוג	הסבר
requestId 🔑	int	קוד בקשה
ordinalNum 🔑	int	מספר סידורי
contents	string	תוכן

Path – תוכן HTML

בטבלה זו שמורים דפי הHTML האחרונים שנשלחו ללקוח. דפי הHTML יכולים להיות גדולים מכמות המידע שSQL מאפשר לשמור בשדה של טקסט, ע"מ לשמור דפים גדולים נחלק את הדפים לכמה רשומות כך שאורך כל רשומה יכול להישמר בשדה מסוג טקסט. כדי לשמור על סדר בין הרשומות נשמור לכל רשומה מספר סידורי מתאים. מפתח ראשי יהיה מורכב מצירוף של 2 השדות קוד בקשה ומספר סידורי בהקלטה.

שם	סוג	הסבר
requestId 🔑	int	קוד בקשה
ordinalNum 🔑	int	מספר סידורי
path	string	תוכן



6.3 מחלקות

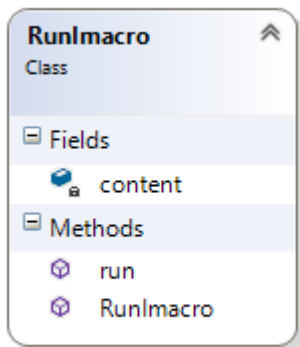
RunImacro

מחלקה זו אחראית על הגלישה הרובוטית.

RunImacro – פעולה בונה (constructor) המקבלת

הקלטה להרצה.

Run – פעולה המבצעת גלישה רובוטית ומאכסנת את התוצאה בקובץ לוקלי, הפעולה מחזירה סטטוס סיום.



מימוש:

```
public iMacros.Status run()
{
    File.WriteAllText("mac.iim", content);
    int timeout = 60;
    iMacros.Status status;
    iMacros.AppClass app = new iMacros.AppClass();
    String result = "";
    status = app.iimInit("-V7", true, "", "", "", timeout);
    if (status != iMacros.Status.sOk) return status;
    result = result + "init " + Convert.ToString(status) + "; ";
    string macro = Directory.GetCurrentDirectory() + "\\mac.iim";
    status = app.iimDisplay("Interface version =\n" +
app.iimGetInterfaceVersion().ToString(), timeout);
    if (status != iMacros.Status.sOk) return status;
    result = result + "display " + Convert.ToString(status) + "; ";
    status = app.iimPlay(macro, timeout);
    if (status != iMacros.Status.sOk) return status;
    result = result + "play " + Convert.ToString(status) + "; ";
    status = app.iimExit(timeout);
    if (status != iMacros.Status.sOk) return status;
    result = result + "exit " + Convert.ToString(status);
    return status;
}
```



מחלקה האחראית על עריכת דפי HTML לפני הפעלת אלגוריתם LCS ע"מ לחסוך

בסיבוכיות זמן ומקום.

המחלקה יוצרת מערך אינדקסים המאפשר גישה מדפי

HTML הערוכים לדפי HTML המקוריים.

EditHtml – פעולה בונה (constructor) המקבלת

מחרוזת מקור ומחרוזת חדשה הפונקציה עורכת את

דפי הHTML.

CalculateSkip – פונקציה הקוראת לפונקציות

CalculatePostSkip ו CalculatePreSkip, תואר לעייל

באלגוריתם המרכזי.

CalculatePreSkip – פונקציה המחשבת את מספר

התווים הראשונים בהם המחרוזות זהות, תואר לעייל

באלגוריתם המרכזי.

CalculatePostSkip – פונקציה המחשבת את מספר

התווים האחרונים בהם המחרוזות זהות, תואר לעייל

באלגוריתם המרכזי.

DelSkip – פונקציה המוחקת את התווים הראשונים

והאחרונים שזהים בין המחרוזות, תואר לעייל באלגוריתם המרכזי.

CharEdit – פונקציה המוחקת תווים שאינם משמעותיים, תואר לעייל באלגוריתם

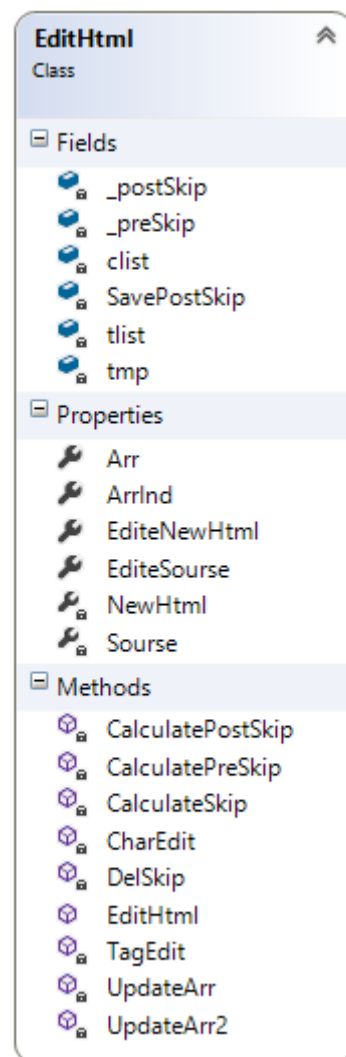
המרכזי.

TagEdit – פונקציה המוחקת תגיות שאינן משמעותיות, תואר לעייל באלגוריתם

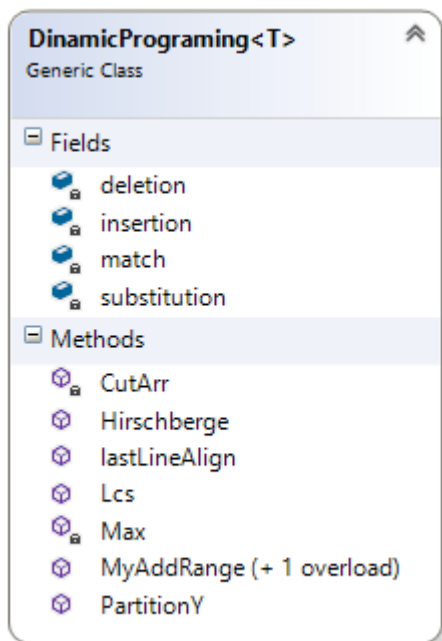
המרכזי.

UpdateArr – פונקציה המעדכנת את מערך האינדקסים לאחר הסרת התווים

שאינם משמעותיים, תואר לעייל באלגוריתם המרכזי.



