

פרויקט באפליקציות רשת אסינכרוניות: StockWizard

תיכון דרכא רמון על שם אילן ואסף רמון | גדרה

מנחה: אלכסנדר גרשברג

מחבר: יהונתן וישנה;ת"ז: 215310269

תשפ"ג

# תוכן עניינים

תוכן עניינים

[תוכן עניינים 1](#_Toc128122563)

[מבוא 3](#_Toc128122564)

[הרקע לפרויקט 3](#_Toc128122565)

[מטרות המערכת 3](#_Toc128122566)

[מטרות על: 3](#_Toc128122567)

[מטרות נלוות: 3](#_Toc128122568)

[תיאור המערכת: 3](#_Toc128122569)

[גבולות המערכת 4](#_Toc128122570)

[עבור תת משתמש 4](#_Toc128122571)

[עבור משתמש עליון: 4](#_Toc128122572)

[עבור מנהל: 4](#_Toc128122573)

[סביבת פיתוח 4](#_Toc128122574)

[שפות תכנות: 4](#_Toc128122575)

[שכבות: 4](#_Toc128122576)

[פלטפורמות של הלקוחות: 5](#_Toc128122577)

[אתגרים מרכזיים: 5](#_Toc128122578)

[חידושים התאמות ועיצוב: 5](#_Toc128122579)

[ניתוח המערכת 6](#_Toc128122580)

[מצב קיים: 6](#_Toc128122581)

[מטרות מערכת עתידיות 6](#_Toc128122582)

[מצב מערכת עתידית 6](#_Toc128122583)

[תרשים ERD 6](#_Toc128122584)

[עץ תהליכים 7](#_Toc128122585)

[בסיס הנתונים (database) 8](#_Toc128122586)

[תרשים DSD 8](#_Toc128122587)

[טבלאות ראשיות 8](#_Toc128122588)

[טבלת Users: 8](#_Toc128122589)

[טבלת Inventory 8](#_Toc128122590)

[טבלת BorrowedItems 9](#_Toc128122591)

[טבלת SubUsers 9](#_Toc128122592)

[טבלת Orders 9](#_Toc128122593)

[צד שרת 11](#_Toc128122594)

[המחלקות המטפלות בנתונים 11](#_Toc128122595)

[תרשים UML 11](#_Toc128122596)

[הקוד: 11](#_Toc128122597)

[שכבת שירותי רשת 13](#_Toc128122598)

[מימוש InventoryFuncs.asmx 13](#_Toc128122599)

[הקוד לBorowwingsDB.asmx 16](#_Toc128122600)

[הקוד לSubUsersServ.asmx 19](#_Toc128122601)

[הקוד לUserDBServ.asmx 26](#_Toc128122602)

[שירותי רשת חיצוניים 31](#_Toc128122603)

[היגיון עסקי (Business Logic) 32](#_Toc128122604)

[קוד של מחלקות נבחרות: 32](#_Toc128122605)

[הקוד של מחלקת User 32](#_Toc128122606)

[הקוד למחלקת Order 34](#_Toc128122607)

[הקוד למחלקת Borrow 35](#_Toc128122608)

[הקוד למחלקת InventoryRow 36](#_Toc128122609)

[הקוד למחלקת SubUser 38](#_Toc128122610)

[צד משתמש 40](#_Toc128122611)

[ממשקי משתמש 40](#_Toc128122612)

[הרחבות: 40](#_Toc128122613)

[מדריך למשתמש 43](#_Toc128122614)

[מדריך למשתמש העליון 43](#_Toc128122615)

# מבוא

## הרקע לפרויקט

שם הפרויקט: StockWizard

תיאור קצר של הפרויקט: הפרויקט נותן לארגונים גדולים לנהל בקלות מערכות מלאי גדולות ומערכות הוצאה, החזרת ציוד (ניהול השאלות מלאי). יש באפליקציה שני סוגי משתמשים: משתמש תחתון, ומשתמש עליון (מנהל המלאי). משתמש עליון הוא מנהל המלאי של הארגון: הוא יכול להוסיף ולהוריד מלאי, להוציא ציוד (ליצור השאלות), וליצור משתמשים תחתונים השייכים אליו. משתמש תחתון שייך למשתמש עליון מסויים, הוא יכול לראות חלק מרשימת המלאי של המשתמש העליון אליו הוא שייך (רק שמות הפריטים), ולשלול הזמנות ציוד למשתמש העליון. המשתמש העליון מקבל את ההזמנות ששלחו המשתמשים התחתונים, והוא יכול לדחות אותם או לקבל אותם.

קהל היעד: ארגונים גדולים שצריכים לנהל מלאים גדולים והשאלות רבות.

הסיבות לבחירת הנושא: בתור ראש מחסן בשבט הצופים שלי([שחר גדרה](http://צופים.com)), אני מבין את האתגר שבניהול מלאי גדול וכמות גדולה של השאלות ציוד במקביל, לכן רציתי ליצור אפליקציה שתסייע בבעיה זו.

## מטרות המערכת

### מטרות על:

* ניהול יעיל של מלאים גדולים
* ניהול ורישום מהיר וקל של השאלות ציוד
* הזמנה קלה של ציוד ממנהל המלאי

### מטרות נלוות:

* עיצוב קל ומובן- הסברים בתוך האפליקציה, עיצוב עקבי, שימוש בכמה שפחות טקסט וכמה שיותר אייקונים.
* צמצום כמות העמודים באפליקציה- שלא יווצר מבוך של מסכים

## תיאור המערכת:

המערכת מכילה מספר רכיבים: 4 שרותי רשת, אפליקציית צד לקוח, ומסד נתונים המכיל 5 טבלאות מידע.

שירותי רשת:

* שירות שאחראי על רשימות המלאי
* שירות שאחראי על ניהול המשתמשים העליונים
* שירות שאחראי על ניהול השאלות הציוד
* שירות שאחראי על ניהול המשתמשים התחתונים וההזמנות שלהם

טבלאות:

* טבלת הזמנות
* טבלת השאלות
* טבלת משתמשים עליונים
* טבלת משתמשים תחתונים
* טבלת מלאי

עבור משתמש מסוג מנהל מלאי:

בצד הלקוח, כאשר המשתמש מוסיף פריטים לרשימת המלאי שלו, הפריט נשלח לשירות הרשת שאחראי על מלאי, שמוסיף אותו למסד הנתונים. שירות רשת זה מקבל גם פקודות מחיקה, ועדכון של פריטים מצד הלקוח.

מנהל המלאי יכול להוסיף, לערוך, או למחוק מצד הלקוח השאלות של פריטים, בקשות אלו נשלחות לשירות השרת שאחראי על ההשאלות שבתורו, מבצע את שינויים אלו במסד הנתונים.

עבור משתמש מסוג תחתון:

בצד הלקוח, המשתמש יכול לראות את הפריטים שבמלאי של מנהל המלאי שלו ויכול להזמין פריטים. צד הלקוח שולח את משימות אלו אל שירות השרת שאחראי על המשתמשים התחתונים שבתורו, מבצע שינויים אלו במסד הנתונים.

## גבולות המערכת

### עבור תת משתמש

יכול לקבל חלק מהמידע בטבלת המלאי, ולשלוח הזמנות, זמינים לו סך הכול 2 מסכים:

1. שליחת הזמנות מלאי למשתמש העליון
2. צפייה בהזמנות שהוא כבר שלח.

### עבור משתמש עליון:

משתמש עליון יכול להוסיף, לשנות, ולמחוק רשומות ברשימת המלאי שלו, הוא יכול להוסיף משתמשים תחתונים שהשייכים אליו ולשנות את פרטייהם, הוא יכול הוסיף השאלות ידנית או לקבל הזמנות מהמשתמשים התחתונים שלו ולהוסיף אותם להשאלות. סך הכול זמינים לו 7 עמודים:

1- התחברות

2- הרשמה

3- שינוי פרטי

4- הוספת משתמשים

5- ניהול מלאי

6- ניהול השאלות

7- ניהול הזמנות

### עבור מנהל:

למשתמש מנהל (system admin) יש גישה למסך בו הוא יכול לנהל את כל המשתמשים הגבוהים(לשנות פרטים, להוסיף, למחוק).

## סביבת פיתוח

כסביבת פיתוח, השתמשתי בvisual studio 2017 community addition של Microsoft לפיתוח אפליקציית הלקוח ושרותי הרשת. לצורך ניהול מסדי הנתונים (sequential relational databases) השתמשתי בSQL server express 2014 של Microsoft.

## שפות תכנות:

לצורך פיתוח האפליקציה השתמשתי בשפות הבאות:

* בC#: לצורך כתיבת הקוד המנהל אצ האירועים בצד הלקוח ובצד השרת.
* בXAML: לצורך עיצוב עמודי האפליקציה (צד לקוח)- זוהי שפת תגיות לעיצוב ממשקים ויזואלים (דומה לHTML).
* בHTML: השתמשתי בHTML לצורך עיצוב עמוד האודות על האפליקציה.
* בSQL: לצורך ניהול הנתונים והמידע של המשתמשים בשרותי השרת.

## שכבות:

בפרוייקט שלי יש 3 שכבות עיקריות: שכבת לקוח (האפליקציה הויזואלית שאיתה מתממשק המשתמש), שכבת שרת – שם מבוצעות פעולות שינוי המידע לפי בקשות המתקבלות מצד הלקוח בין אם בשירותי רשת פנימיים (ששיכים לאותה חברה שמפתחת את צד הלקוח) או בחיצוניים (שירותי רשת של חברות אחרות), שכבת נתונים- זהו מסד הנתונים שמכיל את כל המידע של המשתמשים ושל המערכת, ומאחסן אותם.

## פלטפורמות של הלקוחות:

הלקוחות משתמשים בפלטפורמת Windows, זאת כיוון שהאפליקציה פותחה עבור פלטפורמת UWP (universal windows platform) שהיא הפלטפורמה שבה הכי קל לכתוב אלפיקציות native למחשבי windows- שהם סוג המחשבים הכי נפוץ.

כלומר- הפלטפורמה UWP נבחרה עקב מספר המשתמשים הרב שיכול להשתמש בפלטפורמה זו.

## אתגרים מרכזיים:

כמו שכבר כתבתי, הצורך בפרויקט כזה בא לענות על צורך בו נתקלתי בתור ראש מחסן בצופים. התחלתי את תהליך הפיתוח עם אפליקציה רזה שנועדה לניהול מלאי בלבד, ולאט לאט הוספתי פיצ'רים נוספים למשתמש העליון- הוספת השאלות, ייבוא המלאי מקובץ csv, חיפוש פריטים חדשים לקנייה בגוגל וכו'...

האתגר המרכזי בו נתקלתי בפיתוח הפרוייקט היה שינוי המערכת כך שהיא תתאים לשני סוגי משתמשים (עליון ותחתון), לא תכננתי את שינוי זה בהתחלה ולקח זמן לארגן מחדש את צד השרת והלקוח כך שיתאים לשינוי זה.

אתגר נוסף בו נתקלתי הוא הצגת טבלאות אינטואיטיבות שהמשתמש יכול להשתמש בהן (כיוון שמטבע הפרוייקט יש צורך בטבלאות ורשימות רבות). היה מידע מוגבל בנושא באינטרנט ולאחר מחקר רב למדתי איך עובדים עם פקד ListView ואיך משתמשים בקשירה דו צדדית של מידע לטבלה (two way data binding) כדי שהמשתמש יוכל לשנות ישירות את המידע בעצמו.

## חידושים התאמות ועיצוב:

לאחר התקדמות רבה בפרויקט, למדתי רבות על שיטות עיצוב ופריסת התוכן באפליקציה בuwp. לאחר העמודים הראשונים, למדתי איך לפרוס נתונים בצורה שתתאים לכל מסך בעזרת עמודות ושורות בפקד Grid, איך ליצור רצף של תוכן אנכי ואופקי בעזרת פקד StackLayout ואיך להוריד את כמות המסכים בפרוייקט בעזרת עיצוב מתקדם של ContentDialogs וכו'...

# ניתוח המערכת

## מצב קיים:

כרגע, במערכת יש שני סוגי משתמשים: עליון ותחתון.

משתמש עליון יכול: להירשם ולהיכנס למשתמש שלו, ליצור ולערוך רשימת מלאי, להוסיף השאלות של פריטים מהמלאי, להוסיף משתמשים תחתונים השייכים אליו, ולקבל ולאשר או לדחות הזמנות המשתמשים התחתונים השייכים אליו.

משתמש תחתון: משתמש תחתון יכול לראות רשימה של הפריטים הזמינים במלאי של המשתמש העליון אליו הוא שייך, לבצע הזמנות, ולראות את מצב ההזמנות ששלח.

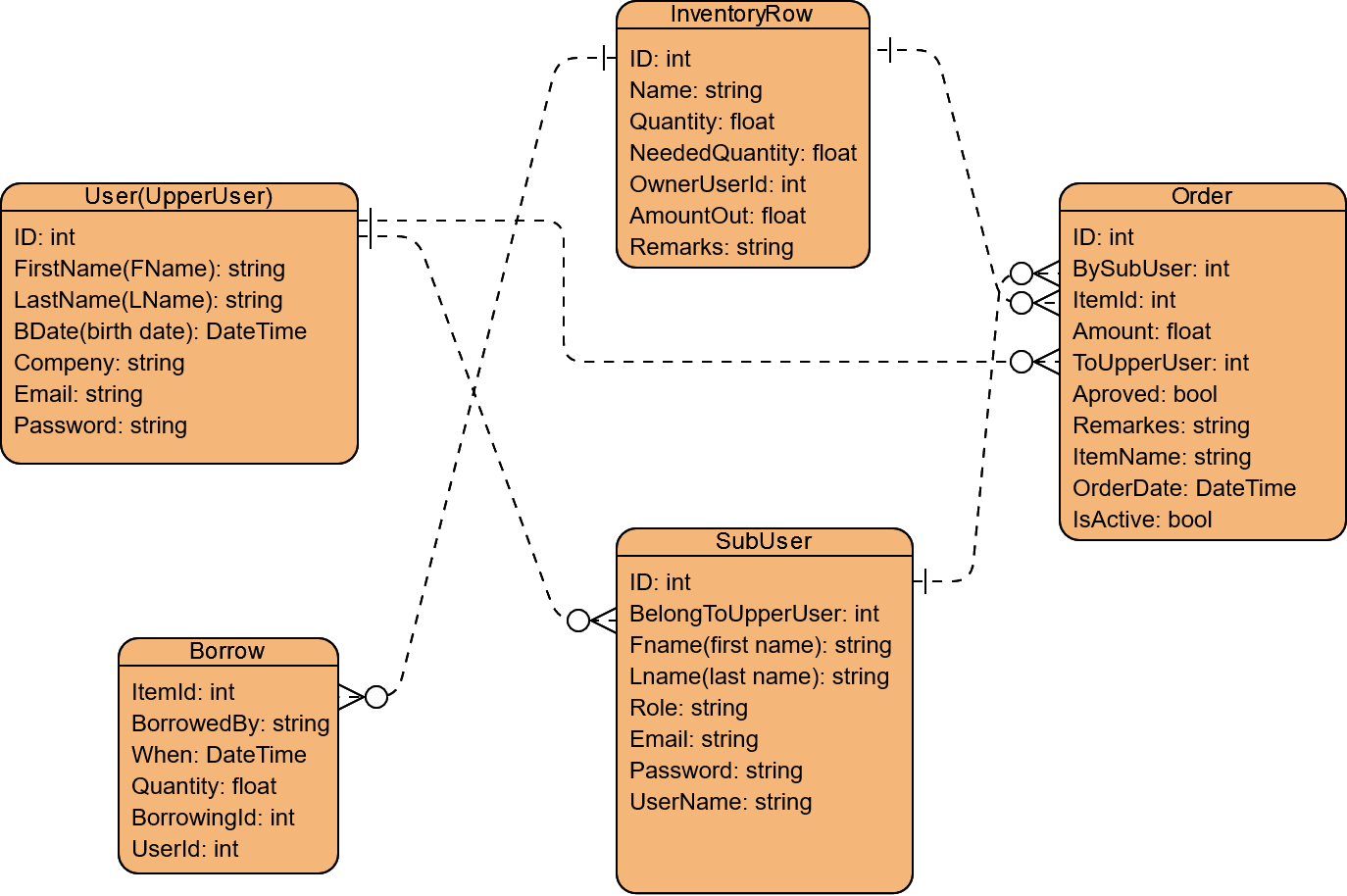
## מטרות מערכת עתידיות

לעתיד, הייתי שם למטרה להוסיף עוד אופציות מעקב של המשתמש התחתון אחרי ההזמנות שלו. כרגע, לאחר שהזמנה מאושרת והופכת להשאלת ציוד, היא רק מופיעה למשתמש כמאושרת ואין לו אופציה לעקוב אחרי השאלות הציוד הפעילות שלו.

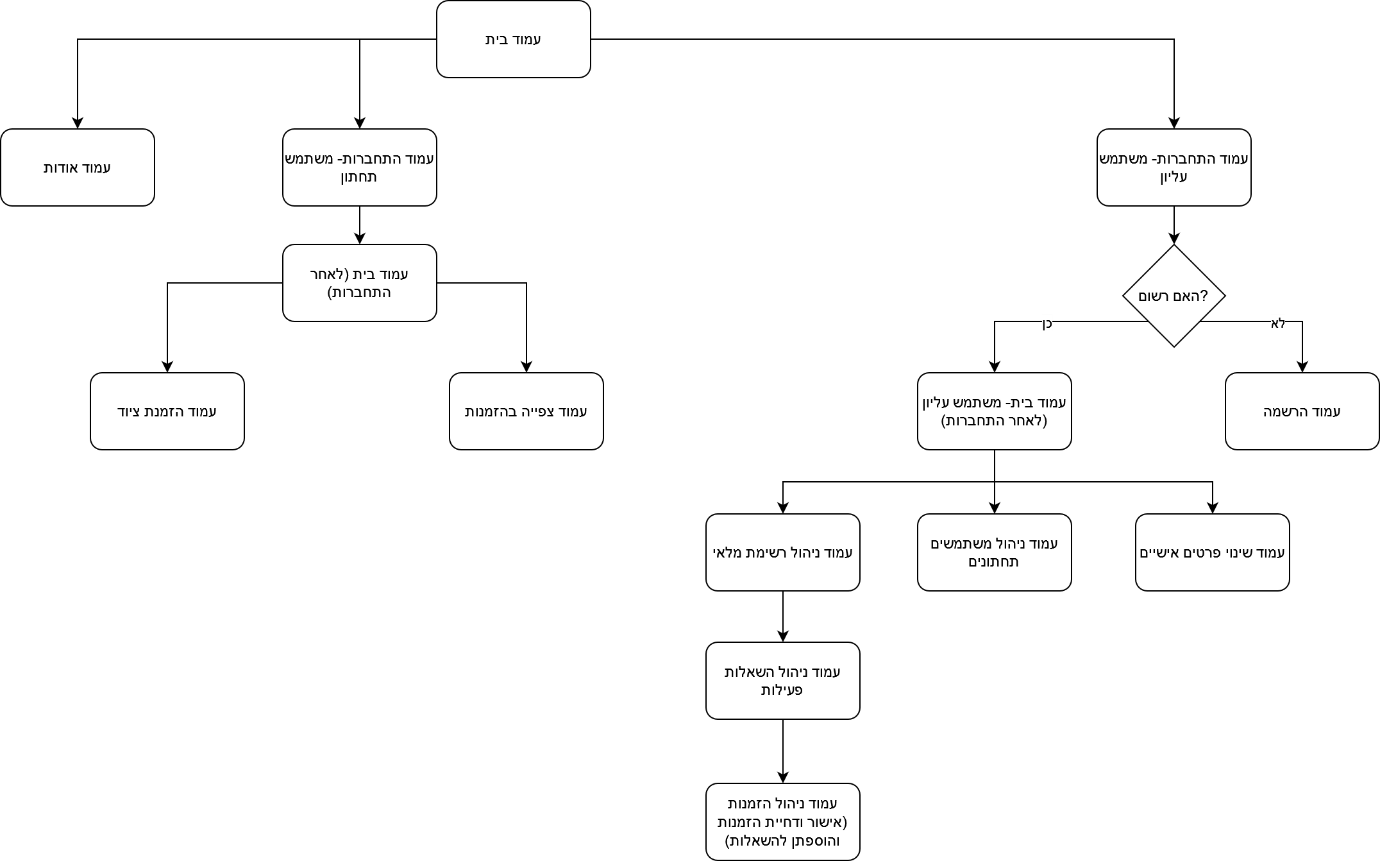
## מצב מערכת עתידית

במערכת עתידית אידיאלית, המשתמש התחתון יוכל לעקוב אחרי ההשאלות הפעילות שלו ומצבן (כאשר היום הוא יכול לעקוב רק אחרי ההזמנות).

## תרשים ERD

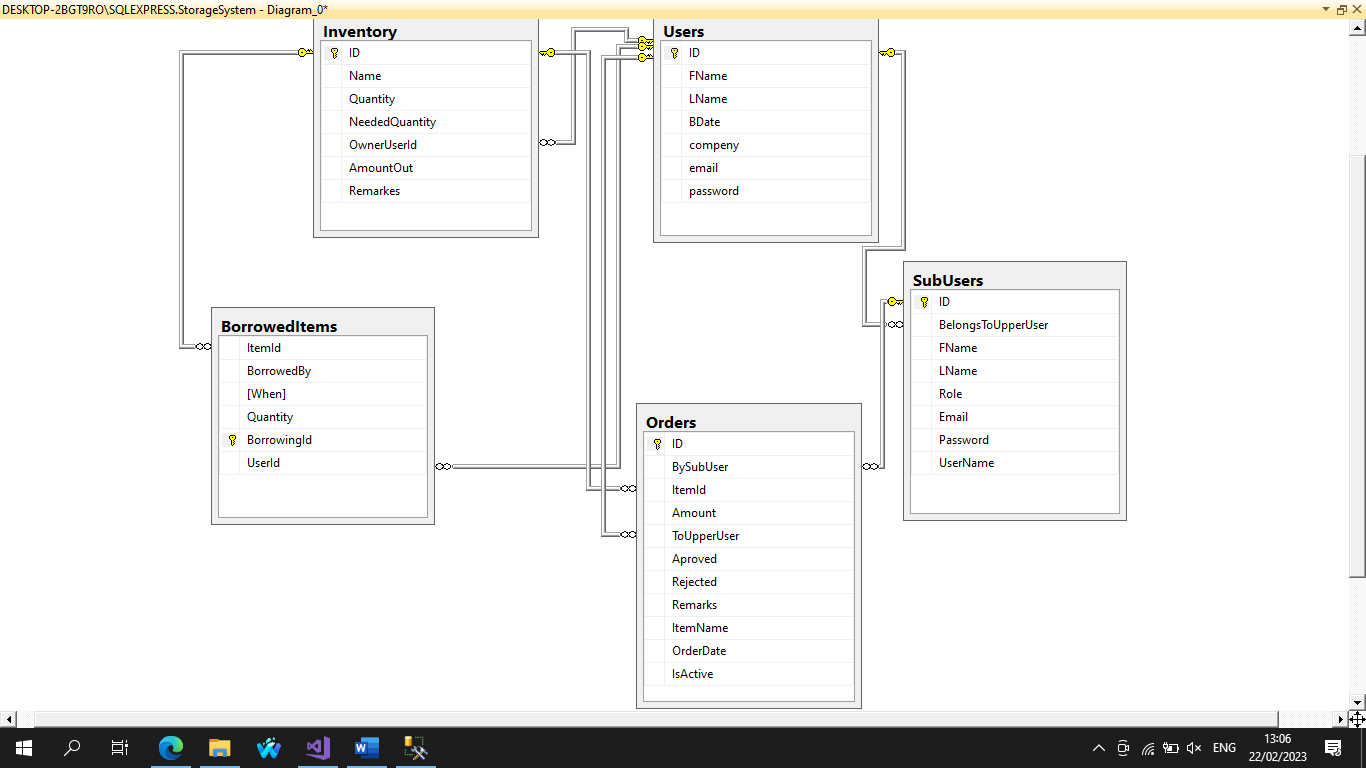


## עץ תהליכים



# בסיס הנתונים (database)

## תרשים DSD



## טבלאות ראשיות

### טבלת Users:

כל רשומה בטבלה מכילה את הפרטים של משתמש עליון במערכת.

הטבלה מכילה מספר שדות:

1. ID- המפתח הראשי והמיוחד של כל משתמש, שדה זה הוא שדה חובה שנקבע אוטומטית לכל משתמש לצורך זיהויו במערכות.
2. fName- שדה טקסט עם השם הפרטי של כל משתמש עליון
3. Lname- שדה טקסט המכיל את השם משפחה של כל משתמש עליון
4. Bdate- שדה מסוג datetime המכיל את תאריך הלידה של המשתמש
5. Compeny- שדה טקסט שמכיל את שם החברה שמייצג המשתמש (במחסן של מי הוא עובד?)
6. Email- שדה טקסט המכיל את כתובת הדוא"ל של המשתמש.
7. Password- שדה טקסט המכיל את קוד הגישה של המשתמש העליון לשירות.

### טבלת Inventory

כל רשומה בטבלה זו מכילה את הפרטים על פריט אחד ברשימת מלאי של אחד ממשתמשי השירות.

הטבלה מכילה את השדות הבאים:

1. ID- שדה ייחודי לכל רשומה, משומש לזיהוי הרשומה והבחתה מאחרות. מיוצר בצורה אוטומטית.
2. Name- שדה טקסט המכיל את השם של הפריט שמיוצג ברשומה.
3. Quantity- שדה מספרי (מסוג עשרוני) שמכיל את הכמות הקיימת במלאי מכל מוצר
4. NeededQuantity- שדה מסוג מספרי (עשרוני) המכיל את הכמות שיש בה צורך במלאי (כמה צריך שתמיד יהיה מפריט זה).
5. OwnerUserId- מייצג את המשתמש שאליו משוייך פריט זה . רשומה זו מקושרת בקשר יחיד אל רבים אל ID בטבלה Users.
6. AmountOut- שדה מסוג מספרי (עשרוני). מייצג את הכמות מפריט זה שכרגע נמצאת מחוץ למחסן (הושאלה החוצה).
7. Remarkes- שדה טקסט המייצג את ההערות של המשתמש על פריט זה (האם יש משהו שהוא רוצה לכתוב על הפריט).

### טבלת BorrowedItems

כל רשומה בטבלה זו מייצגת השאלה של פריט (הוצאתו אל מחוץ למחסן אל אדם אחר).

השדות בטבלה זו:

1. ItemId- שדה זה מייצג את המספר המזהה של הפריט שהושאל. שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת Inventory.
2. BorrowedBy- זהו שדה טקסט שבו השם של האדם ששאל את הפריט מהמלאי.
3. [When]- זהו שדה datetime שמכיל את התאריך שבו בוצעה ההשאלה.
4. Quantity- זהו שדה מספרי (עשרוני) המכיל את הכמות מאותו הפריט שהושאלה.
5. BorrowingId- זהו שדה מספרי, הוא השדה המזהה של כל רשומה, והוא מייוצר בצורה אוטומטית לכל רשומה.
6. UserId- זהו השדה שאומר אל איזה משתמש עליון שייכת ההשאלה (מהמלאי של מי יצא הפריט). שדה זה מקושר בקשר רבים אל יחיד אל השדה ID בטבלת Users.

### טבלת SubUsers

כל רשומה בטבלה זו מייצגת משתמש תחתון (משתמש שמשוייך למשתמש עליון).

השדות בטבלה הם:

1. ID- שדה זה הוא המזהה הייחודי של כל משתמש תחתון, הוא מיוצר אוטומטית עבור כל משתמש תחתון.
2. BelongsToUpperUser- שדה זה הוא שדה מספרי המייצג את המשתמש העליון אליו משתייך המשתמש התחתון. שדה זה מקושר בקשר רבים אל יחיד אל שדה ID בטבלת Users.
3. FName- זהו שדה טקסט המכיל את השם הפרטי של המשתמש.
4. LName- זהו שדה טקסט המכיל את שם המשפחה של המשתמש.
5. Role- זהו שדה טקסט המכיל את תפקיד המשתמש בארגון.
6. Email- זהו שדה טקסט המכיל את הדוא"ל של המשתמש.
7. Password- זהו שדה טקסט המכיל את קוד הגישה של המשתמש.
8. UserName- זהו שדה טקסט ייחודי לכל משתמש (אין שתי משתמשים עם אותו שם משתמש). הוא מייצג את שם המשתמש של המשתמש.

### טבלת Orders

כל רשומה בטבלה זו מייצגת הזמנת ציוד של משתמש תחתון אל משתמש עליון (מחבר בארגון אל מנהל המלאי).

