תיק פרויקט- תוכנה לניהול מחסן, מלאי והשאלות

דרכא רמון על שם אילן ואסף רמון |

נכתב ב UWP

יהונתן וישנה

שנה"ל: תשפ"ג- 2023

תוכן

[צד לקוח 2](#_Toc122786799)

[עמודים באפליקציה 2](#_Toc122786800)

[עמוד ראשי 2](#_Toc122786801)

[עמוד אודות 5](#_Toc122786802)

[עמוד הרשמה 8](#_Toc122786803)

[עמוד כניסה 12](#_Toc122786804)

[עמוד עריכת מלאי 18](#_Toc122786805)

[עמוד שינוי פרטים אישיים 26](#_Toc122786806)

[עמוד לניהול השאלות פעילות 29](#_Toc122786807)

[עמוד מנהל 32](#_Toc122786808)

[טיפוסים וקבצי קוד 37](#_Toc122786809)

[טיפוס Borrow 37](#_Toc122786810)

[קוד לניהול מסד נתונים SQLite (Constants, UserDataBase) 39](#_Toc122786811)

[עצם InventoryRow 43](#_Toc122786812)

[טיפוס User 46](#_Toc122786813)

[צד שרת 49](#_Toc122786814)

[שירות רשת לניהול משתמשים 49](#_Toc122786815)

[טיפוס Connection 50](#_Toc122786816)

[קובץ שירות הרשת עצמו- UserDBServ.asmx 53](#_Toc122786817)

[שירותי רשת לניהול מלאי והשאלות 59](#_Toc122786818)

[שירות הרשת- InventoryFuncs.asmx 60](#_Toc122786819)

[שירות רשת- BorowwingsDB.asmx 63](#_Toc122786820)

[טבלאות SQL 66](#_Toc122786821)

[טבלת Users 66](#_Toc122786822)

[טבלת Inventory 67](#_Toc122786823)

[טבלת BorrowedItems 67](#_Toc122786824)

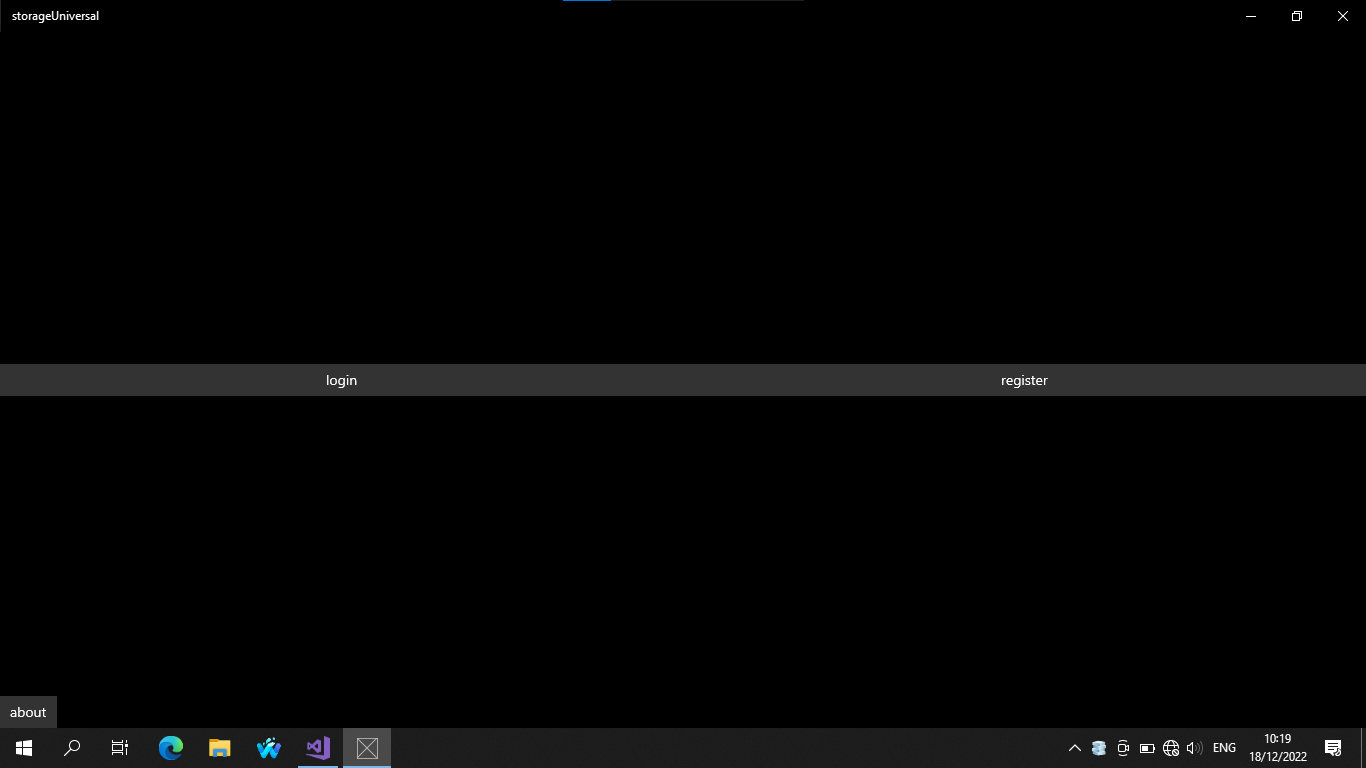
[קשרי הגומלין בין הטבלאות 67](#_Toc122786825)

# צד לקוח

## עמודים באפליקציה

### עמוד ראשי

מטרת עמוד זה היא להפנות את המשתמש לעמודים המשמעותיים האחרים בפרויקט- כניסה, הרשמה, ועמוד אודות. מצורפת למטה תמונה:



הקוד האחורי לעמוד (הBACKEND):

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;

// 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 MainPage()

        {

            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;

        }

        // 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();

