**תכנון ותכנות מערכות ניהול חנות תקליטים**

**מגישה: סיון ארזי**

**תעודת זהות: 213155401**

**מורה: עוז ארווץ**

**בית ספר: מקיף י"א אשדוד**

**שנה: 2020**

תוכן עניינים

[מבוא: 5](#_Toc43231928)

[נושא הפרויקט: 5](#_Toc43231929)

[מהות התוכנה: 5](#_Toc43231930)

[מסמך יזום: 6](#_Toc43231931)

[תהליכים מרכזיים בתוכנה: 6](#_Toc43231932)

[תרשים: 8](#_Toc43231933)

[מדריך למשתמש: 9](#_Toc43231934)

[דף ראשי: 9](#_Toc43231935)

[תהליך ניהול מלאי: 9](#_Toc43231936)

[תהליך ניהול לקוחות: 10](#_Toc43231937)

[צפייה בלקוחות: 10](#_Toc43231938)

[הוספת לקוחות: 10](#_Toc43231939)

[עדכון פרטי לקוח: 11](#_Toc43231940)

[מחיקת לקוח: 11](#_Toc43231941)

[תהליך ניהול עובדים: 12](#_Toc43231942)

[צפייה בעובדים: 12](#_Toc43231943)

[הוספת עובד: 12](#_Toc43231944)

[עדכון פרטי עובד: 13](#_Toc43231945)

[מחיקת עובד: 13](#_Toc43231946)

[תהליך ניהול הוצאות שוטפות: 14](#_Toc43231947)

[הוצאות שוטפות: 14](#_Toc43231948)

[הוספת הוצאות שוטפות: 14](#_Toc43231949)

[עדכון הוצאות שוטפות: 15](#_Toc43231950)

[מחיקת הוצאות שוטפות: 15](#_Toc43231951)

[תהליך ניהול הזמנות מספק: 16](#_Toc43231952)

[הוספת הזמנה מספק: 16](#_Toc43231953)

[עדכון הזמנה מספק: 17](#_Toc43231954)

[מחיקת הזמנה מספק: 17](#_Toc43231955)

[תהליך ניהול ספקים: 18](#_Toc43231956)

[הוספת ספק חדש: 18](#_Toc43231957)

[עדכון פרטי הספק: 19](#_Toc43231958)

[מחיקת ספקים: 19](#_Toc43231959)

[תהליך ניהול אלבומים: 20](#_Toc43231960)

[הוספת אלבום: 20](#_Toc43231961)

[עדכון אלבום: 21](#_Toc43231962)

[מחיקת אלבום: 21](#_Toc43231963)

[תהליך ניהול עסקה: 22](#_Toc43231964)

[הוספת עסקה: 22](#_Toc43231965)

[עדכון עסקה: 23](#_Toc43231966)

[מחיקת עסקה: 23](#_Toc43231967)

[תהליך יצירת דוחות: 24](#_Toc43231968)

[המדריך למתכנת: 25](#_Toc43231969)

[ישויות וקשרים: 25](#_Toc43231970)

[בסיס נתונים: 26](#_Toc43231971)

[טבלת לקוחות: 26](#_Toc43231972)

[טבלת עובדים: 26](#_Toc43231973)

[טבלת ספקים: 26](#_Toc43231974)

[טבלת אלבומים: 26](#_Toc43231975)

[טבלת הוצאות: 26](#_Toc43231976)

[טבלת מלאי: 27](#_Toc43231977)

[טבלת פרטי הזמנות מספקים: 27](#_Toc43231978)

[טבלת הזמנות מספקים: 27](#_Toc43231979)

[טבלת פרטי עסקאות: 27](#_Toc43231980)

[טבלת עסקאות: 27](#_Toc43231981)

[תרשים זרימת מידע – תהליכים מרכזיים: 28](#_Toc43231982)

[ניהול לקוחות: 28](#_Toc43231983)

[ניהול עובדים: 28](#_Toc43231984)

[ניהול ספקים: 29](#_Toc43231985)

[ניהול אלבומים: 29](#_Toc43231986)

[ניהול הוצאות שוטפות: 30](#_Toc43231987)

[ניהול מלאי: 30](#_Toc43231988)

[ניהול הזמנות מספקים: 31](#_Toc43231989)

[ניהול עסקאות: 32](#_Toc43231990)

[יצירת דוחות: 33](#_Toc43231991)

[הסבר על קוד התוכנה: 34](#_Toc43231992)

[דף ראשי: 38](#_Toc43231993)

[ניהול לקוחות: 40](#_Toc43231994)

[ניהול עובדים: 43](#_Toc43231995)

[ניהול ספקים: 46](#_Toc43231996)

[ניהול הוצאות שוטפות: 49](#_Toc43231997)

[ניהול מלאי: 53](#_Toc43231998)

[ניהול אלבומים: 54](#_Toc43231999)

[ניהול עסקאות: 57](#_Toc43232000)

[ניהול הזמנות מספק: 63](#_Toc43232001)

[ניהול דוחות: 69](#_Toc43232002)

[חלון שגיאה: 72](#_Toc43232003)

[סיכום אישי: 73](#_Toc43232004)

# מבוא:

## נושא הפרויקט:

הקמה של מערכת ממוחשבת לחנות תקליטים – “Sivan’s Records”

## מהות התוכנה:

התכונה נכתבה במטרה ליצר דרך יעילה ומסודרת להתנהל עם חנות המוכרת אלבומים. התנהלות עם מערכת שאינה ממוחשבת עלולה ליצור בעיות ולאבד נתונים ולכן כתבתי תוכנה אשר תשתמש בכל מאגרי המידע במחשב, ואז המידע יהיה מגובה. בנוסף זמן החיפוש של מידע (על כל אלבום לדומא) יעיל ומהיר יותר בצורה ממוחשבת מאשר בצורה ידנית.

# מסמך יזום:

## תהליכים מרכזיים בתוכנה:

**ניהול לקוחות** –

1. צפייה בכל הלקוחות
2. הוספת לקוח חדש
3. עדכון פרטים אישיים של לקוח
4. מחיקת לקוח

**ניהול עובדים** –

1. צפייה בכל העובדים
2. הוספת עובד חדש
3. עדכון פרטים אישיים של עובד
4. מחיקת עובד

**ניהול ספקים** –

1. צפייה בכל הספקים
2. הוספת ספק חדש
3. עדכון פרטים אישיים של ספק
4. מחיקת ספק

**ניהול אלבומים** –

1. צפייה בכל האלבומים
2. הוספת אלבום חדש
3. עדכון פרטים אישיים של אלבום
4. מחיקת אלבום

**ניהול הוצאות** –

1. צפייה בכל ההוצאות השוטפות
2. הוספת הוצאה שוטפת חדשה
3. עדכון פרטים של הוצאה שוטפת
4. מחיקת הוצאה שוטפת

**ניהול עסקאות** –

1. צפיה בכל העסקאות
2. הוספת עסקה חדשה
3. עדכון פרטים של עסקה
4. מחיקת עסקה

**ניהול הזמנות מספקים** –

1. צפיה בכל ההזמנות מהספקים
2. הוספת הזמנה מספק חדשה
3. עדכון פרטים של הזמנה מספק
4. מחיקת הזמנה מספק

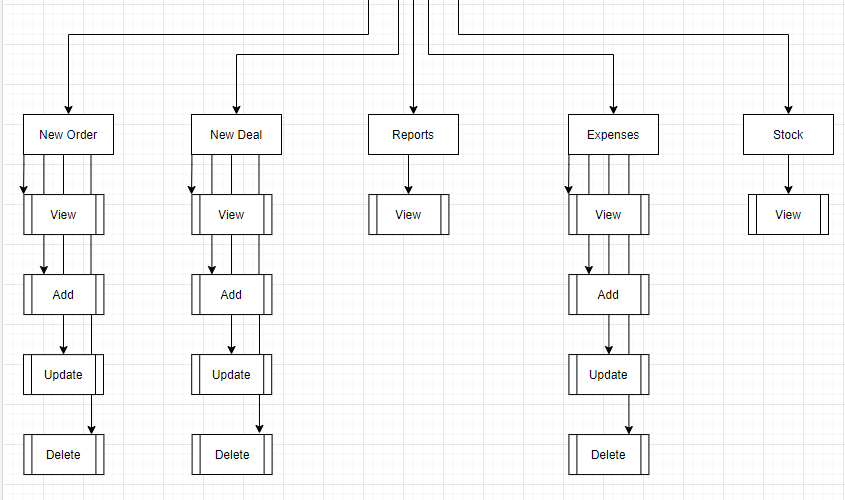
**ניהול מלאי** –

1. צפיה בכל המלאי הקיים

**דו"חות המערכת** –

1. דו"ח הוצאות והכנסות לפי חודשים
2. דו"ח חלוקת הוצאות כוללת

# תרשים:



# מדריך למשתמש:

## דף ראשי:

#### תצלום מסך:

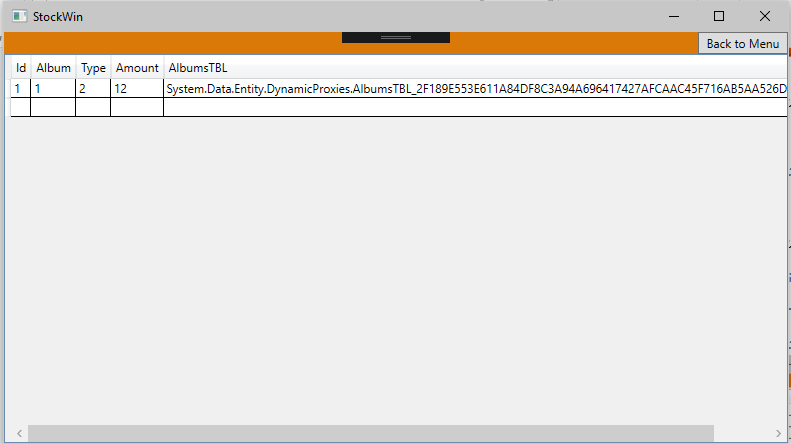


#### הסבר הדף:

דף זה משמש כמסך הבית של התוכנה, דרך דף זה ניתן להגיע לכל האפשרויות והחלונות בתוכנה.

## תהליך ניהול מלאי:

#### תצלום מסך:



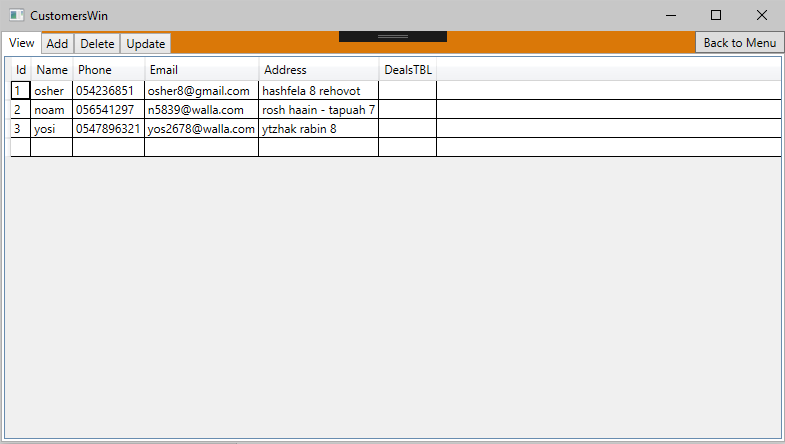
#### הסבר הדף:

בדף זה ניתן לראות את כל מלאי החנות. כלומר איזה אלבומים מאיזה סוגים נמצאים במלאי וכמה.

## תהליך ניהול לקוחות:

### צפייה בלקוחות:

#### תצלום מסך:

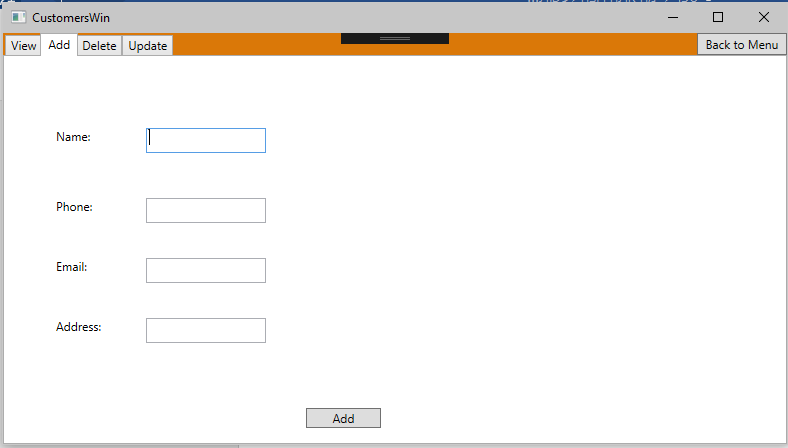


#### הסבר הדף:

דף זה הוא הדף בו ניתן לראות את כל הלקוחות הקיימים במערכת ובנוסף ניתן לגשת באותו החלון לכל אפשרויות נתוני הלקוח בתוכנה - הוספת לקוחות, עדכון לקוח מחיקת לקוח. בנוסף ניתן לחזור למסך הראשי.

### הוספת לקוחות:

#### תצלום מסך:

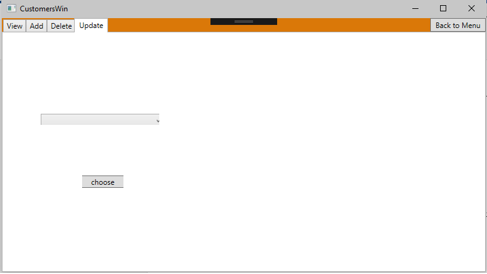


#### הסבר הדף:

דף זה הוא דף הוספת הלקוח. דרך דף זה ניתן להוסיף לקוחות עם פרטיהם החשובים שישמרו בתוכנה. ניתן למלא את הפרטים בתוך התיבות ולאחר מכן ללחוץ על כפתור "Add" שיוסיף את הנתונים לתוכנה.

### עדכון פרטי לקוח:

#### תצלום מסך:

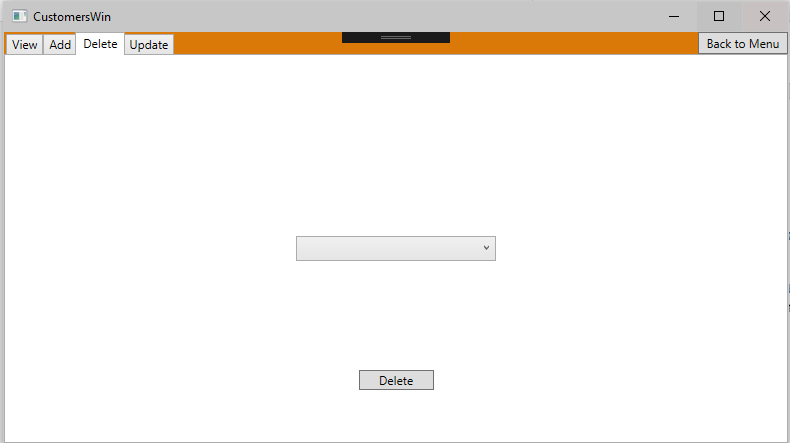


#### הסבר על הדף:

זהו דף בו ראשית ניתן לבחור לקוח השמור במערכת ולאחר הבחירה (לחיצה על כפתור "choose"), יפתחו השדות עם הפרטים של הלקוח בהם ניתן לשנות את הפרטים הקיימים של הלקוח. לאחר סיום העדכון לוחצים על כפתור "Update" ופרטי הלקוח מתעדכנים.

