



מגישות: רות שנקולבסקי ורות ברוורמן

מגמת הנדסת תוכנה

מנחה: הגב' מ. שמעונוביץ

נובמבר 2019

TaskMe

פרויקט גמר לתואר הנדסאי מחשבים



מגישות: רות שנקולבסקי\_\_\_\_\_\_\_\_

ורות ברוורמן\_\_\_\_\_\_\_\_

שם המנחה: הגב' מ. שמעונוביץ

שם רכז המגמה: הגב' ח. ברגמן

נובמבר 2019

לכבוד: יחידת הפרויקטים מה״ט

**הצעה לפרויקט גמר**

**א. פרטי הסטודנטים**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שנת סיום הלימודים** | **טלפון נייד** | **כתובת** | **ת.ז. 9 ספרות** | **שם הסטודנט** |
| 2019 | 0548524231 | יהודה הנשיא 50 | 207815861 | רות ברוורמן |
| 2019 | 0527675554 | קושניר 21 | 208094268 | רות שנקולבסקי |

שם המכללה: סמינר וולף – שלוחת המכללה למנהל ראשל"צ

סמל המכללה: 72395

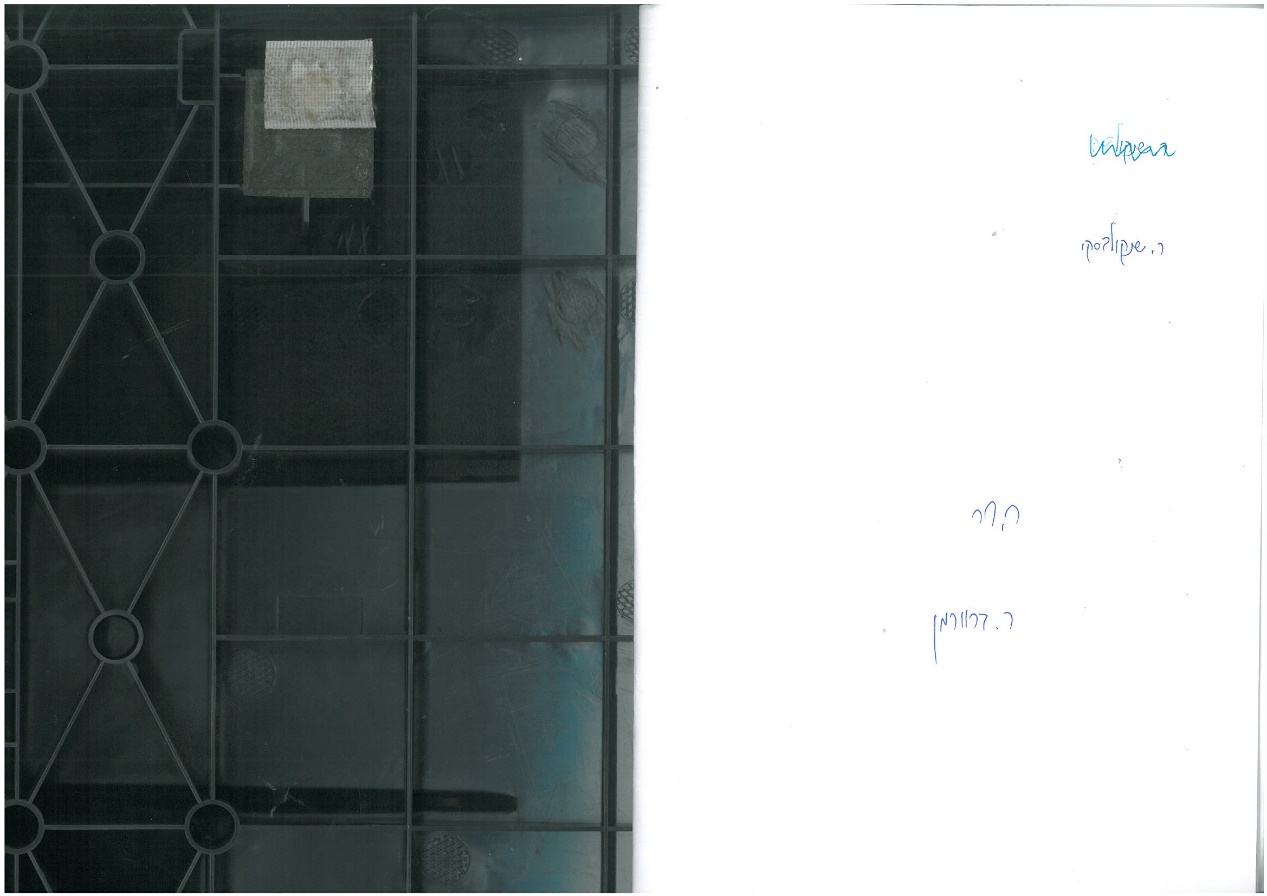
מסלול ההכשרה: הנדסאים

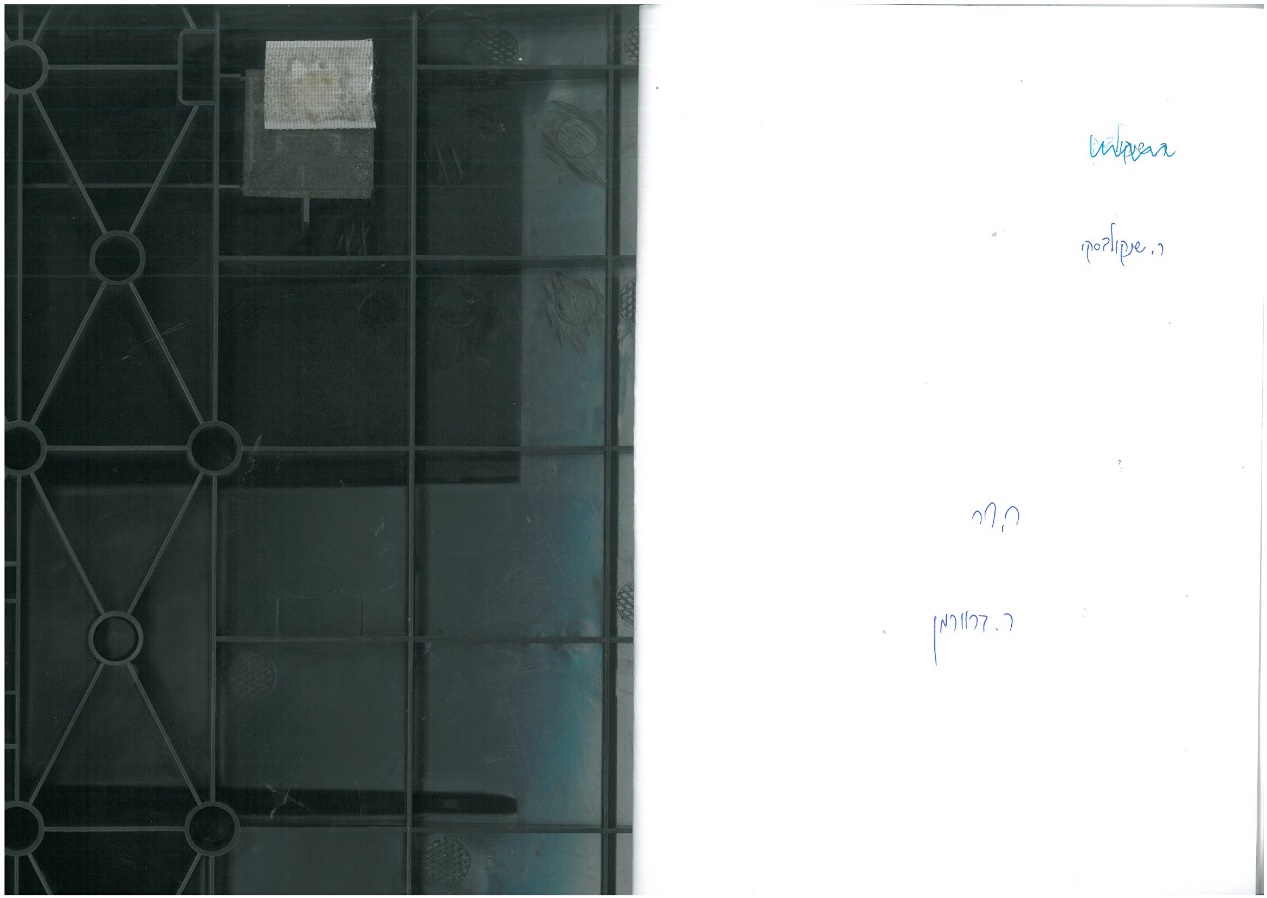
מגמת לימוד: תכנות מחשבים

מקום ביצוע הפרויקט: בסמינר ובבית

**ב. פרטי המנחה האישי**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מקום עבודה/תפקיד | תואר | טלפון נייד | כתובת | שם המנחה |
| המכללה למנהל | B.ED הנדסאי מחשבים | 052-7171295 | חזו"א 11 ב"ב | מ. שמעונוביץ |

 חתימת הסטודנט חתימת הסטודנט חתימת המנחה האישי



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

חתימת הגורם המקצועי מטעם מה״ט

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# שם הפרויקט : **TaskMe**

# רקע

## תיאור ורקע כללי

המערכת שאנו מתעדות לכתוב הינה תוכנה לניהול חברה לפיתוח תוכנה ומעקב אחרי תפקוד עובדי

החברה והפרויקטים המתנהלים בה. המערכת נותנת מעקב יעיל ונח דרך טיימר למדידת זמני עבודה

ודיאגרמות למדידת הזמן המוקצב מול תפוקת העובד, וכן מדידת כדאיות הפרויקטים. המערכת נותנת

התראות על אי מילוי משימה החל מיום לפני סיום הדד-ליין. התוכנה מספקת למשתמש ממשק תכנותי

נח.

## מטרות המערכת

* הקמת מערכת ידידותית ופשוטה לתפעול עבור חברה לפיתוח תוכנה.
* המערכת תבצע מעקב ומדידה של שעות עבודה ביחס לפרויקטים קיימים.
* לכל פרויקט תתאפשר מדידת רווחיות בכל שלב (כמה שעות עבודה בוצעו בכל פרויקט, מתוך השעות שהוגדרו לו, וכן כמה אחוזים מהעבודה הסופית בוצעו).
* הכלי יהיה גם כלי ניהולי שבו ניתן יהיה לראות את ניצול תפוקת העובד- כמה משימות מוגדרות לו, כמה שעות דיווח לפי חודש.
* המערכת תפיק דוחות ע"פ פרמטר של שם פרויקט/חודש/עובד.

# סקירת מצב קיים בשוק

קיימים אתרים העוסקים בניהול בית תוכנה. האתר שלנו מתייחד בכך שהוא מאפשר הצגת מדדים בזמן אמת,

מעקב יעיל, מדויק ונח אחר הנעשה בחברה. בנוסף שולח התראות לפני מועד סיום משימה במידה ולא הושלמה.

# מה הפרויקט אמור לחדש או לשפר

* דיאגרמות להצגת הנתונים בצורה מוחשית.
* טיימר למדידת זמן עבודה.
* שליחת התראות לפני מועד סיום המשימה.

# דרישות מערכת ופונקציונאליות

## דרישות מערכת, סביבת הטמעה ושימוש.

המערכת תעבור קומפילציה בסביבת Visual studio.

בצד השרת המערכת אמורה לרוץ בסביבת שרת אשר מריץ IIS לקבלת בקשות לתצוגת דפי אינטרנט.

המשתמש יוכל להריץ את האתר בכל מכשיר אשר מותקן עליו דפדפן אינטרנט.

## שרידות, ביצועים, התמודדות עם עומסים

* צד השרת מריץ את xpressE IIS מסוגל להתמודד עם מספר קריאות בו זמנית.
* הדף לא נטען מחדש עם כל שמירה של נתונים, אלא רק מתעדכן, וזה חוסך עומס גדול על השרת.

## דרישות פונקציונאליות

### מנהל -

#### כניסה למערכת - באמצעות שם משתמש וסיסמא וזיהוי כמנהל.

#### ניהול עובדים -

##### המערכת מאפשרת למנהל להגדיר עובד חדש.

##### המערכת מאפשרת למנהל לעדכן את פרטי העובד האישיים ושיוכו לראש צוות

##### מסוים, ע"י שליפת עובד מרשימת עובדים.

##### המערכת מאפשרת למנהל לשנות סטטוס עובד מפעיל ללא פעיל ולהפך, ע"י

##### שליפת עובד מרשימת עובדים.

##### המערכת מאפשרת למנהל לצפות בכל פרטי העובדים.

#### ניהול פרויקטים -

##### המערכת מאפשרת למנהל להגדיר נתוני פרויקט חדש הכוללים הגדרת שעות

##### פיתוח, שעות בדיקה, שעות עיצוב ושיוך ראש צוות לפרויקט.

##### המערכת מאפשרת למנהל לעדכן שעות בפרויקט ע"י שליפת פרויקט מרשימת

##### פרויקטים, עדכון השעות כולל: שעות פיתוח, שעות בדיקה ושעות עיצוב.

##### המערכת מאפשרת למנהל לצפות ברשימת הפרויקטים.

##### המערכת מאפשרת למנהל לשנות סטטוס סיום פרויקט מ-false ל-true ע"י

##### שליפת פרויקט מרשימת פרויקטים.

#### הפקת דוחות -

##### המערכת מאפשרת למנהל להפיק דו"ח הספקים לפי סיווג של פרויקט ו/או עובד

##### ו/או תאריך.

##### המערכת מאפשרת למנהל לשמור את הדוחות כקובץ טבלאי.

##### המערכת מציגה למנהל גרפים על תפוקת העובדים.

##### המערכת מציגה למנהל גרפים על מצב הפרויקטים.

### ראש צוות –

#### כניסה למערכת - באמצעות שם משתמש וסיסמא וזיהוי כראש צוות.

#### המערכת מאפשרת לראש צוות לצפות ברשימת העובדים שתחתיו.

#### המערכת מאפשרת לראש צוות לעדכן שעות עבודה לעובדים שתחתיו ע"י שליפת עובד

#### מרשימת עובדים בפרויקט מסוים.

#### המערכת מאפשרת לראש צוות לצפות ברשימת הפרויקטים שתחתיו.

#### המערכת מאפשרת לראש צוות להגדיר משימה בפרויקט לעובד שתחתיו.

##### המערכת תבצע שליפת משימה מרשימת משימות.

##### המערכת תבצע שליפת עובד מרשימת עובדים שבחסותו.

#### המערכת מציגה לראש צוות גרפים על תפוקת עובד שתחתיו באופן כללי או בפרויקט

#### מסוים.

#### המערכת מציגה לראש צוות גרפים על התקדמות הפרויקטים.

### עובד –

#### כניסה למערכת - באמצעות שם משתמש וסיסמא וזיהוי כעובד.

#### המערכת מאפשרת לעובד לבצע משימה במשך זמן, ולכן בכל פעם שנכנס לביצוע

#### משימה הוא מעדכן תחילת ביצוע כך ששעון המערכת יתחיל למנות זמן ביצוע המשימה.

#### המערכת מאפשרת לעובד לעצור ביצוע משימה גם באופן זמני ע"י עדכון מונה השעון.

#### המערכת מאפשרת לעובד לצפות בתרשים יחס הזמן בין הזמן שהוגדר למשימה לבין זמן

#### התקדמות העבודה.

#### המערכת תזהה הודעת שעון ותשלח התראת סיום פרויקט לעובד.

### שעון המערכת –

#### המערכת תפעיל את שעון המערכת למנית משך זמן עבודה על משימה ע"פ עדכון עובד, ותסיים את המשימה ע"פ עדכון עובד.

#### המערכת מקבלת את תאריך יום המחר ותתריע את העובדים המתאימים על פרויקט שאמור להסתיים מחר.

# בעיות צפויות במהלך הפיתוח

## הבעיות:

### בעיה 1:

### מחיקת ראש צוות – לכל ראש צוות משויכים עובדים אשר נמצאים תחתיו.

המנהל עלול לנסות למחוק ראש צוות אשר משויכים אליו עובדים ללא סידור העובדים תחת ראש

צוות חדש.

### בעיה 2:

### מחיקת עובד המקושר לפרויקט – לכל עובד משויכות משימות מהפרויקטים. עובד אשר

### משויך לכמה משימות והמנהל ינסה למחוק אותו ללא שיוך המשימות לעובד אחר.

## פתרונות אפשריים:

### לבעיה 1:

### פתרון 1: חסימת מחיקת ראש צוות.

פתרון 2: הודעה האומרת שאין אפשרות למחוק ראש צוות לפני שיוך העובדים תחתיו לראש צוות שונה ואז שיוך העובדים ומחיקת ראש הצוות.

### לבעיה 2:

### פתרון 1: העברת המשימות למנהל הפרויקט.

## הפתרון הנבחר עבור כל אחת מהבעיות:

### לבעיה 1: פתרון 2.

### לבעיה 2: פתרון 1.

# פתרון טכנולוגי נבחר

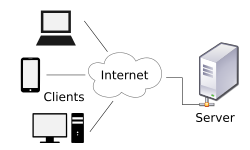
## טופולוגית הפתרון

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

מסד נתונים – DB's של Sql-server

ממשק משתמש בצד הלקוח: דפדפן אינטרנט כלשהו: Chrom, Firefox , explorer Internet

## דיאגרמה



## טכנולוגיות בשימוש:

כתיבת צד הלקוח: Angular + TypeScript .

כתיבת צד השרת: שירות אינטרנט באמצעות WEB-API .

מסד נתונים באמצעות Sql-server.

## שפות הפיתוח:

## 

### בצד השרת:

C# - WebApi

### בצד הלקוח:

### Angular, TypeScript, Html, Css

## תיאור הארכיטקטורה הנבחרת

צד לקוח: אנגולר משום שהיא טכנולוגיה מתקדמת המבוקשת מאד בתעשייה.

היא מונחית עצמים, מובנית, נכתבת בצורה ממודרת ומאורגנת. קל לאתר ולתקן בה שגיאות.

מצאנו לנכון שכדאי להתמקצע בה.

צד שרת: השתמשנו בשפת C# שהיא שפה נוחה לשימוש המפתח.

## חלוקה לתכניות ומודולים:

### בצד השרת:

התוכנית נבנתה בצורת חשיבה של מודל שלושת השכבות.

התוכנית חולקה לשלוש שכבות כאשר כל שכבה מטפלת בתחום שונה ומכילה כמה מחלקות. בתוך כל מחלקה הוגדרו משתנים ואובייקטים מתאימים המיוחדים למחלקה. כל מחלקה מטפלת בתחום מסוים ומתקשרת למחלקה המתאימה לה. צורת כתיבה זו נועדה להקל על המתכנת הן בכתיבת הקוד והן בתחזוקה השוטפת של הקוד. כמובן שבין המחלקות שבשכבות השונות קיים קשר, וקיימת התייחסות מתוך מחלקה אחת לאובייקטים המוגדרים במחלקות אחרות, אך הכל נעשה תוך שמירה על בטחון ואבטחת הנתונים והפונקציות.

1. DAL – Data Access Layer - שכבה זו מורכבת ממקור נתונים –מסד הנתונים שלנו וממערכת התוכנה Entity-Framework אשר תפקידה לקרוא את המידע הנדרש למערכת, לשמור את העדכונים, ולהוסיף מידע חדש או למחוק פרטי מידע קיימים.
2. DTO - בשכבה זו נגדיר את הישויות של הפרויקט אל מול מסד הנתונים.
3. BLL – Business Logic Layer – השכבה שאמונה על הלוגיקה של המערכת, עוסקת בעיבוד המידע, בחישובים שונים ושליחתו לשכבת התצוגה.

בשכבה זו נמשש את הפונקציונאליות של המערכת.

1. GUI – Graphic User Interface – שכבת ממשק המשתמש – שכבה המקשרת בין צד שרת לצד לקוח.

### בצד הלקוח:

Component - חלוקה למסכים.

Service – מחלקה המספקת מידע לשאר המחלקות.

Model - ישויות.

Module – יבוא קבצים והגדרת מערכת.

# סביבת השרת

## ממשק המשתמש/לקוח – GUI

ממשק המשתמש יתקבל כ- HTML ויהיה מוצג באמצעות דפדפן אינטרנט .

## ממשקים למערכות אחרות / API

התממשקות לשרת הgmail לשליחת מיילים של google.