UpdateArr2 – פונקציה המעדכנת את מערך האינדקסים לאחר הסרת התגיות שאינן משמעותיות, תואר לעייל באלגוריתם המרכזי.



DynamicProgramming

מחלקה גנרית המוצאת את מחרוזת LCS

Hirschberge – פונקציה ראשית המוצאת את מחרוזת LCS בצורה רקורסיבית, תואר לעייל באלגוריתם המרכזי.

lastLineAlign – פונקציה המחשבת ניקוד של שורה מסוימת, תואר לעייל באלגוריתם המרכזי.

CutArr – פונקציה המחזירה חלק ממחרוזת ע"י אינדקס התחלה ואינדקס סיום, תואר לעייל באלגוריתם המרכזי.

PartitionY – פונקציה המקבלת ניקוד של 2 שורות ומוצאת את האינדקס בו יש לחלק, תואר לעייל באלגוריתם המרכזי.

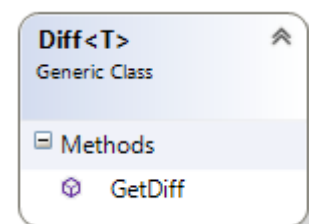
Lcs – פונקציה המוצאת מחרוזת LCS בצורה פשוטה, תואר לעייל באלגוריתם המרכזי.

Max – פונקציה המקבלת 3 מספרים ומחזירה את המקסימלי מבניהם.

MyAddRange – פונקציה גנרית המחברת 2 רשימות.

Diff

מחלקה גנרית האחראית על מציאת החלקים השונים בין 2 רצפים לאחר שנמצא LCS שלהם.

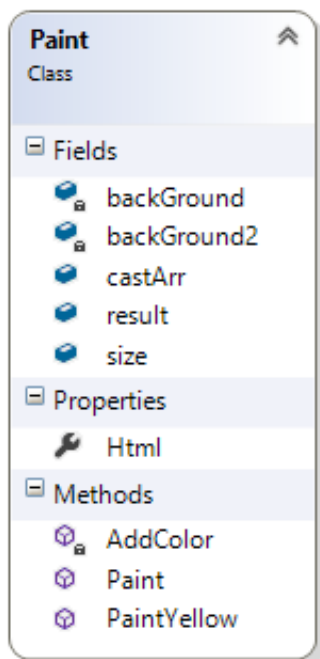


GetDiff – פעולה סטטית המקבלת רצף מקור וLCS

ומחזירה את החלקים השונים.



Paint



מחלקה זו אחראית על צביעת התוויות שהשתנו .

Paint –פעולה בונה (constructor) המקבלת את הדף לצביעה, מערך אינדקסים של תווים שהשתנו (תוצאה אחרי הרצה של LCS) וגודלו, מערך תווים הממיר לאינדקסים מעודכנים.

addColor –פונקציה המקבלת מיקום התחלה וסוף של תגית וצובעת אותה.

PaintYellow – פונקציה המוצאת את ההתחלה והסוף של התגיות שהשתנו ושולחת אותם ל `addColor`, תואר לעיל באלגוריתם המרכזי.

מימוש:

```
private string backGround = " style='background-color:yellow'";
private string backGround2 = " ;background-color:yellow";
private void addColor(int begin, int end)
{
    int ind = Html.IndexOf("style", begin, end - begin + 1), pos;
    // in case the tag doesn't have "style"
    if (ind == -1)
    {
        pos = Html.IndexOf("/", begin, end - begin + 1);
        if (pos == -1)
        {
            Html = Html.Insert(end, backGround);
        }
        else
        {
            Html = Html.Insert(pos, backGround);
        }
    }
    // in case the tag has "style"
    else
    {
        char[] c = new char[2];
```



```

c[0] = "";
c[1] = "".ToCharArray()[0];
pos = Html.IndexOfAny(c, ind, end - begin + 1);
pos = Html.IndexOfAny(c, pos + 1, end - begin + 1);
if (pos != -1)
{ Html = Html.Insert(pos, backGround2); }
}
}

```

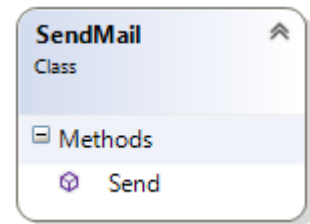
SendMail

מחלקה זו אחראית על שליחת דפי HTML למשתמש.

Send – פונקציה סטטית המקבלת HTML כמחרוזת תווים,

שם של האתר וכתובת יעד.

הפונקציה יוצרת דף HTML ושולחת אותו למשתמש.



מימוש:

```

public static void Send(string to, string html, string name)
{
    File.WriteAllText("tmp.html", html);
    MailMessage mail = new MailMessage();
    SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com", 587);
    mail.From = new MailAddress("projectmbyht@gmail.com");
    mail.To.Add(to);
    mail.Subject = "on time "+name;
    mail.AlternateViews.Add(getEmbeddedImage("logo.png"));
    Attachment item = new Attachment("tmp.html");
    mail.Attachments.Add(item);
    SmtpServer.UseDefaultCredentials = false;
    SmtpServer.Credentials = new
System.Net.NetworkCredential("projectmbyht@gmail.com", "project123");
    SmtpServer.EnableSsl = true;
    SmtpServer.Send(mail);
    item.Dispose();
    File.Delete("tmp.html");
}

//add image
private static AlternateView getEmbeddedImage(String filePath)
{
    LinkedResource inline = new LinkedResource(filePath);
    inline.ContentId = Guid.NewGuid().ToString();
}

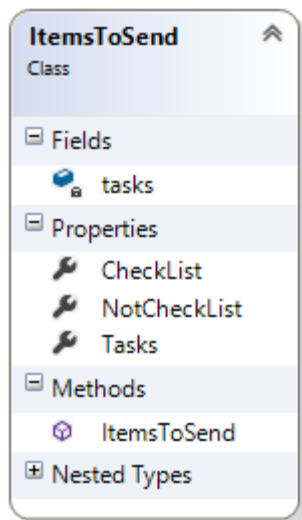
```



```

string htmlBody = @"<img src='cid:" + inline.ContentId + @"/>";
AlternateView alternateView =
AlternateView.CreateAlternateViewFromString(htmlBody, null,
MediaTypeNames.Text.Html);
alternateView.LinkedResources.Add(inline);
return alternateView;
}

```



ItemsToSend

מחלקה זו אחראית על ניהול בקשות שצריכות להתבצע בתזמון הנוכחי.