                    }

                }

            }

        }

        //sends user to login

        //שולח את המשתמש לעמוד הכניסה

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

        {

            login.SentFrom = typeof(MainPage);

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

        }

        //sends user to register

        //שולח את המשתמש לעמוד הרשמה

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

        {

            Register.SentFrom = typeof(MainPage);

            Frame.Navigate(typeof(Register));

        }

        //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));

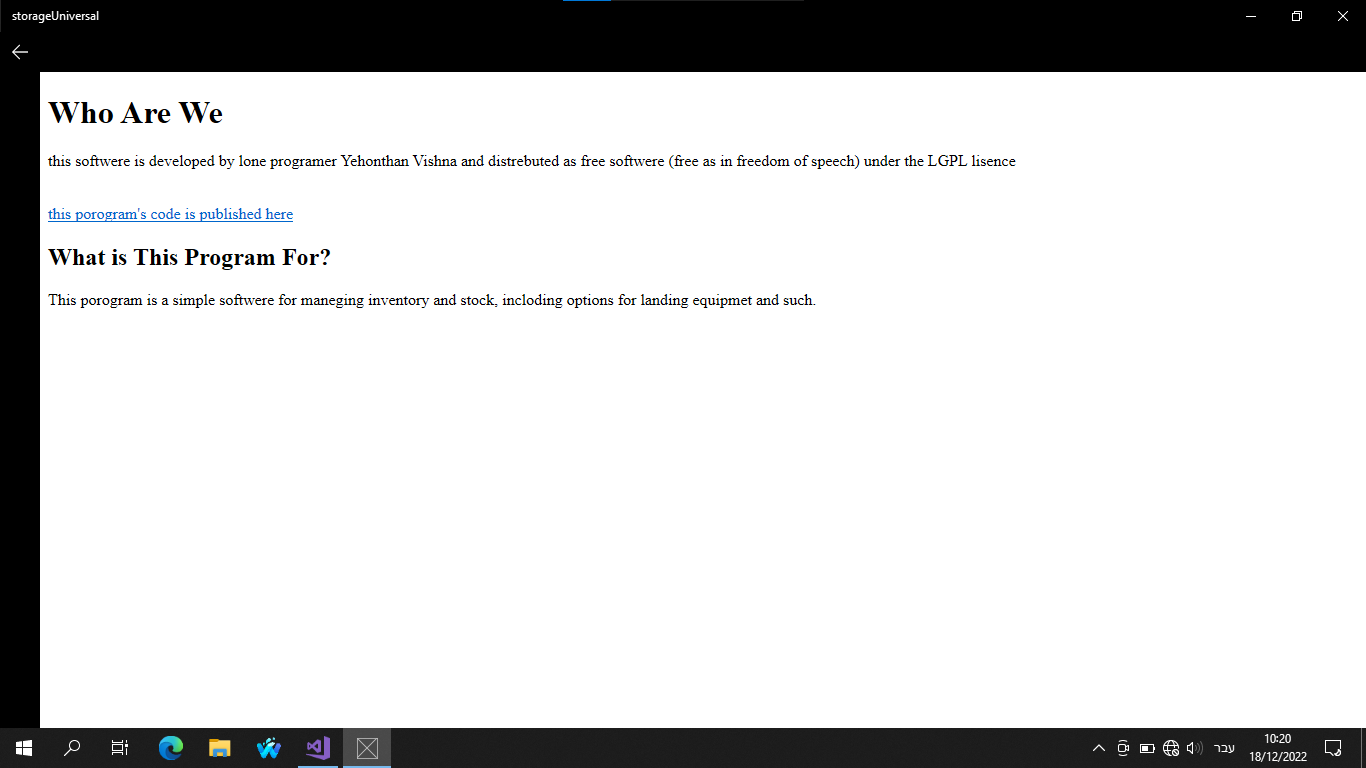
        }

    }

}

### עמוד אודות

מטעמי נוחות עיצוב, העדפתי לשלב את עמוד האודות על האפליקציה בצורת עמוד HTML והוספתי אותו במסגרת אוביקט WEBVIEW לעמוד באפליקציה. מטרת עמוד זה היא להסביר אודות האפליקציה, ומה מטרתה.



הקוד האחורי לעמוד (BACKEND):

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();