### מחיקת לקוח:

#### תצלום מסך:



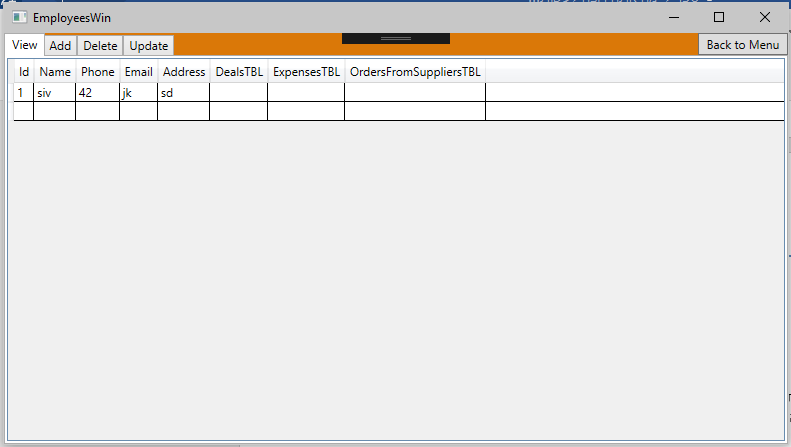
#### הסבר על הדף:

זהו דף בו ניתן לבחור לקוח הנמצא במערכת ולמחוק אותו באמצעות לחיצה על כפתור "delete".

## תהליך ניהול עובדים:

### צפייה בעובדים:

#### תצלום מסך:

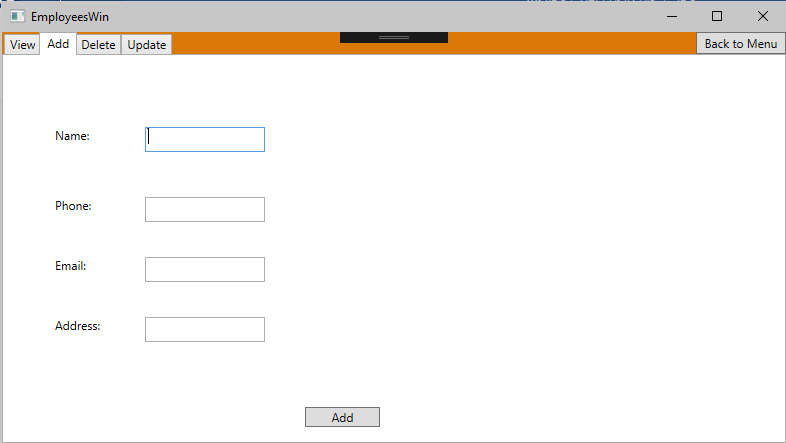


#### הסבר הדף:

דף זה הוא הדף בו ניתן לראות את כל העובדים הקיימים במערכת ובנוסף ניתן לגשת באותו החלון לכל אפשרויות נתוני העובד בתוכנה - הוספה, עדכון, מחיקת. בנוסף ניתן לחזור למסך הראשי.

### הוספת עובד:

#### תצלום מסך:

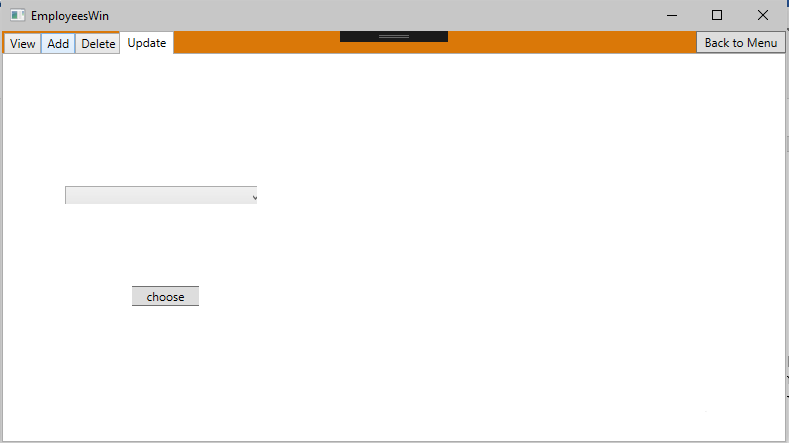


#### הסבר הדף:

דף זה הוא דף הוספת העובד. דרך דף זה ניתן להוסיף עובדים עם פרטיהם החשובים שישמרו בתוכנה. ניתן למלא את הפרטים בתוך התיבות ולאחר מכן ללחוץ על כפתור "Add" שיוסיף את הנתונים לתוכנה.

### עדכון פרטי עובד:

#### תצלום מסך:

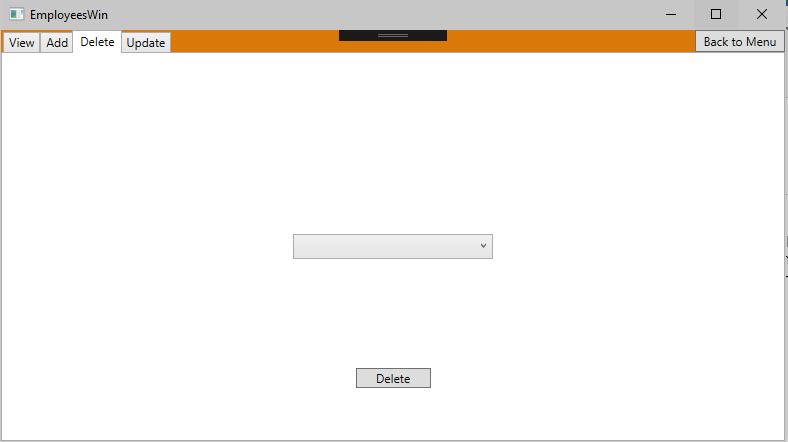


#### הסבר על הדף:

זהו דף בו ראשית ניתן לבחור עובד השמור במערכת ולאחר הבחירה (לחיצה על כפתור "choose"), יפתחו השדות עם הפרטים של העובד בהם ניתן לשנות את הפרטים הקיימים שלו. לאחר סיום העדכון לוחצים על כפתור "Update" ופרטי העובד מתעדכנים.

### מחיקת עובד:

#### תצלום מסך:



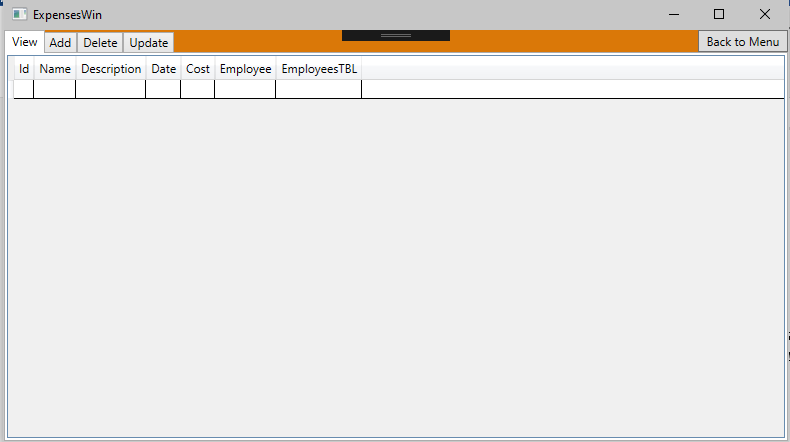
#### הסבר על הדף:

זהו דף בו ניתן לבחור עובד הנמצא במערכת ולמחוק אותו באמצעות לחיצה על כפתור "delete".

## תהליך ניהול הוצאות שוטפות:

### הוצאות שוטפות:

#### תצלום מסך:

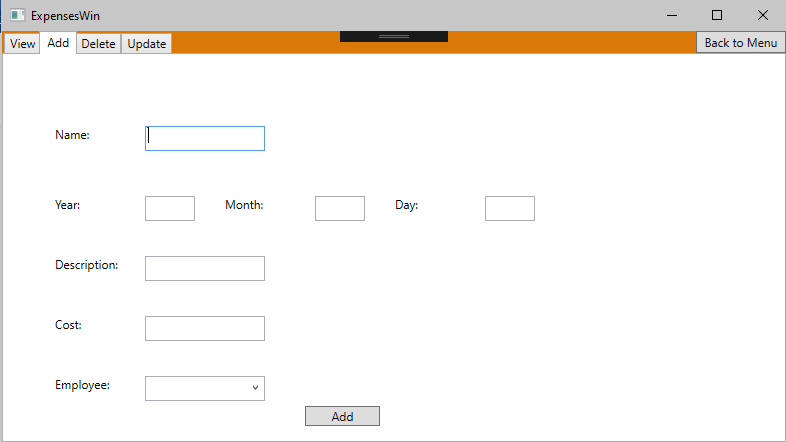


#### הסבר על הדף:

דף זה הוא הדף בו ניתן לראות את כל ההוצאות הקיימות במערכת ובנוסף ניתן לגשת באותו החלון לכל אפשרויות נתוני ההוצאות בתוכנה – הוספה, עדכון, מחיקה. בנוסף ניתן לחזור למסך הראשי.

### הוספת הוצאות שוטפות:

#### תצלום מסך:

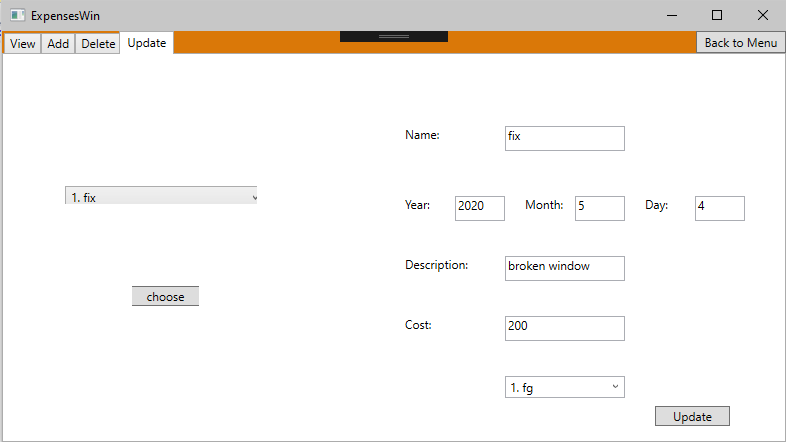


#### הסבר על הדף:

דף זה הוא דף הוספת הוצאות שוטפות. דרך דף זה ניתן להוסיף את שם ההוצאה, עלות ההוצאה והתאריך. ניתן למלא את הפרטים בתוך התיבות ולאחר מכן ללחוץ על כפתור "הוסף" שיוסיף את הנתונים לתוכנה.

### עדכון הוצאות שוטפות:

#### תצלום מסך:

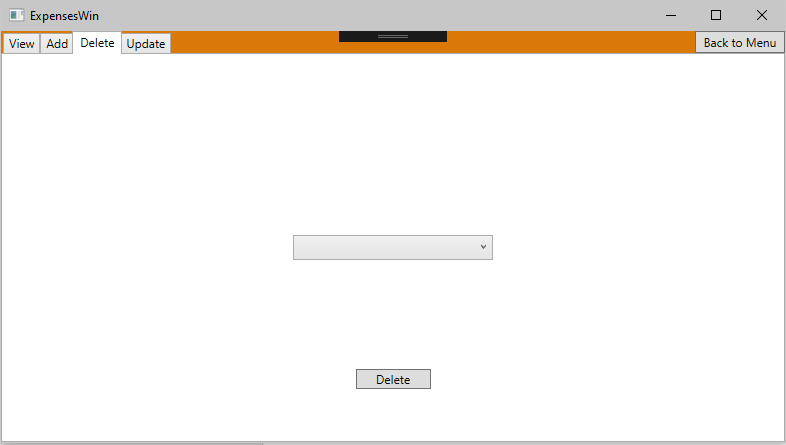


#### הסבר על הדף:

דף זה מציג את פרטי ההוצאות השוטפות שנשמרו במערכת ומאפשר לעדכנם.

### מחיקת הוצאות שוטפות:

#### תצלום מסך:

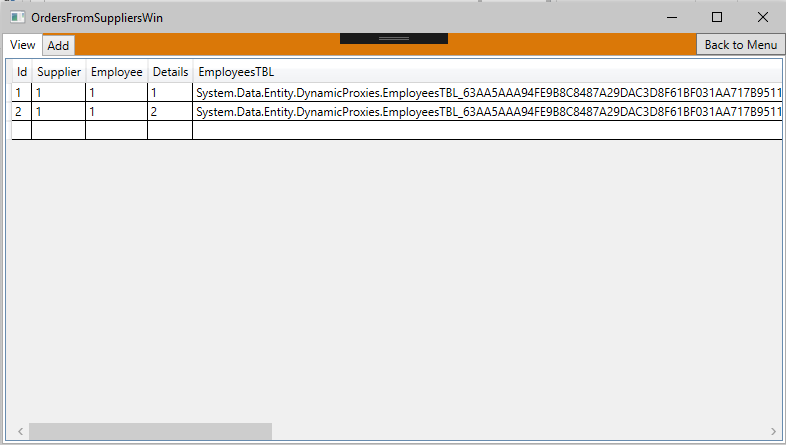


#### הסבר על הדף:

זהו דף בו ניתן לבחור הוצאה שוטפת הנמצאת במערכת ולמחוק אותה באמצעות לחיצה על כפתור "delete".

### תהליך ניהול הזמנות מספק:

#### תצלום מסך:

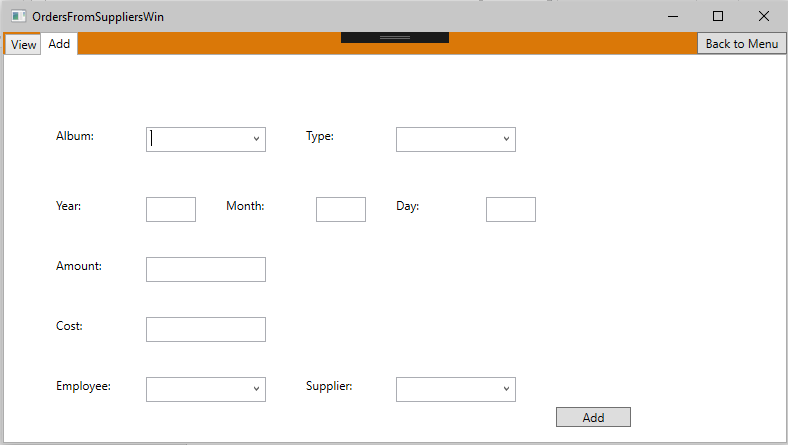


#### הסבר על הדף:

דף זה הוא הדף בו ניתן לראות את כל ההזמנות שנעשו מספקים במערכת ובנוסף ניתן לגשת באותו החלון לכל אפשרויות נתוני ההוצאות בתוכנה – הוספה, עדכון, מחיקה. בנוסף ניתן לחזור למסך הראשי.