ItemsToSend – פעולה בונה (constructor) השולפת

ממסד הנתונים את כל הבקשות לתוך Tasks.
 הפונקציה ממיינת את הבקשות שצריכות להתבצע בתזמון הנוכחי לבקשות שיש לבדק (CheckList) ולבקשות שיש לשלח בלי לבדק (Not CheckList).

```

public ItemsToSend()
{
    CheckList = new List<Request>(); //Request to check
    NotCheckList = new List<Request>(); //Request to send with out check
    List<Request> requestList = new List<Request>();
    Boolean flag;
    var t = Tasks.FrequencyToRequest;
    var yyy = Tasks.FrequencyToRequest.ToList();
    Tasks.Frequency.ToList();
    Tasks.Path.ToList();
    Tasks.Contents.ToList();
    Tasks.FrequencyDays.ToList();
    Tasks.User.ToList();
    var tmp = Tasks.Request.ToList();
    foreach (Request request in tmp)
    {
        flag = false;
        var frequencyToRequestTmp =
request.FrequencyToRequest.ToList();
        foreach (FrequencyToRequest frequencyToRequest in
frequencyToRequestTmp)
        { switch (frequencyToRequest.Frequency.frequencyId)

```



```

    {
        case (int)Frequency.DAYLY:
            flag = true;
            break;
        case (int)Frequency.WEEKLY:
            foreach (FrequencyDays day in
frequencyToRequest.FrequencyDays.ToList())
            { if ((int.Parse(day.day)) == (int)DateTime.Today.DayOfWeek)
                {
                    flag = true;
                    break;
                }
            }
            break;
        case (int)Frequency.MONTHLY:
            var frequencyDaysTmp =
frequencyToRequest.FrequencyDays.ToList();
            foreach (FrequencyDays day in frequencyDaysTmp)
            {
                if ((int.Parse(day.day) == (int)DateTime.Now.Day))
                {
                    flag = true;
                    break;
                }
            }
            break;
        default:
            break;
    }
    if (flag)
    {
        if (frequencyToRequest.isComper == true)
        {
            CheckList.Add(request);
        }
        else
        {
            NotCheckList.Add(request);
        }
        break;
    }
}
}
}
}
}

```

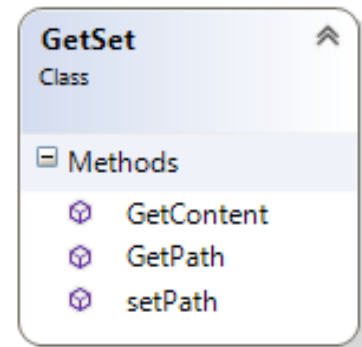


GetSet

מחלקה האחראית על שליפה והשמה של Path, Content השמורים ככמה רשומות שונות.

GetContent – פונקציה השולפת את Content.

GetPath – פונקציה השולפת את Path.



SetPath – פונקציה המעדכנת את Path.

מימוש:

```
public static string GetContent(Request r)
{
    string str = "";
    var list = r.Contents.ToList();
    for (int i = 0; i < list.Count(); i++)
    {
        str += list.Where(p => p.ordinalNum == i + 1).ToList()[0].contents;
    }
    return str;
}
```

```
public static void setPath(Request r, string str)
{
    List<Path> list = r.Path.ToList();

    for (int i = 0; i < list.Count(); i++)
    {
        r.Path.Remove(list[i]);
    }
    ItemsToSend.Tasks.SaveChanges();
    int ind = 1; string tmp; Path path;
    while (str.Length > 4000) //max length in SQL field
    {
        tmp = str.Substring(0, 4000);
        path = new Path();
        path.requestId = r.requestId;
        path.Request = r;
        path.ordinalNum = ind++;
        path.path1 = tmp;
    }
}
```



```
r.Path.Add(path);
ItemsToSend.Tasks.SaveChanges();
str = str.Substring(4000, str.Length - 4000);
}
if (str.Length > 0) //last record
{
    path = new Path();
    path.requestId = r.requestId;
    path.Request = r;
    path.ordinalNum = ind++;
    path.path1 = str;
    r.Path.Add(path);
    ItemsToSend.Tasks.SaveChanges();
}
}
```

בנוסף לכל המחלקות הנ"ל, יש מחלקות המתארות את הטבלאות במסד נתונים.
(Entity).



6.4 פונקציות עזר

מלבד הפונקציות שתוארו בסעיף קודם שנכתבו ב-C#, השתמשנו בפונקציות עזר נוספות בשפות שונות.

חורדת extension ל- Google Chrome כקובץ zip

Asp.net

```
public FileResult DownloadFiles()
{
    //Define file Type
    string fileType = "application/octet-stream";

    //Define Output Memory Stream
    var outputStream = new MemoryStream();
    // System.IO.Compression.ZipFile d;

    //Create object of ZipFile library
    using (ZipFile zipFile = new ZipFile())
    {
        //Add Root Directory Name "Files" or Any string name
        zipFile.AddDirectoryByName("Files");

        //Get all filepath from folder
        String[] files = Directory.GetFiles(Server.MapPath("/onTime"));
        foreach (string file in files)
        {
            string filePath = file;

            //Adding files from filepath into Zip
            zipFile.AddFile(filePath, "OnTime");
        }

        Response.ClearContent();
        Response.ClearHeaders();

        //Set zip file name
        Response.AppendHeader("content-disposition", "attachment;
filename=OnTime.zip");

        //Save the zip content in output stream
        zipFile.Save(outputStream);
    }
}
```



```

//Set the cursor to start position
outputStream.Position = 0;

//Dispance the stream
return new FileStreamResult(outputStream, fileType);
}