            }

        }

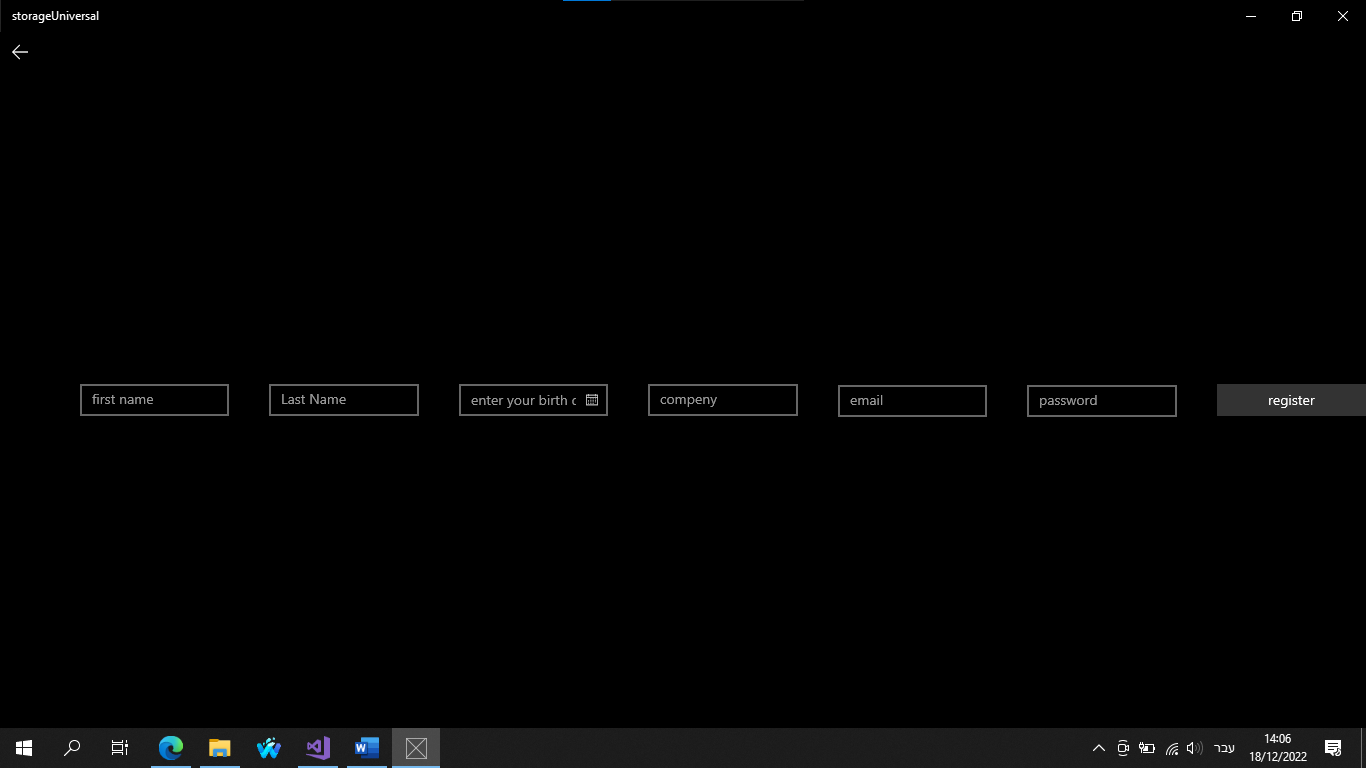
    }

}

### עמוד הרשמה

מטרת עמוד ההרשמה היא לקבל את פרטי המשתמש ולבדוק אותם (שהשדות חוקיים, ואין כפילויות באימיילים), להעביר אותם לשירות הרשת- כאשר שירות הרשת יוסיף את פרטי המשתמשים למסד הנתונים.

תמונת העמוד:



בדיקות ההרשמה הנוכחיות:

עד כה, לפני ההרשמה, נבדק האם המייל חוקי, ונבדק שהמייל לא קיים כבר במסד נתונים, לאחר ההרשמה (בהנחה שהושלמה בהצלחה), העמוד שולח את המשתמש חזרה לעמוד הראשי.

הקוד האחורי (BACKEND):

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();

            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;

            }

            var date = BDate.Date;

            DateTime time = date.Value.DateTime;

            usr.BDate = time;

            usr.Fname = FN.Text;

            usr.Lname = LN.Text;

            usr.Email = email.Text;

            usr.Password = pass.Text;

            usr.Compeny = compeny.Text;

            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();

            }

        }

        private void Grid\_KeyDown(object sender, KeyRoutedEventArgs e)

        {

        }

    }

}

### עמוד כניסה

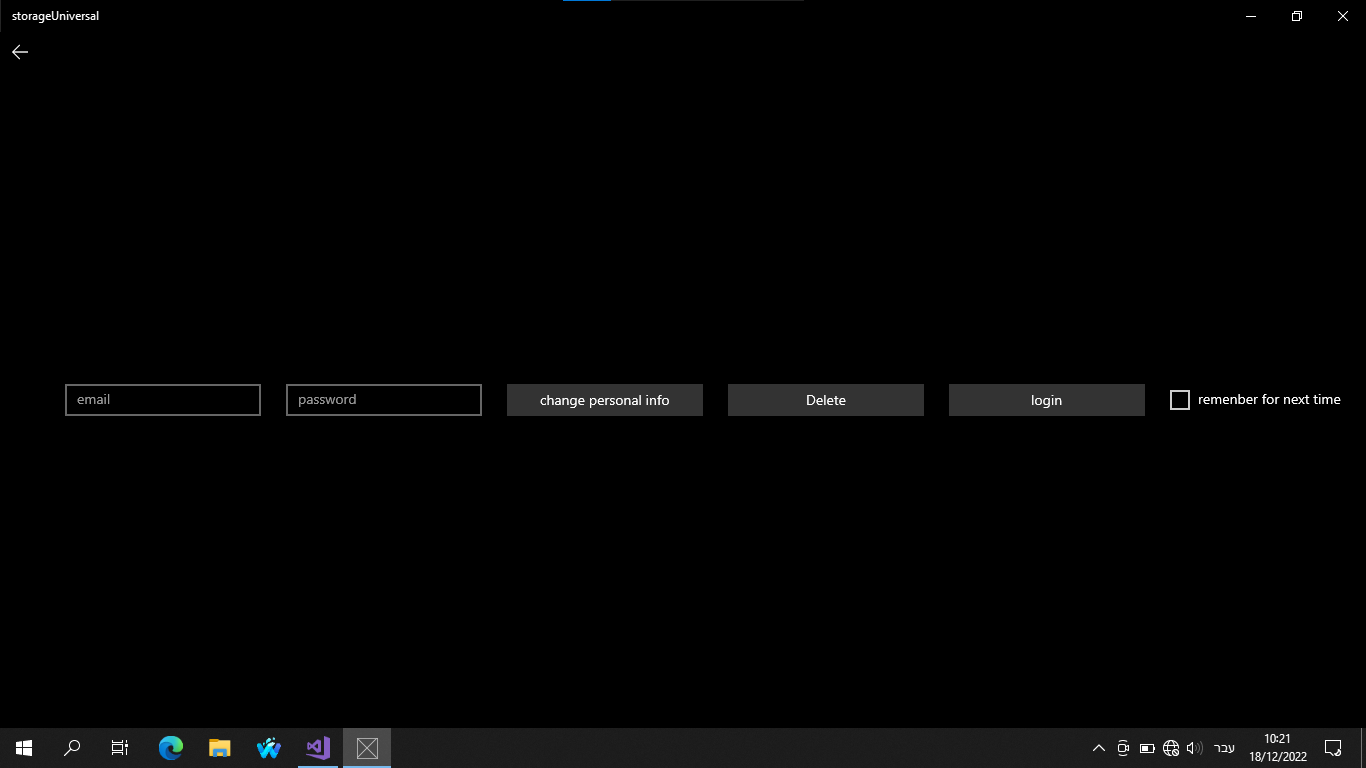
מטרת עמוד הכניסה היא לקחת מהמשתמש את אימייל והסיסמה שלו, ואז לחבר אותו למערכת ולתת לו לבצע מספר פעולות:

* לערוך את פרטיו האישיים
* למחוק את המשתמש שלו
* להיכנס לעמוד בו יראה את כל הפרטי מלאי, וההשאלות האקטיביות שלו.
* להעביר את מנהל המערכת (בעל האימייל: [admin@administrator.adm](mailto:admin@administrator.adm) והסיסמה: Admin) לעמוד ניהול משתמשים.

בנוסף, בעמוד זה יישמתי שימוש במסד נתונים מקומי: SQL LITE כדי לשמור את המשתמש האחרון שהתחבר למערכת במסד נתונים מקומי, כדי שיוכל להתחבר אוטומטית בכניסה הבאה (יש לו בחירה האם לשמור את נתוניו מקומית או לא). כשמשתמש ששמורים במחשבו בSQL LITE פרטי ההתחברות שלו נכנס לעמוד הכניסה, עולה לו תפריט POPUP שנותן לו את הבחירה האם להשתמש בנתוני משתמש השמורים או לא.

אם משתמש מכניס אימייל או סיסמה שלא נכונים, עולה לו מודעת POPUP שמודיעה לו שאחד מפרטי ההתחברות שגויים.

תמונה של עמוד ההתחברות:



הקוד האחורי:

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;

// 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 login : 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;

        public login()

        {

            this.InitializeComponent();

            TryStartAutoLoging();

            // some code to handle mouse back + forward buttons

            // קוד להוספת פונקציית קדימה ואחורה לכפתורי העכבר

            Window.Current.Activate();

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

        }

        // 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)

            {

                res.Text = "user exists, data should be showen";

                var TempFullUsr = await UDBS.GetFullUserAsync(usr);

                FullUser = TempFullUsr;

                InventoryView.SentFrom = typeof(login);

                frame.Navigate(typeof(InventoryView));

            }

            else

            {

                res.Text = "email or password are wrong, try again";

            }

        }

        //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 login);

                logAutoCheck(logPg.Frame, logPg.res);

            }

        }

        //logs in user and sends him to change his personal info

        //שולח את המשתמש לשנות את פרטיו האישיים

        private async void SendToDitailUpdate(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);

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

            bool isAdmin = await UDBS.IsAdminAsync(usr);

            if (!isAdmin)

            {

                if (b)

                {

                    res.Text = "user exists, data should be showen";

                    var TempFullUsr = await UDBS.GetFullUserAsync(usr);

                    FullUser = TempFullUsr;

                    updateUser.SentFrom = typeof(login);

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

                }

                else

                {

                    res.Text = "email or password are wrong, try again";

                }

            }

            else

            {

                FullUser = usr;

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

            }

        }

        //deletes user from system

        //מוחק את המשתמש

        private async void DelBot\_Click(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.DeleteUserAsync(usr);

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

        }

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

        // מחבר את המשתמש, ושולח אותו לרשימת המלאי שלו

        private async void LoginAndSendToInventoryList(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);

            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});

                }

                InventoryView.SentFrom = typeof(login);

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

            }

            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 (FormatException)

            {

                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();