השדות בטבלה הם:

1. ID- שדה מספרי ייחודי לכל רשומה, מיוצר אוטומטית לכל רשומה חדשה.
2. BySubUser- שדה מספרי המייצג ממי הגיע ההזמנה, שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת SubUsers.
3. ItemId- שדה מספרי זה מייצג את הפריט שהמשתמש התחתון רוצה להזמין. שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת Inventory.
4. Amount- שדה מספרי(עשרוני), שדה זה מייצג את הכמות מאותו הפריט שהמשתמש רוצה לשאול.
5. Aproved- שדה בוליאני(bit), שדה זה מייצג את התשובה לשאלה: האם מנהל המלאי אישר את ההזמנה?
6. Rejected- שדה בוליאני(bit), שדה זה מייצג את התשובה לשאלה: האם מנהל המלאי דחה את ההזמנה?
7. Remarks- שדה טקסט, שדה זה מכיל את ההערות שיש למשתמש התחתון אל העליון בנוגע להזמנתו.
8. ItemName- שדה טקסט, לצורך פישוט הגישה לשמות הפריטים, טבלה זו מכיל עותק של השם של הפריט שבהזמנה.
9. OrderDate- שדה תאריך (datetime), שדה זה מכיל את התאריך בו הוזמנה ההזמנה.
10. IsActive- שדה בוליאני (bit), עונה על השאלה: האם ההשאלה עוד פעילה ? (האם היא עוד רלוונטית או שניתן להתעלם ממנה).

# צד שרת

## המחלקות המטפלות בנתונים

### תרשים UML

למחלקת Connection שלי אין תכונות (יש רק פעולות), ולכן אי אפשר ליצור לה תרשים UML.

### הקוד:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

/// <summary>

/// Represents a connection to a SQL database.

/// </summary>

namespace WebServ1

{

    public class Connection

    {

        //תכונות גלובליות שנדרשות לשימוש בחלקים שונים בקוד

        //global properties reqired in different parts of the code

        private SqlConnection con;

        private SqlDataReader reader;

        private DataTable tbl;

        private SqlDataAdapter adapter;

        private string tableName;

        private DataSet ds;

        //מאתחל את החיבור לשרת

        //initelize connection

        public Connection(string conString)

        {

            con = new SqlConnection(conString);

        }

        //פותח את החיבור למסד הנתונים

        //opens connections

        public void openCon()

        {

            con.Open();

        }

        //סוגר את החיבור למסד הנתונים

        //נדרש בשיטה המקושרת

        //colses connection

        public void CloseCon()

        {

            con.Close();

        }

        //מחזיר את עצם החיבור

        //returns connection

        public SqlConnection GetCon()

        {

            return con;

        }

        //מבצע שאילתת שינוי בחיבור קיים

        //exequte a qury in a connected method

        public Boolean ExequteNoneQury(string noneQury)

        {

            SqlCommand comd = new SqlCommand(noneQury, con);

            int a = comd.ExecuteNonQuery();

            comd.Dispose();

            return a > 0;

        }

        //מבצע שאילתת הצגה ומחזיר קורא מידע בשיטה המקושרת

        // Executes a select query on the connected database and returns a reader.

        public SqlDataReader ExequteQury(string qury)

        {

            SqlCommand comd = new SqlCommand(qury, con);

            reader = comd.ExecuteReader();

            return reader;

        }

        //מחזיר טבלת מידע מלאה במידע בהנחה שקורא כבר נוצר

        //returns a datatable loded by pre created reader

        public DataTable GetDataTable()

        {

            tbl = new DataTable();

            tbl.Load(reader);

            return tbl;

        }

        //מקבל שאילתת הצגה ומחזיר טבלה בשיטה המקושרת

        //recives a select qury and returns datatable

        public DataTable GetDataTable(string qury)

        {

            SqlCommand comd = new SqlCommand(qury, con);

            reader = comd.ExecuteReader();

            tbl = new DataTable();

            tbl.Load(reader);

            return tbl;

        }

        //מקבל שאילתת הצגה ומחזיר מתאם מידע בשיטה הלא מקושרת

        //recives a select qury and returns SqlDataAdapter

        public SqlDataAdapter GetAdapter(string qury)

        {

            SqlCommand comd = new SqlCommand(qury, con);

            adapter = new SqlDataAdapter(comd);

            return adapter;

        }

        //מחזיר סט מידע בהנחה שקורא כבר נוצר בשיטה הלא מקושרת

        //Returns DataSet, asumming SqlDataAdapter has alredy been created

        public DataSet GetDataSet(string tableName)

        {

            if (adapter != null)

            {

                DataSet ds = new DataSet();

                adapter.Fill(ds, tableName);

                return ds;

            }

            else

            {

                return null;

            }

        }

        //מקבל שאילתת הצגה ומחזיר סט מידע בשיטה הלא מקושרת

        //takes a select qury, and retuns the qury's respons table

        public DataSet GetDataSet(string tableN, string qury)

        {

            tableName = tableN;

            SqlCommand comd = new SqlCommand(qury, con);

            adapter = new SqlDataAdapter(comd);

            ds = new DataSet();

            adapter.Fill(ds, tableName);

            return ds;

        }

        //מקבל שורת מידע ומכניס אותה לטבלה המתאימה בהנחה שסט מידע כבר נוצר קודם

        //takes a DataRow and inserts it into the coresponding table, assuming dataSet has alredy been previusly created

        public void InsertDataRow(DataRow dr)

        {

            if (ds != null)

            {

                ds.Tables[0].Rows.Add(dr);

            }

            SqlCommandBuilder builder = new SqlCommandBuilder(adapter);

            adapter.InsertCommand = builder.GetInsertCommand();

            adapter.UpdateCommand = builder.GetUpdateCommand();

            adapter.Update(ds, tableName);

        }

        //לוקח סט מידע שעבר שינויים ומעדכן את שינויים אלו במסד הנתונים

        //takes a motifyed DataSet, and applys motifications to db

        public void Update(DataSet a)

        {

            SqlCommandBuilder builder = new SqlCommandBuilder(adapter);

            adapter.InsertCommand = builder.GetInsertCommand();

            adapter.UpdateCommand = builder.GetUpdateCommand();

            adapter.DeleteCommand = builder.GetDeleteCommand();

            adapter.Update(a, tableName);

        }

    }

}

## שכבת שירותי רשת

יש בפרויקט שלי 4 שרותי רשת פנימיים (שנכתבו על ידי אותו "ארגון" שכתב את האפליקציה):

1. UserDBServ.asmx- שירות רשת זה נועד לשילפת והכנסת מידע אודות משתמשים ממסד הנתונים. השימושים העיקריים בו הם לחיבור המשתמשים (אימות ססמה ודוא"ל) ורישום משתמשים.
2. SubUsersServ.asmx- שירות רשת זה נועד להתממשקות מול מידע ופעולות הקשורים למשתמשים תחתונים. שירות רשת זה בשימוש על מנת: לרשום משתמשים, למחוק משתמשים, לחבר משתמשים, ולשלוח הזמנות ממשתמש תחתון אל מנהל המלאי שלו.
3. BorowwingsDB.asmx- שירות רשת זה נועד להוספת ומחיקת השאלות מלאי על ידי המשתמש העליון.
4. InventoryFuncs.asmx- שירות רשת זה נועד להוספת, מחיקת ועריכת פרטי מלאי מרשימת המלאי של המשתמש העליון.

### מימוש InventoryFuncs.asmx

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Data;

using System.Data.SqlClient;

using WpfApp1;

using System.Net;

namespace InventoryServ

{

    /// <summary>

    /// a collection of functions to inteact with inventory items in db

    /// </summary>

    [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

    [System.ComponentModel.ToolboxItem(false)]

    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

    // [System.Web.Script.Services.ScriptService]

    public class InventoryFuncs : System.Web.Services.WebService

    {

        public String constr = "Server = '"+ Dns.GetHostName() +"\\SQLEXPRESS'; Database = StorageSystem; Trusted\_Connection = True; ";

        [WebMethod]

        /// <summary>

        /// returns a DataTable Contaning all the inventory items that belongs to the user. For sequrity user needs to be authenticated with email and password.

        /// מחזיר טבלת מידע עם כל הפריטים ברשימת המלאי של המשתמש

        /// </summary>

        public DataTable GetInventoryUserDataTable(int userId, string email, string password)

        {

            var serv = new UserServ.UserDBServSoapClient();

            var isAllowed = serv.IsUserPermitted(new UserServ.User() { Password = password, Email = email });

            if (isAllowed)

            {

                Connection con = new Connection(constr);

                con.openCon();

                DataSet ds = con.GetDataSet("inventory", "Select \* From inventory where OwnerUserId= " + userId);

                DataTable dt = ds.Tables[0];

                return dt;

            }

            return null;

        }

        [WebMethod]

        ///<summery>

        ///takes a motifyed InventoryRow item, the function applyes motifications to the item's coresponding row in db and returns wether the motifications were secsussfull.

        ///מקבל שורת מלאי שעברה שינויים ומכיל את השינויים לשורות המתאימות במסד הנתונים

        /// </summery>

        public bool changeInventoryRow(InventoryRow inventoryRow)

        {

            Connection con = new Connection(constr);

            con.openCon();

            bool a = con.ExequteNoneQury("update Inventory Set Name = N'" + inventoryRow.Name + "', Quantity = " + inventoryRow.Quantity.ToString() + ", NeededQuantity = " + inventoryRow.NeededQuantity.ToString() + ", Remarkes = N'"+ inventoryRow.Remarkes +"' Where ID = " + inventoryRow.ID.ToString() + " ;");

            con.CloseCon();

            if (a)

            {

            }

            return a;

        }

        [WebMethod]

        ///<summery>

        ///Takes an ItemId and deletes the coresponding item from db. for sequrity perpose, email and password are reqired.

        ///מקבל פרטי משתמש ומספר מזהה של שורת מלאי ומוחק את שורת המלאי בהנחה שהיא שייכת למשתמש שאת פרטיו הכניסו

        /// </summery>

        public bool DeleteInventoryRow(int id, string email, string password)

        {

            var serv = new UserServ.UserDBServSoapClient();

            var isAllowed = serv.GetFullUser(new UserServ.User() {Email= email, Password= password }).ID == GetOwnerID(id);

            if (isAllowed)

            {

                Connection con = new Connection(constr);

                con.openCon();

                bool a = con.ExequteNoneQury("Delete from Inventory where id = " + id + ";");

                con.CloseCon();

                return a;

            }

            else { return false; }

        }

        //returns the user that owns the item with the given itemId

        //מחזיר את מספר המשתמש שבבעלותו נמצא הפריט בעל מספר הפריט שהוכנס

        private int GetOwnerID(int ItemId)

        {

            Connection con = new Connection(constr);

            con.openCon();

            var ds = con.GetDataSet("Tbl", "Select OwnerUserId from Inventory where id =" + ItemId + ";");

            return int.Parse(ds.Tables[0].Rows[0][0].ToString());

        }

        [WebMethod]

        ///<summery>

        ///creates a new empty item, returns the new item's id. For sequrity, email and password are reqiured.

        ///יוצר פריט ריק חדש ברשימת המלאי של המשתמש שאת פריטיו הכניסו ומחזיר את המספר המזהה של הפריט שנוצר

        /// </summery>

        public int getNewItemId(int UserId, string email, string password)

        {

            var serv = new UserServ.UserDBServSoapClient();

            var isAllowed = serv.IsUserPermitted(new UserServ.User() { Password = password, Email = email });

            if (isAllowed)

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("inventory", "select \* from inventory");

                DataRow dr = ds.Tables[0].NewRow();

                dr[4] = UserId;

                con.InsertDataRow(dr);

                con.openCon();

                DataSet dataSet = con.GetDataSet("inventory", "select \* from Inventory where OwnerUserId =" + UserId.ToString() + ";");

                int lastIndex = dataSet.Tables["inventory"].Rows.Count - 1;

                DataRow lastRow = dataSet.Tables["inventory"].Rows[lastIndex];

                con.CloseCon();

                return int.Parse(lastRow[0].ToString());

            }

            else { return int.MinValue; }

        }

    }

}

### הקוד לBorowwingsDB.asmx

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Net;

using WpfApp1;

using System.Data;

using InventoryServ;

namespace InventoryServ

{

    /// <summary>

    ///contains a few methods to interact with all active landings

    /// </summary>

    [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

    [System.ComponentModel.ToolboxItem(false)]

    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

    // [System.Web.Script.Services.ScriptService]

    public class BorowwingsDB : System.Web.Services.WebService

    {

        public static String constr = "Server = '" + Dns.GetHostName() + "\\SQLEXPRESS'; Database = StorageSystem; Trusted\_Connection = True; ";

        /// <summary>

        /// calculates the amount of items borrowed of a given item id

        /// מחשב את כמות הפריטים המושאלים מעצם בעל מספר מזהה נתון לפי סכימת השאלותיו בטבלת ההשאלות

        /// </summary>

        [WebMethod]

        public float CalcAmountOut(int itemId)

        {

            Connection con = new Connection(constr);

            DataSet ds = con.GetDataSet("lands", "select Quantity from BorrowedItems where ItemId=" + itemId.ToString() + ";");

            float amount = 0;

            foreach (DataRow dr in ds.Tables[0].Rows)

            {

                amount += float.Parse(dr[0].ToString());

            }

            return amount;

        }

        [WebMethod]

        ///<summary>

        ///take an item id and updates this item amount out in db according to sum of his landings

        ///מעדכן את כמות הפריטים המושאלים מעצם מסויים ברשומה שלו במסד הנתונים לפי סכימת השאלותיו בטבלת ההשאלות

        ///</summary>

        public bool UpdateAmountOut(int itemId)

        {

            Connection con = new Connection(constr);

            DataSet ds = con.GetDataSet("lands", "select Quantity from BorrowedItems where ItemId=" + itemId.ToString() + ";");

            float amount = 0;

            foreach (DataRow dr in ds.Tables[0].Rows)

            {

                amount += float.Parse(dr[0].ToString());

            }

            con.openCon();

            var s = "Update Inventory Set AmountOut = " + amount.ToString() + " where ID = " + itemId.ToString() + ";";

            bool a = con.ExequteNoneQury(s);

            con.CloseCon();

            return a;

        }

        [WebMethod]

        ///<summary>

        ///update user's item's amount out in db acoording to each item's landings in db.

        ///מעדכן את כמות הפריטים המושאלים מכל פריט ברשימת המלאי של משתמש בעל מספר זיהוי נתון לפי סכימת השאלותיו של כל פריט בטבלת ההשאלות

        ///</summary>

        public bool UpdateUserAmountOut(int UserId)

        {

            Connection con = new Connection(constr);

            DataSet ds = con.GetDataSet("Items", "select ID from Inventory where OwnerUserId=" + UserId.ToString() + ";");

            bool isOk = true;

            foreach (DataRow dr in ds.Tables[0].Rows)

            {

                isOk = isOk && UpdateAmountOut(int.Parse(dr["ID"].ToString()));

            }

            return isOk;

        }

        [WebMethod]

        ///<summary>

        ///adds landing with given properties and adds it to db

        ///מוסיף השאלה בעלת התכונות המתקבלות מהמשתמש למסד הנתונים

        ///</summary>

        public int AddLending(int itemId, string lentForWho, DateTime whenBorowwed, float amountBorowwed, int userId)

        {

            Connection con = new Connection(constr);

            DataSet ds = con.GetDataSet("lands", "select \* from BorrowedItems where 0>1");

            DataRow dr = ds.Tables[0].NewRow();

            dr["ItemId"] = itemId;

            dr["BorrowedBy"] = lentForWho;

            dr["When"] = whenBorowwed;

            dr["Quantity"] = amountBorowwed;

            dr["UserId"] = userId;

            con.InsertDataRow(dr);

            ds = con.GetDataSet("lands1", "select Top 1 \* from BorrowedItems where ItemId =" + itemId.ToString() + "And BorrowedBy = N'"+ lentForWho.ToString() + "' And Quantity =" + amountBorowwed.ToString() + "  ORDER BY BorrowingId DESC");

            int id = int.Parse( ds.Tables[0].Rows[0]["BorrowingId"].ToString());

            return id;

        }

        [WebMethod]

        ///<summary>

        ///returns datatable of all of the user's active landings

        ///מחזיר טבלת מידע עם כל ההשאלות הפעילות של המשוייכות למשתמש

        ///</summary>

        public DataTable GetLandings(int UserId)

        {

            Connection con = new Connection(constr);

            DataSet ds = con.GetDataSet("lands", "select \* from BorrowedItems where UserId=" + UserId.ToString() + ";");

            return ds.Tables["lands"];

        }

        [WebMethod]

        ///<summary>

        ///returns a item with a given item id name in db

        ///מחזיר את שמו של הפריט מרשימת המלאי בעל מספר מזהה נתון

        ///</summary>

        public string getName(int itemId) {

            var con = new Connection(BorowwingsDB.constr);

            con.openCon();

            var ds = con.GetDataSet("item name", "select Name from Inventory where ID = " + itemId + ";");

            con.CloseCon();

            return ds.Tables["item name"].Rows[0]["Name"].ToString();

        }

        //מחזיר את מספר המשתמש שבבעלותו נמצא הפריט בעל מספר הפריט שהוכנס

        public int GetOwnerID(int ItemId)

        {

            Connection con = new Connection(constr);

            con.openCon();

            var ds = con.GetDataSet("Tbl", "Select OwnerUserId from Inventory where id =" + ItemId + ";");

            return int.Parse(ds.Tables[0].Rows[0][0].ToString());

        }

        [WebMethod]

        ///<summary>

        ///deletets a lending, returns wethwe the operation secusseded

        ///מוחק השאלה מחזיר האם הפעולה הצליחה

        /// </summary>

        public bool DeleteLanding(Borrow borrow, string email, string password)

        {

            var s = new UserServ.UserDBServSoapClient();

            var FullUser = s.GetFullUser(new UserServ.User() { Email = email, Password = password });

            if(FullUser.ID == GetOwnerID(borrow.ItemId))

            {

                var con = new Connection(constr);

                con.openCon();

                var qury = "DELETE FROM BorrowedItems WHERE BorrowingId = " + borrow.BorrowingId + ";";

                bool isok = con.ExequteNoneQury(qury);

                if (isok) {

                    UpdateAmountOut(borrow.ItemId);

                    return isok;

                }

                return isok;

            }

            else { return false; }

        }

        [WebMethod]

        ///<summary>

        ///updates a landing ditailes

        ///מעדכן את פרטי ההשאלה מחזיר האם העדכון עבד

        /// </summary>

        public bool updateBorrow(Borrow oldborrow, Borrow newBorrow, string email, string password)

        {

            var s = new UserServ.UserDBServSoapClient();

            var FullUser = s.GetFullUser(new UserServ.User() { Email = email, Password = password });

            if (FullUser.ID == GetOwnerID(oldborrow.ItemId))

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("lands", "select \* from BorrowedItems where BorrowingId =" + oldborrow.BorrowingId + ";");

                DataRow dr = ds.Tables[0].Rows[0];

                dr["ItemId"] = newBorrow.ItemId;

                dr["BorrowedBy"] = newBorrow.BorrowedBy;

                dr["When"] = newBorrow.When;

                dr["Quantity"] = newBorrow.Quantity;

                dr["UserId"] = newBorrow.UserId;

                con.Update(ds);

                return true;

            }

            else { return false; }

        }

        [WebMethod]

        public void updateBorrowTest()

        {

            updateBorrow(new Borrow() { ItemId = 368, UserId = 20, BorrowingId = 28 }, new Borrow() { ItemId = 368, UserId = 20, BorrowedBy = "meme", BorrowingId = 28, Quantity = 99, When = DateTime.Now }, "yv@v.com                      ", "123456789           ");

        }

    }

}

### הקוד לSubUsersServ.asmx

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Services;

namespace WebServ1

{

    /// <summary>

    /// Summary description for SubUsersServ

    /// </summary>

    [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

    [System.ComponentModel.ToolboxItem(false)]

    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

    // [System.Web.Script.Services.ScriptService]

    public class SubUsersServ : System.Web.Services.WebService

    {

        //יוצר את המחרוזת חיבור שסביר שבשימוש לפי ברירת המחדל של אס קיו אל סרוור

        //creates the tipical constring according to sqlserver express defaults

        public String constr = "Server = '" + Dns.GetHostName() + "\\SQLEXPRESS'; Database = StorageSystem; Trusted\_Connection = True; ";

        [WebMethod]

        ///<summary>

        ///מקבל עצם משתמש תחתון ומכניס משתמש מתאים למסד הנתונים

        ///creates a new sub user in db from a subuser object

        /// </summary>

        public int createSubUser(SubUser subUser, User UpperUser)

        {

            var usdb = new UserDBServ();

            if (usdb.IsUserPermitted(UpperUser))

            {

                var con = new Connection(constr);

                con.openCon();

                String nqury = "Insert into SubUsers (BelongsToUpperUser, FName, LName, Role, Email, Password, UserName) Values (" + subUser.BelongsToUpperUser + ", '" + subUser.FName + "', '" + subUser.LName + "', '" + subUser.Role + "', '" + subUser.Email + "', '" + subUser.Password + "', '" + subUser.UserName + "');";

                bool isok = con.ExequteNoneQury(nqury);

                con.CloseCon();

                var ds = con.GetDataSet("usr", "select \* from SubUsers where BelongsToUpperUser = " + subUser.BelongsToUpperUser + ";");

                var id = int.Parse(ds.Tables[0].Rows[ds.Tables[0].Rows.Count - 1]["ID"].ToString());

                return id;

            }

            else

            {

                return -1;

            }

        }

        [WebMethod]

        ///<summary>

        ///מקבל משתמש עליון ומחזיר את המשתמשים התחתונים המשוייכים אליו

        ///takes a upper user's ditailes and returns a datatable of all his subusers

        /// </summary>

        public DataTable getYourSubUsers(User UpperUser)

        {

            var usdb = new UserDBServ();

            if (usdb.IsUserPermitted(UpperUser))

            {

                var con = new Connection(constr);

                con.openCon();

                var sequreUser = usdb.GetFullUser(UpperUser);

                var ds = con.GetDataSet("subs", "Select \* from SubUsers  where BelongsToUpperUser = " + sequreUser.ID + ";");

                return ds.Tables[0];

            }

            return null;

        }

        [WebMethod]

        ///<summary>

        ///מעדכן את פרטי משתמש קיים

        ///updates a subusers

        /// </summary>

        public bool updateSub(SubUser subUser, User user)

        {

            var usdb = new UserDBServ();

            if (usdb.IsUserPermitted(user))

            {

                var con = new Connection(constr);

                con.openCon();

                var ds = con.GetDataSet("p", "Select \* from SubUsers where ID = " + subUser.Id + ";");

                var sequreUser = usdb.GetFullUser(user);

                if (int.Parse(ds.Tables[0].Rows[0]["BelongsToUpperUser"].ToString()) == sequreUser.ID)

                {

                    if (doesUserNameAlredyInUse(subUser.Id, subUser.UserName))

                    {

                        throw new InvalidOperationException("the user "+ subUser.UserName +" name is alredy in use, use a diffrent one");

                    }

                    ds.Tables[0].Rows[0]["BelongsToUpperUser"] = subUser.BelongsToUpperUser;

                    ds.Tables[0].Rows[0]["FName"] = subUser.FName;

                    ds.Tables[0].Rows[0]["LName"] = subUser.LName;

                    ds.Tables[0].Rows[0]["Role"] = subUser.Role;

                    ds.Tables[0].Rows[0]["Email"] = subUser.Email;

                    ds.Tables[0].Rows[0]["Password"] = subUser.Password;

                    ds.Tables[0].Rows[0]["UserName"] = subUser.UserName;

                    con.Update(ds);

                    return true;

                }

            }

            return false;

        }

        private bool doesUserNameAlredyInUse(int SubId, string Username)

        {

            var con = new Connection(constr);

            var ds = con.GetDataSet("p", "Select ID from SubUsers where UserName = '" + Username + "';");

            var usdb = new UserDBServ();

            try

            {

                return int.Parse(ds.Tables[0].Rows[0]["ID"].ToString()) != SubId;

            }

            catch { return false; }

        }

        [WebMethod]

        ///<summary>

        ///takes a admin User object, and a User id. The method deletes the coresponding user from db. Returns wether the delete operation secusseded.

        ///מאפשר למשתמש מנהל לתת את פרטיו כמנהל ואת המספר המזהה של משתמש מסויים ולהסיר את משתמש זה ממסד הנתונים

        /// </summary>

        public bool DeleteSubUser(User UpUser, int id)

        {

            if (doesSubBelongToThisUser(id, UpUser))

            {

                try

                {

                    Connection con = new Connection(constr);

                    con.openCon();

                    bool a = con.ExequteNoneQury("Delete from SubUsers where ID =" + id + ";");

                    return a;

                }

                catch

                {

                    return false;

                }

            }

            else { return false; }

        }

        private bool doesSubBelongToThisUser(int id, User user)

        {

            var con = new Connection(constr);

            string q = "Select \* from SubUsers where ID = " + id + ";";

            var ds = con.GetDataSet("p", q);

            var usdb = new UserDBServ();

            return int.Parse(ds.Tables[0].Rows[0]["BelongsToUpperUser"].ToString()) == user.ID;

        }

        [WebMethod]

        ///<summary>

        ///takes  a  user with username and password, and returns the full user

        ///לוקח עצם של משתמש עם שם משתמש וססמה ומחסיר משתמש מלא

        /// </summary>

        public SubUser GetFullUser(SubUser user)

        {

            if (DoesSubExists(user.UserName, user.Password))

            {

                var con = new Connection(constr);

                var ds = con.GetDataSet("0", "Select top 1 \* from SubUsers where UserName = '" + user.UserName + "' AND Password = '" + user.Password + "' ;");

                var fullUser = new SubUser();

                fullUser.Id = int.Parse(ds.Tables[0].Rows[0]["ID"].ToString());

                fullUser.BelongsToUpperUser = int.Parse(ds.Tables[0].Rows[0]["BelongsToUpperUser"].ToString());

                fullUser.FName = ds.Tables[0].Rows[0]["FName"].ToString();

                fullUser.LName = ds.Tables[0].Rows[0]["LName"].ToString();

                fullUser.Role = ds.Tables[0].Rows[0]["Role"].ToString();

                fullUser.Email = ds.Tables[0].Rows[0]["Email"].ToString();

                fullUser.Password = ds.Tables[0].Rows[0]["Password"].ToString();

                fullUser.UserName = ds.Tables[0].Rows[0]["UserName"].ToString();

                return fullUser;

            }

            else

            {

                return null;

            }

        }

        //אומר האם המשתמש התחתון קיים במערכת

        private bool DoesSubExists(string userName, string password)

        {

            var con = new Connection(constr);

            var ds = con.GetDataSet("0", "Select top 1 \* from SubUsers where UserName = '" + userName + "' AND Password = '" + password + "' ;");

            try

            {

                return ds.Tables[0].Rows[0][0] != null;

            }

            catch { return false; }

        }

        [WebMethod]

        ///<summary>

        ///takes a subuser ditailes and returns a limited inventory table

        /// מקבל משתמש תחתון ומחזיר את רשימת המלאי של הפריטים שהוא יכול לשאול

        /// </summary>

        public DataTable getLimitedSubUserInventoryList(SubUser subUser)

        {

            if (DoesSubExists(subUser.UserName, subUser.Password))

            {

                var con = new Connection(constr);

                var ds = con.GetDataSet("items", "select ID, Name from Inventory where OwnerUserId =" + subUser.BelongsToUpperUser.ToString() + ";");

                return ds.Tables[0];

            }

            return null;

        }

        [WebMethod]

        ///<summary>

        ///takes an order and adds it to the db, extra verubles for sequrity

        /// מקבל הזמנה ומוסיף אותה לטבלה המשתנים הנוספים לאבטחה

        /// </summary>

        /// <returns>

        /// מחזיר את האיי די של ההזמנה החדשה

        /// </returns>

        public int addOrder(Order order, SubUser subUser)

        {

            if (DoesSubExists(subUser.UserName, subUser.Password))

            {

                if (doesSubBelongToThisUser(subUser.Id, new WebServ1.User() { ID = subUser.BelongsToUpperUser }))

                {

                    var con = new Connection(constr);

                    var emptyDs = con.GetDataSet("1", "Select \* from Orders where 1<0");

                    var newDr = emptyDs.Tables[0].NewRow();

                    newDr["BySubUser"] = subUser.Id;

                    newDr["ItemId"] = order.ItemId;

                    newDr["Amount"] = order.Amount;

                    newDr["ToUpperUser"] = order.ToUpperUser;

                    newDr["Aproved"] = "FALSE";

                    newDr["Rejected"] = "FALSE";

                    if (order.Remarkes != null)

                    {

                        newDr["Remarks"] = order.Remarkes.ToString();

                    }

                    newDr["OrderDate"] = DateTime.Now.ToString();

                    newDr["ItemName"] = order.ItemName;

                    con.InsertDataRow(newDr);

                    var updated = con.GetDataSet("1", "Select top 1 ID from Orders where BySubUser = " + subUser.Id + " AND ItemId = " + order.ItemId + " AND Amount = " + order.Amount + " AND ToUpperUser = " + order.ToUpperUser + " ORDER BY ID DESC;");

                    return int.Parse(updated.Tables[0].Rows[0][0].ToString());

                }

                else

                {

                    //throw new UnauthorizedAccessException();

                    return -1;

                }

            }

            else

            {

                //throw new UnauthorizedAccessException();

                return -2;

            }

        }

        [WebMethod]

        ///<summary>

        ///get all users orders

        /// מוציא טבלה עם כל ההזמנות של המשתמש

        /// </summary>

        /// <returns>

        ///returns a datatable with all the user's orders.

