

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

[תוכן עניינים 1](#_Toc127966034)

[מבוא 2](#_Toc127966035)

[הרקע לפרויקט 2](#_Toc127966036)

[מטרות המערכת 2](#_Toc127966037)

[מטרות על: 2](#_Toc127966038)

[מטרות נלוות: 2](#_Toc127966039)

[תיאור המערכת: 2](#_Toc127966040)

[גבולות המערכת 3](#_Toc127966041)

[עבור תת משתמש 3](#_Toc127966042)

[עבור משתמש עליון: 3](#_Toc127966043)

[עבור מנהל: 3](#_Toc127966044)

[סביבת פיתוח 3](#_Toc127966045)

[שפות תכנות: 3](#_Toc127966046)

[שכבות: 3](#_Toc127966047)

[פלטפורמות של הלקוחות: 4](#_Toc127966048)

[אתגרים מרכזיים: 4](#_Toc127966049)

[חידושים התאמות ועיצוב: 4](#_Toc127966050)

[ניתוח המערכת 5](#_Toc127966051)

[מצב קיים: 5](#_Toc127966052)

[מטרות מערכת עתידיות 5](#_Toc127966053)

[מצב מערכת עתידית 5](#_Toc127966054)

[תרשים ERD 5](#_Toc127966055)

[עץ תהליכים 6](#_Toc127966056)

[בסיס הנתונים (database) 7](#_Toc127966057)

[תרשים DSD 7](#_Toc127966058)

[טבלאות ראשיות 7](#_Toc127966059)

[טבלת Users: 7](#_Toc127966060)

[טבלת Inventory 7](#_Toc127966061)

[טבלת BorrowedItems 8](#_Toc127966062)

[טבלת SubUsers 8](#_Toc127966063)

[טבלת Orders 8](#_Toc127966064)

[צד שרת 10](#_Toc127966065)