            }

        }

    }

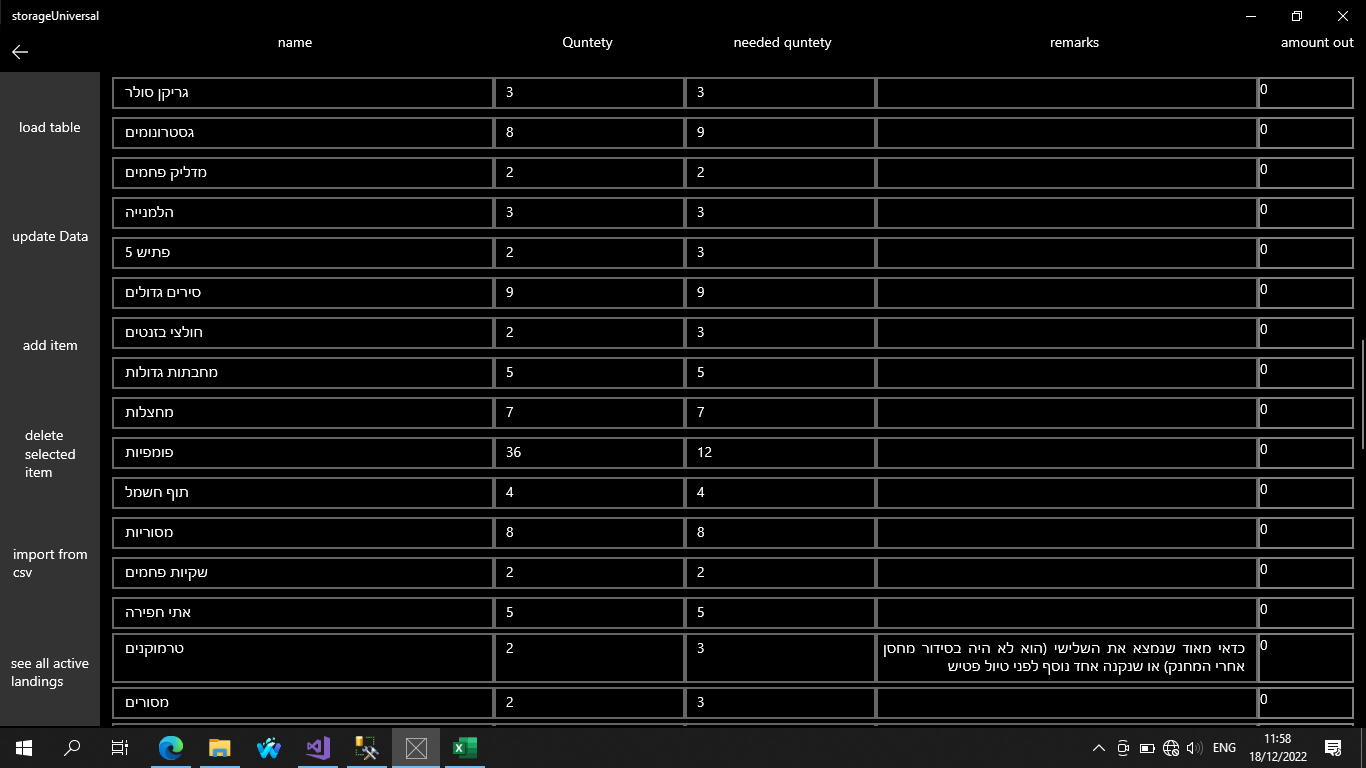
}

### עמוד עריכת מלאי

זהו העמוד המרכזי והמשמעותי ביותר בפרויקט. בעמוד זה, מוצגת עבור המשתמש רשימה של כל המלאי שיש בבעלותו. המשתמש מגיע לעמוד לאחר שהתחבר בעמוד הכניסה, ומוצגת בפניו כל המלאי שלו באובייקט ListView. למשתמש יש מספר דברים שהוא יכול לעשות עם המלאי:

* הוא יכול למחוק פריט
* להוסיף פריט
* לערוך את המידע של הפריט (ישירות בתוך הטבלה)
* לייבא פריטים חדשים מקובץ CSV
* להוסיף השאלה של פריט על ידי קליק ימני על הפריט (יעלה Popup שיבקש ממנו את פרטי ההשאלה, לאחר מכן ההשאלה תישלח לשירות רשת ותתווסף לטבלאות ההשאלות).

תמונה של העמוד:



הקוד האחורי (back end):

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;

// 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 = login.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;

        }

        //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();

                    }

                }

            }

        }

        private void InventoryTbl\_ItemClick(object sender, ItemClickEventArgs e)

        {

        }

        //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);

                UIElement b = sender as UIElement;

                b.ContextFlyout = rightClick;

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

                rightClick.ShowAt(b, point);

            }

        }

        //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 שינוי פרטים אישיים

מטרת עמוד זה היא לאפשר למשתמש לעדכן את פרטיו האישיים במערכת. המשתמש מופנה לעמוד זה מעמוד ההתחברות, ולאחר שהופנה לעמוד זה הוא רואה את פרטיו האישיים כפי שהם מופיעים בטבלת Users במסד הנתונים (פרטים אלו הגיעו למשתמש דרך שירות הרשת).

ביכולתו של המשתמש לשנות את:

* שמו הפרטי
* שם המשפחה שלו
* תאריך הלידה שלו
* כתובת הדוא"ל שלו
* הסיסמה שלו

תמונה של עמוד זה:

![תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הקוד האחורי (back end):

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 = login.FullUser;

        public static Type SentFrom;

        public updateUser()

        {

            this.InitializeComponent();

            //מכניס לשדות את הנתונים הנוכחיים 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(login));

            }

            else

            {

                var failPop = new ContentDialog()

                {

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

                    CloseButtonText = "ok"

                };

                await failPop.ShowAsync();

            }

            }

        //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 לניהול השאלות פעילות

המשתמש מופנה לעמוד זה מעמוד ניהול המלאי. מטרת עמוד זה היא לתת למשתמש אפשרות לראות את כל ההשאלות ציוד שקיימות ברגע זה ואת פרטיהן: שם הציוד שנלקח, הכמות שנלקחה, ועל ידי מי. בסופו של דבר, יהיה בעמוד גם אפשרות לשחרר השאלות ציוד מהמערכת ולמחוק אותן.

תמונה של העמוד:



הקוד האחורי (back end):

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;

// 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;

        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);

            var borrows = new List<codes.Borrow>();

            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());

                await bro.SetName(bro.ItemId);

                borrows.Add(bro);

            }

            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();

            }

        }

    }

}

### עמוד מנהל

מטרת עמוד זה היא לאפשר למנהל המערכת שינוי, עדכון, מחיקה והוספה של משתמשים במערכת או של פרטיהם.

המנהל מופנה לעמוד זה על ידי לחיצה על כפתור עדכון הפרטים בעמוד ההתחברות, זאת לאחר שהוכנסו פרטי המשתמש של המנהל.

כרגע פרטי המנהל הם:

Email: [admin@administrator.adm](mailto:admin@administrator.adm)

Password: Admin

לאחר ששירות הרשת אישר שפרטי המשתמש מתאימים למנהל, המשתמש מופנה לעמוד בו יש טבלה של כל המשתמשים הנוכחיים ופרטיהם, ביכולת המנהל:

* למחוק משתמש
* לשנות פרטי משתמש
* להוסיף משתמש ריק

תמונת העמוד:

תמונה שמכילה טקסט, טלוויזיה, שחור

התיאור נוצר באופן אוטומטי

הקוד האחורי (back end):

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.Xml;

using System.Data;

using Windows.UI.Core;

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

namespace storageUniversal

{

    /// <summary>

    /// This is a page for the admin to update the personal detailes of all existing users

    /// </summary>

    public sealed partial class AdminPanel : Page

    {

        //משתנה המכיל את פרטי ההתחברות של המנהל

        //the admin user

        public UserDBServ.User AdminUser = login.FullUser;

        public static Type SentFrom { get; internal set; }

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

        //user list as it is in the last refresh

        public List<User> UsersOriginal;