        /// </returns>

        public DataTable getOrders(SubUser user)

        {

            if (DoesSubExists(user.UserName, user.Password))

            {

                var sequrUser = GetFullUser(user);

                var con = new Connection(constr);

                DataSet ds = con.GetDataSet("orders", "Select \* From Orders where BySubUser = " + sequrUser.Id + " ;");

                return ds.Tables[0];

            }

            else

            {

                throw new UnauthorizedAccessException();

            }

        }

        [WebMethod]

        ///<summary>

        ///gives the upper user all the details of one of his subUsers based on his id

        /// נותן למשתמש את כל הפרטים של המשמש שאת מספר הזיהוי שלו הוא נותן

        /// </summary>

        /// <returns>

        ///returns the full SubUser.

        /// </returns>

        public SubUser GetYourSubUser(User user, int SubUserID)

        {

            var UserServ = new UserDBServ();

            if (UserServ.IsUserPermitted(user))

            {

                if (doesSubBelongToThisUser(SubUserID, user))

                {

                    var con = new Connection(constr);

                    var ds = con.GetDataSet("0", "Select top 1 \* from SubUsers where ID = " + SubUserID + ";");

                    var fullUser = new SubUser();

                    fullUser.Id = int.Parse(ds.Tables[0].Rows[0]["ID"].ToString());

                    fullUser.BelongsToUpperUser = int.Parse(ds.Tables[0].Rows[0]["BelongsToUpperUser"].ToString());

                    fullUser.FName = ds.Tables[0].Rows[0]["FName"].ToString();

                    fullUser.LName = ds.Tables[0].Rows[0]["LName"].ToString();

                    fullUser.Role = ds.Tables[0].Rows[0]["Role"].ToString();

                    fullUser.Email = ds.Tables[0].Rows[0]["Email"].ToString();

                    fullUser.Password = ds.Tables[0].Rows[0]["Password"].ToString();

                    fullUser.UserName = ds.Tables[0].Rows[0]["UserName"].ToString();

                    return fullUser;

                }

            }

            return null;

        }

        [WebMethod]

        ///<summary>

        ///takes an order and updates it

        /// מעדכן הזמנה

        /// </summary>

        /// <returns>

        ///returns whether the update procces was complited sucssesfuly

        /// </returns>

        public bool UpdateOrderByUpperUser(User user, Order order)

        {

            var UserServ = new UserDBServ();

            if (UserServ.IsUserPermitted(user))

            {

                if (doesOrderBelongToThisUpperUser(order, user))

                {

                    try

                    {

                        var sequreUser = UserServ.GetFullUser(user);

                        var con = new Connection(constr);

                        var ds = con.GetDataSet("0", "Select \* from Orders where ID = " + order.ID);

                        ds.Tables[0].Rows[0]["Amount"] = order.Amount;

                        ds.Tables[0].Rows[0]["Aproved"] = order.Aproved.ToString().ToUpper();

                        ds.Tables[0].Rows[0]["Rejected"] = order.Rejected.ToString().ToUpper();

                        ds.Tables[0].Rows[0]["Remarks"] = order.Remarkes.ToString();

                        con.Update(ds);

                        return true;

                    }

                    catch

                    {

                        return false;

                    }

                }

            }

            return false;

        }

        //בודק האם שורת הזמנה שייכת למשתמש עליון מסויים

        private bool doesOrderBelongToThisUpperUser(Order order, User user)

        {

            var con = new Connection(constr);

            var UserServ = new UserDBServ();

            var sequreUser = UserServ.GetFullUser(user);

            DataSet ds = con.GetDataSet("0", "Select \* From Orders where ID = " + order.ID + ";");

            try {

                bool result = int.Parse(ds.Tables[0].Rows[0]["ToUpperUser"].ToString()) == sequreUser.ID;

                return result;

            }

            catch

            {

                return false;

            }

        }

    }

}

### הקוד לUserDBServ.asmx

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Data;

using System.Data.SqlClient;

using System.Net;

//made by yehonatan vishna

namespace WebServ1

{

    /// <summary>

    /// some functions to interact with users and their details in db

    /// </summary>

    [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

    [System.ComponentModel.ToolboxItem(false)]

    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

    // [System.Web.Script.Services.ScriptService]

    public class UserDBServ : System.Web.Services.WebService

    {

        //יוצר את המחרוזת חיבור שסביר שבשימוש לפי ברירת המחדל של אס קיו אל סרוור

        //creates the tipical constring according to sqlserver express defaults

        public String constr = "Server = '" + Dns.GetHostName() + "\\SQLEXPRESS'; Database = StorageSystem; Trusted\_Connection = True; ";

        [WebMethod]

        ///<summary>

        ///a function that takes user object and adds it to users table in the db.

        ///לוקח עצם משתמש ומוסיף אותו כרשומה בטבלת המשתמשים

        /// </summary>

        public Boolean reg(User usr)

        {

                Connection con = new Connection(constr);

                con.openCon();

                string b = "Insert into [StorageSystem].[dbo].[Users] (FName, LName, BDate, compeny, email, password) Values (N'" + usr.Fname + "', N'" + usr.Lname + "' , " + "CAST(N'" + (usr.BDate.GetValueOrDefault()).ToShortDateString() + "' AS DateTime)" + " , N'" + usr.Compeny + "', N'" + usr.Email + "' , N'" + usr.Password + "');";

                bool a = con.ExequteNoneQury(b);

                con.CloseCon();

                return a;

        }

        [WebMethod]

        ///<summary>

        ///a function that returns whether a user exists in users table in db

        ///פונקציה שאומרת האם משתמש בעל אותו דואל וססמה כמו זה שהתקבל קיים במערכת

        /// </summary>

        public bool IsUserPermitted(User usr)

        {

            try

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("logged", "select \* from users where email = N'" + usr.Email + "' AND password= N'" + usr.Password + "';");

                return ds.Tables[0].Rows[0]["ID"] != null;

            }

            catch {

                return false;

            }

        }

        [WebMethod]

        ///<summary>

        ///adds a new user to users table in db and returns his id.

        ///מוסיף משתמש ריק לטבלת המשמשים ומחזיר את מספר הזיהוי שלו

        /// </summary>

        public int AddEmptyUser()

        {

            Connection con = new Connection(constr);

            con.openCon();

            string b = "Insert into [StorageSystem].[dbo].[Users] (FName, LName, BDate, compeny, email, password) Values (N'new user', N'' ,'' , N'', N'' , N'');";

            bool a = con.ExequteNoneQury(b);

            DataSet ds = con.GetDataSet("newUser", "Select \* from Users where FName = 'new user'");

            int newId = int.Parse(ds.Tables[0].Rows[ds.Tables[0].Rows.Count-1][0].ToString());

            con.CloseCon();

            return newId;

        }

        //made by yehonatan vishna

        [WebMethod]

        ///<summary>

        ///takes a basic user with only email and passwor. returns a full user with all the user details as they are in the db.

        ///לוקח משתמש חלקי עם דואל וסיסמה בלבד ומחזיר משתמש מלא עם כל התכונות של המשתמש כפי שמופיעות במסד הנתונים

        /// </summary>

        public User GetFullUser(User usr)

        {

            User user = new User();

            if (IsUserPermitted(usr))

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("logged", "select \* from users where email = '" + usr.Email + "' AND password= '" + usr.Password + "';");

                user.ID = int.Parse(ds.Tables["logged"].Rows[0]["ID"].ToString());

                user.Fname = ds.Tables["logged"].Rows[0]["FName"].ToString();

                user.Lname = ds.Tables["logged"].Rows[0]["LName"].ToString();

                user.BDate = DateTime.Parse(ds.Tables["logged"].Rows[0]["BDate"].ToString());

                user.Compeny = ds.Tables["logged"].Rows[0]["compeny"].ToString();

                user.Email = ds.Tables["logged"].Rows[0]["email"].ToString();

                user.Password = ds.Tables["logged"].Rows[0]["Password"].ToString();

            }

            return user;

        }

        [WebMethod]

        ///<summary>

        ///takes the admin user details (as an object), an object contaning user's motified detailes, and a user id. returns whether the change of user details in db secseded.

        ///פעולה המאפשרת למנהל לתת מספר מזהה של משתמש ואת פרטיו החדשים בסוף התהליך פריטיו של המשתמש יתעדכנו במסד הנתונים

        /// </summary>

        public bool updateUserByIdAdmin( int id, User Admin, User NewUsr)

        {

            if (IsAdmin(Admin))

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("logged", "select \* from users where ID=" + id.ToString() + ";");

                ds.Tables["logged"].Rows[0]["FName"] = NewUsr.Fname;

                ds.Tables["logged"].Rows[0]["LName"] = NewUsr.Lname;

                ds.Tables["logged"].Rows[0]["BDate"] = NewUsr.BDate;

                ds.Tables["logged"].Rows[0]["compeny"] = NewUsr.Compeny;

                ds.Tables["logged"].Rows[0]["email"] = NewUsr.Email;

                ds.Tables["logged"].Rows[0]["password"] = NewUsr.Password;

                con.Update(ds);

                return true;

            }

            return false;

        }

        [WebMethod]

        ///<summary>

        ///takes the user details (as an object), and an object contaning user's motified detailes. returns whether the change of user details in db secseded.

        ///מקבל את פרטיו הנוכחיים של משתמש ומחליף אותם בפרטים חדשים לפי רצונו של המשתמש

        /// </summary>

        public bool updateUser(User OldUsr, User NewUsr)

        {

            if (IsUserPermitted(OldUsr))

            {

                Connection con = new Connection(constr);

                DataSet ds = con.GetDataSet("logged", "select \* from users where email = '" + OldUsr.Email + "' AND password= '" + OldUsr.Password + "';");

                ds.Tables["logged"].Rows[0]["FName"] = NewUsr.Fname;

                ds.Tables["logged"].Rows[0]["LName"] = NewUsr.Lname;

                ds.Tables["logged"].Rows[0]["BDate"] = NewUsr.BDate;

                ds.Tables["logged"].Rows[0]["compeny"] = NewUsr.Compeny;

                ds.Tables["logged"].Rows[0]["email"] = NewUsr.Email;

                ds.Tables["logged"].Rows[0]["password"] = NewUsr.Password;

                con.Update(ds);

                return true;

            }

            return false;

        }

        //made by yehonatan vishna

        [WebMethod]

        ///<summary>

        ///takes a user object, and deletes the coresponding user from db. Returns wether the delete operation secusseded.

        ///מקבל עצם של משתמש עם דואל וסיסמה ומוחק את משתמש או משתמשים אלו ממסד הנתונים

        /// </summary>

        public bool DeleteUser(User usr) {

            Connection con = new Connection(constr);

            con.openCon();

            return con.ExequteNoneQury("Delete From users where email='" + usr.Email + "' AND password = '" + usr.Password + "'");

        }

        [WebMethod]

        ///<summary>

        ///takes a admin User object, and a User id. The method deletes the coresponding user from db. Returns wether the delete operation secusseded.

        ///מאפשר למשתמש מנהל לתת את פרטיו כמנהל ואת המספר המזהה של משתמש מסויים ולהסיר את משתמש זה ממסד הנתונים

        /// </summary>

        public bool DeleteUserAdmin(User Admin ,int id)

        {

            User user = Admin;

            if (IsAdmin(user))

            {

                try

                {

                    Connection con = new Connection(constr);

                    con.openCon();

                    bool a = con.ExequteNoneQury("Delete from Users where ID =" + id + ";");

                    return a;

                }

                catch

                {

                    return false;

                }

            }

            else { return false; }

        }

        [WebMethod]

        ///<summary>

        ///checks wether a user is a admin.

        ///מקבל עצם של משתמש עם דואל וסיסמה ובודק האל משתמש זה הוא מנהל

        /// </summary>

        public bool IsAdmin(User user)

        {

            if (user.Email.Equals("admin@administrator.adm") && user.Password.Equals("Admin"))

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        [WebMethod]

        ///<summary>

        ///takes a User object. Asuming the user is admin, the method would return a DataTable with all the User's data

        ///מאפשר למהנהל לאמת את עצמו כמנהל ולקבל טבלה עם פרטי כל המשתמשים הקיימים במסד הנתונים

        /// </summary>

        public DataTable GetAdminUserTbl(User user)

        {

            if (IsAdmin(user))

            {

                Connection con = new Connection(constr);

                con.openCon();

                DataSet ds = con.GetDataSet("Users", "Select \* From Users");

                DataTable dt = ds.Tables[0];

                return dt;

            }

            else { return null; }

        }

        /// <summary>

        /// checks whether an email is already regestered with a user.

        ///בודק האם דואל מסויים כבר קיים בטבלת המשתמשים

        /// </summary>

        [WebMethod]

        public bool DoesEmailExist(string email)

        {

            var con = new Connection(constr);

            var ds = con.GetDataSet("users", "Select \* from Users where email = '" + email + "';");

            try

            {

                return ds.Tables["users"].Rows[0]["ID"] != null;

            }

            catch

            {

                return false;

            }

        }

        [WebMethod]

        ///<summary>

        ///get all the orders sent to the Upper User

        /// מוציא טבלה עם כל ההזמנות של המשתמשים הנמוכים השייכים למשתמש הגבוה

        /// </summary>

        /// <returns>

        ///returns a datatable with all the orders sent to the Upper User.

        /// </returns>

        public DataTable getUpperOrders(User user)

        {

            var UpServ = new UserDBServ();

            if (UpServ.IsUserPermitted(user))

            {

                var sequrUser = UpServ.GetFullUser(user);

                var con = new Connection(constr);

                DataSet ds = con.GetDataSet("orders", "Select \* From Orders where ToUpperUser = " + sequrUser.ID + " ;");

                return ds.Tables[0];

            }

            else

            {

                throw new UnauthorizedAccessException();

            }

        }

    }

}

## שירותי רשת חיצוניים

עד כה, השתמשתי בשירות רשת חיצוני אחד. שירות הרשת החיצוני בו השתמשתי נקרא: "costom search engine" של גוגל. שירות הזה הוא מבוסס בקשות API (בשונה משירותי הרשת שהשתמשתי בהם בפרויקט שהם מבוססי פרוטוקול SOAP). במסגרת שירות רשת זה, נשלח לשירות הרשת ביטוי מסויים, והוא מחזיר רשימת עמודי אינטרנט מתוך אתרים שהגדרתי לו מראש (ksp, קרביץ, באג, אופיס דיפו, וכו'...). בפרויקט, אני משתמש בשירות הרשת כדי לעזור למשתמש למצוא בקלות איפה הוא יכול לקנות עוד ממוצר שיש לו במלאי.

הפונקציה בצד הלקוח שמשתמשת בשירות הרשת:

//מבצע חיפוש בגוגל במנוע חיפוש מותאם אישית עם מגוון אתרי קמעונעות

private async Task<Google.Apis.CustomSearchAPI.v1.Data.Search> googleApi(string qury)

{

    string apiKey = "AIzaSyAs4wufxBxLid6quJCN2ynqihPieMxP3qY";

    string cx = "f427caefac6b24f26";

    var svc = new Google.Apis.CustomSearchAPI.v1.CustomSearchAPIService(new BaseClientService.Initializer { ApiKey = apiKey });

    var listRequest = svc.Cse.List();

    listRequest.Q = qury;

    listRequest.Gl = "il";

    listRequest.Cx = cx;

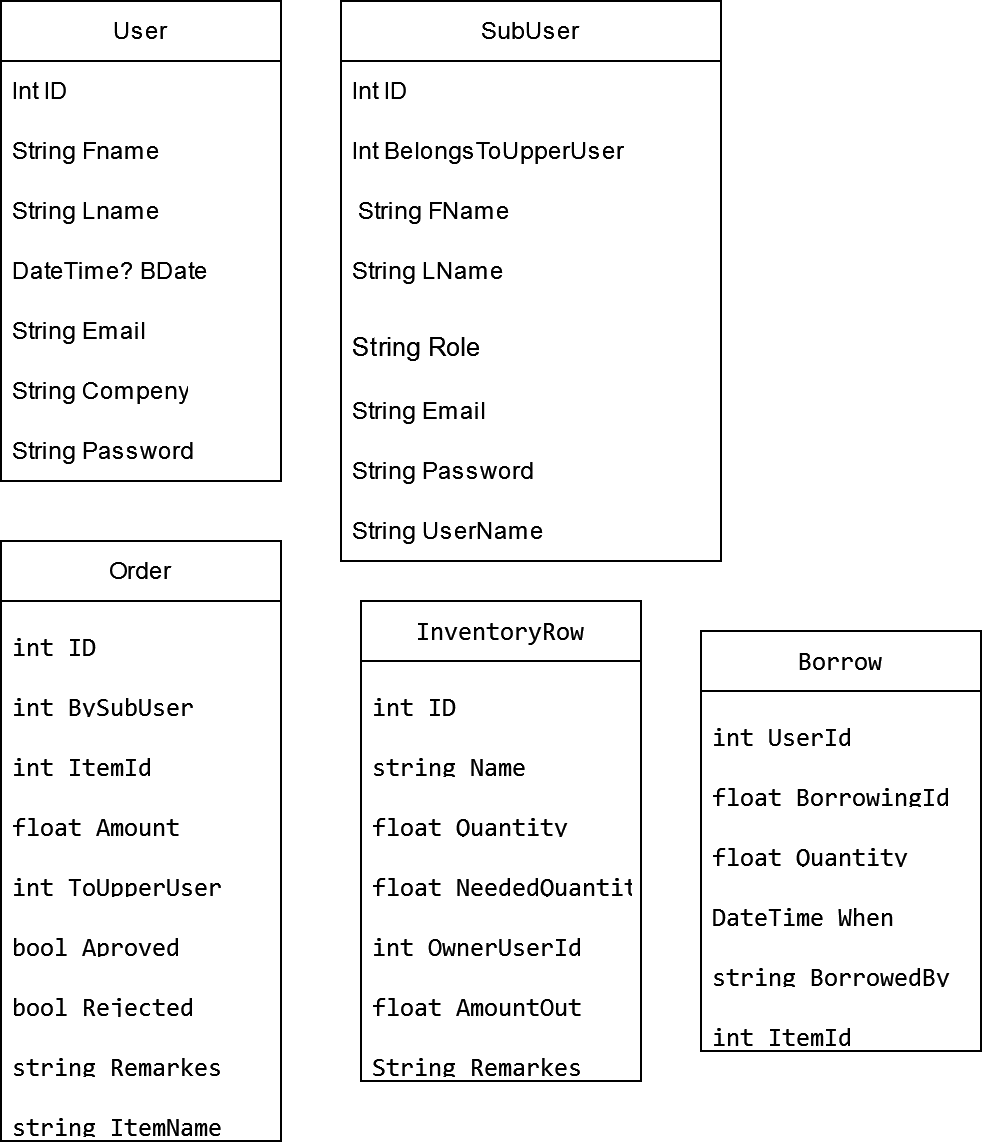
    var search = await listRequest.ExecuteAsync();

    return search;

}

## היגיון עסקי (Business Logic)

לצורך פישוט ניהול הנתונים, נתוני המשתמשים, הפריטים וההזמנות נתונים בטיפוסים שכתבתי. פרמטרים לפונקציות של שירות הרשת מועברים כמשתנים מטיםוסים אלו, אך בשל ההגבלות של פורמט asmx, המידע משירותי רשת אלו לא חוזר כרשימות (לדוגמה List<User>) אלא כטבלאות מידע (datatable).



## קוד של מחלקות נבחרות:

### הקוד של מחלקת User

using System;

namespace WebServ1

{

    public class User

    {

        //תכונות בסיסיות של המשתמש

        //defining basic properties

        private int id;

        private string fname;

        private string lname;

        private DateTime? bdate;

        private string compeny;

        private string email;

        private string password;

        //בנאי ריק

        // empty constractor

        public User()

        {

        }

        //גטרים וסטרים לכל התכונות

        //getters + setters for properties

        public int ID

        {

            get

            {

                return this.id;

            }

            set

            {

                this.id = value;

            }

        }

        public string Fname

        {

            get { return this.fname; }

            set { this.fname = value; }

        }

        public string Lname

        {

            get { return this.lname; }

            set { this.lname = value; }

        }

        public DateTime? BDate

        {

            get { return this.bdate; }

            set { this.bdate = value; }

        }

        public string Compeny

        {

            get { return this.compeny; }

            set { this.compeny = value; }

        }

        public string Email

        {

            get { return this.email; }

            set { this.email = value; }

        }

        public string Password

        {

            get { return this.password; }

            set { this.password = value; }

        }

        //פעולה שיוצרת עצם חדש עם אותם תכונות כמו עצם זה

        // a fution that creatates another User obj identical to this one

        public User copy()

        {

            User a = new User();

            a.ID = this.id;

            a.Fname = this.Fname;

            a.Lname = this.Lname;

            a.BDate = this.BDate;

            a.compeny = this.compeny;

            a.email = this.email;

            a.password = this.password;

            return a;

        }

        //תכונה הבודקת האם משתמש אחר זהה בכל פרטיו למשתמש זה

        // a function that returns wether another User object is identical to this one

        public bool IsSame(User user)

        {

            try

            {

                bool idS = this.ID == user.ID;

                bool FnameS = this.Fname.Equals(user.Fname);

                bool LnameS = this.Lname.Equals(user.Lname);

                bool DateS = this.BDate.Equals(user.BDate);

                bool CompS = this.compeny.Equals(user.Compeny);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                return idS && FnameS && LnameS && DateS && CompS && EmailS && PassS;

            }

            catch

            {

                return false;

            }

        }

        //made by yehonatan vishna

    }

}

### הקוד למחלקת Order

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebServ1

{

    public class Order

    {

        //תכונות של ההזמנה

        private int iD;

        private int bySubUser;

        private int itemId;

        private float amount;

        private int toUpperUser;

        private bool aproved;

        private bool rejected;

        private string remarkes= "";

        private string itemName;

        private DateTime orderDate;

        private bool isActive;

        public Order()

        {

        }

        //גטרים וסטטרים של התכונות

        public int ID { get => iD; set => iD = value; }

        public int BySubUser { get => bySubUser; set => bySubUser = value; }

        public int ItemId { get => itemId; set => itemId = value; }

        public float Amount { get => amount; set => amount = value; }

        public int ToUpperUser { get => toUpperUser; set => toUpperUser = value; }

        public bool Aproved { get => aproved; set => aproved = value; }

        public bool Rejected { get => rejected; set => rejected = value; }

        public string Remarkes { get => remarkes; set => remarkes = value; }

        public string ItemName { get => itemName; set => itemName = value; }

        public DateTime OrderDate { get => orderDate;

            set

            {

                orderDate = value;

            }

            }

        private string status;

        //תכונה שמטרתה לתת סיכום בטקסט למשתמש של מצב הזמנתו

        public string Status

        {

            get { return status; }

            set { status = value; }

        }

        //האם ההזמנה פעילה

        public bool IsActive { get => isActive; set => isActive = value; }

    }

}

### הקוד למחלקת Borrow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WpfApp1;

namespace InventoryServ

{

    public class Borrow

    {

        private int itemId;

        private string borrowedBy;

        private DateTime when;

        private float quantity;

        private float borrowingId;

        private int userId;

        public Borrow() { }

        // because of the desine and the relation of the tables, a qury to the inventory table must be made in order to get the item's name.

        public string ItemName {

            get

            {

                var con = new Connection(BorowwingsDB.constr);

                con.openCon();

                var ds = con.GetDataSet("item name", "select Name from Inventory where ID = " + this.itemId + ";");

                return ds.Tables["item name"].Rows[0]["Name"].ToString();

            }

        }

        public int ItemId

        {

            get { return itemId; }

            set { itemId = value; }

        }

        public string BorrowedBy { get => borrowedBy; set => borrowedBy = value; }

        public DateTime When { get => when; set => when = value; }

        public float Quantity { get => quantity; set => quantity = value; }

        public float BorrowingId { get => borrowingId; set => borrowingId = value; }

        public int UserId { get => userId; set => userId = value; }

    }

}

### הקוד למחלקת InventoryRow

using System;

using System.Collections.Generic;

using System.Linq;

namespace InventoryServ

{

    public class InventoryRow

    {

        //defines some properies

        private int id;

        private string name;

        private float quantity;

        private float neededQuantity;

        private int ownerUserId;

        private float amountOut;

        private string remarks;

        public InventoryRow() { }

        public InventoryRow(int id, string name, int q, int nq, int oui)

        {

            this.id = id;

            this.name = name;

            this.quantity = q;

            this.neededQuantity = nq;

            this.ownerUserId = oui;

        }

        //getters and setters for all the properties listed above

        public int ID

        {

            get { return this.id; }

            set { this.id = value; }

        }

        public string Name

        {

            get { return this.name; }

            set { this.name = value; }

        }

        public float Quantity

        {

            get { return this.quantity; }

            set { this.quantity = value; }

        }

        public float NeededQuantity

        {

            get { return this.neededQuantity; }

            set { this.neededQuantity = value; }

        }

        public int OwnerUserId

        {

            get { return this.ownerUserId; }

            set { this.ownerUserId = value; }

        }

        public float AmountOut

        {

            get { return this.amountOut; }

            set { this.amountOut = value; }

        }

        public String Remarkes

        {

            get { return this.remarks; }

            set { this.remarks = value; }

        }

        // a fution that creatates another InventoryRow obj identical to this one

        public InventoryRow copy()

        {

            InventoryRow a = new InventoryRow();

            a.ID = this.id;

            a.Quantity = this.quantity;

            a.OwnerUserId = this.ownerUserId;

            a.NeededQuantity = this.neededQuantity;

            a.Name = this.name;

            a.Remarkes = this.remarks;

            return a;

        }

        // a function that returns wether another InventoryRow object is identical to this one

        public bool Equal(InventoryRow row)

        {

            bool IdSame = this.id == row.ID;

            bool QSame = this.quantity == row.Quantity;

            bool OUIdSame = this.ownerUserId == row.OwnerUserId;

            bool NQSame = this.neededQuantity == row.NeededQuantity;

            bool NameSame;

            if ((row.Name == null && this.name != null) || (row.Name != null && this.name == null))

            {

                NameSame = false;

            }

            else

            {

                if ((row.Name == null && this.name == null))

                {

                    NameSame = true;

                }

                else

                {

                    NameSame = this.name.Equals(row.Name.ToString());

                }

            }

            bool RSame;

            if ((row.Remarkes == null && this.remarks != null) || (row.Remarkes != null && this.remarks == null))

            {

                RSame = false;

            }

            else

            {

                if ((row.Remarkes == null && this.remarks == null))

                {

                    RSame = true;

                }

                else

                {

                    RSame = this.remarks.Equals(row.Remarkes.ToString());

                }

            }

            return IdSame && QSame && OUIdSame && NQSame && NameSame && RSame;

        }

    }

}

### הקוד למחלקת SubUser

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebServ1

{

    public class SubUser

    {

        //מגדיר תכונות בסיסיות

        //defining reqired properties

        private int id;

        private int belongsToUpperUser;

        private string fName;

        private string lName;

        private string role;

        private string email;

        private string password;

        private string userName;

        public SubUser()

        {

        }

        public int Id { get => id; set => id = value; }

        public int BelongsToUpperUser { get => belongsToUpperUser; set => belongsToUpperUser = value; }

        public string FName { get => fName; set => fName = value; }

        public string LName { get => lName; set => lName = value; }

        public string Role { get => role; set => role = value; }

        public string Email { get => email; set => email = value; }

        public string Password { get => password; set => password = value; }

        public string UserName { get => userName; set => userName = value; }

        public SubUser Copy()

        {

            return new SubUser() { Id = this.id, BelongsToUpperUser = this.BelongsToUpperUser, FName = this.FName, LName = this.LName, Role = this.Role, UserName = this.userName };

        }

        public bool IsSame(SubUser user)

        {

            try

            {

                bool idS = this.Id == user.Id;

                bool FnameS = this.FName.Equals(user.FName);

                bool LnameS = this.LName.Equals(user.LName);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                bool belS = this.BelongsToUpperUser.Equals(user.BelongsToUpperUser);

                bool roleS = this.Role.Equals(user.Role);

                bool Usr = this.UserName.Equals(user.UserName);

                return idS && FnameS && LnameS && EmailS && PassS && belS && roleS && Usr;

            }

            catch

            {

                return false;

            }

        }

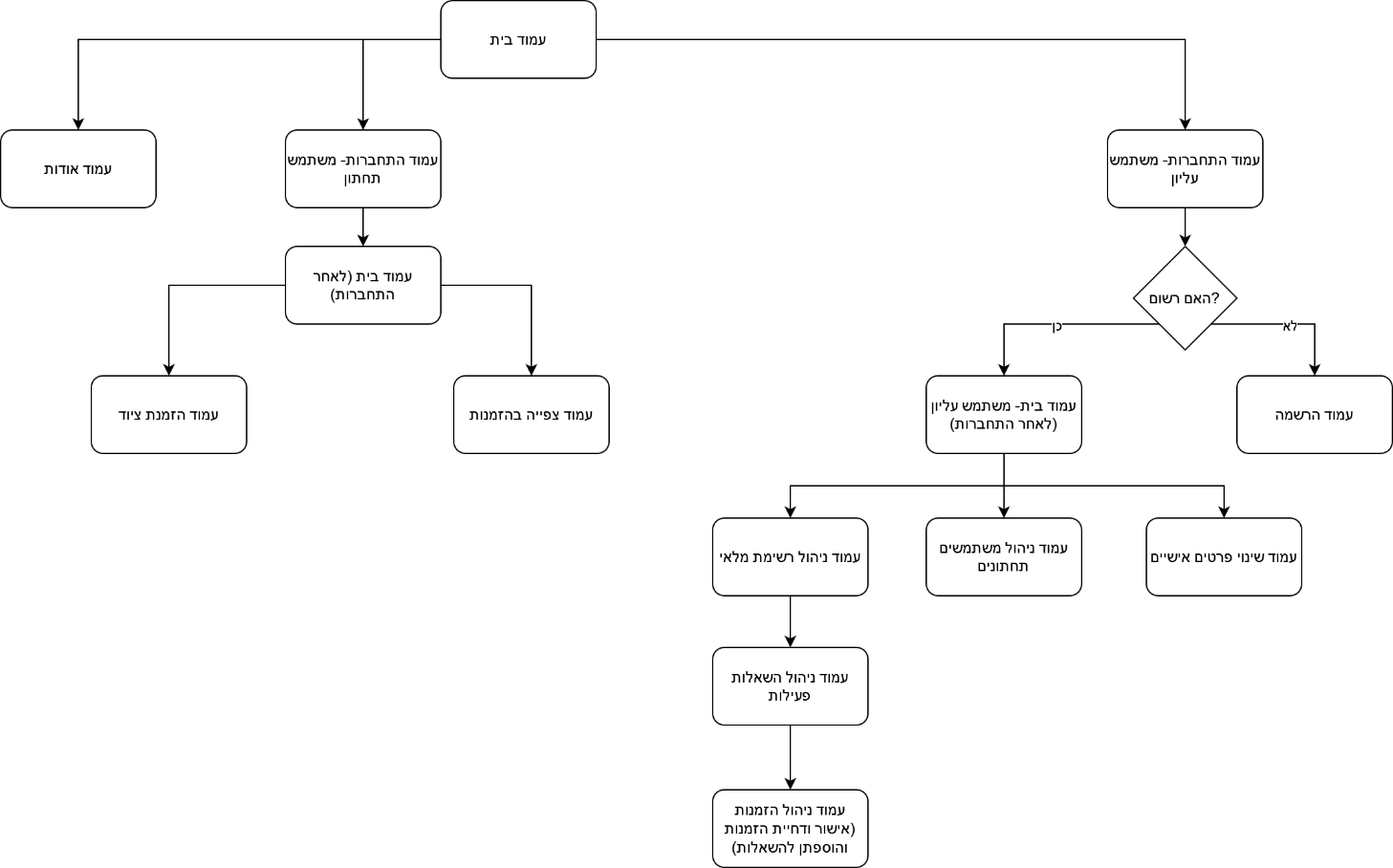
    }

}

# צד משתמש

## ממשקי משתמש

ממשקי המשתמש מתוארים על ידי תרשים הזרימה הבא:



### הרחבות:

כדי לשמור על ממשק נוח למשתמש העליון ולמנהל המערכת, בניתי פקד בעצמי, פקד זה הוא שילוב של checkbox וpasswordBox לכדי פקד אחד, כך שכל תיבת סימון שולטת על האם הססמה של תיבת הססמה שלידה ניתנת לקריאה.

הקוד של ההרחבה:

קוד הxaml:

<UserControl

    x:Class="storageUniversal.PasswordWithCheck"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    d:DesignHeight="300"

    d:DesignWidth="400">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="\*"/>

            <ColumnDefinition Width="0.2\*"/>

        </Grid.ColumnDefinitions>

        <PasswordBox x:Name="pass" Grid.Column="0" PlaceholderText="{x:Bind placeHolder}" Password="{x:Bind Text, Mode=TwoWay}" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>

        <CheckBox Grid.Column="1" Checked="CheckBox\_Checked" Unchecked="CheckBox\_Unchecked" ToolTipService.ToolTip="see your password"></CheckBox>

    </Grid>

</UserControl>

קוד הC# השולט בפקד:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The User Control item template is documented at https://go.microsoft.com/fwlink/?LinkId=234236

namespace storageUniversal

{

    public sealed partial class PasswordWithCheck : UserControl

    {

        public PasswordWithCheck()

        {

            this.InitializeComponent();

        }

        public string Text

        {

            get => (string)GetValue(TextProperty);

            set => SetValue(TextProperty, value);

        }

        public string placeHolder

        {

            get => (string)GetValue(PlaceHolderProp);

            set => SetValue(PlaceHolderProp, value);

        }

        public static readonly DependencyProperty TextProperty =

          DependencyProperty.Register(nameof(Text), typeof(string),

            typeof(PasswordWithCheck), new PropertyMetadata(string.Empty));

        public static readonly DependencyProperty PlaceHolderProp =

  DependencyProperty.Register(nameof(placeHolder), typeof(string),

    typeof(PasswordWithCheck), new PropertyMetadata(string.Empty));

        private void Button\_Click(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void Pass\_GotFocus(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void Pass\_LostFocus(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Hidden;

        }

        private void CheckBox\_Checked(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void CheckBox\_Unchecked(object sender, RoutedEventArgs e)

        {

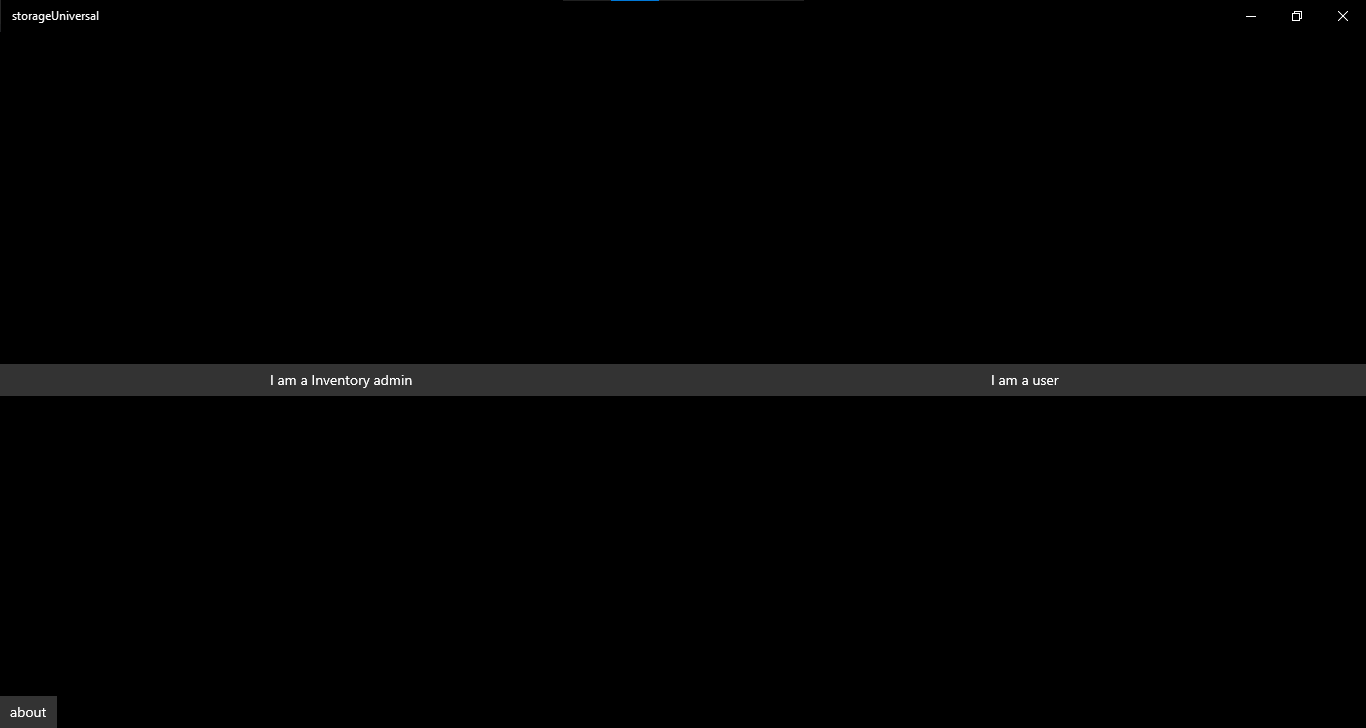
            pass.PasswordRevealMode = PasswordRevealMode.Hidden;

        }

    }

}

## מדריך למשתמש



כאשר משתמש כלשהו נכלנס לתוכנה, יש לו שני אפשרויות, ללכת לעמוד ההתחברות של משתמש עליון- מנהלי המלאי. או ללכת לעמוד ההתחברות של משתמשים תחתונים- האנשים שרק מזמינים ציוד.

בנוסף, יש אפשרות להיכנס לעמוד האודות על התוכנה.

### מדריך למשתמש העליון

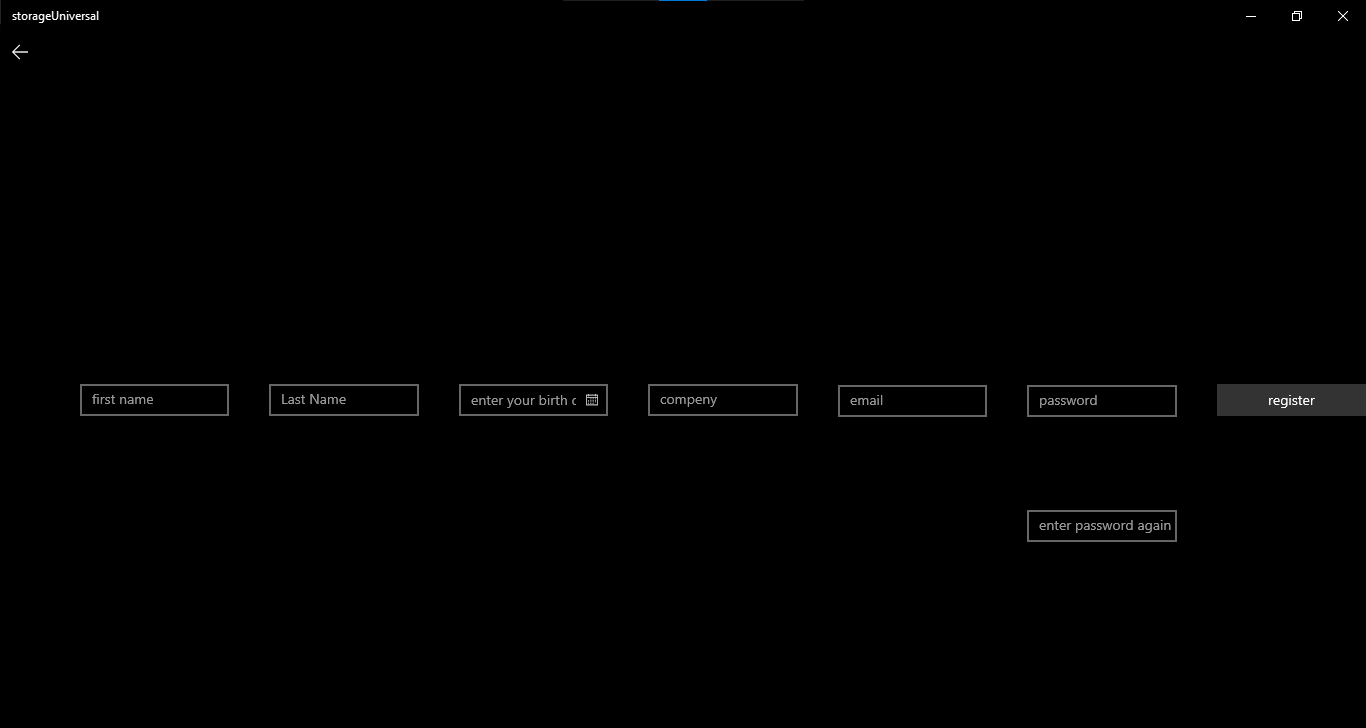
#### עמוד ההתחברות:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

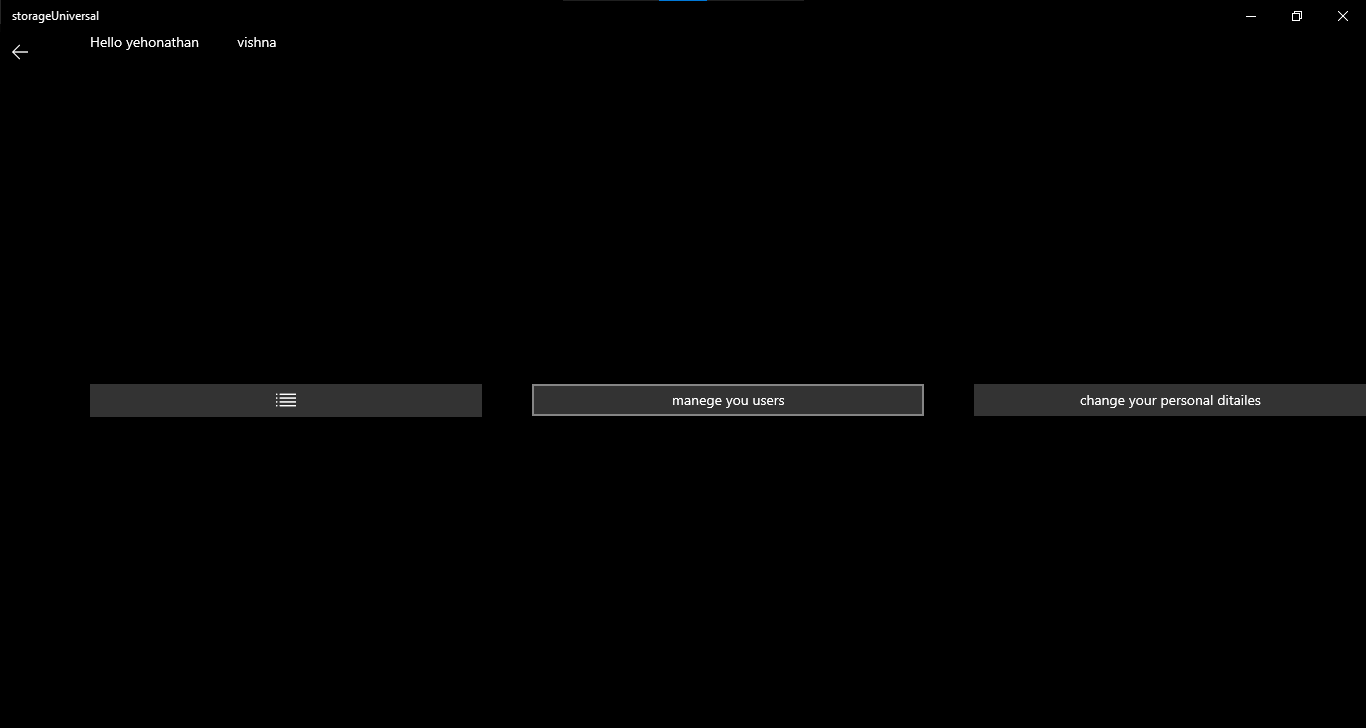
כדי להיכנס למשתמש שלך, עליך להכניס את הדוא"ל והססמה שאיתם נרשמת. ישנה אופציה בצד שמאל שתאפשר לך לשמור את פרטי ההתחברות שלך במערכת כדי שלא תצתרך לרשום אותם שוב בפעם הבאה. במידה שאתה רוצה שהמחשב ישכח את כל פרטי ההתחברות הקודמים, עליך לסמן את תיבת הסימון למטה בצד שמאל וללחוץ על כפתור ההתנתקות. במידה ואין לך עדיין משתמש, אתה יכול ללחוץ על כפתור ההרשמה למטה בצד שמאל.

#### עמוד הרשמה:



בעמוד זה, עליך להכניס את פרטיך כפי שמתואר בשדות. עליך להזין דוא"ל תקין, ולחזור על אותה הססמה פעמיים (בשני שדות הסיסמה). לבסוף, לחץ על כפתור ההרשמה.

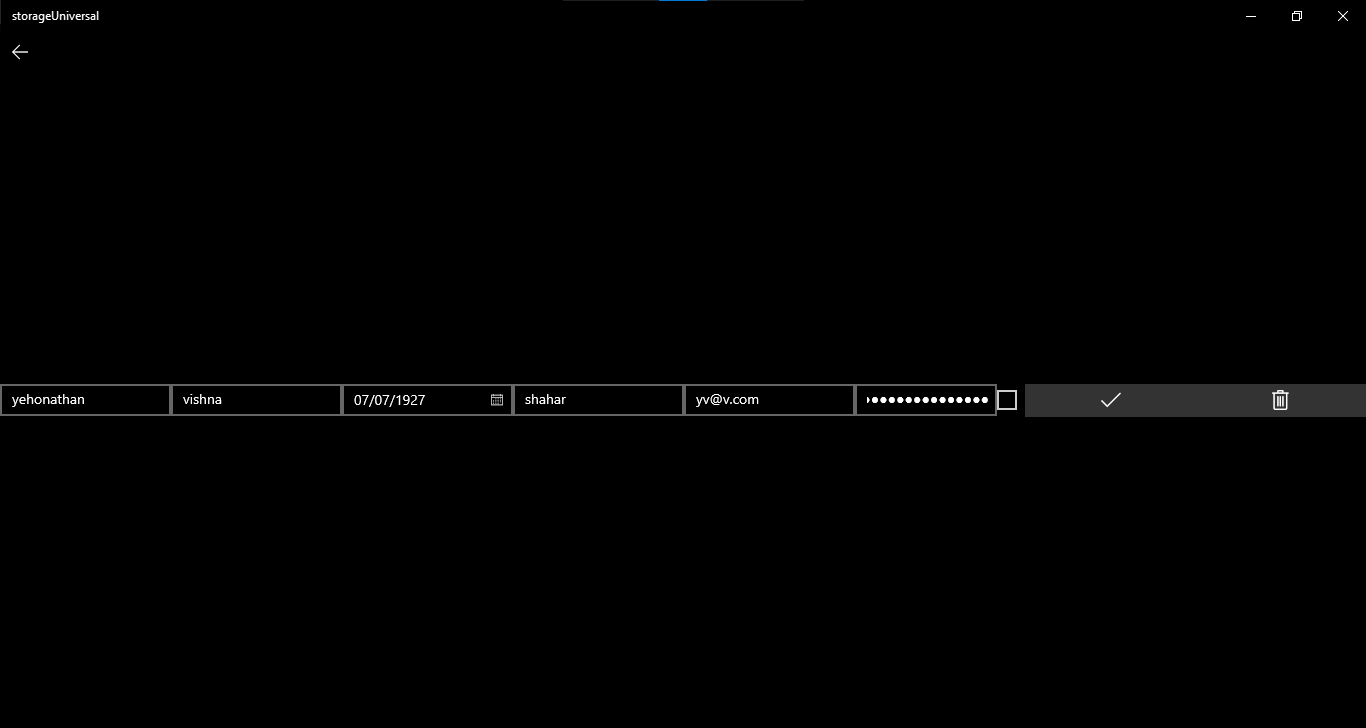
#### עמוד בית:



לאחר שהתחברת, תגיע למסך הבית שלך. ממסך זה, אתה יכול להגיע לשלוש מסכים בעלי תפקידים שונים:

1. מסך עם רשימת המלאי שלך ואפשרויות הניהול של רשימה זו.
2. עמוד שייתן לך לנהל את המשתמשים התחתונים שלך.
3. עמוד שייתן לך לשנות את הפרטים האישיים שלך.

#### עמוד שינוי פרטים אישיים



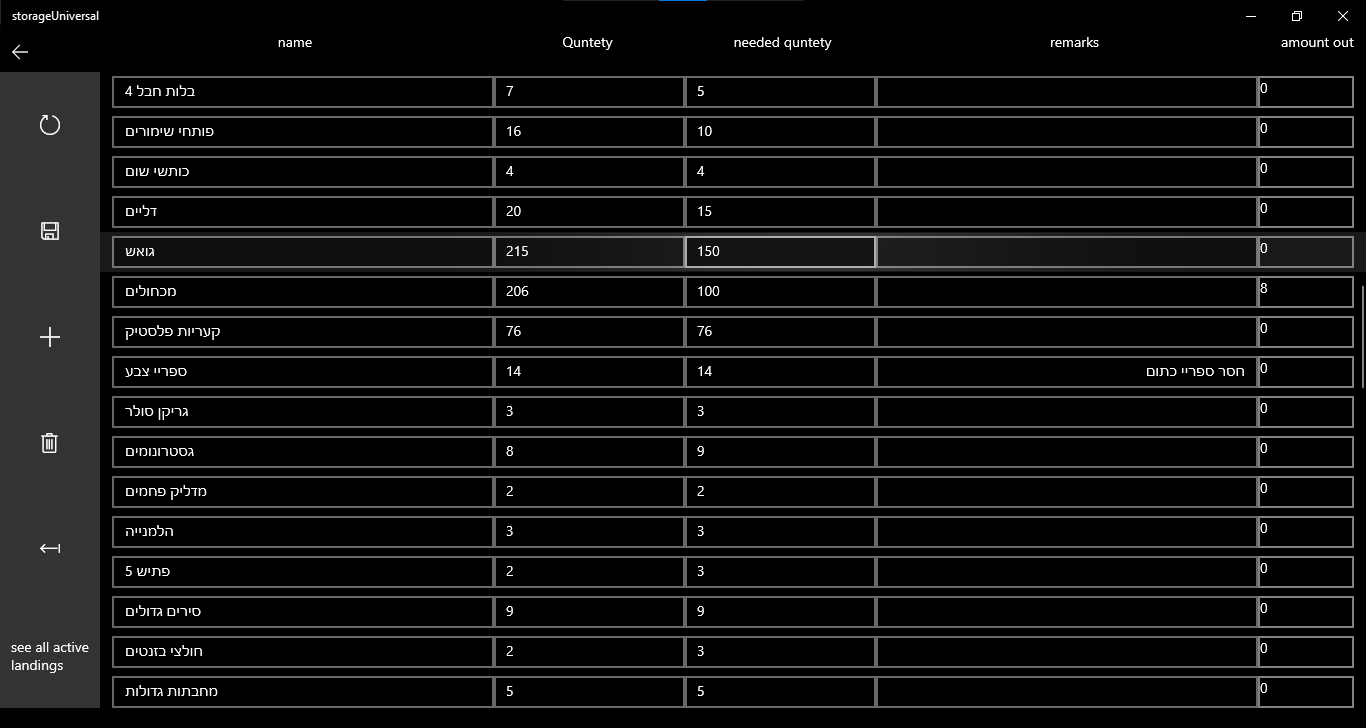
בעמוד זה, מופיעים לפניך כל הפרטים האישיים שלך כולל שם, שם משפחה, תאריך לידה, החברה בה אתה עובד, דוא"ל וסיסמה. אתה יכול אשר את השינויים שעשית בפרטיך האישיים או למחוק לחלוטין את המשתמש שלך מהמערכת.

#### עמוד ניהול משתמשים תחתונים



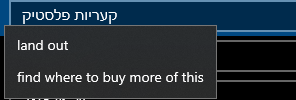
בעמוד זה, אתה רואה טבלה עם כל המשתמשים התחתונים שמשוייכים אליך, אתה יכול להוסיף משתמשים, למחוק משתמשים, ולשנות את פרטיהם האישיים במערכת.

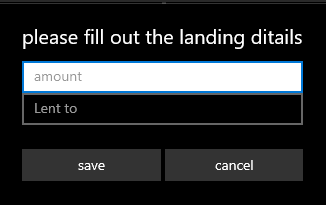
#### עמוד ניהול רשימת מלאי



בעמוד זה, אתה יכול לבחון טבלה המכילה את הפריטים ברשימת המלאי שלך. אתה יכול לשנות את רשימת המלאי שלך ולהוסיף או למחוק ממנה פריטים. אתה יכול לייבא רשימת מלאי שכבר יש לך בקובץ csv ולייבא אותה לתוך הרשימה.

על ידי לחיצה ימנית על פריט לאחר סימונו, אתה יכול להוסיף השאלה של פריט זה.





אתה יכול גם ללחוץ בתפריט הקליק הימני על האופציה של "מצא היכן לקנות עוד מזה". לחיצה על אופציה זו תשלח אותך לעמוד מאתר של קמעונעית המוכרת את פריט זה או דומה.

מעמוד זה אתה יכול לעבור לעמוד אחר, בו אתה יכול לראות את כל ההשאלות הפעילות שהושאלו מרשימת המלאי שלך.

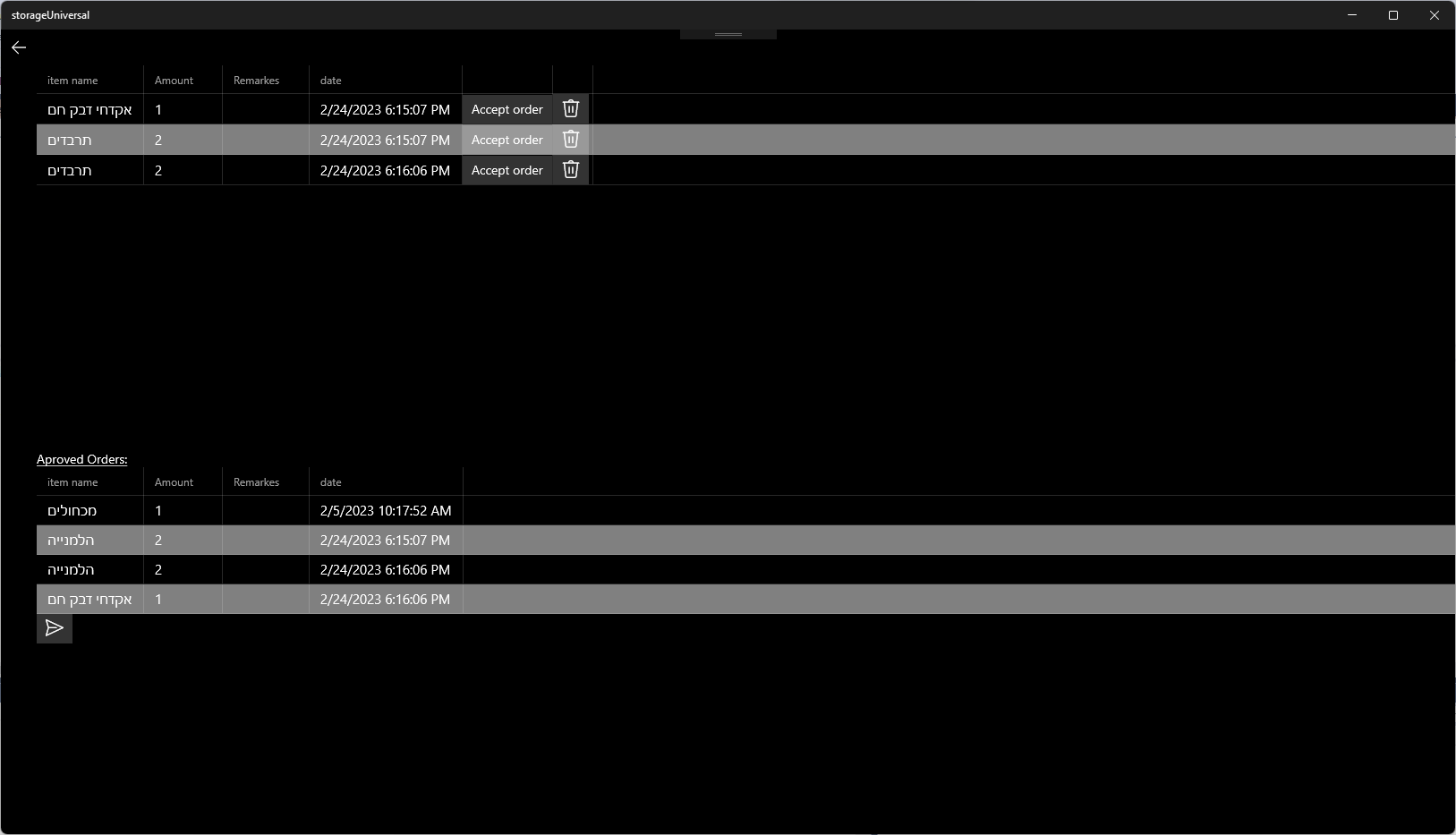
#### עמוד ניהול השאלות



בעמוד זה אתה יכול לראות ולשנות פרטים על ההשאלות הפעילות (אתה יכול להכניס לעמודה של amount lent רק מספר (שלם או עשרוני). אתה יכול למחוק השאלות על ידי לחיצה על אפשרות "delete" בתפריט הקליק הימני.