# 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.

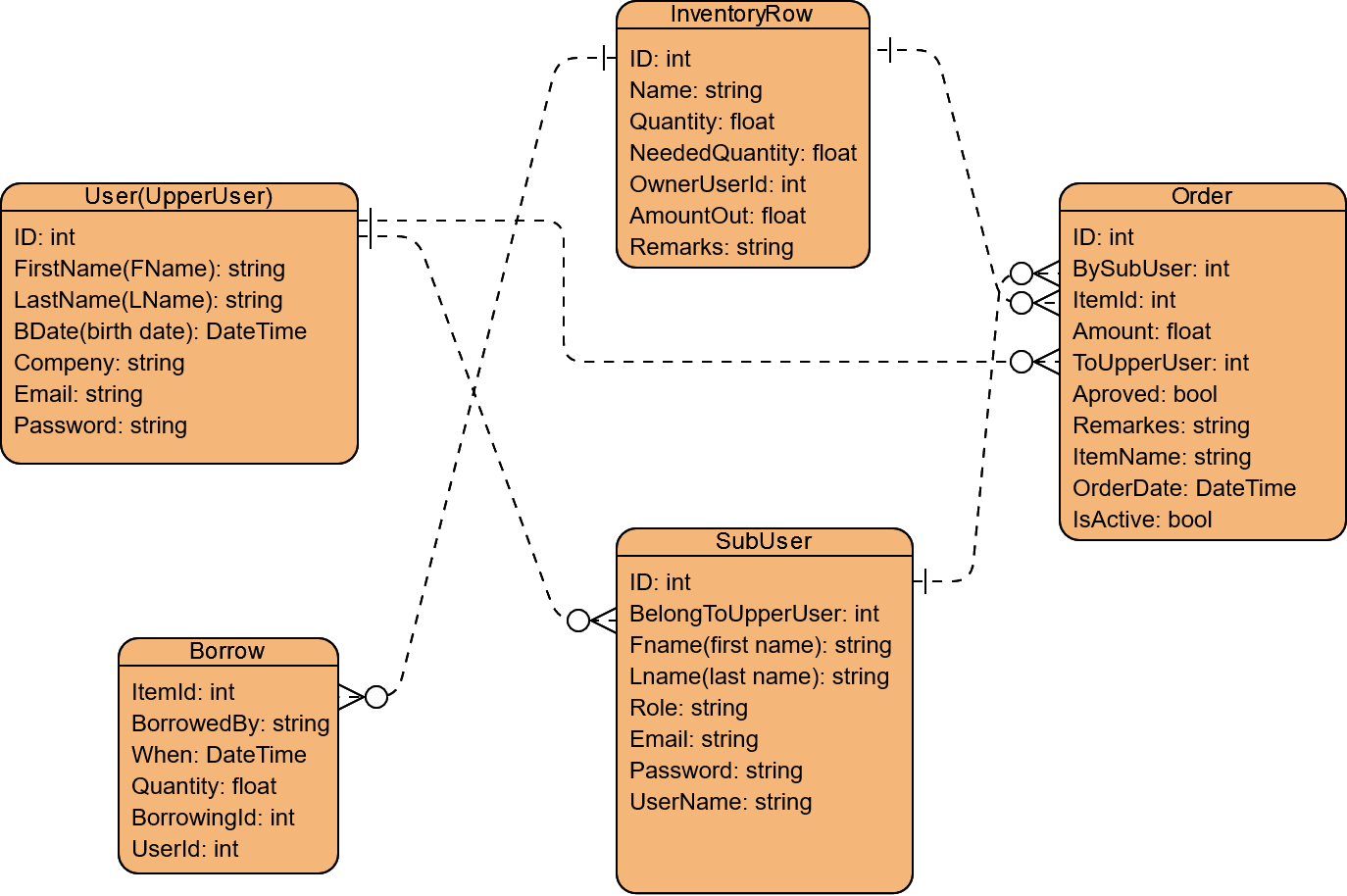
## מטרות מערכת עתידיות

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.