```

טעינת קובץ הקלטה לשרת

Java Script

```

<script>
  window.onload = function () {
    // file label
    var fileInput = document.getElementById('fileInput');
    //label to show image content
    var fileDisplayArea = document.getElementById('fileDisplayArea');
    fileInput.addEventListener('change', function (e) {
      var file = fileInput.files[0];
      var reader = new FileReader();
      reader.onload = function (e) {
        fileDisplayArea.innerText = reader.result;
        document.getElementById('content').value= reader.result;
      }
      reader.readAsText(file);
    });
  }
</script>

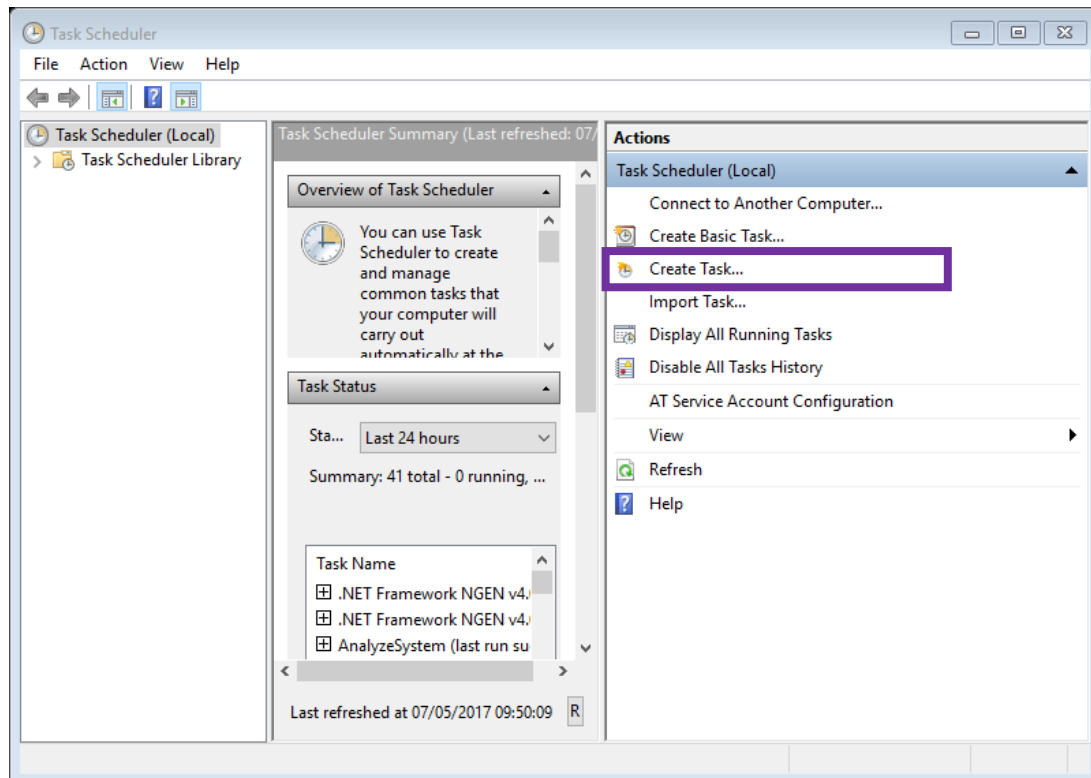
```



7 תזמון משימות

מעקב אחר שינויים מתבצע בסבב של פעם ביום, לצורך הפעלת התוכנית בתדירות זו שתמשנו במתזמן המשימות- Task Scheduler.

הוספת משימה חדשה:



הגדות המשימה:

Create Task

General Triggers Actions Conditions Settings

Name: OnTime

Location:

Author: MBYDOMAIN\318343407

Description:

Security options

When running the task, use the following user account:
MBYDOMAIN\318343407 Change User...

☐ Run only when user is logged on

☒ Run whether user is logged on or not

☒ Do not store password. The task will only have access to local computer resources.

☒ Run with highest privileges

☐ Hidden

Configure for: Windows Vista™, Windows Server™ 2008

OK Cancel

הגדרת תזמון:

Create Task

General Triggers

When you create a task, choose the trigger that starts the task.

Trigger

Daily

New...

Edit Trigger

Begin the task: On a schedule

Settings

☐ One time

☒ Daily

☐ Weekly

☐ Monthly

Start: 07/05/2017 00:00:00 ☐ Synchronize across time zones

Recur every: 1 days

Advanced settings

☐ Delay task for up to (random delay): 1 hour

☐ Repeat task every: 1 hour for a duration of: 1 day

☐ Stop all running tasks at end of repetition duration

☒ Stop task if it runs longer than: 1 day

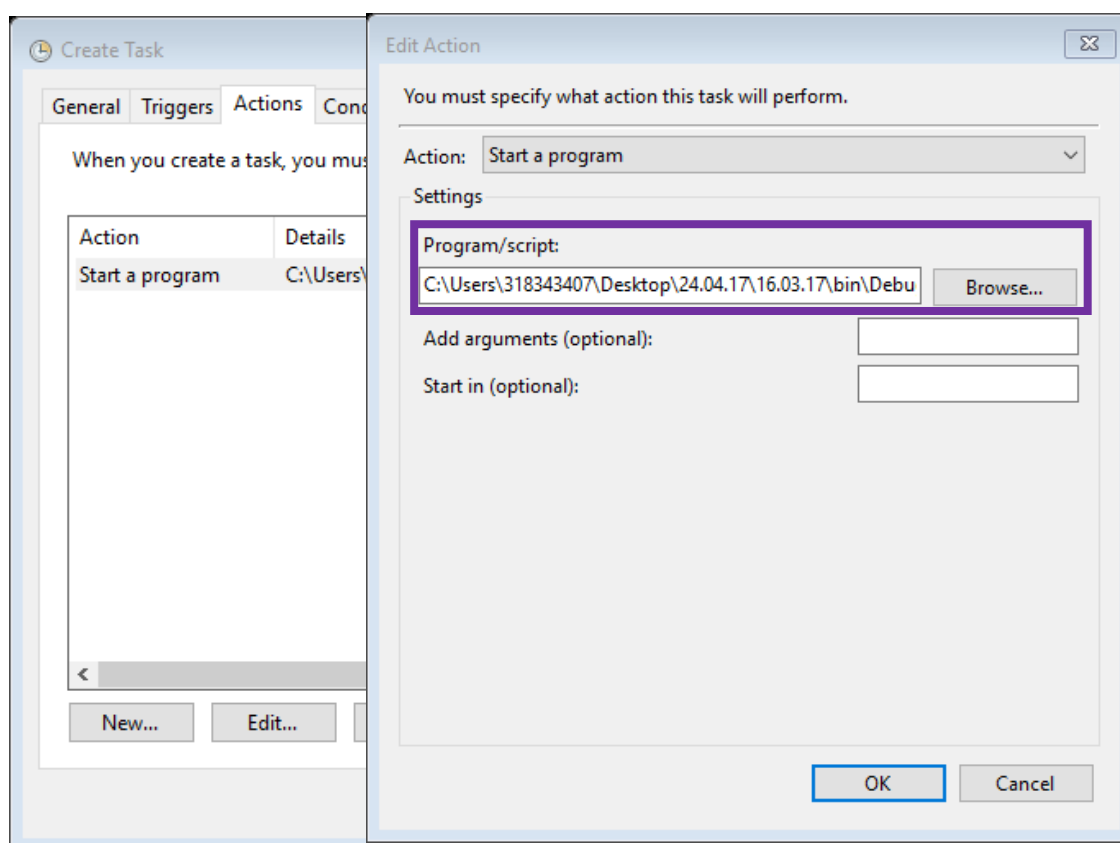
☐ Expire: 07/05/2018 10:04:45 ☐ Synchronize across time zones