        //רשימה של כל המשתמשים כפי שהם כעת בטבלה

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

        public List<User> BinedUsersInTbl;

        public AdminPanel()

        {

            this.InitializeComponent();

            //UsersTbl.Margin = new Thickness(0, back.Height, 0, 0);

            loudTbl();

            //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();

                    }

                }

            }

        }

        //מכניס את כל המשתמשים במערכת לטבלה

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

        public async void loudTbl()

        {

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

            var r = await s.GetAdminUserTblAsync(AdminUser);

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

            foreach(DataRow a in r.Rows)

            {

                User row = new User();

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

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

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

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

                row.BDate = DateTime.Parse(a["BDate"].ToString());

                row.Compeny = a["compeny"].ToString();

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

                Users.Add(row);

            }

            UsersTbl.ItemsSource = Users;

            BinedUsersInTbl = Users;

            UsersOriginal = new List<User>();

            foreach (User Row in Users)

            {

                UsersOriginal.Add(Row.copy());

            }

        }

        ContentDialog changesApplyed = new ContentDialog()

        {

            Title = "Sucsess",

            Content = "The changes to users info where sucsussfully commited to database",

            CloseButtonText = "Ok"

        };

        //converts local User type To User type in web service

        public UserDBServ.User LocalUserToWebUser(User user)

        {

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

            NewUser.ID = user.ID;

            NewUser.Fname = user.Fname;

            NewUser.Lname = user.Lname;

            NewUser.Compeny = user.Compeny;

            NewUser.Email = user.Email;

            if (user.BDate == null)

            {

                NewUser.BDate = DateTime.Parse(DateTimeOffset.MinValue.ToString());

            }

            else

            {

                NewUser.BDate = user.BDate;

            }

            NewUser.Password = user.Password;

            return NewUser;

        }

        //בלחיצה על כפתור העידכון, מעדכן את פרטי המשתמשים במסד נתונים לפי השינויים שעשה המנהל

        //a code for applying changes admin made to User's ditailes in db

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

        {

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

            var CurrentUsers = UsersTbl.Items;

            bool IsAllOk = true;

            foreach (User user in CurrentUsers)

            {

                foreach (User OgUsr in UsersOriginal)

                {

                    if (OgUsr.ID == user.ID)

                    {

                        if (!OgUsr.IsSame(user))

                        {

                            UserDBServ.User NewUser = LocalUserToWebUser(user);

                            UserDBServ.User OldUser = LocalUserToWebUser(OgUsr);

                            bool hadWorked = await s.updateUserByIdAdminAsync(user.ID, AdminUser, NewUser);

                            IsAllOk = IsAllOk && hadWorked;

                        }

                    }

                }

            }

            if (IsAllOk)

            {

                await changesApplyed.ShowAsync();

            }

        }

        //בחיצה על הכפור, מוסיף לטבלה ולמסד הנתונים משתמש ריק

        //code to add a new row representing new user in tbl

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

        {

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

            User NewUser = new User();

            var id = await s.AddEmptyUserAsync();

            NewUser.ID = id;

            BinedUsersInTbl.Add(NewUser);

            UsersOriginal.Add(NewUser);

            UsersTbl.ItemsSource = null;

            UsersTbl.ItemsSource = BinedUsersInTbl;

        }

        //בלחיצה על כפתור המחיקה מוחק את המשתמש המסומן

        //code for deleting selected user from table and db

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

        {

            int SelIn = UsersTbl.SelectedIndex;

            User SelectedUser = UsersTbl.Items[SelIn] as User;

            int id = SelectedUser.ID;

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

            bool hasWorked = await r.DeleteUserAdminAsync(AdminUser,id);

            UsersTbl.ItemsSource = null;

            loudTbl();

        }

        //מחזיר את המשתמש למסך הקודם בלחיצה על כפתור החזרה

        //back button click

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

        {

            Frame frame = Window.Current.Content as Frame;

            if (frame.CanGoBack)

            {

                frame.GoBack();

            }

        }

    }

}

## טיפוסים וקבצי קוד

### טיפוס Borrow

מטרת טיפוס זה היא להכיל את כל התכונות של השאלת ציוד אחת. יש צורך להכיל את תכונות אלו בטיפוס כדי להציגם בטבלה [בעמוד השאלות ציוד](#_עמוד_לניהול_השאלות).

הקוד של הטיפוס:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using storageUniversal.BorowwDb;

namespace storageUniversal.codes

{

    public class Borrow

    {

        //תכונות של טיפוס השאלה

        // Fields

        private int itemId;

        private string borrowedBy;

        private DateTime when;

        private float quantity;

        private float borrowingId;

        private int userId;

        private string name;

        //הבנאי הראשי

        // Default constructor

        public Borrow() { }

        //בנאי שמקבל מספר זהות

        // Constructor with item ID parameter

        private Borrow(int id)

        {

            // Set the item ID

            this.ItemId = id;

        }

        //פעולות גט וסט לכל התכונות

        // getters and setters to all properties

        public int ItemId

        {

            get { return itemId; }

            set { this.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;

        }

        public string Name

        {

            get => name;

            set => name = value;

        }

        //בשל הקשרים בטבלה אין בטבלה של ההשאלות את שם הפריט

        //מטרת פעולה זו היא להשיג משירות הרשת את שם הפריט ולשים אותו בתוך העצם

        // Method to set the name of the user based on their ID

        public async Task<string> SetName(int id)

        {

            // Create a new instance of the BorowwDb.BorowwingsDBSoapClient class

            var b = new BorowwDb.BorowwingsDBSoapClient();

            // Set the name based on the ID using the web service

            name = await b.getNameAsync(id);

            // Return the name

            return name;

        }

    }

}

### קוד לניהול מסד נתונים SQLite (Constants, UserDataBase)

הוספתי בפרויקט רכיבים לניהול מסד נתונים מקומי מסוג SQL lite, מטרת מסד נתונים זה הוא לשמור את הפרטים של המשתמשים שהתחברו בעבר כדי שהם יוכלו להתחבר שוב בעתיד בלי למלא בשנית את פרטיהם.