על ידי לחיצה על האייקון למטה בצד שמאל אתה יכול לעבור לעמוד ניהול ההזמנות.

#### עמוד ניהול הזמנות

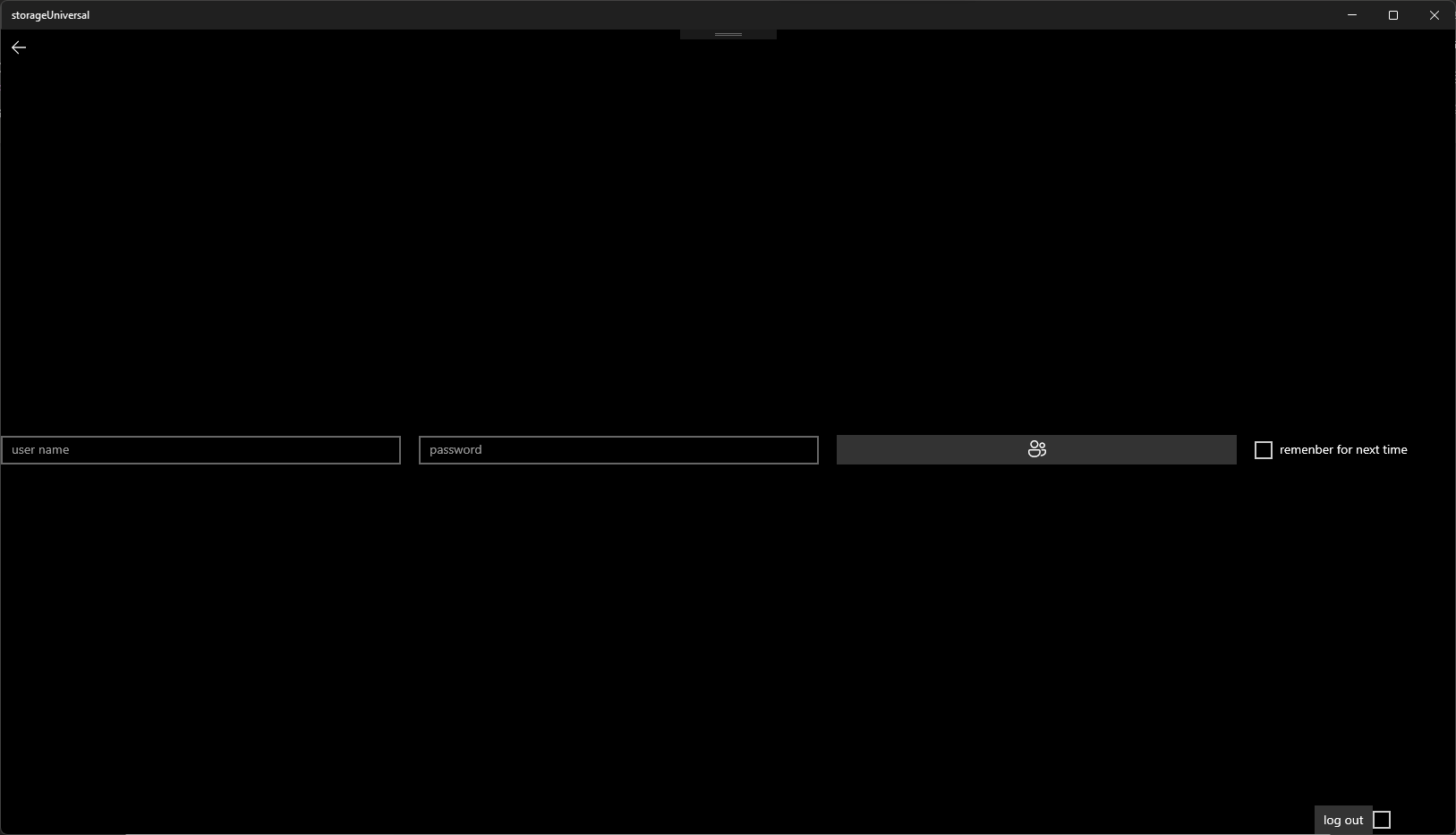


בעמוד זה, המשתמש העליון יכול לראות את הפריטים שהמשתמשים התחתונים ביקשו ממנו. הוא יכול לדחות את הבקשה, או לאשר אותה. מתחת לרשימה של הבקשות המאושרות יש כפתור שליחה. כפתור זה יוסיף את כל ההזמנות המאושרות אל ההשאלות הפעילות.

### מדריך למשתמש התחתון

כמשתמש תחתון, אתה יכול לראות רשימה של הפריטים הזמינים להזמנה. אתה יכול להזמין פריטים, ולראות את מצב הזמנותיך.

#### עמוד התחברות



בעמוד ההתחברות, עליך להזין את שם המשתמש והסיסמה שלך. כדי לשמור את פרטי ההתחברות שלך לפעם הבאה, עליך לסמן את תיבת השמירה. אם ברצונך להתנתק ולמחוק את פרטי החיבור שלך, עליך לסמן את תיבת הבחירה למטה בצד ימין וללחוץ על כפתור ההתנתקות שלידה.

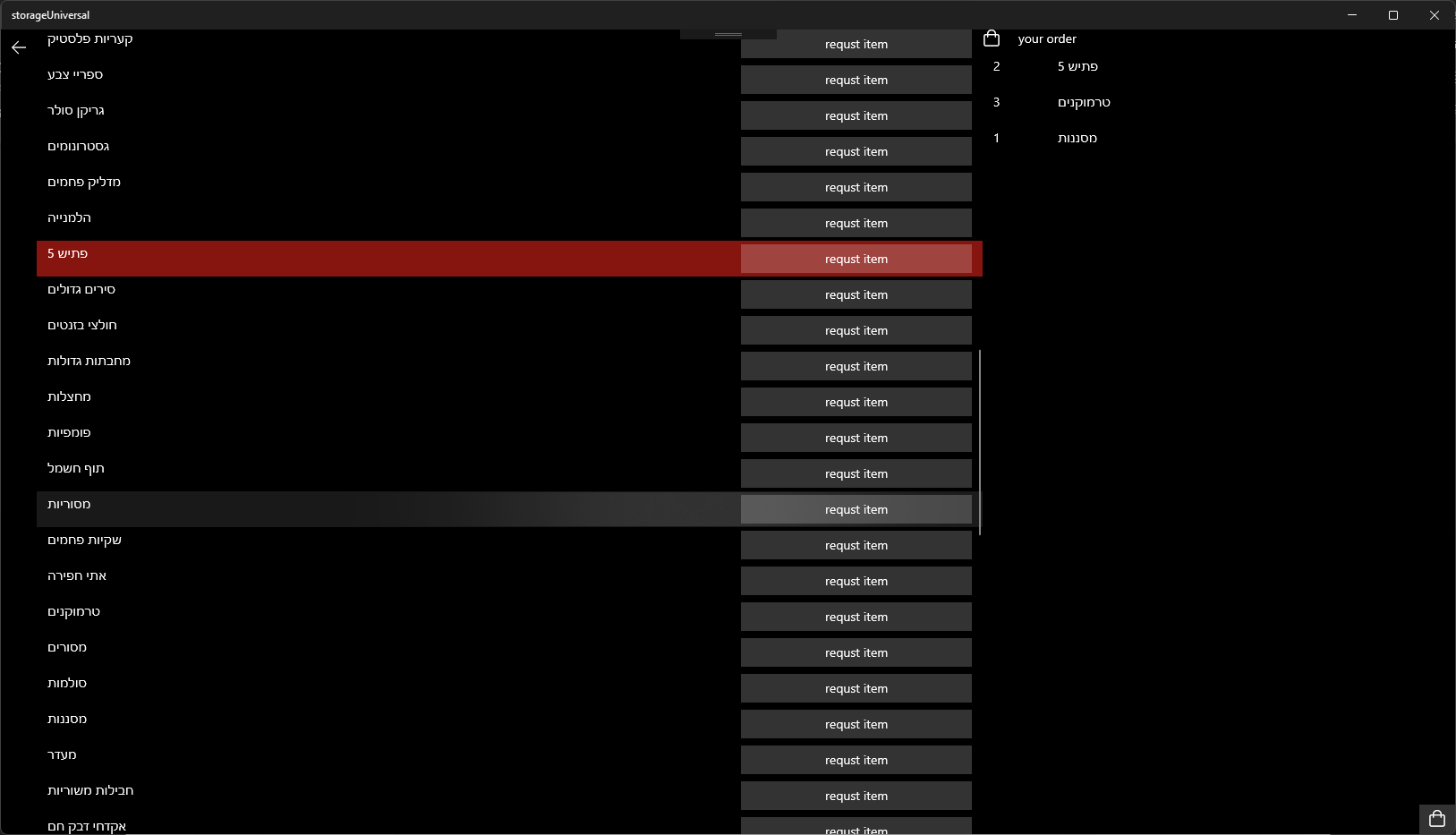
למעבר לעמוד הבית שלך, עליך ללחוץ על כפתור ההתחברות.

#### עמוד הבית



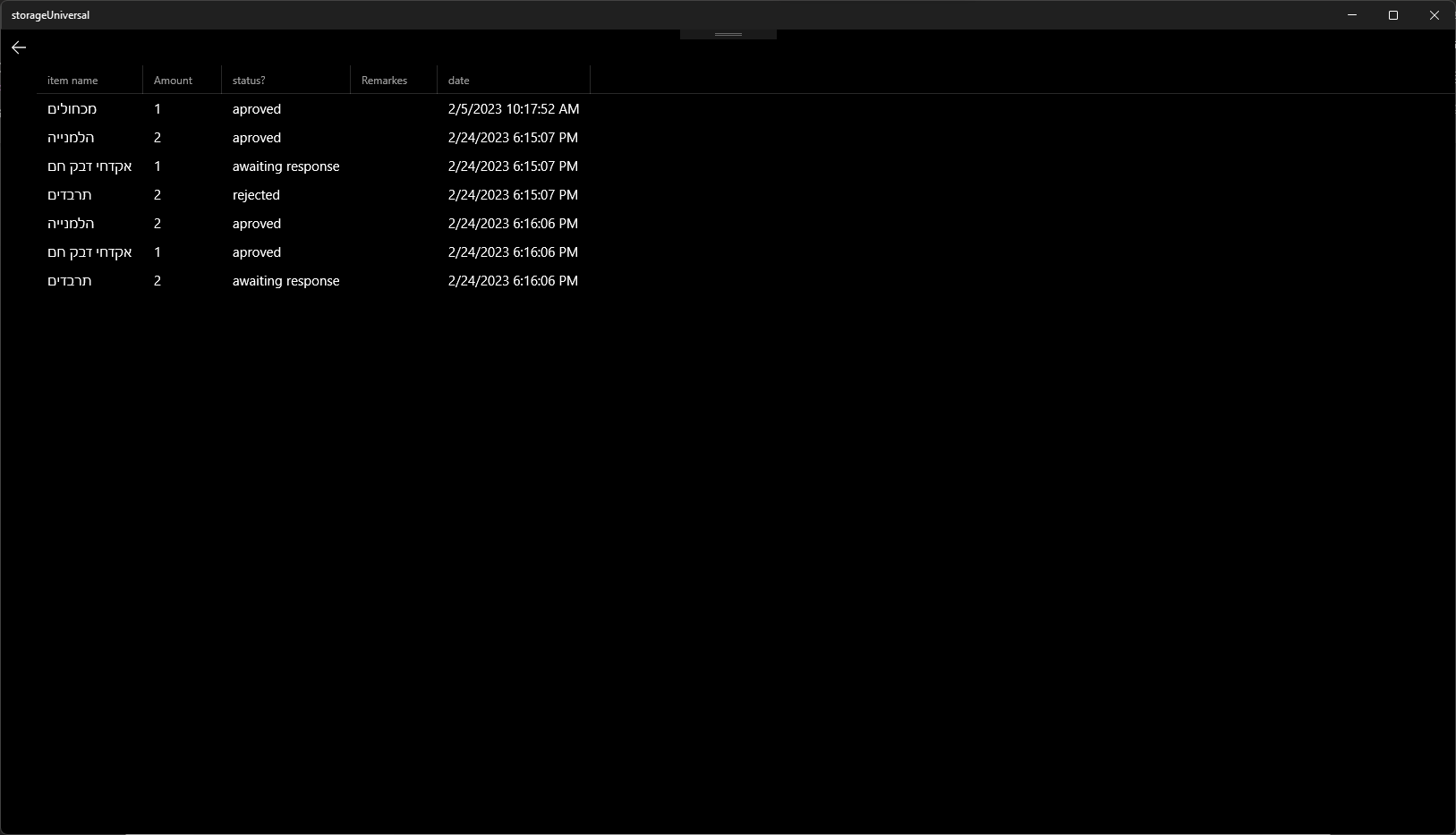
בעמוד זה, אתה יכול לבחור האם ללכת לעמוד שיאפשר לך להזמין ציוד, או לעמוד זו תוכל לראות את מצב ההזמנות הקודמות שלך.

#### עמוד ההזמנה



בעמוד זה, מוצגת לפניך רשימה עם השמות של הפריטים במלאי. עליך ללחוץ על כפתור בקשת הפריט כדי להוסיף אותו להזמנתך. לאחר שסיימת להוסיף את כל הפריטים להזמנה, עליך ללחוץ על כפתור ההזמנה בפינה הימנית-תחתונה של המסך.

#### עמוד צפייה בהזמנות קודמות



בעמוד זה מופיעה לפניך טבלה עם כל ההזמנות הקודמות ששלחת למשתמש העליון שלך ומצבן.

# רפלקציה

במהלך תהליך בניית הפרויקט אני מרגיש שלמדתי רבות אודות נושאים שלא הייתי נחשף אליהם. לפני תחילת העבודה על הפרויקט, לא ידעתי דבר או חצי דבר על נושאים רחבים כמו: קשירת מידע, פונקציות אסינכרוניות ושימוש פרקטי בתכנות מונחה עצמים. אני מרגיש שתהליך בניית הפרוייקט תרם לי מאוד בהרחבת הידע שלי בנושאים אלו, ואילץ אותי ללמוד תכניקות תכנות מתקדמות.

אני מרגיש שהדבר הכי חשוב שלמדתי מתהליך כתיבת הפרויקט הוא היכולת ללמוד בעצמי נושאים. היו מקומות רבים בפרויקט שבהם לא ידעתי לפתור בעיה שבה נתקלתי, בזמנים אלו, אני הייתי חייב לחקור את הנושא בעצמי בעזרת שימוש במשאבים בנושא ברשת- במהלך העבודה על הפרויקט למדתי איך ללמוד לבדי נושאים מגוונים בצורה יעילה.

בהסתכלות לאחור, אני יכול לראות בבירור דברים בפרויקט שהייתי עושה אחרת לחלוטין. אם הייתי בונה עכשיו את הפרויקט מהתחלה עד הסוף- עם כל הניסיון שצברתי מאז, הייתי משתמש באפציות ממשק הרבה יותר מתקדמות כדי לבנות ממשת משתמש יותר נוח ואינטואיטיבי. הייתי עושה שימוש נרחב יותר בקשירת מידע (databinding), בפקדים מתקדמים מאינטרנט שמציגים מידע בצורה נוחה יותר (כמו datagrid של uwp community toolkit) ובטכניקות שונות שהיות חוסכות לי עבודה רבה, ולמדתי רק לאחר כתיבת חלק גדול מהעבודה.

חוץ מזה, העבודה על הפרויקט לימדה אותי שעלי לשקול מספר אופציות לאיך ניתן להשיג מטרה מסויימת, ולבחור באופציה המתאימה ביותר. לדוגמא, כאשר יצרתי את המערכת ששומרת מקומית את פרטי ההתחברות של המשתמשים העליונים, שקלתי רק אופציה אחת- מסד נתונים sqllite מקומי, לקח לי זמן רב ללמוד איך ליצור מסד נתונים שכזה ואיך להשתמש בו. כאשר הגעתי לאותה בעיה במערכת ההתחברות של המשתמשים התחתונים, מצאתי דרך הרבה יותר פשוטה לעשות בעזרת שמירת הגדרות מקומית במערכת windows.

# נספחים:

## הקוד לצד לקוח:

### פקד תיבת סימון עם תיבת סיסמה:

הקוד הויזואלי:

<UserControl

    x:Class="storageUniversal.PasswordWithCheck"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    d:DesignHeight="300"

    d:DesignWidth="400">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="\*"/>

            <ColumnDefinition Width="0.2\*"/>

        </Grid.ColumnDefinitions>

        <PasswordBox x:Name="pass" Grid.Column="0" PlaceholderText="{x:Bind placeHolder}" Password="{x:Bind Text, Mode=TwoWay}" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>

        <CheckBox Grid.Column="1" Checked="CheckBox\_Checked" Unchecked="CheckBox\_Unchecked" ToolTipService.ToolTip="see your password"></CheckBox>

    </Grid>

</UserControl>

הקוד האחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The User Control item template is documented at https://go.microsoft.com/fwlink/?LinkId=234236

namespace storageUniversal

{

    public sealed partial class PasswordWithCheck : UserControl

    {

        public PasswordWithCheck()

        {

            this.InitializeComponent();

        }

        public string Text

        {

            get => (string)GetValue(TextProperty);

            set => SetValue(TextProperty, value);

        }

        public string placeHolder

        {

            get => (string)GetValue(PlaceHolderProp);

            set => SetValue(PlaceHolderProp, value);

        }

        public static readonly DependencyProperty TextProperty =

          DependencyProperty.Register(nameof(Text), typeof(string),

            typeof(PasswordWithCheck), new PropertyMetadata(string.Empty));

        public static readonly DependencyProperty PlaceHolderProp =

  DependencyProperty.Register(nameof(placeHolder), typeof(string),

    typeof(PasswordWithCheck), new PropertyMetadata(string.Empty));

        private void Button\_Click(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void Pass\_GotFocus(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void Pass\_LostFocus(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Hidden;

        }

        private void CheckBox\_Checked(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Visible;

        }

        private void CheckBox\_Unchecked(object sender, RoutedEventArgs e)

        {

            pass.PasswordRevealMode = PasswordRevealMode.Hidden;

        }

    }

}

### העמוד הראשי:

הקוד הויזואלי:

<Page

    x:Class="storageUniversal.MainPage"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Button Grid.Column="0" x:Name="UpUser" Click="UpUser\_Click" VerticalAlignment="Center" HorizontalAlignment="Stretch">I am a Inventory admin</Button>

        <Button Grid.Column="1" x:Name="LowerUser" Click="LowerUser\_Click" VerticalAlignment="Center" HorizontalAlignment="Stretch">I am a user</Button>

        <Button Content="about" Name="about" Click="About\_Click"  HorizontalAlignment="Left" VerticalAlignment="Bottom"/>

    </Grid>

</Page>

הקוד האחורי:

using System;

using System.IO;

using System.Text.RegularExpressions;

using Windows.ApplicationModel.Core;

using Windows.UI.Core;

using Windows.UI.ViewManagement;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using System.Net;

using storageUniversal.xamls;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace storageUniversal

{

    /// <summary>

    /// A basic main page that sends user to login or register

    /// </summary>

    public sealed partial class MainPage : Page

    {

        public static Frame frame;

        public MainPage()

        {

            InitializeComponent();

        }

        override

        protected void OnNavigatedTo(NavigationEventArgs e)

        {

            Frame.IsNavigationStackEnabled = true;

        }

        //sends user to about page

        //שולח את המשתמש לעמוד אודות

        private void About\_Click(object sender, RoutedEventArgs e)

        {

            About.SentBy = typeof(MainPage);

            Frame.Navigate(typeof(About));

        }

        private void UpUser\_Click(object sender, RoutedEventArgs e)

        {

            Frame.Navigate(typeof(UpperLogin));

        }

        private void LowerUser\_Click(object sender, RoutedEventArgs e)

        {

            frame = Frame;

            Frame.Navigate(typeof(LowerLogin));

        }

    }

}

### עמוד אודות

קוד ויזואלי:

<Page

    x:Class="storageUniversal.xamls.About"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal.xamls"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Name="back" Click="Back\_Click" VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <!--for desine flexability, I used HTML + WebView-->

        <WebView Grid.Column="1" Grid.Row="1" Name="webview1" Source="ms-appx-web:///about.html" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using storageUniversal;

using Windows.Storage;

using Windows.UI.Core;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal.xamls

{

    /// <summary>

    /// An empty page that can be used on its own or navigated to within a Frame.

    /// </summary>

    public sealed partial class About : Page

    {

        public static Type SentBy;

        public About()

        {

            this.InitializeComponent();

            //sets so links would be opened in browser instad of localy in the app

            // מגדיר את ההצגה כך שקישורים חיצוניים יפתחו בדפדפן ולא בתוך הפקד

            webview1.NavigationStarting += async (webViewSender, args) =>

            {

                // Cancel the navigation

                args.Cancel = true;

                // Get the URI of the link that was clicked

                var uri = args.Uri;

                // Open the link in the default external browser

                var success = await Windows.System.Launcher.LaunchUriAsync(uri);

                if (success)

                {

                    // The link was opened successfully

                }

                else

                {

                    // An error occurred, the link could not be opened

                }

            };

            // some code to handle mouse back + forward buttons

            // קוד להוספת פונקציית קדימה ואחורה לכפתורי העכבר

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        // קוד להוספת פונקציית קדימה ואחורה לכפתורי העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //Back button

        //קוד לכפתור אחורה- שולח את המשתמש לעמוד הקודם

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

    }

}

קוד html לעמוד אודות

כיוון שעמוד זה נכתב על ידי תוכנת Word, הקוד שלו כולל מעל 800 שורות, ולכן לא אצרף אותו.

### עמוד התחברות: משתמש עליון

קוד ויזואלי

<Page

    x:Class="storageUniversal.UpperLogin"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!--made by yehonatan vishna-->

    <Grid Name="g1" ColumnSpacing="25">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Grid.Row="0" Grid.Column="0" Name="back" Click="Back\_Click" VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <Button Grid.Row="1" Grid.Column="0" x:Name="RegBut" Click="RegBut\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Left" ToolTipService.ToolTip="register">

            <SymbolIcon Symbol="OtherUser"></SymbolIcon></Button>

        <Grid Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2" ColumnSpacing="20">

            <Grid.ColumnDefinitions>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

            </Grid.ColumnDefinitions>

            <Grid Grid.Column="0">

                <Grid.RowDefinitions>

                    <RowDefinition></RowDefinition>

                    <RowDefinition></RowDefinition>

                    <RowDefinition></RowDefinition>

                </Grid.RowDefinitions>

                <TextBox Name="email" Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Stretch"  PlaceholderText="email" VerticalAlignment="Center" />

                <TextBlock Name="IsEmailValidBlock" Grid.Row="0" TextWrapping="Wrap" HorizontalAlignment="Stretch" VerticalAlignment="Bottom" IsColorFontEnabled="True" Foreground="Red" Text=""/>

            </Grid>

            <PasswordBox  Name="password" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center"  PlaceholderText="password" />

            <TextBlock Visibility="Collapsed" Name="res"  Text="" TextWrapping="Wrap" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

            <Button  Grid.Column="2" Name="sendToInventoryTbl" ToolTipService.ToolTip="this button sends you to see your inventory table" Click="LoginAndSendToHome" HorizontalAlignment="Stretch" VerticalAlignment="Center">

                <FontIcon FontFamily="{StaticResource SymbolThemeFontFamily}" Glyph="&#xE8AD;" />

            </Button>

            <CheckBox Grid.Column="3" Name="rememberBox" Content="remenber for next time" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

        </Grid>

        <Grid Grid.Column="1" Grid.Row="1" HorizontalAlignment="Right" VerticalAlignment="Bottom" >

            <Grid.ColumnDefinitions>

                <ColumnDefinition Width="4\*"/>

                <ColumnDefinition Width="0.25\*"/>

            </Grid.ColumnDefinitions>

            <Button Grid.Column="0" x:Name="logout" Click="Logout\_Click">

                log out

            </Button>

            <CheckBox Grid.Column="1" x:Name="forget\_saved\_users" ToolTipService.ToolTip="forget all saved users"></CheckBox>

        </Grid>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net.Mail;

using System.Runtime.InteropServices.WindowsRuntime;

using System.Windows.Input;

using Windows.ApplicationModel.Core;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Core;

using Windows.UI.ViewManagement;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using storageUniversal.codes;

using System.Threading.Tasks;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal

{

    /// <summary>

    /// The page gets the users email and password and signs him in

    /// </summary>

    public sealed partial class UpperLogin : Page

    {

        public static UserDBServ.User usr = new UserDBServ.User();

        public static UserDBServ.User FullUser;

        private UserDBServ.UserDBServSoapClient UDBS = new UserDBServ.UserDBServSoapClient();

        public static Type SentFrom;

        //שולח את המשתמש לעמוד הבית אם הוא כבר התחבר

        public async void gotoHome()

        {

            await Task.Delay(1);

            Frame.Navigate(typeof(xamls.UpperUserHomePage));

        }

        public UpperLogin()

        {

            this.InitializeComponent();

            // some code to handle mouse back + forward buttons

            // קוד להוספת פונקציית קדימה ואחורה לכפתורי העכבר

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

            var frame = Window.Current.Content as Frame;

            var sender = frame.SourcePageType;

            if(sender == typeof(MainPage))

            {

                if(FullUser != null)

                {

                    gotoHome();

                }

                else

                {

                    TryStartAutoLoging();

                }

            }

        }

        // some code to handle mouse back + forward buttons

        // קוד להוספת פונקציית קדימה ואחורה לכפתורי העכבר

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //checking if the user has previusly signed in, if he did, offering him to use previus login info

        //בודק האם המשתמש כבר נכנס בעבר ממחשב זה,אם כן מציע לו להשתמש בחיבור לפי הפרטים שנשמרו

        private async void TryStartAutoLoging()

        {

            var db = new UsersDatabase();

            var a = await db.GetItemsAsync();

            if (a.Count > 0)

            {

                //in case there is alredy content dialog up, using try to avoid erro

                try

                {

                    //הודעת ההצעה

                    ContentDialog getLandingDits = new ContentDialog()

                    {

                        Title = "do you wish to use saved user?",

                        SecondaryButtonText = "yes",

                        SecondaryButtonCommand = new logSavedUser(),

                        SecondaryButtonCommandParameter = this,

                        CloseButtonText = "no"

                    };

                    await getLandingDits.ShowAsync();

                }

                catch { }

            }

        }

        //tryes to log user in

        // מנסה לחבר את המשתמש אוטומטית, כך שאם לא מצליח לא מעלה שגיאה

        public static async void logAutoCheck(Frame frame, TextBlock res)

        {

            var db = new UsersDatabase();

            var a = await db.GetItemsAsync();

            try

            {

                    autoLogin(a[0], frame, res);

            }

            catch { }

        }

        //logs in user autologin

        // מכניס את המשתמש בכניסה אוטומטית לפי הפרטים השמורים

        public static async void autoLogin(User user, Frame frame, TextBlock res)

        {

            var UDBS = new UserDBServ.UserDBServSoapClient();

            usr.Password = user.Password;

            usr.Email = user.Email;

            var a = await UDBS.IsUserPermittedAsync(new UserDBServ.User() {Password= usr.Password, Email = usr.Email });

            bool b = bool.Parse(a.ToString());

            if (b)

            {

                var TempFullUsr = await UDBS.GetFullUserAsync(usr);

                FullUser = TempFullUsr;

                frame.Navigate(typeof(xamls.UpperUserHomePage));

            }

            else

            {

                var fail = new ContentDialog() { Title = "email or password are wrong, try again", CloseButtonText = "ok" };

                await fail.ShowAsync();

            }

        }

        //the command that passes user data to next page from auto login menu

        //הקומנד שיפעל ברגע שהמשתמש יבחר שהוא רוצה להתחבר עם הפרטים השמורים

        public class logSavedUser : ICommand

        {

            public event EventHandler CanExecuteChanged;

            public bool CanExecute(object parameter)

            {

                throw new NotImplementedException();

            }

            public async void Execute(object parameter)

            {

                var logPg = (parameter as UpperLogin);

                logAutoCheck(logPg.Frame, logPg.res);

            }

        }

        //logs in user and sends him to his home page

        // מחבר את המשתמש, ושולח אותו לדף הבית

        private async void LoginAndSendToHome(object sender, RoutedEventArgs e)

        {

            usr.Password = password.Password;

            usr.Email = email.Text;

            if (!isEmailValid(usr.Email))

            {

                IsEmailValidBlock.Text = "email address isn't valid";

                return;

            }

            var a = await UDBS.IsUserPermittedAsync(usr);

            if(await UDBS.IsAdminAsync(usr))

            {

                FullUser = new UserDBServ.User() { Email = usr.Email, Password = usr.Password };

                Frame.Navigate(typeof(AdminPanel));

                return;

            }

            var isval = isEmailValid(usr.Email);

            bool b = bool.Parse(a.ToString());

            if (b)

            {

                //raises sucsess popup

                //מעלה הודעת הצלחה

                var sucsess = new ContentDialog() { Title = "user exists, data should be showen", CloseButtonText="ok" };

                await sucsess.ShowAsync();

                var TempFullUsr = await UDBS.GetFullUserAsync(usr);

                FullUser = TempFullUsr;

                if (rememberBox.IsChecked.Value)

                {

                    var db = new UsersDatabase();

                    await db.DeleteAll();

                    await db.InsertItemAsync(new User() { ID = FullUser.ID, Email = FullUser.Email, Password = FullUser.Password , BDate = FullUser.BDate.Value, Compeny = FullUser.Compeny, Fname = FullUser.Fname, Lname = FullUser.Lname});

                }

                this.Frame.Navigate(typeof(xamls.UpperUserHomePage));

            }

            else

            {

                //raises failiur popup

                //מעלה הודעת שגיאה

                var fail = new ContentDialog() { Title = "email or password are wrong, try again", CloseButtonText = "ok" };

                await fail.ShowAsync();

            }

        }

        //בודק האם האימייל שהתקבל תקין בתחביר שלו

        //checks whether the recived email is valid

        public bool isEmailValid(String emailaddress)

        {

            try

            {

                MailAddress m = new MailAddress(emailaddress);

                foreach(char a in emailaddress.ToList())

                {

                    if (a < 32 || a>122)

                    {

                        return false;

                    }

                }

                return true;

            }

            catch

            {

                return false;

            }

        }

        //send user back to previus page

        //כפתור אחורה- שולח את המשתמש לעמוד הקודם

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

        private void RegBut\_Click(object sender, RoutedEventArgs e)

        {

            Frame.Navigate(typeof(Register));

        }

        //מנתק את המשתמש על ידי איפוס משנה המשתמש

        private void Logout\_Click(object sender, RoutedEventArgs e)

        {

            FullUser = null;

            if (forget\_saved\_users.IsChecked.GetValueOrDefault())

            {

                var db = new UsersDatabase();

                db.DeleteAll();

            }

        }

    }

}

### עמוד הרשמה- משתמש עליון

קוד ויזואלי:

<Page

    x:Class="storageUniversal.Register"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid ColumnSpacing="40">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Name="back" Grid.Column="0" Grid.Row="0" Click="Back\_Click" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <TextBox Name="FN" Grid.Column="1" Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Stretch" PlaceholderText="first name" VerticalAlignment="Center" Width="Auto"/>

        <TextBox Name="LN" Grid.Column="2" Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Stretch" PlaceholderText="Last Name" VerticalAlignment="Center"/>

        <CalendarDatePicker Name="BDate" Grid.Column="3" Grid.Row="1"  PlaceholderText="enter your birth date"  HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

        <TextBox Name="compeny" Grid.Column="4" Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Stretch" PlaceholderText="compeny" VerticalAlignment="Center"/>

        <Grid Grid.Column="5" Grid.Row="1">

            <Grid.RowDefinitions>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

            </Grid.RowDefinitions>

            <TextBox Name="email" Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Stretch" VerticalAlignment="Center" PlaceholderText="email"/>

            <!--a place for invalid password warning-->

            <TextBlock Name="IsEmailValidBlock" Grid.Row="0" TextWrapping="Wrap" HorizontalAlignment="Stretch" VerticalAlignment="Bottom" IsColorFontEnabled="True" Foreground="Red" Text=""/>

        </Grid>

        <Grid Grid.Column="6" Grid.Row="1">

            <Grid.RowDefinitions>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

            </Grid.RowDefinitions>

            <PasswordBox Name="pass" Grid.Row="1" PlaceholderText="password" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

            <PasswordBox Name="passAgain" Grid.Row="2" PlaceholderText="enter password again" HorizontalAlignment="Stretch" VerticalAlignment="Top"/>

            <!--a place for invalid password warning-->

            <TextBlock Name="IsPasswordValidBlock" Grid.Row="0" TextWrapping="Wrap" HorizontalAlignment="Stretch" VerticalAlignment="Bottom" IsColorFontEnabled="True" Foreground="Red" Text=""/>

        </Grid>

        <Button Name="regBot" Grid.Column="7" Grid.Row="1" Click="RegBot\_Click" Content="register" HorizontalAlignment="Stretch" VerticalAlignment="Center" />

        <TextBlock Visibility="Collapsed" Name ="isDone"  HorizontalAlignment="Left" Margin="263,341,0,0" Text="" VerticalAlignment="Top"/>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using System.Threading;

using System.Threading.Tasks;

using Windows.UI.Core;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal

{

    /// <summary>

    /// this page allows the user to register (add his ditails db)

    /// </summary>

    public sealed partial class Register : Page

    {

        public static Type SentFrom;

        public Register()

        {

            this.InitializeComponent();

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            // some code to handle mouse back + forward buttons

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //בודק האם האימייל שהתקבל תקין בתחביר שלו

        //checks whether the recived email is valid

        public bool isEmailValid(String email)

        {

            var trimmedEmail = email.Trim();

            if (trimmedEmail.EndsWith("."))

            {

                return false;

            }

            try

            {

                var addr = new System.Net.Mail.MailAddress(email);

                return addr.Address == trimmedEmail;

            }

            catch

            {

                return false;

            }

        }

        // בודק האם האימייל שהתקבל כבר בשימוש במסד הנתונים

        //checks whether the email alredy axists in db

        public async Task<bool> DoesEmailExistAlready(string email)

        {

            var UDBS = new UserDBServ.UserDBServSoapClient();

            return await UDBS.DoesEmailExistAsync(email);

        }

        //מגיב ללחיצה על כפתור ההרשמה, רושם את המשתמש לשירות

        //responds to click on register button, registers user

        private async void RegBot\_Click(object sender, RoutedEventArgs e)

        {

            UserDBServ.UserDBServSoapClient UDBS = new UserDBServ.UserDBServSoapClient();

            UserDBServ.User usr = new UserDBServ.User();

            //בודק האם האימיל והססמה תקינים ולא קיימים במערכת

            if (!isEmailValid(email.Text)) {

                IsEmailValidBlock.Text = "email is not valid";

                return;

            }

            bool exists = await DoesEmailExistAlready(email.Text);

            if (exists)

            {

                IsEmailValidBlock.Text = "user with this email already exists, use another email adress"; return;

            }

            if (!pass.Password.Equals(passAgain.Password))

            {

                IsPasswordValidBlock.Text = "passwords don't match, try again"; return;

            }

            var date = BDate.Date;

            if (date != null)

            {

                DateTime time = date.Value.DateTime;

                usr.BDate = time;

                //בודק שהמשתמש לא צעיר מידי

                var age = DateTime.Now - time;

                if(age.TotalDays < ((int)(365.25 \* 10)))

                {

                    ContentDialog AgeDialog = new ContentDialog()

                    {

                        Title = "you must be 10 or older to use this service",

                        CloseButtonText = "ok"

                    };

                    await AgeDialog.ShowAsync();

                    return;

                }

            }

            usr.Fname = FN.Text;

            usr.Lname = LN.Text;

            usr.Email = email.Text;

            usr.Password = pass.Password;

            usr.Compeny = compeny.Text;

            //בודק האם כל השדות מלאים

            if(!date.HasValue || usr.Fname == ""|| usr.Lname == ""|| usr.Email == ""|| usr.Password == ""|| usr.Compeny == "") {

                ContentDialog fail = new ContentDialog()

                {

                    Title = "Make sure you fill out all the fileds",

                    CloseButtonText = "ok"

                };

                await fail.ShowAsync();

                return; }

            var a = await UDBS.regAsync(usr);

            bool IsSuccess = bool.Parse(a.ToString());

            //מעלה למשתמש הודעה- האם ההרשמה הסתיימה בהצלחה או לא

            // raises a popup for the user- tells him wether the regestation was completed sucsessfully

            if (IsSuccess)

            {

                ContentDialog sucsessDialog = new ContentDialog()

                {

                    Title = "regestration completed sucssesfully",

                    CloseButtonText = "ok"

                };

                await sucsessDialog.ShowAsync();

                Frame.Navigate(typeof(MainPage));

            }

            else

            {

                ContentDialog FailDialog = new ContentDialog()

                {

                    Title = "regestration failed, please try again",

                    CloseButtonText = "ok"

                };

                await FailDialog.ShowAsync();

            }

        }

        // מגיב ללחיצה על כפתור החזרה, מחזיר את המשתמש למסך הקודם

        //go back to previus page

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

    }

}

### עמוד בית- משתמש עליון:

קוד ויזואלי:

<Page

    x:Class="storageUniversal.xamls.UpperUserHomePage"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal.xamls"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid ColumnSpacing="50">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button x:Name="back" Grid.Column="0" Grid.Row="0" Click="Back\_Click" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <TextBlock x:Name="helloMsg" Grid.ColumnSpan="2" Grid.Column="1" Grid.Row="0" Text="{x:Bind UseName, Mode=TwoWay}"></TextBlock>

        <Button  Grid.Column="1" Grid.Row="1" Name="sendToInventoryTbl" ToolTipService.ToolTip="this button sends you to see your inventory table" Click="LoginAndSendToInventoryList" HorizontalAlignment="Stretch" VerticalAlignment="Center">

            <SymbolIcon Symbol="List"/>

        </Button>

        <Button x:Name="ManegeSubUsers" Grid.Column="2" Grid.Row="1" Click="ManegeSubUsers\_Click" HorizontalAlignment="Stretch">manege you users</Button>

        <Button x:Name="GoToChangeDitailes" Grid.Column="3" Grid.Row="1" Click="GoToChangeDitailes\_Click" HorizontalAlignment="Stretch">change your personal ditailes</Button>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Core;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal.xamls

{

    /// <summary>

    /// An empty page that can be used on its own or navigated to within a Frame.

    /// </summary>

    public sealed partial class UpperUserHomePage : Page

    {

        private string UseName = "Hello "+ UpperLogin.FullUser.Fname + UpperLogin.FullUser.Lname;

        public UpperUserHomePage()

        {

            this.InitializeComponent();

            // some code to handle mouse back + forward buttons

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //logs in user and sends him to his invntory list

        // מחבר את המשתמש, ושולח אותו לרשימת המלאי שלו

        private void LoginAndSendToInventoryList(object sender, RoutedEventArgs e)

        {

            var usr = UpperLogin.FullUser;

            this.Frame.Navigate(typeof(InventoryView));

        }

        //the button sends the user to manege his subusers

        //הכפטור שולח את המשתמש לנהל את המשתמשים שמשוייכים אליו

        private void ManegeSubUsers\_Click(object sender, RoutedEventArgs e)

        {

            Frame.Navigate(typeof(SubUsersManegment));

        }

        //the button sends the user to edit his ditailes

        //הכפטור שולח את המשתמש לערוך את פרטיו האישיים

        private void GoToChangeDitailes\_Click(object sender, RoutedEventArgs e)

        {

            Frame.Navigate(typeof(updateUser));

        }

        //מחזיר את המשתמש לעמוד הקודם

        //goes back to previus page

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

    }

}

### עמוד שינוי פרטים- משתמש עליון:

קוד ויזואלי:

<Page

    x:Class="storageUniversal.updateUser"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Name="back" Grid.Column="0" Grid.Row="0" Click="Back\_Click" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <Grid Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2">

            <Grid.ColumnDefinitions>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

                <ColumnDefinition></ColumnDefinition>

            </Grid.ColumnDefinitions>

            <TextBox Name="FN" Grid.Column="0" PlaceholderText="first name" VerticalAlignment="Center" HorizontalAlignment="Stretch" />

            <TextBox Name="LN" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center"  PlaceholderText="Last Name"/>

            <CalendarDatePicker Name="BDate" Grid.Column="2" PlaceholderText="this is your birthday" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

            <TextBox Name="compeny" Grid.Column="3" PlaceholderText="compeny" VerticalAlignment="Center" HorizontalAlignment="Stretch"/>

            <TextBox Name="newemail" Grid.Column="4" PlaceholderText="email" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

            <local:PasswordWithCheck x:Name="pass" Grid.Column="5" placeHolder="password" HorizontalAlignment="Stretch" VerticalAlignment="Center"/>

            <Button Name="UpdateUsr" Grid.Column="6" Click="Update\_Click"  HorizontalAlignment="Stretch" VerticalAlignment="Center">

                <SymbolIcon Symbol="Accept" HorizontalAlignment="Stretch"></SymbolIcon>

            </Button>

            <Button Grid.Column="7" Name="DelBot" ToolTipService.ToolTip="this button would delete your user from this service" Click="DelBot\_Click"  HorizontalAlignment="Stretch" VerticalAlignment="Center">

                <SymbolIcon Symbol="Delete"></SymbolIcon>

            </Button>

        </Grid>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using storageUniversal;

using System.Threading;

using Windows.UI.Core;

//made by yehonatan vishna

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal

{

    /// <summary>

    /// a page that allows user to update his personal details

    /// </summary>

    public sealed partial class updateUser : Page

    {

        private UserDBServ.UserDBServSoapClient UDBS = new UserDBServ.UserDBServSoapClient();

        public static UserDBServ.User FullUser = UpperLogin.FullUser;

        public static Type SentFrom;

        public updateUser()

        {

            this.InitializeComponent();

            //מכניס לשדות את הנתונים הנוחכיים של המשתמש

            //fills in user current data

            FN.Text = FullUser.Fname;

            LN.Text = FullUser.Lname;

            newemail.Text = FullUser.Email;

            DateTime time = FullUser.BDate.Value;

            BDate.Date = time.Date;

            compeny.Text = FullUser.Compeny;

            pass.Text = FullUser.Password;

            // some code to handle mouse back + forward buttons

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //שולח לשירות הרשת את הנתונים העדכניים של המשתמש

        //sends motified data to web service

        private async void Update\_Click(object sender, RoutedEventArgs e)

        {

            UserDBServ.User NewUser = new UserDBServ.User();

            var date = BDate.Date;

            DateTime time = date.Value.DateTime;

            NewUser.BDate = time;

            NewUser.Fname = FN.Text;

            NewUser.Lname = LN.Text;

            NewUser.Email = newemail.Text;

            NewUser.Password = pass.Text;

            NewUser.Compeny = compeny.Text;

            var resTemp = await UDBS.updateUserAsync(FullUser, NewUser);

            bool res = resTemp;

            if (res)

            {

                var secsussPop = new ContentDialog()

                {

                    Title = "Detailes Update Have Been Sucsussfully Completed", CloseButtonText="ok"

                };

                await secsussPop.ShowAsync();

                FullUser = NewUser;

                Frame.Navigate(typeof(UpperLogin));

            }

            else

            {

                var failPop = new ContentDialog()

                {

                    Title = "Detailes Update Have Been Unsecsussfull", Content = "Please try again.",

                    CloseButtonText = "ok"

                };

                await failPop.ShowAsync();

            }

            }

        //deletes user from system

        //מוחק את המשתמש

        private async void DelBot\_Click(object sender, RoutedEventArgs e)

        {

            var usr = UpperLogin.FullUser;

            var a = await UDBS.DeleteUserAsync(usr);

            bool b = bool.Parse(a.ToString());

        }

        //מחזיר את המשתמש לעמוד הקודם

        //goes back to previus page

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

    }

    }

### עמוד הוספת משתמשים תחתונים:

קוד ויזואלי:

<Page

    x:Class="storageUniversal.xamls.SubUsersManegment"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal.xamls"

    xmlns:data="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"/>

            <ColumnDefinition/>

        </Grid.ColumnDefinitions>

        <StackPanel Orientation="Vertical">

            <Button  Name="back" Click="Back\_Click"  VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

            <Button Name="addUser" Click="AddUser\_Click" VerticalAlignment="Top" ToolTipService.ToolTip="add user">

                <SymbolIcon Symbol="Add"/>

            </Button>

            <Button  Name="refresh" Click="Refresh\_Click" VerticalAlignment="Top" ToolTipService.ToolTip="refresh">

                <SymbolIcon Symbol="Refresh"></SymbolIcon>

            </Button>

            <Button Name="sync" Click="Sync\_Click" VerticalAlignment="Top" ToolTipService.ToolTip="sync">

                <SymbolIcon Symbol="Sync"></SymbolIcon>

            </Button>

            <Button Name="delete" Click="Delete\_Click" VerticalAlignment="Top" ToolTipService.ToolTip="delete selected row">

                <SymbolIcon Symbol="Delete"></SymbolIcon>

            </Button>

        </StackPanel>

        <ListView Grid.Column="1" Grid.Row="1" Name="SubUsersTbl" HorizontalContentAlignment="Stretch" HorizontalAlignment="Stretch" SelectionMode="Extended">

            <ListView.Header>

                <Grid HorizontalAlignment="Stretch" RightTapped="Grid\_RightTapped">

                    <Grid.ColumnDefinitions>

                        <ColumnDefinition/>

                        <ColumnDefinition/>

                        <ColumnDefinition/>

                        <ColumnDefinition/>

                        <ColumnDefinition/>

                        <ColumnDefinition/>

                    </Grid.ColumnDefinitions>

                    <TextBlock Grid.Column="0" HorizontalTextAlignment="Center">User First Name</TextBlock>

                    <TextBlock Grid.Column="1" HorizontalTextAlignment="Center">User Last Name</TextBlock>

                    <TextBlock Grid.Column="2" HorizontalTextAlignment="Center">Role</TextBlock>

                    <TextBlock Grid.Column="3" HorizontalTextAlignment="Center">Email</TextBlock>

                    <TextBlock Grid.Column="4" HorizontalTextAlignment="Center">UserName</TextBlock>

                    <TextBlock Grid.Column="5" HorizontalTextAlignment="Center">Password</TextBlock>

                </Grid>

            </ListView.Header>

            <ListView.ItemTemplate>

                <DataTemplate  x:Name="dataTemp" x:DataType="data:SubUser" >

                    <Grid HorizontalAlignment="Stretch" RightTapped="Grid\_RightTapped">

                        <Grid.ColumnDefinitions>

                            <ColumnDefinition/>

                            <ColumnDefinition/>

                            <ColumnDefinition/>

                            <ColumnDefinition/>

                            <ColumnDefinition/>

                            <ColumnDefinition/>

                        </Grid.ColumnDefinitions>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="0" Name="FN" Text="{Binding FName, Mode=TwoWay}"></TextBox>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="1" Name="LN" Text="{Binding LName, Mode=TwoWay}"></TextBox>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="2" Name="Role" Text="{Binding Role, Mode=TwoWay}"></TextBox>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="3" Name="Email" Text="{Binding Email, Mode=TwoWay}"></TextBox>

                        <!--<PasswordBox HorizontalAlignment="Stretch" Grid.Column="4" Name="Password" Password="{Binding Role, Mode=TwoWay}" PasswordRevealMode ="Peek" IsPasswordRevealButtonEnabled="True"/>-->

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="4" Name="UserName" Text="{Binding UserName, Mode=TwoWay}"></TextBox>

                        <data:PasswordWithCheck Grid.Column="5" Text="{Binding Password, Mode=TwoWay}"></data:PasswordWithCheck>

                        <TextBlock Visibility="Collapsed" Name="ID" Text="{Binding Id}"></TextBlock>

                    </Grid>

                </DataTemplate>

            </ListView.ItemTemplate>

            <ListView.ItemContainerStyle>

                <Style TargetType="ListViewItem">

                    <Setter Property="HorizontalContentAlignment"  Value="Stretch"></Setter>

                    <Setter Property="HorizontalAlignment"  Value="Stretch"></Setter>

                </Style>

            </ListView.ItemContainerStyle>

        </ListView>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Core;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal.xamls

{

    /// <summary>

    /// An empty page that can be used on its own or navigated to within a Frame.

    /// </summary>

    public sealed partial class SubUsersManegment : Page

    {

        //רשימה המכילה את כל המשתמשים כפי שהיו אחרי העדכון האחרון

        //user list as it is in the last refresh

        private List<SubUser> UsersOriginal;

        //רשימה של כל המשתמשים כפי שהם כעת בטבלה

        //users as they are after the changes the user made

        private ObservableCollection<SubUser> BinedUsersInTbl = new ObservableCollection<SubUser>();

        public SubUsersManegment()

        {

            this.InitializeComponent();

            loudTbl();

        }

        public async void LoadTbl()

        {

            loudTbl();

        }

        //מכניס את כל המשתמשים השייכים למשתמש זה לטבלה

        //loads all users's users from db and inserts them to the table (ListView)

        public async void loudTbl()

        {

            var s = new SubUserServ.SubUsersServSoapClient();

            var TmpUsr = UpperLogin.FullUser;

            var usr = new SubUserServ.User() { ID = TmpUsr.ID, BDate = TmpUsr.BDate, Compeny = TmpUsr.Compeny, Email = TmpUsr.Email, Fname = TmpUsr.Fname, Lname = TmpUsr.Lname, Password = TmpUsr.Password };

            var r = await s.getYourSubUsersAsync(usr);

            List<SubUser> Users = new List<SubUser>();

            foreach (DataRow a in r.Rows)

            {

                SubUser row = new SubUser();

                row.Id = int.Parse(a["ID"].ToString());

                row.FName = a["FName"].ToString();

                row.LName = a["LName"].ToString();

                row.Email = a["Email"].ToString();

                row.BelongsToUpperUser = int.Parse(a["BelongsToUpperUser"].ToString());

                row.Role = a["Role"].ToString();

                row.Password = a["Password"].ToString();

                row.UserName = a["UserName"].ToString();

                Users.Add(row);

            }

            if (BinedUsersInTbl.ToList().Count > 0)

            {

                BinedUsersInTbl.Clear();

            }

            foreach (SubUser row in Users)

            {

                BinedUsersInTbl.Add(row);

            }

            SubUsersTbl.ItemsSource = BinedUsersInTbl;

            UsersOriginal = new List<SubUser>();

            foreach (SubUser Row in Users)

            {

                UsersOriginal.Add(Row.Copy());

            }

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            // some code to handle mouse back + forward buttons

            var win = new codes.backMouse();

        }

        //send user back to previus page

        //כפתור אחורה- שולח את המשתמש לעמוד הקודם

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

        private void AddUser\_Click(object sender, RoutedEventArgs e)

        {

            addUserF();

        }

        //פונקציה שמוסיפה משתמש ריק למסד הנתונים ולטבלה

        private async void addUserF()

        {

            var sub = new SubUserServ.SubUsersServSoapClient();

            var usr = new SubUserServ.SubUser() { BelongsToUpperUser = UpperLogin.FullUser.ID };

            var getNewDetails = new StackPanel() { Orientation = Orientation.Vertical, DataContext=usr};

            //מוסיף שדה שם פרטי להודעה

            Binding FirstNameBinding = new Binding();

            var FNBox = new TextBox() { PlaceholderText = "The new user's first name." };

            FirstNameBinding.Source = usr.FName;

            FirstNameBinding.Path = new PropertyPath("FName");

            FirstNameBinding.Mode = BindingMode.TwoWay;

            FirstNameBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(FNBox, TextBox.TextProperty, FirstNameBinding);

            getNewDetails.Children.Add(FNBox);

            //מוסיף שדה שם פרטי להודעה

            Binding LastNameBinding = new Binding();

            var LNBox = new TextBox() { PlaceholderText = "The new user's last name." };

            LastNameBinding.Source = usr.LName;

            LastNameBinding.Path = new PropertyPath("LName");

            LastNameBinding.Mode = BindingMode.TwoWay;

            LastNameBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(LNBox, TextBox.TextProperty, LastNameBinding);

            getNewDetails.Children.Add(LNBox);

            //מוסיף שדה תפקיד להודעה

            Binding RoleBinding = new Binding();

            var RoleBox = new TextBox() { PlaceholderText = "The new user's role." };

            RoleBinding.Source = usr.Role;

            RoleBinding.Path = new PropertyPath("Role");

            RoleBinding.Mode = BindingMode.TwoWay;

            RoleBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(RoleBox, TextBox.TextProperty, RoleBinding);

            getNewDetails.Children.Add(RoleBox);

            //מוסיף שדה שם משתמש להודעה

            Binding UserNameBinding = new Binding();

            var UserName = new TextBox() {PlaceholderText="The new user's username."};

            UserNameBinding.Source = usr.UserName;

            UserNameBinding.Path = new PropertyPath("UserName");

            UserNameBinding.Mode = BindingMode.TwoWay;

            UserNameBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(UserName, TextBox.TextProperty, UserNameBinding);

            getNewDetails.Children.Add(UserName);

            //מוסיף שדה ססמה להודעה

            Binding PasswordBinding = new Binding();

            var PassBox = new PasswordBox() { PlaceholderText = "The new user's password." };

            PasswordBinding.Source = usr.Password;

            PasswordBinding.Path = new PropertyPath("Password");

            PasswordBinding.Mode = BindingMode.TwoWay;

            PasswordBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(PassBox, PasswordBox.PasswordProperty, PasswordBinding);

            getNewDetails.Children.Add(PassBox);

            //מוסיף שדה דואל להודעה

            Binding EmailBinding = new Binding();

            var EmailBox = new TextBox() { PlaceholderText = "The new user's email." };

            EmailBinding.Source = usr.Email;

            EmailBinding.Path = new PropertyPath("Email");

            EmailBinding.Mode = BindingMode.TwoWay;

            EmailBinding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

            BindingOperations.SetBinding(EmailBox, TextBox.TextProperty, EmailBinding);

            getNewDetails.Children.Add(EmailBox);

            //מציג את ההודעה עם הבקשה לנתינת הפרטים

            var ask = new ContentDialog() { Title = "write the new user name", Content=getNewDetails, CloseButtonText = "ok" };

            await ask.ShowAsync();

            var id = await sub.createSubUserAsync(usr, new SubUserServ.User() { Email = UpperLogin.FullUser.Email, Password = UpperLogin.FullUser.Password });

            var localUsr = new SubUser() { BelongsToUpperUser = UpperLogin.FullUser.ID, Id = id, UserName=usr.UserName, Email=usr.Email, FName = usr.FName, LName = usr.LName, Password=usr.Password, Role= usr.Role };

            BinedUsersInTbl.Add(localUsr);

            UsersOriginal.Add(localUsr.Copy());

        }

        private void Refresh\_Click(object sender, RoutedEventArgs e)

        {

            loudTbl();

        }

        //מתבצע כאשר המשתמש מסנכרן את המשתמשים בטבלה עם שרות הרשת

        private async void Sync\_Click(object sender, RoutedEventArgs e)

        {

            var s = new SubUserServ.SubUsersServSoapClient();

            for(int i=0; i< BinedUsersInTbl.Count; i++)

            {

                if (!BinedUsersInTbl.ToList()[i].IsSame(UsersOriginal[i]))

                {

                    try

                    {

                        bool worked = await s.updateSubAsync(conv(BinedUsersInTbl[i]), new SubUserServ.User() { Password = UpperLogin.FullUser.Password, Email = UpperLogin.FullUser.Email, ID = UpperLogin.FullUser.ID });

                    }

                    catch(Exception exeption)

                    {

                        string msg = exeption.Message;

                        if (exeption.Message.Contains("the user"))

                        {

                            msg = msg.Remove(0, 121);

                            int count = 215;

                            msg = msg.Remove(msg.Length-count, count);

                        }

                        var TellAboutError = new ContentDialog() { Title = "the operation couldn't complete sucsussfully", Content = msg, CloseButtonText = "ok" };

                        await TellAboutError.ShowAsync();

                        loudTbl();

                    }

                 }

            }

        }

        //ממיר טיפוס משתמש תחתון מקומי לטיפוס של שירות רשת

        private SubUserServ.SubUser conv(SubUser a)

        {

            return new SubUserServ.SubUser() { BelongsToUpperUser = a.BelongsToUpperUser, Email = a.Email, FName = a.FName, Id = a.Id, LName = a.LName, Password = a.Password, Role = a.Role, UserName = a.UserName};

        }

        //מעלה אפשרויות טפריט לחיצה ימנית

        private List<SubUser> selectedUsers = new List<SubUser>();

        private void Grid\_RightTapped(object sender, RightTappedRoutedEventArgs e)