### הוספת הזמנה מספק:

#### תצלום מסך:

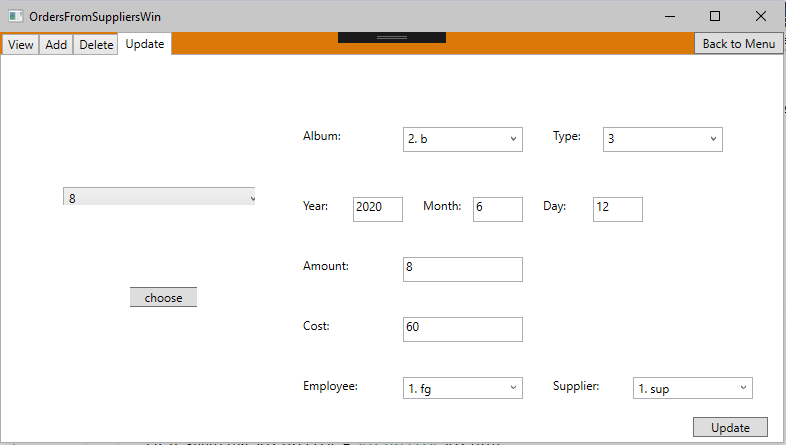
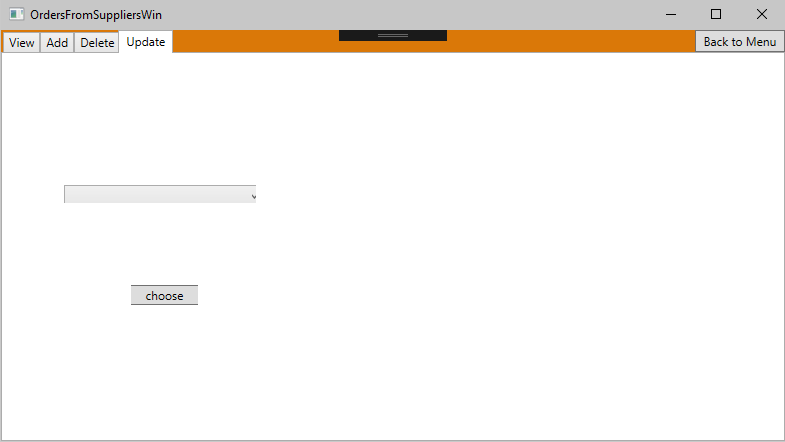


#### הסבר על הדף:

דף זה הוא דף הוספת הזמנה מספק. ניתן למלא את הפרטים בתוך התיבות ולאחר מכן ללחוץ על כפתור "הוסף" שיוסיף את הנתונים לתוכנה.

### עדכון הזמנה מספק:

#### תצלום מסך:

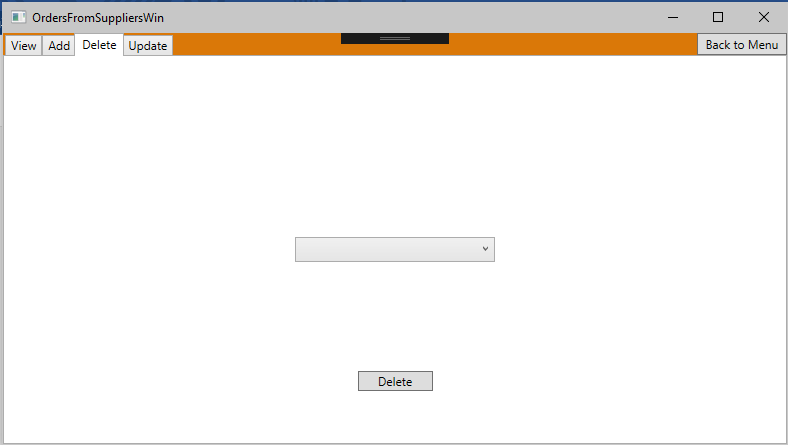
   


#### הסבר על הדף:

דף זה מציג את פרטי ההזמנה מספק שנשמרו במערכת ומאפשר לעדכנם.

### מחיקת הזמנה מספק:

#### תצלום מסך:



#### הסבר על הדף:

זהו דף בו ניתן לבחור הזמנה מספק הנמצאת במערכת ולמחוק אותה באמצעות לחיצה על כפתור "delete".

## תהליך ניהול ספקים:

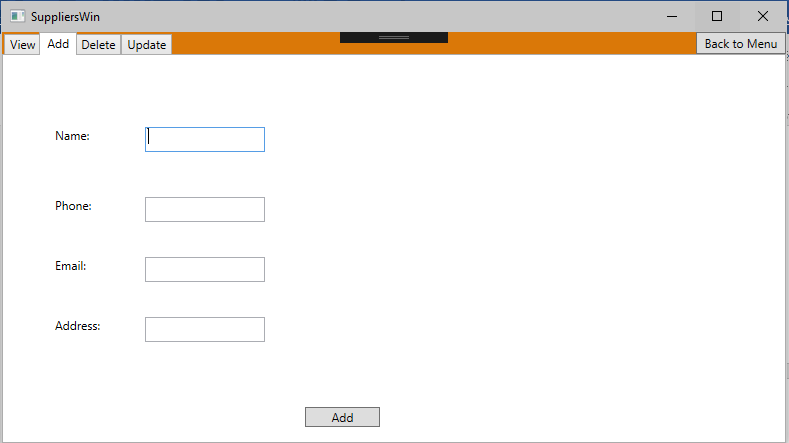
#### תצלום מסך:

#### הסבר על הדף:

דף זה מאפשר גישה נוחה וקלה לפרטי הספקים שנמצאים בבסיס הנתונים של המערכת.

### הוספת ספק חדש:

#### תצלום מסך:

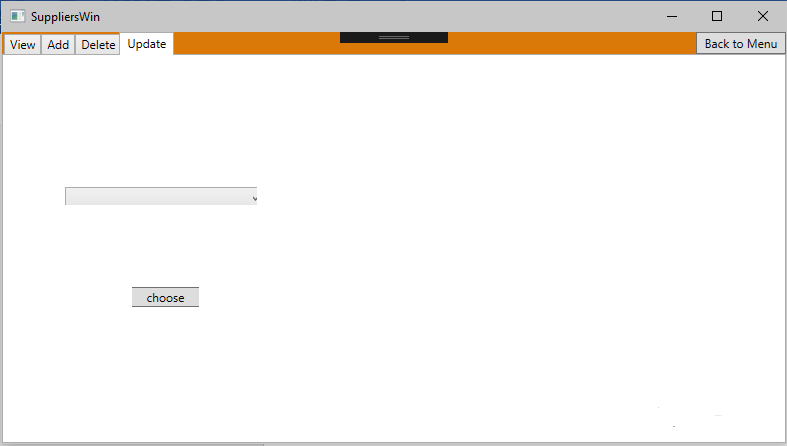
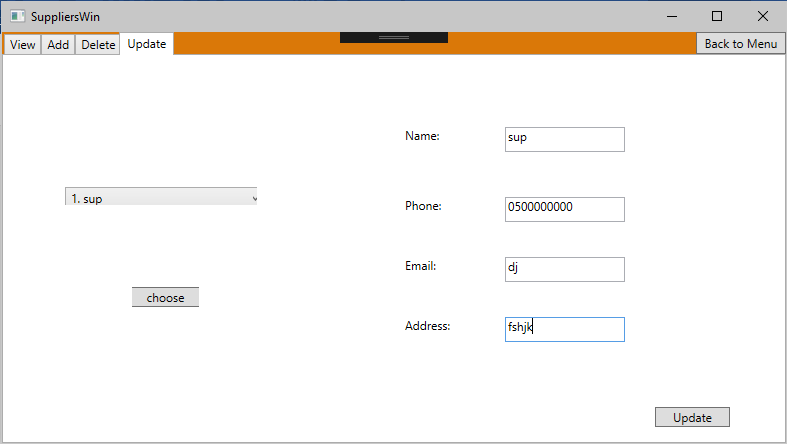


#### הסבר על הדף:

דף זה הוא דף הוספת ספק חדש ובו ניתן להוסיף את פרטי הספק לבסיס הנתונים ע"י לחיצה על כפתור הוסף.

### עדכון פרטי הספק:

#### תצלום מסך:

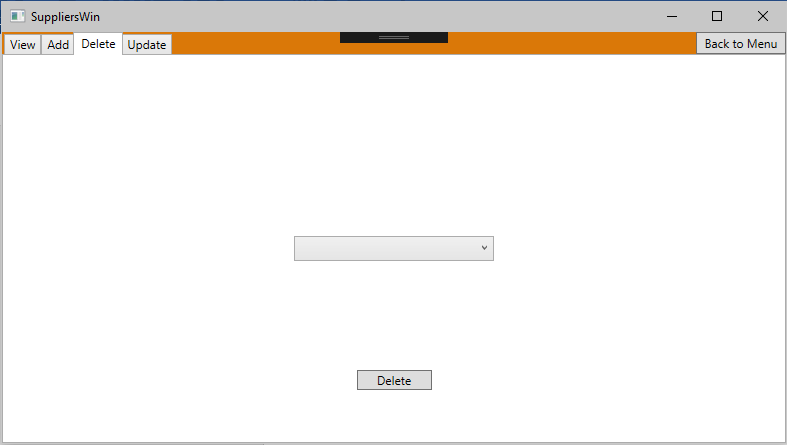


#### הסבר על הדף:

בדף זה ניתן לצפות בפרטי הספקים ואף לעדכנם בדרך נוחה וקלה לשימוש.

### מחיקת ספקים:

#### תצלום מסך:

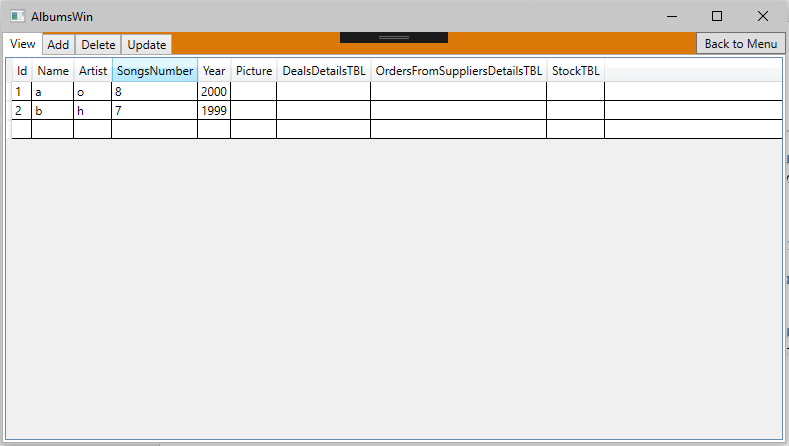


#### הסבר על הדף:

דף זה מאפשר גישה נוחה וקלה לפרטי הספקים שנמצאים בבסיס הנתונים של המערכת.

## תהליך ניהול אלבומים:

#### תצלום מסך:

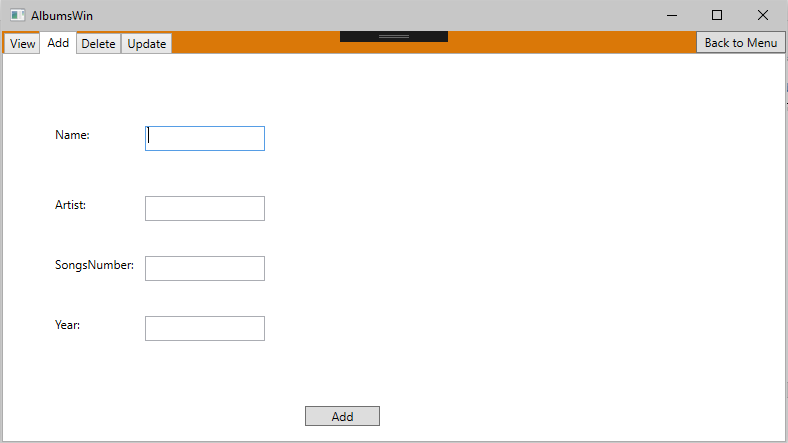


#### הסבר על הדף:

דף זה הוא דף האלבומים. דרכו ניתן לראות את רשימת האלבומים.

### הוספת אלבום:

#### תצלום מסך:

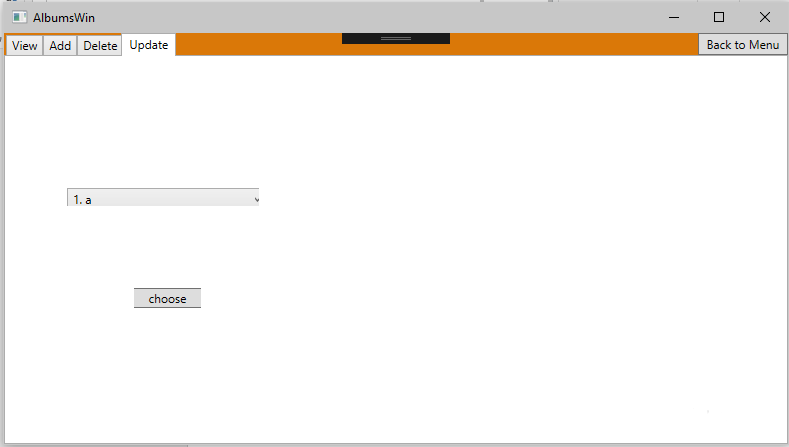


#### הסבר על הדף:

בדף זה ניתן להוסיף יחידה לקטגוריה ע"י כתיבת הפרטים ולחיצה על הכפתור "הוספה".

### עדכון אלבום:

#### תצלום מסך:

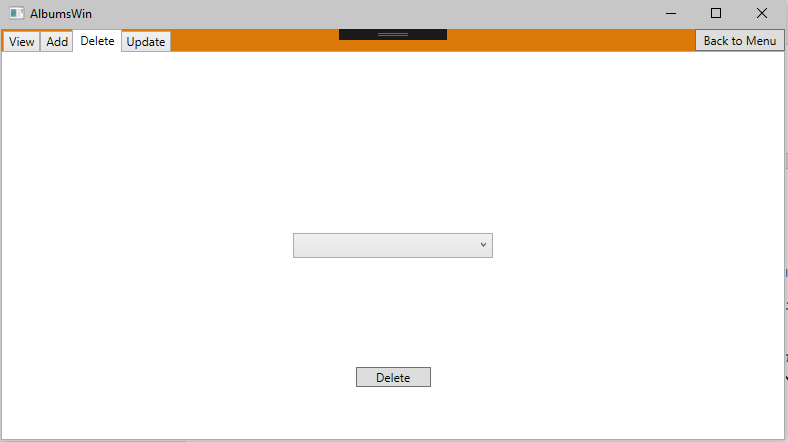


#### הסבר על הדף:

דף זה מטרתו להציג את פרטי היחידות מקטגוריה ולאפשר למשתמש לעדכן אותם בצורה נוחה ע"י לחיצה על כפתור "עדכון".