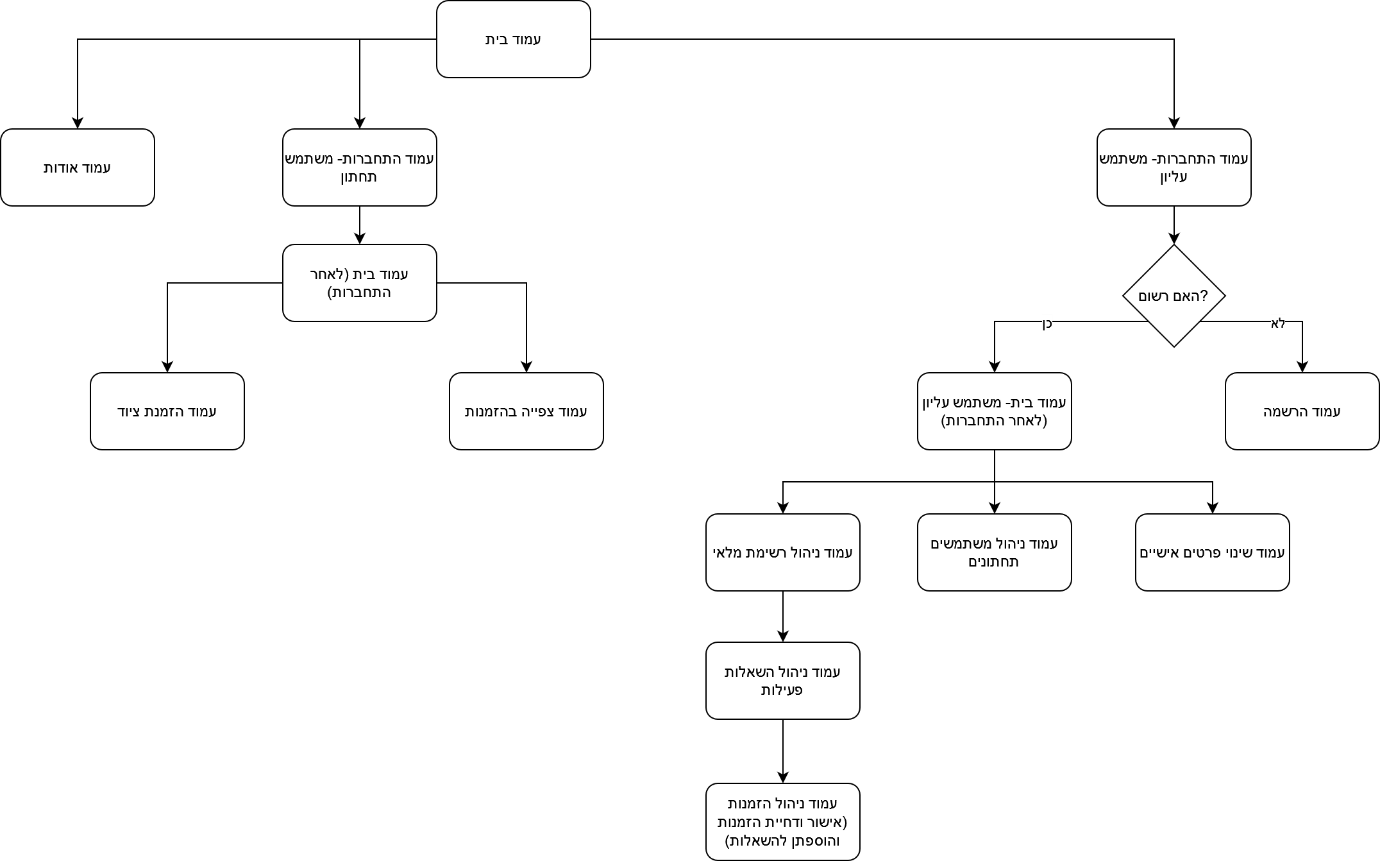
## מצב מערכת עתידית

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

## תרשים ERD

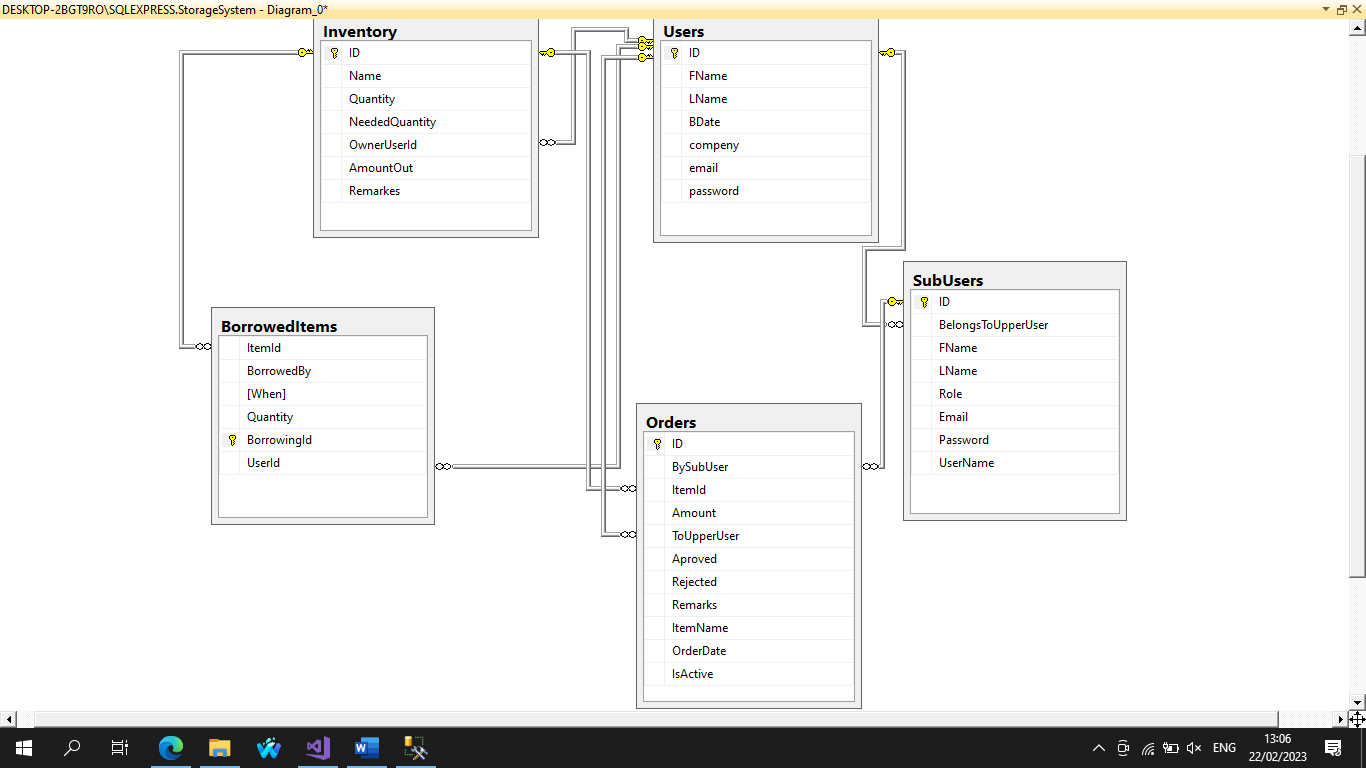


## עץ תהליכים



# 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

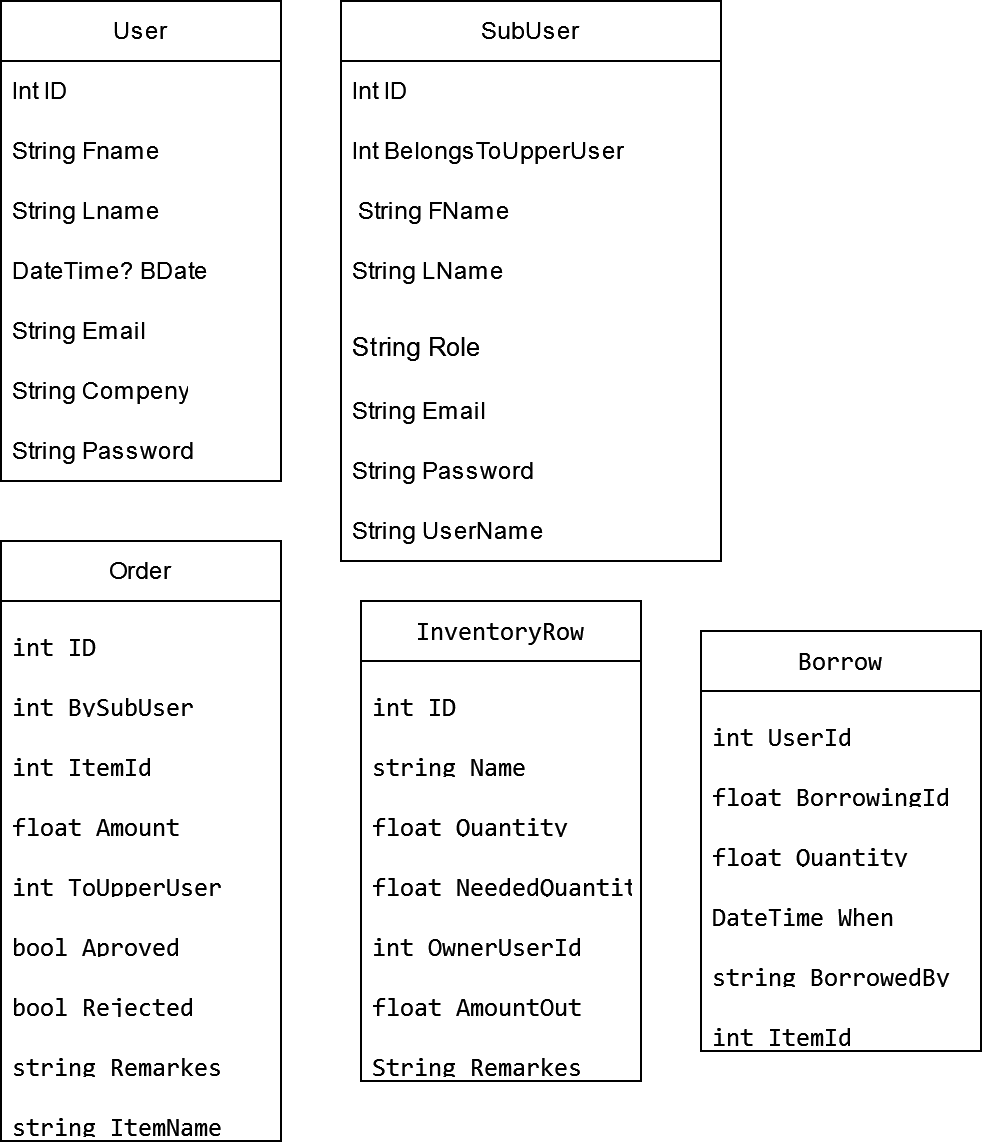
כל רשומה בטבלה זו מייצגת הזמנת ציוד של משתמש תחתון אל משתמש עליון (מחבר בארגון אל מנהל המלאי).

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

## תרשים UML



## 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;

            }

        }

    }

}

## 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();

            }

        }

    }

}