        {

            selectedUsers.Clear();

            foreach (ItemIndexRange range in SubUsersTbl.SelectedRanges.ToList())

            {

                for (int i = range.FirstIndex; i <= range.LastIndex; i++)

                {

                    selectedUsers.Add(SubUsersTbl.Items[i] as SubUser);

                }

            }

            if (selectedUsers.Count <= 0)

            {

                SubUser clickedItem = new SubUser();

                int Id = int.Parse(((sender as Grid).Children.Last() as TextBlock).Text);

                foreach (SubUser a in SubUsersTbl.Items)

                {

                    if (a.Id == Id)

                    {

                        clickedItem = a;

                    }

                }

                SubUsersTbl.SelectedItem = clickedItem;

            }

            MenuFlyout rightClick = new MenuFlyout();

            var icon = new SymbolIcon() { Symbol = Symbol.Delete };

            MenuFlyoutItem deleteOption = new MenuFlyoutItem { Icon = icon, Text = "delete", FontFamily = new FontFamily("Segoe MDL2 Assets") };

            deleteOption.Click += DeleteOption\_Click;

            rightClick.Items.Add(deleteOption);

            UIElement b = sender as UIElement;

            b.ContextFlyout = rightClick;

            Point point = new Point(e.GetPosition(b).X, e.GetPosition(b).Y);

            rightClick.ShowAt(b, point);

        }

        private async void DeleteOption\_Click(object sender, RoutedEventArgs e)

        {

            deleteSelected();

        }

        //מוחק את כל הפריטים ברשימת הנבחרים

        private async void deleteSelected()

        {

            Window.Current.CoreWindow.PointerCursor = new Windows.UI.Core.CoreCursor(Windows.UI.Core.CoreCursorType.Wait, 1);

            var UsrServ = new SubUserServ.SubUsersServSoapClient();

            foreach (SubUser br in selectedUsers)

            {

                var bs = conv(br);

                var isok = await UsrServ.DeleteSubUserAsync(new SubUserServ.User() { ID = UpperLogin.FullUser.ID, Email = UpperLogin.FullUser.Email, Password = UpperLogin.FullUser.Password }, bs.Id);

                if (isok)

                {

                    BinedUsersInTbl.Remove(br);

                }

                else

                {

                    var erro = new ContentDialog() { Title = "something went wrong", CloseButtonText = "ok" };

                    await erro.ShowAsync();

                }

            }

            Window.Current.CoreWindow.PointerCursor = new CoreCursor(CoreCursorType.Arrow, 1);

        }

        //לחיצה על כפטור המחיקה תעלה את זה

        //שונה מלחיצה על המחיקה בטפריט הימני

        private void Delete\_Click(object sender, RoutedEventArgs e)

        {

            selectedUsers.Clear();

            foreach (ItemIndexRange range in SubUsersTbl.SelectedRanges.ToList())

            {

                for (int i = range.FirstIndex; i <= range.LastIndex; i++)

                {

                    selectedUsers.Add(SubUsersTbl.Items[i] as SubUser);

                }

            }

            if (selectedUsers.Count <= 0)

            {

                SubUser clickedItem = new SubUser();

                int Id = int.Parse(((sender as Grid).Children.Last() as TextBlock).Text);

                foreach (SubUser a in SubUsersTbl.Items)

                {

                    if (a.Id == Id)

                    {

                        clickedItem = a;

                    }

                }

                SubUsersTbl.SelectedItem = clickedItem;

            }

            deleteSelected();

        }

    }

}

### עמוד ניהול רשימת מלאי

קוד ויזואלי:

<Page

    x:Class="storageUniversal.InventoryView"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:data="using:storageUniversal"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:Windows10FallCreatorsUpdate="http://schemas.microsoft.com/winfx/2006/xaml/presentation?IsApiContractPresent(Windows.Foundation.UniversalApiContract, 5)"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!--made by yehonatan vishna-->

    <!--a page where user can menege is current inventory-->

    <Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

        <Grid.ColumnDefinitions >

            <ColumnDefinition Width="100"></ColumnDefinition>

            <ColumnDefinition ></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

            <RowDefinition Height="20"></RowDefinition>

        </Grid.RowDefinitions>

        <Button Grid.Column="0" Grid.Row="0" Name="back" Click="Back\_Click" VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <Grid Grid.Column="1" Grid.Row="0" Name="headers" Width="auto">

            <Grid.ColumnDefinitions>

                <ColumnDefinition  Width="2\*"></ColumnDefinition>

                <ColumnDefinition MaxWidth="2000" Width="1\*"></ColumnDefinition>

                <ColumnDefinition MaxWidth="2000" Width="1\*"></ColumnDefinition>

                <ColumnDefinition Width="2\*"></ColumnDefinition>

                <ColumnDefinition MaxWidth="2000" Width="0.5\*"></ColumnDefinition>

            </Grid.ColumnDefinitions>

            <TextBlock Grid.Column="0" HorizontalTextAlignment="Center">name</TextBlock>

            <TextBlock Grid.Column="1" HorizontalTextAlignment="Center">Quntety</TextBlock>

            <TextBlock Grid.Column="2" HorizontalTextAlignment="Center">needed quntety</TextBlock>

            <TextBlock Grid.Column="3" HorizontalTextAlignment="Center">remarks</TextBlock>

            <TextBlock Grid.Column="4" HorizontalTextAlignment="Center">amount out</TextBlock>

        </Grid>

        <ListView Grid.Column="1" Grid.Row="1" Name="InventoryTbl" CanDragItems="True" RightTapped="Gridy\_RightTapped" HorizontalContentAlignment="Stretch" HorizontalAlignment="Stretch" VerticalAlignment="Top" >

            <ListView.ItemTemplate>

                <DataTemplate  x:Name="dataTemp" x:DataType="data:InventoryRow">

                    <Grid   Name="gridy" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

                        <Grid.ColumnDefinitions>

                            <ColumnDefinition MinWidth="50"  Width="2\*"></ColumnDefinition>

                            <ColumnDefinition MinWidth="50" MaxWidth="2000" Width="1\*"></ColumnDefinition>

                            <ColumnDefinition MinWidth="50" MaxWidth="2000" Width="1\*"></ColumnDefinition>

                            <ColumnDefinition MinWidth="50" Width="2\*"></ColumnDefinition>

                            <ColumnDefinition MinWidth="50" MaxWidth="2000" Width="0.5\*"></ColumnDefinition>

                        </Grid.ColumnDefinitions>

                        <Grid.RowDefinitions>

                            <RowDefinition></RowDefinition>

                        </Grid.RowDefinitions>

                        <TextBox Grid.Row="0" TextWrapping="Wrap" FlowDirection="LeftToRight" TextAlignment="Left" HorizontalAlignment="Stretch" Grid.Column="0" Name="NameBlock" Text= "{x:Bind Name, Mode=TwoWay}"/>

                        <TextBox Grid.Row="0" TextWrapping="Wrap" HorizontalAlignment="Stretch" Grid.Column="1" Name="QuantityBlock" Text= "{x:Bind Quantity, Mode=TwoWay}"/>

                        <TextBox Grid.Row="0" TextWrapping="Wrap" HorizontalAlignment="Stretch" Grid.Column="2" Name="NeededQuantityBlock" Text= "{x:Bind NeededQuantity, Mode=TwoWay}"/>

                        <Frame Grid.Row="0" Grid.Column="4" HorizontalAlignment="Stretch" BorderThickness="2" BorderBrush="Gray">

                            <TextBlock  TextWrapping="Wrap" HorizontalAlignment="Stretch" Name="AmountOutBox" Text="{x:Bind AmountOut, Mode=TwoWay}"/>

                        </Frame>

                        <TextBox  Grid.Row="0" TextWrapping="Wrap" HorizontalContentAlignment="Right"  Grid.Column="3" FlowDirection="RightToLeft" TextAlignment="Justify" Name="Remarkes" Text="{x:Bind Remarkes, Mode=TwoWay}"/>

                    </Grid>

                </DataTemplate>

            </ListView.ItemTemplate>

            <ListView.ItemContainerStyle>

                <Style TargetType="ListViewItem">

                    <Setter Property="HorizontalContentAlignment"  Value="Stretch"></Setter>

                    <Setter Property="HorizontalAlignment"  Value="Stretch"></Setter>

                </Style>

            </ListView.ItemContainerStyle>

        </ListView>

        <Grid Grid.Column="0" Grid.Row="1">

            <Grid.RowDefinitions>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

                <RowDefinition></RowDefinition>

            </Grid.RowDefinitions>

            <Button Grid.Row="0" Name="LoadTbl" Click="LoadTbl\_Click" HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

                <!--<TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="load table"></TextBlock>-->

                <SymbolIcon Symbol="Refresh"></SymbolIcon>

            </Button>

            <Button Grid.Row="1" Name="updateDataFromTbl" Click="UpdateDataFromTbl\_Click" HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

                <!--<TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="update Data"></TextBlock>-->

                <SymbolIcon Symbol="Save"></SymbolIcon>

            </Button>

            <Button Grid.Row="2" Name="addItem" Click="AddItem\_Click" HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

                <!--<TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="add item"></TextBlock>-->

                <SymbolIcon Symbol="Add"></SymbolIcon>

            </Button>

            <Button Grid.Row="3" Name="Delete" Click="Delete\_Click" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

                <!--<TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="delete selected item"></TextBlock>-->

                <SymbolIcon Symbol="Delete"></SymbolIcon>

            </Button>

            <Button Grid.Row="4" Name="CsvImport" Click="CsvImport\_Click"  HorizontalAlignment="Stretch" VerticalAlignment="Stretch" ToolTipService.ToolTip="import your existing inventory from csv file">

                <SymbolIcon Symbol="Import"></SymbolIcon>

                <!--<TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="import from csv"></TextBlock>-->

            </Button>

            <Button Grid.Row="5" Name="seeLandings" Click="SeeLandings\_Click"  HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

                <TextBlock HorizontalAlignment="Stretch" VerticalAlignment="Stretch" TextWrapping="WrapWholeWords" Text="see all active landings"></TextBlock>

            </Button>

        </Grid>

    </Grid>

</Page>

קוד אחורי:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using System.Data;

using System.Xml;

using storageUniversal;

using System.Text.RegularExpressions;

using System.Windows.Input;

using System.Collections.ObjectModel;

using Windows.UI.Core;

using Google.Apis.Services;

using Google.Apis.Discovery.v1;

using Google.Apis.Discovery.v1.Data;

using Google.Apis.Services;

using System.Threading.Tasks;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal

{

    /// <summary>

    /// a page where user can menege is current inventory

    /// </summary>

    public sealed partial class InventoryView : Page

    {

        public List<InventoryRow> InventoryBeforeChange;// original inventory

        public ObservableCollection<InventoryRow> InventoryRowesBindedToUser = new ObservableCollection<InventoryRow>();

        public static UserDBServ.User FullUser = UpperLogin.FullUser;//the user that activly uses the page;

        public static Type SentFrom;//the page from which the user were sent

        public static InventoryRow LantItem;

        public InventoryView()

        {

            this.InitializeComponent();

            LoadTblFunc();

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            // some code to handle mouse back + forward buttons

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //מבצע חיפוש בגוגל במנוע חיפוש מותאם אישית עם מגוון אתרי קמעונעות

        private async Task<Google.Apis.CustomSearchAPI.v1.Data.Search> googleApi(string qury)

        {

            string apiKey = "AIzaSyAs4wufxBxLid6quJCN2ynqihPieMxP3qY";

            string cx = "f427caefac6b24f26";

            var svc = new Google.Apis.CustomSearchAPI.v1.CustomSearchAPIService(new BaseClientService.Initializer { ApiKey = apiKey });

            var listRequest = svc.Cse.List();

            listRequest.Q = qury;

            listRequest.Gl = "il";

            listRequest.Cx = cx;

            var search = await listRequest.ExecuteAsync();

            return search;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //מרענן את הטבלה בלחיצה על הכפתור

        //loades the table at a click on the button

        private void LoadTbl\_Click(object sender, RoutedEventArgs e)

        {

            LoadTblFunc();

        }

        //פונקציה שמכניסה לטבלה את המידע העדכני המופיע במסד הנתונים

        //a function that inserts into the table the current data in the db

        private async void LoadTblFunc()

        {

            InventoryServ.InventoryFuncsSoapClient s = new InventoryServ.InventoryFuncsSoapClient();

            var broww = new BorowwDb.BorowwingsDBSoapClient();

            var hasCalcWorked = await broww.UpdateUserAmountOutAsync(FullUser.ID);

            var r = await s.GetInventoryUserDataTableAsync(FullUser.ID, FullUser.Email, FullUser.Password);

            List<InventoryRow> inventoryRows = new List<InventoryRow>();

            foreach (DataRow dr in r.Rows) {

                InventoryRow row = new InventoryRow();

                if (dr["Name"].ToString() != "")

                    row.Name = dr["Name"].ToString();

                row.ID = int.Parse(dr["ID"].ToString());

                if(dr["NeededQuantity"].ToString() != "")

                    row.NeededQuantity = float.Parse(dr["NeededQuantity"].ToString());

                row.OwnerUserId = int.Parse(dr["OwnerUserId"].ToString());

                if (dr["Quantity"].ToString() != "")

                    row.Quantity = float.Parse(dr["Quantity"].ToString());

                if (dr["Remarkes"].ToString() != "")

                    row.Remarkes = dr["Remarkes"].ToString();

                if (dr["AmountOut"].ToString() != "")

                    row.AmountOut = float.Parse(dr["AmountOut"].ToString());

                inventoryRows.Add(row);

            }

            InventoryRowesBindedToUser.Clear();

            foreach (InventoryRow Row in inventoryRows)

            {

                InventoryRowesBindedToUser.Add(Row);

            }

            InventoryTbl.ItemsSource = InventoryRowesBindedToUser;

            InventoryBeforeChange = new List<InventoryRow>();

            foreach (InventoryRow Row in inventoryRows)

            {

                InventoryBeforeChange.Add(Row.copy());

            }

        }

        //מפנה לפונקציה שמעדכנת את השינויים שנעשו בידי המשתמש במסד הנתונים. מופע בלחיצה על כפתור העידכון

        //updates the data in db according to the changes made by the user (directs to a diffrent function). Treiggered at the click on the update button.

        private void UpdateDataFromTbl\_Click(object sender, RoutedEventArgs e)

        {

            UpdateDataToDB();

        }

        // הפונקציה בודקת אלו שינויים נעשו בידי המשתמש בטבלה, ומעדכנת את שינויים אלו בטבלה

        //checks what have been changed in the table by the user and updates the changes to db.

        private async void UpdateDataToDB()

        {

            InventoryRow sel = (InventoryRow)InventoryTbl.SelectedItem;

            ItemCollection ic = InventoryTbl.Items;

            List<InventoryRow> inventoryRows = new List<InventoryRow>();

            int len = 0;

            foreach (InventoryRow a in ic)

            {

                len++;

                InventoryRow ro = new InventoryRow();

                ro.Name = a.Name;

                ro.Quantity = a.Quantity;

                ro.NeededQuantity = a.NeededQuantity;

                ro.ID = a.ID;

                ro.Remarkes = a.Remarkes;

                inventoryRows.Add(ro);

            }

            foreach (InventoryRow a in inventoryRows)

            {

                InventoryServ.InventoryFuncsSoapClient s = new InventoryServ.InventoryFuncsSoapClient();

                foreach (InventoryRow original in InventoryBeforeChange)

                {

                    if (a.ID == original.ID)

                    {

                        if (a.Equal(original))

                        {

                            break;

                        }

                        else

                        {

                            InventoryServ.InventoryRow row = new InventoryServ.InventoryRow();

                            row.ID = a.ID;

                            row.Name = a.Name;

                            row.Quantity = a.Quantity;

                            row.NeededQuantity = a.NeededQuantity;

                            row.Remarkes = a.Remarkes;

                            bool isOk = await s.changeInventoryRowAsync(row);

                        }

                    }

                }

            }

        }

        //מוחק את הרשומה המסומנת בלחיצה על כפתור המחיקה

        //deletes the selected item when delete button is clicked

        private async void Delete\_Click(object sender, RoutedEventArgs e)

        {

            InventoryServ.InventoryFuncsSoapClient s = new InventoryServ.InventoryFuncsSoapClient();

            int index = InventoryTbl.SelectedIndex;

            int id = ((InventoryRow)InventoryTbl.Items[index]).ID;

            var HadWorked = await s.DeleteInventoryRowAsync(id, FullUser.Email, FullUser.Password);

            InventoryRowesBindedToUser.Remove(InventoryRowesBindedToUser[InventoryTbl.SelectedIndex]);

        }

        //מגיב ללחיצה על כתור ההוספה בהוספת רשומה ריקה לטבלה ולמסד הנתונים

        //responds to click by adding item to ListView

        private async void AddItem\_Click(object sender, RoutedEventArgs e)

        {

            InventoryServ.InventoryFuncsSoapClient s = new InventoryServ.InventoryFuncsSoapClient();

            InventoryRow NewRow = new InventoryRow();

            var ItemId = await s.getNewItemIdAsync(FullUser.ID, FullUser.Email, FullUser.Password);

            NewRow.ID = int.Parse(ItemId.ToString());

            NewRow.OwnerUserId = FullUser.ID;

            InventoryRowesBindedToUser.Add(NewRow);

            InventoryBeforeChange.Add(NewRow.copy());

        }

        //הפונקציה מקבלת עצם, ומוסיפה אותו למסד הנתונים תחת המשתמש הנוכחי

        //rwcives a new InventoryRow and add new item to the ListView, and to db

        private async void addItemFunc(InventoryRow a)

        {

            InventoryServ.InventoryFuncsSoapClient s = new InventoryServ.InventoryFuncsSoapClient();

            InventoryRow NewRow = a;

            var ItemId = await s.getNewItemIdAsync(FullUser.ID, FullUser.Email, FullUser.Password);

            NewRow.ID = int.Parse(ItemId.ToString());

            NewRow.OwnerUserId = FullUser.ID;

            var IsOk = await s.changeInventoryRowAsync(new InventoryServ.InventoryRow() { OwnerUserId = NewRow.OwnerUserId, ID = NewRow.ID, AmountOut= NewRow.AmountOut, Name = NewRow.Name, NeededQuantity = NewRow.NeededQuantity, Quantity = NewRow.Quantity, Remarkes = NewRow.Remarkes });

            InventoryRowesBindedToUser.Add(NewRow);

        }

        //מייבא רשימת מלאי של פריטים מקובץ סי.אס.וי

        //import inventory list from csv file

        private async void CsvImport\_Click(object sender, RoutedEventArgs e)

        {

            try

            {

                var picker = new Windows.Storage.Pickers.FileOpenPicker();

                picker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;

                picker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;

                picker.FileTypeFilter.Add(".csv");

                Windows.Storage.StorageFile file = await picker.PickSingleFileAsync();

                var randomAccessStream = await file.OpenReadAsync();

                Stream stream = randomAccessStream.AsStreamForRead();

                StreamReader reader = new StreamReader(stream);

                List<InventoryRow> inventoryRows = new List<InventoryRow>();

                using (reader)

                {

                    string line;

                    while ((line = reader.ReadLine()) != null)

                    {

                        //define pattern

                        Regex csvparser = new Regex(",(?=(?:[^\"]\*\"[^\"]\*\")\*(?![^\"]\*\"))");

                        //separating columns to array

                        string[] x = csvparser.Split(line);

                        /\* do something with x \*/

                        InventoryRow row = new InventoryRow();

                        row.Name = x[0];

                        row.Quantity = float.Parse(x[1]);

                        row.NeededQuantity = float.Parse(x[2]);

                        row.Remarkes = x[3];

                        row.OwnerUserId = FullUser.ID;

                        inventoryRows.Add(row);

                    }

                }

                foreach (InventoryRow row in inventoryRows)

                {

                    addItemFunc(row);

                }

            }

            catch { }

        }

        // שולח את המשתמש לעמוד האחרון בו היה בלחיצה על כפתור החזרה

        //triggered at a click on the back button. Sendes the user back to the previus page.

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

        //מעלה תפריט פופאפ למילוא פרטי השאלה בידי המשתמש בלחיצה על כפתור הוספת ההשאלה בתפריט הקליק הימני של הפריט

        //reviles add landing menu after clicking add landing button

        private async void LandButton\_Click(object sender, RoutedEventArgs e)

        {

            Grid grid = new Grid();

            var rowDefinitions = grid.RowDefinitions;

            int i = 2;

            for(int j=0; j<i; j++)

            {

                rowDefinitions.Add(new RowDefinition());

            }

            TextBox amount = new TextBox { PlaceholderText = "amount" };

            Grid.SetRow(amount, 0); grid.Children.Add(amount);

            TextBox lentTo = new TextBox { PlaceholderText = "Lent to" };

            Grid.SetRow(lentTo, 1); grid.Children.Add(lentTo);

            ContentDialog getLandingDits = new ContentDialog()

            {

                Title = "please fill out the landing ditails",

                Content = grid,

                SecondaryButtonText = "save",

                SecondaryButtonCommandParameter= grid, SecondaryButtonCommand=new saveBtnCmd(),

                CloseButtonText = "cancel"

            };

            await getLandingDits.ShowAsync();

            LoadTblFunc();

        }

        //מעלה כפטור הוספת השאלה בחיצה על פריט בטבלה

        //reviles add landing button after right clicking an item

        private void Gridy\_RightTapped(object sender, RightTappedRoutedEventArgs e)

        {

            if (InventoryTbl.SelectedItem != null)

            {

                LantItem = InventoryTbl.Items[InventoryTbl.SelectedIndex] as InventoryRow;

                MenuFlyout rightClick = new MenuFlyout();

                MenuFlyoutItem firstItem = new MenuFlyoutItem { Text = "land out" };

                firstItem.Click += LandButton\_Click;

                rightClick.Items.Add(firstItem);

                var InventoryItemSelected = InventoryTbl.SelectedItem as InventoryRow;

                MenuFlyoutItem GoogleSearchItem = new MenuFlyoutItem { Text = "find where to buy more of this" };

                GoogleSearchItem.DataContext = InventoryItemSelected;

                GoogleSearchItem.Click += GoogleSearchItem\_Click ;

                rightClick.Items.Add(GoogleSearchItem);

                UIElement b = sender as UIElement;

                b.ContextFlyout = rightClick;

                Point point = new Point(e.GetPosition(b).X, e.GetPosition(b).Y);

                rightClick.ShowAt(b, point);

            }

        }

        //הפעולה שמתבצעת כאשר המשתמש לוחץ על האופציה של מציאת עוד מאותו מוצר

        //פעולה זו שולחת אותו ללינק הראשון ממגוון אתרים קמעוניים שיש להם עמוד לאותו מוצר

        //משתמש בשירות רשת של גוגל

        private async void GoogleSearchItem\_Click(object sender, RoutedEventArgs e)

        {

            var button = sender as MenuFlyoutItem;

            InventoryRow row = button.DataContext as InventoryRow;

            var s = (await googleApi(row.Name)).Items.First().Link;

            var success = await Windows.System.Launcher.LaunchUriAsync( new Uri(s ));

        }

        //הפקודה להוספת השאלה למסד הנתונים

        //the command for adding a landing

        class saveBtnCmd : ICommand

        {

            public event EventHandler CanExecuteChanged;

            public bool CanExecute(object parameter)

            {

                throw new NotImplementedException();

            }

            public async void Execute(object parameter)

            {

                UIElement content = parameter as UIElement;

                float amount = float.Parse(((content as Grid).Children[0] as TextBox).Text);

                string lentTo = ((content as Grid).Children[1] as TextBox).Text;

                var s = new BorowwDb.BorowwingsDBSoapClient();

                var id = await s.AddLendingAsync(LantItem.ID, lentTo, DateTime.Now, amount, FullUser.ID);

                var b = s.UpdateAmountOutAsync(LantItem.ID);

            }

        }

        //לוקח את המשתמש לעמוד בו הוא יכול לראות את כל ההשאלות הנוכחיות מן הרשימה שלו

        //takes the user to a page where he can see all of his active landings

        private void SeeLandings\_Click(object sender, RoutedEventArgs e)

        {

            BrowwingsAndDistractions.user = FullUser;

            BrowwingsAndDistractions.senderPage = typeof(InventoryView);

            Frame.Navigate(typeof(BrowwingsAndDistractions));

        }

    }

}

### עמוד ניהול השאלות- משתמש עליון

קוד ויזואלי

<Page

    x:Class="storageUniversal.BrowwingsAndDistractions"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:data="using:storageUniversal.codes"

    xmlns:local="using:storageUniversal"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:muxc="using:Microsoft.UI.Xaml.Controls"

    xmlns:Windows10FallCreatorsUpdate="http://schemas.microsoft.com/winfx/2006/xaml/presentation?IsApiContractPresent(Windows.Foundation.UniversalApiContract, 5)"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<!--a page for adding and removing borrowings-->

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Grid.Column="0" Grid.Row="0" Name="back" Click="Back\_Click" VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <StackPanel Orientation="Horizontal" Grid.Column="1" Grid.Row="0">

            <Button Name="refresh" Background="Black" BorderBrush="DarkGray" BorderThickness="0" Click="Refresh\_Click" Height="40" Width="40" ToolTipService.ToolTip="refresh the items in the table">

                <SymbolIcon Symbol="Refresh"/>

            </Button>

            <Button Name="syncButton" Background="Black" BorderBrush="DarkGray" BorderThickness="0" Click="SyncButton\_Click" ToolTipService.ToolTip="sync changes to database" Height="40" Width="40">

                <SymbolIcon Symbol="Sync"></SymbolIcon>

            </Button>

        </StackPanel>

        <Button Grid.Column="0" Grid.Row="1" VerticalAlignment="Bottom" ToolTipService.ToolTip="see all pendind order requsts" x:Name="pendingOrders" Click="PendingOrders\_Click">

            <SymbolIcon Symbol="MailReply"></SymbolIcon>

        </Button>

        <ListView Grid.Column="1" Grid.Row="1" Name="LandsTbl" HorizontalContentAlignment="Stretch" HorizontalAlignment="Stretch" IsMultiSelectCheckBoxEnabled="False" SelectionMode="Extended">

            <ListView.Header>

                <Grid HorizontalAlignment="Stretch">

                    <Grid.ColumnDefinitions>

                        <ColumnDefinition Width="\*"></ColumnDefinition>

                        <ColumnDefinition Width="\*"></ColumnDefinition>

                        <ColumnDefinition Width="\*"></ColumnDefinition>

                        <ColumnDefinition Width="0.5\*"></ColumnDefinition>

                    </Grid.ColumnDefinitions>

                    <TextBlock Grid.Column="0" HorizontalAlignment="Center">item name</TextBlock>

                    <TextBlock Grid.Column="1" HorizontalAlignment="Center">amount lent</TextBlock>

                    <TextBlock Grid.Column="2" HorizontalAlignment="Center">taken by</TextBlock>

                    <TextBlock Grid.Column="3" HorizontalAlignment="Center">borrowed in</TextBlock>

                </Grid>

            </ListView.Header>

            <ListView.ItemTemplate>

                <DataTemplate  x:Name="dataTemp" x:DataType="data:Borrow" >

                    <Grid HorizontalAlignment="Stretch" RightTapped="Grid\_RightTapped">

                        <Grid.ColumnDefinitions>

                            <ColumnDefinition Width="\*"></ColumnDefinition>

                            <ColumnDefinition Width="\*"></ColumnDefinition>

                            <ColumnDefinition Width="\*"></ColumnDefinition>

                            <ColumnDefinition Width="0.5\*"></ColumnDefinition>

                        </Grid.ColumnDefinitions>

                        <TextBlock HorizontalAlignment="Stretch" Grid.Column="0" Name="itemName" Text="{Binding Name, Mode=TwoWay}" TextAlignment="DetectFromContent"></TextBlock>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="1" Name="amount" Text="{Binding Quantity, Mode=TwoWay}"></TextBox>

                        <TextBox HorizontalAlignment="Stretch" Grid.Column="2" Name="takenBy" Text="{Binding BorrowedBy, Mode=TwoWay}"></TextBox>

                        <TextBlock HorizontalAlignment="Center" Name="WhenBorrowed" Grid.Column="3" Text="{Binding When}"></TextBlock>

                        <TextBlock Visibility="Collapsed" Name="ID" Text="{Binding BorrowingId}"></TextBlock>

                    </Grid>

                </DataTemplate>

            </ListView.ItemTemplate>

            <ListView.ItemContainerStyle>

                <Style TargetType="ListViewItem">

                    <Setter Property="HorizontalContentAlignment"  Value="Stretch"></Setter>

                    <Setter Property="HorizontalAlignment"  Value="Stretch"></Setter>

                </Style>

            </ListView.ItemContainerStyle>

        </ListView>

    </Grid>

</Page>

קוד אחורי

using System;

using System.Data;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using Windows.UI.Core;

using System.Collections.ObjectModel;

using storageUniversal.xamls.UpperUser;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal

{

    /// <summary>

    /// This page allowes the user to menege his active landings

    /// </summary>

    public sealed partial class BrowwingsAndDistractions : Page

    {

        public static UserDBServ.User user;

        public static Type senderPage;

        //a collection of all the borrows binded in the table

        //אוסף של כל העצמים המוצגים בטבלה

        public static ObservableCollection<codes.Borrow> borrows = new ObservableCollection<codes.Borrow>();

        private List<codes.Borrow> borroesUnTouched = new List<codes.Borrow>();

        public BrowwingsAndDistractions()

        {

            this.InitializeComponent();

            loadTbl();

            //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

            // some code to handle mouse back + forward buttons

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

        //מעלה את כל ההשאלות הפעילות של המשתמש ומכניס אותם לטבלה

        //loads all the user's active borrowings and inserts them into ListView

        private async void loadTbl()

        {

            var BDB = new BorowwDb.BorowwingsDBSoapClient();

            var DataTbl = await BDB.GetLandingsAsync(user.ID);

            borrows.Clear();

            foreach(DataRow dr in DataTbl.Rows)

            {

                var bro = new codes.Borrow();

                bro.ItemId = int.Parse(dr["ItemId"].ToString());

                bro.BorrowedBy = dr["BorrowedBy"].ToString();

                bro.When = DateTime.Parse(dr["When"].ToString());

                bro.Quantity = float.Parse(dr["Quantity"].ToString());

                bro.UserId = int.Parse(dr["UserId"].ToString());

                bro.BorrowingId = int.Parse(dr["BorrowingId"].ToString());

                await bro.SetName(bro.ItemId);

                borrows.Add(bro);

            }

            foreach(codes.Borrow a in borrows)

            {

                borroesUnTouched.Add(a.copy());

            }

            LandsTbl.ItemsSource = borrows;

        }

        //מחזיר את המשתמש לעמוד הקודם

        //takes you back to previos page

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

        private List<codes.Borrow> selectedItems = new List<codes.Borrow>();

        private void Grid\_RightTapped(object sender, RightTappedRoutedEventArgs e)

        {

            selectedItems.Clear();

            foreach (ItemIndexRange range in LandsTbl.SelectedRanges.ToList())

            {

                for(int i=range.FirstIndex; i<=range.LastIndex; i++)

                {

                    selectedItems.Add(LandsTbl.Items[i] as codes.Borrow);

                }

            }

            if (selectedItems.Count <= 0)

            {

                codes.Borrow clickedItem = new codes.Borrow();

                int borrowid = int.Parse(((sender as Grid).Children.Last() as TextBlock).Text);

                foreach (codes.Borrow a in LandsTbl.Items)

                {

                    if (a.BorrowingId == borrowid)

                    {

                        clickedItem = a;

                    }

                }

                LandsTbl.SelectedItem = clickedItem;

            }

                //LandsTbl.SelectRange(new ItemIndexRange(clickedIndex, 1));

                //clickedItem = LandsTbl.Items[LandsTbl.item] as InventoryRow;

                MenuFlyout rightClick = new MenuFlyout();

                var icon = new SymbolIcon() { Symbol = Symbol.Delete };

                MenuFlyoutItem deleteOption = new MenuFlyoutItem { Icon = icon, Text = "delete", FontFamily = new FontFamily("Segoe MDL2 Assets") };

                deleteOption.Click += DeleteOption\_Click;

                rightClick.Items.Add(deleteOption);

                UIElement b = sender as UIElement;

                b.ContextFlyout = rightClick;

                Point point = new Point(e.GetPosition(b).X, e.GetPosition(b).Y);

                rightClick.ShowAt(b, point);

        }

        //מתבצע כאשר המשתמש לוחץ על אפשרות המחיקה

        //מוחק את המשתמש מהטבלה וממסד הנתונים

        private async void DeleteOption\_Click(object sender, RoutedEventArgs e)

        {

            Window.Current.CoreWindow.PointerCursor = new Windows.UI.Core.CoreCursor(Windows.UI.Core.CoreCursorType.Wait, 1);

            var BorServ = new BorowwDb.BorowwingsDBSoapClient();

            foreach(codes.Borrow br in selectedItems)

            {

                var bs = new BorowwDb.Borrow() { BorrowingId = br.BorrowingId, BorrowedBy= br.BorrowedBy, ItemId = br.ItemId, Quantity = br.Quantity, UserId = br.UserId, When = br.When };

                var isok =await BorServ.DeleteLandingAsync(bs, user.Email, user.Password);

            }

            loadTbl();

            Window.Current.CoreWindow.PointerCursor = new CoreCursor(CoreCursorType.Arrow, 1);

        }

        //מרענן את הטבלה

        private void Refresh\_Click(object sender, RoutedEventArgs e)

        {

            loadTbl();

        }

        //מסנכרן את הטבלה עם מסד הנתונים

        private async void SyncButton\_Click(object sender, RoutedEventArgs e)

        {

            syncToDataBase();

        }

        //ממיר השאלה מקומית להשאלה מהטיפוס של שרות הרשת

        public BorowwDb.Borrow convert(codes.Borrow borrow)

        {

            return new BorowwDb.Borrow() { BorrowedBy = borrow.BorrowedBy, BorrowingId = borrow.BorrowingId, ItemId = borrow.ItemId, Quantity = borrow.Quantity, UserId = borrow.UserId, When = borrow.When};

        }

        //מסנכרן את הטבלה עם מסד הנתונים

        public async void syncToDataBase()

        {

            Window.Current.CoreWindow.PointerCursor = new Windows.UI.Core.CoreCursor(Windows.UI.Core.CoreCursorType.Wait, 1);

            var bdb = new BorowwDb.BorowwingsDBSoapClient();

            for(int i=0; i<borrows.Count; i++)

            {

                var old = borroesUnTouched[i];

                await bdb.updateBorrowAsync(convert(old), convert(LandsTbl.Items[i] as codes.Borrow), user.Email, user.Password);

            }

            Window.Current.CoreWindow.PointerCursor = new CoreCursor(CoreCursorType.Arrow, 1);

        }

        //לוקח את המשתמש לעמוד המכיל את כל ההשאלות הממתינות לו

        private void PendingOrders\_Click(object sender, RoutedEventArgs e)

        {

            Frame.Navigate(typeof(UpperSeeOrders));

        }

    }

}

### עמוד ניהול הזמנות

קוד ויזואלי

<Page

    x:Class="storageUniversal.xamls.UpperUser.UpperSeeOrders"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:local="using:storageUniversal.xamls.UpperUser"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:controls ="using:Microsoft.Toolkit.Uwp.UI.Controls"

    mc:Ignorable="d"

    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="40"></ColumnDefinition>

            <ColumnDefinition></ColumnDefinition>

        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>

            <RowDefinition Height="40"></RowDefinition>

            <RowDefinition></RowDefinition>

            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>

        <Button Grid.Row="0" Grid.Column="0" Name="back" Click="Back\_Click" VerticalAlignment="Top" HorizontalAlignment="Left" Style="{StaticResource NavigationBackButtonNormalStyle}"/>

        <controls:DataGrid Grid.Column="1" Grid.Row="1" x:Name="UnAprovedOrders"

    AutoGenerateColumns="False" SelectionMode="Single" IsReadOnly="False" GridLinesVisibility="All" AlternatingRowBackground="Gray"

    ItemsSource="{x:Bind local:UpperSeeOrders.UnAprovedBindedOrders}" >

            <controls:DataGrid.Columns>

                <controls:DataGridTextColumn Header="item name" Binding="{Binding ItemName}" IsReadOnly="True"/>

                <controls:DataGridTextColumn Header="Amount" Binding="{Binding Amount, Mode=TwoWay}"/>

                <controls:DataGridTextColumn Header="Remarkes" Binding="{Binding Remarkes, Mode=TwoWay}"/>

                <controls:DataGridTextColumn Header="date" Binding="{Binding OrderDate}" IsReadOnly="True"/>

                <controls:DataGridTemplateColumn >

                    <controls:DataGridTemplateColumn.CellTemplate>

                        <DataTemplate>

                            <Button x:Name="AcceptOrder" Click="AcceptOrder\_Click">

                                <TextBlock>Accept order</TextBlock>

                            </Button>

                        </DataTemplate>

                    </controls:DataGridTemplateColumn.CellTemplate>

                </controls:DataGridTemplateColumn>

                <controls:DataGridTemplateColumn >

                    <controls:DataGridTemplateColumn.CellTemplate>

                        <DataTemplate>

                            <Button x:Name="RejectOrder" Click="RejectOrder\_Click" ToolTipService.ToolTip="reject order">

                                <SymbolIcon Symbol="Delete"></SymbolIcon>

                            </Button>

                        </DataTemplate>

                    </controls:DataGridTemplateColumn.CellTemplate>

                </controls:DataGridTemplateColumn>

            </controls:DataGrid.Columns>

        </controls:DataGrid>

        <StackPanel Orientation="Vertical" Grid.Column="1" Grid.Row="2" >

            <TextBlock TextDecorations="Underline">Aproved Orders:</TextBlock>

            <controls:DataGrid x:Name="AprovedOrders"

        AutoGenerateColumns="False" SelectionMode="Single" IsReadOnly="False" GridLinesVisibility="All" AlternatingRowBackground="Gray"

        ItemsSource="{x:Bind local:UpperSeeOrders.AprovedBindedOrders}" >

                <controls:DataGrid.Columns>

                    <controls:DataGridTextColumn Header="item name" Binding="{Binding ItemName}" IsReadOnly="True"/>

                    <controls:DataGridTextColumn Header="Amount" Binding="{Binding Amount, Mode=TwoWay}"/>

                    <controls:DataGridTextColumn Header="Remarkes" Binding="{Binding Remarkes, Mode=TwoWay}"/>

                    <controls:DataGridTextColumn Header="date" Binding="{Binding OrderDate}" IsReadOnly="True"/>

                    <!--<controls:DataGridTemplateColumn >

                        <controls:DataGridTemplateColumn.CellTemplate>

                            <DataTemplate>

                                <Button x:Name="AcceptOrder" Click="AcceptOrder\_Click">

                                    <TextBlock>Accept order</TextBlock>

                                </Button>

                            </DataTemplate>

                        </controls:DataGridTemplateColumn.CellTemplate>

                    </controls:DataGridTemplateColumn>

                    <controls:DataGridTemplateColumn >

                        <controls:DataGridTemplateColumn.CellTemplate>

                            <DataTemplate>

                                <Button x:Name="RejectOrder" Click="RejectOrder\_Click" ToolTipService.ToolTip="reject order">

                                    <SymbolIcon Symbol="Delete"></SymbolIcon>

                                </Button>

                            </DataTemplate>

                        </controls:DataGridTemplateColumn.CellTemplate>

                    </controls:DataGridTemplateColumn>-->

                </controls:DataGrid.Columns>

            </controls:DataGrid>

            <Button x:Name="addAllAprovedToBorrowings" Click="AddAllAprovedToBorrowings\_Click" ToolTipService.ToolTip="add all the aproved orders to active borowwings">

                <SymbolIcon Symbol="Send"></SymbolIcon>

            </Button>

        </StackPanel>

    </Grid>

</Page>

קוד אחורי

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

using System.Collections.ObjectModel;

using System.Data;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace storageUniversal.xamls.UpperUser

{

    /// <summary>

    /// An empty page that can be used on its own or navigated to within a Frame.

    /// </summary>

    public sealed partial class UpperSeeOrders : Page

    {

        //רשימות של ההזמנות שאושרו ושלא אושרו

        public static ObservableCollection<SubUserServ.Order> UnAprovedBindedOrders = new ObservableCollection<SubUserServ.Order>();

        public static ObservableCollection<SubUserServ.Order> AprovedBindedOrders = new ObservableCollection<SubUserServ.Order>();

        private List<SubUserServ.Order> orders = new List<SubUserServ.Order>();

        public UpperSeeOrders()

        {

            this.InitializeComponent();

            LoadTables();

        }

        //ממיר את המשתמש מעמוד הכניסה למשתמש שמתאים לשירות הרשת של התת משתמשים

        public SubUserServ.User FullUpperUserFromSubServ

        {

            get

            {

                return new SubUserServ.User() { ID = UpperLogin.FullUser.ID, BDate = UpperLogin.FullUser.BDate, Compeny = UpperLogin.FullUser.Compeny, Email = UpperLogin.FullUser.Email, Fname = UpperLogin.FullUser.Fname, Lname = UpperLogin.FullUser.Lname, Password = UpperLogin.FullUser.Password };

            }

        }

        //מעלה את הטבלאות לפי המידע משירות הרשת

        public async void LoadTables()

        {

            var serv = new UserDBServ.UserDBServSoapClient();

            var tbldata = await serv.getUpperOrdersAsync(UpperLogin.FullUser);

            orders.Clear();

            foreach (DataRow row in tbldata.Rows)

            {

                var order = new SubUserServ.Order();

                order.ItemName = row["ItemName"].ToString();

                order.Amount = float.Parse(row["Amount"].ToString());

                order.ID = int.Parse(row["ID"].ToString());

                order.ItemId = int.Parse(row["ItemId"].ToString());

                order.BySubUser = int.Parse(row["BySubUser"].ToString());

                order.ToUpperUser = int.Parse(row["ToUpperUser"].ToString());

                order.Aproved = bool.Parse(row["Aproved"].ToString());

                order.Rejected = bool.Parse(row["Rejected"].ToString());

                order.Remarkes = row["Remarks"].ToString();

                if (DateTime.TryParse(row["OrderDate"].ToString(), out DateTime OrderDate))

                {

                    order.OrderDate = OrderDate;

                }

                orders.Add(order);

            }

            AprovedBindedOrders.Clear();

            UnAprovedBindedOrders.Clear();

            foreach(var order in orders)

            {

                if (!order.Rejected) {

                    if (order.Aproved)

                    {

                        AprovedBindedOrders.Add(order);

                    }

                    else

                    {

                        UnAprovedBindedOrders.Add(order);

                    }

                }

            }

        }

        //send user back to previus page

        //כפתור אחורה- שולח את המשתמש לעמוד הקודם

        private void Back\_Click(object sender, RoutedEventArgs e)

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

        //לחיצה מאשרת את ההזמנה

        private async void AcceptOrder\_Click(object sender, RoutedEventArgs e)

        {

            Button btn = sender as Button;

            var order = btn.DataContext as SubUserServ.Order;

            order.Aproved = true;

            var serv = new SubUserServ.SubUsersServSoapClient();

            bool a = await serv.UpdateOrderByUpperUserAsync(FullUpperUserFromSubServ, order);

            UnAprovedBindedOrders.Remove(order);

            AprovedBindedOrders.Add(order);

        }

        //לחיצה דוחה את ההזמנה

        private async void RejectOrder\_Click(object sender, RoutedEventArgs e)

        {

            Button btn = sender as Button;

            var order = btn.DataContext as SubUserServ.Order;

            order.Rejected = true;

            var serv = new SubUserServ.SubUsersServSoapClient();

            bool a = await serv.UpdateOrderByUpperUserAsync(FullUpperUserFromSubServ, order);

            UnAprovedBindedOrders.Remove(order);

        }

        //פעולה שמוסיפה את ההשאלות שאאושרו לרשימת ההשאלות

        private async void AddAllAprovedToBorrowings\_Click(object sender, RoutedEventArgs e)

        {

            var borowwServ = new BorowwDb.BorowwingsDBSoapClient();

            var SubServ = new SubUserServ.SubUsersServSoapClient();

            foreach(var order in AprovedBindedOrders)

            {

                var borrower = await SubServ.GetYourSubUserAsync(FullUpperUserFromSubServ, order.BySubUser);

                int borrowing\_ID = await borowwServ.AddLendingAsync(order.ItemId,borrower.FName + " " + borrower.LName, DateTime.Now, order.Amount, UpperLogin.FullUser.ID);

                borowwServ.UpdateAmountOutAsync(order.ItemId);

            }

            AprovedBindedOrders.Clear();

        }

    }

}

### קוד למחלקה User

using System;

namespace storageUniversal

{

    public class User

    {

        //תכונות בסיסיות של המשתמש

        //defining basic properties

        private int id;

        private string fname;

        private string lname;

        private DateTime bdate;

        private string compeny;

        private string email;

        private string password;

        //בנאי ריק

        // empty constractor

        public User()

        {

        }

        //גטרים וסטרים לכל התכונות

        //getters + setters for properties

        public int ID

        {

            get

            {

                return this.id;

            }

            set

            {

                this.id = value;

            }

        }

        public string Fname

        {

            get { return this.fname; }

            set { this.fname = value; }

        }

        public string Lname

        {

            get { return this.lname; }

            set { this.lname = value; }

        }

        public DateTime BDate

        {

            get { return this.bdate; }

            set { this.bdate = value; }

        }

        //על מנת להציג את תכונת תאריך הלידה בפקד סטנדרטי יש צורך להוציא את המידע בצורת סוג דייטטיימאופסט עם יכולת להיות תכונה ריקה

        // מטרת תכונה זו היא להמיר את תאריך הלידה לסוג זה ולהיתממשק עם פקד בחירת התאריך

        //In order to display BDate in standard CalendarDatePicker I had to write a property that would return BDate as a nullable DateTimeOffset

        public DateTimeOffset? BDateTimeOffset

        {

            get { try { return DateTimeOffset.Parse(this.BDate.ToString()); }

                catch { return null; }

                }

            set { this.bdate = DateTime.Parse(value.ToString()); }

        }

        public string Compeny

        {

            get { return this.compeny; }

            set { this.compeny = value; }

        }

        public string Email

        {

            get { return this.email; }

            set { this.email = value; }

        }

        public string Password

        {

            get { return this.password; }

            set { this.password = value; }

        }

        //פעולה שיוצרת עצם חדש עם אותם תכונות כמו עצם זה

        // a fution that creatates another User obj identical to this one

        public User copy() {

            User a = new User();

            a.ID = this.id;

            a.Fname = this.Fname;

            a.Lname = this.Lname;

            a.BDate = this.BDate;

            a.compeny = this.compeny;

            a.email = this.email;

            a.password = this.password;

            return a;

        }

        //תכונה הבודקת האם משתמש אחר זהה בכל פרטיו למשתמש זה

        // a function that returns wether another User object is identical to this one

        public bool IsSame(User user)

        {

            try

            {

                bool idS = this.ID == user.ID;

                bool FnameS = this.Fname.Equals(user.Fname);

                bool LnameS = this.Lname.Equals(user.Lname);

                bool DateS = this.BDate.Equals(user.BDate);

                bool CompS = this.compeny.Equals(user.Compeny);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                return idS && FnameS && LnameS && DateS && CompS && EmailS && PassS;

            }

            catch {

                return false;

            }

            }

        //made by yehonatan vishna

    }

}

### קוד למחלקת InventoryRow

using System;

using System.Collections.Generic;

using System.Linq;

namespace storageUniversal

{

    public class InventoryRow

    {

        //מגדיר את התכונות של העצם

        //defines some properies

        private int id;

        private string name;

        private float quantity;

        private float neededQuantity;

        private int ownerUserId;

        private float amountOut;

        private string remarks;

        //פעולות בונות

        //constractores

        public InventoryRow() { }

        public InventoryRow(int id, string name, int q, int nq, int oui)

        {

            this.id = id;

            this.name = name;

            this.quantity = q;

            this.neededQuantity = nq;

            this.ownerUserId = oui;

        }

        //גטרים וסטרים לכל התכונות

        //getters and setters for all the properties listed above

        public int ID

        {

            get { return this.id; }

            set { this.id = value; }

        }

        public string Name

        {

            get { return this.name; }

            set { this.name = value; }

        }

        public float Quantity

        {

            get { return this.quantity; }

            set { this.quantity = value; }

        }

        public float NeededQuantity

        {

            get { return this.neededQuantity; }

            set { this.neededQuantity = value; }

        }

        public int OwnerUserId

        {

            get { return this.ownerUserId; }

            set { this.ownerUserId = value; }

        }

        public float AmountOut

        {

            get { return this.amountOut; }

            set { this.amountOut = value; }

        }

        public String Remarkes

        {

            get { return this.remarks; }

            set { this.remarks = value; }

        }

        //פונקציה שיוצרת עצם שורת מלאי זהה עם אותם תכונות

        // a fution that creatates another InventoryRow obj identical to this one

        public InventoryRow copy()

        {

            InventoryRow a = new InventoryRow();

            a.ID = this.id;

            a.Quantity = this.quantity;

            a.OwnerUserId = this.ownerUserId;

            a.NeededQuantity = this.neededQuantity;

            a.Name = this.name;

            a.Remarkes = this.remarks;

            return a;

        }

        //פונקציה שבודקת האם עצם רשימת מלאי אחר זהה לעצם זה בכל תכונותיו

        // a function that returns wether another InventoryRow object is identical to this one

        public bool Equal(InventoryRow row)

        {

            bool IdSame = this.id == row.ID;

            bool QSame = this.quantity == row.Quantity;

            bool OUIdSame = this.ownerUserId == row.OwnerUserId;

            bool NQSame = this.neededQuantity == row.NeededQuantity;

            bool NameSame;

            if ((row.Name == null && this.name != null) || (row.Name != null && this.name == null))

            {

                NameSame = false;

            }

            else

            {

                if ((row.Name == null && this.name == null))

                {

                    NameSame = true;

                }

                else

                {

                    NameSame = this.name.Equals(row.Name.ToString());

                }

            }

            bool RSame;

            if ((row.Remarkes == null && this.remarks != null) || (row.Remarkes != null && this.remarks == null))

            {

                RSame = false;

            }

            else

            {

                if ((row.Remarkes == null && this.remarks == null))

                {

                    RSame = true;

                }

                else

                {

                    RSame = this.remarks.Equals(row.Remarkes.ToString());

                }

            }

            return IdSame && QSame && OUIdSame && NQSame && NameSame && RSame;

        }

    }

}

### קוד למחלקת SubUser

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace storageUniversal

{

    public class SubUser

    {

        //מגדיר תכונות בסיסיות

        //defining reqired properties

        private int id;

        private int belongsToUpperUser;

        private string fName;

        private string lName;

        private string role;

        private string email;

        private string password;

        private string userName;

        public SubUser()

        {

        }

        public int Id { get => id; set => id = value; }

        public int BelongsToUpperUser { get => belongsToUpperUser; set => belongsToUpperUser = value; }

        public string FName { get => fName; set => fName = value; }

        public string LName { get => lName; set => lName = value; }

        public string Role { get => role; set => role = value; }

        public string Email { get => email; set => email = value; }

        public string Password { get => password; set => password = value; }

        public string UserName { get => userName; set => userName = value; }

        public SubUser Copy()

        {

            return new SubUser() { Id = this.id, BelongsToUpperUser = this.BelongsToUpperUser, FName = this.FName, LName = this.LName, Role = this.Role, UserName = this.userName };

        }

        public bool IsSame(SubUser user)

        {

            try

            {

                bool idS = this.Id == user.Id;

                bool FnameS = this.FName.Equals(user.FName);

                bool LnameS = this.LName.Equals(user.LName);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                bool belS = this.BelongsToUpperUser.Equals(user.BelongsToUpperUser);

                bool roleS = this.Role.Equals(user.Role);

                bool Usr = this.UserName.Equals(user.UserName);

                return idS && FnameS && LnameS && EmailS && PassS && belS && roleS && Usr;

            }

            catch

            {

                return false;

            }

        }

    }

}

### קוד למחלקת UsersDatabase

הקוד הזה בשימוש ליצירת מסד נתונים מקומי ב sqllite

using Microsoft.Data.Sqlite;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Threading.Tasks;

namespace storageUniversal

{

    public class UsersDatabase

    {

        SqliteConnection Database;

        public UsersDatabase()

        {

        }

        //מוודא שמסד הנתונים מאותחל אם לא מאתחל אותו

        //makes shure the db is initilized

        async Task Init()

        {

            // Return if the database has already been initialized

            if (Database != null)

                return;

            // Create a new Sqlite database connection using the specified database file path

            Database = new SqliteConnection($"Filename={Constants.DatabasePath}");

            // Open the database connection

            Database.Open();

            // Create the table command string with the specified columns and data types

            //שאילתה שיוצרת את מסד הנתונים אם הוא לא כבר קיים

            String tableCommand = "CREATE TABLE IF NOT " +

        "EXISTS Users ( [ID][int] IDENTITY NOT NULL," +

        "[FName] [nchar] (20) NULL," +

        "[LName] [nchar] (20) NULL," +

        "[BDate] [datetime] NULL," +

        "[compeny] [nchar] (20) NULL," +

        "[email] [nchar] (30) NULL," +

        "[password] [nchar] (20) NULL)";

            // Create a new Sqlite command using the table command string and the database connection

            SqliteCommand createTable = new SqliteCommand(tableCommand, Database);

            // Execute the command to create the table

            createTable.ExecuteReader();

        }

        //מחזיר רשימה של כל המשתמשים במסד הנתונים המקומי

        //returns a User list of all users in db

        public async Task<List<User>> GetItemsAsync()

        {

            // Initialize the database

            await Init();

            // Create a query string to select all rows from the Users table

            var qury = "select \* from Users";

            // Create a new Sqlite command using the query string and the database connection

            var comd = new Microsoft.Data.Sqlite.SqliteCommand(qury, Database);

            // Execute the command to retrieve the rows from the table

            var reader = comd.ExecuteReader();

            // Create a new DataTable to hold the rows

            var tbl = new DataTable();

            // Load the rows into the DataTable

            tbl.Load(reader);

            // Create a list to hold the user objects

            var users = new List<User>();

            // Loop through each row in the DataTable

            foreach (DataRow dr in tbl.Rows)

            {

                // Create a new User object and populate its properties with the data from the DataRow

                var user = new User();

                user.ID = int.Parse(dr["ID"].ToString());

                user.Password = dr["password"].ToString();

                user.Email = dr["email"].ToString();

                user.Fname = dr["FName"].ToString();

                user.Lname = dr["LName"].ToString();

                user.Compeny = dr["compeny"].ToString();

                // Add the user to the list of users

                users.Add(user);

            }

            // Return the list of users

            return users;

        }

        //מקבל עצם של משתמש ומכניס את המידע המוכל בעצם למסד הנתונים

        //takes a user object and inserts it into the db

        public async Task<bool> InsertItemAsync(User item)

        {

            // Initialize the database

            await Init();

            // Create a string containing the query to insert a new row into the Users table with the data from the provided User object

            string nonQury = "Insert into Users (ID, email, password, FName, LName, BDate, compeny) values (" + item.ID.ToString() + ",'" + item.Email + "', '" + item.Password + "','" + item.Fname.ToString() + "', '" + item.Lname + "', " + "CAST('" + item.BDate.ToShortDateString() + "' AS DateTime)" + ", '" + item.Compeny + "');";

            // Create a new Sqlite command using the query string and the database connection

            var comd = new SqliteCommand(nonQury, Database);

            // Execute the command to insert the new row

            var a = comd.ExecuteNonQuery();

            // Dispose of the command object

            comd.Dispose();

            // Return a value indicating whether the insertion was successful

            return a > 0;

        }

        //מוחק את כל המשתמשים במסד הנתונים

        //deletes all users in db

        public async Task<bool> DeleteAll()

        {

            // Initialize the database

            await Init();

            // Create a string containing the query to delete all rows from the Users table

            string nonQury = "Delete from Users";

            // Create a new Sqlite command using the query string and the database connection

            var comd = new SqliteCommand(nonQury, Database);

            // Execute the command to delete the rows

            var a = comd.ExecuteNonQuery();

            // Dispose of the command object

            comd.Dispose();

            // Return a value indicating whether the deletion was successful

            return a > 0;

        }

    }

}

### הקוד למחלקת backMouse

הקוד בשימוש להוספה מהירה של מעבר עמוד אחורה עם כפתורי העכבר

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Windows.UI.Core;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

namespace storageUniversal.codes

{

    public class backMouse

    {

        public backMouse()

        {

            Window.Current.Activate();

            Window.Current.CoreWindow.PointerPressed += CoreWindow\_PointerPressed;

        }

        //קוד לניווט אחורה וקדימה עם כפטורי הניווט של העכבר

        // some code to handle mouse back + forward buttons

        private void CoreWindow\_PointerPressed(CoreWindow sender, PointerEventArgs args)

        {

            if (args.CurrentPoint.Properties.IsXButton1Pressed)

            {

                Frame frame = Window.Current.Content as Frame;

                if (frame.CanGoBack)

                {

                    frame.GoBack();

                }

            }

            else

            {

                if (args.CurrentPoint.Properties.IsXButton2Pressed)

                {

                    Frame frame = Window.Current.Content as Frame;

                    if (frame.CanGoForward)

                    {

                        frame.GoForward();

                    }

                }

            }

        }

    }

}

## הקוד לצד שרת:

הקוד למחלקות צד שרת ולשירותי הרשת כבר צורף בחלק המתאים בעבודה.