## שימוש בחבילות תוכנה

* Entity Framework
* BootStrap

# מבני נתונים וארגון קבצים

## שיטת האחסון

דיסק קשיח ודיסקים לגיבוי.

## מבני הנתונים

## מנגנוני התאוששות מנפילה/ קריסה/ תמיכה בטראנזקציות.

במקרה של קריסת המערכת IIS מועלה באופן אוטומטי.

המערכת מסתמכת על מנגנון הטראנזקציות של SQL-SERVER המנוהל ללא התערבות המשתמש.

# תרשימי מערכת מרכזיים

## Use Case

**מנהל**

נכנס למערכת

מוסיף עובד חדש

מוחק עובד

מעדכן פרטי עובד

מסיים פרויקט

מוסיף פרויקט חדש

מעדכן שעות בפרויקט

צופה בפרטי העובדים

צופה בפרטי הפרויקטים

צופה בתרשים מצב פרויקט

שמירה כקובץ טבלאי

צופה בגרפים על תפוקת העובדים

שליפת עובד מרשימת עובדים

שליפת פרויקט מרשימת פרויקטים

שליפת עובד מרשימת עובדים

מפיק דו"ח

בחירת פרויקט

בחירת עובד

ציון תאריך

<include>

<include>

<include>

<include>

<include>

<extend>

<extend>

<extend>

<extend>

<extend>

<extend>

<include>

<include>

# 

צופה ברשימת העובדים שתחתיו

צופה ברשימת הפרויקטים שתחתיו

<include>

**ראש צוות**

נכנס למערכת

מגדיר משימה לעובד

שליפת עובד מרשימת עובדים

שליפת משימה מרשימת משימות

שליפת פרויקט מרשימת פרויקטים

שליפת עובד מרשימת עובדים

צופה בתרשימי הספק עבודה לעובד

מעדכן משך זמן משימה לעובד

צופה בהתקדמות פרויקט

מקבל התראת סיום פרויקט

<include>

<include>

<include>

<extend>

<include>

<include>

<include>

מעדכן עצירת משימה

<include>

מעדכן תחילת ביצוע משימה

שליפת משימה מרשימת אישית

**עובד**

נכנס למערכת

מקבל התראת סיום פרויקט

צופה בתרשים יחס הזמן למשימה

# ­

**שעון המערכת**

מונה משך זמן משימה

מחזיר תאריך יום המחר לצורך התראת סיום פרויקט

# תיאור המרכיב האלגוריתמי – חישובי

## איזה בעיה בא לפתור, איך יפתור?

## גניבת סיסמא ממשתמש ונסיון שימוש בה –הצפנת הסיסמא עוד לפני שליחתו לשרת ושימוש בטוקן

## לאורך כל זמן השימוש בפרויקט.

## איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)

## דיאגרמות לפי עובד/פרויקט/חודש, דוחות עם אפשרות יצוא לאקסל.

# תיאור/התייחסות לנושאי אבטחת מידע

אבטחת נתונים גבוהה – הצפנת הקוד לפני שליחתו לשרת ושליחתו רק באופן המאובטח.

החזרת טוקן לכל משתמש ולא שימוש בסיסמת המשתמש כמזהה משתמש.

# משאבים הנדרשים לפרויקט:

## מספר שעות המוקדש לפרויקט: 720

## חלוקת עבודה בין חברי הצוות: עבודת צוות

## ציוד נדרש: מחשב

## תוכנות נדרשות: Visual Studio, Visual Studio Code

## ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט: לא רלוונטי.

## ספרות ומקורות מידע

[https://stackoverflow.com](https://stackoverflow.com/)

[https://www.w3schools.com](https://www.w3schools.com/)

<https://angular-ui.github.io>

<https://www.asp.net/web-api>

https://getbootstrap.com

# תכנית עבודה ושלבים למימוש הפרויקט:

|  |  |  |
| --- | --- | --- |
| **שלב** | **משך זמן** | **תאריכים משוערים** |
| * + ייזום הרעיון | כשבועיים | אפריל 2019 |
| * + ניתוח מערכת | כשבוע | אפריל 2019 |
| * + ניתוח מבנה נתונים | כשבוע | מאי 2019 |
| * + אפיון UX - UI | כשבועיים | מאי 2019 |
| * + כתיבת הלוגיקה העסקית | כחודשיים | יולי 2019 |
| * + כתיבת ממשק המשתמש | כחודש | אוגוסט 2019 |
| * + עיצוב | כחודש | אוגוסט 2019 |

# תכנון הבדיקות שיבוצעו

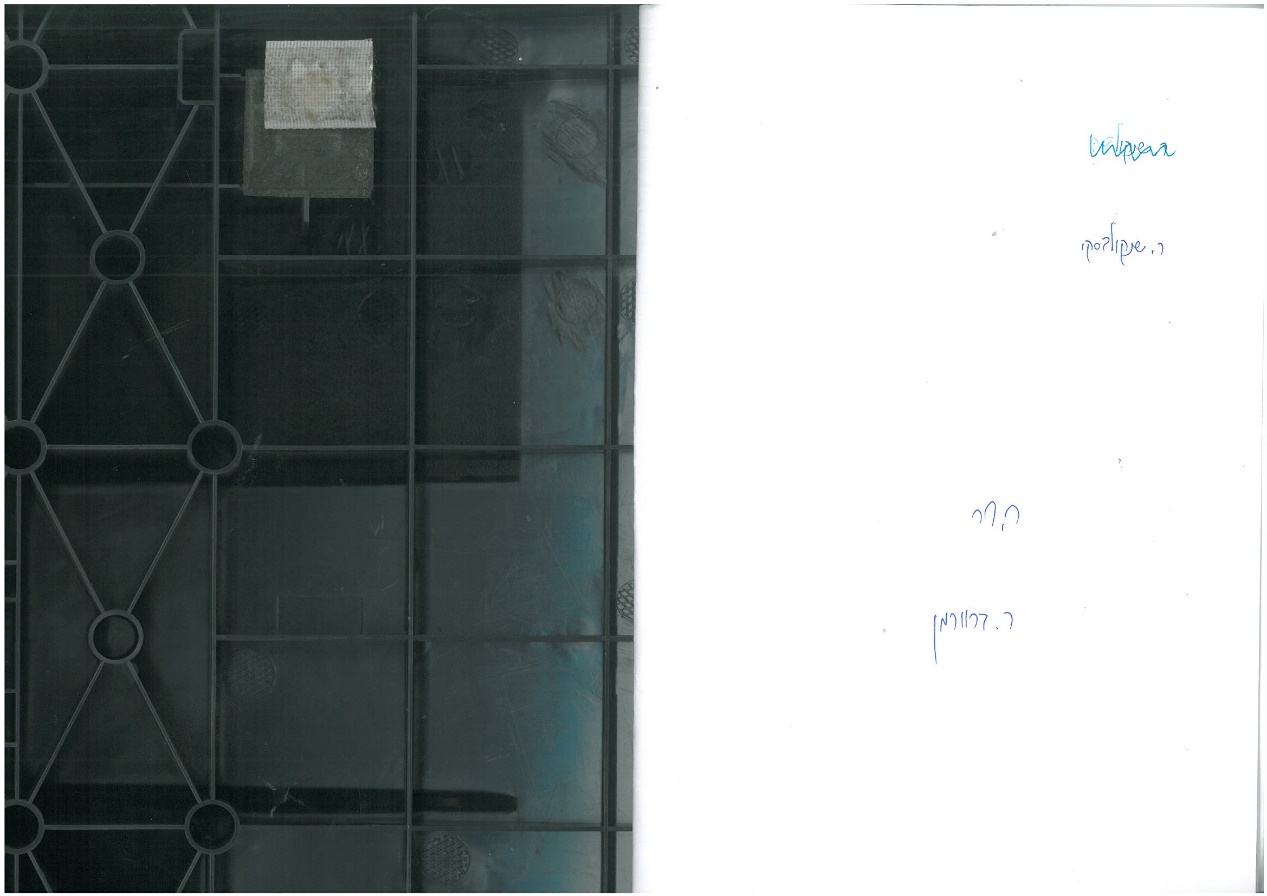
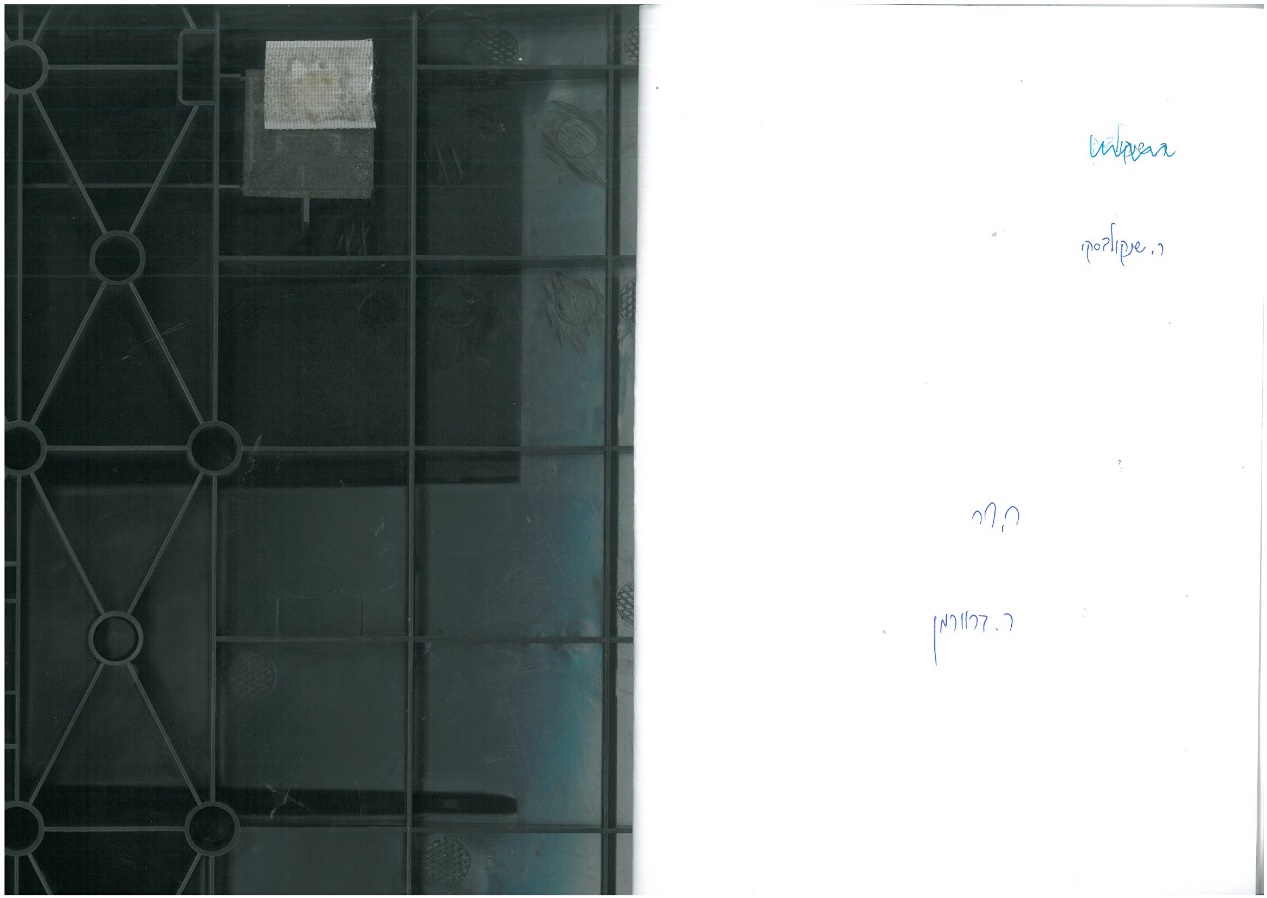
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **מספר בדיקה** | **מס` דרישה במסמך אפיון** | **מקרי הבדיקה** | **ידנית/ אוטומטית** | **חשיבות** | **הערות** |
| 1 | 5.3.1.1  5.3.2.1  5.3.3.1 | בדיקת כניסה למערכת באמצעות שם משתמש וסיסמא קיימים. | ידנית | גבוהה |  |
| 2 | 5.3.1.1 | בדיקה שכניסה למערכת באמצעות שם משתמש וסיסמא של משתמש מנהל מאפשרת פעולות השייכות למנהל. | ידנית | גבוהה |  |
| 3 | 5.3.1.2.1. | בעת יצירת משתמש חדש ע"י המנהל בדיקה שהסיסמא לא קיימת. | ידנית | גבוהה |  |
| 4 | 5.3.1.2.1. | לאחר יצירת משתמש חדש ע"י המנהל בדיקה שהמשתמש אכן נרשם בטבלת משתמשים. | ידנית | גבוהה |  |
| 5 | 5.3.1.2.2 | לאחר עדכון פרטי משתמש ע"י מנהל בדיקה שפרטי המשתמש אכן עודכנו בטבלת משתמשים. | ידנית | גבוהה |  |
| 6 | 5.3.1.2.3 | לאחר שינוי סטטוס עובד בדיקה שהשדה השתנה לסטטוס הרצוי. | ידנית | גבוהה |  |
| 7 | 5.3.1.3.1 | בעת הוספת פרויקט חדש ע"י המנהל בדיקה שהפרויקט אכן נרשם בטבלת פרויקטים. | ידנית | גבוהה |  |
| 8 | 5.3.1.3.2 | לאחר עדכון שעות לפרויקט ע"י המנהל בדיקה שהשעות אכן עודכנו בטבלת פרויקטים. | ידנית | בינונית |  |
| 9 | 5.3.1.3.4 | לאחר שינוי סטטוס סיום פרויקט בדיקה שהשדה השתנה לtrue. | ידנית | גבוהה |  |
| 10 | 5.3.1.4.1 | בדיקה שהמערכת מפיקה את הדו"ח ע"פ הפרמטר הרצוי. | ידנית | בינונית |  |
| 11 | 5.3.1.4.2 | בדיקה שהדו"ח נשמר כקובץ טבלאי. | ידנית | בינונית |  |
| 12 | 5.32.1 | בדיקה שכניסה למערכת באמצעות שם משתמש וסיסמא של ראש צוות מאפשרת פעולות השייכות לראש צוות. | ידנית | גבוהה |  |
| 13 | 5.3.2.3 | לאחר עדכון שעות עבודה לעובד בדיקה שהשעות אכן עודכנו. | ידנית | גבוהה |  |
| 14 | 5.3.3.1 | בדיקה שכניסה למערכת באמצעות שם משתמש וסיסמא של עובד מאפשרת פעולות השייכות לעובד. | ידנית | גבוהה |  |
| 15 | 5.3.3.2 | בעת לחיצת עובד על התחלת משימה בדיקה שהטיימר מתחיל לפעול. | ידנית | גבוהה |  |
| 16 | 5.3.3.3 | בעת לחיצת עובד על עצירת משימה בדיקה שהטיימר מפסיק לפעול. | ידנית | גבוהה |  |
| 17 | 5.3.4.3 | בדיקה שהתראות על סיום פרויקט נשלחו לעובדים המתאימים. | ידנית | בינונית |  |

# בדיקות יחידה (unit test)

לא רלוונטי.

# בקרת גרסאות (version control)

כרגע לא רלוונטי. במידה ויהיה רלוונטי הבדיקה תבוצע ע"י .GIT



הערות הגורם המקצועי מטעם מה"ט:

אישור הגורם המקצועי מטעם מה"ט:

שם:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**חתימת הסטודנט חתימת המנחה האישי**

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

הערות ראש המגמה במכללה:

אישור ראש המגמה במכללה:

שם:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**תודות**

בראש ובראשונה נודה לבורא העולם "החונן לאדם דעת" שהביאנו עד הלום.

למשפחותינו היקרות שהיו איתנו לאורך כל הדרך, עודדו ותמכו ובעיקר – האמינו בנו עוד מההתחלה.

לאמהות היקרות שלנו שוויתרו לנו על שעות של עזרה בבית כדי שהפרויקט יהיה מושלם יותר...

לצוות "סמינר בית יעקב ע"ש הרב וולף" ובראשם מרכזת המגמה הגב' חנה ברגמן שנווטה אותנו במסירות ובהשקעה ללא גבול.

לתמר שנקולבסקי, המנחה הצמודה שלוותה אותנו לאורך כל הפרויקט בסבלנות וכשרון רב. עודדה, הדריכה וענתה לשאלותינו בכל עת, ועזרה לנו להגיע לתוצר המושלם.

ולחברותינו היקרות שהתענינו ועזרו ופרגנו מכל הלב על התקדמות והצלחה.

על זאת ועוד תודותינו

רות ורות

**הצהרה**

הצהרת התלמידים/ות

אנו,

רות שנקולבסקי - ת.ז 208094268

ורות ברוורמן - ת.ז 207815861

החתומים/ות מטה מצהירות בזאת כי כל הפרויקט המוגש בספר זה,

הינו פרי עבודתנו בלבד על בסיס הנחיתה של המנחה תמר שנקולבסקי ותוך הסתמכות על לימודינו במכללה והמידע אשר רכשנו תוך כדי העבודה.

חתימת התלמידים/ות: רות שנקולבסקי ורות ברוורמן

חתימת המנחה:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**תקציר**

ניהול חברה עם עובדים רבים ופרויקטים שונים עלול לעורר קשיים בנוגע לסדרי התנהלות ונתינת פקודות והרשאות לעובדים. ניהול לא תקין וכושל יביא חורבן על החברה. משימות יתעופפו בלי קבלת אחריות של אי מי עליהן, הרשאות ינתנו ללא שימת לב ודגש ועובדים ישארו חסרי מענה בכל תקלה או בעיה בלי אפשרות לקבל מענה.

המנהל לא יודע אלו פרויקטים כדאיים לו, אלו עובדים לא שווה לו להעסיקם, ובכלל – מה מצב הפרויקטים לעת עתה – מה הצפי שהוא צריך לומר ללקוחות.

לעומת זאת, ניהול תקין, מבטיח ויעיל ינהל את כלל העובדים ויתן לכולם להתרכז בעיקר - פיתוח וקידום החברה, סיום משימות ופרויקטים במהירות, דבר היביא לנסיקת החברה לגבהים.

האתר שלנו מבטיח ניהול תקין ויעיל תוך שימת דגש על חווית משתמש.

האתר מנהל את כל הצד הטכני בחברה.

כל משתמש הנכנס לאתר יוצגו לו המסכים המתאימים לו לפי דרגתו בחברה –

* מנהל –
* מסכי ניהול עובדים –

כולל הוספה, מחיקה, עריכה והוספת/ הורדת הרשאות לעובדים לפרויקטים מסוימים.

* מסכי ניהול פרויקטים –

כולל הוספת חדשים ועריכת הקיימים. בנוסף מוצגים הפרויקטים בצורת דוחות על מנת להקל על המנהל במעקב אחר העובדים והפרויקטים הקיימים עם אפשרות להדפיס לקובץ אקסל. המנהל יכול לפלטר אחר עובד מסוים/חודש מסוים ועוד.

* מסכי ניהול צוותים –

מעקב וניהול ראשי הצוותים והעובדים תחתם.

* ראש צוות –
* מסכי מעקב אחר הפרויקטים שתחתיו.
* אפשרות להוספת שעות עבודה לעובדים תחתיו.
* תצוגת גרפים של עובדים ופרויקטים עם אפשרות תצוגה לפי עובד/פרויקט מסוים. יש אפשרות להדפיס את הגרף או לייצא כתמונה.
* עובד –

במסך העובד מוצגים כל המשימות עליהן העובד צריך לעבוד. העובד יכול להתחיל משימה ע"י לחיצה על כפתור התחלת המשימה, זמן העבודה נספר ונסכם לעובד. בנוסף, מוצג לעובד גרף של כל משימותיו עם כל השעות הנצרכות לכל פרויקט לעומת השעות שכבר עבד עליהן.