☒ Enabled

OK Cancel



הגדרת פעולה:



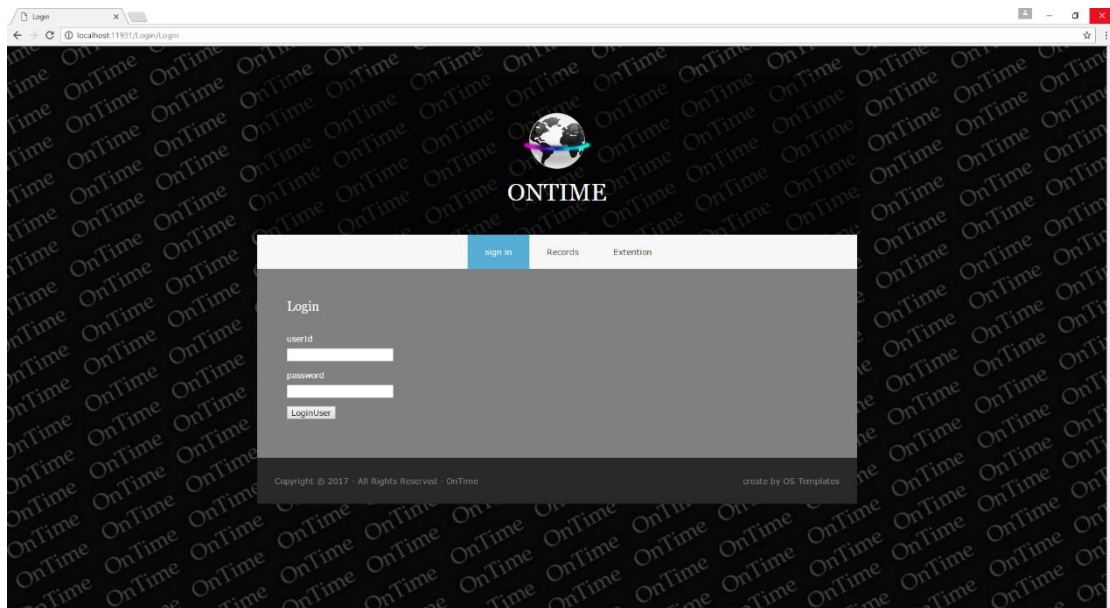
8 ממשקי המערכת ומדריך למשתמש

8.1 מדריך למשתמש באתר

-מדריך לא מושלם בכלל, מחכה לסיום עיצוב האתר

8.1.1 דף כניסה

בכניסה לאתר המשתמש יתבקש להזין שם משתמש וסיסמא.



משתמש חדש יתבקש להזין פרטים מלאים ופרטי תשלום



8.1.2 עריכת הבקשות של המשתמש

Request

Id	Name	date	download	
1	aa	1/ 1/1999	Details	X
2	bb	1/ 1/2017	Details	X
3	cc	1/ 1/2017	Details	X
16		4/ 5/2017	Details	X
18	shiri	4/ 5/2017	Details	X

Copyright © 2017 - All Rights Reserved - OnTime create by OS Templates

הוספת בקשה חדשה

CreateRequest

name

Select a mm file:

Choose File Unnamed Macro.htm

VERSION BUILD=844 RECORDS=OK

URL goto=file:///C:/Users/318343807/desktop/html.html

TAG POS=4 TYPE=INPUT:TEXT ATTR=ID:txt11 CONTENT=#00

TAG POS=5 TYPE=INPUT:TEXT ATTR=ID:txt11 CONTENT=www

TAG POS=4 TYPE=INPUT:NUMBER ATTR=ID:txt11 CONTENT=767

Create

Back to List

Copyright © 2017 - All Rights Reserved - OnTime create by OS Templates

פרטי בקשה

Request

id	Name	date	download
1	aa	1/ 1/1999	Download
2	bb	1/ 1/2017	Download
3	cc	1/ 1/2017	Download
16	dd	4/ 1/2017	Download
18	dd	4/ 1/2017	Download