### מחיקת אלבום:

#### תצלום מסך:

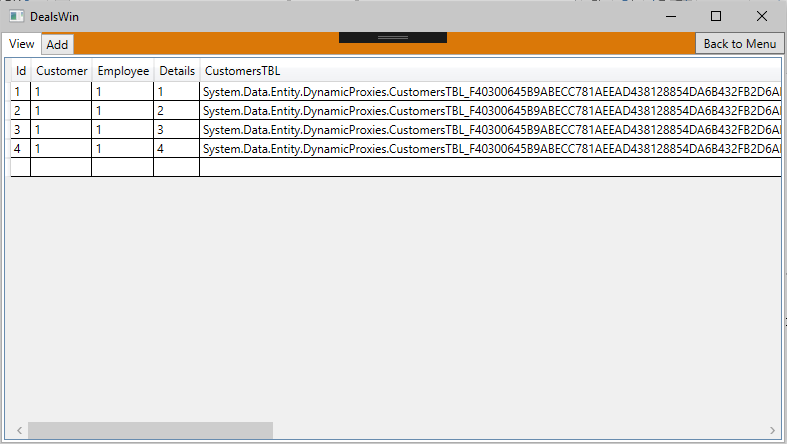


#### הסבר על הדף:

ניתן למחוק אלבום על ידי בחירת אלבום מהמערכת ולחיצה על כפתור מחק.

## תהליך ניהול עסקה:

#### תצלום מסך:

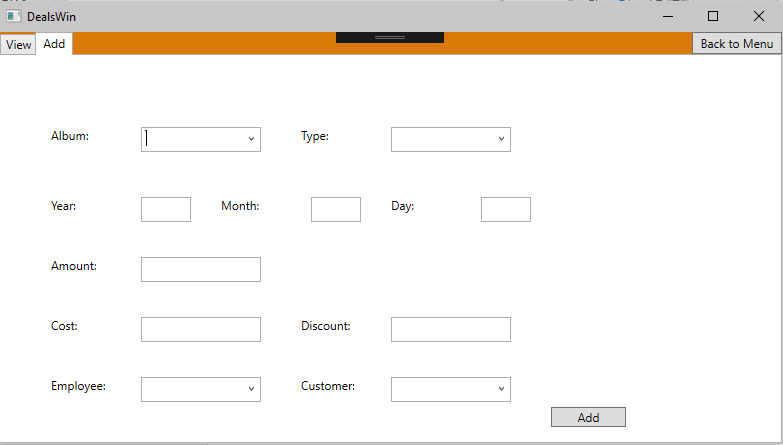


#### הסבר על הדף:

דף זה מאפשר גישה נוחה וקלה לפרטי העסקאות שנמצאות בבסיס הנתונים של המערכת.

### הוספת עסקה:

#### תצלום מסך:

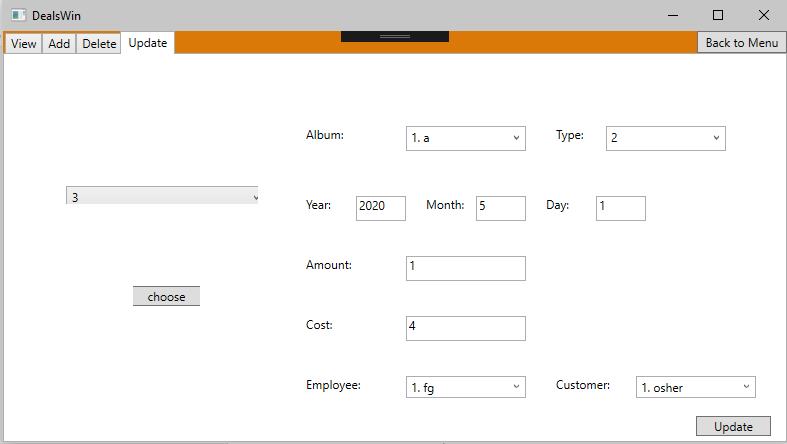
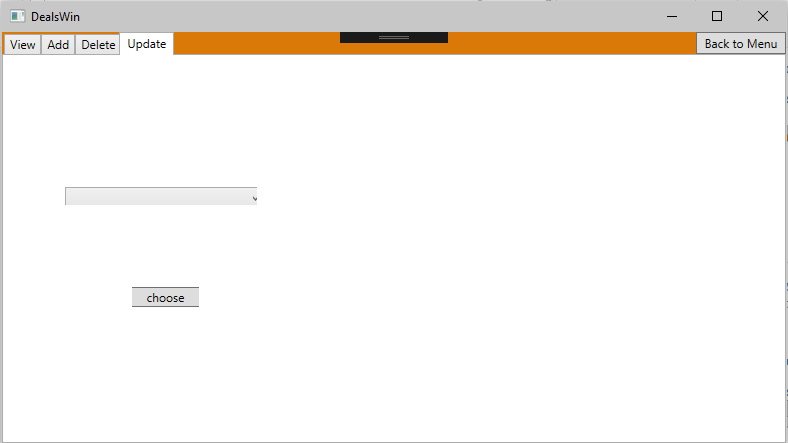


#### הסבר על הדף:

דף זה מאפשר למשתמש ליצור עסקה של לקוח.

### עדכון עסקה:

#### תצלום מסך:

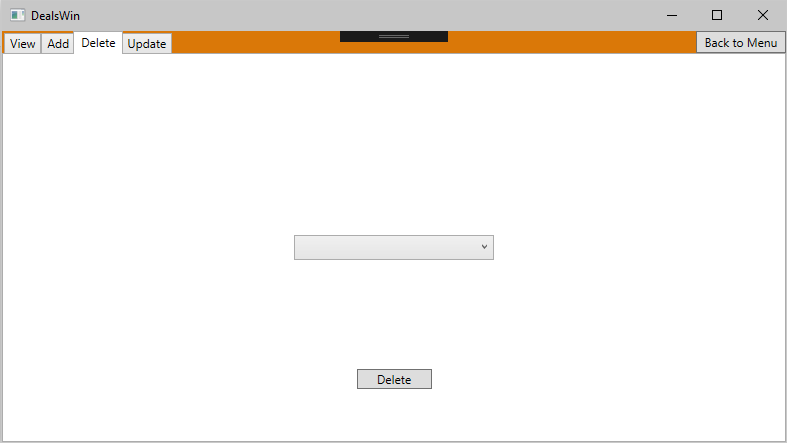
   


#### הסבר על הדף:

דף זה מציג את פרטי העסקה שנשמרו במערכת ומאפשר לעדכנם.

### מחיקת עסקה:

#### תצלום מסך:

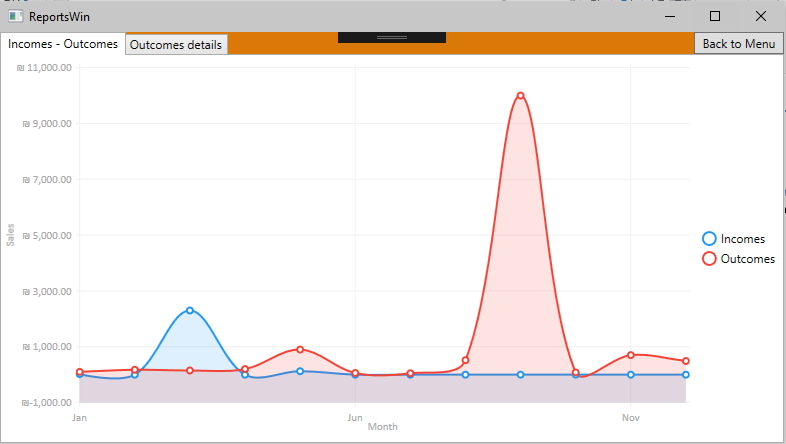


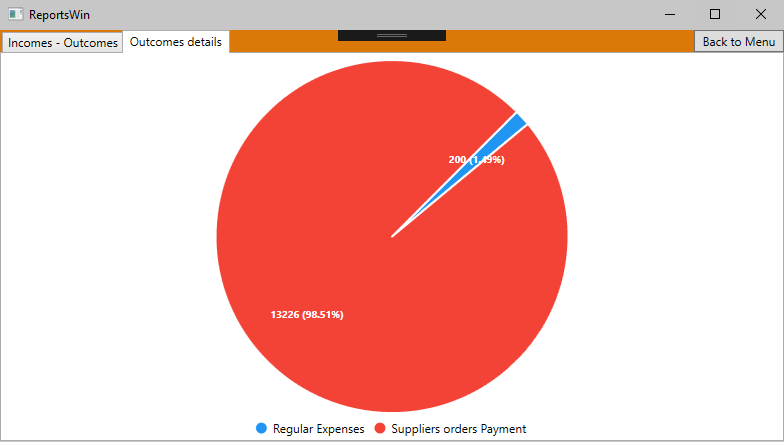
#### הסבר על הדף:

זהו דף בו ניתן לבחור עסקה הנמצאת במערכת ולמחוק אותה באמצעות לחיצה על כפתור "delete".

## תהליך יצירת דוחות:

#### תצלום מסך:



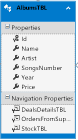


#### הסבר על הדף:

דף זה מראה דוחות הנוצרים בהתאם לנתונים השמורים בתוכנה. ניתן לראות כמה סוגי דוחות המייצגים חישובים שונים

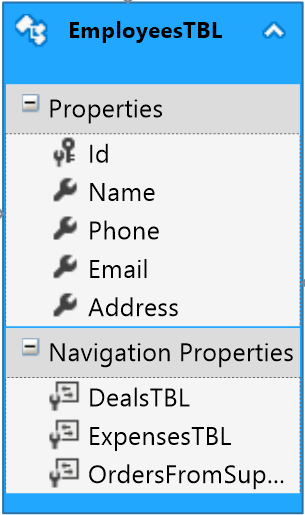
# המדריך למתכנת:

## ישויות וקשרים:



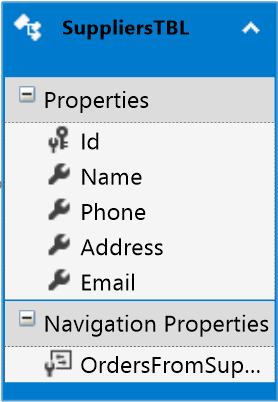
## בסיס נתונים:

### טבלת לקוחות:

* Id – מפתח ראשי, מספר מזהה של האובייקט.
* Name – שם מלא
* Phone – טלפון לתקשורת
* Email – דוא"ל לתקשורת
* Address – כתובת

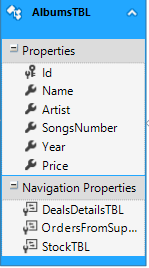
### טבלת עובדים:

* Id – מפתח ראשי, מספר מזה של האובייקט.
* Name – שם מלא
* Phone – טלפון לתקשורת
* Email – דוא"ל לתקשורת
* Address – כתובת



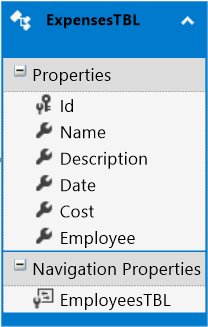
### טבלת ספקים:

* Id – מפתח ראשי, מספר מזהה של האובייקט.
* Name – שם מלא
* Phone – טלפון לתקשורת
* Email – דוא"ל לתקשורת
* Address – כתובת



### טבלת אלבומים:

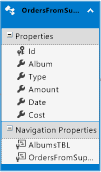
* Id – מפתח ראשי, מספר מזהה של האובייקט.
* Name – שם
* Artist – זמר האלבום
* SongsNumber – מספר השירים באלבום
* Year – שנת הוצאת האלבום
* Pirice – מחיר האלבום



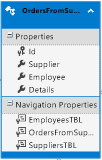
### טבלת הוצאות:

* Id – מפתח ראשי, מספר מזהה של האובייקט.
* Name – שם
* Description – תיאור על מה התבזבז הכסף
* Date – תאריך (יום חודש שנה)
* Cost – מחיר
* Employee – העובד שהיה אחראי על ההוצאה, קישור לטבלת עובדים.

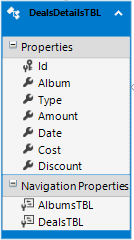
### טבלת מלאי:

* Id – מפתח ראשי, מספר מזה של האובייקט
* Album – אלבום (קישור לטבלת אלבומים)
* Type – הסוג החומרי של האלבום (דיסק תקליט קסטה)
* Amount – כמות הפעמים שמוצר זה נמצא במלאי

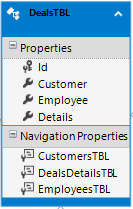
### טבלת פרטי הזמנות מספקים:

* Id – מפתח ראשי, מספר מזה של האובייקט
* Album – אלבום (קישור לטבלת אלבומים)
* Type – הסוג החומרי של האלבום (דיסק תקליט קסטה)
* Amount – כמות הפעמים שמזמינים מוצר זה
* Date – תאריך (יום חודש שנה)
* Cost – מחיר

### טבלת הזמנות מספקים:

* Id – מפתח ראשי, מספר מזה של האובייקט
* Supplier – ספק (קישור לטבלת ספקים)
* Employee – עובד (קישור לטבלת עובדים)
* Details – פרטי ההזמנה (קישור לטבלת פרטי הזמנות מספקים)

### טבלת פרטי עסקאות:

* Id – מפתח ראשי, מספר מזה של האובייקט
* Album – אלבום (קישור לטבלת אלבומים)
* Type – הסוג החומרי של האלבום (דיסק תקליט קסטה)
* Amount – כמות המוצרים מסוג זה שהלקוח קונה
* Date – תאריך (יום חודש שנה)
* Cost – מחיר
* Discount – אחוז הנחה על הקניה

### טבלת עסקאות:

* Id – מפתח ראשי, מספר מזה של האובייקט
* Customer – עובד (קישור לטבלת עובדים)
* Employee – עובד (קישור לטבלת עובדים)
* Details – פרטי ההזמנה (קישור לטבלת פרטי עסקאות)