הרכיבים שהוספתי הם:

* טיפוס Constants: מטרת טיפוס זה היא להכיל פרטים קבועים שנחוצים כדי ליצור מסד נתונים מקומי כמו המסלול שבו ישמר מסד הנתונים במערכת הקבצים, ושם מסד הנתונים.
* טיפוס UserDataBase: מטרת טיפוס זה היא לפשט את תהליך השימוש במסד הנתונים במקומות האחרים באפליקציה, זאת על ידי יצירת ממשק שבו הטיפוס יוצר את מסד הנתונים ואת הקשר אליו, ופעולות בטיפוס יוציאו ויכניסו רשומות במסד הנתונים כטיפוסים מסוג User.

הקוד לטיפוס constants:

using System.IO;

using Windows.Storage;

public static class Constants

{

    /// <summary>

    /// this code segment defines a few constants nessesry to create sql lite data base

    /// </summary>

    //the filename in which the db will be saved

    //שם הקובץ בו ישמר מסד הנתונים

    public const string DatabaseFilename = "TodoSQLite.db3";

    //the path to the db file

    //המסלול לקובץ המכיל את מסד הנתונים

    public static string DatabasePath =>

        Path.Combine(ApplicationData.Current.LocalFolder.Path, "sqliteSample.db");

}

הקוד לטיפוס UserDataBase:

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;

        }

    }

}

### עצם InventoryRow

על מנת שנוכל להציג פרטי שורת מלאי ממסד הנתונים בטבלה כמו [בעמוד עריכת המלאי](#_עמוד_עריכת_מלאי), יש צורך בטיפוס שיכיל את מידע זה. טיפוס זה מכיל את התכונות:

* שם הפריט
* מספר הפריט
* כמות הקיימת מהפריט
* הכמות בה יש צורך מאותו פריט
* הערות כלליות על פריט זה

הקוד של טיפוס זה:

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;

        }

    }

}

### טיפוס User

על מנת לאחסן ביתר קלות את פרטי המשתמשים, וכדי שהמנהל יוכל לראות את פרטיהם בטבלה ב[עמוד המנהל](#_עמוד_מנהל) יש להשתמש בעצמים שיכילו את פרטי המשתמשים. לצורך זה בחרתי בטיפוס User. טיפוס User מכיל את כלל המידע על המשתמש בינהם:

* מספר מזהה
* שם פרטי
* שם משפחה
* תאריך לידה
* החברה בה הם עובדים
* כתובת הדוא"ל שלהם
* הסיסמה שלהם

הקוד לטיפוס:

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

    }

}

# צד שרת

## שירות רשת לניהול משתמשים

מטרת שירות רשת זה היא לנהל את המידע על המשתמשים באתר- הוא מנהל את טבלת Users. עיקר פעולת שירות רשת זה היא:

* לעדכן את פרטי המשתמשים
* לאמת את קיומו של המשתמש במסד הנתונים
* לבדוק האם משתמש הוא במקרה מנהל
* למחוק משתמשים
* להוסיף משתמשים חדשים

תמונה של שירות רשת זה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שירות רשת זה מכיל מספר קבצים וטיפוסים אותם מייד אסביר.

מלבד הטיפוס [Connection](#_טיפוס_Connection) שעליו ארחיב מייד, יש בשירות רשת זה גם את הטיפוס [User](#_טיפוס_User) שזהה כמעט לגמרי לאותו הטיפוס בצד המשתמש, ולכן לא אסביר אותו בשנית.

### טיפוס Connection

על מנת לחסוך עבודה קשה ומרובה בפניות מול מסד הנתונים, יצרתי טיפוס שיצור פעולות פשוטות הדרושות לי להתממשקות מול מסד הנתונים: הפעלת שאילתות הצגה והוצאת מידע לטיפוסים שונים, הפעלת שאילתות שינוי ושינוי מידע בטבלאות וכו'...

הקוד של הטיפוס:

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 WpfApp1

{

    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);

        }

    }

}

### קובץ שירות הרשת עצמו- UserDBServ.asmx

מטרת קובץ שירות רשת זה הוא לאפשר שינויי, עדכון, הצגה, מחיקה, אימות ובדיקת רמת גישה (האם מנהל או לא) של משתמשים בידי כל משתמש לגבי פרטיו שלו ולגבי כל המשתמשים בעבור מנהל המערכת.

תמונה של שירות רשת זה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הקוד של שירות רשת זה:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using WpfApp1;

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;

            }

        }

    }

}

## שירותי רשת לניהול מלאי והשאלות

יש בפרויקט זה שני שירותי רשת ראשיים:

* BorowwingsDB.asmx- המכיל פעולות לניהול ההשאלות עבור כל משתמש
* InventoryFuncs.asmx- המכיל פעולות להתממשקות מול רשימות המלאי של כל משתמש.