userId
1

name
aa

date
01/01/1999 00:00:00

day
1

download
x

Frequency RequestId

day

Copyright © 2017 - All Rights Reserved - OnTime create by OS Templates

Download Extension 8.1.3 – הורדת התוסף

sign in Records Extention

Home

Welcom to On Time

Copyright © 2017 - All Rights Reserved - OnTime create by OS Templates


localhost:11931/Request/DownloadFiles

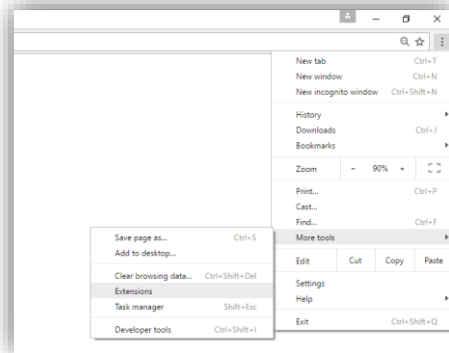
OnTime (3).zip

Show all

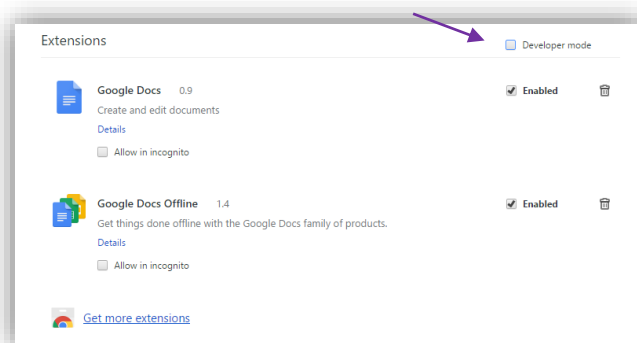
Google Chrome extension 8.2

8.2.1 מדריך התקנת Google Chrome extension

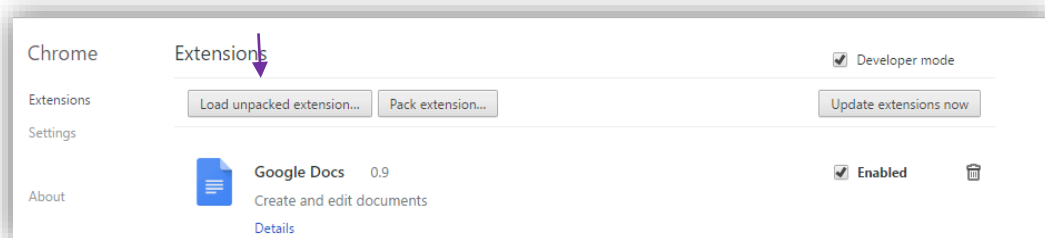
1. הכנס ל **chrome://extensions** בדפדפן שלך או בחר  בפינה הימנית של הדפדפן ולבחור **Extensions** תחת תפריט **Tools**.




2. וודא כי Developer mode מסומן.

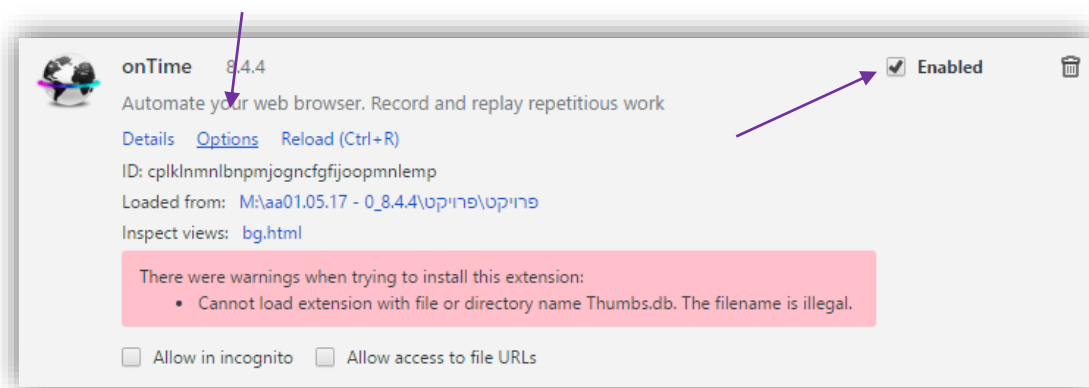


3. לחץ על **Load unpacked extension** ונתב לתיקיה בה התוסף שמור.

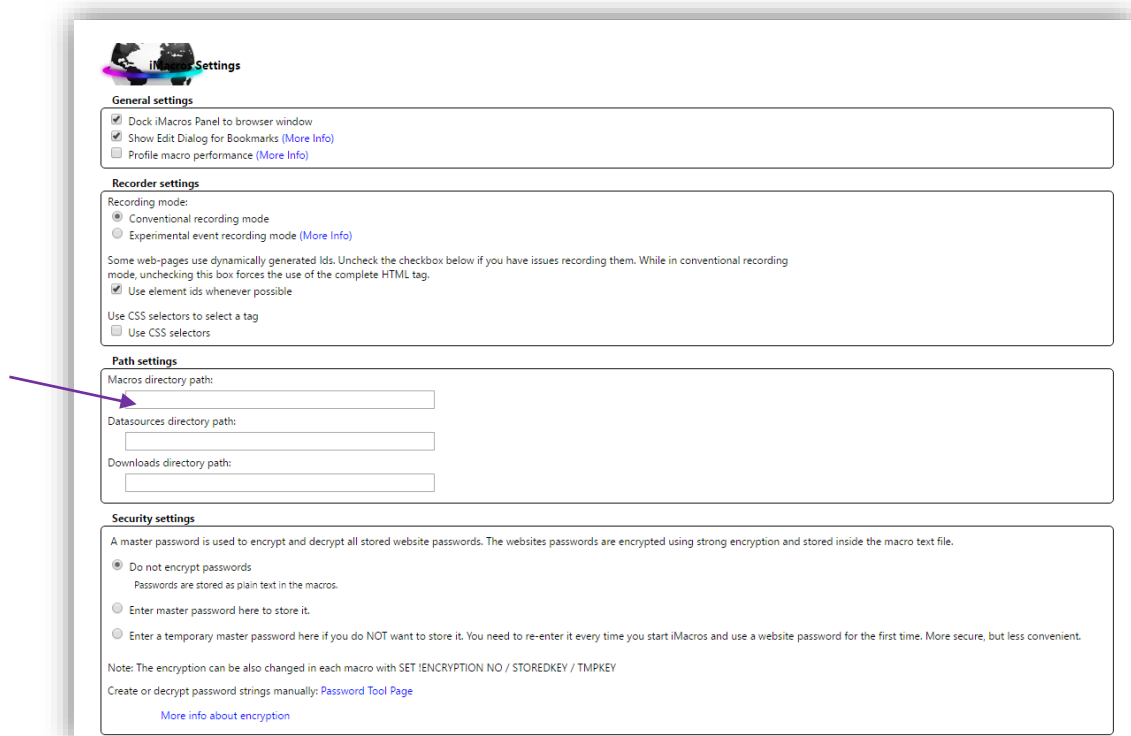


בשלב זה התוסף טעון ואמור להתווסף icon  לדפדפן, אם אינו מופיע יש לוודא כי התוסף מאופשר.





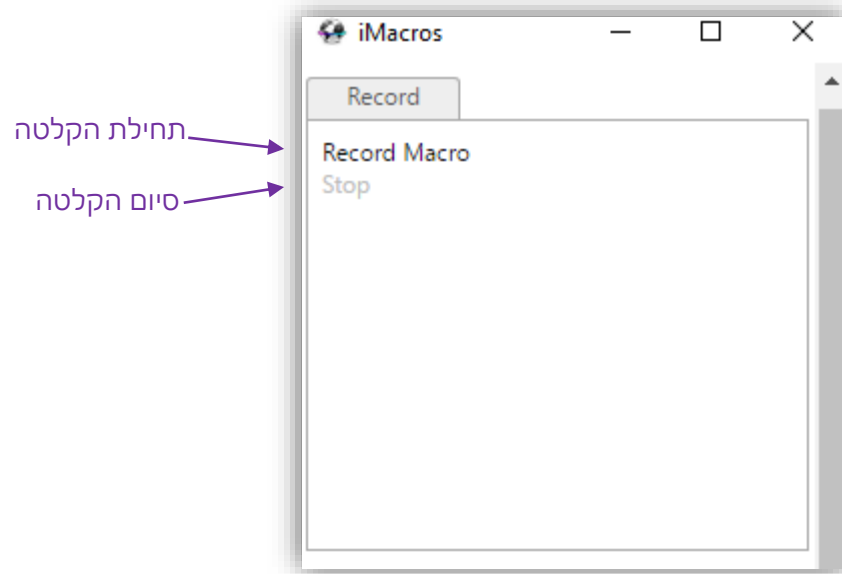
4. בחר ב Options והגדר ב Path settings את הנתונים בהם ישמרו ההקלטות.