## תרשים זרימת מידע – תהליכים מרכזיים:

### ניהול לקוחות:

הוספת פרטי לקוח לטבלה

עדכון הנתונים בטבלה

טבלת לקוחות

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

### ניהול עובדים:

הוספת פרטי עובד לטבלה

עדכון הנתונים בטבלה

טבלת עובדים

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

### ניהול ספקים:

הוספת פרטי ספק לטבלה

עדכון הנתונים בטבלה

טבלת ספקים

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

### ניהול אלבומים:

הוספת פרטי אלבום לטבלה

עדכון הנתונים בטבלה

טבלת אלבומים

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

### ניהול הוצאות שוטפות:

טבלת הוצאות שוטפות

הוספת פרטי הוצאה לטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

טבלת עובדים

### ניהול מלאי:

טבלת מלאי

שליפת נתונים מהטבלה

שליפת נתונים מהטבלה

טבלת אלבומים

### ניהול הזמנות מספקים:

טבלת הזמנות מספקים

שליפת נתונים מהטבלה

הוספת פרטי הזמנה לטבלה

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלאות

שליפת נתונים מהטבלאות

שליפת נתונים מהטבלאות

טבלת פרטי הזמנות מספקים

טבלת ספקים

טבלת עובדים

שליפת נתונים מהטבלה + הוספת פרטי פרטי ההזמנה

טבלת אלבומים

### ניהול עסקאות:

טבלת עסקאות

שליפת נתונים מהטבלה

הוספת פרטי הזמנה לטבלה

שליפת נתונים מהטבלה

עדכון הנתונים בטבלה

עדכון הנתונים בטבלה

שליפת נתונים מהטבלה

שליפת נתונים מהטבלאות

שליפת נתונים מהטבלאות

שליפת נתונים מהטבלאות

טבלת פרטי עסקאות

טבלת לקוחות

טבלת עובדים

שליפת נתונים מהטבלה + הוספת פרטי פרטי ההזמנה

טבלת אלבומים

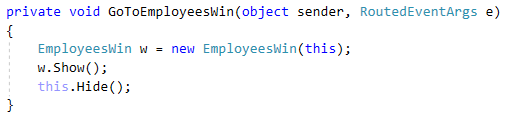
### יצירת דוחות:

כל טבלאות בסיס הנתונים

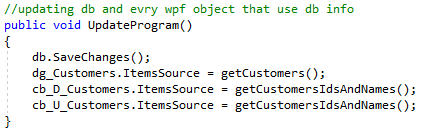
שליפת נתונים

## הסבר על קוד התוכנה:

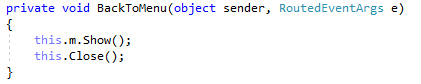
#### פונקציות שחוזרות על עצמן במהלך הקוד:



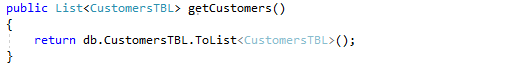
פונקציה זו אחראית על פתיחת חלון מסויים האחראי על תהליך כלשהו. ישנם 9 פונקציות כאלה המכילות חלונות שונים אך המבנה של 9 הפונקציות הוא אותו מבנה. בדוגמא הנ"ל הפונקציה הקשורה לחלון ניהול העובדים.



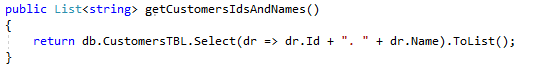
פינקציה זו מופיעה בכל תהליך ניהול (חוץ מדוחות) ומה שהיא עושה זה בעצם ראשית לשמור את השינויים שהתרחשו בממסד נתונים כתוצאה מפעולה מסויימת בתהליך ניהול מסויים, ואז לעדכן את כל התיבות המכילות רשימות שמתבססות על נתונים מהממסד נתונים.



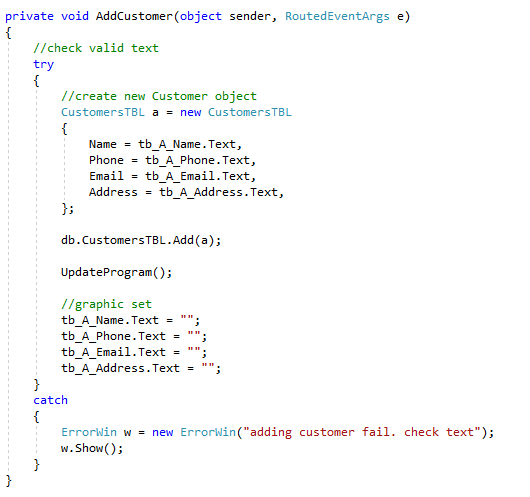
פונקציה זו זהה בכל החלונות (חוץ מהחלון הראשי) והיא קשורה לכפור המחזיר את המשתמש לחלון הבית. תוכן הפונקציה אומר שהיא פותחת שוב את החלון הראשי שהיה מוחבא (לא סגור!), וסוגרת לגמרי את החלון הנוכחי.



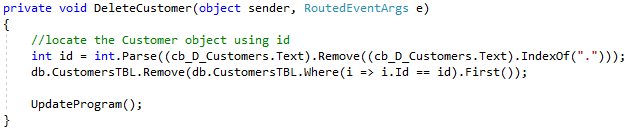
פונקציה זו אחראית להוציא מטבלה מסויימת את כל השורות עם כל המשתנים וליצור מהם רשימה.



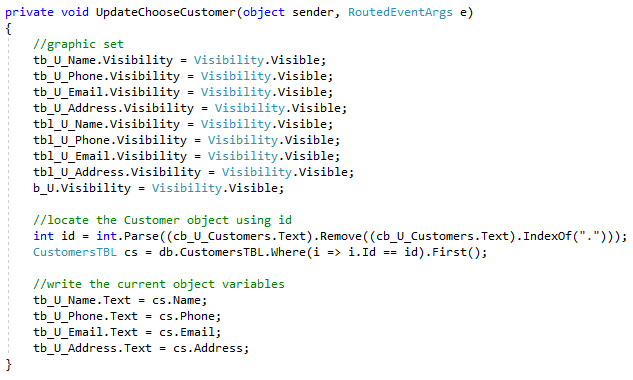
פונקצייה זו אחרית ליצור רשימה של מחרוזות מטבלה מסויימת כאשר המחרוזת מורכבת מהמפתח הראשי – ID ולעיתים גם מעוד מאפיין (בדוגמא הנ"ל – השם).



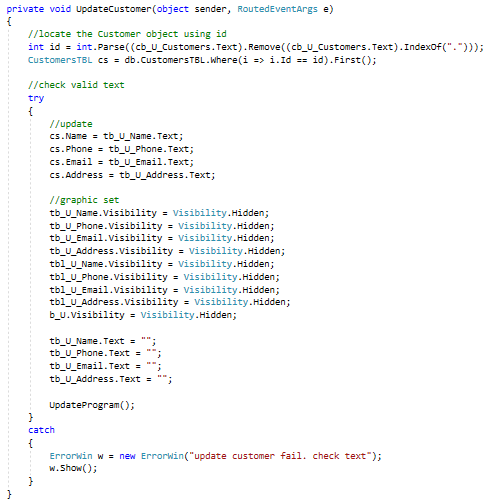
פונקציית ההוספה. לכל תהליך ניהול (חוץ מהדוחות) ישנה פונקציית הוספה. בכל תהליך הפונקצייה מכילה לעיתים חישובים טיפה שונים ושורות קוד נוספות אך התהליך הכללי המשותף לכולם בפונקציית ההוספה הוא: יצירת אובייקט מסוג הטבלה הרלוונטית לאותו תהליך באמצעות הטקסט שמילא המשתמש -> הוספת משתנה זה לממסד נתונים -> עדכון הממסד נתונים -> ניקוי תיבות הטקסט אותן מילא הלקוח.



פונקצייה זו היא פונציית המחיקה, היא מופיעה בכל תהליך ניהול (חוץ מדוחות) כאשר אופן הפעולה שלה הוא ראשית לבודד את הID של המוצר משאר הטקסט שהוא מופיע בו בחלון עצמו ובעזרת הID הזה למצוא את האובייקט המבוקש ולהסיר אותו



פונקצייה זו נמצאת בכל תהליך ניהול (חוץ מהדוחות) ואחראית על בחירת האובייקט אותו נרצה לעדכן. הפונקצייה מכילה לעיתים חישובים טיפה שונים ושורות קוד נוספות אך התהליך הכללי המשותף לכולם בפונקצייה זו הוא: חשיפת תיבות הטקסט אותן המשתמש יכול לערוך -> מציאת האובייקט באמצעות הID שבחר המשתמש בחלון -> באמצעות האובייקט, רשימת כל המאפיינים הנוכחיים שלו בתיבות הטקסט אותן יכול המשתמש לשנות (כמאפיינים דיפולטיים).



פונקציית העדכון. נמצאת בכל תהליך ניהול (חוץ מהדוחות) ומכילה לעיתים חישובים טיפה שונים ושורות קוד נוספות אך התהליך הכללי המשותף לכולם בפונקצייה זו הוא: מציאת האובייקט באמצעות הID שבחר המשתמש בחלון -> עדכון המאפיינים של אותו אובייקט באמצעות המשתנה ובעמצאות הנתונים שמילא המשתמש בתיבות הטקסט -> הסתרה מחודשת של תיבות הטקסט אותן המשתמש יכול לערוך -> איפוס תיבות הטקסט אותן יכול המשתמש לשנות -> שמירת הנתונים המחודשים בממסד.

### דף ראשי:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

//the main window contaun buttens that open every window responsible for spesific prosses in the project

DataBaseEntities db = new DataBaseEntities();

public MainWindow()

{

InitializeComponent();

}

private void GoToEmployeesWin(object sender, RoutedEventArgs e)

{

EmployeesWin w = new EmployeesWin(this);

w.Show();

this.Hide();

}

private void GoToCustomersWin(object sender, RoutedEventArgs e)

{

CustomersWin w = new CustomersWin(this);

w.Show();

this.Hide();

}

private void GoToStockWin(object sender, RoutedEventArgs e)

{

StockWin w = new StockWin(this);

w.Show();

this.Hide();

}

private void GoToSuppliersWin(object sender, RoutedEventArgs e)

{

SuppliersWin w = new SuppliersWin(this);

w.Show();

this.Hide();

}

private void GoToAlbusWin(object sender, RoutedEventArgs e)

{

AlbumsWin w = new AlbumsWin(this);

w.Show();

this.Hide();

}

private void GoToExpensesWin(object sender, RoutedEventArgs e)

{

ExpensesWin w = new ExpensesWin(this);

w.Show();

this.Hide();

}

private void GoToOrdersFromSuppliersWin(object sender, RoutedEventArgs e)

{

OrdersFromSuppliersWin w = new OrdersFromSuppliersWin(this);

w.Show();

this.Hide();

}

private void GoToDealsWin(object sender, RoutedEventArgs e)

{

DealsWin w = new DealsWin(this);

w.Show();

this.Hide();

}

private void GoToReportsWin(object sender, RoutedEventArgs e)

{

ReportsWin w = new ReportsWin(this);

w.Show();

this.Hide();

}

}

}

### ניהול לקוחות:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for CustomersWin.xaml

/// </summary>

public partial class CustomersWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public CustomersWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void AddCustomer(object sender, RoutedEventArgs e)

{

//check valid text

try

{

//create new Customer object

CustomersTBL a = new CustomersTBL

{

Name = tb\_A\_Name.Text,

Phone = tb\_A\_Phone.Text,

Email = tb\_A\_Email.Text,

Address = tb\_A\_Address.Text,

};

db.CustomersTBL.Add(a);

UpdateProgram();

//graphic set

tb\_A\_Name.Text = "";

tb\_A\_Phone.Text = "";

tb\_A\_Email.Text = "";

tb\_A\_Address.Text = "";

}

catch

{

ErrorWin w = new ErrorWin("adding customer fail. check text");

w.Show();

}

}

private void DeleteCustomer(object sender, RoutedEventArgs e)

{

//locate the Customer object using id

int id = int.Parse((cb\_D\_Customers.Text).Remove((cb\_D\_Customers.Text).IndexOf(".")));

db.CustomersTBL.Remove(db.CustomersTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseCustomer(object sender, RoutedEventArgs e)

{

//graphic set

tb\_U\_Name.Visibility = Visibility.Visible;

tb\_U\_Phone.Visibility = Visibility.Visible;

tb\_U\_Email.Visibility = Visibility.Visible;

tb\_U\_Address.Visibility = Visibility.Visible;

tbl\_U\_Name.Visibility = Visibility.Visible;

tbl\_U\_Phone.Visibility = Visibility.Visible;

tbl\_U\_Email.Visibility = Visibility.Visible;

tbl\_U\_Address.Visibility = Visibility.Visible;

b\_U.Visibility = Visibility.Visible;

//locate the Customer object using id

int id = int.Parse((cb\_U\_Customers.Text).Remove((cb\_U\_Customers.Text).IndexOf(".")));

CustomersTBL cs = db.CustomersTBL.Where(i => i.Id == id).First();

//write the current object variables

tb\_U\_Name.Text = cs.Name;

tb\_U\_Phone.Text = cs.Phone;

tb\_U\_Email.Text = cs.Email;

tb\_U\_Address.Text = cs.Address;

}

private void UpdateCustomer(object sender, RoutedEventArgs e)

{

//locate the Customer object using id

int id = int.Parse((cb\_U\_Customers.Text).Remove((cb\_U\_Customers.Text).IndexOf(".")));

CustomersTBL cs = db.CustomersTBL.Where(i => i.Id == id).First();

//check valid text

try

{

//update

cs.Name = tb\_U\_Name.Text;

cs.Phone = tb\_U\_Phone.Text;

cs.Email = tb\_U\_Email.Text;

cs.Address = tb\_U\_Address.Text;

//graphic set

tb\_U\_Name.Visibility = Visibility.Hidden;

tb\_U\_Phone.Visibility = Visibility.Hidden;

tb\_U\_Email.Visibility = Visibility.Hidden;

tb\_U\_Address.Visibility = Visibility.Hidden;

tbl\_U\_Name.Visibility = Visibility.Hidden;

tbl\_U\_Phone.Visibility = Visibility.Hidden;

tbl\_U\_Email.Visibility = Visibility.Hidden;

tbl\_U\_Address.Visibility = Visibility.Hidden;

b\_U.Visibility = Visibility.Hidden;

tb\_U\_Name.Text = "";

tb\_U\_Phone.Text = "";

tb\_U\_Email.Text = "";

tb\_U\_Address.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("update customer fail. check text");

w.Show();

}

}

public List<CustomersTBL> getCustomers()

{

return db.CustomersTBL.ToList<CustomersTBL>();

}

public List<string> getCustomersIdsAndNames()

{

return db.CustomersTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use db info

public void UpdateProgram()

{

db.SaveChanges();

dg\_Customers.ItemsSource = getCustomers();

cb\_D\_Customers.ItemsSource = getCustomersIdsAndNames();

cb\_U\_Customers.ItemsSource = getCustomersIdsAndNames();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול עובדים:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for EmployeesWin.xaml

/// </summary>

public partial class EmployeesWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public EmployeesWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void AddEmployee(object sender, RoutedEventArgs e)

{

try {

//creating new object

EmployeesTBL a = new EmployeesTBL

{

Name = tb\_A\_Name.Text,

Phone = tb\_A\_Phone.Text,

Email = tb\_A\_Email.Text,

Address = tb\_A\_Address.Text,

};

db.EmployeesTBL.Add(a);