**מפתח עניינים**

**תקציר...........................................................................................................................................17**

**מדריך למתכנת.....................................................................................................................20 1.**

**טכנולוגיות אסטרטגיות ...............................................................................................20 1.1**

**1.2 תיאור מבנה הפרויקט.....................................................................................................21**

**1.3 עקרונות התכנון..............................................................................................................24**

**1.3.1 עקרונות תאורטיים.................................................................................................24**

**1.3.2 עקרונות תכנותים...................................................................................................26**

**1.4 תרשימים..........................................................................................................................27**

**1.4.1 עץ תהליכים.............................................................................................................27**

**28……………………………………………………………..Use-Case1.4.2 תרשים**

**1.4.3 תרשים מראה מחלקות..........................................................................................30**

**1.5 תרשים נתונים מאוחסנים.............................................................................................31**

**............................................................................................................31SQL 1.5.1 תרשים**

**1.6 תוכן הפרויקט..................................................................................................................32**

**...........................................................32Entities1.6.1 תיאור המחלקות – מחלקות ב**

**........................................................................................37Controllers 1.6.2 תיאור ה-**

**2. טכנולוגיות.......................................................................................................................................57**

**2.1 אנגולר...............................................................................................................................57**

**..................................................................................................................58TypeScript 2.2**

**..................................................................................................59VisualStudio2017 2.3**

**3. מדריך למשתמש....................................................................................................................60**

**3.1 הוראות כלליות לשימוש באתר....................................................................................60**

**3.2 מסכים...............................................................................................................................61**

**3.2.1 מסכי מנהל...............................................................................................................62**

**3.2.2 מסכי ראש צוות.......................................................................................................68**

**3.2.3 מסכי עובד................................................................................................................71**

**4. סיכום ומסקנות......................................................................................................................73**

**5. נספחים....................................................................................................................................74**

**6. ביבליוגרפיה...........................................................................................................................75**

**1. מדריך** **למתכנת:**

**1.1 טכנולוגיות אסטרטגיות:**

**Server side** לצורך כתיבת צד השרת –

כאשר הפרויקט האחראי על זרימת הנתונים נכתב בטכנולוגית , C# השתמשנו בשפת

.asp .net web api

**Client side** לצורך כתיבת צד הלקוח –

.8 גרסה **Framework angular**השתמשנו ב

. **Ajax** נעשית באמצעות Server ל - Clientהתקשורת בין ה -

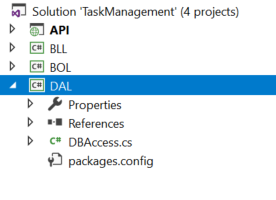
.visual studio 2017הפיתוח התבצע בסביבת

.(המונחים המודגשים מוסברים בפרק טכנולוגיות סעיף 2)

**1.2 תיאור מבנה הפרויקט**

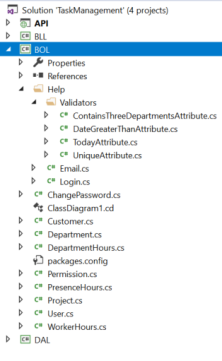
**Server side 1.2.1** מבנה ה-

השרת נבנה בהתאם למודל השכבות וכולל את הפרויקטים הבאים:



DAL – Data Access Layer א.

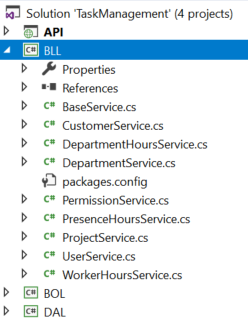
פרויקט זה משמש כשמו – כאחראי על המידע, כלומר על מסד הנתונים. פרויקט זה מתחבר למסד הנדרש לפרויקט.



BOL – Bussines Object Layer ב.

מכיל את המודלים המקבילים לישויות מסד הנתונים ומשמש מעטפת

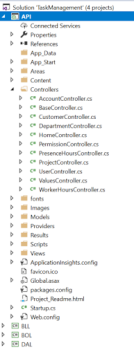
.DALלאבטחת הגישה לשכבת ה



BLL – Bussines Logic Layer ג.

פרויקט זה מכיל את עיקר הלוגיקה של צד השרת,

ומטפל בעיקר בשליפות ושמירת הנתונים.

,מתממשק האתר עמו API - חשיפת web api ד.

כגון: התחברות משתמש, הוספת משתמש, שליפת נתונים , ועוד.

**Client** **side**מבנה ה- **1.2.2**

Angular Framework מורכב מ- Client מבנה ה-

פותח ע"י גוגל. הרעיון המונח מאחורי אנגולר- רעיון הקומפונטטות.

.TypeScript ומדף CSS , מדף HTML קומפוננטה היא למעשה יחידה לוגית המורכבת מדף

.Css3ומעוצב ע"י Html5 ממשק משתמש נכתב ע"י

אחראיים על העיצוב והמראה הכללי, ודפי ה- CSS מהווים את התצוגה, דפי ה-HTML דפי

לצד כוללים בתוכם פונקציות לוגיות שונות, התחברות לשרתים ולספריות, והתחברותTypeScript

.Ajaxהשרת ע"י קריאות

**1.3 עקרונות התכנון**

**1.3.1 עקרונות תאורטיים**

**הורשה:**

בין(Is-as)בהורשה משתמשים בתכנות מונחה - עצמים, ירושה היא דרך לבסס יחס "סוג - של"

המימוש הנפוץ של הגדרה זו היא בעזרת מחלקות. מחלקה יכולה לקבל בירושה תכונות עצמים.

והתנהגות של מחלקת – האם שלה (נקראת גם "מחלקת-על", "מחלקת אב" או מחלקת בסיס).

היחסים בין מחלקות מגדירים היררכיה של מחלקות. מנגנון הירושה מסייע במידול של תחומים בדרך דומה לזאת המוגדרת באופן "טבעי" על ידי העוסקים בהם . לדוגמה: אם מרובע הוא סוג-של (או מקרה פרטי של) מצולע, ניתן להגדיר יחס ירושה בין המחלקה מרובע למחלקה מצולע. על פי רוב, אובייקט של המחלקה היורשת (מרובע) יתמוך בכל הפעולות המוגדרות על מצולע כלשהו, ובנוסף יתמוך בפעולות נוספות, המיוחדות למרובע. בדומה, ניתן להגדיר יחס ירושה בין המחלקה ריבוע למחלקה מרובע. שימושים :

ירושה מאפשרת להשיג מספר מטרות :

1. שימוש חוזר בקוד. בעזרת ירושה ניתן לכתוב מחלקות בעלות טווח התנהגות נרחב, תוך כתיבת קטעי קוד קטנים יחסית שמרחיבים מחלקה קיימת .

2. מידול של העולם האמתי באופן דומה לזה המוגדר על ידי האדם. למשל, ניתן לכתוב היררכיה של מחלקות המקבילה לטקסונומיה המקובלת בביולוגיה (יען היא סוג של ציפור, האדם הוא סוג של יונק, וכן הלאה).

3. פולימורפיזם (רב צורתיות) של זמן-ריצה. משתנה מטיפוס (סטטי) מסוים יכול להתנהג בצורות שונות, לפי הטיפוס של האובייקט שהוא מתייחס אליו בזמן ריצה. למשל משתנה מהמחלקה **בעל חיים** שנדרש לבצע פעולה מסוג "השמע קול" יבצע פעולות שונות לחלוטין במקרה שהאובייקט הוא מהמחלקה **כלב** או **חתול** . ניתן לבצע זאת גם ללא ירושה (בעזרת פקודות תנאי), אך ירושה מאפשרת מימוש מודולרי, גמיש וקל להרחבה .

בפרויקט השתמשנו בהורשה:

, **BaseService** בפרויקט יורשים מה- **services** לדוגמא: כל ה -

public class UserService : BaseService

{

public int Login(string email, string password)

{

...

}

...

}

**הפרדת שכבות**

פרויקט צד השרת מופרד לשכבות כפי שתואר לעיל כך שכל שכבה אחראית על רכיב מסוים בודד.

מהו מודל הפרדת השכבות ?

תבנית עיצוב בסיסית שמגדירה הפרדת האפליקציה לשכבות:

שכבת נתונים, שכבת לוגיקה ושכבת ממשק משתמש.

(ככל שהתכנה מורכבת יותר ייתכנו שכבות נוספות)

**למה זה טוב?**

**תחזוקה:**

ניתן להחליף או לתקן מימוש פנימי של שכבה אחת בארכיטקטורה בלי לשנות שכבה אחרת.

**נוחות פיתוח:**

אדם אחד עובד על רכיב בתכנה, אדם אחר עובד על רכיב אחר, כל עוד שהחתימות זהות ניתן לשלב כוחות ולייעל זמני פיתוח.

**בדיקות:**

תקלה כלשהיא מבודדת בכל שכבה בנפרד, לדוגמא אם לא קיבלנו רשימת נתונים לתצוגה נבדוק קודם כל את שלב הלוגיקה – שלב הביניים. אם הנתונים שם תקינים נדע בוודאות שגם בשכבת

.UI הנתונים תקינים וכל שנותר הוא לפתור את התקלה ברמת ה- DALה -

**שימוש חוזר:**

UI אהובים עלינו ועובדים היטב, ניתן להחליף את פלטפורמת הBLL+DAL נניח שהרכיבים

לטכנולוגיה אחרת במינימום מאמץ.

**אבטחה:**

שלא לשכבה מסוימת בלי לחשוף מבני נתונים, לוגיקה או נתוניםInterface-API נוח יותר להגדיר

רלוונטי למשתמש, כמו גם חסימה בפני האקרים ברמות שונות.

**Server**

בצד השרת מתייחסים לפעולות המבוצעות ע"י השרת במערכת יחסי לקוח-שרת ברשת מחשבים.

בדרך כלל, שרת היא תכנית מחשב, כגון שרת אינטרנט, שפועל על שרת מרוחק, נגיש מהמחשב המקומי של משתמש או תחנת העבודה. פעילות יכולה להתבצע בצד השרת כי הם דורשים גישה למידע או פונקציונלי שאינו זמין בלקוח, או לדרוש התנהגות אופיינית שאינה אמינה כאשר היא נעשית בצד הלקוח.

פעולות בצד השרת כוללות גם עיבוד ואחסון של נתונים מלקוח לשרת, אשר ניתן לצפות על ידי קבוצה של לקוחות.

**Client**

בצד הלקוח מתייחסים לפעולות המבוצעות על ידי הלקוח ביחסי לקוח-שרת ברשת מחשבים. בדרך כלל, הלקוח הוא יישום מחשב, כגון דפדפן אינטרנט, שפועל על המחשב המקומי של משתמש או תחנת עבודה ומתחבר לשרת כנדרש. פעילות יכולה להתבצע בצד הלקוח כי היא דורשת גישה למידע או פונקציונלי שנגיש בלקוח, אך לא בשרת, כי המשתמש צריך לצפות בהם או לספק קלט, או מכיוון שהשרת חסר כוח העיבוד כדי לבצע את הפעולות במועד לכל הלקוחות שהוא משרת. בנוסף, אם ניתן לבצע פעולות על ידי הלקוח, בלי לשלוח נתונים ברשת, הם יכולים לקחת פחות זמן, להשתמש בפחות רוחב פס, וכרוך בסיכון ביטחוני פחות.

משתמשים יכולים ,FTP או HTTP כאשר השרת מספק נתונים באופן נפוץ, למשל על פי פרוטוקולי

להיות בחירתם של מספר תכניות לקוח (רוב הדפדפנים המודרניים יכולים לבקש ולקבל נתונים באמצעות שני פרוטוקולים אלה). במקרה של יישומים מיוחדים, מתכנתים יכולים לכתוב פרוטוקול שלהם לשרת, ללקוח ולתקשורת, שניתן להשתמש בם רק אחד עם השני.

תכניות הפועלות על המחשב המקומי של משתמש מבלי שולח או מקבלים נתונים דרך רשת אינן לקוחות נחשבים, ופעולות בצד הלקוח כך הפעולות של תכניות כאלה לא תחשבנה.

**עקרונות תכנותיים 1.3.2**

**LocalStorge**

הוא אובייקט שמאחסן נתונים מקומית במחשב הלקוח.

.string מסוג key-value צורת השמירה נעשית באמצעות

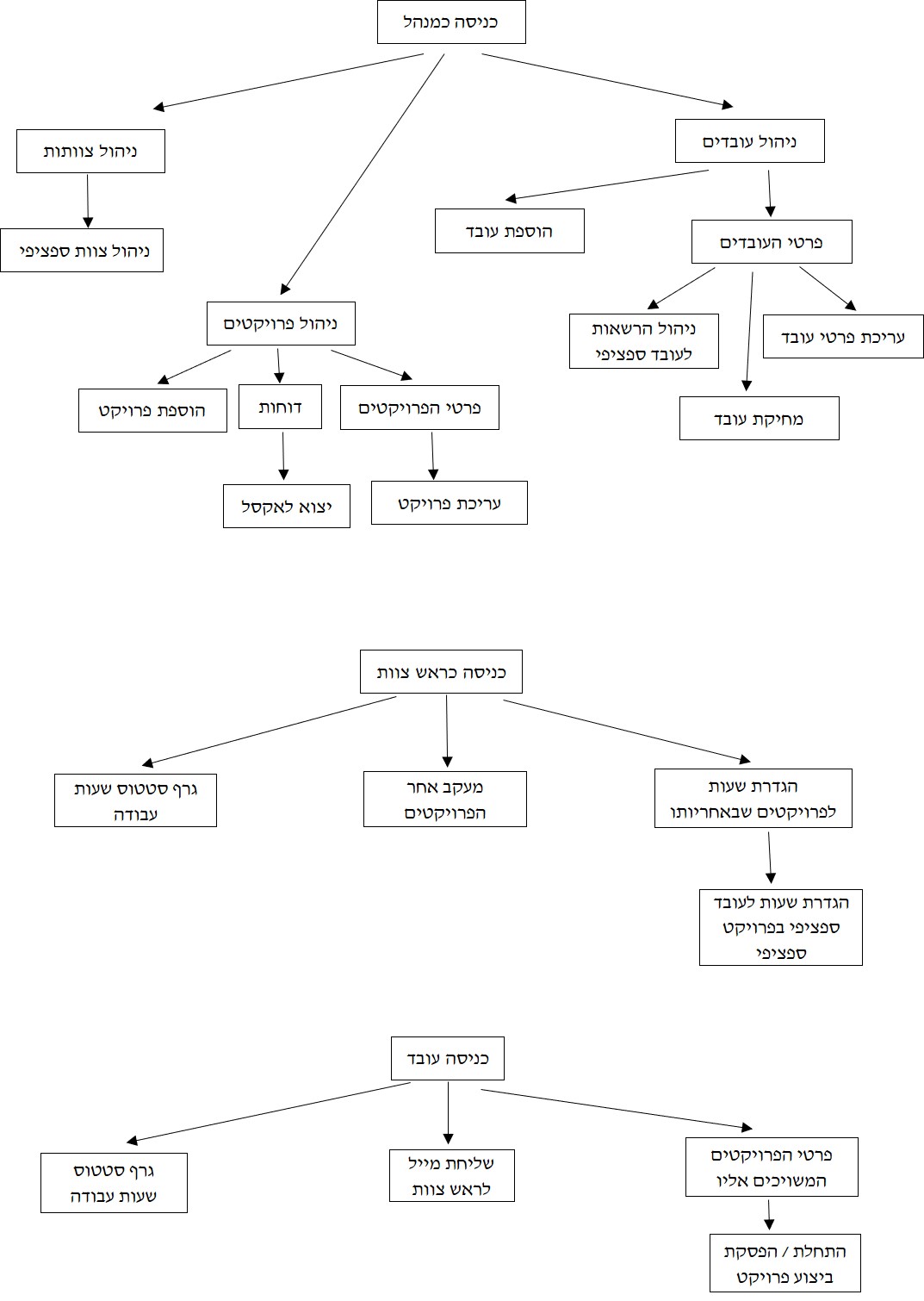
ישארו גם לאחר סגירת הדפדפן. localStorge אורך חיי הנתונים המאוחסנים ב-

localStorage.setItem(Global.USER,JSON.stringify(user));

this.userId = JSON.parse(localStorage.getItem(Global.USER)).userId;

* 1. **תרשימים**:

**1.4.1 עץ תהליכים:**



**1.4.2 תרשים Use-Case:**

**מנהל**

נכנס למערכת

מוסיף עובד חדש

מוחק עובד

מעדכן פרטי עובד

מסיים פרויקט

מוסיף פרויקט חדש

מעדכן שעות בפרויקט

צופה בפרטי העובדים

צופה בפרטי הפרויקטים

צופה בתרשים מצב פרויקט

שמירה כקובץ טבלאי

צופה בגרפים על תפוקת העובדים

שליפת עובד מרשימת עובדים

שליפת פרויקט מרשימת פרויקטים

שליפת עובד מרשימת עובדים

מפיק דו"ח

בחירת פרויקט

בחירת עובד

ציון תאריך

<include>

<include>

<include>

<include>

<include>

<extend>

<extend>

<extend>

<extend>

<extend>

<extend>

<include>

**ראש צוות**

נכנס למערכת

מגדיר משימה לעובד

שליפת עובד מרשימת עובדים

שליפת משימה מרשימת משימות

שליפת פרויקט מרשימת פרויקטים

שליפת עובד מרשימת עובדים

צופה בתרשימי הספק עבודה לעובד

מעדכן משך זמן משימה לעובד

צופה בהתקדמות פרויקט

מקבל התראת סיום פרויקט

<include>

<include>

<include>

<extend>

<include>

<include>

<include>

מעדכן עצירת משימה

<include>

מעדכן תחילת ביצוע משימה

שליפת משימה מרשימת אישית

**עובד**

נכנס למערכת

מקבל התראת סיום פרויקט

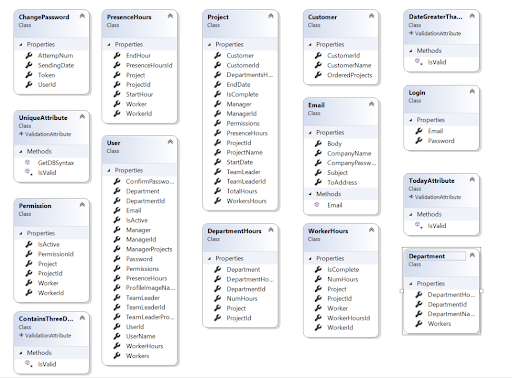
צופה בתרשים יחס הזמן למשימה

**שעון המערכת**

מונה משך זמן משימה

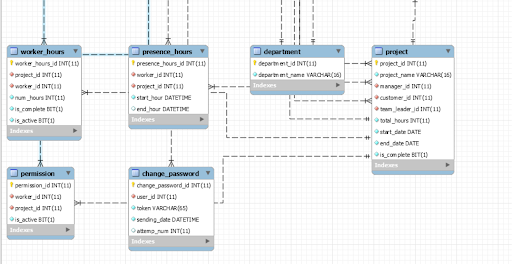
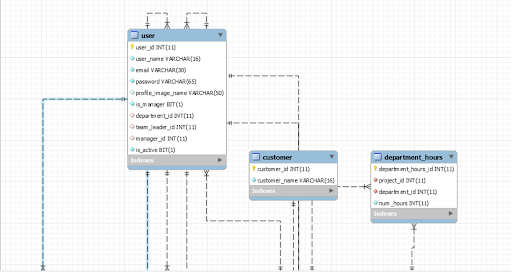
מחזיר תאריך יום המחר לצורך התראת סיום פרויקט

**1.4.3 תרשים מראה מחלקות**



* 1. **תרשים נתונים מאוחסנים**

**1.5.1** **תרשים** SQL:



**1.6 תוכן הפרויקט**

**1.6.1 תיאור המחלקות - מחלקות ב-Entities**

**User - הוספת משתמש חדש:**

public class User

{

[Key]

public int UserId { get; set; }

[Required(ErrorMessage = "Required field")]

[StringValidator(MaxLength = 15, MinLength = 2)]

[RegularExpression(@"[A-Za-z0-9]+",

ErrorMessage = "User name can contain only letters")]

public string UserName { get; set; }

[Required(ErrorMessage = "Required field")]

[Unique]

[EmailAddress(ErrorMessage = "Email is not valid")]

public string Email { get; set; }

[Required(ErrorMessage = "Required field")]

[Unique]

[RegularExpression(@"[A-Za-z0-9]+",

ErrorMessage = "Password can contain only letters and numbers")]

[MinLength(64, ErrorMessage = "Password is not valid")]

[MaxLength(64, ErrorMessage = "Password is not valid")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required(ErrorMessage = "Required field")]

[DataType(DataType.Password)]

[Compare("Password", ErrorMessage =

"The password and confirmation password do not match")]

[NotMapped]

public string ConfirmPassword { get; set; }

public string ProfileImageName { get; set; }

[ForeignKey("Department")]

public int? DepartmentId { get; set; }

[ForeignKey("TeamLeader")]

public int? TeamLeaderId { get; set; }

[ForeignKey("Manager")]

public int? ManagerId { get; set; }

public bool IsActive { get; set; }

}

**Project - הוספת פרויקט חדש:**

public class Project

{

[Key]

public int ProjectId { get; set; }

[Required(ErrorMessage = "Required field")]

[Unique]

[StringValidator(MaxLength = 15, MinLength = 2)]

[RegularExpression(@"[A-Za-z0-9]+",

ErrorMessage = "Project name can contain only letters")]

public string ProjectName { get; set; }

[ForeignKey("Manager")]

public int ManagerId { get; set; }

[ForeignKey("Customer")]

public int CustomerId { get; set; }

[ForeignKey("TeamLeader")]

public int TeamLeaderId { get; set; }

[Range(1, int.MaxValue, ErrorMessage = "TotalHours must be above 0")]

public int TotalHours { get; set; }

[Required(ErrorMessage = "Required field")]

[Column(TypeName = "Date")]

[DateGreaterThan]

public DateTime StartDate { get; set; }

[Required(ErrorMessage = "Required field")]

[Column(TypeName = "Date")]

[DateGreaterThan]

public DateTime EndDate { get; set; }

public bool IsComplete { get; set; }

}

**Department - המחלקות בחברה:**

public class Department

{

[Key]

public int DepartmentId { get; set; }

[Required(ErrorMessage = "Required field")]

[StringValidator(MaxLength = 15, MinLength = 2)]

[RegularExpression(@"[A-Za-z]+",

ErrorMessage = "Customer name can contain only letters")]

public string DepartmentName { get; set; }

}

**Customer - הוספת לקוח חדש:**

public class Customer

{

[Key]

public int CustomerId { get; set; }

[Required(ErrorMessage = "Required field")]

[StringValidator(MaxLength = 15, MinLength = 2)]

[RegularExpression(@"[A-Za-z]+",

ErrorMessage = "Customer name can contain only letters")]

public string CustomerName { get; set; }

}

**WorkerHours- שעות עבודה המוקצבות לעובד:**

public class WorkerHours

{

[Key]

public int WorkerHoursId { get; set; }

[ForeignKey("Project")]

public int ProjectId { get; set; }

[ForeignKey("Worker")]

public int WorkerId { get; set; }

[Range(0, int.MaxValue, ErrorMessage = "'NumHours' must be above 0")]

public int NumHours { get; set; }

public bool IsComplete { get; set; }

}

**PresenceHours- שעות נוכחות לעובד:**

public class PresenceHours

{

[Key]

public int PresenceHoursId { get; set; }

[ForeignKey("Worker")]

public int WorkerId { get; set; }

[ForeignKey("Project")]

public int ProjectId { get; set; }

[Required(ErrorMessage = "Required field")]

[Column(TypeName = "Date")]

[Today]

public DateTime StartHour { get; set; }

[Column(TypeName = "Date")]

[Today]

public DateTime? EndHour { get; set; }

}

**Permission- הרשאות:**

public class Permission

{

[Key]

public int PermissionId { get; set; }

[ForeignKey("Worker")]

public int WorkerId { get; set; }

[ForeignKey("Project")]

public int ProjectId { get; set; }

public bool IsActive { get; set; }

}

**DepartmentHours – שעות מוקצבות לעבודה לכל פרויקט לפי מחלקות:**

public class DepartmentHours

{

[Key]

public int DepartmentHoursId { get; set; }

[ForeignKey("Project")]

public int ProjectId { get; set; }

[ForeignKey("Department")]

public int DepartmentId { get; set; }

[Range(0, int.MaxValue, ErrorMessage = "'NumHours' must be above 0")]

}

**ChangePassword– החלפת סיסמא:**

public class ChangePassword

{

public int UserId { get; set; }

public string Token { get; set; }

public DateTime SendingDate { get; set; }

public int AttempNum { get; set; }

}

**1.6.2 Controllers:**

**- UserService**

כניסה למערכת -

public static User Login(string email, string password)

{

try

{

string query = $"{GetSimpleUsersQuery()} WHERE

email = '{email}' AND password = '{password}' AND is\_active=1;";

Func<MySqlDataReader, User> func = (reader) =>

{

User user = null;

if (reader.Read())

{

user = BaseService.InitUser(reader);

user.Password = string.Empty;

}

return user;

};

return DBAccess.RunReader(query, func);

}

catch (Exception ex)

{

throw ex;

}

}

קבלת כל העובדים לפי ראש צוות –

public static List<User> GetAllTeamUsers(int teamLeaderId)

{

try

{

string query = $"{GetUsersQuery()} AND u.team\_leader\_id = {teamLeaderId};";

return GetAllUsers(query);

}

catch (Exception ex)

{

throw ex;

}

}

קבלת כל ראשי הצוותות -

public static List<User> GetAllTeamLeaders(int managerId)

{

try

{

string query = $"{GetUsersQuery()} AND u.team\_leader\_id IS NULL AND u.manager\_id={managerId};";

List<User> teamLeaders = GetUsers(query);

return teamLeaders;

}

catch (Exception ex)

{

throw ex;

}

}

קבלת עובד מסוים -

public static User GetUserById(int userId)

{

try

{

string query = $"{GetUsersQuery()} AND u.user\_id={userId};";

Func<MySqlDataReader, User> InitUser = (reader) =>

{

User user = null;

if (reader.Read())

{

user = BaseService.InitUser(reader);

user.Password = string.Empty;

}

return user;

};

User foundUser = DBAccess.RunReader(query, InitUser);

if (foundUser != null)

{

foundUser.Permissions = PermissionService.GetPermissions(foundUser.UserId);

}

return foundUser;

}

catch (Exception ex)

{

throw ex;

}

}

הוספת עובד חדש -

public static bool AddUser(User user)

{

try

{

string query = "INSERT INTO task\_management.user" +

"(user\_name, email, password, profile\_image\_name, department\_id, team\_leader\_id,manager\_id) " +

$"VALUES('{user.UserName}', '{user.Email}', '{user.Password}', " +$"{BaseService.GetStringValueOrNull(user.ProfileImageName)}, " +

$"{BaseService.GetIntValueOrNull(user.DepartmentId)}, " +

$"{BaseService.GetIntValueOrNull(user.TeamLeaderId)}, " +

$"{BaseService.GetIntValueOrNull(user.ManagerId)});" +

"SELECT @@IDENTITY;";

object userId = DBAccess.RunScalar(query);

bool created;

if (userId != null)

{

created = true;

user.UserId = Convert.ToInt32(userId);

WorkerHoursService.AddWorkerHoursToTeamProjects(user);

}

else

{

created = false;

}

return created;

}

catch (Exception ex)

{

throw ex;

}

}

עריכת עובד –

public static bool EditUser(User user)

{

try

{

User oldUser = GetUserById(user.UserId);

string query = "UPDATE task\_management.user " +

$"SET user\_name='{user.UserName}'," +

$"email='{user.Email}'," +

$"profile\_img\_name={BaseService.GetStringValueOrNull(user.ProfileImgName)}," +$"department\_id={BaseService.GetIntValueOrNull(user.DepartmentId)}," +

$"team\_leader\_id={BaseService.GetIntValueOrNull(user.TeamLeaderId)} " +

$"where user\_id={user.UserId};";

bool edited = DBAccess.RunNonQuery(query) == 1;

if (edited)

{

//manage worker hours to team-projects if the worker moves team

if (oldUser.TeamLeaderId != user.TeamLeaderId && user.TeamLeaderId != null)

{

PermissionService.DeleteUnnecessaryPermissions(user);

WorkerHoursService.AddWorkerHoursToTeamProjects(user);

}

}

return edited;

}

catch (Exception ex)

{

throw ex;

}

}

מחיקת עובד –

public static bool DeleteUser(int userId)

{

try

{

string query = $"UPDATE task\_management.user SET is\_active=0 where user\_id={userId} AND is\_active=1;";

bool deleted = DBAccess.RunNonQuery(query) == 1;

if (deleted)

{

PermissionService.GetPermissions(userId).ForEach(

permission =>{

PermissionService.DeletePermission(

permission.PermissionId); });

}

return deleted;

}

catch (Exception ex)

{

throw ex;

}

}

שליחת מייל מהעובד לממונה עליו–

public static bool SendEmail(Email email, User user)

{

try

{

User manager = GetUserById((int)user.ManagerId);

email.ToAddress.Add(manager.Email);

email.Body += $"\nFrom {user.UserName}";

return BaseService.SendEmail(email);

}

catch (Exception ex)

{

throw ex;

}

}

שליחת מייל במקרה של שכיחת סיסמא –

public static bool ForgotPassword(string email)

{

User user = GetUserByEmail(email);

if (user == null)

return false;

string myToken = GenerateRandomNo();

bool created = AddChangePassword(user.UserId, myToken);

if (created)

{

Email myEmail = new Email()

{

Subject = "A verification code has been sent to you",

Body = $"Please enter the following verification code:{myToken}" +

$" The code is only valid for ten minutes.",

};

myEmail.ToAddress.Add(user.Email);

BaseService.SendEmail(myEmail);

return true;

}

return false;

}

בדיקת אמיתות הסיסמא אם אפשרות של ניסיון של עד 3 פעמים לצורך אבטחת הסיסמא–

public static bool ConfirmToken(ChangePassword changePassword)

{

try

{

string query = "SELECT COUNT(\*) FROM task\_management. change\_password " +

$"WHERE user\_id = {changePassword.UserId} AND token = '{changePassword.Token}' AND attemp\_num<3;";

int count = Convert.ToInt32(DBAccess.RunScalar(query));

if (count > 0)

{

return true;

}

query = "UPDATE task\_management.change\_password " +

$"SET attemp\_num=attemp\_num+1 WHERE user\_id ={changePassword.UserId};";

DBAccess.RunNonQuery(query);

return false;

}

catch (Exception ex)

{

throw ex;

}

}

שינוי סיסמא –

public static bool ChangePassword(User user)

{

try

{

string query = $"UPDATE task\_management.user SET password='{user.Password}' where user\_id={user.UserId};";

bool edited = DBAccess.RunNonQuery(query) == 1;

return edited;

}

catch (Exception ex)

{

throw ex;

}

}

קבלת כל העובדים לפי פרויקט ולפי מחלקה –

public static List<User> GetAllUsers(int departmentId, int projectId)

{

try

{

string query = $"{GetUsersQuery()} AND u.department\_id={departmentId} AND u.user\_id in " +

$"(SELECT worker\_id FROM task\_management.worker\_hours WHERE project\_id={projectId})";

List<User> userList = GetAllUsers(query);

return userList;

}

catch (Exception ex)

{

throw ex;

}

}

מחיקת הסיסמא החילופית לאחר המתנה של 10 דקות לצורך אבטחה–

public static async Task DeleteToken()

{

try

{

string query = $"delete from task\_management.change\_password where sending\_date < DATE\_SUB(NOW(), INTERVAL 10 MINUTE); ";

DBAccess.RunNonQuery(query);

}

catch (Exception ex)

{

throw ex;

}

}

שאילתה בסיסית של שליפת עובד מרשימת העובדים –

private static string GetSimpleUsersQuery()

{

string query = "SELECT \* FROM task\_management.user";

return query;

}

קבלת כל העובדים –

private static List<User> GetAllUsers(string query)

{

try

{

List<User> userList = GetUsers(query);

//join permission of project for each user

userList.ForEach(user =>

{

user.Permissions = PermissionService.GetPermissions(user.UserId);

});

return userList;

}

catch (Exception ex)

{

throw ex;

}

}

**ProjectService**

הוספת פרויקט חדש -

public static bool AddProject(Project newProject)

{

try

{

string query = "START TRANSACTION;";

query += $"INSERT INTO task\_management.project(project\_name,manager\_id,customer\_id,team\_leader\_id,total\_hours,start\_date,end\_date) VALUES('{newProject.ProjectName}',{newProject.ManagerId},{newProject.CustomerId},{newProject.TeamLeaderId},{newProject.TotalHours},{BaseService.FormatDate(newProject.StartDate)},{BaseService.FormatDate(newProject.EndDate)});";

//take id of the inserted project

query += "SELECT @@IDENTITY INTO @project\_id;";

//add the divided hours for each department

foreach (DepartmentHours departmentHour in newProject. DepartmentsHours)

{

query += $"INSERT INTO task\_management.department\_hours( project\_id,department\_id,num\_hours) VALUES(@project\_id,{departmentHour.DepartmentId},{departmentHour.NumHours});";

}

if (newProject.Permissions != null)

foreach (Permission permission in newProject.Permissions)

{

query += $"INSERT INTO task\_management.permission( worker\_id,project\_id) VALUES({permission.WorkerId},@project\_id);";

query += $"INSERT INTO task\_management.worker\_hours( project\_id,worker\_id,is\_complete) VALUES (@project\_id,{permission.WorkerId},1);";

}

//add worker-hours with default value-0 to all team workers

List<User> teamWorkers = UserService.GetAllTeamUsers(newProject.TeamLeaderId);

teamWorkers.ForEach(worker =>

{

query += $"INSERT INTO task\_management.worker\_hours( project\_id,worker\_id,is\_complete)

VALUES (@project\_id,{worker.UserId},1);";

});

query += "COMMIT;";

int? x = DBAccess.RunNonQuery(query);

return x >= 4;

}

catch (Exception ex)

{

throw ex;

}

}

קבלת פרויקט מסוים-

public static Project GetProjectById(int projectId)

{

try

{

string query = $"{GetProjectsQuery()} WHERE project\_id ={projectId}";

Func<MySqlDataReader, Project> InitProject = (reader) =>

{

Project project = null;

if (reader.Read())

{

project = BaseService.InitProject(reader);

}

return project;

};

Project foundProject = DBAccess.RunReader(query, InitProject);

return foundProject;

}

catch (Exception ex)

{

throw ex;

}

}

קבלת כל הפרויקטים –

public static List<Project> GetAllProjects()

{

try

{

string query = $"{GetAllProjectsQuery()};";

try

{

List<Project> projectsList = GetAllProjects(query);

return projectsList;

}

catch (Exception ex)

{

throw ex;

}

}

catch (Exception ex)

{

throw ex;

}

}

קבלת פרויקטים לפי ראש צוות-

public static List<Project> GetProjectsByTeamLeaderId(int teamLeaderId)

{

try

{

string query = $"{GetProjectsQuery()} WHERE p.team\_leader\_id ={teamLeaderId} ORDER BY project\_name";

List<Project> projectList = GetProjects(query);

return projectList;

}

catch (Exception ex)

{

throw ex;

}

}

עדכון על סיום פרויקט-

private static bool EditProject(Project project)

{

try

{

string query = $"UPDATE task\_management.project SET is\_complete=1 WHERE project\_id={project.ProjectId};";

return DBAccess.RunNonQuery(query) == 1;

}

catch (Exception ex)

{

throw ex;

}

}

שאילתה בסיסית של קבלת כל הפרויקטים-

private static string GetAllProjectsQuery()

{

string query = "SELECT \* FROM task\_management.project";

return query;

}

* DepartmentService

קבלת כל המחלקות –

public static List<Department> GetAllDepartments()

{

try

{

string query = "SELECT \* FROM task\_management.department;";

Func<MySqlDataReader, List<Department>> func = (reader) =>

{

List<Department> departments = new List<Department>();

while (reader.Read())

{

departments.Add(BaseService.InitDepartment(reader));

}

return departments;

};

List<Department> departmentList = DBAccess.RunReader(query, func);

return departmentList;

}

catch (Exception ex)

{

throw ex;

}

}

* PermissionService

הוספת הרשאות לעובד בפרויקט –

public static int AddPermission(Permission permission)

{

try

{

string query;

query = "INSERT INTO task\_management.permission( worker\_id,project\_id) " +

$"VALUES({permission.WorkerId},{permission.ProjectId});" +"SELECT @@IDENTITY;";

int permissionId = Convert.ToInt32(DBAccess.RunScalar(query));

List<WorkerHours> workerHoursList = WorkerHoursService.GetWorkerHoursPerProject(permission.WorkerId, permission.ProjectId);

if (workerHoursList.Count == 0)

{

WorkerHours workerHours = new WorkerHours { ProjectId = permission.ProjectId, WorkerId = permission.WorkerId };

WorkerHoursService.AddWorkerHours(workerHours);

}

return permissionId;

}

catch (Exception ex)

{

throw ex;

}

}

מחיקת הרשאות לעובד בפרויקט –

public static bool DeletePermission(int permissionId)

{

try

{

string query = $"UPDATE task\_management.permission SET is\_active=0 WHERE permission\_id={permissionId} AND is\_active=1;";

return DBAccess.RunNonQuery(query) == 1;

}

catch (Exception ex)

{

throw ex;

}

}

קבלת הרשאות לעובד–

public static List<Permission> GetPermissions(int userId)

{

string query = "SELECT per.\*, u.user\_name,pro.project\_name FROM task\_management.permission per " +

"JOIN task\_management.user u ON per.worker\_id = u.user\_id JOIN task\_management.project pro ON per.project\_id = pro.project\_id " +$"WHERE per.is\_active = 1 and per.worker\_id = {userId};";

return GetPermissions(query);

}

* CustomerService

קבלת רשימת לקוחות–

public static List<Customer> GetAllCustomers()

{

try

{

string query = "SELECT \* FROM task\_management.customer ORDER BY customer\_name;";

Func<MySqlDataReader, List<Customer>> func = (reader) =>

{

List<Customer> customers = new List<Customer>();

while (reader.Read())

{

customers.Add(BaseService.InitCustomer(reader));

}

return customers;

};

List<Customer> customerList = DBAccess.RunReader(query, func);

return customerList;

}

catch (Exception ex)

{

throw ex;

}

}

* WorkerHoursService

שאילתה קבלת שעות עבודה לעובד לפי פרויקט –

private static string GetWorkerHoursQuery()

{

string query = $"SELECT w.\*,project\_name,user\_name, email,department\_name " +

$"FROM task\_management.worker\_hours w " +

$"JOIN task\_management.project p ON w.project\_id=p.project\_id " +

$"JOIN task\_management.user u ON w.worker\_id=u.user\_id " +

$"JOIN task\_management.department d ON u.department\_id=d.department\_id";

return query;

}

הוספת שעות עבודה לעובד בפרויקט–

public static bool AddWorkerHours(WorkerHours workerHours)

{

try

{

string query = "INSERT INTO task\_management.worker\_hours( project\_id,worker\_id,num\_hours,is\_complete) " +

$"VALUES ({workerHours.ProjectId},{workerHours.WorkerId}, {workerHours.NumHours},1);";

return DBAccess.RunNonQuery(query) == 1;

}

catch (Exception ex)

{

throw ex;

}

}

* PresenceHoursService

קבלת שעות נוכחות לפי עובד -

public static List<dynamic> GetPresenceStatusPerWorkers(int teamLeaderId)

{

string query =

//create view that select presence status

"CREATE VIEW task\_management.presence\_status " +

"AS " +

"SELECT user\_name, pro.project\_name,w.num\_hours, IFNULL(SUM(TIMESTAMPDIFF(SECOND, start\_hour, end\_hour) / 3600), 0) AS presence " +

"FROM task\_management.user u " +

"JOIN task\_management.worker\_hours w ON w.worker\_id = u.user\_id " +

"JOIN task\_management.project pro ON w.project\_id = pro.project\_id " +

"LEFT JOIN task\_management.presence\_hours pre ON pre.project\_id = pro.project\_id AND pre.project\_id = w.project\_id AND pre.worker\_id = u.user\_id AND pre.worker\_id = w.worker\_id " +

$"WHERE u.team\_leader\_id = {teamLeaderId} AND(MONTH(start\_hour) is null or MONTH(start\_hour) = MONTH(CURRENT\_DATE())) GROUP BY user\_name, pre.project\_id; " +

//select sum of all projects presence status per worker and details of every project

"SELECT user\_name,null as project\_name,sum(num\_hours) AS num\_hours, sum(presence) AS presence " +

"FROM task\_management.presence\_status " +

"GROUP BY user\_name " +

"UNION SELECT \* FROM task\_management.presence\_status; " +

"DROP VIEW task\_management.presence\_status";

Func<MySqlDataReader, List<dynamic>> InitPresenceStatusList = (reader) =>

{

List<dynamic> presenceStatusList = new List<dynamic>();

while (reader.Read())

{

var presenceStatus = new { UserName = reader.GetString(0), ProjectName = reader.IsDBNull(1) ? null : reader.GetString(1), ProjectHours = reader.GetInt32(2), PresenceHours = reader.GetDecimal(3) };

presenceStatusList.Add(presenceStatus);

}

return presenceStatusList;

};

return DBAccess.RunReader(query, InitPresenceStatusList);

}

קבלת שעות נוכחות לפי פרויקט –

public static List<dynamic> GetPresenceStatusPerProjects(int workerId)

{

string query ="SELECT pro.project\_name, IFNULL(SUM(TIMESTAMPDIFF(SECOND, start\_hour, end\_hour) / 3600), 0) AS presence, num\_hours " +

"FROM task\_management.worker\_hours w " +

"LEFT JOIN task\_management.project pro " +

"ON pro.project\_id = w.project\_id " +

"LEFT JOIN task\_management.presence\_hours pre " +

$"ON pro.project\_id = pre.project\_id AND pre.worker\_id = {workerId} AND MONTH(start\_hour) = MONTH(CURRENT\_DATE()) " +

$"WHERE w.worker\_id = {workerId} " +

"AND w.num\_hours > 0 " +

"GROUP BY pre.project\_id;";

Func<MySqlDataReader, List<dynamic>> InitPresenceStatusList = (reader) =>

{

List<dynamic> presenceStatusList = new List<dynamic>();

while (reader.Read())

{

var presenceStatus = new { ProjectName = reader.GetString(0), PresenceHours = reader.GetDecimal(1), ProjectHours = reader.GetInt32(2) };

presenceStatusList.Add(presenceStatus);

}

return presenceStatusList;

};

return DBAccess.RunReader(query, InitPresenceStatusList);

}

* BaseService

public static string GetStringValueOrNull(string value)

{

return value != null ? $"'{value}'" : "null";

}

public static string GetIntValueOrNull(int? value)

{

return value != null ? $"{value}" : "null";

}

public static string FormatDate(DateTime date,string format="yyyy-MM-dd")

{

return $"'{date.ToString(format)}'";

}

שליחת מייל –

public static bool SendEmail(Email email)

{

try

{

SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587);

smtpClient.EnableSsl = true;

smtpClient.Timeout = 1000000;

smtpClient.UseDefaultCredentials = false;

NetworkCredential basicCredential = new NetworkCredential(email.CompanyName,email.CompanyPassword);

smtpClient.Credentials = basicCredential;

MailMessage mailMessage = new MailMessage();

mailMessage.From = new MailAddress(email.CompanyName);

mailMessage.To.Add(toAddress));

mailMessage.Subject = email.Subject;

mailMessage.Body = email.Body;

mailMessage.BodyEncoding = Encoding.UTF8;

smtpClient.Send(mailMessage);

return true;

}

catch

{

return false;

}

}

בדיקת תאריך דד-ליין לצורך שליחת מייל התראה –

public static async Task DeadLineEmail()

{

// Set up a timer that triggers every day.

System.Timers.Timer timer = new System.Timers.Timer();

timer.Interval = 60000 \* 60 \* 24;

timer.Elapsed += new ElapsedEventHandler(OnStart);

string time = ConfigurationManager.AppSettings["deadLineEmailHour"];

int hour = int.Parse(time.Split(':')[0]);

int minute = int.Parse(time.Split(':')[1]);

DateTime currentDateTime = DateTime.Now;

DateTime dateTimeToStart = currentDateTime.Date + new TimeSpan(hour, minute, 0);

if (dateTimeToStart < currentDateTime)

dateTimeToStart= dateTimeToStart.AddDays(1);

TimeSpan timeout = dateTimeToStart - currentDateTime;

Thread.Sleep(timeout);

timer.Start();

}

שליחת מייל יום לפני תאריך סיום משימה –

public static void sendDeadLineEmail()

{

List<Project> projects = ProjectService.GetAllProjects(DateTime.Today);

projects.ForEach(project =>

{

Email emailToTeamLeader = new Email() { Subject = "Deadline is Here!!", Body = $"Your project {project.ProjectName} is finish tomorrow!\n please pay attention to verify your workers finished their task." };

Email emailToWorkers = new Email() { Subject = "Deadline is Here!!", Body = $"Your project {project.ProjectName} is finish tomorrow!\n please pay attention to finish your task." };

List<WorkerHours> workerHoursList = WorkerHoursService.GetUncompletedWorkersHours(project.ProjectId);

List<string> emailAddresses = workerHoursList.Select( workerHours => workerHours.Worker.Email).ToList();

emailToWorkers.ToAddress.AddRange(emailAddresses);

if (workerHoursList.Count > 0)

emailToTeamLeader.ToAddress.Add(UserService.GetUserById(project.TeamLeaderId).Email);

SendEmail(emailToWorkers);

SendEmail(emailToTeamLeader);

});

}

**2. טכנולוגיות:**

**2.1Angular –**

היא תשתית מוכנה (framework) קוד פתוח ליישומי רשת, המתוחזקת על ידי גוגל ועל ידי קהילה רחבה של מפתחים **.**

התשתית מיועדת לפתרון אתרים בפיתוח single page application , ופישוט הפיתוח והבדיקות של יישומים אלו , באמצעות תשתית תוכנה לארכיטקטורות צד לקוח כמו mvc או mvvm יחד עם רכיבים בהם משתמשים בדרך כלל ביישומי אינטרנט עשירים.

**Single page application** – הם יישומי רשת שמטרתם לתת חווית משתמש מהירה וזורמת יותר, הדומה יותר לתוכנת מחשב רגילה (שאינה יישום רשת).

ביישומים אלו כל הקוד הדרוש HTML, CSS,JS מגיע אל הדפדפן בטעינת דף אחת ומשאבים נוספים נטענים בצורה דינמית, בדרך כלל כתגובה לפעולות המשתמש.

דף האינטרנט עשוי להשתנות מעט על מנת לתת למשתמש הבנה טובה יותר של הניווט בדף.

כתיבת אפליקציית אנגולר מתבצעת ע"י כתיבת HTML שבתוכו משולבים תוספות של אנגולר, כתיבת components כדי לשלוט על ה-templates,כתיבה של services כדי לשלוט על הלוגיקה,

ושילוב של component ו- service ליצירת modules .

**2.2 TypeScript** –

Javascriptכשפה, עוברת שינויים רבים בשנים האחרונות. השימוש ב - Javascript הפך לכה נפוץ בגלל השימוש המוגבר והמוגדל ברשת האינטרנט לשפת הפיתוח הנפוצה בעולם. כחלק מגידול השימוש בשפה עלה הצורך להוסיף יכולות רבות לשפה לרבות יכולות מתקדמות הקשורות לעולם ה"תכנות מונחה עצמים" . Javascript בפני עצמה תתמוך ביכולות רבות בשנים הקרובות מפני שמי שמריץ את הקוד שנכתב ב- Javascript הוא הדפדפן עצמו. נוצר מצב שכל דפדפן תומך בגרסה שונה של Javascript דבר המסבך את המפתח בבחירת הגרסה הנכונה לפיתוח.

Microsoft זיהתה את הצורך ופיתחה **Typescript** שהיא הרחבה ל-JavaScript שמוסיפה לשפה את כל מה שהיה חסר לה. Typescript מאפשרת לנו לכתוב קוד אחיד המכיל את כל יתרונות ה- JavaScript ומוכן כבר היום ליכולות שיתמכו ב- JavaScriptבעתיד.

החיסרון הוא שלא ניתן להריץ היום Typescript בכל הדפדפנים.

כאן נכנס לסיפור ה- (TSC) Typescript compilerאו בעברית "המהדר" .

ה compiler - יודע לקחת את הקוד שכתבנו ב- Typescript ולהמיר אותו ל- JavaScript נקי גם בגרסאות הכי נמוכות שלו (ניתן לבחור את הגרסה המובקשת). בצורה זו ניתן לכתוב מערכת ב- Typescript ולהיות בטוחים שהקוד ירוץ בצורה תקינה בכל הדפדפנים , ובעתיד כאשר כל הדפדפנים יתמכו בגרסאות המוקדמות של JavaScript נוכל רק לשנות את הגדרת ה-compiler, ולקבל את ההמרה לגרסה החדשה והעדכנית ביותר, ושוב, כך אנחנו כותבים קוד היום שכבר מוכן לאיך שתראה השפה בעוד מספר שנים.

2.3 - **Visual Studio 2017**

.Visual studio סביבת הפיתוח המרכזית בעולם המייקרוסופטי נקראת

סביבה זו מכילה מאות אפשרויות ותומכת במספר שפות תכנות. בנוסף, יש לציין, כי רוב שפות התכנות והטכנולוגיות הנתמכות על ידי סביבת פיתוח זו הן חלק מתשתית הפיתוח של חברת מיקרוסופט.

. תשתית זו מכילה מספר עצום של ספריות קוד וטכנולוגיות המקלות עלינו NET framework

בתהליך יצירת התוכנה.

3. מדריך למשתמש:

3.1 הוראות כלליות לשימוש באתר:

על מנת להיכנס למערכת יש באפשרותך לעבור דרך מסך ההתחברות/ההרשמה.

אם המערכת מזהה שהמשתמש רשום היא מנווטת אותו ישירות לדף המתאים ללא צורך בהרשמה מראש.

במקרה של שכחת סיסמא ישלח טוקן למשתמש, שים לב: הטוקן תקף ל-10 דקות בלבד!

לכל דרגה בחברה מסכים שונים.

לכלל המשתמשים תהיה תמיד אפשרות של התנתקות - ע"י לחיצה על התנתקות תתרחש יציאה מן האתר, כלומר בפעם הבאה שייכנסו דרך המחשב הנוכחי לא ישמרו פרטי המשתמש.

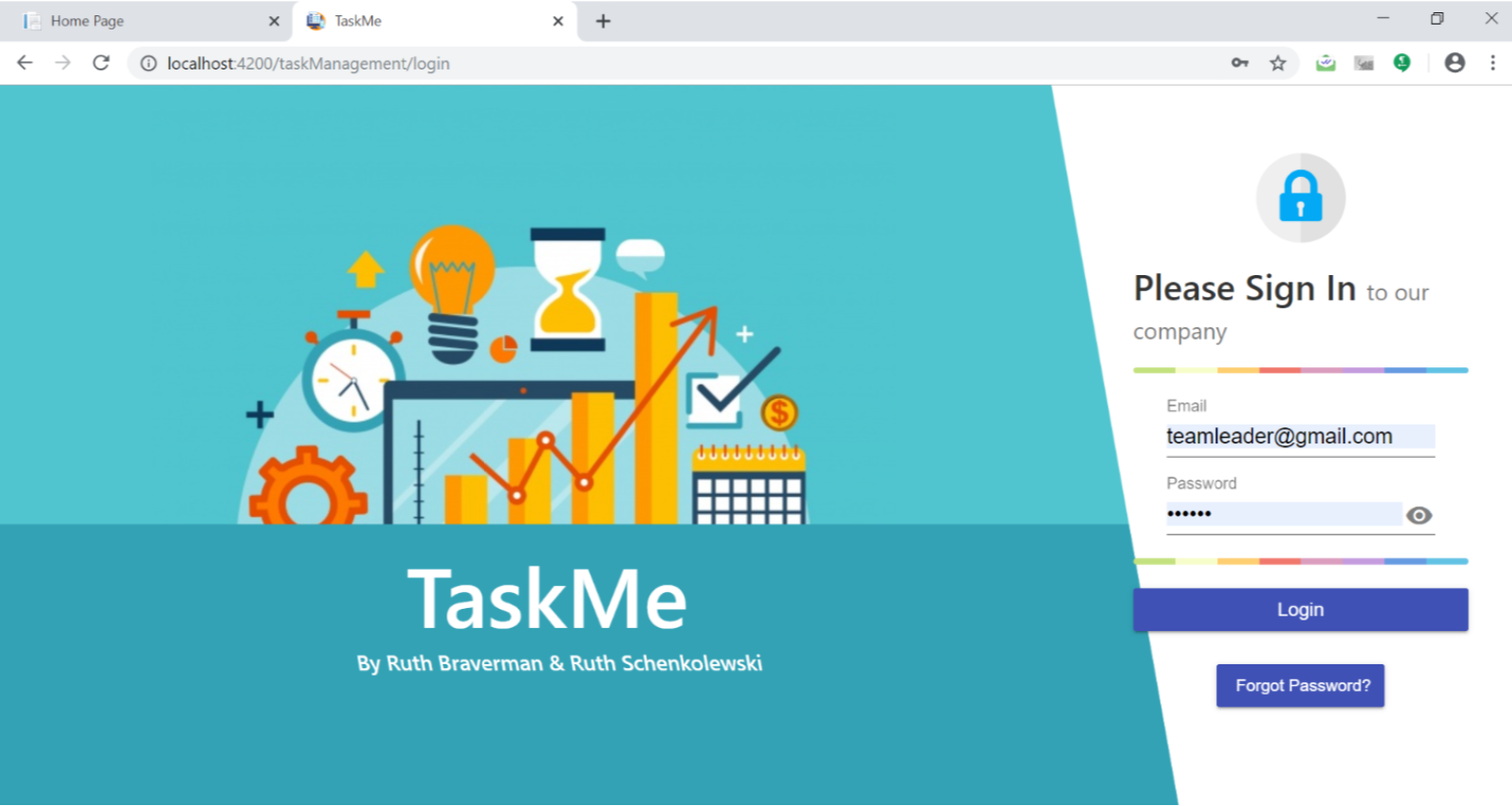
* מנהל הנכנס לאתר יוצגו לו המסכים הבאים:
* ניהול משתמשים – במסך זה יוכל המנהל לראות את כל העובדים.

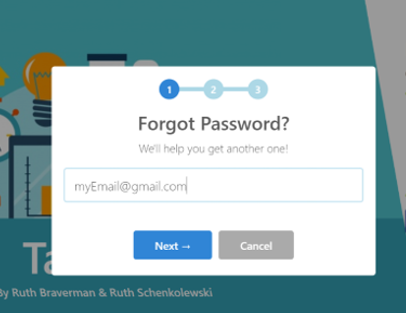
על כל עובד יופיעו כמה כפתורים: מחיקת עובד ועריכת פרטי עובד וכן עריכת הרשאות לפרויקטים. בלחיצה על עריכה יפתח טופס המכיל את פרטי העובד, המנהל יוכל לעדכן את פרטי העובד וכן להוסיף/לשנות את תמונת העובד. במידה והנתונים שהוזנו אינם נכונים תשלח הודעה מתאימה.

* ניהול פרויקטים – במסך זה יוכל המנהל לראות את כל הפרויקטים כולם או לפי מיונים וסינונים – את כל פרטיהם כולל מצבם – קצב התקדמותם.
* ניהול ראשי צוותות – ניהול ראשי הצוותות והעובדים שתחתיהם.
* ראש צוות הנכנס לאתר יוצגו לו המסכים הבאים:
* מעקב אחר העובדים שתחת פיקודו
* מעקב אחר הפרויקטים שתחתיו ועריכת שעות נצרכות לפי מחלקות ולפי עובדים.
* גרף המציג התקדמות כל עובד שתחת אותו ראש צוות בפרויקט/ים השונים שתחתיו.
* עובד הנכנס לאתר יופיע לו מסך בו יוכל נהל את כל הפרויקטים והמשימות שניתנו לו. העובד יוכל לראות את כל המשימות שנשלחו לו, לבצע התחלת משימה או סיום וכן לראות בגרף כמה סיים מתוך המשימות המוטלות עליו. בנוסף יוכל העובד לשלוח מייל לממונה עליו.

**3.2 מסכים:**

**כניסה –**

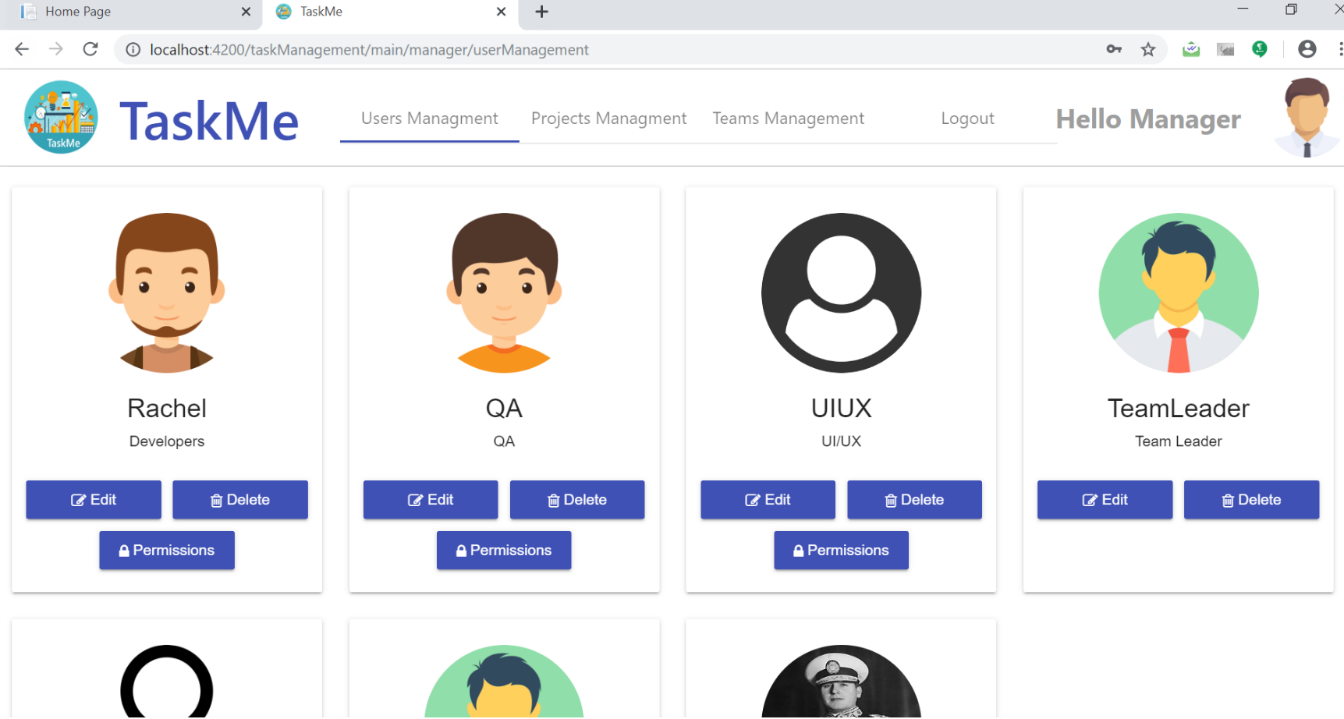


**שכיחת סיסמא -**

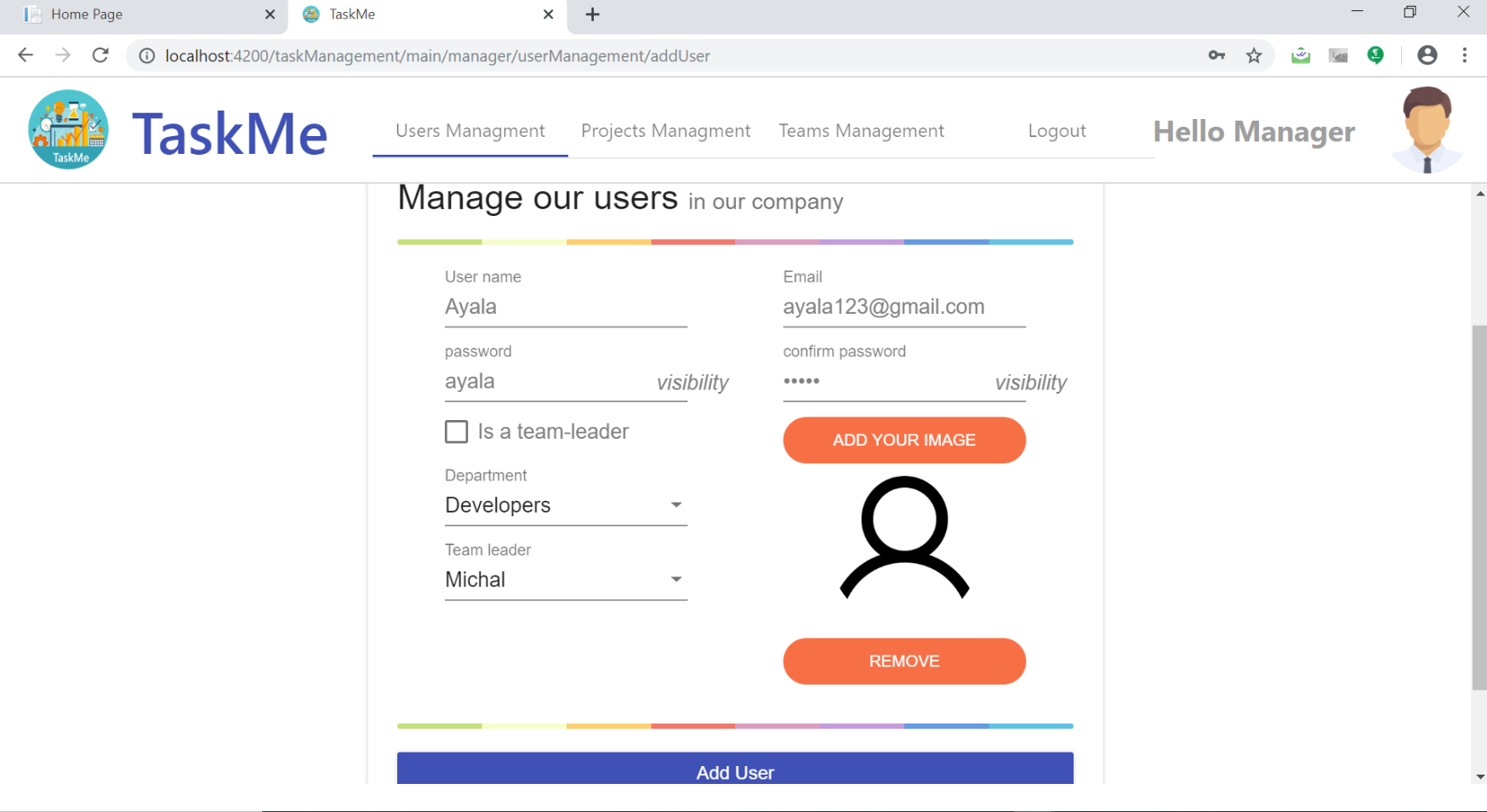
**מסכי מנהל –**

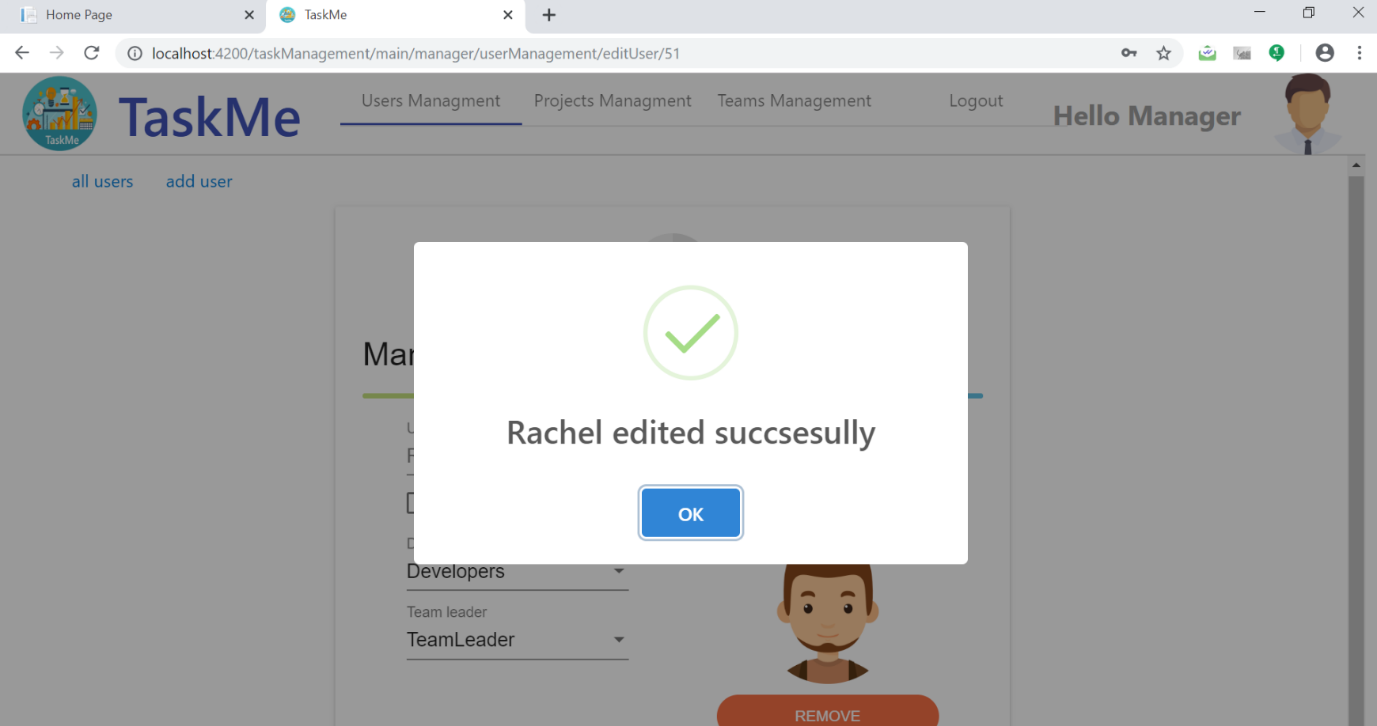
* **ניהול עובדים –**

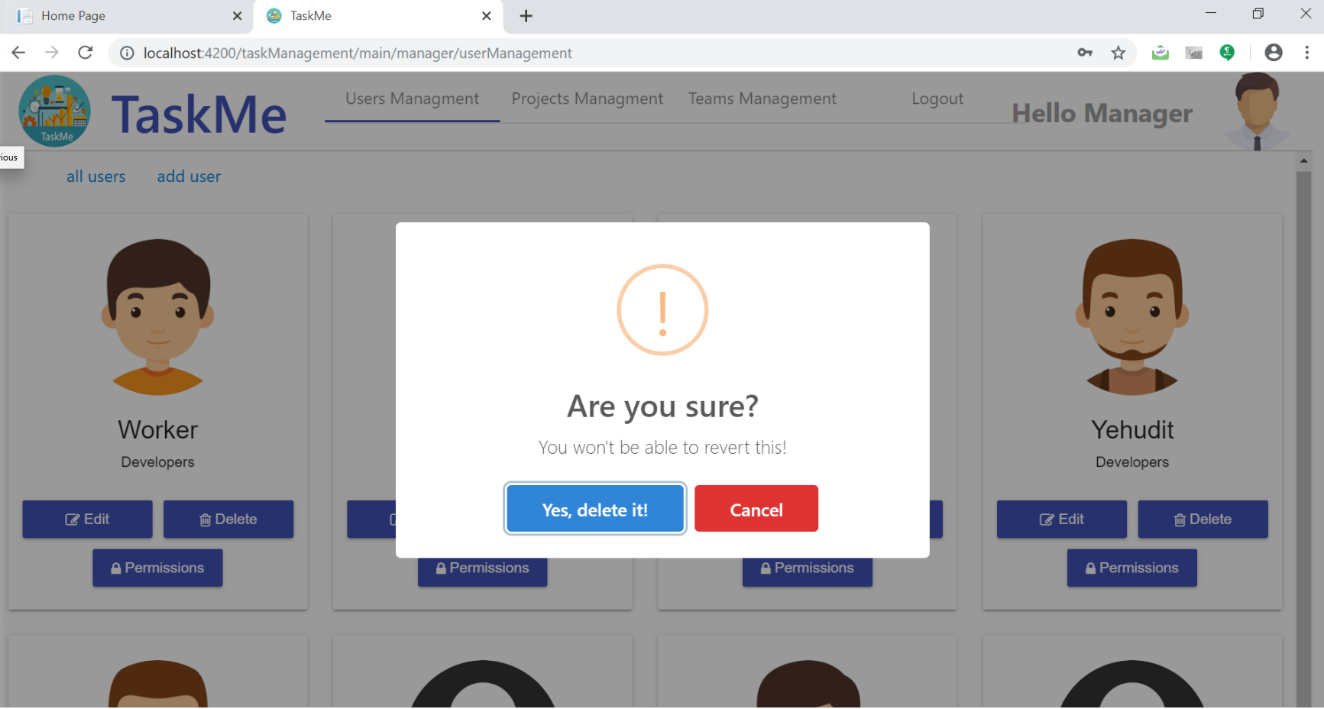
**כל העובדים –**



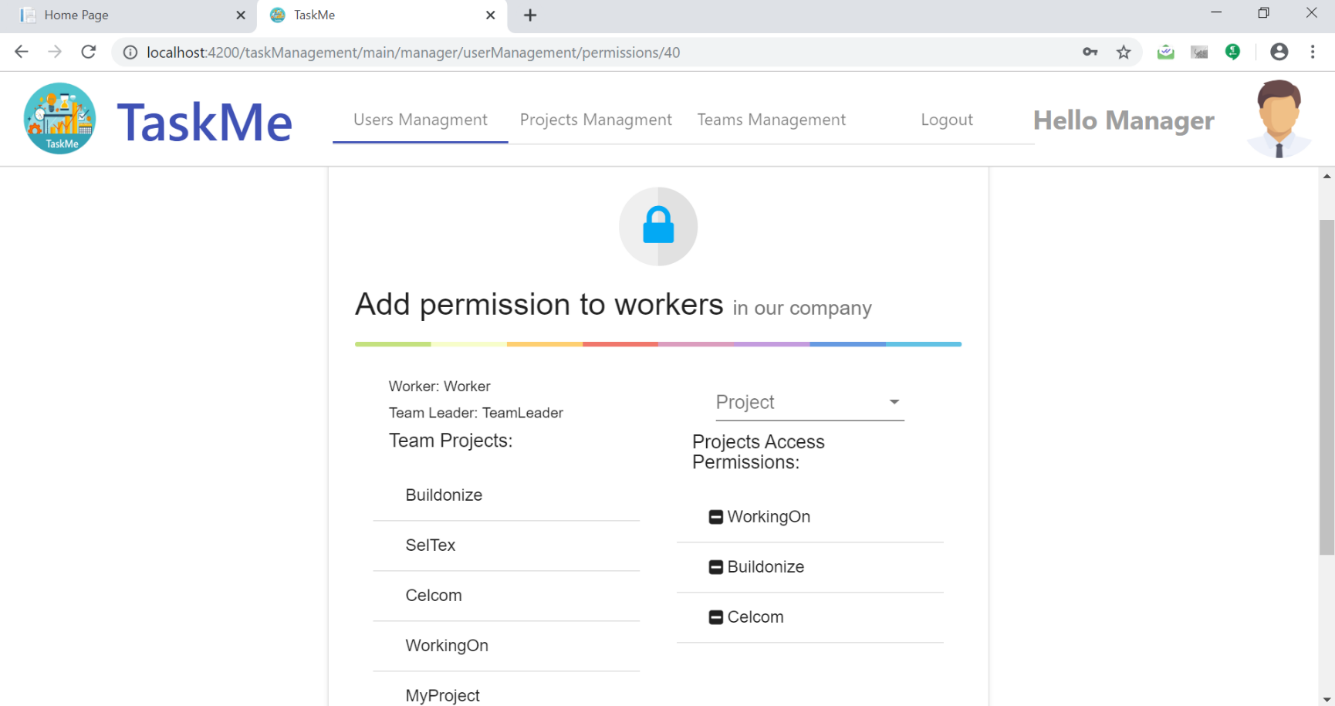
**הוספת עובד חדש –**



 **עריכת פרטי עובד –**

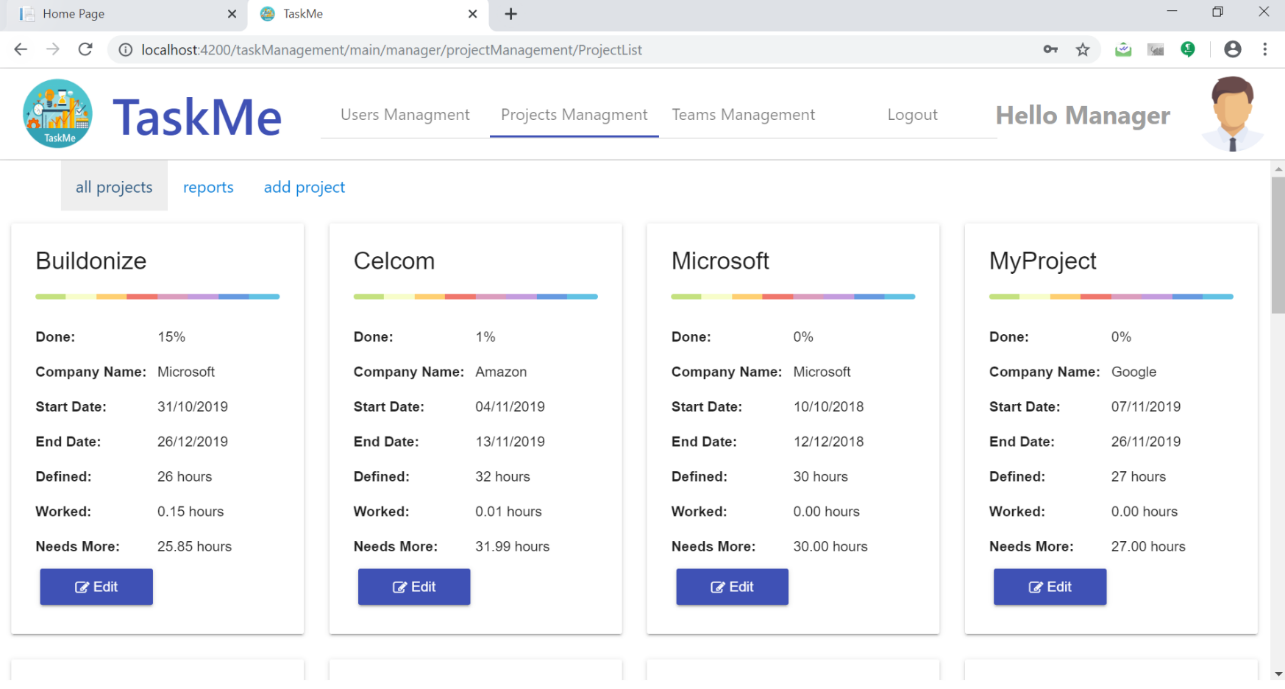
 **מחיקת עובד –**

**ניהול הרשאות –**

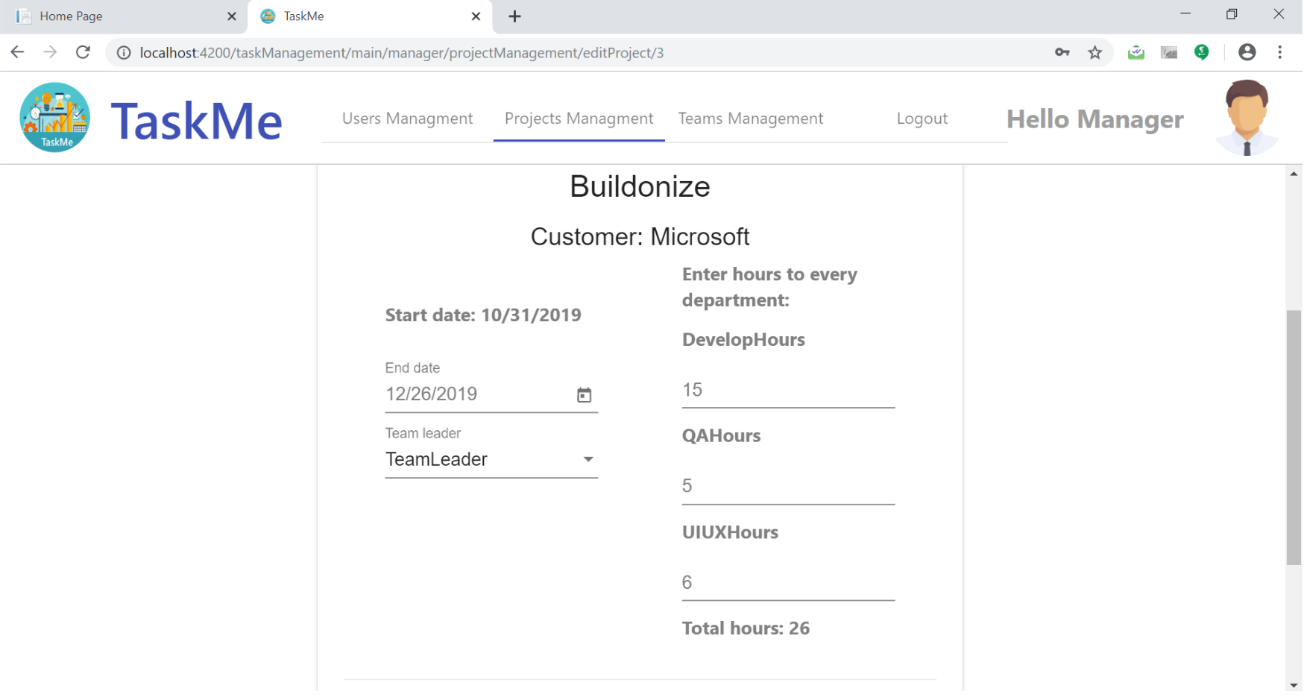


* **ניהול פרויקטים –**

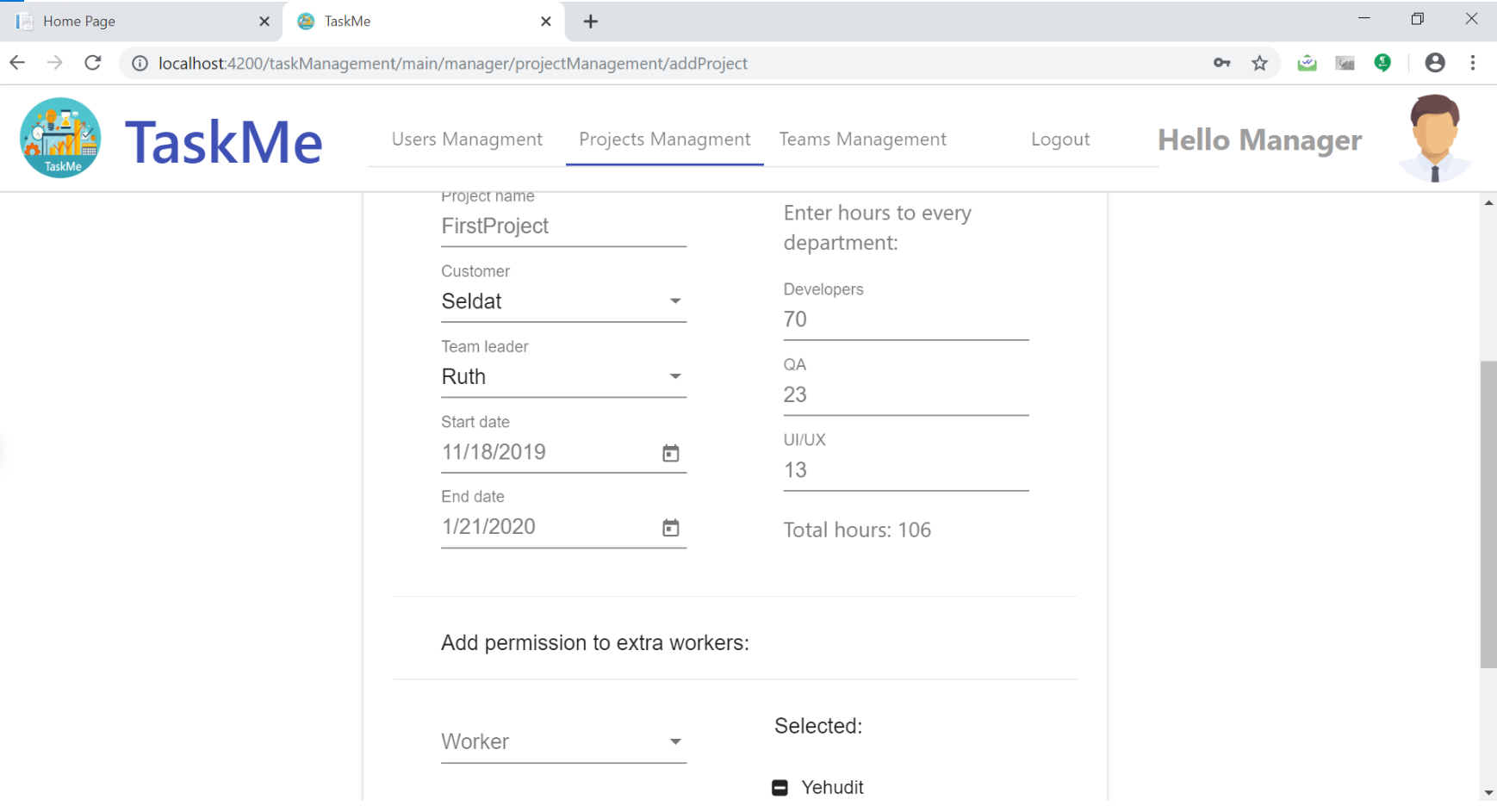
**כל הפרויקטים –**



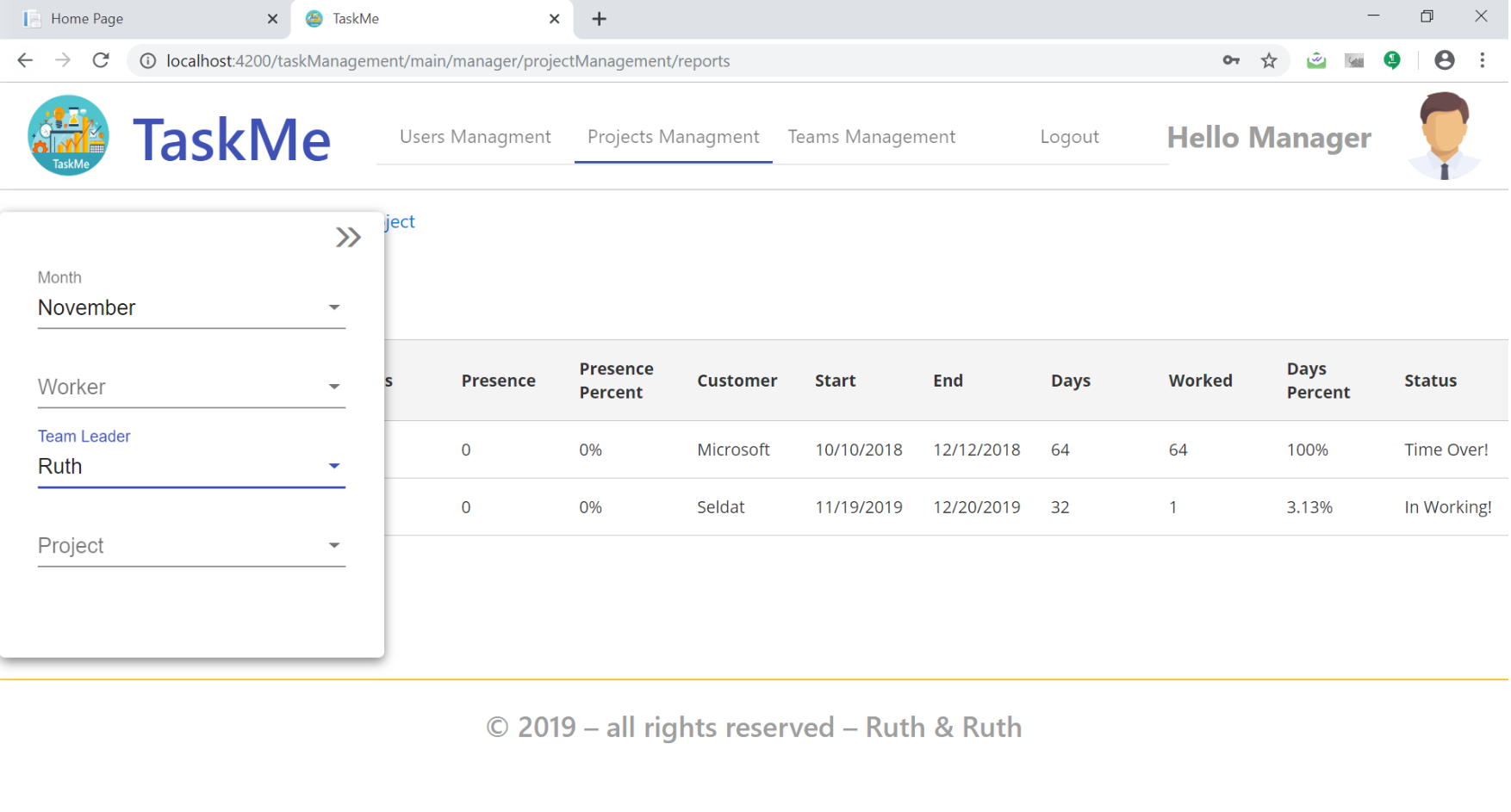
**עריכת פרטי פרויקט –**

****

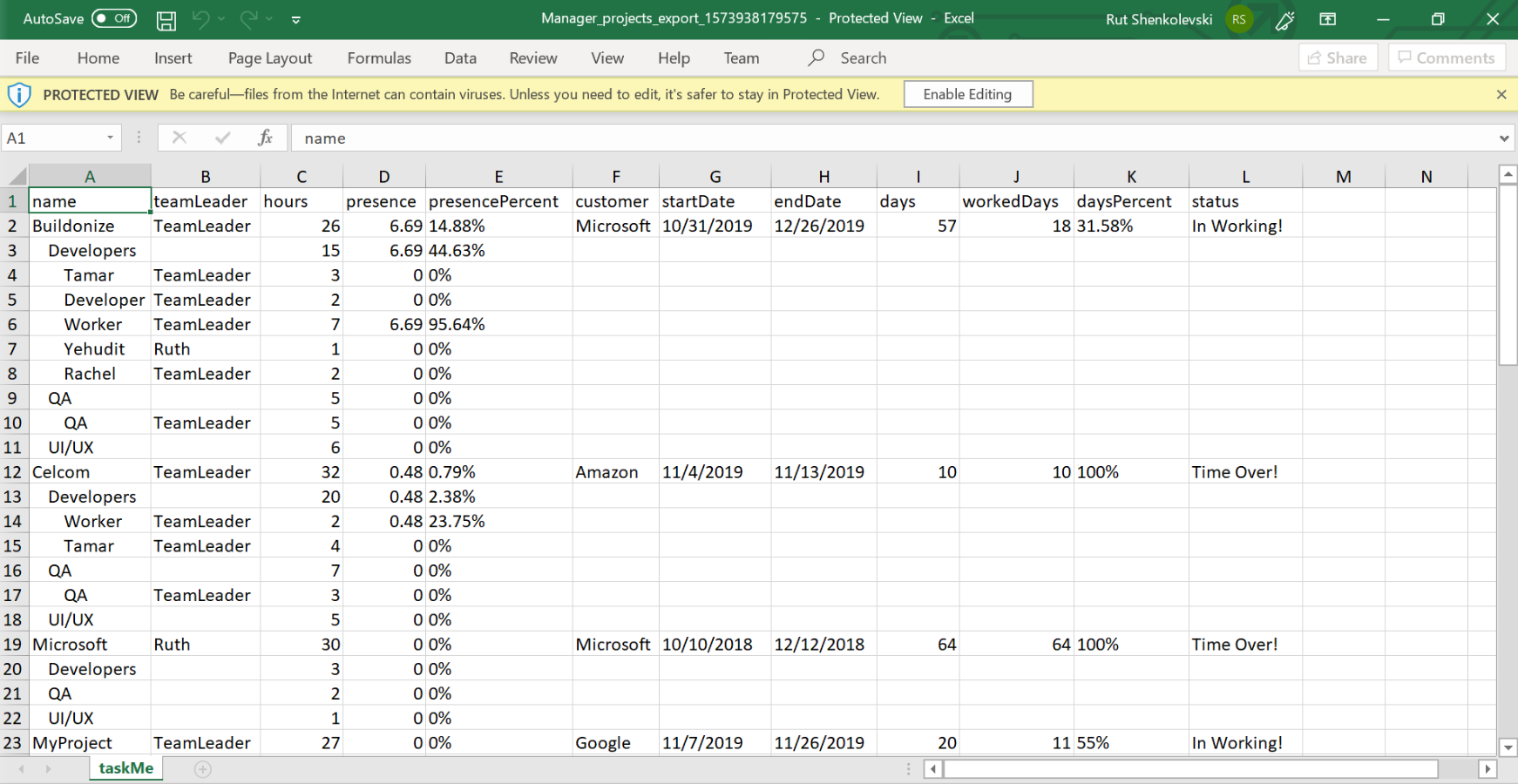
**הוספת פרויקט חדש –**



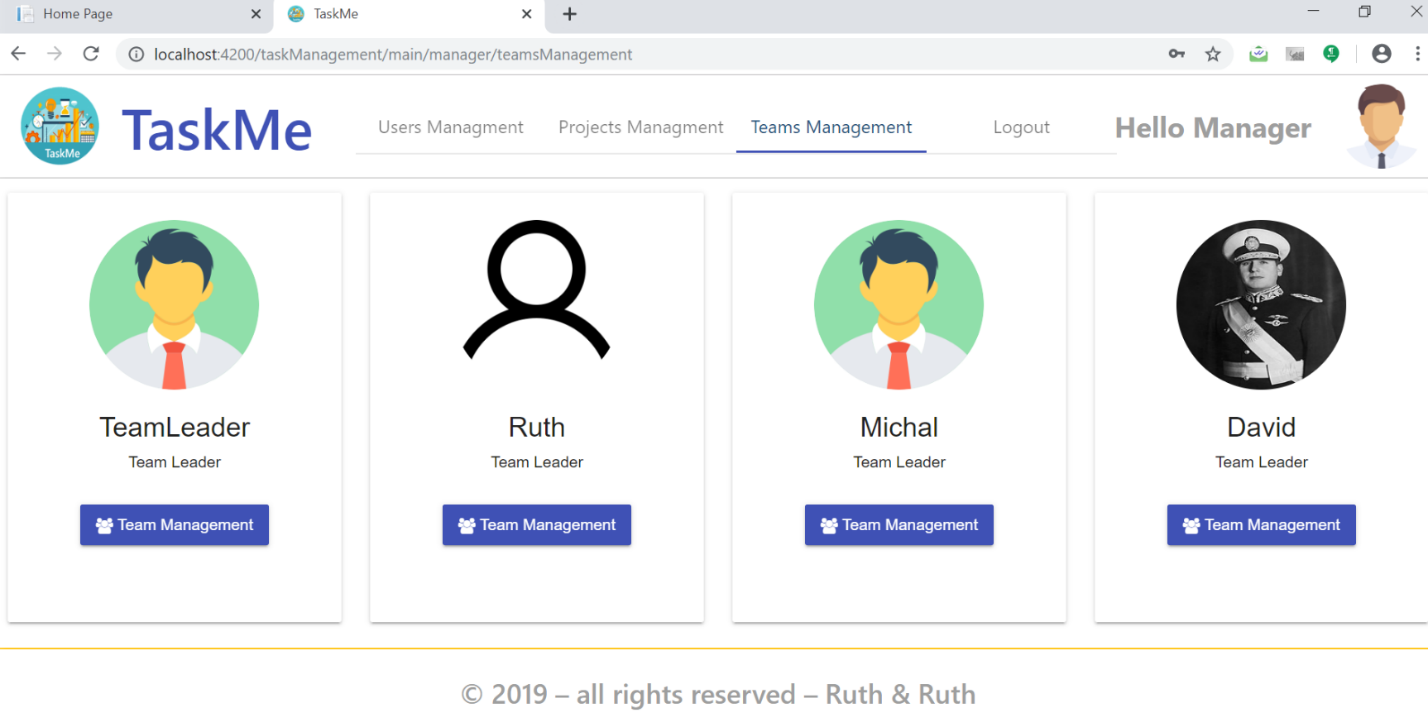
**תצוגת הפרויקטים בצורת דוח –**



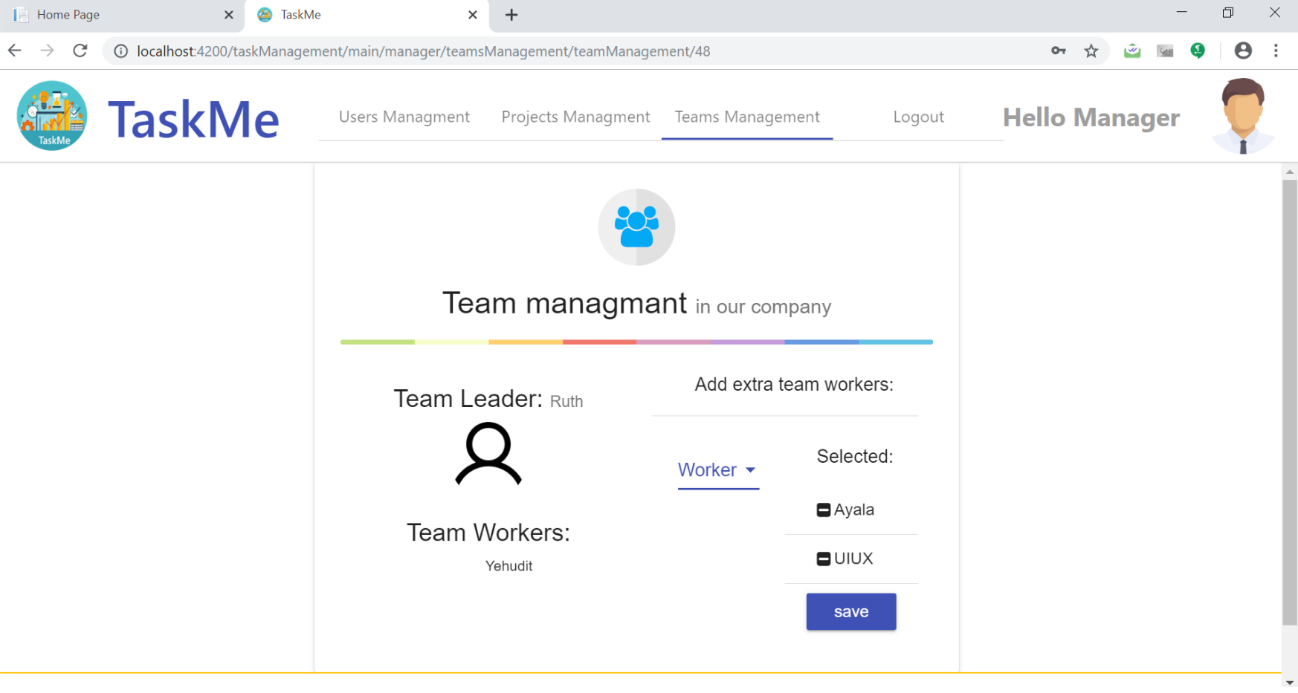
**יצוא הדוחות לאקסל –**



* **ניהול צוותות –**

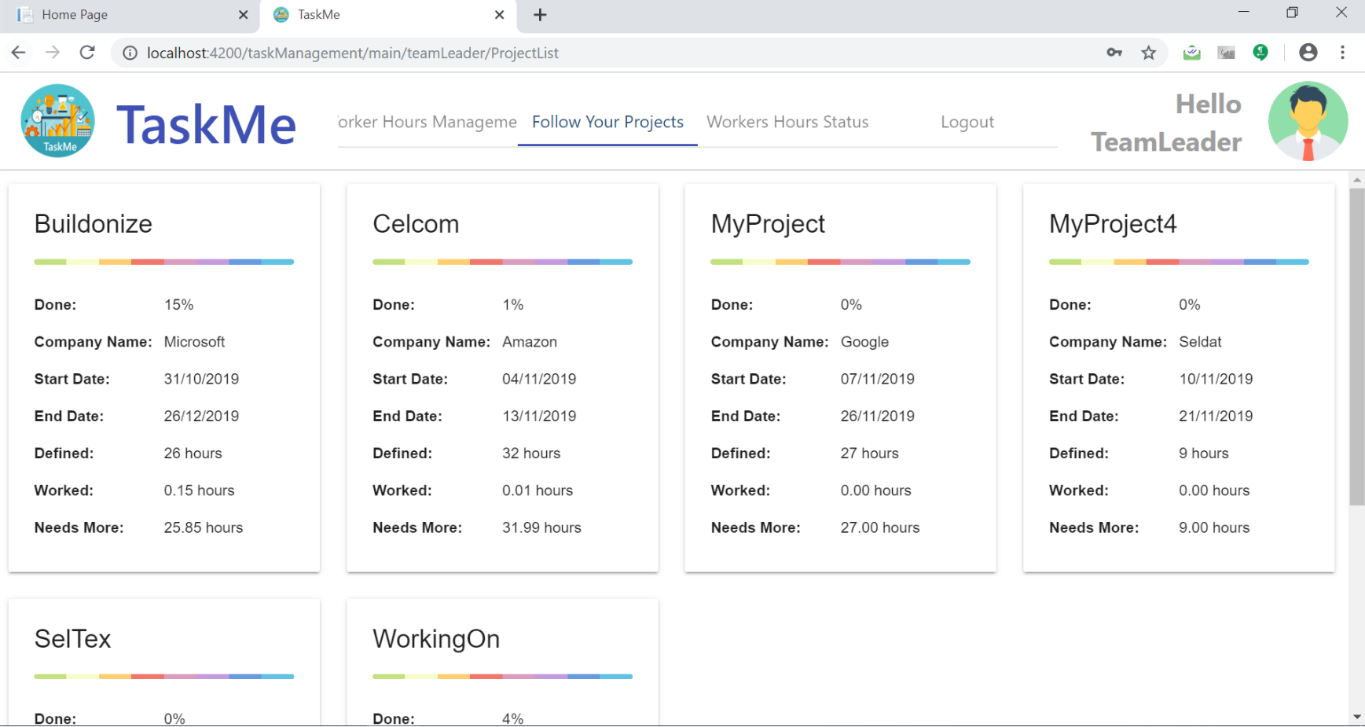
**כל ראשי הצוות –**

**עריכת העובדים המשויכים לראש צוות –**

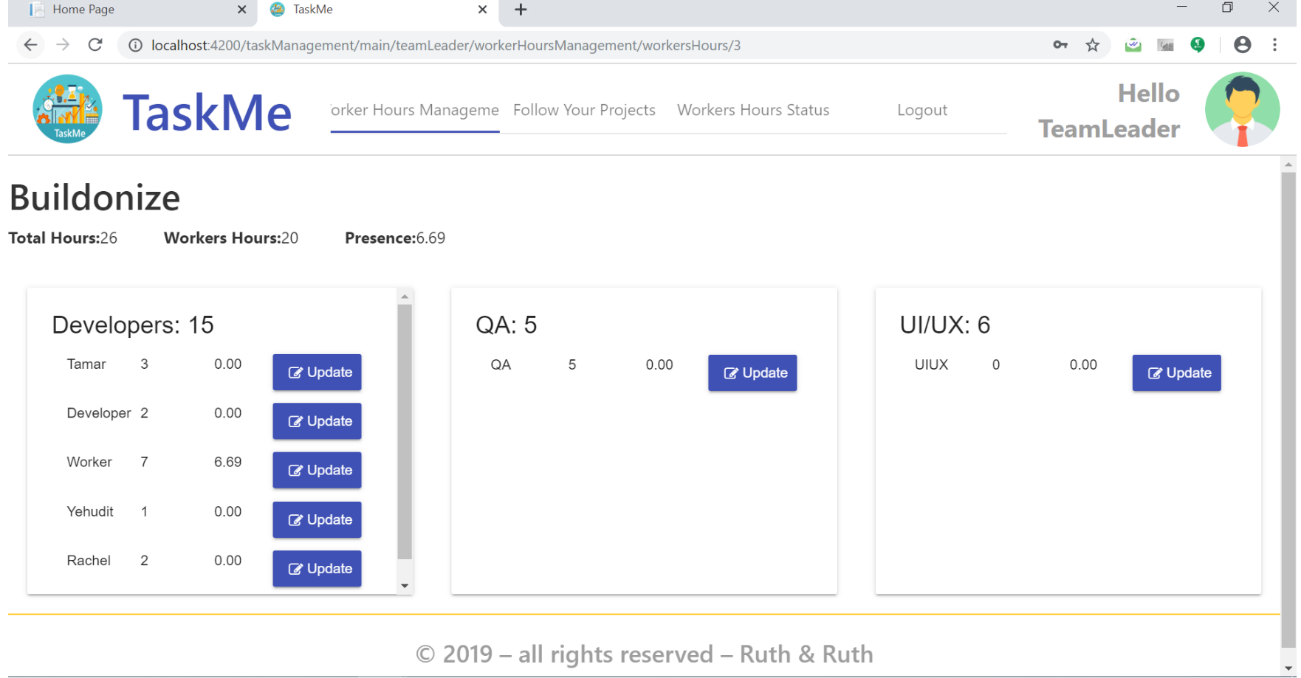
****

**מסכי ראש צוות –**

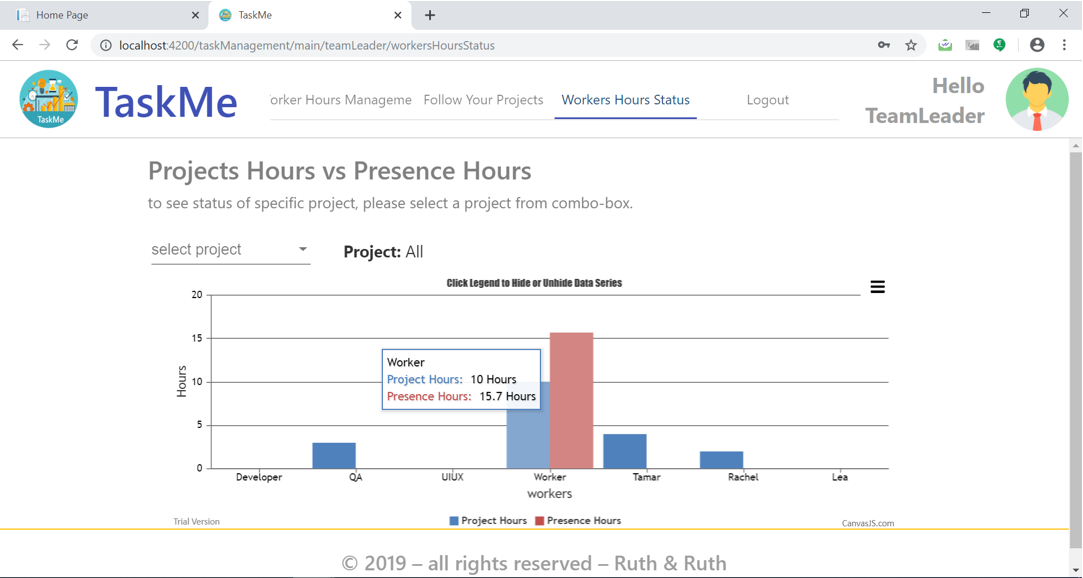
**מעקב אחר הפרויקטים שתחת ניהולו –**



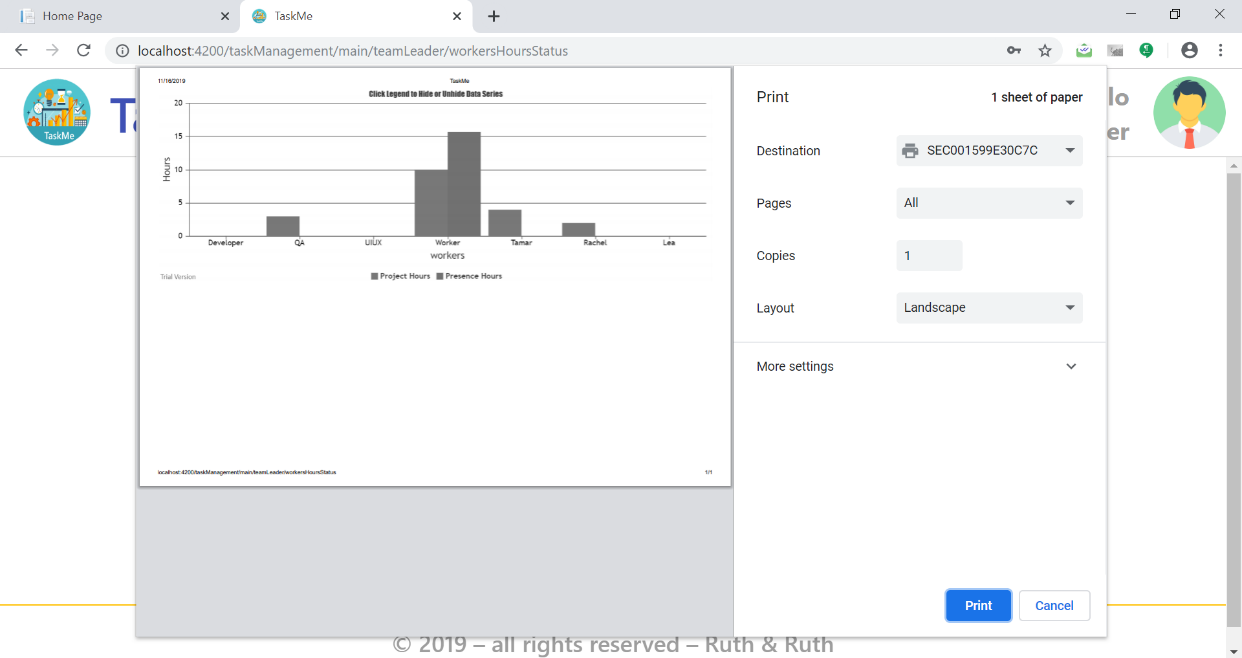
**תצוגת כל הפרויקטים שתחת אחריותו ועדכון שעות עבודה לפי מחלקות –**

**עריכת שעות לכל עובד –**

**תצוגת גרף של כל העובדים וכל הפרויקטים שתחת ניהולו –**

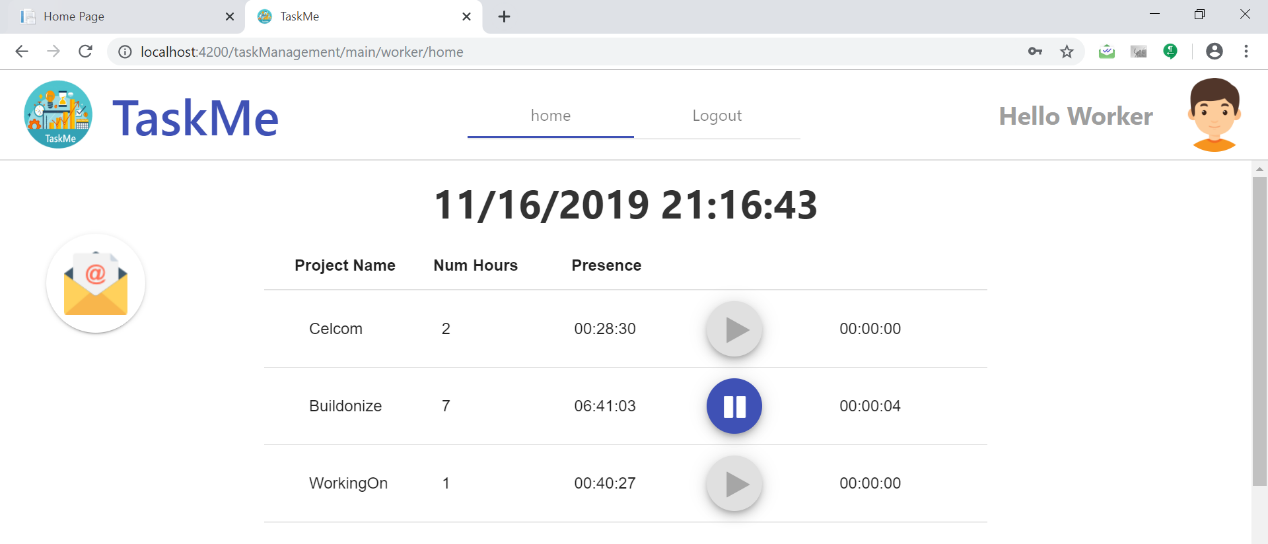


**ייצוא הגרף כתמונה או הדפסתו –**

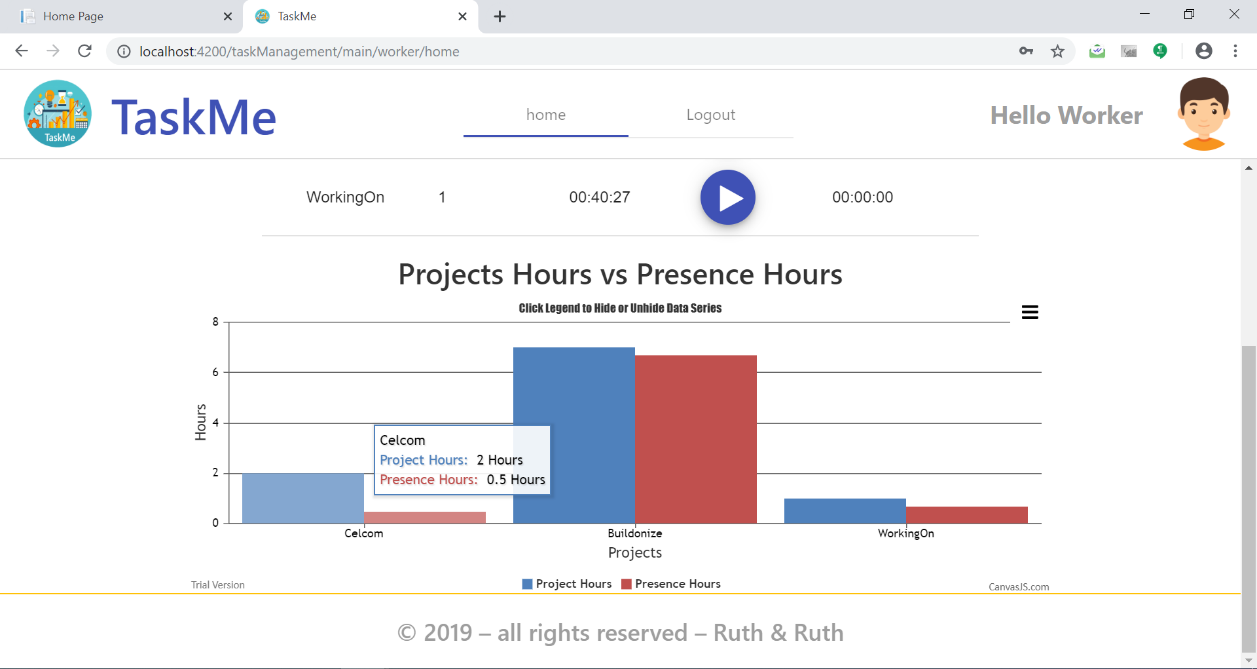
****

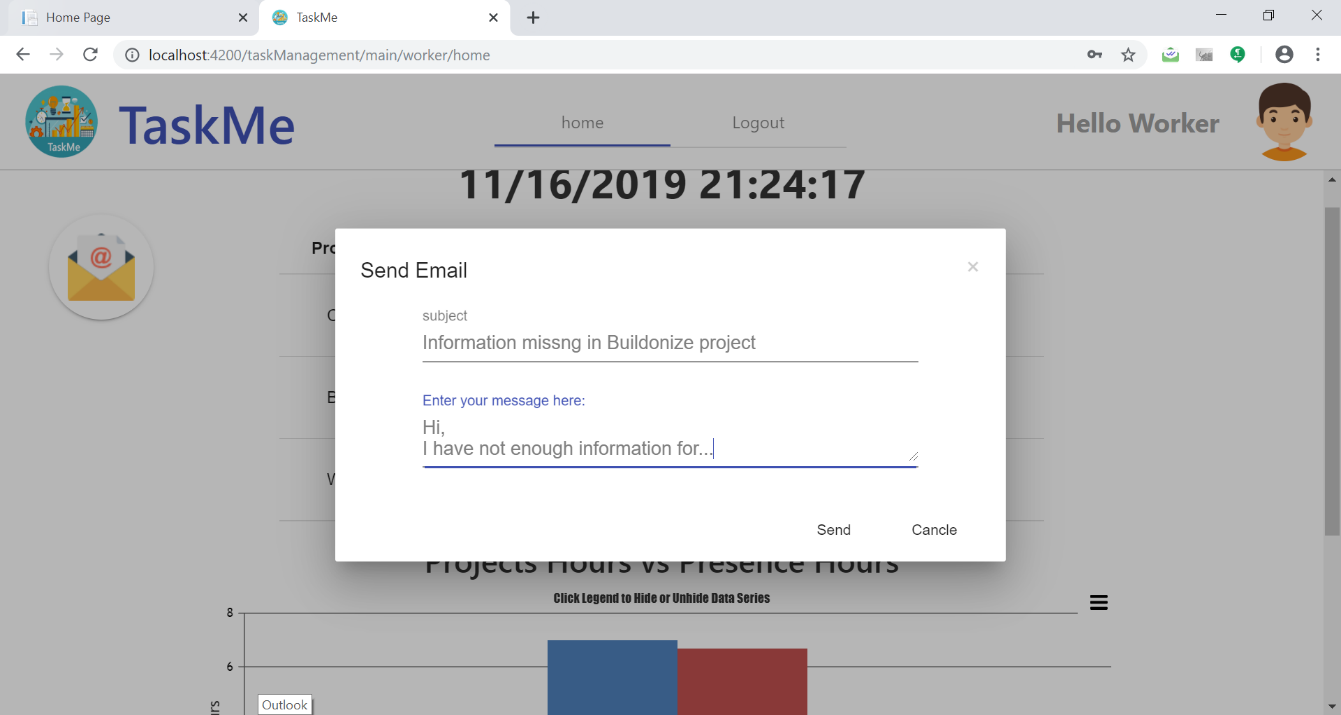
**מסכי עובד –**

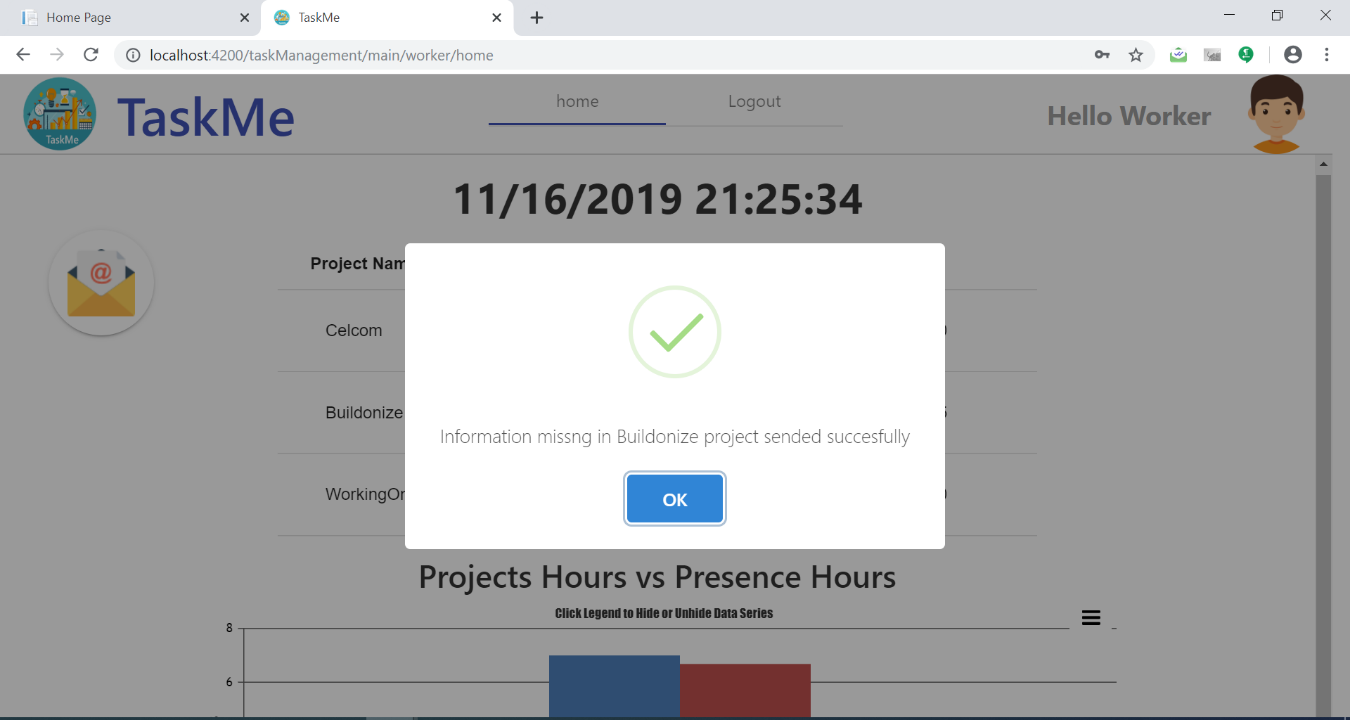
**התחלה/סיום משימה –**

****

**תצוגת גרף של כל הפרויקטים המשויכות אליו עם שעות נדרשות לעומת שעות נוכחות –**

****

**שליחת מייל לראש הצוות שלו –**

****

**4. סיכום ומסקנות:**

כשניגשנו לתכנון פרויקט הגמר, ידענו שאת כל הידע, העמל והיכולות שלנו נשקיע בו.

והיום, במבט לאחור יודעות אנחנו שאכן עצם הידיעה הראשונית והשאיפה היא זאת שהביאה אותנו עד הלום.

האתרTaskMe מושלם – אתר אמיתי מהחל ועד כלה, מההצעה ועד התוצאה.

תחילה בנינו אותו על הדף: תכננו מחלקות, מסכים ורכיבים.

לאחר מכן פנינו לבנותו על המחשב. מחלק זה למדנו רבות, למדנו כיצד לגלות טעויות הקוטעות את הרצת הפרויקט ותוקעות את המשך בנייתו. למדנו כיצד להשתמש בפונקציות רבות ומגוונות הנמצאות ב-.net ברכיבים ובמאפיינים.

במהלך בניית הפרויקט נתקלנו במספר דברים שגרמו לנו לא פעם לשנות מחלקות שלמות או למחקם כליל וכך גם לגבי רכיבים ומשתנים רבים בפרויקט.

למדנו לא להתייאש, גם לאחר שעובדים על פתרון לבעיה במשך שבוע ולאחר מכן מוצאים דרך יעילה יותר, לא מוותרים וממשים אותה.

ניתן לומר כי הפרויקט תרם לנו כסטודנטיות וכמתכנתות, הוא הכניס אותנו עמוק לתוך המבנה הפנימי היוצר כל תוכנת מחשב באשר היא. ניתוח ואפיון של מערכת וכך נוכל תכנת פרויקטים נוספים בהרכב גדול יותר.

בעתיד יהיה ניתן לשכלל פרויקט זה , להוסיף לו פונקציונליות, הרחבת סוגי הבעיות ועוד.

לסיכום, לאחר שעות רבות של עמל ויזע אנו חשות שהפקנו רבות מכך,

וכמו שנאמר כבר בחז"ל: "יגעת ומצאת תאמין" - כי רק האדם היגע יזכה לראות פירותיו.

ולוואי שנזכה גם הלאה להמשיך ולעמול בכל תחום שניפגש בחיינו.

**5. נספחים:**

**5.1 Sha256** הצפנת סיסמת משתמש –

הצפנת סיסמא על ידי שימוש באלגוריתם sha256 מתוך ספרית async-sha256.

מאחורי אלגוריתם זה עומדת לוגיקה מורכבת שבסופו של דבר התוצאה היא string המכיל 64 ספרות הקסאדצימאליות. (כל ספרה הקסאדצימאלית היא 4 ספרות בינאריות ולכן אלגוריתם זה מכונה sha256 (64\*4=256)).

בצורה כזו נשמרת הסיסמא בבסיס הנתונים.

במקרה של פריצה למערכת על ידי אקרים אין באפשרותם לדעת מה הסיסמא בעקבות ההצפנה. (לא ניתן להפעיל את האלגוריתם בצורה הפוכה ולקבל את הערך המקורי).

**6. ביבליוגרפיה:**

**Stackoverflow**

**Developer.google.com**

**Angular material**

**Bootstrap**

**Codepen**

**W3school**