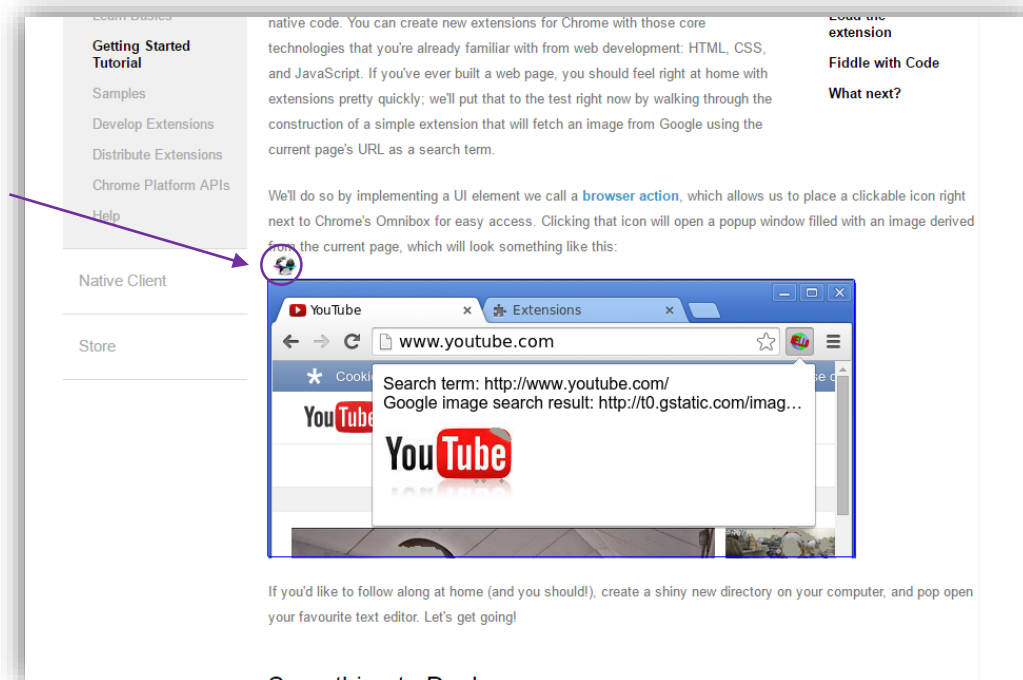
8.2.2 מדריך למשתמש בתוסף

הגישה לתוסף היא דרך icon  המופיע בחלק הימני של הדפדפן.

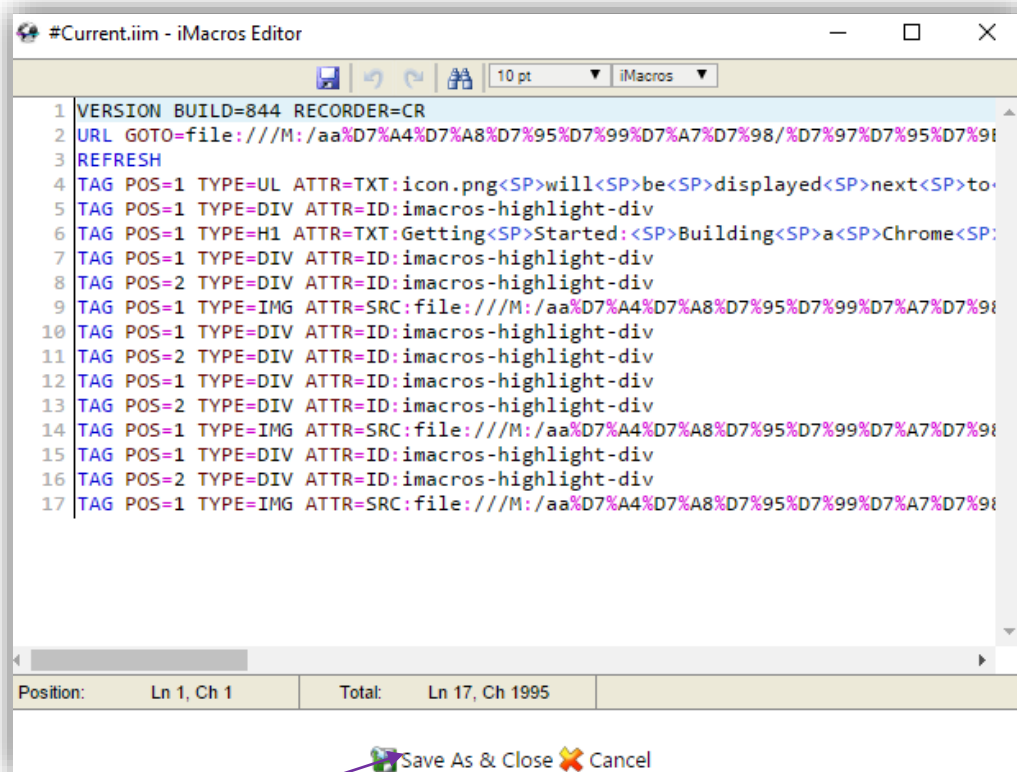
מסך ראשי



במהלך ההקלטה לאחר כל פעולה שמתבצעת בדפדפן מופיע icon המסמל כי הפעולה נקלטה.



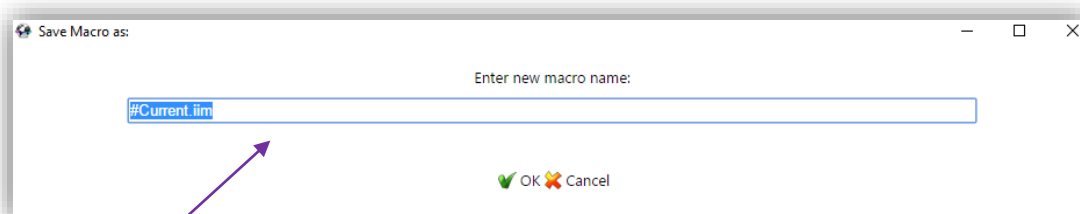
כאשר נמצא הדף המבוקש יש ללחוץ על Stop לסיום ההקלטה.
בסיום ההקלטה מוצגת ההקלטה להתרשמות ועריכה.



```
1 VERSION BUILD=844 RECORDER=CR
2 URL GOTO=file:///M:/aa%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98/%D7%97%D7%95%D7%98
3 REFRESH
4 TAG POS=1 TYPE=UL ATTR=TXT:icon.png<SP>will<SP>be<SP>displayed<SP>next<SP>to
5 TAG POS=1 TYPE=DIV ATTR=ID:imacros-highlight-div
6 TAG POS=1 TYPE=H1 ATTR=TXT:Getting<SP>Started:<SP>Building<SP>a<SP>Chrome<SP>
7 TAG POS=1 TYPE=DIV ATTR=ID:imacros-highlight-div
8 TAG POS=2 TYPE=DIV ATTR=ID:imacros-highlight-div
9 TAG POS=1 TYPE=IMG ATTR=SRC:file:///M:/aa%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98
10 TAG POS=1 TYPE=DIV ATTR=ID:imacros-highlight-div
11 TAG POS=2 TYPE=DIV ATTR=ID:imacros-highlight-div
12 TAG POS=1 TYPE=DIV ATTR=ID:imacros-highlight-div
13 TAG POS=2 TYPE=DIV ATTR=ID:imacros-highlight-div
14 TAG POS=1 TYPE=IMG ATTR=SRC:file:///M:/aa%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98
15 TAG POS=1 TYPE=DIV ATTR=ID:imacros-highlight-div
16 TAG POS=2 TYPE=DIV ATTR=ID:imacros-highlight-div
17 TAG POS=1 TYPE=IMG ATTR=SRC:file:///M:/aa%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98
```

שמירת ההקלטה

שמירה של ההקלטה- ההקלטה נשמרת בתקיה שהוגדרה כתיקיית יעד של ה-
.extension



שם ההקלטה

לאחר שההקלטה נשמרה, יש להעלות אותה לאתר ולהגדיר את התזמון שלה כך
שיוכל להתבצע מעקב אחר דף האינטרנט המבוקש.



9 מסקנות

תוך כדי הגדרת המערכת אותה נרצה לבנות, הבנו שלא נוכל להסתפק בכתובת URL אלא נצטרך להשתמש בגלישה רובוטית, מכיוון שחלק מהמידע בדף מופיע רק לאחר ביצוע שורת פעולות על הדף הראשי.

בשלב הראשון היה נראה לנו כי זוהי משימה בלתי אפשרית עבורנו כסטודנטיות, ניסינו לחקור את צורת העבודה של הדפדפן, למצוא דרך לתת הוראות לביצוע על אתרים ברשת.

לאחר עבודת מחקר מקיפה הגענו למסקנה כי בצורה כזאת נגיע לגמר רק עם גלישה רובוטית ולא נספיק לטפל בחלקים אחרים של המערכת. בשלב זה התוודענו למושג API-application program interface שמאפשר שימוש בממשק הרחבות קימות תוך התאמת השימוש לצרכים הרלוונטיים. בדקנו כמה כלים המממשים גלישה רובוטית כמו selenium, fiddler ובחרנו להשתמש בImacros שכתוב כקוד פתוח וניתן להתאמה מלאה. מתוך הניסיון הנ"ל למדנו שמערכת מורכבת לא משתלם לבנות מאפס מכיוון שניתן להשתמש בממשקי API יעילים שנכתבו ע"י צוותות של מתכנתים שיכולים לממש חלק מדרישות המערכת.

מלבד העובדה שכלים שכאלה יעילים ותקינים, שימוש בהם חוסך בזמן ומאפשר להתמקד בשדרוג חלקים אחרים של המערכת כך שאיכות התוצר תהיה מקסימלית.

מאחר והחלטנו כי לא נתמקד בישום גלישה רובוטית הייתה לנו אפשרות ליעל חלק אחר של המערכת. נקודת התורפה של המערכת הייתה השוואת גרסאות שהתבצעה ע"י מציאת מחרוזת LCS בצורה פשוטה, פעולה שדרשה מרחב זיכרון גדול וזמן ריצה ארוך מאד (דף HTML סטנדרטי רץ במחשב במשך כמה שעות). חיפשנו דרך חילופית אך הבנו שמציאת מחרוזת LCS תיתן תוצאה מדויק ביותר וכי חברות ותיקות ואמינות משוות אף הן בצורה כזו (כגון Google).

במקביל הכרנו כמה דרכים לממש מציאת מחרוזת LCS אך שום מימוש לא היה מושלם לגמרי. חלקם התמקדו בחסכון זמני ריצה (כגון HIRSCHBERG) וחלקם בחיסכון במקום בזיכרון (כגון תכנות דינאמי), הבנו כי לא תמיד מספיק להשתמש ברעיון בודד שלעיתים יפתור בעיה ספציפית ולא יתייחס ליעול מכלול המערכת, נוכל להיעזר ברעיונות ובכיווני חשיבה, אך את האלגוריתם הסופי נצטרך לבנות בעצמינו בשילוב עקרונות של כמה אלגוריתמים שונים. ואכן חקרנו את כל המימושים ביסודיות והצלחנו לחבר כמה אלגוריתמים שונים עד שקיבלנו תוצאה אופטימלית.



10 ביבליוגרפיה

he.wikipedia.org
developer.chrome.com/extension
stackoverflow.com
github.com
www.codeproject.com
internet-israel.com
wiki.imacros.net/iMacros_for_Chrome
www.seleniumhq.org
www.telerik.com
citeseerx.ist.psu.edu
www.columbia.edu
Msdn. Microsoft.Com
www.codechef.com
www.tutorialspoint.com
interactivepython.org
www.geeksforgeeks.org
csharp.net-informations.com
www.emailarchitect.net
www.aspsnippets.com
www.c-sharpcorner.com
en.wikibooks.org
robots.thoughtbot.com
www.ics.uci.edu
code.tutsplus.com
tutorialzine.com