UpdateProgram();

//graphic set

tb\_A\_Name.Text = "";

tb\_A\_Phone.Text = "";

tb\_A\_Email.Text = "";

tb\_A\_Address.Text = "";

}

catch

{

ErrorWin w = new ErrorWin("adding fail.");

}

}

private void DeleteEmployee(object sender, RoutedEventArgs e)

{

//locate the object using id + remove from db

int id = int.Parse((cb\_D\_Employees.Text).Remove((cb\_D\_Employees.Text).IndexOf(".")));

db.EmployeesTBL.Remove(db.EmployeesTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseEmployee(object sender, RoutedEventArgs e)

{

//graphic set

tb\_U\_Name.Visibility = Visibility.Visible;

tb\_U\_Phone.Visibility = Visibility.Visible;

tb\_U\_Email.Visibility = Visibility.Visible;

tb\_U\_Address.Visibility = Visibility.Visible;

tbl\_U\_Name.Visibility = Visibility.Visible;

tbl\_U\_Phone.Visibility = Visibility.Visible;

tbl\_U\_Email.Visibility = Visibility.Visible;

tbl\_U\_Address.Visibility = Visibility.Visible;

int id = int.Parse((cb\_U\_Employees.Text).Remove((cb\_U\_Employees.Text).IndexOf(".")));

EmployeesTBL em = db.EmployeesTBL.Where(i => i.Id == id).First();

tb\_U\_Name.Text = em.Name;

tb\_U\_Phone.Text = em.Phone;

tb\_U\_Email.Text = em.Email;

tb\_U\_Address.Text = em.Address;

}

private void UpdateEmployee(object sender, RoutedEventArgs e)

{

try

{

int id = int.Parse((cb\_U\_Employees.Text).Remove((cb\_U\_Employees.Text).IndexOf(".")));

EmployeesTBL em = db.EmployeesTBL.Where(i => i.Id == id).First();

em.Name = tb\_U\_Name.Text;

em.Phone = tb\_U\_Phone.Text;

em.Email = tb\_U\_Email.Text;

em.Address = tb\_U\_Address.Text;

//graphic set

tb\_U\_Name.Visibility = Visibility.Hidden;

tb\_U\_Phone.Visibility = Visibility.Hidden;

tb\_U\_Email.Visibility = Visibility.Hidden;

tb\_U\_Address.Visibility = Visibility.Hidden;

tbl\_U\_Name.Visibility = Visibility.Hidden;

tbl\_U\_Phone.Visibility = Visibility.Hidden;

tbl\_U\_Email.Visibility = Visibility.Hidden;

tbl\_U\_Address.Visibility = Visibility.Hidden;

tb\_U\_Name.Text = "";

tb\_U\_Phone.Text = "";

tb\_U\_Email.Text = "";

tb\_U\_Address.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("update fail.");

}

}

public List<EmployeesTBL> getEmployees()

{

return db.EmployeesTBL.ToList<EmployeesTBL>();

}

public List<string> getEmployeesIdsAndNames()

{

return db.EmployeesTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use

public void UpdateProgram()

{

db.SaveChanges();

dg\_Employees.ItemsSource = getEmployees();

cb\_D\_Employees.ItemsSource = getEmployeesIdsAndNames();

cb\_U\_Employees.ItemsSource = getEmployeesIdsAndNames();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול ספקים:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for SuppliersWin.xaml

/// </summary>

public partial class SuppliersWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public SuppliersWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void AddSupplier(object sender, RoutedEventArgs e)

{

try

{

//create new object

SuppliersTBL a = new SuppliersTBL

{

Name = tb\_A\_Name.Text,

Phone = tb\_A\_Phone.Text,

Email = tb\_A\_Email.Text,

Address = tb\_A\_Address.Text,

};

db.SuppliersTBL.Add(a);

//graphic set

tb\_A\_Name.Text = "";

tb\_A\_Phone.Text = "";

tb\_A\_Email.Text = "";

tb\_A\_Address.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("fail adding supplier.");

}

}

private void DeleteSupplier(object sender, RoutedEventArgs e)

{

//locate the object using id + remove it

int id = int.Parse((cb\_D\_Suppliers.Text).Remove((cb\_D\_Suppliers.Text).IndexOf(".")));

db.SuppliersTBL.Remove(db.SuppliersTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseSupplier(object sender, RoutedEventArgs e)

{

//graphic set

tb\_U\_Name.Visibility = Visibility.Visible;

tb\_U\_Phone.Visibility = Visibility.Visible;

tb\_U\_Email.Visibility = Visibility.Visible;

tb\_U\_Address.Visibility = Visibility.Visible;

tbl\_U\_Name.Visibility = Visibility.Visible;

tbl\_U\_Phone.Visibility = Visibility.Visible;

tbl\_U\_Email.Visibility = Visibility.Visible;

tbl\_U\_Address.Visibility = Visibility.Visible;

//locate the object using id

int id = int.Parse((cb\_U\_Suppliers.Text).Remove((cb\_U\_Suppliers.Text).IndexOf(".")));

SuppliersTBL cs = db.SuppliersTBL.Where(i => i.Id == id).First();

//write the current object variables

tb\_A\_Name.Text = cs.Name;

tb\_U\_Phone.Text = cs.Phone;

tb\_U\_Email.Text = cs.Email;

tb\_U\_Address.Text = cs.Address;

}

private void UpdateSupplier(object sender, RoutedEventArgs e)

{

try

{

//locate the object using id

int id = int.Parse((cb\_U\_Suppliers.Text).Remove((cb\_U\_Suppliers.Text).IndexOf(".")));

SuppliersTBL cs = db.SuppliersTBL.Where(i => i.Id == id).First();

//update

cs.Name = tb\_U\_Name.Text;

cs.Phone = tb\_U\_Phone.Text;

cs.Email = tb\_U\_Email.Text;

cs.Address = tb\_U\_Address.Text;

//graphic set

tb\_U\_Name.Visibility = Visibility.Hidden;

tb\_U\_Phone.Visibility = Visibility.Hidden;

tb\_U\_Email.Visibility = Visibility.Hidden;

tb\_U\_Address.Visibility = Visibility.Hidden;

tbl\_U\_Name.Visibility = Visibility.Hidden;

tbl\_U\_Phone.Visibility = Visibility.Hidden;

tbl\_U\_Email.Visibility = Visibility.Hidden;

tbl\_U\_Address.Visibility = Visibility.Hidden;

tb\_A\_Name.Text = "";

tb\_U\_Phone.Text = "";

tb\_U\_Email.Text = "";

tb\_U\_Address.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("fail supplier update.");

}

}

public List<SuppliersTBL> getSuppliers()

{

return db.SuppliersTBL.ToList<SuppliersTBL>();

}

public List<string> getSuppliersIdsAndNames()

{

return db.SuppliersTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use db info

public void UpdateProgram()

{

db.SaveChanges();

dg\_Suppliers.ItemsSource = getSuppliers();

cb\_D\_Suppliers.ItemsSource = getSuppliersIdsAndNames();

cb\_U\_Suppliers.ItemsSource = getSuppliersIdsAndNames();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול הוצאות שוטפות:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for ExpensesWin.xaml

/// </summary>

public partial class ExpensesWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public ExpensesWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void AddExpense(object sender, RoutedEventArgs e)

{

try

{

DateTime dt = new DateTime(int.Parse(tb\_A\_Year.Text), int.Parse(tb\_A\_Month.Text), int.Parse(tb\_A\_Day.Text));

//create new object

ExpensesTBL a = new ExpensesTBL

{

Name = tb\_A\_Name.Text,

Date = dt,

Description = tb\_A\_Description.Text,

Cost = int.Parse(tb\_A\_Cost.Text),

Employee = int.Parse((cb\_A\_Employee.Text).Remove((cb\_A\_Employee.Text).IndexOf("."))),

};

db.ExpensesTBL.Add(a);

//graphic set

tb\_A\_Name.Text = "";

tb\_A\_Year.Text = "";

tb\_A\_Month.Text = "";

tb\_A\_Day.Text = "";

tb\_A\_Description.Text = "";

tb\_A\_Cost.Text = "";

cb\_A\_Employee.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("adding expens fail.");

}

}

private void DeleteExpense(object sender, RoutedEventArgs e)

{

//locate the object using id

int id = int.Parse((cb\_D\_Expenses.Text).Remove((cb\_D\_Expenses.Text).IndexOf(".")));

db.ExpensesTBL.Remove(db.ExpensesTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseExpense(object sender, RoutedEventArgs e)

{

try

{

//graphic set

tb\_U\_Name.Visibility = Visibility.Visible;

tb\_U\_Year.Visibility = Visibility.Visible;

tb\_U\_Month.Visibility = Visibility.Visible;

tb\_U\_Day.Visibility = Visibility.Visible;

tb\_U\_Description.Visibility = Visibility.Visible;

tb\_U\_Cost.Visibility = Visibility.Visible;

tbl\_U\_Name.Visibility = Visibility.Visible;

tbl\_U\_Year.Visibility = Visibility.Visible;

tbl\_U\_Month.Visibility = Visibility.Visible;

tbl\_U\_Day.Visibility = Visibility.Visible;

tbl\_U\_Description.Visibility = Visibility.Visible;

tbl\_U\_Cost.Visibility = Visibility.Visible;

b\_U.Visibility = Visibility.Visible;

cb\_U\_Employee.Visibility = Visibility.Visible;

//locate the object using id

int id = int.Parse((cb\_U\_Expenses.Text).Remove((cb\_U\_Expenses.Text).IndexOf(".")));

ExpensesTBL cs = db.ExpensesTBL.Where(i => i.Id == id).First();

EmployeesTBL em = db.EmployeesTBL.Where(i => i.Id == cs.Employee).First();

//write the current object variables

tb\_U\_Name.Text = cs.Name;

tb\_U\_Year.Text = cs.Date.Year.ToString();

tb\_U\_Month.Text = cs.Date.Month.ToString();

tb\_U\_Day.Text = cs.Date.Day.ToString();

tb\_U\_Description.Text = cs.Description;

tb\_U\_Cost.Text = cs.Cost.ToString();

cb\_U\_Employee.Text = cs.Employee + ". " + em.Name;

}

catch

{

ErrorWin w = new ErrorWin("expens update choose fail.");

}

}

private void UpdateExpense(object sender, RoutedEventArgs e)

{

try

{

//locate the object using id

int id = int.Parse((cb\_U\_Expenses.Text).Remove((cb\_U\_Expenses.Text).IndexOf(".")));

ExpensesTBL cs = db.ExpensesTBL.Where(i => i.Id == id).First();

DateTime dt = new DateTime(int.Parse(tb\_U\_Year.Text), int.Parse(tb\_U\_Month.Text), int.Parse(tb\_U\_Day.Text));

//updating

cs.Name = tb\_U\_Name.Text;

cs.Date = dt;

cs.Description = tb\_U\_Description.Text;

cs.Cost = int.Parse(tb\_U\_Cost.Text);

cs.Employee = int.Parse((cb\_U\_Employee.Text).Remove((cb\_U\_Employee.Text).IndexOf(".")));

//graphic set

tb\_U\_Name.Visibility = Visibility.Hidden;

tb\_U\_Year.Visibility = Visibility.Hidden;

tb\_U\_Month.Visibility = Visibility.Hidden;

tb\_U\_Day.Visibility = Visibility.Hidden;

tb\_U\_Description.Visibility = Visibility.Hidden;

tb\_U\_Cost.Visibility = Visibility.Hidden;

tbl\_U\_Name.Visibility = Visibility.Hidden;

tbl\_U\_Year.Visibility = Visibility.Hidden;

tbl\_U\_Month.Visibility = Visibility.Hidden;

tbl\_U\_Day.Visibility = Visibility.Hidden;

tbl\_U\_Description.Visibility = Visibility.Hidden;

tbl\_U\_Cost.Visibility = Visibility.Hidden;

b\_U.Visibility = Visibility.Hidden;

cb\_U\_Employee.Visibility = Visibility.Hidden;

tb\_U\_Name.Text = "";

tb\_U\_Year.Text = "";

tb\_U\_Month.Text = "";

tb\_U\_Day.Text = "";

tb\_U\_Description.Text = "";

tb\_U\_Cost.Text = "";

cb\_U\_Employee.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("expens update fail.");

}

}

public List<ExpensesTBL> getExpenses()

{

return db.ExpensesTBL.ToList<ExpensesTBL>();

}

public List<string> getExpensesIdsAndNames()

{

return db.ExpensesTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public List<string> getEmployeesIdsAndNames()

{

return db.EmployeesTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use db info

public void UpdateProgram()

{

db.SaveChanges();

cb\_A\_Employee.ItemsSource = getEmployeesIdsAndNames();

cb\_U\_Employee.ItemsSource = getEmployeesIdsAndNames();

dg\_Expenses.ItemsSource = getExpenses();

cb\_D\_Expenses.ItemsSource = getExpensesIdsAndNames();

cb\_U\_Expenses.ItemsSource = getExpensesIdsAndNames();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול מלאי:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for StockWin.xaml

/// </summary>

public partial class StockWin : Window

{

//the stock shipts when there is a new deal - products deleted,

//or when there is ane order from supplier - add products.

//so there is no option for user to edit or update stock.

//only view

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public StockWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

public List<StockTBL> getStock()

{

return db.StockTBL.ToList<StockTBL>();

}

public void UpdateProgram()

{

dg\_Stock.ItemsSource = getStock();

}

}

}

### ניהול אלבומים:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for AlbumsWin.xaml

/// </summary>

public partial class AlbumsWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public AlbumsWin(MainWindow m)

{

InitializeComponent();

this.m = m;

UpdateProgram();

}

private void AddAlbum(object sender, RoutedEventArgs e)

{

try

{

//create new Album object

AlbumsTBL a = new AlbumsTBL

{

Name = tb\_A\_Name.Text,

Artist = tb\_A\_Artist.Text,

SongsNumber = int.Parse(tb\_A\_SongsNumber.Text),

Year = int.Parse(tb\_A\_Year.Text),

Price = int.Parse(tb\_A\_Price.Text),

};

db.AlbumsTBL.Add(a);

UpdateProgram();

//graphic set

tb\_A\_Name.Text = "";

tb\_A\_Artist.Text = "";

tb\_A\_SongsNumber.Text = "";

tb\_A\_Year.Text = "";

tb\_A\_Price.Text = "";

}

catch

{

ErrorWin w = new ErrorWin("Album adding fail. check text");

w.Show();

}

}

private void DeleteAlbum(object sender, RoutedEventArgs e)

{

//locate the Album object using id

int id = int.Parse((cb\_D\_Albums.Text).Remove((cb\_D\_Albums.Text).IndexOf(".")));

db.AlbumsTBL.Remove(db.AlbumsTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseAlbum(object sender, RoutedEventArgs e)

{

//graphic set

tb\_U\_Name.Visibility = Visibility.Visible;

tb\_U\_Artist.Visibility = Visibility.Visible;

tb\_U\_SongsNumber.Visibility = Visibility.Visible;

tb\_U\_Year.Visibility = Visibility.Visible;

tb\_U\_Price.Visibility = Visibility.Visible;

tbl\_U\_Name.Visibility = Visibility.Visible;

tbl\_U\_Artist.Visibility = Visibility.Visible;

tbl\_U\_SongsNumber.Visibility = Visibility.Visible;

tbl\_U\_Year.Visibility = Visibility.Visible;

tbl\_U\_Price.Visibility = Visibility.Visible;

b\_U.Visibility = Visibility.Visible;

//locate the Album object using id

int id = int.Parse((cb\_U\_Albums.Text).Remove((cb\_U\_Albums.Text).IndexOf(".")));

AlbumsTBL cs = db.AlbumsTBL.Where(i => i.Id == id).First();

//write the current object variables

tb\_U\_Name.Text = cs.Name;

tb\_U\_Artist.Text = cs.Artist;

tb\_U\_SongsNumber.Text = cs.SongsNumber.ToString();

tb\_U\_Year.Text = cs.Year.ToString();

tb\_U\_Price.Text = cs.Price.ToString();

}

private void UpdateAlbum(object sender, RoutedEventArgs e)

{

//locate the Album object using id

int id = int.Parse((cb\_U\_Albums.Text).Remove((cb\_U\_Albums.Text).IndexOf(".")));

AlbumsTBL cs = db.AlbumsTBL.Where(i => i.Id == id).First();

try

{

//update

cs.Name = tb\_U\_Name.Text;

cs.Artist = tb\_U\_Artist.Text;

cs.SongsNumber = int.Parse(tb\_U\_SongsNumber.Text);

cs.Year = int.Parse(tb\_U\_Year.Text);

cs.Price = int.Parse(tb\_U\_Price.Text);

//graphic set

tb\_U\_Name.Visibility = Visibility.Hidden;

tb\_U\_Artist.Visibility = Visibility.Hidden;

tb\_U\_SongsNumber.Visibility = Visibility.Hidden;

tb\_U\_Year.Visibility = Visibility.Hidden;

tb\_U\_Price.Visibility = Visibility.Hidden;

tbl\_U\_Name.Visibility = Visibility.Hidden;

tbl\_U\_Artist.Visibility = Visibility.Hidden;

tbl\_U\_SongsNumber.Visibility = Visibility.Hidden;

tbl\_U\_Year.Visibility = Visibility.Hidden;

tbl\_U\_Price.Visibility = Visibility.Hidden;

b\_U.Visibility = Visibility.Hidden;

tb\_U\_Name.Text = "";

tb\_U\_Artist.Text = "";

tb\_U\_SongsNumber.Text = "";

tb\_U\_Year.Text = "";

tb\_U\_Price.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("update album fail. check text");

w.Show();

}

}

public List<AlbumsTBL> getAlbums()

{

return db.AlbumsTBL.ToList<AlbumsTBL>();

}

public List<string> getAlbumsIdsAndNames()

{

return db.AlbumsTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use db info

public void UpdateProgram()

{

db.SaveChanges();

dg\_Albums.ItemsSource = getAlbums();

cb\_D\_Albums.ItemsSource = getAlbumsIdsAndNames();

cb\_U\_Albums.ItemsSource = getAlbumsIdsAndNames();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול עסקאות:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for DealsWin.xaml

/// </summary>

public partial class DealsWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public DealsWin(MainWindow m)

{

InitializeComponent();

this.m = m;

//list of all albums types

List<string> lst = new List<string>();

lst.Add("Disk");

lst.Add("Record");

lst.Add("Cassette");

cb\_A\_Type.ItemsSource = lst;

cb\_U\_Type.ItemsSource = lst;

UpdateProgram();

}

private void AddDeal(object sender, RoutedEventArgs e)

{

try

{

DateTime dt = new DateTime(int.Parse(tb\_A\_Year.Text), int.Parse(tb\_A\_Month.Text), int.Parse(tb\_A\_Day.Text));

//albume type to int

int t = 0;

if (cb\_A\_Type.Text == "Disk")

{

t = 1;

}

else if (cb\_A\_Type.Text == "Record")

{

t = 2;

}

else if (cb\_A\_Type.Text == "Cassette")

{

t = 3;

}

//check if the stock allow the deal

int alId = int.Parse((cb\_A\_Album.Text).Remove((cb\_A\_Album.Text).IndexOf(".")));

int a = int.Parse(tb\_A\_Amount.Text);

if ((db.StockTBL.Where(i => i.Album == alId && i.Type == t && i.Amount >= a)).Any())

{

AlbumsTBL alb = db.AlbumsTBL.Where(i => i.Id == alId).First();

//price calculation

int c = alb.Price \* t \* int.Parse(tb\_A\_Amount.Text);

if (tb\_A\_Discount.Text != "" && int.Parse(tb\_A\_Discount.Text) > 0)

{

c = c - ((c \* (int.Parse(tb\_A\_Discount.Text))) / 100);

if (c < 0)

{

c = 0;

}

}

Console.WriteLine(c);

//create new deal (+ details) object

DealsDetailsTBL b = new DealsDetailsTBL

{

Album = int.Parse((cb\_A\_Album.Text).Remove((cb\_A\_Album.Text).IndexOf("."))),

Type = t,

Amount = int.Parse(tb\_A\_Amount.Text),

Date = dt,

Cost = c,

Discount = int.Parse(tb\_A\_Discount.Text),

};

db.DealsDetailsTBL.Add(b);

db.SaveChanges();

//create new object

DealsTBL d = new DealsTBL

{

Customer = int.Parse((cb\_A\_Customer.Text).Remove((cb\_A\_Customer.Text).IndexOf("."))),

Employee = int.Parse((cb\_A\_Employee.Text).Remove((cb\_A\_Employee.Text).IndexOf("."))),

Details = b.Id,

};

db.DealsTBL.Add(d);

db.SaveChanges();

//update stock

StockTBL s = db.StockTBL.Where(i => i.Album == alId && i.Type == t).First();

s.Amount -= int.Parse(tb\_A\_Amount.Text);

db.SaveChanges();

}

//graphic set

cb\_A\_Album.Text = "";

cb\_A\_Type.Text = "";

tb\_A\_Amount.Text = "";

tb\_A\_Year.Text = "";

tb\_A\_Month.Text = "";

tb\_A\_Day.Text = "";

tb\_A\_Discount.Text = "";

cb\_A\_Employee.Text = "";

cb\_A\_Customer.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("adding fail. check text");

w.Show();

}

}

private void DeleteDeal(object sender, RoutedEventArgs e)

{

//locate object using id

int id = int.Parse(cb\_D\_Deals.Text);

db.DealsTBL.Remove(db.DealsTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseDeal(object sender, RoutedEventArgs e)

{

try

{

//graphic set

cb\_U\_Album.Visibility = Visibility.Visible;

cb\_U\_Type.Visibility = Visibility.Visible;

tb\_U\_Amount.Visibility = Visibility.Visible;

tb\_U\_Year.Visibility = Visibility.Visible;

tb\_U\_Month.Visibility = Visibility.Visible;

tb\_U\_Day.Visibility = Visibility.Visible;

tb\_U\_Discount.Visibility = Visibility.Visible;

cb\_U\_Employee.Visibility = Visibility.Visible;

cb\_U\_Customer.Visibility = Visibility.Visible;

tbl\_U\_Album.Visibility = Visibility.Visible;

tbl\_U\_Type.Visibility = Visibility.Visible;

tbl\_U\_Amount.Visibility = Visibility.Visible;

tbl\_U\_Year.Visibility = Visibility.Visible;

tbl\_U\_Month.Visibility = Visibility.Visible;

tbl\_U\_Day.Visibility = Visibility.Visible;

tbl\_U\_Discount.Visibility = Visibility.Visible;

tbl\_U\_Employee.Visibility = Visibility.Visible;

tbl\_U\_Customer.Visibility = Visibility.Visible;

b\_U.Visibility = Visibility.Visible;

//locate the object using id

int id = int.Parse(cb\_U\_Deals.Text);

DealsTBL cs = db.DealsTBL.Where(i => i.Id == id).First();

DealsDetailsTBL csD = db.DealsDetailsTBL.Where(i => i.Id == cs.Details).First();

EmployeesTBL em = db.EmployeesTBL.Where(i => i.Id == cs.Employee).First();

CustomersTBL c = db.CustomersTBL.Where(i => i.Id == cs.Customer).First();

AlbumsTBL a = db.AlbumsTBL.Where(i => i.Id == csD.Album).First();

//write the current object variables

cb\_U\_Album.Text = csD.Album + ". " + a.Name;

cb\_U\_Type.Text = csD.Type.ToString();

tb\_U\_Amount.Text = csD.Amount.ToString();

tb\_U\_Year.Text = csD.Date.Year.ToString();

tb\_U\_Month.Text = csD.Date.Month.ToString();

tb\_U\_Day.Text = csD.Date.Day.ToString();

tb\_U\_Discount.Text = csD.Discount.ToString();

cb\_U\_Employee.Text = cs.Employee + ". " + em.Name;

cb\_U\_Customer.Text = cs.Customer + ". " + c.Name;

}

catch

{

ErrorWin w = new ErrorWin("deal update choose fail.");

}

}

private void UpdateDeal(object sender, RoutedEventArgs e)

{

try

{

DateTime dt = new DateTime(int.Parse(tb\_U\_Year.Text), int.Parse(tb\_U\_Month.Text), int.Parse(tb\_U\_Day.Text));

//locate object using id

int id = int.Parse(cb\_U\_Deals.Text);

DealsTBL cs = db.DealsTBL.Where(i => i.Id == id).First();

DealsDetailsTBL csD = db.DealsDetailsTBL.Where(i => i.Id == cs.Details).First();

int t = 0;

if (cb\_U\_Type.Text == "Disk")

{

t = 1;

}

else if (cb\_U\_Type.Text == "Record")

{

t = 2;

}

else if (cb\_U\_Type.Text == "Cassette")

{

t = 3;

}

else

{

t = int.Parse(cb\_U\_Type.Text);

}

//update

csD.Album = int.Parse((cb\_U\_Album.Text).Remove((cb\_U\_Album.Text).IndexOf(".")));

csD.Type = t;

csD.Amount = int.Parse(tb\_U\_Amount.Text);

csD.Date = dt;

csD.Discount = int.Parse(tb\_U\_Discount.Text);

cs.Customer = int.Parse((cb\_U\_Customer.Text).Remove((cb\_U\_Customer.Text).IndexOf(".")));

cs.Employee = int.Parse((cb\_U\_Employee.Text).Remove((cb\_U\_Employee.Text).IndexOf(".")));

int alId = int.Parse((cb\_U\_Album.Text).Remove((cb\_U\_Album.Text).IndexOf(".")));

AlbumsTBL alb = db.AlbumsTBL.Where(i => i.Id == alId).First();

//price calculation

int c = alb.Price \* t \* int.Parse(tb\_U\_Amount.Text);

if (tb\_U\_Discount.Text != "" && int.Parse(tb\_U\_Discount.Text) > 0)

{

c = c - (c \* (int.Parse(tb\_U\_Discount.Text)) / 100);

if (c < 0)

{

c = 0;

}

}

csD.Cost = c;

//graphic set

cb\_U\_Album.Visibility = Visibility.Hidden;

cb\_U\_Type.Visibility = Visibility.Hidden;

tb\_U\_Amount.Visibility = Visibility.Hidden;

tb\_U\_Year.Visibility = Visibility.Hidden;

tb\_U\_Month.Visibility = Visibility.Hidden;

tb\_U\_Day.Visibility = Visibility.Hidden;

tb\_U\_Discount.Visibility = Visibility.Hidden;

cb\_U\_Employee.Visibility = Visibility.Hidden;

cb\_U\_Customer.Visibility = Visibility.Hidden;

tbl\_U\_Album.Visibility = Visibility.Hidden;

tbl\_U\_Type.Visibility = Visibility.Hidden;

tbl\_U\_Amount.Visibility = Visibility.Hidden;

tbl\_U\_Year.Visibility = Visibility.Hidden;

tbl\_U\_Month.Visibility = Visibility.Hidden;

tbl\_U\_Day.Visibility = Visibility.Hidden;

tbl\_U\_Discount.Visibility = Visibility.Hidden;

tbl\_U\_Employee.Visibility = Visibility.Hidden;

tbl\_U\_Customer.Visibility = Visibility.Hidden;

b\_U.Visibility = Visibility.Hidden;

cb\_U\_Album.Text = "";

cb\_U\_Type.Text = "";

tb\_U\_Amount.Text = "";

tb\_U\_Year.Text = "";

tb\_U\_Month.Text = "";

tb\_U\_Day.Text = "";

tb\_U\_Discount.Text = "";

cb\_U\_Employee.Text = "";

cb\_U\_Customer.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("update fail.");

}

}

public List<DealsTBL> getDeals()

{

return db.DealsTBL.ToList<DealsTBL>();

}

public List<int> getDealsIds()

{

return db.DealsTBL.Select(dr => dr.Id).ToList();

}

public List<string> getEmployeesIdsAndNames()

{

return db.EmployeesTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public List<string> getAlbumsIdsAndNames()

{

return db.AlbumsTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public List<string> getCustomersIdsAndNames()

{

return db.CustomersTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public void UpdateProgram()

{

db.SaveChanges();

dg\_Deals.ItemsSource = getDeals();

cb\_A\_Employee.ItemsSource = getEmployeesIdsAndNames();

cb\_U\_Employee.ItemsSource = getEmployeesIdsAndNames();

cb\_A\_Album.ItemsSource = getAlbumsIdsAndNames();

cb\_U\_Album.ItemsSource = getAlbumsIdsAndNames();

cb\_A\_Customer.ItemsSource = getCustomersIdsAndNames();

cb\_U\_Customer.ItemsSource = getCustomersIdsAndNames();

cb\_D\_Deals.ItemsSource = getDealsIds();

cb\_U\_Deals.ItemsSource = getDealsIds();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול הזמנות מספק:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for OrdersFromSuppliersWin.xaml

/// </summary>

public partial class OrdersFromSuppliersWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public OrdersFromSuppliersWin(MainWindow m)

{

InitializeComponent();

this.m = m;

//list of all albums types

List<string> lst = new List<string>();

lst.Add("record");

lst.Add("disk");

lst.Add("cassette");

cb\_A\_Type.ItemsSource = lst;

cb\_U\_Type.ItemsSource = lst;

UpdateProgram();

}

private void AddOrderFromSupplier(object sender, RoutedEventArgs e)

{

try

{

DateTime dt = new DateTime(int.Parse(tb\_A\_Year.Text), int.Parse(tb\_A\_Month.Text), int.Parse(tb\_A\_Day.Text));

//turn type to int

int t = 0;

if (cb\_A\_Type.Text == "record")

{

t = 1;

}

else if (cb\_A\_Type.Text == "disk")

{

t = 2;

}

else if (cb\_A\_Type.Text == "cassette")

{

t = 3;

}

//create new object - detais

OrdersFromSuppliersDetailsTBL b = new OrdersFromSuppliersDetailsTBL

{

Album = int.Parse((cb\_A\_Album.Text).Remove((cb\_A\_Album.Text).IndexOf("."))),

Type = t,

Amount = int.Parse(tb\_A\_Amount.Text),

Date = dt,

Cost = int.Parse(tb\_A\_Cost.Text),

};

db.OrdersFromSuppliersDetailsTBL.Add(b);

db.SaveChanges();

//create new object

OrdersFromSuppliersTBL a = new OrdersFromSuppliersTBL

{

Supplier = int.Parse((cb\_A\_Supplier.Text).Remove((cb\_A\_Supplier.Text).IndexOf("."))),

Employee = int.Parse((cb\_A\_Employee.Text).Remove((cb\_A\_Employee.Text).IndexOf("."))),

Details = b.Id,

};

db.OrdersFromSuppliersTBL.Add(a);

db.SaveChanges();

//check if the products exsist - if true, chant the product amount / if false, create new product

if (db.StockTBL.Where(i => i.Album == b.Album && i.Type == b.Type).Any())

{

(db.StockTBL.Where(i => i.Album == b.Album && i.Type == b.Type).First()).Amount += b.Amount;

}

else

{

StockTBL s = new StockTBL

{

Album = b.Album,

Type = b.Type,

Amount = b.Amount,

};

db.StockTBL.Add(s);

}

UpdateProgram();

//graphic set

cb\_A\_Album.Text = "";

cb\_A\_Type.Text = "";

tb\_A\_Amount.Text = "";

tb\_A\_Year.Text = "";

tb\_A\_Month.Text = "";

tb\_A\_Day.Text = "";

tb\_A\_Cost.Text = "";

cb\_A\_Employee.Text = "";

cb\_A\_Supplier.Text = "";

}

catch

{

ErrorWin w = new ErrorWin("add order fail.");

}

}

private void DeleteOrderFromSupplier(object sender, RoutedEventArgs e)

{

//locate the object using id

int id = int.Parse(cb\_D\_OrdersFromSuppliers.Text);

db.OrdersFromSuppliersTBL.Remove(db.OrdersFromSuppliersTBL.Where(i => i.Id == id).First());

UpdateProgram();

}

private void UpdateChooseOrderFromSupplier(object sender, RoutedEventArgs e)

{

try

{

//graphic set

cb\_U\_Album.Visibility = Visibility.Visible;

cb\_U\_Type.Visibility = Visibility.Visible;

tb\_U\_Amount.Visibility = Visibility.Visible;

tb\_U\_Year.Visibility = Visibility.Visible;

tb\_U\_Month.Visibility = Visibility.Visible;

tb\_U\_Day.Visibility = Visibility.Visible;

tb\_U\_Cost.Visibility = Visibility.Visible;

cb\_U\_Employee.Visibility = Visibility.Visible;

cb\_U\_Supplier.Visibility = Visibility.Visible;

tbl\_U\_Album.Visibility = Visibility.Visible;

tbl\_U\_Type.Visibility = Visibility.Visible;

tbl\_U\_Amount.Visibility = Visibility.Visible;

tbl\_U\_Year.Visibility = Visibility.Visible;

tbl\_U\_Month.Visibility = Visibility.Visible;

tbl\_U\_Day.Visibility = Visibility.Visible;

tbl\_U\_Cost.Visibility = Visibility.Visible;

tbl\_U\_Employee.Visibility = Visibility.Visible;

tbl\_U\_Supplier.Visibility = Visibility.Visible;

b\_U.Visibility = Visibility.Visible;

//locate the object using id

int id = int.Parse(cb\_U\_OrdersFromSuppliers.Text);

OrdersFromSuppliersTBL cs = db.OrdersFromSuppliersTBL.Where(i => i.Id == id).First();

OrdersFromSuppliersDetailsTBL csD = db.OrdersFromSuppliersDetailsTBL.Where(i => i.Id == cs.Details).First();

EmployeesTBL em = db.EmployeesTBL.Where(i => i.Id == cs.Employee).First();

SuppliersTBL s = db.SuppliersTBL.Where(i => i.Id == cs.Supplier).First();

AlbumsTBL a = db.AlbumsTBL.Where(i => i.Id == csD.Album).First();

//write the current object variables

cb\_U\_Album.Text = csD.Album + ". " + a.Name;

cb\_U\_Type.Text = csD.Type.ToString();

tb\_U\_Amount.Text = csD.Amount.ToString();

tb\_U\_Year.Text = csD.Date.Year.ToString();

tb\_U\_Month.Text = csD.Date.Month.ToString();

tb\_U\_Day.Text = csD.Date.Day.ToString();

tb\_U\_Cost.Text = csD.Cost.ToString();

cb\_U\_Employee.Text = cs.Employee + ". " + em.Name;

cb\_U\_Supplier.Text = cs.Supplier + ". " + s.Name;

}

catch

{

ErrorWin w = new ErrorWin("choose update order fail.");

}

}

private void UpdateOrderFromSupplier(object sender, RoutedEventArgs e)

{

try

{

DateTime dt = new DateTime(int.Parse(tb\_U\_Year.Text), int.Parse(tb\_U\_Month.Text), int.Parse(tb\_U\_Day.Text));

//locate the object using id

int id = int.Parse(cb\_U\_OrdersFromSuppliers.Text);

OrdersFromSuppliersTBL cs = db.OrdersFromSuppliersTBL.Where(i => i.Id == id).First();

OrdersFromSuppliersDetailsTBL csD = db.OrdersFromSuppliersDetailsTBL.Where(i => i.Id == cs.Details).First();

//album type to int

int t = 0;

if (cb\_U\_Type.Text == "record")

{

t = 1;

}

else if (cb\_U\_Type.Text == "disk")

{

t = 2;

}

else if (cb\_U\_Type.Text == "cassette")

{

t = 3;

}

else

{

t = int.Parse(cb\_U\_Type.Text);

}

//update

csD.Album = int.Parse((cb\_U\_Album.Text).Remove((cb\_U\_Album.Text).IndexOf(".")));

csD.Type = t;

csD.Amount = int.Parse(tb\_U\_Amount.Text);

csD.Date = dt;

csD.Cost = int.Parse(tb\_U\_Cost.Text);

cs.Supplier = int.Parse((cb\_U\_Supplier.Text).Remove((cb\_U\_Supplier.Text).IndexOf(".")));

cs.Employee = int.Parse((cb\_U\_Employee.Text).Remove((cb\_U\_Employee.Text).IndexOf(".")));

//graphic set

cb\_U\_Album.Visibility = Visibility.Hidden;

cb\_U\_Type.Visibility = Visibility.Hidden;

tb\_U\_Amount.Visibility = Visibility.Hidden;

tb\_U\_Year.Visibility = Visibility.Hidden;

tb\_U\_Month.Visibility = Visibility.Hidden;

tb\_U\_Day.Visibility = Visibility.Hidden;

tb\_U\_Cost.Visibility = Visibility.Hidden;

cb\_U\_Employee.Visibility = Visibility.Hidden;

cb\_U\_Supplier.Visibility = Visibility.Hidden;

tbl\_U\_Album.Visibility = Visibility.Hidden;

tbl\_U\_Type.Visibility = Visibility.Hidden;

tbl\_U\_Amount.Visibility = Visibility.Hidden;

tbl\_U\_Year.Visibility = Visibility.Hidden;

tbl\_U\_Month.Visibility = Visibility.Hidden;

tbl\_U\_Day.Visibility = Visibility.Hidden;

tbl\_U\_Cost.Visibility = Visibility.Hidden;

tbl\_U\_Employee.Visibility = Visibility.Hidden;

tbl\_U\_Supplier.Visibility = Visibility.Hidden;

b\_U.Visibility = Visibility.Hidden;

cb\_U\_Album.Text = "";

cb\_U\_Type.Text = "";

tb\_U\_Amount.Text = "";

tb\_U\_Year.Text = "";

tb\_U\_Month.Text = "";

tb\_U\_Day.Text = "";

tb\_U\_Cost.Text = "";

cb\_U\_Employee.Text = "";

cb\_U\_Supplier.Text = "";

UpdateProgram();

}

catch

{

ErrorWin w = new ErrorWin("choose update order fail.");

}

}

public List<OrdersFromSuppliersTBL> getOrdersFromSuppliers()

{

return db.OrdersFromSuppliersTBL.ToList<OrdersFromSuppliersTBL>();

}

public List<int> getOrdersFromSuppliersIds()

{

return db.OrdersFromSuppliersTBL.Select(dr => dr.Id).ToList();

}

public List<string> getEmployeesIdsAndNames()

{

return db.EmployeesTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public List<string> getAlbumsIdsAndNames()

{

return db.AlbumsTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

public List<string> getSuppliersIdsAndNames()

{

return db.SuppliersTBL.Select(dr => dr.Id + ". " + dr.Name).ToList();

}

//updating db and evry wpf object that use db info

public void UpdateProgram()

{

db.SaveChanges();

dg\_OrdersFromSuppliers.ItemsSource = getOrdersFromSuppliers();

cb\_A\_Employee.ItemsSource = getEmployeesIdsAndNames();

cb\_U\_Employee.ItemsSource = getEmployeesIdsAndNames();

cb\_A\_Album.ItemsSource = getAlbumsIdsAndNames();

cb\_U\_Album.ItemsSource = getAlbumsIdsAndNames();

cb\_A\_Supplier.ItemsSource = getSuppliersIdsAndNames();

cb\_U\_Supplier.ItemsSource = getSuppliersIdsAndNames();

cb\_D\_OrdersFromSuppliers.ItemsSource = getOrdersFromSuppliersIds();

cb\_U\_OrdersFromSuppliers.ItemsSource = getOrdersFromSuppliersIds();

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

}

}

### ניהול דוחות:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using LiveCharts;

using LiveCharts.Wpf;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for ReportsWin.xaml

/// </summary>

public partial class ReportsWin : Window

{

DataBaseEntities db = new DataBaseEntities();

private MainWindow m;

public ReportsWin(MainWindow m)

{

InitializeComponent();

this.m = m;

try

{

IncomesOutcomes();

}

catch

{

ErrorWin w = new ErrorWin("Incomes Outcomes Report fail.");

}

try

{

OutcomesDetails();

}

catch

{

ErrorWin w = new ErrorWin("Outcomes Details Report fail.");

}

}

public void IncomesOutcomes()

{

//calcuklate every income to the store (frome the sales)

int i1 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 1).Sum(i => i.Cost);

int i2 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 2).Sum(i => i.Cost);

int i3 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 3).Sum(i => i.Cost);

int i4 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 4).Sum(i => i.Cost);

int i5 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 5).Sum(i => i.Cost);

int i6 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 6).Sum(i => i.Cost);

int i7 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 7).Sum(i => i.Cost);

int i8 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 8).Sum(i => i.Cost);

int i9 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 9).Sum(i => i.Cost);

int i10 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 10).Sum(i => i.Cost);

int i11 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 11).Sum(i => i.Cost);

int i12 = db.DealsDetailsTBL.ToList().FindAll(i => i.Date.Month == 12).Sum(i => i.Cost);

int o1 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 1).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 1).Sum(i => i.Cost);

int o2 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 2).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 2).Sum(i => i.Cost);

int o3 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 3).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 3).Sum(i => i.Cost);

int o4 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 4).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 4).Sum(i => i.Cost);

int o5 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 5).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 5).Sum(i => i.Cost);

int o6 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 6).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 6).Sum(i => i.Cost);

int o7 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 7).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 7).Sum(i => i.Cost);

int o8 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 8).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 8).Sum(i => i.Cost);

int o9 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 9).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 9).Sum(i => i.Cost);

int o10 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 10).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 10).Sum(i => i.Cost);

int o11 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 11).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 11).Sum(i => i.Cost);

int o12 = db.OrdersFromSuppliersDetailsTBL.ToList().FindAll(i => i.Date.Month == 12).Sum(i => i.Cost) + db.ExpensesTBL.ToList().FindAll(i => i.Date.Month == 12).Sum(i => i.Cost);

SeriesCollection = new SeriesCollection

{

new LineSeries

{

Title = "Incomes",

Values = new ChartValues<int> { i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12 }

},

new LineSeries

{

Title = "Outcomes",

Values = new ChartValues<int> { o1, o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12 }

}

};

Labels = new[] { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

YFormatter = value => value.ToString("C");

DataContext = this;

}

public void OutcomesDetails()

{

int supPay = db.OrdersFromSuppliersDetailsTBL.ToList().Sum(i => i.Cost);

int regExp = db.ExpensesTBL.ToList().Sum(i => i.Cost);

Func<ChartPoint, string> labelPoint = chartPoint => string.Format("{0} ({1:P})", chartPoint.Y, chartPoint.Participation);

pie.Series = new SeriesCollection

{

new PieSeries

{

Title = "Regular Expenses",

Values = new ChartValues<int> { regExp },

DataLabels = true,

LabelPoint = labelPoint

},

new PieSeries

{

Title = "Suppliers orders Payment",

Values = new ChartValues<int> { supPay },

DataLabels = true,

LabelPoint = labelPoint

}

};

pie.LegendLocation = LegendLocation.Bottom;

}

private void BackToMenu(object sender, RoutedEventArgs e)

{

this.m.Show();

this.Close();

}

public SeriesCollection SeriesCollection { get; set; }

public string[] Labels { get; set; }

public Func<double, string> YFormatter { get; set; }

}

}

### חלון שגיאה:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace SivanArazi\_RecordStore

{

/// <summary>

/// Interaction logic for ErrorWin.xaml

/// </summary>

public partial class ErrorWin : Window

{

//evry time there is an error, this window open withe error explenation

public ErrorWin(string s)

{

InitializeComponent();

t.Text = s;

}

}

}

# סיכום אישי:

במהלך עשיית הפרוייקט למדתי המון מתחום המחשבים וגם למדתי המון על עצמי. למדתי שאני לא מוותרת ומתאמצת כדי להגיע לתוצאה הכי טובה שיכולה להיות. בנוסף פיתחתי את יכולת הלמידה העצמית שלי מהאינטרנט וממקורות נוספים אחרים. במהלך השנה לא רק עבדתי על הפרוייקט שלי אלא גם נחשפתי לרעיונות של אחרים וראיתי כיצד תהליכים דומים נעשים בצורה שונה מה שפיתח לי יותר את יכולת המחשבה. העבודה על הפרוייקט הייתה מתישה וארוכה אך בסופו של דבר אני יכולה להגיד שאני מרוצה מהתוצאה ושלמה עם הפרוייקט שלי. אני מאוד מאמינה שבעתיד אזדקק ליכולות שפיתחתי השנה. לסיכום, נהנתי לעבוד על הפרוייקט ואני שלמה איתו וגאה בו ובעצמי.