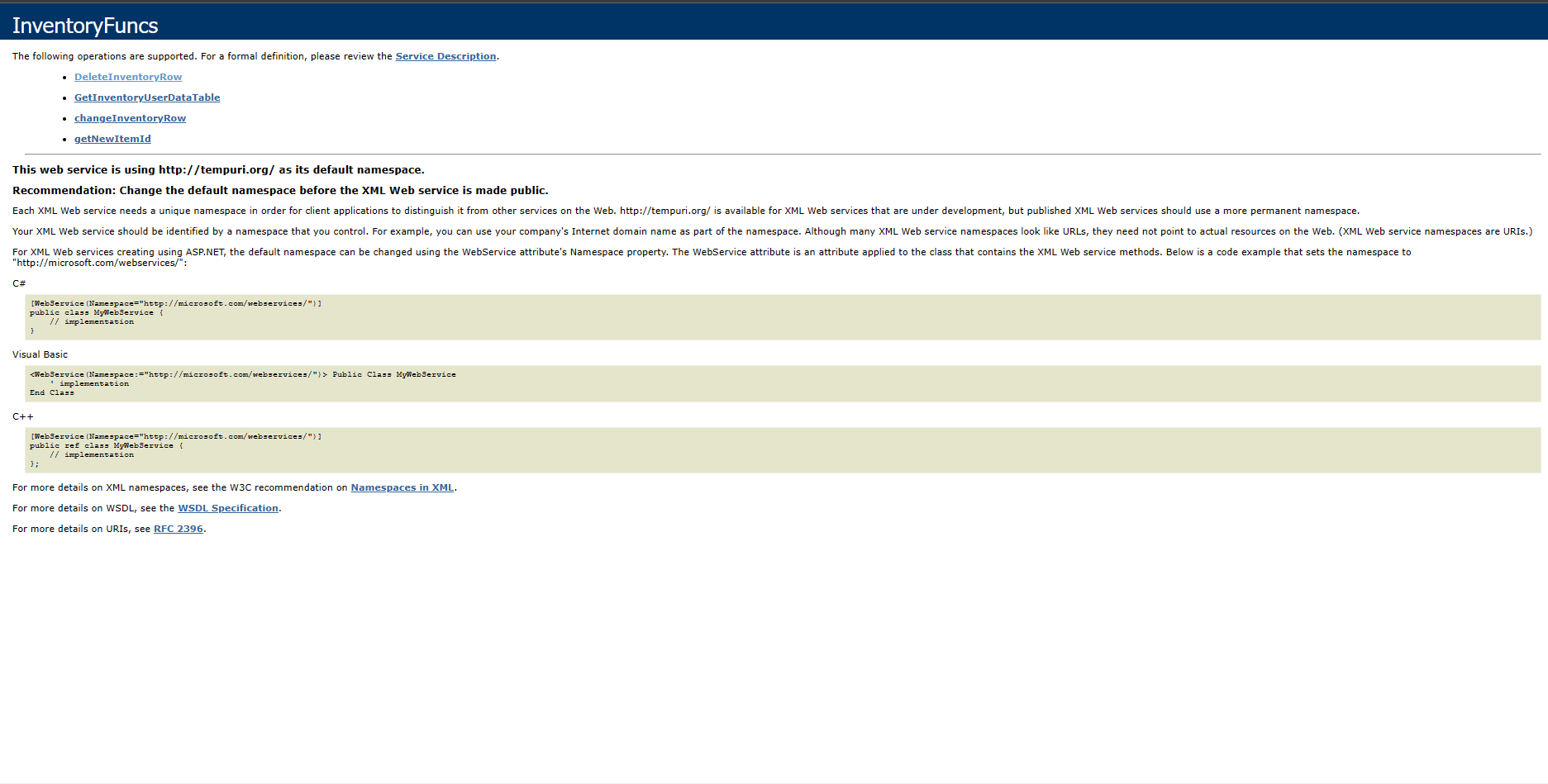
בנוסף, מכיל הפרויקט בו נמצאים שירות הרשת מספר טיפוסים, אך כולם זהים כמעט או לגמרי לטיפוסים אלו בצד לקוח או בשירות הרשת השני ולכן לא אסביר אותם בשנית. טיפוסים אלו הם:

* Connection- [הנמצא בשירות הרשת השני](#_טיפוס_Connection)
* InventoryRow- [הנמצא בצד הלקוח](#_עצם_InventoryRow)
* Borrow- [הנמצא בצד הלקוח](#_טיפוס_Borrow)

### שירות הרשת- InventoryFuncs.asmx

מטרת שירות רשת זה היא להוסיף, לשנות, למחוק, ולהציג את רשימת המלאי של כל משתמש.

תמונה של שירות רשת זה:



הקוד של שירות רשת זה:

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; }

        }

    }

}

### שירות רשת- BorowwingsDB.asmx

מטרת שירות רשת זה הוא לבצע את כל הפעולות הקשורות לעידכון, יצירה, מחיקה, והוספה של השאלות לטבלת ההשאלות במסד הנתונים:

תמונה של שירות רשת זה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הקוד של שירות רשת זה:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Net;

using WpfApp1;

using System.Data;

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();

        }

    }

}

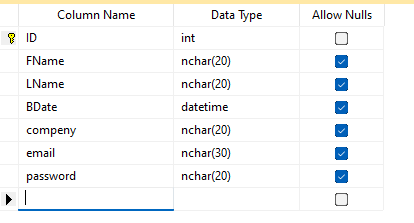
## טבלאות SQL

בפרויקט שלי השתמשתי עד כה ב3 טבלאות SQL בעלות תפקידים שונים:

* טבלת Users: שתפקידה לאחסן את פרטיהם האישיים של כל המשתמשים
* טבלת Inventory: שתפקידה להחזיק את כל פרטי המלאי של כל המשתמשים ופרטי פריטים אלו.
* טבלת BorrowedItems- שתפקידה לאחסן את הפרטים על כל ההשאלות הקיימות כרגע עבור כל המשתמשים.

### טבלת Users

תמונה של תכנון הטבלה:



### טבלת Inventory

תמונה של תכנון הטבלה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

### טבלת BorrowedItems

תמונה של תכנון הטבלה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

### קשרי הגומלין בין הטבלאות

ניתן לראות שיש מספר שדות הקשורים זה בזה בין הטבלות:

* קשר יחיד לרבים בין ID בUsers (יחיד) לOwnerUserId ב Inventory (רבים)
* קשר יחיד לרבים בין ID ב Users(יחיד) ל UserId ב BorrowedItems (רבים)
* קשר יחיד לרבים בין ID ב Inventory(יחיד) ל ItemId בBorrowedItems (רבים)

המחשה של הקשרים בתרשים:

