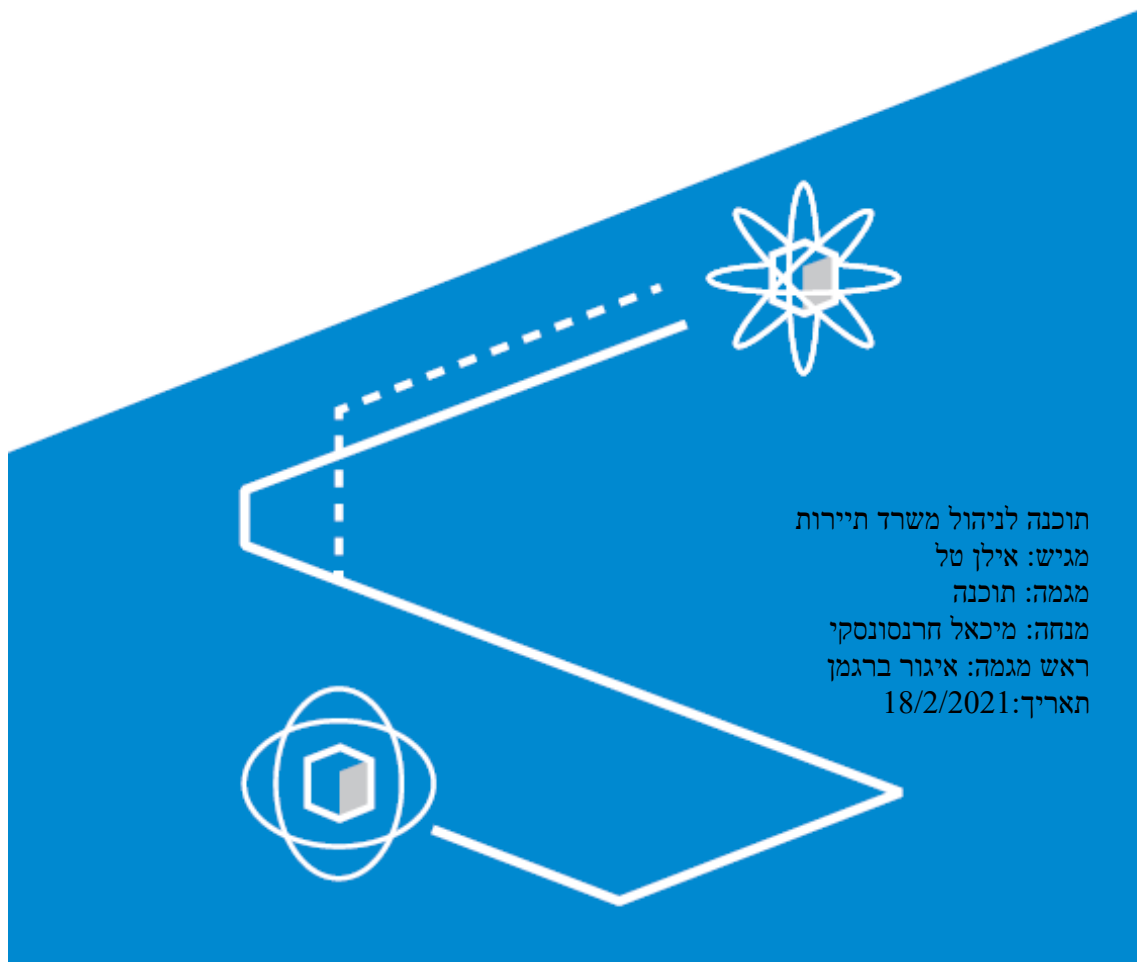


המכללה
הטכנולוגית
באר שבע

בשיתוף WORLD ORT
קדימה מדע



פרויקט גמר



מדור חשבונות סטודנטים ופרויקטים

רחוב בזל 71, ת.ד. 45 באר-שבע 8410001

טל: 08-6462502 | פקס: 08-6462501

gabrielak@tcb.ac.il

1954-ב | www.tcb.ac.il

המכללה
הטכנולוגית
באר שבע

בשיתוף WORLD ORT
קדימה מדע



סימוכין : כלל 2

הצהרת סטודנט

שם הסטודנט ל. ס'ל ת.ז. 312201619

שם המכללה בה לומד הסטודנט : מכללה טכנולוגית באר – שבע קדימה מדע(ע"ר)

אני הח"מ, מצהיר בזאת כי פרויקט הגמר וספר הפרויקט המצ"ב נעשו על ידי בלבד.

פרויקט הגמר נעשה על סמך הנושאים שלמדתי במכללה ובאופן עצמאי.

פרויקט הגמר וספר הפרויקט נעשו על בסיס הנחייתו של המנחה האישי.

מקורות המידע בהם השתמשתי לביצוע פרויקט הגמר מצוינים ברשימת המקורות המצוינים בספר הפרויקט.

אני מודע לאחריות שהנני מקבל על עצמי על ידי חתימתו על הצהרה זו שכל הנאמר בה אמת ורק אמת.

תאריך : 18.02.2021

חתימת הסטודנט : ל. ס'ל

אישור המנחה האישי

הריני מאשר שהפרויקט בוצע בהנחייתי, קראתי את ספר הפרויקט ומצאתי כי הוא

מוכן לצורך הגשת הסטודנט להגנה על פרויקט הגמר.

תאריך : 18.02.2021

שם המנחה : מיכל חרסונסקי חתימה : [Signature]

אישור ראש המגמה

הריני מאשר שספר הפרויקט מוכן לצורך הגשת הסטודנט להגנה על פרויקט הגמר

תאריך : 18.02.2021

שם ראש המגמה : איגור ברגמן חתימה : [Signature]

המכללה
הטכנולוגית
באר שבע

בעיתון WORLD ORT
קדימה מדע



08/06/2020

לכבוד
טל אילן ת.ז. 312201619
אוכמנית 34
באר שבע
א.ג.נ.,

הנדון: אישור נושא הפרויקט

הנני להודיעך כי הצעת הפרויקט בנושא "תוכנה לניהול משרד תיירות"
בהנחיית - מר ברגמן איגור, מר חרסונסקי מיכאל, אשרה ע"י המפקח בתאריך
01/06/2020.

הערות המפקח:

הנחיות לפרויקט ולמבנה פרויקט הגמר ניתן למצוא באתר המכללה באינטרנט,
מגמות – מדור פרויקטים.

בכבוד רב,
מדור פרויקטים

העתק: מנחה
תיק אישי

מדור חשבונות סטודנטים ופרויקטים
רחוב בזל 71, ת.ד. 45 באר-שבע 8410001
טל: 08-6462502 | פקס: 08-6462501
nahrielsk@tcb.ac.il

1954-ב | מוסדה | www.tcb.ac.il

**** יש לקרוא את הנספח להצעת הפרוייקט בקובץ הצעת פרוייקט PDF**

הצעה לפרויקט גמר

א. פרטי הסטודנטים

שם הסטודנט	ת.ז. 9 ספרות	כתובת	טלפון נייד	תאריך סיום הלימודים
אילן טל	312201619	אוכמנית 34	0544216619	

שם המכללה: מכללה טכנולוגית באר שבע סמל המכללה: 72204 מסלול ההכשרה: הנדסאים.
מגמת לימוד: תוכנה מקום ביצוע הפרויקט: מכללה טכנולוגית באר שבע

ב. פרטי המנחה האישי

שם המנחה *	כתובת	טלפון נייד	תואר	מקום עבודה/תפקיד
חרנסונסקי מיכאל	אביתר הכהן 7 באר שבע	0527438238	הנדסת מערכות מידע ממוחשבת תואר שני	מכללה הטכנולוגית מרצה
ברגמן איגור	גרינשפן הרשל 39 באר שבע	0547805384	הנדסת מערכות מידע ממוחשבת תואר שני	מכללה הטכנולוגית מרצה ראש מגמה

* עבור מנחה אישי חדש יש לצרף קורות חיים, ניסיון מקצועי ותעודות השכלה לאישור מה"ט.



חתימת הגורם המקצועי מטעם מה"ט



חתימת המנחה האישי



חתימת הסטודנט

1. שם הפרויקט: תוכנה לניהול משרד תיירות

2. רקע

2.1. תיאור ורקע כללי

תוכנה שמנהלת משרד תיירות בעזרת זימון חבילות למגוון יעדים. פתיחת משתמשים והרשאות. מחיקה/עדכון/יצירת עובדים או לקוחות

2.2. מטרות המערכת

ליצור מערכת מעודכנת של הלקוחות והחבילות שהם לוקחים, הלקוח יכול לראות את החבילות שהוא כבר קנה, ומאגר מעודכן של העובדים והמנהלים. המנהל יכול לראות את העסקאות שעובדים עשו.

3. סקירת מצב קיים בשוק, אילו בעיות קיימות

קיימת בעיה של מאגר נתונים מעודכן, לקוחות לא זוכרים את הטיולים שהם לקחו ואת המחיר שהם שילמו. המנהלים לא יכולים לראות איזה עסקאות ומכירות העובדים שלהם עשו בצורה נוחה.

4. מה הפרויקט אמור לחדש או לשפר

הפרויקט מביא שיפור במערכות הקיימות בכך שללקוח יש את האפשרות להישאר מעודכן בחבילות טיולים שהוא לקח. העובדים יכולים לראות את המידע של הלקוחות למחוק אותם ולהוסיף להם חבילות. מנהלים יכולים לראות את העסקאות שהעובדים שלהם עשו ולמחוק את העובדים שלהם.

5. דרישות מערכת ופונקציונאליות

5.1. דרישות מערכת

סביבת הטמעה ושימוש. שרירות, ביצועים והתמודדות עם עומסים.

שימוש בתוכנה Visual Studio לכתיבת קוד התוכנה וקומפילציה בסביבת עבודה של שפת תוכנה C# המתקשרת לבסיס נתונים של טבלאות עם כל הנתונים. המערכת תותקן בעיקר על מחשבים מהדור הנוכחי.

5.2. דרישות פונקציונאליות

רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך.

משתמש רגיל(לקוח):

- לערוך את המידע של המשתמש שלו.
- להיכנס למערכת בעזרת שם משתמש וסיסמא.
- לקבוע טיולים וחבילות.
- לשלם על טיולים או חבילות.

משתמש עובד(מוכר):

- יכול למחוק לקוחות.
- יכול להוסיף משתמש לקוח.
- יכול לראות מידע של לקוחות.
- לקבוע ללקוחות טיולים וחבילות.
- לערוך את המידע של המשתמש שלו.
- יכול לבטל טיולים וחבילות של לקוחות.
- לקחת תשלום מטיול או לקוח.
- להיכנס למערכת בעזרת שם משתמש וסיסמא.

משתמש מנהל:

- לערוך את המידע של המשתמש שלו.
- להיכנס למערכת בעזרת שם משתמש וסיסמא.
- יכול להוסיף משתמש עובד.
- יכול למחוק עובדים.
- יכול לראות את היסטורית המכירות של כל עובד.

6. בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד):

6.1. תיאור הבעיות- הללו כפועל יוצא של דרישות המשתמש מהתוכנה.

בעיה 1:

לתת מגבלות בין הרשאות של המשתמשים, לעשות הפרדה בין מנהל, עובד, לקוח.

בעיה 2:

כפילויות של פרופילים מידע כמו ת.ז, אימייל, לא יכול להיות זהה לכמה משתמשים.

בעיה 3:

הצפנה של סיסמאות.

6.2. פתרונות אפשריים. (נא ציין פתרונות אפשריים וחלופות ארכיטקטוניות):

פתרון 1:

ניתוב של ההרשאות כך שלכל משתמש יהיה את החלון והאפשרויות שהמשתמש שלו יכול לגשת.

פתרון 2:

כשמוסיפים משתמש או מעדכנים אותו, נבדוק שהמידע לא מופיע כבר במשתמשים אחרים בטבלאות הנתונים

פתרון 3:

נשתמש בהצפנה SHA1 ונשווה בין ההצפנה שהמשתמש מכניס מאשר הסיסמאות עצמם.

7. פתרון טכנולוגי נבחר:

נשתמש בבסיס נתונים שמורכב מטבלאות ושם נבצע את כל הלוגיקה והמבנה של המערכת, סיסמאות ומערכת ההרשאות תיבנה מול מסד הנתונים להרשמה וגישה למערכת אנחנו נשתמש בקשר שבין הדפים למבנה של בסיס נתונים.

7.1. טופולוגית הפתרון:

המערכת תתפרש ל2 חלקים, צד משתמש – המערכת עצמה בנויה מ C#, בסיס נתונים Database.
- כלומר המערכת תתפרש רובה בפיתוח דרך C# וכל מרכיבה, את הנתונים עצמם יהיו מורכבים מטבלאות בנויות בתוך מסדר נתונים התממשקו לתוך המערכת.

7.2. טכנולוגיות בשימוש.(איזה ומדוע בכמה מילים)

טכנולוגיה של C# בשיתוף עם בסיס נתונים Access מבנה העבודה מול C# נוחה מאוד.
עם אפשרות ליצור תוכנה למשתמש ידיוותית ונוחה הן למשתמש והן למפתח.

7.3. שפות הפיתוח:(איזה שפות ומדוע בכמה מילים?)

שפת C# ואקסס, שפת C# נוחה מאוד כדי ליצור תוכנה ב GUI וניתן לבדוק בזמן אמת שאכן הדברים מתפקדים.

7.4. תיאור הארכיטקטורה הנבחרת- הסבר בכמה מילים מדוע

הארכיטקטורה בעיקר נבנתה על דפים של C# וגישה למשתמשים על פי רמת הרשאה של משתמש, כשהכל נבנה תחת בסיס נתונים שיושב נכון לעכשיו במחשב מקומי, ניתן ליצור גיבוי מקומי או דרך שרת בעתיד.

7.5. חלוקה לתוכניות ומודולים

אין חלוקה למודלים הכול עובד מקומי. עתידית ניתן לחלק את המערכת למודלים.

7.6. סביבת השרת (מקומי, וירטואלי, ענן, שירות אירוח)

עבודה מקומית אך הקובץ בסיס הנתונים בעתיד יכול להיות בשרת כולל גיבוי של נתונים. כרגע הבסיס נתונים יהיה מקומית בלבד במחשב.

7.7. ממשק המשתמש/לקוח – GUI

ממשק שעובד דרך סי שארפ בתוכנה Visual Studio.

7.8. ממשקים למערכות אחרות / API:

אין.

7.9. שימוש בחבילות תוכנה:

אין.

8. שימוש במבני נתונים וארגון קבצים

8.1. נא פרט את מבני הנתונים

משתמשים (לקוח עובד מנהל):

- שם משתמש.
- סיסמא.
- הרשאה.
- ת.ז.

חבילות:

- מספר חבילה.
- מס טיסה.
- מחיר.

הזמנות:

- מס הזמנה.
- מס לקוח.
- מס עובד.
- סטטוס האם שולם.

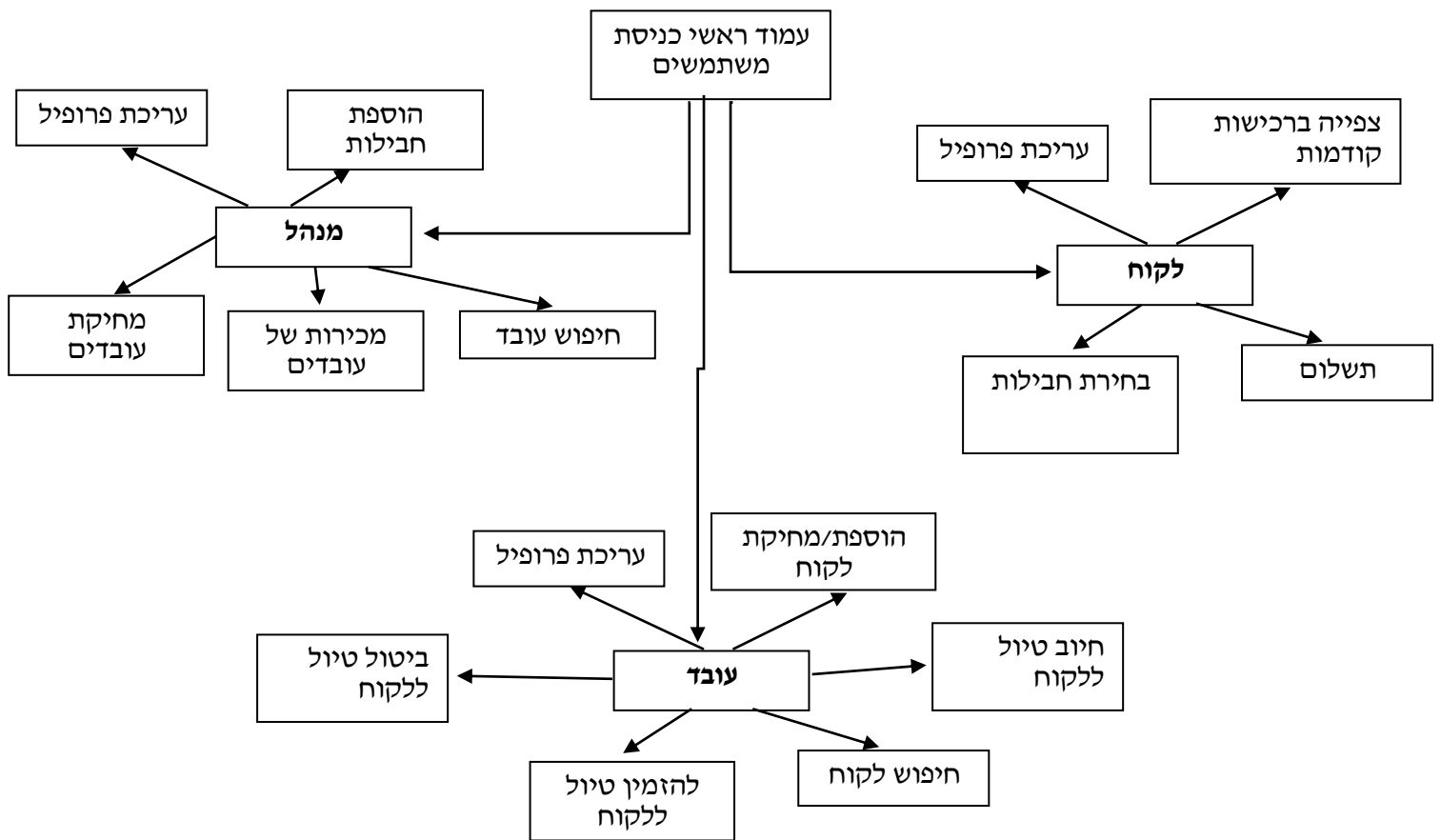
8.2. נא פרט את שיטת האחסון (מאגר, קבצים ובאיזה טכנולוגיה)

אחסון כל הפרטים בבסיס הנתונים Access .

8.3. נא ציין מנגנוני התאוששות מנפילה\קריסה\תמיכה בטראנזקציות.

המערכת מסוגלת לעמוד בעומס עקב ניתוב נתון של המערכת מול מסד הנתונים.

9. תרשימי מערכת מרכזיים



10. תיאור המרכיב האלגוריתמי – חישובי

10.1. תיאור הבעיה : בכניסת המשתמשים למערכת שכל משתמש יקבל שימוש במערכת במסגרת ההגבלות של המשתמש .

פתרון : מבנה לוגי נכון של המערכת וניתוב כל משתמש לדפים שהוא אמור להשתמש ולגשת אליהם.

10.2. איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)

איסוף מידע על לקוחות והעובדים כמה העובדים מוכרים ואיזה טיולים הלקוחות מזמינים.

11. תיאור/התייחסות לנושאי אבטחת מידע

נא לציין אזורים הדורשים אבטחה, כגון : שרת, בקרת גישה לאתר, חשבונות משתמשים, מאגרי מידע וכיצד ניתן מענה.
ניתן לדאוג לאבטח את בסיס הנתונים שבו קיימים רוב הנתונים של המשתמשים וסיסמאות, מאגרי המידע והפרופילים.
נא ציין מס' מקרים ותגובות להם ניתן מענה אבטחתי.
ניתן לאבטח את הבסיס נתונים בעזרת הצפנה של הקובץ שרק למנהל מערכת יש גישה לשם.

12. משאבים הנדרשים לפרויקט:

12.1. 300-350

12.2. ציוד נדרש

• מחשב

12.3. תוכנות נדרשות

• Access

• Visual Studio2017+

12.4. ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט

הרשאות למשתמשים והצפנה של מידע.

12.5. ספרות ומקורות מידע

13. תכנית עבודה ושלבנים למימוש הפרויקט

- יזום הרעיון
- ניתוח המערכת
- ניתוח מבנה הנתונים
- כתיבה לוגית של המערכת
- כתיבת ממשק המערכת
- כתיבת ממשק ה- ADMIN והמזכירה
- עיצוב
- בדיקת תוכנה

14. תכנון הבדיקות שיבוצעו

14.1. נא פרט בטבלה, בדיקות תהליכיות ברמת משתמש בהן נדרשת המערכת לעמוד

(full Flow)

סוג הבדיקה	חשיבות	מקרי הבדיקה	בדיקה
אוטומטית	גבוהה	בדיקה של שם משתמש וסיסמא תקינים	1
אוטומטית	גבוהה	בדיקה האם זה משתמש מנהל, עובד או לקוח	2
אוטומטית	גבוהה	בדיקה האם הסיסמא שהמשתמש מכניס תואם להצפנה של הסיסמא המקורית	3

14.2. נא פרט בטבלה, מס מייצג של בדיקות יחידה למודולים המרכזיים בהן נדרשת

המערכת לעמוד. (unit test)

עדיין לא ידוע, התווספו בדיקות תוך כדי בנייה.

15. בקרת גרסאות (version control) :

- גרסה 1.0 : הכנה סקיצה של מסד נתונים
 גרסה 1.01 : בניית מסד נתונים.
 גרסה 1.02 : תחילת בניית המערכת.
 גרסה 1.03 : תכנות המערכת.
 גרסה 2.0 : הגרסה הסופית של המערכת.



חתימת המנחה האישי




חתימת הסטודנט

ג. הערות ראש המגמה במכללה

מאושר _____

ד. אישור ראש המגמה



שם: ברגמן איגור _____ חתימה: _____ תאריך: 18/05/2020

ה. הערות הגורם המקצועי מטעם מה"ט

ו. אישור הגורם המקצועי מטעם מה"ט

צביקה שטרקמן
1/6/20

שם: _____ חתימה: _____ תאריך: _____

תוכן עניינים

- מבוא
- מסך כניסה
 - מסך כניסה
 - הרשמה בתור לקוח חדש
- לקוח
 - מסך לקוח
 - עדכון מידע
 - הזמנת טיול
 - תשלום על הזמנות
 - היסטורית הזמנות
- עובד
 - מסך עובד
 - עדכון מידע
 - הוספת ארץ
 - חיפוש לקוח
 - הדפסת כרטיסי טיסה
 - הוספת טיול
 - חיפוש טיול
- מנהל
 - מסך מנהל
 - עדכון מידע
 - הוספת עובד
 - חיפוש עובד
- דוחות
 - דוח לקוח
 - דוח עובד
 - קבלה על הזמנה
 - דוח טיול ללקוח
 - דוח טיול לעובד/מנהל
 - כרטיסי טיסה
- ממסד נתונים
- קוד תוכנית
- קוד מחלקות

מבוא

תוכנה "ניהול משרד תיירות", זו תוכנה שמשמשת חברת טיולים שנותנת אפשרות להוציא חבילות טיולים ולראות דוחות שונים.

ללקוחות יש אפשרות להיכנס ולהירשם למערכת בעזרת מחשב שנמצא במשרדי החברה ולהזמין חבילות טיולים ולראות את ההיסטוריה של הרכישות שלהם, להוציא קבלות על הזמנות ששולמו, ולהוציא דף שמראה מידע של טיול שהם יכולים לרכוש.

העובדים יכולים להפסיק/לתת גישה למערכת ללקוחות, להוסיף לקוחות, להוסיף טיולים ולהוסיף תמונות, לחפש לקוחות, להזמין ללקוח טיול, לחייב או לבטל את ההזמנה. העובד יכול להוציא את אותם הטפסים שלקוח יכול להוציא ובנוסף להוציא טופס שמראה מידע של לקוח כולל המידע שלו וכל ההזמנות ששולמו, לא שולמו ובוטלו וטופס נוסף ועובד גם יכול להוציא טופס על טיול לראות את ההזמנות ששולמו, לא שולמו ובוטלו. העובד גם יכול להוציא כרטיסי טיסה לטיול ללקוחות על הזמנות ששולמו.

למנהלים יש גישה לאותם דברים כמו לעובד חוץ מהאפשרות להזמין טיול ללקוח, בנוסף המנהל יכול להוסיף עובדים ולהפסיק או להחזיר את הגישה לעובד למערכת. המנהל יכול להוציא דוח של עובד הכולל את הפרטים שלו עם הזמנות שהוא הזמין ששולמו/לא שולמו/ביטלו.

התוכנה פותחה במערכת הפעלה : Microsoft Windows 10.

בתוכנות: Access , Microsoft Visual Studio 2017.

הקוד נכתב בשפת : C#.

התוכנה נוחה לשימוש למשתמש וכתובה בשפה האנגלית.

מסך כניסהמסך כניסה

Login


Welcome to I.T traveling company


17-02-2021 14:50:36


User name:

Password:

Permission: Client









לקוחות, עובדים ומנהלים יכולים להיכנס למערכת, יש אפשרות ללקוח חדש להירשם משתמש שביטלו את הגישה שלו או הזנה לא נכונה יצא טקסט מתאים.



הרשמה בתור לקוח חדש

✕ Add Client

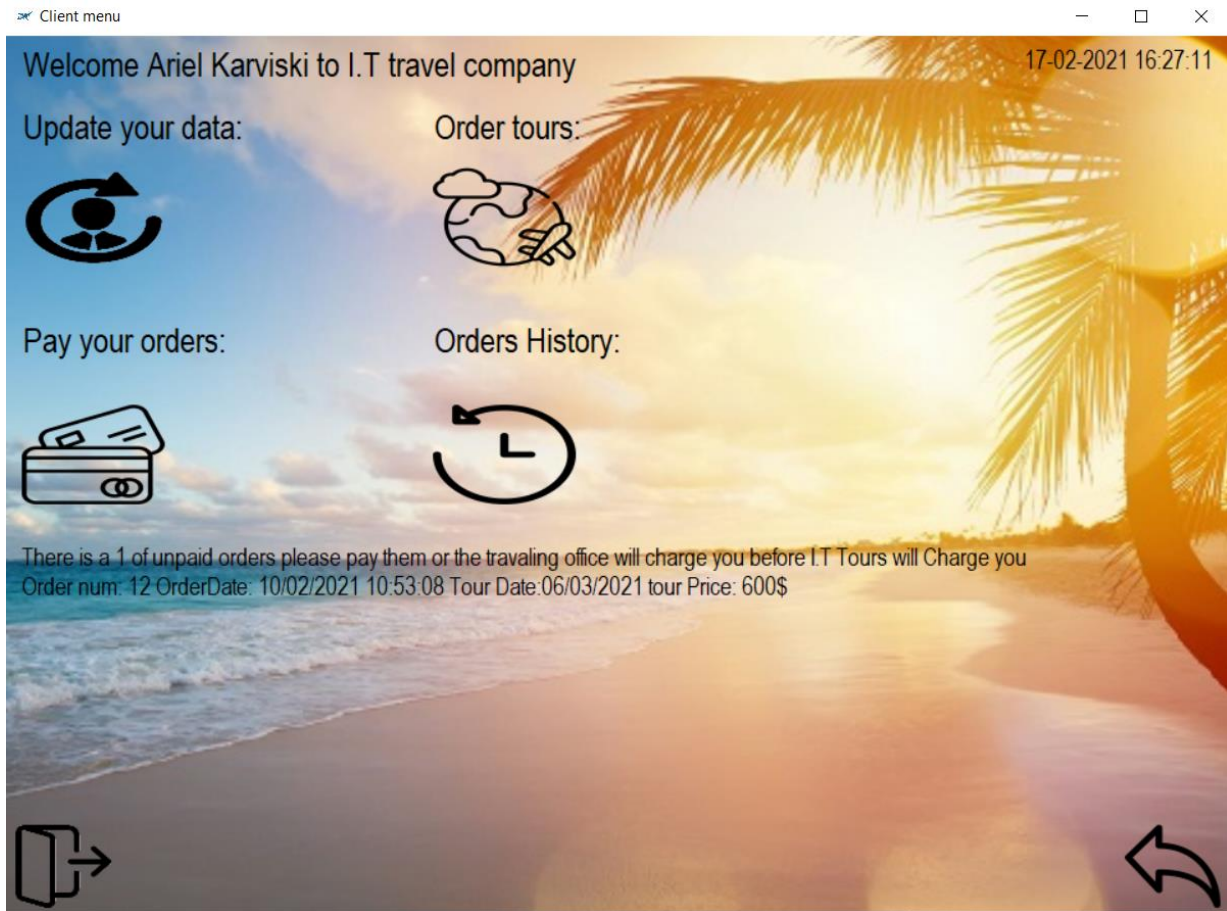
17-02-2021 14:56:26

Enter Data to join client

User Name:	<input type="text"/>	First Name:	<input type="text"/>
Password:	<input type="text"/>	Last Name:	<input type="text"/>
ID:	<input type="text"/>	Address:	<input type="text"/>
Email:	<input type="text"/>	City:	<input type="text"/>
Phone:	<input type="text"/>	Birthdate:	<input type="text" value="17/02/2021"/>
Pic:			

הרשמה למערכת בתור לקוח המערכת בודקת האם המידע מתאים ואז מוסיפה את הלקוח.

לקוחמסך לקוח

הלקוח יכול להיכנס לדברים השונים שהוא יכול לעשות וגם רואה הודעה לגבי הזמנות לטיולים שהוא עדיין לא שילם עליהם.

עדכון מידע

Update client


Update your data 17-02-2021 16:29:28

User Name:

ID:

Email:

Phone:

Pic: 

First Name:

Last Name:

Address:


City:


Birth date:


Old Password:


New Password:

Confirm Password:









לקוח יכול לעדכן חלק מהמידע שלו אם המידע מתאים או המידע מתעדכן, יש אפשרות לעדכן גם את הסיסמא.

הזמנת טיול

Client order tours

Make a new order:

3 Rome Italy 600

17-02-2021 16:33:30

Flight number: 98877464 Tour ID: 3

Price: 600\$ Tour Name: Rome

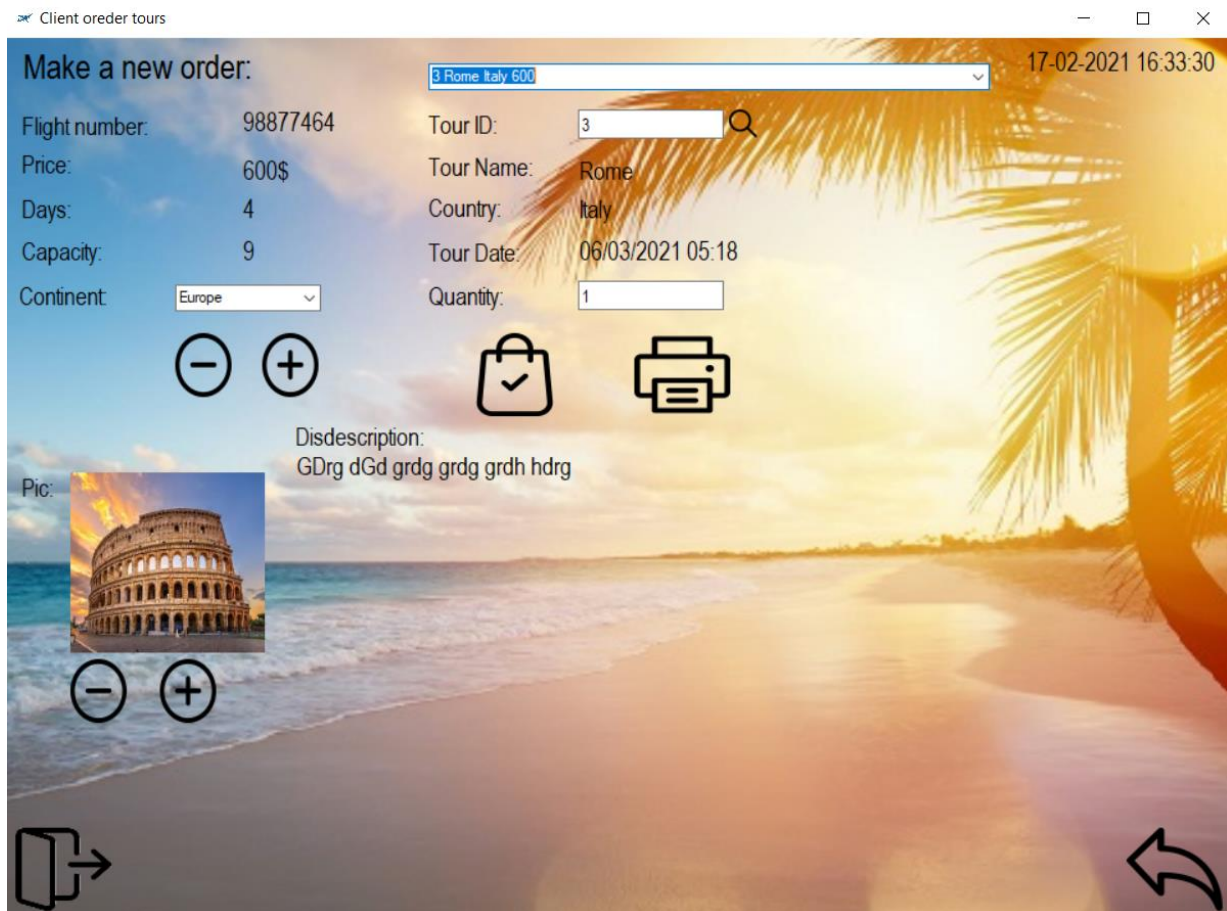
Days: 4 Country: Italy

Capacity: 9 Tour Date: 06/03/2021 05:18

Continent: Europe Quantity: 1

Disdescription: GDrg dGd grdg grdg grdh hdrd

Pic:



אפשר לבחור טיול ולהזמין אותו, בודק גם כמה מקומות הוזמנו אם מספר המקומות שווה או קטן מהמקומות שנשארו אז ההזמנה מתבצעת. מראה רק טיולים שהתאריך שלהם בעתיד.

תשלום על הזמנות


Client Unpaid Orders



Pay Your Orders

Order: 12 On: 06/03/2021 To: Number of people: 1 Price: 600 Total Price: 600

17-02-2021 16:37:16

Tour Name: Rome	Worker Name: None None
Tour Date: 06/03/2021 05:18	Price: 600\$
Country: Italy	Palces: 1
Days: 4	Total: 600\$
Flight: 98877464	Active: Order Active






לקוח רואה את רשימת ההזמנות שלא שולמו ובוחר אם לשלם עליהם.



היסטורית הזמנות

Client Orders

Orders history: Order:1 On:05/02/2021 To:9 Number of people:3 Price:600 Total Price:1800 17-02-2021 16:40:14

Tour Name: Rome	Worker Name: None None
Tour Date: 05/02/2021 05:18	Price: 600\$
Country: Italy	Palces: 3
Days: 4	Total: 1800\$
Flight: 98877464	Payment: Paid Up

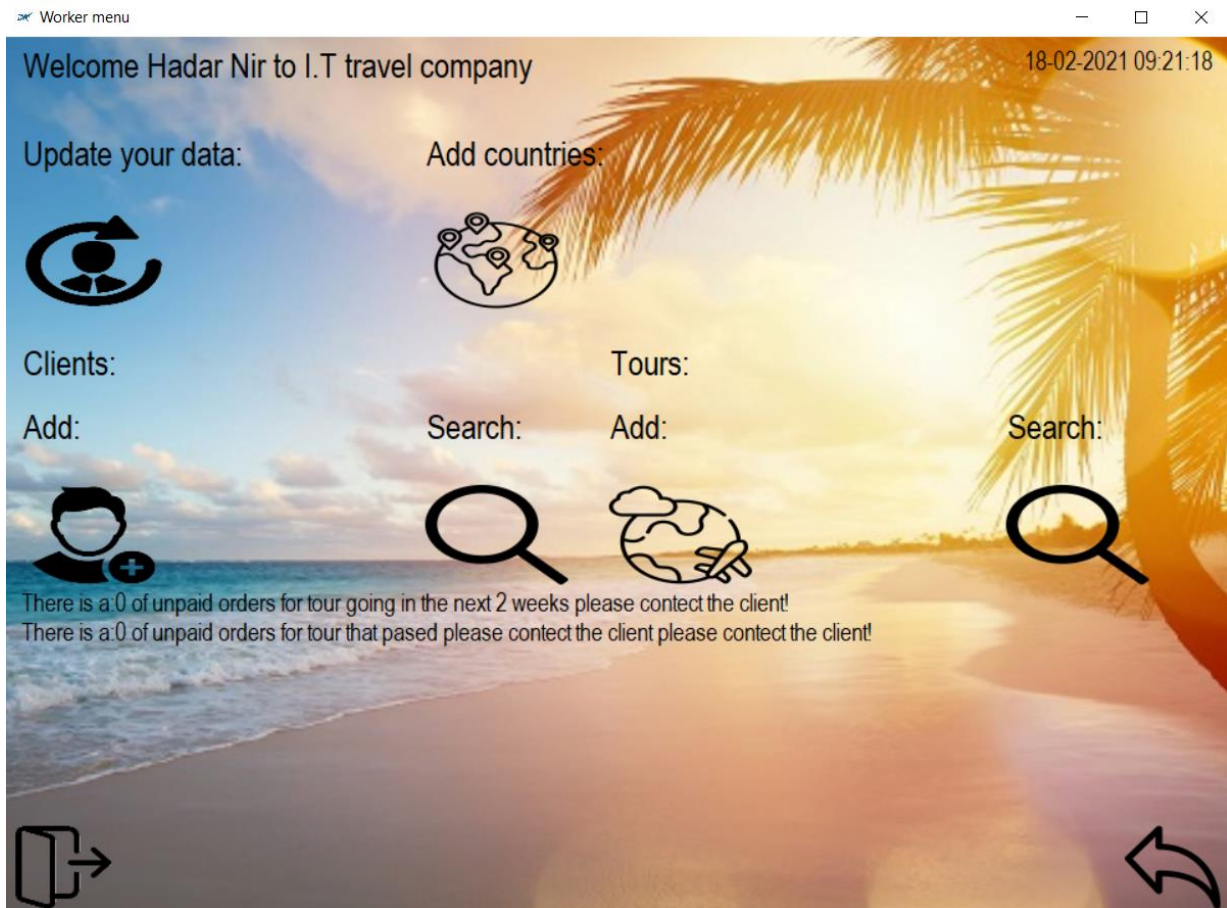



לקוח יכול לראות את כל ההזמנות שלו ולהוציא קבלה להזמנה ששולמה.

עובד

לעובד יש גם גישה לדף הוספת לקוח, הזמנת טיול, תשלום על הזמנות והיסטורית הזמנות של לקוח.

מסך עובד







העובד יכול להיכנס לדברים השונים שהוא יכול לעשות וגם רואה הודעה לגבי הזמנות שלא שולמו לטיולים שמתחילים בפחות מ14 ימים והזמנות שלא שולמו לטיולים שכבר עברו.



עדכון מידע

Update Worker 18-02-2021 09:25:07

Update your data

User Name:	<input type="text" value="hadar"/>	First Name:	<input type="text" value="Hadar"/>	Old Password:	<input type="text"/>
ID:	<input type="text" value="027019660"/>	Last Name:	<input type="text" value="Nir"/>	New Password:	<input type="text"/>
Salary:	<input type="text" value="5500"/>	Address:	<input type="text" value="Sigalon 12"/>	Confirm Password:	<input type="text"/>
admission date:	<input type="text" value="31/01/2021"/>	City:	<input type="text" value="Dimona"/>		
		Birth date:	<input type="text" value="01/06/1994"/>		

Pic:    

עובד יכול לעדכן חלק מהמידע שלו אם המידע מתאים אז המידע מתעדכן, יש אפשרות לעדכן גם את הסיסמא.

הוספת ארץ

Add Country
18-02-2021 09:27:29

Country Name:

Continent:
North america

+

↺

Country Name:
United States

→

↶

אפשר להוסיף ארץ ולהתאים אותה ליבשת יש אפשרות גם לשנות את השם של הארץ והיבשת של הארץ.

חיפוש לקוח

Find client

18-02-2021 09:42:32

Search a client

1 027019660 4486464684 Afst@dgr.com Ariel

User Name: Ariel


ID: 027019660

Email: Afst@dgr.com

Phone: 4486464684

Age: 26

Unpaid Orders: 1

Pic: 

First Name: Ariel

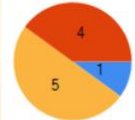
Last Name: Karviski

Address: VSgse 33

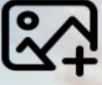











City: Besf

Activity: Active

Client Orders chart



Legend: Unpaid Orders (blue), Paid Orders (yellow), Canceled Ord... (red)

Icons:            

עובד יכול לחפש את הלקוחות, לעדכן את המידע, לעצור או לתת לו גישה לתוכנה, להדפיס דוח מידע, להיכנס לדף תשלומים של לקוח, להיכנס לדף הזמנת טיול, לראות היסטורית הזמנות ולהדפיס כרטיסי טיסה.

הדפסת כריסי טיסה

Flight Tickets 18-02-2021 09:46:12

Enter Data to join as client

Order num: 11 Tour Date: 26/02/2021 tour Price: 3000\$ Client name: Ariel Karvski Client Phone: 4486464684 Client Mail: Afef@dgr.com

AIR TICKET			BOARDING PASS		
Name of passenger	DATE	TIME	NAME		
	26/02/2021	06:10	FROM	Israel	
Class	GATE	SEAT	TO	Germany	
First Class	12	A3	DATE	26/02/2021	TIME 06:10
From	FLIGHT	A2	GATE	12	SEAT
Israel	66564353		FLIGHT	66564353	
Destination					
Germany					

Barcode: [Barcode]

Print icons: [Printer icon] [Printer icon]

Navigation icons: [Back icon] [Forward icon]

העובד מקבל רשימה של הזמנות ששולמו על טיולים שיהיו בעתיד ונותן מספר כרטיסים לפי מספר המקומות שהוזמנו ושומר את המקומות על שם ההזמנה יש אפשרות להדפיס מחדש את הכרטיס בשם אחר למקרה שהשם לא נכון.

הוספת טיול


✕ Add Tour



18-02-2021 09:53:53

Enter Data to add tour

Tour Name:	<input type="text"/>	Flight number:	<input type="text"/>	Tour Date:	<input type="text" value="18/02/2021"/>
Country:	<input type="text" value="United States"/>	Tour Gate:	<input type="text"/>	Days:	<input type="text"/>
Price:	<input type="text" value="1"/>	Return Flight number:	<input type="text"/>	Flight Time:	<input type="text" value="00"/> : <input type="text" value="00"/>
Capacity:	<input type="text"/>	Return Tour Gate:	<input type="text"/>	Return Time:	<input type="text" value="00"/> : <input type="text" value="00"/>

Disdescription:



העובד יכול להוסיף טיול ולשים תמונות לטיול.

חיפוש טיול

Find Tour

Enter Data to Update tour 1 Rome Italy 600 18-02-2021 10:02:13

Tour ID: 1

Tour Name: Rome

Country: Italy

Price: 600

Capacity: 0

Disdescription: GDrG dGd grdg grdg grdh hdrG

Flight number: 98877464

Tour Gate: 13

Return Flight number: 87564534

Return Gate: 32

Tour Date: 18/02/2021

Days: 4


Flight Time: 05 : 18

Return Time: 07 : 06

Tour chart

Legend: Taken (Blue), Empty (Orange)

Chart Data: Taken: 10, Empty: 0

Pic: 

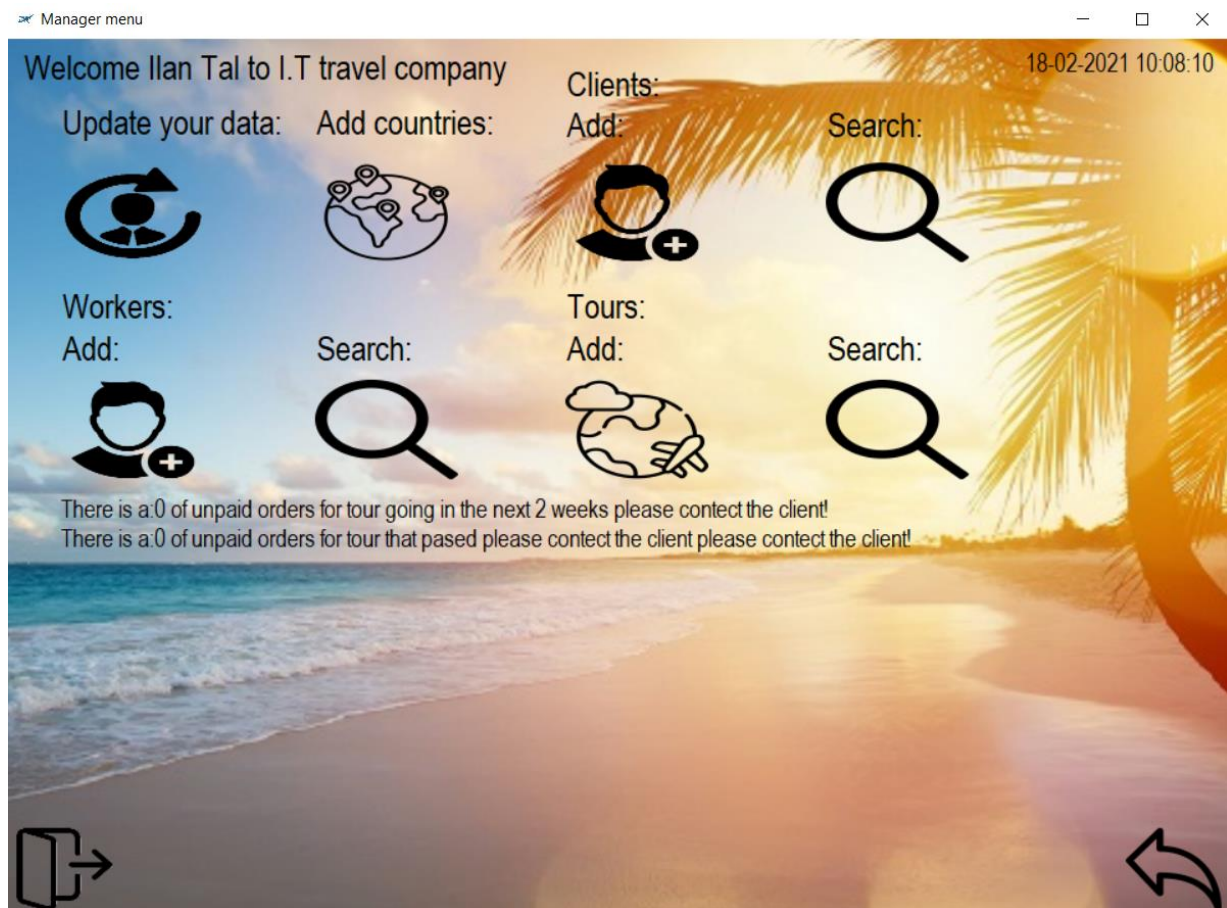
Navigation icons: Minus, Plus, Print, Mobile, Back, Forward, Refresh, Home, Search, etc.

יש אפשרות לראות את הטיולים להוסיף ולמחוק תמונות להוציא דוח מידע לטיול ולשכפל את הטיול לתאריך חדש עם מידע אחר (מספר טיסה, שעות ועוד).

מנהל

למנהל יש אותה גישה שיש לעובד חוץ מהזמנת טיול ללקוח.

דף מנהל









הננהל יכול להיכנס לדברים השונים שהוא יכול לעשות וגם רואה הודעה לגבי הזמנות שלא שולמו לטיולים שמתחילים בפחות מ14 ימים והזמנות שלא שולמו לטיולים שכבר עברו.

עדכון מידע

Update Manager 18-02-2021 10:13:08

Update your data

User Name:	<input type="text" value="ilan"/>	First Name:	<input type="text" value="Ilan"/>	Old Password:	<input type="password"/>
ID:	312201619	Last Name:	<input type="text" value="Tal"/>	New Password:	<input type="password"/>
Salary:	9000	Address:	<input type="text" value="Atad 4"/>	Confirm Password:	<input type="password"/>
admission date:	10/09/2020	City:	<input type="text" value="Beer-Sheva"/>		
Job:	<input type="text" value="CEO"/>	Birth date:	<input type="text" value="16/1/1994"/>		
Department:	General management				
Pic:					

Icons:     

מנהל יכול לעדכן חלק מהמידע שלו אם המידע מתנאים אז המידע מתעדכן, יש אפשרות לעדכן גם את הסיסמא.

הוספת עובד

✕ Add worker

18-02-2021 10:14:32

Add worker:

User Name:	<input type="text"/>	First Name:	<input type="text"/>
Password:	<input type="text"/>	Last Name:	<input type="text"/>
ID:	<input type="text"/>	Address:	<input type="text"/>
Salary:	<input type="text"/>	City:	<input type="text"/>
Birthdate:	<input type="text" value="18/02/2021"/>		
Pic:			

המנהל מוסיף עובד המערכת בודקת האם המידע מתאים ואז מוסיפה את הלקוח.

חיפוש עובד

Find worker

Search a worker

9 027019660 Hadar Nir hadar

18-02-2021 10:17:27

User Name: hadar

ID: 027019660

Absorption: 18/02/2021 10:17:23

Salary: 5500

Age: 26

Pic:

First Name: Hadar

Last Name: Nir

Address: Sigalon 12

City: Dimona

Activity: Active

OFF

Worker Orders chart

Unpaid Orders

Paid Orders

Canceled Ord...

1

0

1

מנהל יכול לחפש את העובדים, לעדכן את המידע, לעצור או לתת לו גישה לתוכנה, ולהדפיס דוח מידע.

דוחותדוח לקוח

Client Data From

Client: Ariel Karviski

Print Time:18/02/2021 10:23:20

[Email:Afsf@dgr.com](mailto:Afsf@dgr.com)

PKID: 1

ID: 027019660

Name: Ariel Karviski

Address: VSgse 33

Birthdate: 01/12/1994

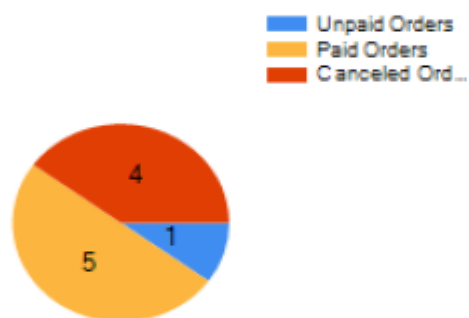
City: Besf

Username:Ariel

Activity=True

Phone number:4486464684

Client Orders chart



Client Data From
 Client: Ariel Karviski
 Print Time:18/02/2021 10:23:20

Ariel Karviski paid Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Payment	Price	Quantity	Total
1	1	Rome	05/02/2021	27/01/2021 09:03:40	600\$	3	1800\$
5	1	Rome	05/02/2021	01/02/2021 10:14:26	600\$	1	600\$
9	1	Rome	05/02/2021	04/02/2021 13:33:58	600\$	1	600\$
10	2	Auchshtenflath	26/02/2021	08/02/2021 12:41:44	1000\$	1	1000\$
11	2	Auchshtenflath	26/02/2021	09/02/2021 10:29:02	1000\$	3	3000\$
							7000\$

Client Data From
Client: Ariel Karviski
Print Time:18/02/2021 10:23:20

Ariel Karviski unpaid Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Worker Name	Price	Quantity	Total
12	3	Rome	06/03/2021	None	600\$	1	600\$
							600\$

Client Data From
 Client: Ariel Karviski
 Print Time:18/02/2021 10:23:20

Ariel Karviski Canceled Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Worker Name	Price	Quantity	Total
2	1	Rome	05/02/2021	None None	600\$	1	600\$
3	1	Rome	05/02/2021	None None	600\$	3	1800\$
4	1	Rome	05/02/2021	None None	600\$	1	600\$
8	1	Rome	05/02/2021	Hadar Nir	600\$	1	600\$
							4200\$



Worker Data From

Worker: Hadar Nir

Print Time: 18/02/2021 10:29:33

PKID: 9

ID: 027019660

Name: Hadar Nir

Address: Sigalon 12

Birthdate: 01/06/1994

City: Dimona

Username: hadar

Activity=True

Salary: 5500

Absorption Date: 31/01/2021 12:19:13

Worker Orders chart

Unpaid Orders
Paid Orders
Canceled Ord...



Worker Data From
Worker: Hadar Nir
Print Time:18/02/2021 10:29:33

Hadar Nir paid Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Payment	Price	Quantity	Total
9	1	Rome	05/02/2021	04/02/2021 13:33:58	600\$	1	600\$
							600\$

Worker Data From
Worker: Hadar Nir
Print Time:18/02/2021 10:29:33

Hadar Nir unpaid Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Client ID	Price	Quantity	Total
							0\$

Worker Data From
Worker: Hadar Nir
Print Time:18/02/2021 10:29:33

Hadar Nir Canceled Orders list							
Order ID	Tour ID	Tour Name	Tour Date	Client ID	Price	Quantity	Total
8	1	Rome	05/02/2021	027019660	600\$	1	600\$
							600\$



Receipt For Tour:1
for: Ariel Karviski ID:027019660

Order:

Order number:1

Order date:25/01/2021 17:20:37

Payment date:27/01/2021 09:03:40

Tour:

Tour PKID is: 1

Tour Name is: Rome

Flight number: 98877464

Price for a seat: 600\$

Country: Italy

Days:4

Date:05/02/2021

Time:05:18

Gate:13

Return Flight number:87564534

return time:07:06

return Gate:32

Worker:

PKID:8

UserName:None

Total payment:1800\$ For 3 Places.



Tour Data From
Tour: Rome
Print Time:18/02/2021 10:37:37

Tour PKID is: 3
Tour Name is: Rome
Flight number: 98877464
Price: 600\$
Continent:Europe
Country: Italy
Days:4
Date:06/03/2021
Time:05:18
Gate:13
Return Flight number:87564534
return time:07:06
return Gate:32

GDrg dGd grdg grdg grdh hdrq





Tour Data From

Tour: Rome

Print Time:18/02/2021 10:43:50

Tour PKID is: 1

Tour Name is: Rome

Flight number: 98877464

Price: 600\$

Continent:Europe

Country: Italy

Days:4

Date:05/02/2021

Time:05:18

Gate:13

Return Flight number:87564534

return time:07:06

return Gate:32

Tour chart

Taken
 Empty



Tour Data From
 Tour: Rome
 Print Time:18/02/2021 10:43:50

1 Rome paid Orders list							
Order ID	Client Name	Worker Name	Tour Date	Order Date	Payment	Quantity	Total
1	Ariel Karviski	None None	05/02/2021	25/01/2021 17:20:37	27/01/2021 09:03:40	3	1800\$
5	Ariel Karviski	None None	05/02/2021	31/01/2021 11:13:49	01/02/2021 10:14:26	1	600\$
6	Doron Sofer	None None	05/02/2021	31/01/2021 19:28:41	31/01/2021 19:29:46	2	1200\$
7	Doron Sofer	None None	05/02/2021	31/01/2021 19:29:14	01/02/2021 13:50:47	3	1800\$
9	Ariel Karviski	Hadar Nir	05/02/2021	04/02/2021 13:33:46	04/02/2021 13:33:58	1	600\$
							6000\$

Tour Data From
Tour: Rome
Print Time:18/02/2021 10:43:50

1 Rome unpaid Orders list							
Order ID	Client Name	Worker Name	Tour Date	Order Date	Payment	Quantity	Total
							0\$

Tour Data From
 Tour: Rome
 Print Time:18/02/2021 10:43:50

1 Rome Canceled Orders list							
Order ID	Client Name	Worker Name	Tour Date	Order Date	Payment	Quantity	Total
2	Ariel Karviski	None None	05/02/2021	25/01/2021 17:20:42	01/01/0001 00:00:00	1	600\$
3	Ariel Karviski	None None	05/02/2021	25/01/2021 17:20:51	01/01/0001 00:00:00	3	1800\$
4	Ariel Karviski	None None	05/02/2021	31/01/2021 09:39:10	01/01/0001 00:00:00	1	600\$
8	Ariel Karviski	Hadar Nir	05/02/2021	03/02/2021 11:06:22	01/01/0001 00:00:00	1	600\$
							3600\$













✈️ AIR TICKET			✈️ BOARDING PASS	
Name of passenger	DATE	TIME	NAME	Ilan Tal
Ilan Tal	26/02/2021	06:10	FROM	Israel
Class	GATE	SEAT	TO	Germany
First Class	12	A2	DATE	26/02/2021
From	FLIGHT		TIME	06:10
Israel	66564353		GATE	12
Destination			SEAT	A2
Germany			FLIGHT	66564353



✈️ AIR TICKET			✈️ BOARDING PASS	
Name of passenger	DATE	TIME	NAME	Ilan Tal
Ilan Tal	04/03/2021	06:08	FROM	Germany
Class	GATE	SEAT	TO	Israel
First Class	31	A2	DATE	04/03/2021
From	FLIGHT		TIME	06:08
Germany	09876543		GATE	31
Destination			SEAT	A2
Israel			FLIGHT	09876543



ממסד נתונים

Tables		
	Clients	
	Continents	
	Countries	
	Departments	
	Flight_Ticket	
	Managers	
	Orders	
	permissions	
	Pics_Tours	
	Tours	
	Workers	

קודטופס הוספת לקוח

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography; namespace IlanTalproTCB
{
    public partial class AddClientForm : Form
    {
        Function f1 = new Function();//Functions to check input data
        Client Client_c = new Client();//The new client class

        /*
        In this Form you add client to the database
        */
        public AddClientForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();//date and a clock
                SetDates();// set date time picker

                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg;
                *.jpe; *.jfif; *.png";//filter to get only picters
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }

        /*
        Set date time picker, client cant pick a birthday as future date
        */
        private void SetDates()
        {
            DTPClientBD.MaxDate = DateTime.Now;
            DTPClientBD.Value = DateTime.Now;
        }

        /*
        Show date and a clock on the Label
        */
        private void TDate_Tick(object sender, EventArgs e)
        {
            DateTime time = DateTime.Now;
            LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
        }

        /*
        Exit the program from Exit Button
        User need to conform the exit
        use fade timer
        */
        private void PBExit_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
            DialogResult.Yes)
            {
                TExit.Start();
            }
        }

        private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
        {

```

```

        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
        MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }

    /*
    Return to the last Form
    if its no permission its return to the Login menu
    if its worker permission its return to the worker menu
    if its manager permission its return to the manager menu
    */
    private void PBBack_Click(object sender, EventArgs e)
    {
        if (LoginForm.PKID == -1)
        {
            var LoginForm = new LoginForm();
            LoginForm.Closed += (s, args) => this.Close();
            LoginForm.Show();
            this.Hide();
        }
        else if (LoginForm.PKID != -1 && LoginForm.Permission == "Worker")
        {
            var WorkerMenuForm = new WorkerMenuForm();
            WorkerMenuForm.Closed += (s, args) => this.Close();
            WorkerMenuForm.Show();
            this.Hide();
        }
        else if (LoginForm.PKID != -1 && LoginForm.Permission == "Manager")
        {
            var ManagerMenuForm = new ManagerMenuForm();
            ManagerMenuForm.Closed += (s, args) => this.Close();
            ManagerMenuForm.Show();
            this.Hide();
        }
    }

    /*
    Add pic to client and copy pic to the client folder, dont copy if the pic its in the folder
    already
    */
    private void PBAddPic_Click(object sender, EventArgs e)
    {
        if (OFDAddPic.ShowDialog() == DialogResult.OK)
        {
            if (!new System.IO.FileInfo(@"Pic\Clients\" + OFDAddPic.SafeFileName).Exists)
            {
                System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Clients\" +
                OFDAddPic.SafeFileName);
            }
            PBClientPic.Image = Image.FromFile(@"Pic\Clients\" + OFDAddPic.SafeFileName);
            LPicName.Text = OFDAddPic.SafeFileName;
        }
    }

    /*
    Add client check the client data if the client data is suitable
    and if ID, email, username and phone number its a not exists
    and add it to the Database
    */
    private void PBAddClient_Click(object sender, EventArgs e)
    {
        bool flag = true;
        string str = "";
        f1.CheckUsername(TBUsername.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "User name needs 8-4 characters long and first 3 characters are letters \n";
        }
        f1.CheckPassword(TBPassword.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;

```

```

        str += "Password must contain 10 characters long \n";
    }
    f1.CheckID(TBID.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Worng ID\n";
    }
    f1.CheckPhone(TBPhone.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Phone must contain 10 digits\n";
    }
    f1.CheckDOB(DateTime.Parse(DTPClientBD.Text));
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Age must be number between 18 to 130\n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have at least 6 characters long and no number in the first 3
characters and finish with number\n";
    }
    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City must Start with Capital letter and over 3 letters long\n";
    }
    f1.CheckEmail(TBEmail.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Wrong Email\n";
    }
    if (LPicName.Text.Length<2)
    {
        flag = false;
        str += "Add picture\n";
    }
    LError.Text = str;
    if (flag)
    {
        f1.Checkdup(TBID.Text, "ID", "Clients");
        if (f1.GetAnswer()==false)
        {
            flag = false;
            str += "There is a Client with the same ID \n";
        }
        f1.Checkdup(TBUsername.Text, "UserName", "Clients");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Client with the same User name \n";
        }
        f1.Checkdup(TBEmail.Text, "Email", "Clients");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Client with the same Email \n";
        }
        f1.Checkdup(TBPhone.Text, "PhoneNumber", "Clients");
        if (f1.GetAnswer() == false)
        {

```

```

        flag = false;
        str += "There is a Client with the same Phone Number \n";
    }
    LError.Text = str;
    if (flag)
    {
        int m = f1.MakePKID("Clients")+1;
        Client_c = new Client(m, TBFName.Text, TBLName.Text, TBAddress.Text, DTPClientBD.Text,
TBCity.Text, TBUsername.Text, TBPASSWORD.Text, LPicName.Text, TBID.Text, TBEmail.Text, TBPhone.Text, true);
        Client_c.AddClientToDB();
        LError.Text = Client_c.GetFirstName()+" "+ Client_c.GetLastName()+" Added To data
base";
    }
}
}
}
/*
TB for ID and Phone number get only digits chars
*/
private void TBID_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) )
    {
        e.Handled = true;
    }
}
private void TBPhone_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
}
/*
button change color when mouse hover and leave
and tooltip show the text
*/
private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    TTMouseHover.Show("Add client", PBAddClient);
    PBAddClient.BackColor = Color.WhiteSmoke;
}
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.Transparent;
}
}
private void PBAddPic_MouseHover(object sender, EventArgs e)
{
    TTMouseHover.Show("Add pic", PBAddPic);
    PBAddPic.BackColor = Color.WhiteSmoke;
}
private void PBAddPic_MouseLeave(object sender, EventArgs e)
{
    PBAddPic.BackColor = Color.Transparent;
}
}
private void PBExit_MouseHover(object sender, EventArgs e)
{
    TTMouseHover.Show("Exit", PBExit);
    PBExit.BackColor = Color.WhiteSmoke;
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}
}
private void PBBack_MouseHover(object sender, EventArgs e)
{
    TTMouseHover.Show("Back", PBBack);
    PBBack.BackColor = Color.WhiteSmoke;
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
}
/*
timer fade
*/
private void TExit_Tick(object sender, EventArgs e)

```

```

    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tool tip draw background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

טופס הוספת ארץ

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    worker and manager can add country and change country name
    */
    public partial class AddCountryForm : Form
    {
        Function f1 = new Function();
        Country Country1 = new Country();
        ContinentList Continentslist = new ContinentList();
        CountryList countryList = new CountryList();
        int ContinentIndex = 0, CountryIndex = 0;

        public AddCountryForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                FillCBContinent();
                FillCBCountry();
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        fill country combobox with names
        */
        public void FillCBCountry()
        {
            CBCountrys.Items.Clear();
            foreach (Country i in countryList.getCountryList())
            {
                CBCountrys.Items.Add(i.GetCountryName());
            }
            CBCountrys.SelectedIndex = CountryIndex;
        }
        /*
        fill continent combobox with names
        */
    }
}

```

```

*/
public void FillCBContinent()
{
    CBContinent.Items.Clear();
    foreach (Continent i in Continentslist.GetContinentsList())
    {
        CBContinent.Items.Add(i.GetContinentName());
    }
    CBContinent.SelectedIndex = ContinentIndex;
}
/*
date and clock timer
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
Exit with fade timer
*/
private void PBCExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
go to last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    if (LoginForm.PKID != -1 && LoginForm.Permission == "Worker")
    {
        var WorkerMenuForm = new WorkerMenuForm();
        WorkerMenuForm.Closed += (s, args) => this.Close();
        WorkerMenuForm.Show();
        this.Hide();
    }
    else if (LoginForm.PKID != -1 && LoginForm.Permission == "Manager")
    {
        var ManagerMenuForm = new ManagerMenuForm();
        ManagerMenuForm.Closed += (s, args) => this.Close();
        ManagerMenuForm.Show();
        this.Hide();
    }
}
/*
add/update country to data base
*/
private void PBAddCountry_Click(object sender, EventArgs e)
{
    f1.CheckCountry(TBCountryName.Text);
    if (f1.GetAnswer())
    {
        f1.Checkdup(TBCountryName.Text, "CountryName", "Countries");
        if (f1.GetAnswer() == false)
        {
            LError.Text = "There is a Country with the same Name \n";
        }
        else
        {
            int m = f1.MakePKID("Countries") + 1;

```

```

        Country1 = new Country(m,
Continentslist.GetContinentsList()[ContinentIndex].GetPKID(), TBCountryName.Text);
Country1.AddCountryToDB();
CBCountrys.Items.Add(Country1.GetCountryName());
countryList.getCountryList().Add(Country1);
LError.Text = "Country Added\n";
    }
}
else
{
    LError.Text = "Worng Country name\n\n";
}
}
private void PBUpdateCountry_Click(object sender, EventArgs e)
{
    if (TBCountryName.Text == CBCountrys.SelectedItem.ToString())
    {
        LError.Text = "Same Country name";
    }
    else
    {
        f1.CheckCountry(TBCountryName.Text);
        if (f1.GetAnswer())
        {
            f1.Checkdup(TBCountryName.Text, "CountryName", "Countries");
            if (f1.GetAnswer() == false)
            {
                LError.Text = "There is a Country with the same Name \n";
            }
            else
            {
                countryList.getCountryList()[CountryIndex].SetCountryName(TBCountryName.Text);
countryList.getCountryList()[CountryIndex].SetContinentPKID(Continentslist.GetContinentsList()[Continent
Index].GetPKID());
                countryList.getCountryList()[CountryIndex].UpdateCountryName();
                LError.Text = "Country name Updated\n";
                CBCountrys.Items[CountryIndex] = TBCountryName.Text;
            }
        }
        else
        {
            LError.Text = "Worng Country name\n";
        }
    }
}
}
/*
change combobox selected index
*/
private void CBContinent_SelectedIndexChanged(object sender, EventArgs e)
{
    ContinentIndex = CBContinent.SelectedIndex;
}
private void CBCountrys_SelectedIndexChanged(object sender, EventArgs e)
{
    CountryIndex = CBCountrys.SelectedIndex;
}
/*
hover and tooltip mouse hover
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}
private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
private void PBAddCountry_MouseHover(object sender, EventArgs e)
{

```



```

        PBAddCountry.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Add country", PBAddCountry);
    }
    private void PBAddCountry_MouseLeave(object sender, EventArgs e)
    {
        PBAddCountry.BackColor = Color.Transparent;
    }
    private void PBUpdateCountry_MouseHover(object sender, EventArgs e)
    {
        PBUpdateCountry.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Change country name", PBUpdateCountry);
    }
    private void PBUpdateCountry_MouseLeave(object sender, EventArgs e)
    {
        PBUpdateCountry.BackColor = Color.Transparent;
    }
    /*
    fade timer
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tool tip background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

סופס הוספת טיול

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    add tour form user enter data and make a new tour
    */
    public partial class AddTourForm : Form
    {
        sha1 sh = new sha1();
        Function f1 = new Function();
        Tour t = new Tour();
        int i = 0;
        CountryList cl = new CountryList();
        Country c = new Country();
        public AddTourForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                DTPTour.Value = DateTime.Now;
                DTPTour.MinDate = DateTime.Now;
                FillCBCountry();
            }
        }
    }
}

```

```

        FillCBTime();
        OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg;
*.jpe; *.jfif; *.png"; //filter to get only picters
        TTMouseHover.OwnerDraw = true;
        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill the combobox with countries
*/
public void FillCBCountry()
{
    CBCountry.Items.Clear();
    foreach (Country i in cl.getCountryList())
    {
        CBCountry.Items.Add(i.GetCountryName());
    }
    CBCountry.SelectedIndex = i;
}
/*
fill the comboboxs with the hours and minutes times
*/
public void FillCBTime()
{
    for (int num=0;num<60;num++)
    {
        if (num < 10)
        {
            CBHour.Items.Add("0" + num.ToString());
            CBMinutes.Items.Add("0" + num.ToString());
            CBHourReturn.Items.Add("0" + num.ToString());
            CBMinutesReturn.Items.Add("0" + num.ToString());
        }
        else if (num < 24)
        {
            CBHour.Items.Add(num.ToString());
            CBHourReturn.Items.Add(num.ToString());
            CBMinutesReturn.Items.Add(num.ToString());
            CBMinutes.Items.Add(num.ToString());
        }
        else
        {
            CBMinutesReturn.Items.Add(num.ToString());
            CBMinutes.Items.Add(num.ToString());
        }
    }
    CBHour.SelectedIndex = 0;
    CBMinutes.SelectedIndex = 0;
    CBHourReturn.SelectedIndex = 0;
    CBMinutesReturn.SelectedIndex = 0;
}
/*can add pics only after the tour was added*/
public void ShowPicdata()
{
    PBAddTourPic.Visible = true;
    PBAddPic.Visible = true;
    LPic.Visible = true;
    PBTourPic.Visible = true;
}
public void HidePicdata()
{
    PBAddTourPic.Visible = false;
    PBAddPic.Visible = false;
    LPic.Visible = false;
    PBTourPic.Visible = false;
}
/*add tour to the database only if the input data good*/
private void PBAddTour_Click(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckFNumber(TBFlightnumber.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Flight number need 8 characters long\n";
    }
}

```

```

f1.CheckFNumber(TBRFN.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Return Flight number need 8 characters long\n";
}
f1.CheckPrice(TBPrice.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Price need start at least with 300$\n";
}
f1.CheckDays(TBDays.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Enter days\n";
}
f1.CheckDays(TBCapacity.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Enter Capacity\n";
}
f1.CheckName(TBTourName.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Tour name Start with Capital letter and more then 3 letters long\n";
}
if (TBGate.Text == "" || TBGate.Text == "0" && int.Parse(TBGate.Text) < 1000)
{
    flag = false;
    str += "Enter gate number less then 1000\n";
}
if (TBGate.Text == "" || TBGate.Text == "0" && int.Parse(TBGate.Text) < 1000)
{
    flag = false;
    str += "Enter return gate number less then 1000\n";
}
if (flag == false)
{
    LError.Text = str;
    HidePicdata();
}
else
{
    f1.Checkdup(String.Format("{0:dd/MM/yyyy}", DTPTour.Value), TBTourName.Text, "Tours");
    if (f1.GetAnswer() == false)
    {
        str += "There is a tour with the same name on the same date!\n";
    }
    else
    {
        f1.CheckTourdupFN(String.Format("{0:dd/MM/yyyy}",
DTPTour.Value), TBFlightnumber.Text);
        if (f1.GetAnswer() == false)
        {
            str += "There is a tour with the same FlightNumber on the same date!\n";
        }
        else
        {
            f1.CheckTourdupFN(DateTime.Parse(String.Format("{0:dd/MM/yyyy}",
DTPTour.Value)).AddDays(int.Parse(TBDays.Text)).ToString("dd/MM/yyyy"), TBRFN.Text);
            if (f1.GetAnswer() == false)
            {
                str += "There is a tour with the same Return FlightNumber on the same
date!\n";
            }
            else
            {
                int m = f1.MakePKID("Tours") + 1;
                t = new Tour(m, TBTourName.Text, TBFlightnumber.Text,
int.Parse(TBPrice.Text), c.GetPKID().ToString(), RTBDisdescription.Text, int.Parse(TBDays.Text),
String.Format("{0:dd/MM/yyyy}", DTPTour.Value), int.Parse(TBCapacity.Text),
CBHour.SelectedItem.ToString() + ":" + CBMinutes.SelectedItem.ToString(), int.Parse(TBGate.Text),
CBHourReturn.SelectedItem.ToString() + ":" + CBMinutesReturn.SelectedItem.ToString(), TBRFN.Text,
int.Parse(TBRGate.Text));
                t.AddTourToDB();
            }
        }
    }
}

```

```

        LError.Text = t.PrintTour();
        ShowPicdata();
    }
}
}
LError.Text = str;
}
/*add pic to the tour*/
private void PBAddTourPic_Click(object sender, EventArgs e)
{
    if (LPicName.Text.Length<3)
    {
        LError.Text = "Add picture to tour";
    }
    else
    {
        f1.CheckPicdup(LPicName.Text, t.GetTourID().ToString());
        if (f1.GetAnswer() == false)
        {
            LError.Text = "The picture is in the tour\n";
        }
        else
        {
            int m = f1.MakePKID("Pics_Tours") + 1;
            PicsTours p = new PicsTours(m, t.GetTourID(), LPicName.Text);
            p.AddPicToDB();
            LError.Text = "Picture added\n";
        }
    }
}
/*user pick a pic to add*/
private void PBAddPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        LPicName.Text = OFDAddPic.SafeFileName;
        if (!new System.IO.FileInfo(@"Pic\Tours\" + LPicName.Text).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Tours\" +
LPicName.Text);
        }
        PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
    }
}
/*cloak and date*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*Exit with fade timer*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
}
/*go to last form*/
private void PBBack_Click(object sender, EventArgs e)
{
    if (LoginForm.Permission == "Worker")
    {
        var WorkerMenuForm = new WorkerMenuForm();
        WorkerMenuForm.Closed += (s, args) => this.Close();
    }
}

```

```

        WorkerMenuForm.Show();
        this.Hide();
    }
    else if (LoginForm.Permission == "Manager")
    {
        var ManagerMenuForm = new ManagerMenuForm();
        ManagerMenuForm.Closed += (s, args) => this.Close();
        ManagerMenuForm.Show();
        this.Hide();
    }
}
/*flight numbers, gates, price, cap and days are only digits*/
private void TBFlightnumber_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBPrice_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBDays_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBCapacity_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBGate_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBRFN_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBRGate_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
/*hover and tooltip withmouse*/
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.Transparent;
}
private void PBAddTourPic_MouseLeave(object sender, EventArgs e)
{
    PBAddTourPic.BackColor = Color.Transparent;
}
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{

```

```

        PBExit.BackColor = Color.Transparent;
    }
    private void PBBack_MouseHover(object sender, EventArgs e)
    {
        PBBack.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Back", PBBack);
    }
    private void PBBack_MouseLeave(object sender, EventArgs e)
    {
        PBBack.BackColor = Color.Transparent;
    }
    private void PBAAddPic_MouseHover(object sender, EventArgs e)
    {
        PBAAddPic.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Search pic", PBAAddPic);
    }
    private void PBAAddPic_MouseLeave(object sender, EventArgs e)
    {
        PBAAddPic.BackColor = Color.Transparent;
    }
    private void PBAAddTourPic_MouseHover(object sender, EventArgs e)
    {
        PBAAddTourPic.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Add pic", PBAAddTourPic );
    }
    private void PBAAddTour_MouseHover(object sender, EventArgs e)
    {
        PBAAddTour.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Add Tour", PBAAddTour);
    }
    }
    /*change selected country*/
    private void CBCountry_SelectedIndexChanged(object sender, EventArgs e)
    {
        i = CBCountry.SelectedIndex;
        c = cl.getCountryList()[i];
    }
    }
    /*fade timer*/
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    }
    /*tooltip background*/
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
    }
}

```

טופס הוספת עובד

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    add worker form
    add worker to the data base

```

```

*/
public partial class AddWorkerForm : Form
{
    sha1ceypto sh = new sha1ceypto();
    Function f1 = new Function();
    Worker w = new Worker();
    public AddWorkerForm()
    {
        try
        {
            this.BackgroundImage = Properties.Resources.background;
            InitializeComponent();
            SetDates();
            TDate.Start();
            OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg;
*.jpe; *.jfif; *.png";
            TTMouseHover.OwnerDraw = true;
            TTMouseHover.ForeColor = Color.Black;
            TTMouseHover.BackColor = Color.White;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString(), "ERROR");
        }
    }
    /*
    Set date time picker, client cant pick a birthday as future date
    */
    private void SetDates()
    {
        DTPWorkerBD.MaxDate = DateTime.Now;
        DTPWorkerBD.Value = DateTime.Now;
    }
    /*
    Show date and a clock on the Label
    */
    private void TDate_Tick(object sender, EventArgs e)
    {
        DateTime time = DateTime.Now;
        LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
    }

    private void PBExit_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            TExit.Start();
        }
    }
    /*
    Exit the program from Exit Button
    User need to conform the exit
    use fade timer
    */
    private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }
    /*
    Return to the last Form
    */
    private void PBBack_Click(object sender, EventArgs e)
    {
        var ManagerMenuForm = new ManagerMenuForm();
        ManagerMenuForm.Closed += (s, args) => this.Close();
        ManagerMenuForm.Show();
        this.Hide();
    }
}

```

```

/*
add pic
*/
private void PBAAddPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        LPicName.Text = OFDAddPic.SafeFileName;
        if (!new System.IO.FileInfo(@"Pic\Workers\" + LPicName.Text).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Workers\" +
LPicName.Text);
        }
        PBClientPic.Image = Image.FromFile(@"Pic\Workers\" + LPicName.Text);
    }
}
/*
add worker
*/
private void PBAAddClient_Click(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "User name need 8-4 characters long and first 3 characters are letters \n";
    }
    f1.CheckPassword(TBPassword.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Password must have 10 characters long \n";
    }
    f1.CheckID(TBID.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Worng ID \n";
    }
    f1.CheckDOB(DateTime.Parse(DTPWorkerBD.Text));
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Age must be number more then 18\n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have lest 6 characters long and no number in the first 3 characters
and finish with number\n";
    }
    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City must Start with Capital letter and over 3 letters long\n";
    }
    if (LPicName.Text.Length < 2)
    {
        flag = false;
        str += "Add picture\n";
    }
    f1.CheackSalary(TBSalary.Text);
    if (f1.GetAnswer() == false)
    {

```



```

        flag = false;
        str += "Salary must be between 4500-8000\n";
    }
    LError.Text = str;
    if (flag)
    {
        f1.Checkdup(TBID.Text, "ID", "Workers");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Worker with the same ID \n";
        }
        f1.Checkdup(TBUsername.Text, "UserName", "Workers");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Worker with the same User name \n";
        }
    }

    LError.Text = str;
    if (flag)
    {
        int m = f1.MakePKID("Workers")+1;
        w = new Worker(m, TBName.Text, TBLName.Text, TBAddress.Text, DTPWorkerBD.Text,
        TBCity.Text, TBUsername.Text, sh.GetSHA1(TBPassword.Text), LPicName.Text, TBID.Text,
        int.Parse(TBSalary.Text), true);
        w.AddWorkerToDB();
        LError.Text = w.PrintWorker;
    }
}
}
/*
ID, salary get only digits
*/
private void TBID_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) )
    {
        e.Handled = true;
    }
}
private void TBSalary_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
/*
button change color when mouse hover and leave
and tooltip show the text
*/
private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add worker", PBAddClient);
}

private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.Transparent;
}

private void PBAddPic_MouseHover(object sender, EventArgs e)
{
    PBAddPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update pic", PBAddPic);
}

private void PBAddPic_MouseLeave(object sender, EventArgs e)
{
    PBAddPic.BackColor = Color.Transparent;
}

private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}

```

```

private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}

private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
/*
timer fade
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tool tip draw background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

טופס מסך לקוח

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    this is clients menu he can go to all of his forms
    update data, order tour, payments, orders history
    */
    public partial class ClientMenuForm : Form
    {
        Function f1 = new Function();
        Client c = new Client(LoginForm.PKID.ToString());
        ReceiptList rl = new ReceiptList();
        public ClientMenuForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                rl.BuildReceiptsByClientUnpaidActive(LoginForm.PKID);
                if (rl.GetReceiptsList().Count != 0) //client get massege on orders that he didnt pay
                {
                    LShowClientData.Text = rl.PrintReceiptsData();
                }
                LHeader.Text = "Welcome " + c.GetFirstName() + " " + c.GetLastName() + " to I.T travel
company";
                TTMouseHover.OwnerDraw = true;
            }
        }
    }
}

```

```

        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
date timer
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
exit with fade timer
*/
private void PExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
*/
private void PExit_MouseHover(object sender, EventArgs e)
{
    PExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PExit);
}
private void PExit_MouseLeave(object sender, EventArgs e)
{
    PExit.BackColor = Color.Transparent;
}

private void PBack_MouseHover(object sender, EventArgs e)
{
    PBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBack);
}
private void PBack_MouseLeave(object sender, EventArgs e)
{
    PBack.BackColor = Color.Transparent;
}

private void PUpdateClient_MouseHover(object sender, EventArgs e)
{
    PUpdateClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update your data", PUpdateClient);
}
private void PUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PUpdateClient.BackColor = Color.Transparent;
}

private void PClientOrderTours_MouseHover(object sender, EventArgs e)
{
    PClientOrderTours.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Order tour", PClientOrderTours);
}
private void PClientOrderTours_MouseLeave(object sender, EventArgs e)

```

```

{
    PBClientOrderTours.BackColor = Color.Transparent;
}

private void PBClientPaymentOrder_MouseHover(object sender, EventArgs e)
{
    PBClientPaymentOrder.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Orders Payment", PBClientPaymentOrder);
}
private void PBClientPaymentOrder_MouseLeave(object sender, EventArgs e)
{
    PBClientPaymentOrder.BackColor = Color.Transparent;
}

private void PBClientPastOrders_MouseHover(object sender, EventArgs e)
{
    PBClientPastOrders.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Orders history", PBClientPastOrders);
}
private void PBClientPastOrders_MouseLeave(object sender, EventArgs e)
{
    PBClientPastOrders.BackColor = Color.Transparent;
}
/*
go to the forms
*/
private void PBBack_Click(object sender, EventArgs e)
{
    var LoginForm = new LoginForm();
    LoginForm.Closed += (s, args) => this.Close();
    LoginForm.Show();
    this.Hide();
}
private void PBUpdateClient_Click(object sender, EventArgs e)
{
    var UpdateClientForm = new UpdateClientForm();
    UpdateClientForm.Closed += (s, args) => this.Close();
    UpdateClientForm.Show();
    this.Hide();
}
private void PBClientOrderTours_Click(object sender, EventArgs e)
{
    var OrderTour = new ClientOrderTourForm();
    OrderTour.Closed += (s, args) => this.Close();
    OrderTour.Show();
    this.Hide();
}
private void PBClientPaymentOrder_Click(object sender, EventArgs e)
{
    var ClientUnpaidOrders = new ClientUnpaidOrders();
    ClientUnpaidOrders.Closed += (s, args) => this.Close();
    ClientUnpaidOrders.Show();
    this.Hide();
}
private void PBClientPastOrders_Click(object sender, EventArgs e)
{
    var ClientOrdersForm = new ClientOrdersForm();
    ClientOrdersForm.Closed += (s, args) => this.Close();
    ClientOrdersForm.Show();
    this.Hide();
}
/*
fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tool tip draw background
*/

```

```

private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

טופס מסך לקוח

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    this is clients menu he can go to all of his forms
    update data, order tour, payments, orders history
    */
    public partial class ClientMenuForm : Form
    {
        Function f1 = new Function();
        Client c = new Client(LoginForm.PKID.ToString());
        ReceiptList rl = new ReceiptList();
        public ClientMenuForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                rl.BuildReceiptsByClientUnpaidActive(LoginForm.PKID);
                if (rl.GetReceiptsList().Count != 0) //client get massege on orders that he didnt pay
                {
                    LShowClientData.Text = rl.PrintReceiptsData();
                }
                LHeader.Text = "Welcome " + c.GetFirstName() + " " + c.GetLastName() + " to I.T travel
company";
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        date timer
        */
        private void TDate_Tick(object sender, EventArgs e)
        {
            DateTime time = DateTime.Now;
            LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
        }
        /*
        exit with fade timer
        */
        private void PBEExit_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                TExit.Start();
            }
        }
        private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
            if (confirm == DialogResult.Yes)

```

```

    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBUpdateClient_MouseHover(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update your data", PBUpdateClient);
}
private void PBUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.Transparent;
}

private void PBClientOrderTours_MouseHover(object sender, EventArgs e)
{
    PBClientOrderTours.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Order tour", PBClientOrderTours);
}
private void PBClientOrderTours_MouseLeave(object sender, EventArgs e)
{
    PBClientOrderTours.BackColor = Color.Transparent;
}

private void PBClientPaymentOrder_MouseHover(object sender, EventArgs e)
{
    PBClientPaymentOrder.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Orders Payment", PBClientPaymentOrder);
}
private void PBClientPaymentOrder_MouseLeave(object sender, EventArgs e)
{
    PBClientPaymentOrder.BackColor = Color.Transparent;
}

private void PBClientPastOrders_MouseHover(object sender, EventArgs e)
{
    PBClientPastOrders.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Orders history", PBClientPastOrders);
}
private void PBClientPastOrders_MouseLeave(object sender, EventArgs e)
{
    PBClientPastOrders.BackColor = Color.Transparent;
}
/*
go to the forms
*/
private void PBBack_Click(object sender, EventArgs e)
{
    var LoginForm = new LoginForm();
    LoginForm.Closed += (s, args) => this.Close();
    LoginForm.Show();
}

```

```

        this.Hide();
    }
    private void PBUpdateClient_Click(object sender, EventArgs e)
    {
        var UpdateClientForm = new UpdateClientForm();
        UpdateClientForm.Closed += (s, args) => this.Close();
        UpdateClientForm.Show();
        this.Hide();
    }
    private void PBClientOrderTours_Click(object sender, EventArgs e)
    {
        var OrderTour = new ClientOrderTourForm();
        OrderTour.Closed += (s, args) => this.Close();
        OrderTour.Show();
        this.Hide();
    }
    private void PBClientPaymentOrder_Click(object sender, EventArgs e)
    {
        var ClientUnpaidOrders = new ClientUnpaidOrders();
        ClientUnpaidOrders.Closed += (s, args) => this.Close();
        ClientUnpaidOrders.Show();
        this.Hide();
    }
    private void PBClientPastOrders_Click(object sender, EventArgs e)
    {
        var ClientOrdersForm = new ClientOrdersForm();
        ClientOrdersForm.Closed += (s, args) => this.Close();
        ClientOrdersForm.Show();
        this.Hide();
    }
    /*
    fade timer
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tool tip draw background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}
}

```

טופס היסטורית הזמנות

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    client can see all his orders and print a receipt to paid orders
    */
    public partial class ClientOrdersForm : Form
    {
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Receipt r = new Receipt();
    }
}

```

```

ReceiptList rl = new ReceiptList();
Country con = new Country();
int i = 0;
public ClientOrdersForm()
{
    try
    {
        this.BackgroundImage = Properties.Resources.background;
        InitializeComponent();
        TDate.Start();
        if (LoginForm.Permission == "Client")
        {
            rl.BuildReceiptsByClient(LoginForm.PKID);
        }
        else
        {
            rl.BuildReceiptsByClient(FindClientForm.clientid);
        }
        if (rl.GetReceiptsList().Count != 0)
        {
            r = rl.GetReceiptsList()[i];
            FillCBReceipt();
        }
        else
        {
            PBPlus.Visible = false;
            PBMinus.Visible = false;
            CBReceiptList.Visible = false;
            LHeader.Text = "There arent any Orders";
        }
        TTMouseHover.OwnerDraw = true;
        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill the orders of the client in the combobox
*/
public void FillCBReceipt()
{
    CBReceiptList.Items.Clear();
    foreach (Receipt i in rl.GetReceiptsList())
    {
        CBReceiptList.Items.Add("Order:" + i.GetOrder().GetOrderNumber() + " On:" +
i.GetTour().GetDate() + " To:" + i.GetTour().GetCountry() + " Number of people:" +
i.GetOrder().GetQuantity() + " Price:" + i.GetTour().GetPrice() + " Total Price:" + (i.GetTour().GetPrice() *
i.GetOrder().GetQuantity()));
    }
    CBReceiptList.SelectedIndex = i;
}
/*
put data in the form
*/
public void fillReceiptdata()
{
    LRTourName.Text = r.GetTour().GetTourname();
    LRTourDate.Text = r.GetTour().GetDate() + r.GetTour().GetTimeHM();
    LRCountry.Text = con.GetCountryName();
    LRDays.Text = r.GetTour().GetDay().ToString();
    LRFlight_number.Text = r.GetTour().GetFlight_numberID();
    if (r.GetWorker().GetPKID() == 4)
    {
        LRWName.Text = "Client ordered the tour";
    }
    else
    {
        LRWName.Text = r.GetWorker().GetFirstName() + " " + r.GetWorker().GetLastName();
    }
    LRPrice.Text = r.GetTour().GetPrice().ToString() + "$";
    LRPlaces.Text = r.GetOrder().GetQuantity().ToString();
    LRTotal.Text = (r.GetTour().GetPrice() * r.GetOrder().GetQuantity()).ToString() + "$";
    if (r.GetOrder().GetPayment() == true)
    {
        LRPaymentStatus.Text = "Paid Up";
        PBMakePDFOrder.Visible = true;
    }
}

```



```

else
{
    LRPaymentStatus.Text = "Not Paid";
    PBMakePDFOrder.Visible = false;
}

}
/*
if list empty put clear form data
*/
public void ClearReceiptdata()
{
    LRTourName.Text = "";
    LRTourDate.Text = "";
    LRCountry.Text = "";
    LRDays.Text = "";
    LRFlight_number.Text = "";
    LRWName.Text = "";
    LRPrice.Text = "";
    LRPlaces.Text = "";
    LRTotal.Text = "";
    LRPaymentStatus.Text = "";
    PBPlus.Visible = false;
    PBMinus.Visible = false;
    CBReceiptList.Visible = false;
}
/*
select other item in orders list
*/
private void CBReceiptList_SelectedIndexChanged(object sender, EventArgs e)
{
    i = CBReceiptList.SelectedIndex;
    r = rl.GetReceiptsList()[i];
    con = new Country(r.GetTour().GetCountry());
    fillReceiptdata();
}
/*
date and clock
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
back too last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    if (LoginForm.Permission == "Client")
    {
        var ClientMenuForm = new ClientMenuForm();
        ClientMenuForm.Closed += (s, args) => this.Close();
        ClientMenuForm.Show();
    }
}

```

```

        this.Hide();
    }
    else
    {
        var FindClientForm = new FindClientForm();
        FindClientForm.Closed += (s, args) => this.Close();
        FindClientForm.Show();
        this.Hide();
    }
}
/*
next/last order
*/
private void PBPlus_Click(object sender, EventArgs e)
{
    i++;
    if (i == r1.GetReceiptsList().Count)
    {
        i = 0;
    }
    r = r1.GetReceiptsList()[i];
    CBReceiptList.SelectedIndex = i;
}
private void PBMinus_Click(object sender, EventArgs e)
{
    i--;
    if (i < 0)
    {
        i = r1.GetReceiptsList().Count - 1;
    }
    r = r1.GetReceiptsList()[i];
    CBReceiptList.SelectedIndex = i;
}
/*
make a Receipt PDF
*/
private void PBMakePDFOrder_Click(object sender, EventArgs e)
{
    if (CBReceiptList.Items.Count != 0)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            string Path = saveFileDialog1.FileName;

            r1.GetReceiptsList()[i].PrintReceiptDataPDF(System.IO.Path.GetFileNameWithoutExtension(saveFileDialog1.FileName), System.IO.Path.GetDirectoryName(Path));
        }
    }
}
/*
hover and tooltip
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
private void PBMinus_MouseHover(object sender, EventArgs e)
{
    PBMinus.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Last order", PBMinus);
}
private void PBMinus_MouseLeave(object sender, EventArgs e)
{
    PBMinus.BackColor = Color.Transparent;
}

```

```

    }

    private void PBPlus_MouseHover(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Next order", PBPlus);
    }
    private void PBPlus_MouseLeave(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.Transparent;
    }

    private void PBMakePDFOrder_MouseHover(object sender, EventArgs e)
    {
        PBMakePDFOrder.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Make a receipt", PBMakePDFOrder);
    }
    private void PBMakePDFOrder_MouseLeave(object sender, EventArgs e)
    {
        PBMakePDFOrder.BackColor = Color.Transparent;
    }
    /*
    fade timer
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tooltip background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}
}

```

טופס הזמנת טיולים

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    /*
    this is the client order tours form the client can see the tours in the future and can order places
    to the tour
    */
    public partial class ClientOrderTourForm : Form
    {
        Function f1 = new Function();
        Tour t = new Tour();
        TourList tours = new TourList();
        TourList Toursbycon = new TourList();
        Order o = new Order();
        PicsTourList picsTour = new PicsTourList();
        PicsTourList picsToTour = new PicsTourList();
        Country con = new Country();
        ContinentList cl = new ContinentList();
        PicsTours pictour = new PicsTours();
        Continent c = new Continent();
    }
}

```

```

int i = 0;
int j = 0;
int k = 0;
public ClientOrderTourForm()
{
    try
    {
        this.BackgroundImage = Properties.Resources.background;
        InitializeComponent();
        TDate.Start();
        FillCBContinent();
        tours.GetToursByDate();
        if (Toursbycon.GetTourList().Count > 0)//no tours
        {
            t = Toursbycon.GetTourList()[i];
            fillTourdata();
        }
        else
        {
            EmptyTourdata();
        }
        TTMouseHover.OwnerDraw = true;
        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill the Continent combobox
*/
public void FillCBContinent()
{
    CBContinent.Items.Clear();
    foreach (Continent cont in cl.GetContinentsList())
    {
        CBContinent.Items.Add(cont.GetContinentName());
    }
    CBContinent.SelectedIndex = k;
}
/*
fill the tour in the same Continent in the combobox
*/
public void FillCBTours()
{
    bool flag = false;
    CBToursList.Items.Clear();
    foreach (Tour i in Toursbycon.GetTourList())
    {
        Country tempc = new Country(i.GetCountry());
        flag = true;
        CBToursList.Items.Add(i.GetTourID() + " " + i.GetTourname() + " " +
tempc.GetCountryName() + " " + i.GetPrice());
    }
    if (flag)
    {
        CBToursList.SelectedIndex = 0;
        LError.Text = "";
        PBTourPic.Visible = true;
        PBPrintTourPDF.Visible = true;
    }
    else
    {
        PBPrintTourPDF.Visible = false;
        PBTourPic.Visible = false;
        LError.Text = "No Tours on that Continent";
        EmptyTourdata();
        CBToursList.ResetText();
    }
}
/*
if there is no tour empty the data
*/
public void EmptyTourdata()
{
    TBTourID.Text = "";
    LTourFN.Text = "";
    LTourPrice.Text = "";
}

```

```

    LTourDays.Text = "";
    LTourCap.Text = "";
    LTourDatein.Text = "";
    LTourN.Text = "";
    LTourCountry.Text = "";
    LShowDisdescription.Text = "";
}
/*
put tour of the tour
*/
public void fillTourdata()
{
    TBTourID.Text = Toursbycon.GetTourList()[i].GetTourID().ToString();
    LTourFN.Text = Toursbycon.GetTourList()[i].GetFlight_numberID();
    LTourPrice.Text = Toursbycon.GetTourList()[i].GetPrice().ToString()+"$";
    LTourDays.Text = Toursbycon.GetTourList()[i].GetDay().ToString();
    LTourCap.Text = Toursbycon.GetTourList()[i].GetCapacity().ToString();
    LTourDatein.Text = Toursbycon.GetTourList()[i].GetDate()+"
"+Toursbycon.GetTourList()[i].GetTimeHM();
    LTourN.Text = Toursbycon.GetTourList()[i].GetTourname();
    LTourCountry.Text = con.GetCountryName();
    LShowDisdescription.Text = Toursbycon.GetTourList()[i].Getdisdescription();
    if (Toursbycon.GetTourList()[i].GetCapacity()==0)
    {
        PBAddOrder.Visible = false;
        LError.Text = "Tour full";
    }
    else
    {
        PBAddOrder.Visible = true;
    }
}
/*
client order tour check if there is enough places in the tour
add the order in the database
sub the tour places
*/
private void PBAddOrder_Click(object sender, EventArgs e)
{
    if (Toursbycon.GetTourList().Count != 0)
    {
        if (int.Parse(TBQuantity.Text) < 1)
        {
            LError.Text = "Enter Quantity!";
        }
        else
        {
            if (int.Parse(TBQuantity.Text) > t.GetCapacity())
            {
                LError.Text = "There are not enough places on the trip";
            }
            else
            {
                int m = f1.MakePKID("Orders") + 1;
                if (LoginForm.Permission == "Client")
                {
                    o = new Order(m, int.Parse(LoginForm.PKID.ToString()), 8, t.GetTourID(),
false, int.Parse(TBQuantity.Text));
                }
                else
                {
                    o = new Order(m, int.Parse(FindClientForm.clientid.ToString()),
int.Parse(LoginForm.PKID.ToString()), t.GetTourID(), false, int.Parse(TBQuantity.Text));
                }
                o.AddOrderToDB();
                t.SetCapacity(t.GetCapacity() - int.Parse(TBQuantity.Text));
                t.UpdateCapacity();
                fillTourdata();
                LError.Text = o.PrintOrder();
            }
        }
    }
}
/*
move to the next/last tour
*/
private void PBPlus_Click(object sender, EventArgs e)
{
    if (Toursbycon.GetTourList().Count != 0)
    {
        i++;
    }
}

```

```

        if (i == Toursbycon.GetTourList().Count)
        {
            i = 0;
        }
        t = Toursbycon.GetTourList()[i];
        CBToursList.SelectedIndex = i;
    }
}
private void PBMinus_Click(object sender, EventArgs e)
{
    if (Toursbycon.GetTourList().Count != 0)
    {
        i--;
        if (i < 0)
        {
            i = Toursbycon.GetTourList().Count - 1;
        }
        t = Toursbycon.GetTourList()[i];
        CBToursList.SelectedIndex = i;
    }
}
/*
move to the next/last picters
*/
private void PBPicPlus_Click(object sender, EventArgs e)
{
    if (Toursbycon.GetTourList().Count != 0)
    {
        if (picsToTour.GetTourPicList().Count != 0)
        {
            j++;
            if (j >= picsToTour.GetTourPicList().Count)
            {
                j = 0;
            }
            LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
            PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
        }
    }
}
private void PBPicMinus_Click(object sender, EventArgs e)
{
    if (Toursbycon.GetTourList().Count != 0)
    {
        if (picsToTour.GetTourPicList().Count != 0)
        {
            j--;
            if (j < 0)
            {
                j = picsToTour.GetTourPicList().Count - 1;
            }
            LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
            PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
        }
    }
}
/*
make a PDF wite tour data in Pdf for clients
*/
private void PBPrintTourPDF_Click(object sender, EventArgs e)
{
    if (SFDTourPDF.ShowDialog() == DialogResult.OK)
    {
        string Path = SFDTourPDF.FileName;

t.PrintTourDataPDFClient(System.IO.Path.GetFileNameWithoutExtension(SFDTourPDF.FileName),
System.IO.Path.GetDirectoryName(Path));
    }
}
/*
change Continent change the tour list with the same Continent
*/
private void CBContinent_SelectedIndexChanged(object sender, EventArgs e)
{
    k = CBContinent.SelectedIndex;
    c = cl.GetContinentsList()[k];
    Toursbycon = tours.GetToursByContinent(c.GetPKID());
    FillCBTours();
}
/*

```

```

change tour and put the tour data in form
*/
private void CBToursList_SelectedIndexChanged(object sender, EventArgs e)
{
    i = CBToursList.SelectedIndex;
    t = tours.GetTourList()[i];
    j = 0;
    con = new Country(tours.GetTourList()[i].GetCountry());
    fillTourdata();
    picsToTour.SetTourPicList(picsTour.FindPicByTourID(t.GetTourID()));
    if (picsToTour.GetTourPicList().Count != 0)
    {
        LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
        PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
    }
    else
    {
        LPicName.Text = "";
        PBTourPic.Image = null;
    }
}
/*
find tour by tour id
*/
private void PBFTID_Click(object sender, EventArgs e)
{
    string str = "";
    if (TBTourID.Text=="")
    {
        str = "Enter Tour ID\n";
    }
    else
    {
        int temp = int.Parse(TBTourID.Text);
        Tour newt = new Tour();
        t = tours.FindTourbyID(TBTourID.Text);
        i = tours.GetTourList().IndexOf(t);
        if (t.GetTourID() != 0)
        {
            Country c = new Country(t.GetCountry());
            CBCContinent.SelectedIndex = c.GetContinentPKID() - 1;
            newt = Toursbycon.FindTourbyID(temp.ToString());
            i = tours.GetTourList().IndexOf(newt);
        }
        if (i != -1)
        {
            CBToursList.SelectedIndex = i;
            str = "Tour Found";
        }
        else
        {
            str = "Tour with that ID dont exist";
            EmptyTourdata();
        }
    }
    LError.Text = str;
}
/*
Exit form with fade
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}

```

```

    }
}/*
return to last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    if (LoginForm.Permission == "Client")
    {
        var ClientrMenuForm = new ClientMenuForm();
        ClientrMenuForm.Closed += (s, args) => this.Close();
        ClientrMenuForm.Show();
        this.Hide();
    }
    else
    {
        var FindClientForm = new FindClientForm();
        FindClientForm.Closed += (s, args) => this.Close();
        FindClientForm.Show();
        this.Hide();
    }
}
/*
Textbox and labels that get only digits
*/
private void TBTourID_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBQuantity_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
/*
hover and leave mouse
*/
private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    PBAddOrder.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Order tour", PBAddOrder);
}
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddOrder.BackColor = Color.Transparent;
}

private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBPicPlus_MouseHover(object sender, EventArgs e)
{
    PBPicPlus.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Next pic", PBPicPlus);
}
private void PBPicPlus_MouseLeave(object sender, EventArgs e)
{
    PBPicPlus.BackColor = Color.Transparent;
}

```



```

}

private void PBPicMinus_MouseHover(object sender, EventArgs e)
{
    PBPicMinus.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("previous pic", PBPicMinus);
}
private void PBPicMinus_MouseLeave(object sender, EventArgs e)
{
    PBPicMinus.BackColor = Color.Transparent;
}

private void PBPlus_MouseHover(object sender, EventArgs e)
{
    PBPlus.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Next Tour", PBPlus);
}
private void PBPlus_MouseLeave(object sender, EventArgs e)
{
    PBPlus.BackColor = Color.Transparent;
}

private void PBMinus_MouseHover(object sender, EventArgs e)
{
    PBMinus.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("previous Tour", PBMinus);
}
private void PBMinus_MouseLeave(object sender, EventArgs e)
{
    PBMinus.BackColor = Color.Transparent;
}

private void PBFTID_MouseHover(object sender, EventArgs e)
{
    PBFTID.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find tour by tour ID", PBFTID);
}
private void PBFTID_MouseLeave(object sender, EventArgs e)
{
    PBFTID.BackColor = Color.Transparent;
}

private void PBPrintTourPDF_MouseHover(object sender, EventArgs e)
{
    PBPrintTourPDF.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Data in PDF", PBPrintTourPDF);
}
private void PBPrintTourPDF_MouseLeave(object sender, EventArgs e)
{
    PBPrintTourPDF.BackColor = Color.Transparent;
}
/*
exit fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
clock timer
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
Tool tip get background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
}

```

```

        e.DrawBorder();
        e.DrawText();
    }
}
}

```

טופס תשלום הזמנות

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    in this form client can pay his unpaid orders
    he get a list of his unpaid orders
    workers and managers can cancel orders and the tour add places to the tour
    */
    public partial class ClientUnpaidOrders : Form
    {
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Receipt r = new Receipt();
        ReceiptList rl = new ReceiptList();
        Country con = new Country();
        int i = 0;
        public ClientUnpaidOrders()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                if (LoginForm.Permission == "Client")
                {
                    rl.BuildReceiptsByClientUnpaid(LoginForm.PKID);
                    PBDeleteOrder.Visible = false;
                }
                else
                {
                    rl.BuildReceiptsByClientUnpaid(FindClientForm.clientid);
                }
                if (rl.GetReceiptsList().Count != 0)
                {
                    r = rl.GetReceiptsList()[i];
                    FillCBReceipt();
                }
                else
                {
                    PBPayOrder.Visible = false;
                    PBPlus.Visible = false;
                    PBMinus.Visible = false;
                    CBReceiptList.Visible = false;
                    PBDeleteOrder.Visible = false;
                    LHeader.Text = "There Not Unpaid Orders";
                }
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        fill unpaid orders combo box
        */
        public void FillCBReceipt()
        {
            CBReceiptList.Items.Clear();
            foreach (Receipt i in rl.GetReceiptsList())
            {

```

```

        CBReceiptList.Items.Add("Order:" + i.GetOrder().GetOrderNumber() + " On:" +
i.GetTour().GetDate() + " To:" + con.GetCountryName() + " Number of people:" +
i.GetOrder().GetQuantity() + " Price:" + i.GetTour().GetPrice() + " Total Price:" + (i.GetTour().GetPrice() *
i.GetOrder().GetQuantity()));
    }
    CBReceiptList.SelectedIndex = i;
}
/*
put order data in the form
*/
public void fillReceiptdata()
{
    LRTourName.Text = r.GetTour().GetTourname();
    LRTourDate.Text = r.GetTour().GetDate() + r.GetTour().GetTimeHM();
    LRCountry.Text = con.GetCountryName();
    LRDays.Text = r.GetTour().GetDay().ToString();
    LRFlight_number.Text = r.GetTour().GetFlight_numberID();
    if (r.GetWorker().GetPKID() == 4)
    {
        LRWName.Text = "Client ordered the tour";
    }
    else
    {
        LRWName.Text = r.GetWorker().GetFirstName() + " " + r.GetWorker().GetLastName();
    }
    LRPrice.Text = r.GetTour().GetPrice().ToString() + "$";
    LRPlaces.Text = r.GetOrder().GetQuantity().ToString();
    LRTotal.Text = (r.GetTour().GetPrice() * r.GetOrder().GetQuantity()).ToString() + "$";
    if (r.GetOrder().GetActive())
    {
        LRAActive.Text = "Order Active";
        PBPayOrder.Visible = true;
        if (LoginForm.Permission == "Client")
        {
            PBDeleteOrder.Visible = false;
        }
        else
        {
            PBDeleteOrder.Visible = true;
        }
    }
    else
    {
        LRAActive.Text = "Order Inactive";
        PBPayOrder.Visible = false;
        PBDeleteOrder.Visible = false;
    }
}
/*
if no orders in the list it empty the form
*/
public void ClearReceiptdata()
{
    LRTourName.Text = "";
    LRTourDate.Text = "";
    LRCountry.Text = "";
    LRDays.Text = "";
    LRFlight_number.Text = "";
    LRWName.Text = "";
    LRPrice.Text = "";
    LRPlaces.Text = "";
    LRTotal.Text = "";
    PBPayOrder.Visible = false;
    PBPlus.Visible = false;
    PBMinus.Visible = false;
    CBReceiptList.Visible = false;
}
/*
date and clock
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{

```

```

        if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            TExit.Start();
        }
    }
    private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }
    /*
move to last form
*/
    private void PBBack_Click(object sender, EventArgs e)
    {
        if (LoginForm.Permission == "Client")
        {
            var ClientMenuForm = new ClientMenuForm();
            ClientMenuForm.Closed += (s, args) => this.Close();
            ClientMenuForm.Show();
            this.Hide();
        }
        else
        {
            var FindClientForm = new FindClientForm();
            FindClientForm.Closed += (s, args) => this.Close();
            FindClientForm.Show();
            this.Hide();
        }
    }
    /*
move orders
*/
    private void PBPlus_Click(object sender, EventArgs e)
    {
        i++;
        if (i == r1.GetReceiptsList().Count)
        {
            i = 0;
        }
        r = r1.GetReceiptsList()[i];
        CBReceiptList.SelectedIndex = i;
    }
    private void PBMinus_Click(object sender, EventArgs e)
    {
        i--;
        if (i < 0)
        {
            i = r1.GetReceiptsList().Count - 1;
        }
        r = r1.GetReceiptsList()[i];
        CBReceiptList.SelectedIndex = i;
    }
    /*
pay orders
*/
    private void PBPayOrder_Click(object sender, EventArgs e)
    {
        r.GetOrder().PayOrder();
        r1.GetReceiptsList().RemoveAt(i);
        CBReceiptList.Items.RemoveAt(i);
        if (r1.GetReceiptsList().Count == 0)
        {
            PBPayOrder.Visible = false;
            LHeader.Text = "There Not Unpaid Orders";
            ClearReceiptdata();
            CBReceiptList.ResetText();
            CBReceiptList.Items.Clear();
        }
        else
    }

```

```

        {
            i = 0;
            r = r1.GetReceiptsList()[i];
            CBRceiptList.SelectedIndex = i;
        }
    }
    /*
    Delete orders
    */
    private void PBDeleteOrder_Click(object sender, EventArgs e)
    {
        r1.GetOrder().UnActiveOrder();

        r1.GetReceiptsList()[i].GetTour().SetCapacity(r1.GetReceiptsList()[i].GetTour().GetCapacity() +
        r1.GetReceiptsList()[i].GetOrder().GetQuantity());
        r1.GetReceiptsList()[i].GetTour().UpdateCapacity();
        r1.GetReceiptsList()[i].GetOrder().SetActive(false);
        r = r1.GetReceiptsList()[i];
        fillReceiptdata();
    }
    /*
    mouse hover
    */
    private void PBExit_MouseHover(object sender, EventArgs e)
    {
        PBExit.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Exit", PBExit);
    }
    private void PBExit_MouseLeave(object sender, EventArgs e)
    {
        PBExit.BackColor = Color.Transparent;
    }

    private void PBBack_MouseHover(object sender, EventArgs e)
    {
        PBBack.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Back", PBBack);
    }
    private void PBBack_MouseLeave(object sender, EventArgs e)
    {
        PBBack.BackColor = Color.Transparent;
    }

    private void PBON_MouseHover(object sender, EventArgs e)
    {
        PBPayOrder.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Pay order", PBPayOrder);
    }
    private void PBON_MouseLeave(object sender, EventArgs e)
    {
        PBPayOrder.BackColor = Color.Transparent;
    }

    private void PBMinus_MouseHover(object sender, EventArgs e)
    {
        PBMinus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Last order", PBMinus);
    }
    private void PBMinus_MouseLeave(object sender, EventArgs e)
    {
        PBMinus.BackColor = Color.Transparent;
    }

    private void PBPlus_MouseHover(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Next order", PBPlus);
    }
    private void PBPlus_MouseLeave(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.Transparent;
    }

    private void PBDeleteOrder_MouseHover(object sender, EventArgs e)
    {
        PBDeleteOrder.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Cancel order", PBDeleteOrder);
    }

```

```

    }
    private void PBDeleteOrder_MouseLeave(object sender, EventArgs e)
    {
        PBDeleteOrder.BackColor = Color.Transparent;
    }
    /*
    Change item in Receipt List
    */
    private void CBReceiptList_SelectedIndexChanged(object sender, EventArgs e)
    {
        i = CBReceiptList.SelectedIndex;
        r = rl.GetReceiptsList()[i];
        con = new Country(r.GetTour().GetCountry());
        fillReceiptdata();
    }
    /*
    fade timer
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    Tooltip background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

סופס מציאת לקוח

```

using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    find client form
    can update clients and see orders, print pdf data and order tours and flight ticket
    */
    public partial class FindClientForm : Form
    {
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Client c = new Client();
        public static int clientid;
        ClientList Clients = new ClientList();
        int i = 0;
        public FindClientForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                ClientChart.Titles.Add("Client Orders chart");
                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";
            }
        }
    }
}

```

```

        if (LoginForm.Permission == "Manager")
        {
            PBClientOrderTour.Visible = false;
        }
        if (Clients.GetClientList().Count != 0)
        {
            c = Clients.GetClientList()[i];

            clientid = c.GetPKID();
            FillCBClients();
        }
        else
        {
            PBMinus.Visible = false;
            PBMakePDFClient.Visible = false;
            PBPlus.Visible = false;
            PBOFF.Visible = false;
            PBON.Visible = false;
            PBUpdPic.Visible = false;
            PBFlightTicket.Visible = false;
            PBClientOrders.Visible = false;
            PBUpaidOrders.Visible = false;
            PBUpdateClient.Visible = false;
        }
        TTMouseHover.OwnerDraw = true;
        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill client combo box
*/
public void FillCBClients()
{
    CBCClientList.Items.Clear();
    foreach (Client i in Clients.GetClientList())
    {
        CBCClientList.Items.Add(i.GetPKID() + " " + i.GetID() + " " + i.GetPhoneNumber() + " " +
i.GetMail() + " " + i.GetUserName());
    }
    CBCClientList.SelectedIndex = i;
}
/*
fill client data in the form
*/
public void fillClientdata()
{
    foreach (var series in ClientChart.Series)
    {
        series.Points.Clear();
    }
    ClientChart.Series["s1"].IsValueShownAsLabel = true;
    ClientChart.Series["s1"].Points.AddXY("Unpaid Orders", c.GetOrderCount().ToString());
    ClientChart.Series["s1"].Points.AddXY("Paid Orders", c.GetOrderCount2().ToString());
    ClientChart.Series["s1"].Points.AddXY("Canceled Orders", c.GetOrderCount3().ToString());
    TBAddress.Text = Clients.GetClientList()[i].GetAddress();
    TBUsername.Text = Clients.GetClientList()[i].GetUserName();
    TBLName.Text = Clients.GetClientList()[i].GetLastName();
    TBFName.Text = Clients.GetClientList()[i].GetFirstName();
    TBEmail.Text = Clients.GetClientList()[i].GetMail();
    TBPhone.Text = Clients.GetClientList()[i].GetPhoneNumber();
    TBCity.Text = Clients.GetClientList()[i].GetCity();
    LShowAge.Text = Clients.GetClientList()[i].GetBirthdate();
    TBID.Text = Clients.GetClientList()[i].GetID();
    PBClientPic.Image = Image.FromFile(@"Pic\Clients\" + Clients.GetClientList()[i].GetPic());
    LPicName.Text = Clients.GetClientList()[i].GetPic();
    LShowAge.Text = Clients.GetClientList()[i].ShowAge().ToString() ;
    if (Clients.GetClientList()[i].GetActivity())
    {
        LActiveornot.Text = "Active";
        PBON.Visible = false;
        PBOFF.Visible = true;
    }
    else

```

```

    {
        LActiveornot.Text = "Inactive";
        PBON.Visible = true;
        PBOFF.Visible = false;
    }
    int counter = c.GetOrderCount();
    LCounter.Text = counter.ToString();
    if (counter>3)
    {
        LCounter.ForeColor = Color.Red;
    }
    else
    {
        LCounter.ForeColor = Color.Black;
    }
}
/*
Clock and date
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
}
/*
go to last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    if (LoginForm.Permission == "Worker")
    {
        var WorkerMenuForm = new WorkerMenuForm();
        WorkerMenuForm.Closed += (s, args) => this.Close();
        WorkerMenuForm.Show();
        this.Hide();
    }
    else if (LoginForm.Permission == "Manager")
    {
        var ManagerMenuForm = new ManagerMenuForm();
        ManagerMenuForm.Closed += (s, args) => this.Close();
        ManagerMenuForm.Show();
        this.Hide();
    }
}
}
/*
Update pic
*/
private void PBUpdPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        LPicName.Text = OFDAddPic.SafeFileName;
        if (!new System.IO.FileInfo(@"Pic\Clients\" + LPicName.Text).Exists)
        {

```



```

        System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Clients\" +
LPicName.Text);
    }
    PBClientPic.Image = Image.FromFile(@"Pic\Clients\" + LPicName.Text);
}
}
/*
Update client cheack UN PN EMAIL duplicate
*/
private void PBUpdateClient_Click(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "User name need 8-4 chars long and first 3 chars are letters \n";
    }
    f1.CheckPhone(TBPhone.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Phone must have 10 digits\n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have leat 6 chars long and no number in the first 3 chars and
finish with number\n";
    }

    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckEmail(TBEmail.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Wrong Email\n";
    }
    if (LPicName.Text.Length < 2)
    {
        flag = false;
        str += "Add picture\n";
    }
    LError.Text = str;
    if (flag)
    {
        if (TBUsername.Text != c.GetUserName())
        {
            f1.Checkdup(TBUsername.Text, "UserName", "Clients");
            if (f1.GetAnswer() == false)
            {
                flag = false;
                str += "There is a Client with the same User name \n";
            }
        }
        if (TBEmail.Text!=c.GetMail())
        {
            f1.Checkdup(TBEmail.Text, "Email", "Clients");
            if (f1.GetAnswer() == false)
            {
                flag = false;
            }
        }
    }
}

```

```

        str += "There is a Client with the same Email \n";
    }
}
if (TBPhone.Text != c.GetPhoneNumber())
{
    f1.Checkdup(TBPhone.Text, "PhoneNumber", "Clients");
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "There is a Client with the same Phone Number \n";
    }
}
if (flag)
{
    i = Clients.GetClientList().IndexOf(c);
    c.SetUserName(TBUsername.Text);
    c.SetFirstName(TBFName.Text);
    c.SetLastName(TBLName.Text);
    c.SetAddress(TBAddress.Text);
    c.SetCity(TBCity.Text);
    c.SetMail(TBEmail.Text);
    c.SetPhoneNumber(TBPhone.Text);
    c.SetPic(LPicName.Text);
    c.UpdateClientToDB();
    str = "User updated";
    Clients.GetClientList()[i] = c;
    FillCBClients();
}
LError.Text = str;
}
}
/*
Last/next client
*/
private void PBMinus_Click(object sender, EventArgs e)
{
    i--;
    if (i < 0)
    {
        i = Clients.GetClientList().Count-1;
    }
    c = Clients.GetClientList()[i];
    CBCClientList.SelectedIndex = i;
}
private void PBPlus_Click(object sender, EventArgs e)
{
    i++;
    if (i == Clients.GetClientList().Count)
    {
        i = 0;
    }
    c = Clients.GetClientList()[i];
    CBCClientList.SelectedIndex = i;
}
/*
Find client by id/UN/Email/PN
*/
private void PBFCID_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckID(TBID.Text);
    if (f1.GetAnswer() == false)
    {
        str += "Wrong ID";
    }
    else
    {
        c = Clients.FindClientbyID(TBID.Text);
        i = Clients.GetClientList().IndexOf(c);
        if (i != -1)
        {
            CBCClientList.SelectedIndex=i;
            str = "User Found by ID";
        }
        else
        {
            str = "User with that ID dont exist";
        }
    }
}
LError.Text = str;

```

```

}
private void PBFCUN_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        str = "User name need 8-4 chars long and first 3 chars are letters \n";
    }
    else
    {
        c = Clients.FindClientbyUN(TBUsername.Text);
        i = Clients.GetClientList().IndexOf(c);
        if (i != -1)
        {
            CBClientList.SelectedIndex = i;
            str = "User Found by user name";
        }
        else
        {
            str = "User with that User name dont exist";
        }
    }
    LError.Text = str;
}
private void PBFCBPN_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckPhone(TBPhone.Text);
    if (f1.GetAnswer() == false)
    {
        str += "Phone must have 10 digits\n";
    }
    else
    {
        c = Clients.FindClientbyPN(TBPhone.Text);
        i = Clients.GetClientList().IndexOf(c);
        if (i != -1)
        {
            CBClientList.SelectedIndex = i;
            str = "User Found by Phone number";
        }
        else
        {
            str = "User with that Phone number dont exist";
        }
    }
    LError.Text = str;
}
private void PBFCBEM_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckEmail(TBEmail.Text);
    if (f1.GetAnswer() == false)
    {
        str += "Wrong Email\n";
    }
    else
    {
        c = Clients.FindClientbyEMail(TBEmail.Text);
        i = Clients.GetClientList().IndexOf(c);
        if (i != -1)
        {
            CBClientList.SelectedIndex = i;
            str = "User Found by Email";
        }
        else
        {
            str = "User with that Email dont exist";
        }
    }
    LError.Text = str;
}
/*
Make client data PDF
*/
private void PBMakePDFClient_Click(object sender, EventArgs e)
{
    if (Clients.GetClientList().Count != 0)
    {

```

```

        if (SFDSaveClientFile.ShowDialog() == DialogResult.OK)
        {
            string Path = SFDSaveClientFile.FileName;
            this.ClientChart.SaveImage("pic/MyClientChart.png",
System.Drawing.Imaging.ImageFormat.Png);

c.PrintClientDataPDF(System.IO.Path.GetFileNameWithoutExtension(SFDSaveClientFile.FileName),
"pic/MyClientChart.png", System.IO.Path.GetDirectoryName(Path));
        }
    }
}
/*
Activate/deactivate client
*/
private void PBON_Click(object sender, EventArgs e)
{
    i = Clients.GetClientList().IndexOf(c);
    c.UpdateClientActivityToDB(true);
    int temp = i;
    fillClientdata();
    Clients.GetClientList()[temp] = c;
    LError.Text = "User is now active";
}
private void PBOFF_Click(object sender, EventArgs e)
{
    i = Clients.GetClientList().IndexOf(c);
    c.UpdateClientActivityToDB(false);
    int temp = i;
    fillClientdata();
    Clients.GetClientList()[temp] = c;
    LError.Text = "User is now inactive";
}
/*
go to other forms
*/
private void PBUpaidOrders_Click(object sender, EventArgs e)
{
    var ClientUnpaidOrders = new ClientUnpaidOrders();
    ClientUnpaidOrders.Closed += (s, args) => this.Close();
    ClientUnpaidOrders.Show();
    this.Hide();
}

private void PBClientOrders_Click(object sender, EventArgs e)
{
    var ClientOrdersForm = new ClientOrdersForm();
    ClientOrdersForm.Closed += (s, args) => this.Close();
    ClientOrdersForm.Show();
    this.Hide();
}
private void PBClientOrderTour_Click(object sender, EventArgs e)
{
    {
        var ClientOrderTourForm = new ClientOrderTourForm();
        ClientOrderTourForm.Closed += (s, args) => this.Close();
        ClientOrderTourForm.Show();
        this.Hide();
    }
}
private void PBFlightTicket_Click(object sender, EventArgs e)
{
    var FlightTicketForm = new FlightTicketForm();
    FlightTicketForm.Closed += (s, args) => this.Close();
    FlightTicketForm.Show();
    this.Hide();
}
/*
hover and tooltip with mouse
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)

```

```

{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBUpdPic_MouseHover(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update pic", PBUpdPic);
}
private void PBUpdPic_MouseLeave(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.Transparent;
}

private void PBUpdateClient_MouseHover(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update client", PBUpdateClient);
}
private void PBUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.Transparent;
}

private void PBFCUN_MouseHover(object sender, EventArgs e)
{
    PBFCUN.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Search by user name", PBFCUN);
}
private void PBFCUN_MouseLeave(object sender, EventArgs e)
{
    PBFCUN.BackColor = Color.Transparent;
}

private void PBFCID_MouseHover(object sender, EventArgs e)
{
    PBFCID.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Search by ID", PBFCID);
}
private void PBFCID_MouseLeave(object sender, EventArgs e)
{
    PBFCID.BackColor = Color.Transparent;
}
}

```

סופס מציאת טיול

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    find tour form user can find form and make a new one with difrent data
    */
    public partial class FindTour : Form
    {
        Function f1 = new Function();
        Tour t = new Tour();
        TourList tours = new TourList();
        PicsTourList picsTour = new PicsTourList();
        PicsTourList picsToTour = new PicsTourList();
        PicsTours pictour = new PicsTours();
        Country c = new Country();
        CountryList cl = new CountryList();
        int i = 0;
        int j = 0;
    }
}

```

```

int k = 0;
public FindTour()
{
    try
    {
        this.BackgroundImage = Properties.Resources.background;
        InitializeComponent();
        TDate.Start();
        DTPTour.Value = DateTime.Now;
        DTPTour.MinDate = DateTime.Now;
        TourChart.Titles.Add("Tour chart");
        if (tours.GetTourList().Count != 0)
        {
            t = tours.GetTourList()[i];
            FillCBCountry();
            FillCBTime();
            FillCBTours();
            fillTourdata();
        }
        else
        {
            PBMinus.Visible = false;
            PBPlus.Visible = false;
            PBMakePFDTourData.Visible = false;
            PBAddTour.Visible = false;
            PBAddPic.Visible = false;
            PBAddTourPic.Visible = false;
            PBDeletePic.Visible = false;
            PBPicPlus.Visible = false;
            PBPicMinus.Visible = false;
        }
        TTMouseHover.OwnerDraw = true;
        TTMouseHover.ForeColor = Color.Black;
        TTMouseHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill combobox with countries
*/
public void FillCBCountry()
{
    CBCountry.Items.Clear();
    foreach (Country i in cl.getCountryList())
    {
        CBCountry.Items.Add(i.GetCountryName());
    }
    CBCountry.SelectedIndex = k;
    c = cl.getCountryList()[k];
}
/*
fill the comboboxs with the hours and minutes times
*/
public void FillCBTime()
{
    for (int num = 0; num < 60; num++)
    {
        if (num < 10)
        {
            CBHour.Items.Add("0" + num.ToString());
            CBMinutes.Items.Add("0" + num.ToString());
            CBHourReturn.Items.Add("0" + num.ToString());
            CBMinutesReturn.Items.Add("0" + num.ToString());
        }
        else if (num < 24)
        {
            CBHour.Items.Add(num.ToString());
            CBHourReturn.Items.Add(num.ToString());
            CBMinutesReturn.Items.Add(num.ToString());
            CBMinutes.Items.Add(num.ToString());
        }
        else
        {
            CBMinutesReturn.Items.Add(num.ToString());
            CBMinutes.Items.Add(num.ToString());
        }
    }
}

```

```

        CBHour.SelectedIndex = 0;
        CBMinutes.SelectedIndex = 0;
        CBHourReturn.SelectedIndex = 0;
        CBMinutesReturn.SelectedIndex = 0;
    }
    /*
    fill the comboboxs with the tours in the database
    */
    public void FillCBTours()
    {
        CBToursList.Items.Clear();
        foreach (Tour i in tours.GetTourList())
        {
            Country cont = new Country(i.GetCountry());
            CBToursList.Items.Add(i.GetTourID() + " " + i.GetTourname() + " " +
cont.GetCountryName() + " " + i.GetPrice());
        }

        CBToursList.SelectedIndex = k;
    }
    /*
    fill the form with tour data
    */
    public void fillTourdata()
    {
        foreach (var series in TourChart.Series)
        {
            series.Points.Clear();
        }
        TourChart.Series["s1"].IsValueShownAsLabel = true;
        TourChart.Series["s1"].Points.AddXY("Taken", tours.GetTourList()[i].getTourTakenSeats());
        TourChart.Series["s1"].Points.AddXY("Empty", tours.GetTourList()[i].GetCapacity());
        TBTourID.Text = tours.GetTourList()[i].GetTourID().ToString();
        TBFlightnumber.Text = tours.GetTourList()[i].GetFlight_numberID();
        TBPrice.Text = tours.GetTourList()[i].GetPrice().ToString();
        TBDays.Text = tours.GetTourList()[i].GetDay().ToString();
        TBCapacity.Text = tours.GetTourList()[i].GetCapacity().ToString();
        TBTourName.Text = tours.GetTourList()[i].GetTourname();
        RTBDisdescription.Text = tours.GetTourList()[i].Getdisdescription();
        TBGate.Text = tours.GetTourList()[i].GetGate().ToString();
        TBRFN.Text = tours.GetTourList()[i].getFlight_number_Return().ToString();
        if (DateTime.Parse(tours.GetTourList()[i].GetDate()) > DateTime.Now)
        {
            DPTour.Value = DateTime.Parse(tours.GetTourList()[i].GetDate());
        }
        TBGate.Text= tours.GetTourList()[i].GetGateReturn().ToString();
        CBHour.SelectedIndex = int.Parse(tours.GetTourList()[i].GetTimeHM()[0].ToString()) *
10+int.Parse(tours.GetTourList()[i].GetTimeHM()[1].ToString());
        CBMinutes.SelectedIndex= int.Parse(tours.GetTourList()[i].GetTimeHM()[3].ToString()) * 10 +
int.Parse(tours.GetTourList()[i].GetTimeHM()[4].ToString());
        CBHourReturn.SelectedIndex =
int.Parse(tours.GetTourList()[i].GetTimeHMReturn()[0].ToString()) * 10 +
int.Parse(tours.GetTourList()[i].GetTimeHMReturn()[1].ToString());
        CBMinutesReturn.SelectedIndex =
int.Parse(tours.GetTourList()[i].GetTimeHMReturn()[3].ToString()) * 10 +
int.Parse(tours.GetTourList()[i].GetTimeHMReturn()[4].ToString());
    } /*add tour to the database only if the input data good*/
    private void PBAddTour_Click(object sender, EventArgs e)
    {
        bool flag = true;
        string str = "";
        f1.CheckFNumber(TBFlightnumber.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "Flight number must contain 8 chars\n";
        }
        f1.CheckFNumber(TBRFN.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "Return Flight number must contain 8 chars\n";
        }
        f1.CheckPrice(TBPrice.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "price needs to be at least with 300$\n";
        }
        f1.CheckDays(TBDays.Text);
    }

```

```

if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Enter days\n";
}
f1.CheckDays(TBCapacity.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Enter Capacity\n";
}
f1.CheckName(TBTourName.Text);
if (f1.GetAnswer() == false)
{
    flag = false;
    str += "Tour name Start with Capital letter and more then 3 letters long\n";
}
if (TBGate.Text == "" || TBGate.Text == "0" && int.Parse(TBGate.Text) < 1000)
{
    flag = false;
    str += "Enter gate number less then 1000\n";
}
if (flag == false)
{
    LError.Text = str;
}
else
{
    f1.CheckTouidup(String.Format("{0:dd/MM/yyyy}", DPTour.Value), TBTourName.Text);
    if (f1.GetAnswer() == false)
    {
        str += "There is a tour with the same name on the same date!\n";
    }
    else
    {
        f1.CheckTouidupFN(String.Format("{0:dd/MM/yyyy}", DPTour.Value),
TBFlightnumber.Text);
        if (f1.GetAnswer() == false)
        {
            str += "There is a tour with the same FlightNumber on the same date!\n";
        }
        else
        {
            f1.CheckTouidupFN(DateTime.Parse(String.Format("{0:dd/MM/yyyy}",
DPTour.Value)).AddDays(int.Parse(TBDays.Text)).ToString("dd/MM/yyyy"), TBRFN.Text);
            if (f1.GetAnswer() == false)
            {
                str += "There is a tour with the same Return FlightNumber on the same
date!\n";
            }
            else
            {
                int m = f1.MakePKID("Tours") + 1;
                Tour tr = new Tour(m, TBTourName.Text, TBFlightnumber.Text,
int.Parse(TBPrice.Text), c.GetPKID().ToString(), RTBDisdescription.Text, int.Parse(TBDays.Text),
String.Format("{0:dd/MM/yyyy}", DPTour.Value), int.Parse(TBCapacity.Text),
CBHour.SelectedItem.ToString() + ":" + CBMinutes.SelectedItem.ToString(), int.Parse(TBGate.Text),
CBHourReturn.SelectedItem.ToString() + ":" + CBMinutesReturn.SelectedItem.ToString(), TBRFN.Text,
int.Parse(TBRGate.Text));
                tr.AddTourToDB();
                LError.Text = tr.PrintTour();
                int k = 0;
                for (k = 0; k < picsToTour.GetTourPicList().Count; k++)
                {
                    int pictourid = f1.MakePKID("Pics_Tours") + 1;
                    pictour = new PicsTours(pictourid, m,
picsToTour.GetTourPicList()[k].GetPicName());
                    pictour.AddPicToDB();
                    pictourid++;
                }
                tours = new TourList();
                picsTour = new PicsTourList();
                picsToTour = new PicsTourList();
                picsToTour.SetTourPicList(picsTour.FindPicByTourID(tr.GetTourID()));
                FillCBTours();
                CBToursList.SelectedIndex = CBToursList.Items.Count - 1;
            }
        }
    }
}
}

```



```

    }
}
/*add pic to tour*/
private void PBAddTourPic_Click(object sender, EventArgs e)
{
    if (LPicName.Text.Length < 3)
    {
        LError.Text = "Add picture to tour";
    }
    else
    {
        f1.CheckPicdup(LPicName.Text, tours.GetTourList()[i].GetTourID().ToString());
        if (f1.GetAnswer() == false)
        {
            LError.Text = "The picture is in the tour\n";
        }
        else
        {
            int m = f1.MakePKID("Pics_Tours") + 1;
            PicsTours p = new PicsTours(m, t.GetTourID(), LPicName.Text);
            p.AddPicToDB();
            picsToTour.GetTourPicList().Add(p);
            picsTour.GetTourPicList().Add(p);
            LError.Text = "Pic Added\n";
            LPicName.Text = OFDAddPic.SafeFileName;
            j = picsToTour.GetTourPicList().IndexOf(p);
        }
    }
}
/*delete pic from tour*/
private void PBDeletePic_Click(object sender, EventArgs e)
{
    f1.CheckPicdup(LPicName.Text, tours.GetTourList()[i].GetTourID().ToString());
    if (f1.GetAnswer() == false)
    {
        int p = picsTour.GetTourPicList().IndexOf(picsToTour.GetTourPicList()[j]);
        picsToTour.GetTourPicList()[j].DeletePicToDB();
        picsTour.GetTourPicList().RemoveAt(p);
        picsToTour.SetTourPicList(picsTour.FindPicByTourID(t.GetTourID()));
        LPicName.Text = "";
        PBTourPic.Image = null;
    }
}
/*user pick a pic to add*/
private void PBAddPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        LPicName.Text = OFDAddPic.SafeFileName;
        if (!new System.IO.FileInfo(@"Pic\Tours\" + LPicName.Text).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Tours\" +
LPicName.Text);
        }
        PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
    }
}
/*next/last tour*/
private void PBPlus_Click(object sender, EventArgs e)
{
    i++;
    if (i == tours.GetTourList().Count)
    {
        i = 0;
    }
    t = tours.GetTourList()[i];
    CBToursList.SelectedIndex = i;
}
private void PBMinus_Click(object sender, EventArgs e)
{
    i--;
    if (i < 0)
    {
        i = tours.GetTourList().Count - 1;
    }
    t = tours.GetTourList()[i];
    CBToursList.SelectedIndex = i;
}
/*next/last pic*/
private void PBPicPlus_Click(object sender, EventArgs e)
{

```

```

        if (picsToTour.GetTourPicList().Count != 0)
        {
            j++;
            if (j >= picsToTour.GetTourPicList().Count)
            {
                j = 0;
            }
            LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
            PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
        }
    }
    private void PBPicMinus_Click(object sender, EventArgs e)
    {
        if (picsToTour.GetTourPicList().Count != 0)
        {
            j--;
            if (j < 0)
            {
                j = picsToTour.GetTourPicList().Count - 1;
            }
            LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
            PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
        }
    }
    /*slected tour changed*/
    private void CBToursList_SelectedIndexChanged(object sender, EventArgs e)
    {
        i = CBToursList.SelectedIndex;
        t = tours.GetTourList()[i];
        j = 0;
        c = new Country(t.GetCountry());
        CBCountry.SelectedItem = c.GetCountryName();
        fillTourdata();
        picsToTour.SetTourPicList(picsTour.FindPicByTourID(t.GetTourID()));
        if (picsToTour.GetTourPicList().Count != 0)
        {
            LPicName.Text = picsToTour.GetTourPicList()[j].GetPicName();
            PBTourPic.Image = Image.FromFile(@"Pic\Tours\" + LPicName.Text);
        }
        else
        {
            LPicName.Text = "";
            PBTourPic.Image = null;
        }
    }
    /*find tour by ID*/
    private void PBFID_Click(object sender, EventArgs e)
    {
        string str = "";
        if (TBTourID.Text=="")
        {
            str = "Enter Tour ID\n";
        }
        else
        {
            t = tours.FindTourbyID(TBTourID.Text);
            i = tours.GetTourList().IndexOf(t);
            if (i != -1)
            {
                CBToursList.SelectedIndex = i;
                str = "Tour Found";
            }
            else
            {
                str = "Tour with that ID dont exist";
            }
        }
        LError.Text = str;
    }
    /*Date and clock*/
    private void TDate_Tick(object sender, EventArgs e)
    {
        DateTime time = DateTime.Now;
        LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
    }
    /*Exit with fade timer*/
    private void PBExit_Click(object sender, EventArgs e)
    {

```

```

        if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            TExit.Start();
        }
    }
    private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }
    /*Go to last form*/
    private void PBBack_Click(object sender, EventArgs e)
    {
        if (LoginForm.Permission == "Worker")
        {
            var WorkerMenuForm = new WorkerMenuForm();
            WorkerMenuForm.Closed += (s, args) => this.Close();
            WorkerMenuForm.Show();
            this.Hide();
        }
        else if (LoginForm.Permission == "Manager")
        {
            var ManagerMenuForm = new ManagerMenuForm();
            ManagerMenuForm.Closed += (s, args) => this.Close();
            ManagerMenuForm.Show();
            this.Hide();
        }
    }
    /*Make PDF data file*/
    private void PBMakePFDTourData_Click(object sender, EventArgs e)
    {
        if (SFDTourPDF.ShowDialog() == DialogResult.OK)
        {
            string Path = SFDTourPDF.FileName;
            this.TourChart.SaveImage("pic/MyClientChart.png",
System.Drawing.Imaging.ImageFormat.Png);
            t.PrintTourDataPDF(System.IO.Path.GetFileNameWithoutExtension(SFDTourPDF.FileName),
"pic/MyClientChart.png", System.IO.Path.GetDirectoryName(Path));
        }
    }
    /*flight numbers, gates, price,cap and days are only digits*/
    private void TBFlightnumber_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }
    private void TBAge_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }
    private void TBPrice_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }
    private void TBDays_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }
    private void TBCapacity_KeyPress(object sender, KeyPressEventArgs e)

```

```

{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBTourID_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
private void TBGate_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}

private void TBRFN_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}

private void TBRGate_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
/*hover and tooltip withmouse*/
private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add tour", PBAddTour);
}
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.Transparent;
}

private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}

private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBAddPic_MouseHover(object sender, EventArgs e)
{
    PBAddPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Serch pic", PBAddPic);
}

private void PBAddPic_MouseLeave(object sender, EventArgs e)
{
    PBAddPic.BackColor = Color.Transparent;
}

private void PBAddTourPic_MouseHover(object sender, EventArgs e)
{

```

```

        PBAddTourPic.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Add pic", PBAddTourPic);
    }

    private void PBAddTourPic_MouseLeave(object sender, EventArgs e)
    {
        PBAddTourPic.BackColor = Color.Transparent;
    }

    private void PBPicPlus_MouseHover(object sender, EventArgs e)
    {
        PBPicPlus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Next pic", PBPicPlus);
    }

    private void PBPicPlus_MouseLeave(object sender, EventArgs e)
    {
        PBPicPlus.BackColor = Color.Transparent;
    }

    private void PBPicMinus_MouseHover(object sender, EventArgs e)
    {
        PBPicMinus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Last pic", PBPicMinus);
    }

    private void PBPicMinus_MouseLeave(object sender, EventArgs e)
    {
        PBPicMinus.BackColor = Color.Transparent;
    }

    private void PBDeletePic_MouseHover(object sender, EventArgs e)
    {
        PBDeletePic.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Delete pic", PBDeletePic);
    }

    private void PBDeletePic_MouseLeave(object sender, EventArgs e)
    {
        PBDeletePic.BackColor = Color.Transparent;
    }

    private void PBPlus_MouseHover(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Next tour", PBPlus);
    }

    private void PBPlus_MouseLeave(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.Transparent;
    }

    private void PBMinus_MouseHover(object sender, EventArgs e)
    {
        PBMinus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Last tour", PBMinus);
    }

    private void PBMinus_MouseLeave(object sender, EventArgs e)
    {
        PBMinus.BackColor = Color.Transparent;
    }

    private void PBFTID_MouseHover(object sender, EventArgs e)
    {
        PBFTID.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Find tour by ID", PBFTID);
    }

    private void PBFTID_MouseLeave(object sender, EventArgs e)
    {
        PBFTID.BackColor = Color.Transparent;
    }

    private void PBMakePFDTourData_MouseHover(object sender, EventArgs e)
    {
        PBMakePFDTourData.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Make tour PDF data", PBMakePFDTourData);
    }

```

```

    }

    private void PBMakePFDTourData_MouseLeave(object sender, EventArgs e)
    {
        PBMakePFDTourData.BackColor = Color.Transparent;
    }

    /*fade timer*/
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*tooltip background*/
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

טופס מציאת עובד

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    find worker form
    manager can find workers
    */
    public partial class FindWorkerForm : Form
    {
        sha1ceypto sh = new sha1ceypto();
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Worker w = new Worker();
        WorkerList Workers = new WorkerList();
        int i = 0;
        public FindWorkerForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";
                if (Workers.GetWorkerList().Count != 0)
                {
                    WorkerChart.Titles.Add("Worker Orders chart");
                    w = Workers.GetWorkerList()[i];
                    FillCBWorkers();
                }
                else
                {
                    PBMinus.Visible = false;
                    PBPlus.Visible = false;
                    PBUpdateWorker.Visible = false;
                    PBUpdPic.Visible = false;
                    PBON.Visible = false;
                    PBOFF.Visible = false;
                }
            }
            catch { }
        }
    }
}

```

```

        PBShowWorkerDataFile.Visible = false;
    }
    TTMouseHover.OwnerDraw = true;
    TTMouseHover.ForeColor = Color.Black;
    TTMouseHover.BackColor = Color.White;
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString(), "ERROR");
}
}
/*
fill worker combobox
*/
public void FillCBWorkers()
{
    CBWorkerList.Items.Clear();
    foreach (Worker work in Workers.GetWorkerList())
    {
        CBWorkerList.Items.Add(work.GetPKID() + " " + work.GetID() + " " + work.GetFirstName() +
" " + work.GetLastName() + " " + work.GetUserName());
    }
    CBWorkerList.SelectedIndex = 0;
}
/*
fill worker data in the form
*/
public void fillWorkerdata()
{
    foreach (var series in WorkerChart.Series)
    {
        series.Points.Clear();
    }
    WorkerChart.Series["s1"].IsValueShownAsLabel = true;
    WorkerChart.Series["s1"].Points.AddXY("Unpaid Orders", w.GetOrderCount().ToString());
    WorkerChart.Series["s1"].Points.AddXY("Paid Orders", w.GetOrderCount2().ToString());
    WorkerChart.Series["s1"].Points.AddXY("Canceled Orders", w.GetOrderCount3().ToString());
    TBAddress.Text = Workers.GetWorkerList()[i].GetAddress();
    TBUsername.Text = Workers.GetWorkerList()[i].GetUserName();
    TBLName.Text = Workers.GetWorkerList()[i].GetLastName();
    TBFName.Text = Workers.GetWorkerList()[i].GetFirstName();
    LShowDateAbsorption.Text = Workers.GetWorkerList()[i].GetADate().ToString();
    TBPSalary.Text = Workers.GetWorkerList()[i].GetSalary().ToString();
    TBCity.Text = Workers.GetWorkerList()[i].GetCity();
    LShowAge.Text = Workers.GetWorkerList()[i].ShowAge().ToString();
    TBID.Text = Workers.GetWorkerList()[i].GetID();
    PBClientPic.Image = Image.FromFile(@"Pic\Workers\" + Workers.GetWorkerList()[i].GetPic());
    LPicName.Text = Workers.GetWorkerList()[i].GetPic();
    if (Workers.GetWorkerList()[i].GetActivity())
    {
        LActiveornot.Text = "Active";
        PBON.Visible = false;
        PBOFF.Visible = true;
    }
    else
    {
        LActiveornot.Text = "Inactive";
        PBON.Visible = true;
        PBOFF.Visible = false;
    }
    if (Workers.GetWorkerList()[i].GetPKID() == 8)
    {
        PBUpDateWorker.Visible = false;
    }
    else
    {
        PBUpDateWorker.Visible = true;
    }
}
/*
date and clock
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
exit with fade timer

```

```

    */
    private void PBExit_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            TExit.Start();
        }
    }
    private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }
    /*
    to last form
    */
    private void PBBack_Click(object sender, EventArgs e)
    {
        var ManagerMenuForm = new ManagerMenuForm();
        ManagerMenuForm.Closed += (s, args) => this.Close();
        ManagerMenuForm.Show();
        this.Hide();
    }
    /*
    update pic
    */
    private void PBUpdPic_Click(object sender, EventArgs e)
    {
        if (OFDAddPic.ShowDialog() == DialogResult.OK)
        {
            LPicName.Text = OFDAddPic.SafeFileName;
            if (!new System.IO.FileInfo(@"Pic\Workers\" + LPicName.Text).Exists)
            {
                System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Workers\" +
LPicName.Text);
            }
            PBClientPic.Image = Image.FromFile(@"Pic\Workers\" + LPicName.Text);
        }
    }
    /*
    update worker
    */
    private void PBUpDateClient_Click(object sender, EventArgs e)
    {
        bool flag = true;
        string str = "";
        f1.CheckUsername(TBUsername.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "User name need 8-4 chars long and first 3 chars are letters \n";
        }
        f1.CheckName(TBLName.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "Last name Start with Capital letter and more then 3 letters long\n";
        }
        f1.CheckAddress(TBAddress.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "Address must have leat 6 chars long and no number in the first 3 chars and
finish with number\n";
        }

        f1.CheckName(TBCity.Text);
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "City Start with Capital letter and more then 3 letters long\n";
        }
    }

```



```

    }
    f1.CheackSalary(TBPSalary.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Salary must be between 4500-8000\n";
    }
    if (LPicName.Text.Length < 2)
    {
        flag = false;
        str += "Add picture\n";
    }
    LError.Text = str;
    if (flag)
    {
        if (TBUsername.Text != w.GetUserName())
        {
            f1.Checkdup(TBUsername.Text, "UserName", "Clients");
            if (f1.GetAnswer() == false)
            {
                flag = false;
                str += "There is a Client with the same User name \n";
            }
        }

        if (flag)
        {
            i = Workers.GetWorkerList().IndexOf(w);
            w.SetUserName(TBUsername.Text);
            w.SetFirstName(TBFName.Text);
            w.SetLastName(TBLName.Text);
            w.SetAddress(TBAddress.Text);
            w.SetCity(TBCity.Text);
            w.SetSalary(int.Parse(TBPSalary.Text));
            w.SetPic(LPicName.Text);
            w.UpdateWorkerToDB();
            str = "User updated";
            Workers.GetWorkerList()[i] = w;
        }
        LError.Text = str;
    }
}
/*
next/last worker
*/
private void PBMinus_Click(object sender, EventArgs e)
{
    i--;
    if (i < 0)
    {
        i = Workers.GetWorkerList().Count-1;
    }
    w = Workers.GetWorkerList()[i];
    CBWorkerList.SelectedIndex = i;
}
private void PBPlus_Click(object sender, EventArgs e)
{
    i++;
    if (i == Workers.GetWorkerList().Count)
    {
        i = 0;
    }
    w = Workers.GetWorkerList()[i];
    CBWorkerList.SelectedIndex = i;
}
/*
find worker by iD, user name,
*/
private void PBFCID_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckID(TBID.Text);
    if (f1.GetAnswer() == false)
    {
        str += "Wrong ID";
    }
    else
    {
        w = Workers.FindWorkerbyID(TBID.Text);
        i = Workers.GetWorkerList().IndexOf(w);
    }
}

```

```

        if (i != -1)
        {
            CBWorkerList.SelectedIndex = i;
            str = "User Found by ID";
        }
        else
        {
            str = "User with that ID dont exist";
        }
    }
    LError.Text = str;
}
private void PBFCUN_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        str = "User name need 8-4 chars long and first 3 chars are letters \n";
    }
    else
    {
        w = Workers.FindWorkerbyUN(TBUsername.Text);
        i = Workers.GetWorkerList().IndexOf(w);
        if (i != -1)
        {
            CBWorkerList.SelectedIndex = i;
            str = "User Found by user name";
        }
        else
        {
            str = "User with that User name dont exist";
        }
    }
    LError.Text = str;
}
/*
activate/deactivate worker
*/
private void PBON_Click(object sender, EventArgs e)
{
    i = Workers.GetWorkerList().IndexOf(w);
    w.UpdateWorkerActivityToDB(true);
    Workers.GetWorkerList()[i] = w;
    fillWorkerdata();
    LError.Text = "User is now active";
}

private void PBOFF_Click(object sender, EventArgs e)
{
    i = Workers.GetWorkerList().IndexOf(w);
    w.UpdateWorkerActivityToDB(false);
    Workers.GetWorkerList()[i] = w;
    fillWorkerdata();
    LError.Text = "User is now inactive";
}
/*
make worker data PDF
*/
private void BOShowWorkerDataFile_Click(object sender, EventArgs e)
{
    if (SFDSavePDF.ShowDialog() == DialogResult.OK)
    {
        string Path = SFDSavePDF.FileName;
        this.WorkerChart.SaveImage("pic/MyClientChart.png",
System.Drawing.Imaging.ImageFormat.Png);
        w.PrintWorkerDataPDF(System.IO.Path.GetFileNameWithoutExtension(SFDSavePDF.FileName),
"pic/MyClientChart.png", System.IO.Path.GetDirectoryName(Path));
    }
}
/*
button change color when mouse hover and leave
and tooltip show the text
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{

```

```

    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}

private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBUpdPic_MouseHover(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update pic", PBUpdPic);
}

private void PBUpdPic_MouseLeave(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.Transparent;
}

private void PBUpDateWorker_MouseHover(object sender, EventArgs e)
{
    PBUpDateWorker.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update worker", PBUpDateWorker);
}

private void PBUpDateWorker_MouseLeave(object sender, EventArgs e)
{
    PBUpDateWorker.BackColor = Color.Transparent;
}

private void PBFCUN_MouseHover(object sender, EventArgs e)
{
    PBFCUN.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find worker username", PBFCUN);
}

private void PBFCUN_MouseLeave(object sender, EventArgs e)
{
    PBFCUN.BackColor = Color.Transparent;
}

private void PBFCID_MouseHover(object sender, EventArgs e)
{
    PBFCID.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find worker ID", PBFCID);
}

private void PBFCID_MouseLeave(object sender, EventArgs e)
{
    PBFCID.BackColor = Color.Transparent;
}

private void PBOFF_MouseHover(object sender, EventArgs e)
{
    PBOFF.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Deactivate worker", PBOFF);
}

private void PBOFF_MouseLeave(object sender, EventArgs e)
{
    PBOFF.BackColor = Color.Transparent;
}

private void PBON_MouseHover(object sender, EventArgs e)
{
    PBON.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Activate worker", PBON);
}

private void PBON_MouseLeave(object sender, EventArgs e)
{
    PBON.BackColor = Color.Transparent;
}

private void PBMinus_MouseHover(object sender, EventArgs e)
{
    PBMinus.BackColor = Color.WhiteSmoke;

```

```

        TTMouseHover.Show("Last worker", PBMinus);
    }

    private void PBMinus_MouseLeave(object sender, EventArgs e)
    {
        PBMinus.BackColor = Color.Transparent;
    }

    private void PBPlus_MouseHover(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Next worker", PBPlus);
    }

    private void PBPlus_MouseLeave(object sender, EventArgs e)
    {
        PBPlus.BackColor = Color.Transparent;
    }

    private void B0ShowWorkerDataFile_MouseHover(object sender, EventArgs e)
    {
        PBShowWorkerDataFile.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Make worker PDF data", PBShowWorkerDataFile);
    }

    private void B0ShowWorkerDataFile_MouseLeave(object sender, EventArgs e)
    {
        PBShowWorkerDataFile.BackColor = Color.Transparent;
    } /*
    id and salary get only digits
    */
    private void TBID_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }

    private void TBPSalary_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }

    /*
    chhange worker on combobox
    */
    private void CBWorkerList_SelectedIndexChanged(object sender, EventArgs e)
    {
        i = CBWorkerList.SelectedIndex;
        w = Workers.GetWorkerList()[i];
        fillWorkerdata();
    }
    /*
    fade timer
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tool tip draw background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    Flight ticket form
    print tickets by client orders
    */
    public partial class FlightTicketForm : Form
    {
        ReceiptList r1 = new ReceiptList();
        Flight_TicketList FTL = new Flight_TicketList();
        Flight_TicketList FTLTaken = new Flight_TicketList();
        Function f1 = new Function();
        int i = 0, j = 0;
        public FlightTicketForm()
        {
            try
            {
                InitializeComponent();
                r1.BuildReceiptsByClientForFlightTicket(FindClientForm.clientid);
                if (r1.GetReceiptsList().Count != 0)
                {
                    fillCBOrders();
                    TDate.Start();
                    LFromAT.Text = "Israel";
                    LFromBP.Text = "Israel";
                    CBClass.SelectedIndex = 0;
                    CBTours.SelectedIndex = 0;
                    CBSeats.SelectedIndex = 0;
                }
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        put tour data in the form
        */
        public void PutData()
        {
            i = CBTours.SelectedIndex;
            LDateAT.Text = r1.GetReceiptsList()[i].GetTour().GetDate();
            LDateBP.Text = r1.GetReceiptsList()[i].GetTour().GetDate();
            LCountryAT.Text = r1.GetReceiptsList()[i].GetTour().GetCountryName();
            LCountryBP.Text = r1.GetReceiptsList()[i].GetTour().GetCountryName();
            LTimeAT.Text = r1.GetReceiptsList()[i].GetTour().GetTimeHM();
            LTimeBP.Text = r1.GetReceiptsList()[i].GetTour().GetTimeHM();
            LFNAT.Text = r1.GetReceiptsList()[i].GetTour().GetFlight_numberID();
            LFNBP.Text = r1.GetReceiptsList()[i].GetTour().GetFlight_numberID();
            LGateAT.Text = r1.GetReceiptsList()[i].GetTour().GetGate().ToString();
            LGateBP.Text = r1.GetReceiptsList()[i].GetTour().GetGate().ToString();
            CBSeats.Items.Clear();
            FTL.Flight_TicketListByTourID(r1.GetReceiptsList()[i].GetTour().GetTourID());
            FTLTaken.Flight_TicketTakenListByTourID(r1.GetReceiptsList()[i].GetTour().GetTourID(),
            r1.GetReceiptsList()[i].GetOrder().GetOrderNumber());
            CBSeats.Items.Clear();
            CBSeatsTaken.Items.Clear();
            foreach (Flight_Ticket r in FTL.GetFlightTicketList())
            {
                CBSeats.Items.Add(r.GetSeat());
            }
            foreach (Flight_Ticket r in FTLTaken.GetFlightTicketList())
            {

```

```

        CBSeatsTaken.Items.Add(r.GetSeat());
    }
    if (FTLTaken.GetFlightTicketList().Count==r1.GetReceiptsList()[i].GetOrder().GetQuantity())
    {
        CBSeats.Visible = false;
        PBPrintTicket.Visible = false;
        CBSeatsTaken.Visible = true;
        PBPrintTForTaken.Visible = true;
        CBSeatsTaken.SelectedIndex = 0;

    }
    else if (FTLTaken.GetFlightTicketList().Count == 0)
    {
        CBSeats.Visible = true;
        PBPrintTicket.Visible = true;
        CBSeatsTaken.Visible = false;
        PBPrintTForTaken.Visible = false;
        CBSeats.SelectedIndex = 0;
    }
    else
    {
        CBSeats.Visible = true;
        PBPrintTicket.Visible = true;
        CBSeatsTaken.Visible = true;
        PBPrintTForTaken.Visible = true;
        CBSeats.SelectedIndex = 0;
        CBSeatsTaken.SelectedIndex = 0;
    }
}
/*
fill combobox with orders
*/
public void fillCBOrders()
{
    CBTours.Items.Clear();
    foreach (Receipt r in r1.GetReceiptsList())
    {
        CBTours.Items.Add(r.PrintDataForWorkerMenu());
    }
}
/*
date and clock
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
Exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
}
/*
go to last form
*/
private void PBBack_Click(object sender, EventArgs e)
{

```

```

var FindClientForm = new FindClientForm();
FindClientForm.Closed += (s, args) => this.Close();
FindClientForm.Show();
this.Hide();
}
/*
change items in the combo box
*/
private void CBTours_SelectedIndexChanged(object sender, EventArgs e)
{
    PutData();
}
private void CBSeats_SelectedIndexChanged(object sender, EventArgs e)
{
    j = CBSeats.SelectedIndex;
}
/*
make 2 flight tickets in and back
and save the seat for that client
*/
private void PBPrintTicket_Click(object sender, EventArgs e)
{
    if (SFDPDFTicket.ShowDialog() == DialogResult.OK)
    {
        f1.CheckName(TBPName.Text);
        if (f1.GetAnswer() == true)
        {
            string Path = SFDPDFTicket.FileName;
            rl.GetReceiptsList()[i].PrintTicketPDF(TBPName.Text,
CBClass.SelectedItem.ToString(), CBSeats.SelectedItem.ToString(),
System.IO.Path.GetFileNameWithoutExtension(SFDPDFTicket.FileName),
System.IO.Path.GetDirectoryName(Path));

FTL.GetFlightTicketList()[j].SetOrderID(rl.GetReceiptsList()[i].GetOrder().GetOrderNumber());
FTL.GetFlightTicketList()[j].Update_Flight_Ticket_TODB();
FTL.GetFlightTicketList().RemoveAt(j);
CBSeats.Items.RemoveAt(j);
PutData();
        }
    }
}
/*
remake 2 flight tickets in and back
for saved client ticket
*/
private void PBPrintTForTaken_Click(object sender, EventArgs e)
{
    f1.CheckName(TBPName.Text);
    if (f1.GetAnswer() == true)
    {
        if (SFDPDFTicket.ShowDialog() == DialogResult.OK)
        {
            string Path = SFDPDFTicket.FileName;
            rl.GetReceiptsList()[i].PrintTicketPDF(TBPName.Text,
CBClass.SelectedItem.ToString(), CBSeatsTaken.SelectedItem.ToString(),
System.IO.Path.GetFileNameWithoutExtension(SFDPDFTicket.FileName),
System.IO.Path.GetDirectoryName(Path));
        }
    }
}
/*
Hover and tooltip with mouse
*/
private void PBPrintTForTaken_MouseHover(object sender, EventArgs e)
{
    PBPrintTForTaken.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Reprint Filght Ticket", PBPrintTForTaken);
}

private void PBPrintTForTaken_MouseLeave(object sender, EventArgs e)
{
    PBPrintTForTaken.BackColor = Color.Transparent;
}

private void PBPrintTicket_MouseHover(object sender, EventArgs e)
{
    PBPrintTicket.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Print new Filght Ticket", PBPrintTicket);
}
private void PBPrintTicket_MouseLeave(object sender, EventArgs e)

```

```

{
    PBPrintTicket.BackColor = Color.Transparent;
}
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}

private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
/*
fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tooltip background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

טופס כניסה למערכת

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    /*
    this is the login form the user can login with his username and password
    */
    public partial class LoginForm : Form
    {
        public static int PKID = -1;
        public static string Permission = "";
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        Function f1 = new Function();
        Function f2 = new Function();
    }
}

```



```

public LoginForm()
{
    this.BackgroundImage = Properties.Resources.background;
    InitializeComponent();
    TDate.Start();
    FillCBpermission();
    TTMouseHover.OwnerDraw = true;
    TTMouseHover.ForeColor = Color.Black;
    TTMouseHover.BackColor = Color.White;
}
/*
fill the combobox with the permissions
*/
public void FillCBpermission()
{
    dr = connec.SandQuery("Select permission from permissions");
    while (dr.Read())
    {
        CBPermissions.Items.Add(dr["permission"].ToString());
    }
    dr.Close();
    connec.closeCon();
    CBPermissions.SelectedIndex = 0;
}
/*
exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void LoginForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to
exit", "EXIT", MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
date timer
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
login first cheack if username and password are valid
cheack if there is same user and encoded password in the right permission show message
if user still active he go to the right from else show message
if it login the user PKID and permission saved for other uses
*/
private void PBLogin_Click(object sender, EventArgs e)
{
    bool activity=false;
    LActivity.Visible = false;
    LNotLogin.Visible = false;
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer()==false)
    {
        LErrorUsername.Visible = true;
    }
    else
    {
        LErrorUsername.Visible = false;
    }
    f2.CheckPassword(TBPassword.Text);
    if (f2.GetAnswer() == false)
    {

```

```

        LErrorPassword.Visible = true;
    }
    else
    {
        LErrorPassword.Visible = false;
    }
    if (f1.GetAnswer() && f2.GetAnswer())
    {
        dr=connec.SandQuery("Select * from "+ CBPermissions.SelectedItem.ToString() + "s where
UserName='" + TBUsername.Text+ "' and Passw='" + sh.GetSHA1(TBPassword.Text) + "'");
        while (dr.Read())
        {
            PKID = int.Parse(dr["PKID"].ToString());
            activity = bool.Parse(dr["active"].ToString());
        }
        dr.Close();
        connec.closeCon();

        if (PKID == -1)
        {
            LNotLogin.Visible = true;
        }
        else
        {
            if (activity)
            {
                LActivity.Visible = false;
                Permission = CBPermissions.SelectedItem.ToString();
                this.Hide();
                if (CBPermissions.SelectedItem.ToString() == "Client")
                {
                    var CForm = new ClientMenuForm();
                    CForm.Closed += (s, args) => this.Close();
                    CForm.Show();
                }
                else if (CBPermissions.SelectedItem.ToString() == "Worker")
                {
                    var WForm = new WorkerMenuForm();
                    WForm.Closed += (s, args) => this.Close();
                    WForm.Show();
                }
                else if (CBPermissions.SelectedItem.ToString() == "Manager")
                {
                    var MForm = new ManagerMenuForm();
                    MForm.Closed += (s, args) => this.Close();
                    MForm.Show();
                }
            }
            else
            {
                PKID = -1;
                LActivity.Visible = true;
            }
        }
    }
}

/*
go to add client form
*/
private void PBNewC_Click(object sender, EventArgs e)
{
    this.Hide();
    var ACForm = new AddClientForm();
    ACForm.Closed += (s, args) => this.Close();
    ACForm.Show();
}

private void PBLogin_MouseHover(object sender, EventArgs e)
{
    PBLogin.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Login", PBLogin);
}

private void PBLogin_MouseLeave(object sender, EventArgs e)
{
    PBLogin.BackColor = Color.Transparent;
}

private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
}

```

```

        TTMouseHover.Show("Exit", PBExit);
    }
    private void PBExit_MouseLeave(object sender, EventArgs e)
    {
        PBExit.BackColor = Color.Transparent;
    }

    private void PBNewC_MouseHover(object sender, EventArgs e)
    {
        PBNewC.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("New Client", PBNewC);
    }
    private void PBNewC_MouseLeave(object sender, EventArgs e)
    {
        PBNewC.BackColor = Color.Transparent;
    }
    /*
    time fade
    */
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }
    /*
    tool tip draw background
    */
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}
}

```

טופס דף מנהל

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    manager menu form manager can go to other forms
    can see close unpaid orders and oreders for pased tours
    */
    public partial class ManagerMenuForm : Form
    {
        sha1ceypto sh = new sha1ceypto();
        Function f1 = new Function();
        Manager m = new Manager( LoginForm.PKID.ToString());
        ReceiptList rl = new ReceiptList();
        public ManagerMenuForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                rl.BuildReceiptsByUnpaid();
                Lda.Text = rl.PrintReceiptsDataWorker();
                LHeader.Text = "Welcome " + m.GetFirstName() + " " + m.GetLastName() + " to I.T travel
company";

                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
            }
            catch { }
        }
    }
}

```

```

        TTHoverHover.BackColor = Color.White;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*date and clock*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*Exit with fade timer*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
}
/*go to last form*/
private void PBBack_Click(object sender, EventArgs e)
{
    var LoginForm = new LoginForm();
    LoginForm.Closed += (s, args) => this.Close();
    LoginForm.Show();
    this.Hide();
}
/*go to other form*/
private void PBUpDataManager_Click(object sender, EventArgs e)
{
    var UPManagerForm = new UpdateManagerForm();
    UPManagerForm.Closed += (s, args) => this.Close();
    UPManagerForm.Show();
    this.Hide();
}
private void PBAddCountries_Click(object sender, EventArgs e)
{
    var AddCountryForm = new AddCountryForm();
    AddCountryForm.Closed += (s, args) => this.Close();
    AddCountryForm.Show();
    this.Hide();
}
private void PBAddClient_Click(object sender, EventArgs e)
{
    var AddClientForm = new AddClientForm();
    AddClientForm.Closed += (s, args) => this.Close();
    AddClientForm.Show();
    this.Hide();
}
private void PBFindClient_Click(object sender, EventArgs e)
{
    var UPManagerClientForm = new FindClientForm();
    UPManagerClientForm.Closed += (s, args) => this.Close();
    UPManagerClientForm.Show();
    this.Hide();
}
}

```

```

private void PBAddWorker_Click(object sender, EventArgs e)
{
    var AddWorkerForm = new AddWorkerForm();
    AddWorkerForm.Closed += (s, args) => this.Close();
    AddWorkerForm.Show();
    this.Hide();
}

private void PBFindWorkers_Click(object sender, EventArgs e)
{
    var FindWorkerForm = new FindWorkerForm();
    FindWorkerForm.Closed += (s, args) => this.Close();
    FindWorkerForm.Show();
    this.Hide();
}

private void PBAddTour_Click(object sender, EventArgs e)
{
    var AddtourForm = new AddTourForm();
    AddtourForm.Closed += (s, args) => this.Close();
    AddtourForm.Show();
    this.Hide();
}

private void PBFindTour_Click(object sender, EventArgs e)
{
    var UpdateTourForm = new FindTour();
    UpdateTourForm.Closed += (s, args) => this.Close();
    UpdateTourForm.Show();
    this.Hide();
}

private void PBUpdateClient_Click(object sender, EventArgs e)
{
    var UpdateClientForm = new UpdateWorkerForm();
    UpdateClientForm.Closed += (s, args) => this.Close();
    UpdateClientForm.Show();
    this.Hide();
}

private void BtnAddClient_Click(object sender, EventArgs e)
{
    var AddClientForm = new AddClientForm();
    AddClientForm.Closed += (s, args) => this.Close();
    AddClientForm.Show();
    this.Hide();
}

private void BtnFindClient_Click(object sender, EventArgs e)
{
    var UPManagerClientForm = new FindClientForm();
    UPManagerClientForm.Closed += (s, args) => this.Close();
    UPManagerClientForm.Show();
    this.Hide();
}

private void BtnWorkers_Click(object sender, EventArgs e)
{
    var AddWorkerForm = new AddWorkerForm();
    AddWorkerForm.Closed += (s, args) => this.Close();
    AddWorkerForm.Show();
    this.Hide();
}

private void BtnFindWorker_Click(object sender, EventArgs e)
{
    var FindWorkerForm = new FindWorkerForm();
    FindWorkerForm.Closed += (s, args) => this.Close();
    FindWorkerForm.Show();
    this.Hide();
}

private void BtnAddTour_Click(object sender, EventArgs e)
{
    var AddtourForm = new AddTourForm();
    AddtourForm.Closed += (s, args) => this.Close();
    AddtourForm.Show();
    this.Hide();
}

private void BtnFindTour_Click(object sender, EventArgs e)
{
    var UpdateTourForm = new FindTour();
    UpdateTourForm.Closed += (s, args) => this.Close();
    UpdateTourForm.Show();
    this.Hide();
}
/*hover and tooltip with mouse hover*/

```

```

private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
private void PBUpDataManager_MouseHover(object sender, EventArgs e)
{
    PBUpDataManager.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update your data", PBUpDataManager);
}
private void PBUpDataManager_MouseLeave(object sender, EventArgs e)
{
    PBUpDataManager.BackColor = Color.Transparent;
}

private void PBAddCountries_MouseHover(object sender, EventArgs e)
{
    PBAddCountries.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add country", PBAddCountries);
}
private void PBAddCountries_MouseLeave(object sender, EventArgs e)
{
    PBAddCountries.BackColor = Color.Transparent;
}
private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add client", PBAddClient);
}
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.Transparent;
}

private void PBFindClient_MouseHover(object sender, EventArgs e)
{
    PBFindClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find client", PBFindClient);
}
private void PBFindClient_MouseLeave(object sender, EventArgs e)
{
    PBFindClient.BackColor = Color.Transparent;
}
private void PBAddWorker_MouseHover(object sender, EventArgs e)
{
    PBAddWorker.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add Worker", PBAddWorker);
}
private void PBAddWorker_MouseLeave(object sender, EventArgs e)
{
    PBAddWorker.BackColor = Color.Transparent;
}

private void PBFindWorkers_MouseHover(object sender, EventArgs e)
{
    PBFindWorkers.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find Worker", PBFindWorkers);
}
private void PBFindWorkers_MouseLeave(object sender, EventArgs e)
{
    PBFindWorkers.BackColor = Color.Transparent;
}

private void PBAddTour_MouseHover(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.WhiteSmoke;

```

```

        TTMouseHover.Show("Add tour", PBAddTour);
    }
    private void PBAddTour_MouseLeave(object sender, EventArgs e)
    {
        PBAddTour.BackColor = Color.Transparent;
    }

    private void PBFindTour_MouseHover(object sender, EventArgs e)
    {
        PBFindTour.BackColor = Color.WhiteSmoke;
        TTMouseHover.Show("Find tour", PBFindTour);
    }
    private void PBFindTour_MouseLeave(object sender, EventArgs e)
    {
        PBFindTour.BackColor = Color.Transparent;
    }

    /*Fade timer*/
    private void TExit_Tick(object sender, EventArgs e)
    {
        if (this.Opacity > 0.0)
        {
            this.Opacity -= 0.075;
        }
        else
        {
            TExit.Stop();
            Application.ExitThread();
        }
    }

    /*tooltip background*/
    private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
    {
        e.DrawBackground();
        e.DrawBorder();
        e.DrawText();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    update client form
    client can update his data
    */
    public partial class UpdateClientForm : Form
    {
        sha1crypto sh = new sha1crypto();
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Client c = new Client(LoginForm.PKID.ToString());
        public UpdateClientForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";//filter to get only picters
                TDate.Start();
            }
        }
    }
}

```

סופס עדכון לקוח

```

        fillClientdata();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString(), "ERROR");
    }
}
/*
fill client data in the form
*/
public void fillClientdata()
{
    TBAddress.Text = c.GeteAddress();
    TBUsername.Text = c.GetUserName();
    TBLName.Text = c.GetLastName();
    TBFName.Text = c.GetFirstName();
    TBEmail.Text = c.GetMail();
    TBPhone.Text = c.GetPhoneNumber();
    TBCity.Text = c.GetCity();
    LShowAge.Text = c.GetBirthdate();
    LShowID.Text = c.GetID();
    PBClientPic.Image = Image.FromFile(@"Pic\Clients\" + c.GetPic());
    LPicName.Text = c.GetPic();
}
/*
date and clock
*/
private void TDate_Tick(object sender, EventArgs e)
{
    DateTime time = DateTime.Now;
    LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
}
/*
Exit with fade timer
*/
private void PBExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
    {
        TExit.Start();
    }
}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    var ClientMenuForm = new ClientMenuForm();
    ClientMenuForm.Closed += (s, args) => this.Close();
    ClientMenuForm.Show();
    this.Hide();
}
/*
put new pic
*/
private void PBUpdPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        if (!new System.IO.FileInfo(@"Pic\Clients\" + OFDAddPic.SafeFileName).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Clients\" +
OFDAddPic.SafeFileName);
        }
    }
}

```



```

        PBClientPic.Image = Image.FromFile(@"Pic\Clients\" + OFDAddPic.SafeFileName);
        LPicName.Text = OFDAddPic.SafeFileName;
    }
}
/*
update client
check input first and phone number,email,username duplicate
*/
private void PBUpdateClient_Click(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "User name needs 8-4 characters long and first 3 characters are letters \n";
    }
    f1.CheckPhone(TBPhone.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Phone must have 10 digits\n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have leat 6 characters long and no number in the first 3 characters
and finish with number\n";
    }
    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckEmail(TBEmail.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Wrong Email\n";
    }
    if (LPicName.Text.Length < 2)
    {
        flag = false;
        str += "Add picture\n";
    }
    LError.Text = str;
    if (flag)
    {
        if (TBUsername.Text != c.GetUserName())
        {
            f1.Checkdup(TBUsername.Text, "UserName", "Clients");
            if (f1.GetAnswer() == false)
            {
                flag = false;
                str += "There is a Client with the same User name \n";
            }
        }
        if (TBEmail.Text != c.GetMail())
        {
            f1.Checkdup(TBEmail.Text, "Email", "Clients");
            if (f1.GetAnswer() == false)
            {
                flag = false;
                str += "There is a Client with the same Email \n";
            }
        }
    }
}

```

```

    }
    if (TBPhone.Text != c.GetPhoneNumber())
    {
        f1.Checkdup(TBPhone.Text, "PhoneNumber", "Clients");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Client with the same Phone Number \n";
        }
    }
    if (flag)
    {
        c.SetUserName(TBUsername.Text);
        c.SetFirstName(TBFName.Text);
        c.SetLastName(TBLName.Text);
        c.SetAddress(TBAddress.Text);
        c.SetCity(TBCity.Text);
        c.SetMail(TBEmail.Text);
        c.SetPhoneNumber(TBPhone.Text);
        c.SetPic(LPicName.Text);

        c.UpdateClientToDB();
        str = "User updated";
    }
    LError.Text = str;
}
}
/*
update password
*/
private void PBUpdatePassword_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckPassword(TBNP.Text);
    if (f1.GetAnswer() == false)
    {
        str += "New Password must be 10 characters long \n";
    }
    else
    {
        string pass1 = sh.GetSHA1(TBOP.Text);
        if (pass1 == c.GetPassword())
        {
            if (TBCNP.Text == TBNP.Text)
            {
                c.UpdateClientPasswordToDB(sh.GetSHA1(TBNP.Text));
                str = "Password Updated";
                c = new Client("Select * from Clients where PKID=" + LoginForm.PKID.ToString() +
""");
            }
            else
            {
                str = "New password and confirm password are not the same\n";
            }
        }
        else
        {
            str = "Old Password Wrong\n";
        }
    }
    LError.Text = str;
}
/*
phone get only digits
*/
private void TBPhone_KeyPress_1(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
/*
hover and tooltip with mouse
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)

```

```

{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}
private void PBUpdPic_MouseHover(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add Pic", PBUpdPic);
}

private void PBUpdPic_MouseLeave(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.Transparent;
}

private void PBUpdateClient_MouseHover(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update User", PBUpdateClient);
}
private void PBUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PBUpdateClient.BackColor = Color.Transparent;
}

private void PBUpdatePassword_MouseHover(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Change Password", PBUpdatePassword);
}
private void PBUpdatePassword_MouseLeave(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor = Color.Transparent;
}
}
/*
fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*tooltip background*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

```

עדכון מנהל

```

using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    update Manager
    Manager can update his data
    */
    public partial class UpdateManagerForm : Form
    {
        sha1ceypto sh = new sha1ceypto();
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Manager m = new Manager(LoginForm.PKID.ToString());
        public UpdateManagerForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg;
*.jpe; *.jfif; *.png";
                TDate.Start();
                fillWorkerdata();
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        fill manager data in form
        */
        public void fillWorkerdata()
        {
            TBAddress.Text = m.GeteAddress();
            TBUsername.Text = m.GetUserName();
            TBLName.Text = m.GetLastName();
            TBFName.Text = m.GetFirstName();
            LShowSalary.Text = m.GetSalary().ToString();
            LAddDate.Text = m.GetADate().ToString("dd/MM/yyyy");
            TBCity.Text = m.GetCity();
            LShowAge.Text = m.GetBirthdate();
            LShowID.Text = m.GetID();
            PBClientPic.Image = Image.FromFile(@"Pic\Managers\" + m.GetPic());
            LPicName.Text = m.GetPic();
            TBJob.Text = m.GetJob();
            LShowDepartment.Text = m.GetDepartment();
        }
        /*
        date and clock
        */
        private void TDate_Tick(object sender, EventArgs e)
        {
            DateTime time = DateTime.Now;
            LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
        }
        /*
        exit with fade timer
        */
        private void PBExit_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
            {
                TExit.Start();
            }
        }
        private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
            if (confirm == DialogResult.Yes)
            {
                e.Cancel = true;
                TExit.Start();
            }
            else if (confirm == DialogResult.No)

```

```

    {
        e.Cancel = true;
    }
}
/*
to last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    var ManagerMenuForm = new ManagerMenuForm();
    ManagerMenuForm.Closed += (s, args) => this.Close();
    ManagerMenuForm.Show();
    this.Hide();
}
/*
update pic
*/
private void PBUpdPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        if (!new System.IO.FileInfo(@"Pic\Managers\" + OFDAddPic.SafeFileName).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Managers\" +
OFDAddPic.SafeFileName);
        }
        PBClientPic.Image = Image.FromFile(@"Pic\Managers\" + OFDAddPic.SafeFileName);
        LPicName.Text = OFDAddPic.SafeFileName;
    }
}
/*
update data
*/
private void PBUpdateManager_Click_1(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "User name needs 8-4 characters long and first 3 characters are letters \n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBJob.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Job must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have lest 6 chars long and no number in the first 3 chars and
finish with number\n";
    }
    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City must Start with Capital letter and more then 3 letters long\n";
    }
    if (LPicName.Text.Length < 2)
    {
        flag = false;
        str += "Add picture\n";
    }
}

```

```

LError.Text = str;
if (flag)
{
    if (TBUsername.Text != m.GetUserName())
    {
        f1.Checkdup(TBUsername.Text, "UserName", "Managers");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Manager with the same User name \n";
        }
    }
    if (TBJob.Text != m.GetJob())
    {
        f1.Checkdup(TBJob.Text, "Job", "Managers");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Manager with the same Job \n";
        }
    }
    if (flag)
    {
        m.SetUserName(TBUsername.Text);
        m.SetFirstName(TBFName.Text);
        m.SetLastName(TBLName.Text);
        m.SetAddress(TBAddress.Text);
        m.SetCity(TBCity.Text);
        m.SetPic(LPicName.Text);
        m.SetJob(TBJob.Text);
        m.UpdateManagerToDB();
        str = "User updated";
    }
    LError.Text = str;
}
}
/*
update password
*/
private void PBUUpdatePassword_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckPassword(TBNP.Text);
    if (f1.GetAnswer() == false)
    {
        str += "new Password must have 10 chars long \n";
    }
    else
    {
        string pass1 = sh.GetSHA1(TBOP.Text);
        if (pass1 == m.GetPassword())
        {
            if (TBCNP.Text == TBNP.Text)
            {
                m.UpdateManagerPasswordToDB(sh.GetSHA1(TBNP.Text));
                str = "Password Updated";
                m = new Manager(LoginForm.PKID.ToString());
            }
            else
            {
                str = "New password and confirm password are not the same\n";
            }
        }
        else
        {
            str = "Old Password Wrong\n";
        }
    }
    LError.Text = str;
}
/*
hover and tooltip mouse hover
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}

private void PBExit_MouseLeave(object sender, EventArgs e)

```

```

{
    PBExit.BackColor =Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}

private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor =Color.Transparent;
}

private void PBUpdPic_MouseHover(object sender, EventArgs e)
{
    PBUpdPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update pic", PBUpdPic);
}

private void PBUpdPic_MouseLeave(object sender, EventArgs e)
{
    PBUpdPic.BackColor =Color.Transparent;
}

private void PBUpdateClient_MouseHover(object sender, EventArgs e)
{
    PBUpdateManager.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update data", PBUpdateManager);
}

private void PBUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PBUpdateManager.BackColor =Color.Transparent;
}

private void PBUpdatePassword_MouseHover(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Change password", PBUpdatePassword);
}

private void PBUpdatePassword_MouseLeave(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor =Color.Transparent;
}
/*
fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tooltip background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    /*
    update worker
    worker can update his data
    */
    public partial class UpdateWorkerForm : Form
    {
        sha1ceypto sh = new sha1ceypto();
        Function f1 = new Function();
        Myconn connec = new Myconn();
        Worker w = new Worker(LoginForm.PKID.ToString());
        public UpdateWorkerForm()
        {
            try
            {
                this.BackgroundImage = Properties.Resources.background;
                InitializeComponent();
                TDate.Start();
                OFDAddPic.Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";
                fillWorkerdata();
                TTMouseHover.OwnerDraw = true;
                TTMouseHover.ForeColor = Color.Black;
                TTMouseHover.BackColor = Color.White;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "ERROR");
            }
        }
        /*
        fill worker data in the form
        */
        public void fillWorkerdata()
        {
            TBAddress.Text = w.GeteAddress();
            TBUsername.Text = w.GetUserName();
            TBLastName.Text = w.GetLastName();
            TBFirstName.Text = w.GetFirstName();
            LShowSalary.Text = w.GetSalary().ToString();
            LAddDate.Text = w.GetADate().ToString("dd/MM/yyyy");
            TBCity.Text = w.GetCity();
            LShowAge.Text = w.GetBirthdate();
            LShowID.Text = w.GetID();
            PBClientPic.Image = Image.FromFile(@"Pic\Workers\" + w.GetPic());
            LPicName.Text = w.GetPic();
        }
        /*
        date and clock
        */
        private void TDate_Tick(object sender, EventArgs e)
        {
            DateTime time = DateTime.Now;
            LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
        }
        /*
        Exit with fade timer
        */
        private void PBExit_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
                DialogResult.Yes)
            {
                TExit.Start();
            }
        }
    }
}

```



```

}
private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
    MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        e.Cancel = true;
        TExit.Start();
    }
    else if (confirm == DialogResult.No)
    {
        e.Cancel = true;
    }
}
/*
last form
*/
private void PBBack_Click(object sender, EventArgs e)
{
    var WorkerMenuForm = new WorkerMenuForm();
    WorkerMenuForm.Closed += (s, args) => this.Close();
    WorkerMenuForm.Show();
    this.Hide();
}
/*
put new pic
*/
private void PBUpdPic_Click(object sender, EventArgs e)
{
    if (OFDAddPic.ShowDialog() == DialogResult.OK)
    {
        LPicName.Text = OFDAddPic.SafeFileName;
        if (!new System.IO.FileInfo(@"Pic\Workers\" + LPicName.Text).Exists)
        {
            System.IO.File.Copy(Path.GetFullPath(OFDAddPic.FileName), @"Pic\Workers\" +
LPicName.Text);
        }
        PBClientPic.Image = Image.FromFile(@"Pic\Workers\" + LPicName.Text);
    }
}
/*
update worker
cheack input first and username duplicate
*/
private void PBUpdateWorker_Click(object sender, EventArgs e)
{
    bool flag = true;
    string str = "";
    f1.CheckUsername(TBUsername.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "User name needs 8-4 characters long and first 3 characters are letters \n";
    }
    f1.CheckName(TBFName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "First name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckName(TBLName.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Last name must Start with Capital letter and more then 3 letters long\n";
    }
    f1.CheckAddress(TBAddress.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "Address must have lest 6 characters long and no number in the first 3 characters
and finish with number\n";
    }
    f1.CheckName(TBCity.Text);
    if (f1.GetAnswer() == false)
    {
        flag = false;
        str += "City must Start with Capital letter and more then 3 letters long\n";
    }
}

```

```

if (LPicName.Text.Length < 2)
{
    flag = false;
    str += "Add picture\n";
}
LError.Text = str;
if (flag)
{
    if (TUsername.Text != w.GetUserName())
    {
        f1.Checkdup(TUsername.Text, "UserName", "Workers");
        if (f1.GetAnswer() == false)
        {
            flag = false;
            str += "There is a Worker with the same User name \n";
        }
    }

    if (flag)
    {
        w.SetUserName(TUsername.Text);
        w.SetFirstName(TBFName.Text);
        w.SetLastName(TBLName.Text);
        w.SetAddress(TBAddress.Text);
        w.SetCity(TBCity.Text);
        w.SetPic(LPicName.Text);
        w.UpdateWorkerToDB();
        str = "User updated";
    }
    LError.Text = str;
}
}
/*
update password
*/
private void PBUpdatePassword_Click(object sender, EventArgs e)
{
    string str = "";
    f1.CheckPassword(TBNP.Text);
    if (f1.GetAnswer() == false)
    {
        str += "new Password must have 10 chars long \n";
    }
    else
    {
        string pass1 = sh.GetSHA1(TBOP.Text);
        if (pass1 == w.GetPassword())
        {
            if (TBCNP.Text == TBNP.Text)
            {
                w.UpdateWorkerPasswordToDB(sh.GetSHA1(TBNP.Text));
                str = "Password Updated";
                w = new Worker(LoginForm.PKID.ToString());
            }
            else
            {
                str = "New password and confirm password are not the same\n";
            }
        }
        else
        {
            str = "Old Password Wrong\n";
        }
    }
    LError.Text = str;
}
/*
hover and tooltip with mouse
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)

```

```

{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBUdpPic_MouseHover(object sender, EventArgs e)
{
    PBUdpPic.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update pic", PBUdpPic);
}
private void PBUdpPic_MouseLeave(object sender, EventArgs e)
{
    PBUdpPic.BackColor = Color.Transparent;
}

private void PBUpdateClient_MouseHover(object sender, EventArgs e)
{
    PBUpdateWorker.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update data", PBUpdateWorker);
}
private void PBUpdateClient_MouseLeave(object sender, EventArgs e)
{
    PBUpdateWorker.BackColor = Color.Transparent;
}

private void PBUpdatePassword_MouseHover(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Change Passwoord", PBUpdatePassword);
}
private void PBUpdatePassword_MouseLeave(object sender, EventArgs e)
{
    PBUpdatePassword.BackColor = Color.Transparent;
}
/*
dade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tooltip background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{

```

סופס דף עובד

```

/*
worker menu form
update data, add/find tour, add/find client, add country
*/
public partial class WorkerMenuForm : Form
{
    Function f1 = new Function();
    Worker w = new Worker( LoginForm.PKID.ToString());
    ReceiptList rl = new ReceiptList();
    public WorkerMenuForm()
    {
        try
        {
            this.BackgroundImage = Properties.Resources.background;
            InitializeComponent();
            TDate.Start();
            rl.BuildReceiptsByUnpaid();
            LShowWorkerData.Text = rl.PrintReceiptsDataWorker();
            LHeader.Text = "Welcome " + w.GetFirstName() + " " + w.GetLastName() + " to I.T travel
company";
            TTMouseHover.OwnerDraw = true;
            TTMouseHover.ForeColor = Color.Black;
            TTMouseHover.BackColor = Color.White;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString(), "ERROR");
        }
    }
    /*
date and clock
*/
    private void TDate_Tick(object sender, EventArgs e)
    {
        DateTime time = DateTime.Now;
        LDate.Text = time.ToString("dd-MM-yyyy HH:mm:ss");
    }
    /*
exit with fade timer
*/
    private void PBExit_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Are You sure you want to exit", "EXIT", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            TExit.Start();
        }
    }
    private void AddClientForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        DialogResult confirm = MessageBox.Show("Are You sure you want to exit", "EXIT",
MessageBoxButtons.YesNo);
        if (confirm == DialogResult.Yes)
        {
            e.Cancel = true;
            TExit.Start();
        }
        else if (confirm == DialogResult.No)
        {
            e.Cancel = true;
        }
    }
    /*
open other forms
*/
    private void PBBack_Click(object sender, EventArgs e)
    {
        var LoginForm = new LoginForm();
        LoginForm.Closed += (s, args) => this.Close();
        LoginForm.Show();
        this.Hide();
    }
    private void BtnMakeTicket_Click(object sender, EventArgs e)
    {
        var FlightTicketForm = new FlightTicketForm();
        FlightTicketForm.Closed += (s, args) => this.Close();
        FlightTicketForm.Show();
        this.Hide();
    }
}

```

```

private void PBUpDataWorker_Click(object sender, EventArgs e)
{
    var UPWorkerForm = new UpdateWorkerForm();
    UPWorkerForm.Closed += (s, args) => this.Close();
    UPWorkerForm.Show();
    this.Hide();
}
private void PBAddCountries_Click(object sender, EventArgs e)
{
    var AddCountryForm = new AddCountryForm();
    AddCountryForm.Closed += (s, args) => this.Close();
    AddCountryForm.Show();
    this.Hide();
}
private void PBAddClient_Click(object sender, EventArgs e)
{
    var AddClientForm = new AddClientForm();
    AddClientForm.Closed += (s, args) => this.Close();
    AddClientForm.Show();
    this.Hide();
}
private void PBFindClient_Click(object sender, EventArgs e)
{
    var UPWorkeFindClientForm = new FindClientForm();
    UPWorkeFindClientForm.Closed += (s, args) => this.Close();
    UPWorkeFindClientForm.Show();
    this.Hide();
}
private void PBAddTour_Click(object sender, EventArgs e)
{
    var addtour = new AddTourForm();
    addtour.Closed += (s, args) => this.Close();
    addtour.Show();
    this.Hide();
}
private void PBFindTour_Click(object sender, EventArgs e)
{
    var UpdateTourForm = new FindTour();
    UpdateTourForm.Closed += (s, args) => this.Close();
    UpdateTourForm.Show();
    this.Hide();
}
/*
mouse hover and leave
*/
private void PBExit_MouseHover(object sender, EventArgs e)
{
    PBExit.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Exit", PBExit);
}
private void PBExit_MouseLeave(object sender, EventArgs e)
{
    PBExit.BackColor = Color.Transparent;
}

private void PBBack_MouseHover(object sender, EventArgs e)
{
    PBBack.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Back", PBBack);
}
private void PBBack_MouseLeave(object sender, EventArgs e)
{
    PBBack.BackColor = Color.Transparent;
}

private void PBUpDataWorker_MouseHover(object sender, EventArgs e)
{
    PBUpDataWorker.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Update your data", PBUpDataWorker);
}
private void PBUpDataWorker_MouseLeave(object sender, EventArgs e)
{
    PBUpDataWorker.BackColor = Color.Transparent;
}

private void PBAddCountries_MouseHover(object sender, EventArgs e)
{
    PBAddCountries.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add Countries", PBAddCountries);
}

```

```

private void PBAddCountries_MouseLeave(object sender, EventArgs e)
{
    PBAddCountries.BackColor = Color.Transparent;
}

private void PBAddClient_MouseHover(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add client", PBAddClient);
}
private void PBAddClient_MouseLeave(object sender, EventArgs e)
{
    PBAddClient.BackColor = Color.Transparent;
}

private void PBFindClient_MouseHover(object sender, EventArgs e)
{
    PBFindClient.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find Clients", PBFindClient);
}
private void PBFindClient_MouseLeave(object sender, EventArgs e)
{
    PBFindClient.BackColor = Color.Transparent;
}

private void PBAddTour_MouseHover(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Add Tour", PBAddTour);
}
private void PBAddTour_MouseLeave(object sender, EventArgs e)
{
    PBAddTour.BackColor = Color.Transparent;
}

private void PBFindTour_MouseHover(object sender, EventArgs e)
{
    PBFindTour.BackColor = Color.WhiteSmoke;
    TTMouseHover.Show("Find tour", PBFindTour);
}
private void PBFindTour_MouseLeave(object sender, EventArgs e)
{
    PBFindTour.BackColor = Color.Transparent;
}
}
/*
Fade timer
*/
private void TExit_Tick(object sender, EventArgs e)
{
    if (this.Opacity > 0.0)
    {
        this.Opacity -= 0.075;
    }
    else
    {
        TExit.Stop();
        Application.ExitThread();
    }
}
/*
tool tip background
*/
private void TTMouseHover_Draw(object sender, DrawToolTipEventArgs e)
{
    e.DrawBackground();
    e.DrawBorder();
    e.DrawText();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    class Client : Person
    {
        protected string EMail;
        protected string PhoneNumber;
        /*
        empty constructor
        */
        public Client()
        {
        }
        /*
        constructor from person
        */
        public Client(int pid, string FN, string LN, string Add, string BD, string icity, string UN,
string PS, string iPic, string id, string Mail, string PN, bool a)
        : base(pid, FN, LN, Add, BD, icity, UN, PS, iPic, id, a)
        {
            EMail = Mail;
            PhoneNumber = PN;
        }
        /*
        copy constructor
        */
        public Client(Client c)
        {
            PKID = c.PKID;
            FirstName = c.FirstName;
            LastName = c.LastName;
            Address = c.Address;
            birthdate = c.birthdate;
            City = c.City;
            UserName = c.UserName;
            Password = c.Password;
            Pic = c.Pic;
            ID = c.ID;
            Activity = c.Activity;
            EMail = c.EMail;
            PhoneNumber = c.PhoneNumber;
        }
        /*
        constructor from Database
        */
        public Client(string CPKID)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("Select * from Clients where PKID=" + CPKID+";");
            while (dr.Read())
            {
                PKID= int.Parse(dr["PKID"].ToString());
                FirstName = dr["FirstName"].ToString();
                LastName = dr["LastName"].ToString();
                Address = dr["Address"].ToString();
                birthdate = dr["Birthdate"].ToString();
                City= dr["City"].ToString();
                UserName = dr["UserName"].ToString();
                Password = dr["Passw"].ToString();
                Pic = dr["Pic"].ToString();
                ID = dr["ID"].ToString();
                EMail = dr["Email"].ToString();
                PhoneNumber= dr["PhoneNumber"].ToString();
                Activity=(bool)dr["active"];
            }
            dr.Close();
        }
    }
}

```

```

        connec.closeCon();
    }
    /*
    set and get
    */
    public void SetMail(string Mail)
    {
        EMail = Mail;
    }
    public void SetPhoneNumber(string PN)
    {
        PhoneNumber = PN;
    }
    public string GetMail()
    {
        return EMail;
    }
    public string GetPhoneNumber()
    {
        return PhoneNumber;
    }
    /*
    get client count of unpaid orders
    */
    public int GetOrderCount()
    {
        int OrderCount = 0;
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("SELECT COUNT(ClientPKID) AS [counter] FROM Orders Where ClientPKID =
" + PKID + " and Payment=false and Active= True;");
        while (dr.Read())
        {
            OrderCount = int.Parse(dr["counter"].ToString());
        }
        dr.Close();
        connec.closeCon();
        return OrderCount;
    }
    /*
    get client count of paid orders
    */
    public int GetOrderCount2()
    {
        int OrderCount = 0;
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("SELECT COUNT(ClientPKID) AS [counter] FROM Orders Where ClientPKID =
" + PKID + " and Payment=True and Active= True;");
        while (dr.Read())
        {
            OrderCount = int.Parse(dr["counter"].ToString());
        }
        dr.Close();
        connec.closeCon();
        return OrderCount;
    }
    /*
    get client count of canceled orders
    */
    public int GetOrderCount3()
    {
        int OrderCount = 0;
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("SELECT COUNT(ClientPKID) AS [counter] FROM Orders Where ClientPKID =
" + PKID + " and Active= false;");
        while (dr.Read())
        {
            OrderCount = int.Parse(dr["counter"].ToString());
        }
        dr.Close();
        connec.closeCon();
        return OrderCount;
    }
    /*
    print data
    */
    public string PrintClient => base.PrintPerson()

```



```

        + string.Format("\nPhone number:" + PhoneNumber);
    /*
    add client to data base
    */
    public void AddClientToDB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("INSERT INTO Clients VALUES ( '" + UserName + "', '" +
sh.GetSHA1(Password) + "', '" + FirstName + "', '" + LastName + "', '" + Address + "', '" + birthdate +
"' , '" + City + "', '" + Pic + "', '" + ID + "', '" + EMail + "', '" + PhoneNumber + "', "+Activity+"
, "+PKID+" );");
        dr.Close();
        connec.closeCon();
    }
    /*
    update client on the data base
    */
    public void UpdateClientToDB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("UPDATE Clients SET UserName = '"+UserName+ "', FirstName = '" +
FirstName + "' , LastName = '" + LastName+ "' , Address = '"+Address + "' , City= '"+ City+ "' ,
Pic='"+Pic+ "' , Email = '"+EMail+ "' , PhoneNumber = '"+PhoneNumber + "' WHERE PKID = "+PKID+" ");
        dr.Close();
        connec.closeCon();
    }
    /*
    update client password on the data base
    */
    public void UpdateClientPasswordToDB(string newpassword)
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("UPDATE Clients SET Passw ='" + newpassword + "'" WHERE PKID = "+PKID+";
");
        dr.Close();
        connec.closeCon();
    }
    /*
    update client activity on the data base
    */
    public void UpdateClientActivityToDB(bool Active)
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("UPDATE Clients SET active =" + Active.ToString() + " WHERE PKID = " +
PKID + "; ");
        dr.Close();
        connec.closeCon();
        Activity = Active;
    }
    /*
    show age by Birthday
    */
    public new int ShowAge()
    {
        return base.ShowAge();
    }
    /*
    make client PDF data file
    */
    public void PrintClientDataPDF(string fileName, string ChartLocation, string FileLocation)
    {
        ReceiptList rl = new ReceiptList();
        Document doc = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
        PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation+"/"+fileName+
".pdf", FileMode.Create));
        int i, sum=0;
        doc.Open();

        iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance("pic/plane.png");
        image.SetAbsolutePosition(doc.LeftMargin, wri.PageSize.GetTop(doc.TopMargin)-15f);
        image.ScaleAbsolute(40f, 40f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/Clients/" + Pic);

```

```

        image.SetAbsolutePosition(wri.PageSize.GetRight(doc.RightMargin)-60f,
wri.PageSize.GetTop(doc.TopMargin)-15f);
        image.ScaleAbsolute(50f, 50f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/IT.png");
        image.ScaleAbsolute(50f, 50f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2,
wri.PageSize.GetTop(doc.TopMargin) - 15f);
        doc.Add(image);

        Paragraph p = new Paragraph(new Chunk(new iTextSharp.text.pdf.draw.LineSeparator(0.0F,
100.0F, BaseColor.BLACK, Element.ALIGN_LEFT, 1)));
        doc.Add(p);

        Paragraph paragraph1 = new Paragraph("Client Data From \nClient: "+FirstName+" "+LastName +
"\nPrint Time:" + DateTime.Now);
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);

        Font link = FontFactory.GetFont(EMail, 12, Font.UNDERLINE,BaseColor.BLUE);
        Anchor anchor = new Anchor("Email:"+EMail, link);
        doc.Add(anchor);

        Paragraph paragraph2 = new Paragraph(PrintClient);
        doc.Add(paragraph2);
        image = iTextSharp.text.Image.GetInstance(ChartLocation);
        image.ScaleAbsolute(250f, 250f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2, (PageSize.A4.Height -
image.ScaledHeight) / 2);
        doc.Add(image);
        doc.NewPage();
        doc.Add(paragraph1);
        doc.Add(p);
        PdfPTable table = new PdfPTable(8);
        PdfPCell cell = new PdfPCell(new Phrase(FirstName+" "+LastName+ " paid Orders list"));
        cell.Colspan = 9;
        cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
        table.AddCell(cell);
        table.AddCell("Order ID");
        table.AddCell("Tour ID");
        table.AddCell("Tour Name");
        table.AddCell("Tour Date");
        table.AddCell("Payment");
        table.AddCell("Price");
        table.AddCell("Quantity");
        table.AddCell("Total");
        r1.BuildReceiptsByClientpaidActive(PKID);
        int counter = r1.GetReceiptsList().Count;
        for(i=0;i<counter;i++)
        {
            table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourID().ToString());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourname());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
            table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetPaymentDate().ToString());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetPrice().ToString()+"$");
            table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
            table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity())*
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString()+"$");
            sum += (r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
        }
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell(sum.ToString()+"$");
        doc.Add(table);
        doc.NewPage();
        doc.Add(paragraph1);
        doc.Add(p);
        r1.BuildReceiptsByClientUnpaidActive(PKID);
        table = new PdfPTable(8);
        cell = new PdfPCell(new Phrase(FirstName + " " + LastName + " unpaid Orders list"));
        cell.Colspan = 9;
        cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right

```

```

table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Tour ID");
table.AddCell("Tour Name");
table.AddCell("Tour Date");
table.AddCell("Worker Name");
table.AddCell("Price");
table.AddCell("Quantity");
table.AddCell("Total");
counter = rl.GetReceiptsList().Count;
sum = 0;
for (i = 0; i < counter; i++)
{
    table.AddCell(rl.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetTourID().ToString());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetTourname());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(rl.GetReceiptsList()[i].GetWorker().GetFirstName() + " " +
rl.GetReceiptsList()[i].GetWorker().GetLastName());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetPrice().ToString() + "$");
    table.AddCell(rl.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((rl.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(rl.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += (rl.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(rl.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
rl.BuildReceiptsByClientUnpaidUnActive(PKID);
table = new PdfPTable(8);
cell = new PdfPCell(new Phrase(FirstName + " " + LastName + " Canceled Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Tour ID");
table.AddCell("Tour Name");
table.AddCell("Tour Date");
table.AddCell("Worker Name");
table.AddCell("Price");
table.AddCell("Quantity");
table.AddCell("Total");
counter = rl.GetReceiptsList().Count;
for (i = 0; i < counter; i++)
{
    table.AddCell(rl.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetTourID().ToString());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetTourname());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(rl.GetReceiptsList()[i].GetWorker().GetFirstName() + " " +
rl.GetReceiptsList()[i].GetWorker().GetLastName());
    table.AddCell(rl.GetReceiptsList()[i].GetTour().GetPrice().ToString() + "$");
    table.AddCell(rl.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((rl.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(rl.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += (rl.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(rl.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.Close();
}

```

```

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    class ClientList
    {
        public List<Client> Clients = new List<Client>();
        /*
        constructor
        */
        public ClientList()
        {
            Clients.Clear();
            Client c = new Client();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Clients ORDER BY PKID");
            while (dr.Read())
            {
                c = new Client(int.Parse(dr["PKID"].ToString()), dr["FirstName"].ToString(),
                dr["LastName"].ToString(), dr["Address"].ToString(), dr["Birthdate"].ToString(), dr["City"].ToString(),
                dr["UserName"].ToString(), dr["Passw"].ToString(), dr["Pic"].ToString(), dr["ID"].ToString(),
                dr["Email"].ToString(), dr["PhoneNumber"].ToString(), bool.Parse(dr["active"].ToString()));
                Clients.Add(c);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        Set and get
        */
        public void SetClientList(List<Client> c)
        {
            Clients = c;
        }
        public List<Client> GetClientList()
        {
            return Clients;
        }
        /*
        find client in the list
        */
        public Client FindClientbyID(string ID)
        {
            Client c = new Client();
            var result = from s in Clients where s.GetID() == ID select s;
            foreach (var Clients in result)
            {
                c = Clients;
            }
            return c;
        }
        public Client FindClientbyUN(string UN)
        {
            Client c = new Client();
            var result = from s in Clients where s.GetUserName() == UN select s;
            foreach (var Clients in result)
            {
                c = Clients;
            }
            return c;
        }
        public Client FindClientbyEMail(string Email)
        {
            Client c = new Client();
            var result = from s in Clients where s.GetMail() == Email select s;
            foreach (var Clients in result)
            {
                c = Clients;
            }
            return c;
        }
    }
}

```

```

public Client FindClientbyPN(string PN)
{
    Client c = new Client();
    var result = from s in Clients where s.GetPhoneNumber() == PN select s;
    foreach (var Clients in result)
    {
        c = Clients;
    }
    return c;
}
}
}

```

יבשת

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class Continent
    {
        private int PKID;
        private string ContinentName;
        /*
        constructor
        */
        public Continent(int ID, string CN)
        {
            PKID = ID;
            ContinentName = CN;
        }
        /*
        empty constructor
        */
        public Continent()
        {
        }
        /*
        constructor feom data base
        */
        public Continent(string CPKID)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("Select * from Continents where PKID=" + CPKID + "");
            while (dr.Read())
            {
                PKID = int.Parse(dr["PKID"].ToString());
                ContinentName = dr["ContinentName"].ToString();
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        get and set
        */
        public void SetPKID(int pk)
        {
            PKID = pk;
        }
        public void SetCountryName(string cn)
        {
            ContinentName = cn;
        }
        public int GetPKID()
        {
            return PKID;
        }
        public string GetContinentName()
        {
            return ContinentName;
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class ContinentList
    {
        public List<Continent> Continents = new List<Continent>();
        /*
        constructor
        */
        public ContinentList()
        {
            Continents.Clear();
            Continent c = new Continent();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Continents ORDER BY PKID");
            while (dr.Read())
            {
                c = new Continent(int.Parse(dr["PKID"].ToString()), dr["ContinentName"].ToString());
                Continents.Add(c);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        set and get
        */
        public void SetContinentsList(List<Continent> c)
        {
            Continents = c;
        }
        public List<Continent> GetContinentsList()
        {
            return Continents;
        }
    }
}

```

ארץ

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class Country
    {
        private int PKID;
        private int ContinentPKID;
        private string CountryName;
        /*
        constructor
        */
        public Country(int pk,int c,string cn)
        {
            PKID = pk;
            ContinentPKID = c;
            CountryName = cn;
        }
        /*
        empty constructor
        */
        public Country() { }
        /*
        constructor by data base
        */
        public Country(string CPKID)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("Select * from Countries where PKID=" + CPKID + ";");
            while (dr.Read())
            {

```

```

        PKID = int.Parse(dr["PKID"].ToString());
        ContinentPKID = int.Parse(dr["ContinentPKID"].ToString());
        CountryName = dr["CountryName"].ToString();
    }
    dr.Close();
    connec.closeCon();
}
/*
get and set
*/
public void SetPKID(int pk)
{
    PKID = pk;
}
public void SetContinentPKID(int pk)
{
    ContinentPKID = pk;
}
public void SetCountryName(string cn)
{
    CountryName = cn;
}
public int GetPKID()
{
    return PKID;
}
public int GetContinentPKID()
{
    return ContinentPKID;
}
public string GetCountryName()
{
    return CountryName;
}
/*
add Country to the data base
*/
public void AddCountryToDB()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("INSERT INTO Countries VALUES (" + PKID+" , "+ ContinentPKID + " ,
""+CountryName+""");");
    dr.Close();
    connec.closeCon();
}
/*
update Country
*/
public void UpdateCountryName()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("UPDATE Countries SET CountryName = '" + CountryName + "',
ContinentPKID="+ ContinentPKID + " WHERE PKID = " + PKID + " ");");
    dr.Close();
    connec.closeCon();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class CountryList
    {
        public List<Country> Countrys = new List<Country>();
        /*
        constructor
        */
        public CountryList()
        {
            Countrys.Clear();
            Country c = new Country();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Countries ORDER BY PKID");
            while (dr.Read())
            {
                c = new Country(int.Parse(dr["PKID"].ToString()),
int.Parse(dr["ContinentPKID"].ToString()), dr["CountryName"].ToString());
                Countrys.Add(c);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        constructor by continent
        */
        public CountryList(int ConPKID)
        {
            Countrys.Clear();
            Country c = new Country();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Countries where ContinentPKID=
"+ConPKID+" ORDER BY PKID");
            while (dr.Read())
            {
                c = new Country(int.Parse(dr["PKID"].ToString()),
int.Parse(dr["ContinentPKID"].ToString()), dr["CountryName"].ToString());
                Countrys.Add(c);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        get and set
        */
        public List<Country> getCountryList()
        {
            return Countrys;
        }
        public void SetCountryList(List<Country> c)
        {
            Countrys = c;
        }
    }
}

```



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.IO;
namespace IlanTalproTCB
{
    class Flight_Ticket
    {
        private int PKID;
        private int OrderID;
        private string Seat;
        private int TourID;
        /*
        empty constructor
        */
        public Flight_Ticket()
        { }
        /*
        constructor
        */
        public Flight_Ticket(int PID, int OID, string SeatIN, int TID)
        {
            PKID = PID;
            OrderID = OID;
            Seat = SeatIN;
            TourID = TID;
        }
        /*
        constructor from database
        */
        public Flight_Ticket(int ID)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("SELECT * FROM Flight_Ticket where PKID=" + ID + ";");
            while (dr.Read())
            {
                PKID = int.Parse(dr["PKID"].ToString());
                OrderID = int.Parse(dr["OrderID"].ToString());
                Seat = dr["Seat"].ToString();
                TourID = int.Parse(dr["TourID"].ToString());
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        get abd set
        */
        public int GetPKID()
        {
            return PKID;
        }
        public int GetOrderID()
        {
            return OrderID;
        }
        public int GetTourID()
        {
            return TourID;
        }
        public void SetSeat(string Nseat)
        {
            Seat = Nseat ;
        }
        public void SetPKID(int NPKID)
        {
            PKID=NPKID;
        }
        public void SetOrderID(int NorderID)
        {
            OrderID=NorderID;
        }
        public void SetTourID(int NTID)
        {
            TourID=NTID;
        }
    }
}

```

```

    }
    public string GetSeat()
    {
        return Seat;
    }
    /*
    update Flight Ticket in database
    */
    public void Update_Flight_Ticket_TODB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("UPDATE Flight_Ticket SET OrderID = " + OrderID + " WHERE PKID = " +
PKID + " ;");
        dr.Close();
        connec.closeCon();
    }
    /*
    add Flight Ticket to the database
    */
    public void Add_Flight_Ticket_TODB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("INSERT INTO Flight_Ticket VALUES ( '" + PKID + "',"+OrderID+", '" +
Seat + "', '" + TourID + "');");
        dr.Close();
        connec.closeCon();
    }
}
}

```

רשימת כרטיסי טיסה

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class Flight_TicketList
    {
        public List<Flight_Ticket> Flight_Tickets = new List<Flight_Ticket>();
        /*
        empty constructor
        */
        public Flight_TicketList()
        {
        }
        /*
        get and set
        */
        public List<Flight_Ticket> GetFlightTicketList()
        {
            return Flight_Tickets;
        }
        public void SetFlightTicketList(List<Flight_Ticket> ft)
        {
            Flight_Tickets = ft;
        }
        /*
        constructor by tour id
        */
        public void Flight_TicketListByTourID(int TourID)
        {
            Flight_Tickets.Clear();
            Flight_Ticket FT = new Flight_Ticket();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Flight_Ticket where TourID= " + TourID
+ "and OrderID= 0 ORDER BY PKID;");
            while (dr.Read())
            {
                FT = new Flight_Ticket(int.Parse(dr["PKID"].ToString()),
int.Parse(dr["OrderID"].ToString()), dr["Seat"].ToString(), int.Parse(dr["TourID"].ToString()));
                Flight_Tickets.Add(FT);
            }
            dr.Close();
            connec.closeCon();
        }
    }
}

```

```

/*
constractor by tour id and order id
*/
public void Flight_TicketTakenListByTourID(int TourID,int OrderID)
{
    Flight_Tickets.Clear();
    Flight_Ticket FT = new Flight_Ticket();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Flight_Ticket where TourID= " + TourID
+ "and OrderID= "+OrderID+" ORDER BY PKID;");
    while (dr.Read())
    {
        FT = new Flight_Ticket(int.Parse(dr["PKID"].ToString()),
int.Parse(dr["OrderID"].ToString()), dr["Seat"].ToString(), int.Parse(dr["TourID"].ToString()));
        Flight_Tickets.Add(FT);
    }
    dr.Close();
    connec.closeCon();
}
}
}

```

פונקציות

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;
using System.Data.OleDb;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    class Function
    {
        private bool Answer;
        /*
        constractor
        */
        public Function()
        {
            Answer = true;
        }
        public Function(bool a)
        {
            Answer = a;
        }
        /*
        get and set
        */
        public bool GetAnswer()
        {
            return Answer;
        }
        public void SetAnswer(bool a)
        {
            Answer = a;
        }
        /*
        check password
        10 chars long
        and its digits and letters only
        */
        public void CheckPassword(string Password)
        {
            if (Password.Length != 10)
            {
                Answer= false;
                return;
            }
            int i = 0;
            for (i = 0; i < 10; i++)
            {
                if (!((Password[i] >= 'A' && Password[i] <= 'Z') || (Password[i] >= 'a' && Password[i]
<= 'z') || (Password[i] >= '0' && Password[i] <= '9'))))
                {

```

```

        Answer= false;
        return;
    }
    Answer= true;
}
/*
check username
first capital letter
4-8 chars long
first 3 are letters other can be digits
*/
public void CheckUsername(string username)
{
    if (username.Length > 8 || username.Length < 4)
    {
        Answer = false;
        return;
    }
    int i = 0;
    for (i = 0; i < 3; i++)
    {
        if (!((username[i] >= 'A' && username[i] <= 'Z') || (username[i] >= 'a' && username[i]
<= 'z'))))
        {
            Answer = false;
            return;
        }
    }
    for (i = 3; i < username.Length; i++)
    {
        if (!((username[i] >= 'A' && username[i] <= 'Z') || (username[i] >= 'a' && username[i]
<= 'z') || (username[i] >= '0' && username[i] <= '9'))))
        {
            Answer = false;
            return;
        }
    }
    Answer = true;
    return;
}
/*
check Country
at least 4 chars long
first capital letter
letters only
*/
public void CheckCountry(string username)
{
    if (username.Length < 4)
    {
        Answer = false;
        return;
    }
    int i = 0;
    if (!((username[i] >= 'A' && username[i] <= 'Z') ))
    {
        Answer = false;
        return;
    }
    for (i = 1; i < username.Length; i++)
    {
        if (!((username[i] >= 'A' && username[i] <= 'Z') || (username[i] >= 'a' && username[i]
<= 'z') || (username[i] == ' ' || username[i] == '-'))))
        {
            Answer = false;
            return;
        }
    }
    Answer = true;
    return;
}
/*
check if its a real ID
*/
public void CheckID(string ID)
{
    if (ID.Length != 9)
    {
        Answer = false;
    }
}

```

```

        return;
    }
    int i = 0, sum = 0;
    for (i = 0; i < 9; i++)
    {
        if (!((ID[i] >= '0' && ID[i] <= '9')))
        {
            Answer = false;
            return;
        }
        if (i % 2 == 0)
        {
            sum += (ID[i]) - '0';
        }
        else
        {
            sum += (((ID[i]) - '0') * 2) / 10 + (((ID[i]) - '0') * 2) % 10;
        }
    }
    Answer = sum % 10 == 0;
}
/*
check phone number
10 digits long
*/
public void CheckPhone(string Phone)
{
    if (Phone.Length != 10)
    {
        Answer = false;
        return;
    }
    int i = 0;
    for (i = 0; i < 10; i++)
    {
        if (!((Phone[i] >= '0' && Phone[i] <= '9')))
        {
            Answer = false;
            return;
        }
    }
    Answer = true;
}
/*
check if the age is more then 18 years
*/
public void CheckDOB(DateTime DOB)
{
    int now = int.Parse(DateTime.Now.ToString("yyyyMMdd"));
    int dob = int.Parse(DOB.ToString("yyyyMMdd"));
    int age = (now - dob) / 10000;
    Answer = age >= 18;
}
/*
check name
at least 3 chars long
first capital letter
letters and spaces only
*/
public void CheckName(string Name)
{
    if (Name.Length < 3)
    {
        Answer = false;
        return;
    }
    if (!((Name[0] >= 'A' && Name[0] <= 'Z')))
    {
        Answer = false;
        return;
    }
    int i = 1;
    for (i = 1; i < Name.Length; i++)
    {
        if (!((Name[i] >= 'a' && Name[i] <= 'z')) && !(Name[i] == ' ') && !(Name[i] == '-'))
        {
            if (!((Name[i] >= 'A' && Name[i] <= 'Z')) && !(Name[i-1] == ' ') && !(Name[i-1] ==
'-'))

```

```

        {
            Answer = false;
            return;
        }

    }
    if (Name[i] == ' ' && Name[i-1] == ' ')
    {
        Answer = false;
        return;
    }

}
if ((Name[i-1] == ' '))
{
    Answer = false;
    return;
}
Answer = true;
}
/*
check Address
at least 5 chars long
can have a letters and digits at the and
*/
public void CheckAddress(string Address)
{
    bool flag = false;
    if (Address.Length < 5 || Address.Length == 0)
    {
        Answer = false;
        return;
    }
    int i = 0;
    for (i = 0; i < 3; i++)
    {
        if (!((Address[i] >= 'A' && Address[i] <= 'Z') || (Address[i] >= 'a' && Address[i] <=
'z'))))
        {
            Answer = false;
            return;
        }
    }
    for (i = 3; i < Address.Length; i++)
    {
        if (!((Address[i] >= 'A' && Address[i] <= 'Z') || (Address[i] >= 'a' && Address[i] <=
'z') ))
        {
            if (Address[i] != ' ')
            {
                Answer = false;
                return;
            }
            else if ( i + 1 < Address.Length && Address[i+1] >= '0' && Address[i+1] <= '9')
            {
                i++;
                break;
            }
        }
    }
    for (; i < Address.Length; i++)
    {
        flag = true;
        if (!((Address[i] >= '0' && Address[i] <= '9'))
        {
            Answer = false;
            return;
        }
    }
    Answer = flag ;
}
/*
cheack Email
*/
public void CheckEmail(string Email)
{
    Answer = false;
    Regex r = new Regex(@"^[a-zA-Z0-9_+@-]+\.[a-zA-Z0-9_+@-]+@[a-zA-Z0-9_+@-]+\.[a-zA-Z0-9_+@-]+");
    if (r.IsMatch(Email))

```

```

    {
        Answer = true;
    }
}
/*
check salary
from 4500 to 8000
*/
public void CheackSalary(string Salary)
{
    Answer = true;
    if (Salary == "")
    {
        Answer = false;
    }
    else if (int.Parse(Salary) < 4500 || int.Parse(Salary) > 8000)
    {
        Answer = false;
    }
}
/*
check duplicate data in the database
*/
public void Checkdup(string Data, string CHeader, string table)
{
    string str = "";
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("Select * from " + table + " where " + CHeader + " = '" + Data + "'");
    while (dr.Read())
    {
        str += dr[CHheader];
    }
    dr.Close();
    connec.closeCon();
    Answer = (str == "");
}
/*
check duplicate tourname on the same date
*/
public void CheckTourdup(string Datadate, string Tname)
{
    string str = "";
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("Select * from Tours where Tour_Date = '" + Datadate + "' and Tour_Name = '" + Tname + "'");
    while (dr.Read())
    {
        str += dr["Tour_Name"];
    }
    dr.Close();
    connec.closeCon();
    Answer = (str == "");
}
/*
check duplicate flight number on the same date
*/
public void CheckTourdupFN(string Datadate, string FN)
{
    string str = "";
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("Select * from Tours where Tour_Date = '" + Datadate + "' and Flight_number = '" + FN + "'");
    while (dr.Read())
    {
        str += dr["Tour_Name"];
    }
    dr.Close();
    connec.closeCon();
    Answer = (str == "");
}
/*
check duplicate of the same pic on the same tour
*/
public void CheckPicdup(string Data1, string data2)
{
    string str = "";
    OleDbDataReader dr;

```

```

Myconn connec = new Myconn();
dr = connec.SandQuery("Select * from Pics_Tours where Pic_Name = '" + Data1 + "' and TourID
=" + data2);
while (dr.Read())
{
    str += dr["Pic_Name"];
}
dr.Close();
connec.closeCon();
Answer = (str == "");
}
/*
make the next PKID
*/
public int MakePKID( string table)
{
    string str = "";
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("Select MAX(PKID) AS NPKID from " + table + ";");
    while (dr.Read())
    {
        str = dr["NPKID"].ToString();
    }
    dr.Close();
    connec.closeCon();
    if (str=="")
    {
        return 0;
    }
    return int.Parse(str);
}
/*
cheack flight number
8 chars long
*/
public void CheckFNumber(string str)
{
    if (str.Length==8 && int.Parse(str)!=0)
    {
        Answer = true;
    }
    else
    {
        Answer = false;
    }
}
/*
cheack tour price
must be more then 300$
*/
public void CheckPrice(string str)
{
    if (str.Length > 2)
    {
        if (int.Parse(str) > 299)
        {
            Answer = true;
        }
        else
        {
            Answer = false;
        }
    }
    else
    {
        Answer = false;
    }
}
/*
cheack tour days
more then 0 days
*/
public void CheckDays(string str)
{
    if (str.Length != 0 && int.Parse(str) != 0)
    {
        Answer = true;
    }
    else

```



```

        {
            Answer = false;
        }
    }
}

```

מנהל

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;

namespace IlanTalproTCB
{
    class Manager : Worker
    {
        private string Job;
        private int Department;
        /*
        empty constructor
        */
        public Manager()
        {
        }
        /*
        constructor
        */
        public Manager(int pid, string FN, string LN, string Add, string BD, string icity, string UN,
string PS, string iPic, string id, int s, string j, int dep, bool a)
            : base(pid, FN, LN, Add, BD, icity, UN, PS, iPic, id, s, a)
        {
            Job = j;
            Department = dep;
        }
        /*
        constructor by database
        */
        public Manager(string MPKID)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("Select * from Managers where PKID=" + MPKID + ";");
            while (dr.Read())
            {
                PKID = int.Parse(dr["PKID"].ToString());
                FirstName = dr["FirstName"].ToString();
                LastName = dr["LastName"].ToString();
                Address = dr["Address"].ToString();
                birthdate = dr["Birthdate"].ToString();
                City = dr["City"].ToString();
                UserName = dr["UserName"].ToString();
                Password = dr["Passw"].ToString();
                Pic = dr["Pic"].ToString();
                ID = dr["ID"].ToString();
                Salary = int.Parse(dr["Salary"].ToString());
                DateAbsorption = Convert.ToDateTime(dr["DateAbsorption"]);
                Activity = (bool)dr["active"];
                Job = dr["Job"].ToString();
                Department = int.Parse(dr["Department"].ToString());
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        get and set
        */
        public void SetJob(string j)
        {
            Job = j;
        }
        public void SetDep(int dep)
        {
            Department = dep;
        }
        public string GetJob()

```

```

    {
        return Job;
    }
    public int Getdep()
    {
        return Department;
    }
    /*
    get Department from database
    */
    public string GetDepartment()
    {
        string dep = "";
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("Select * from Departments where PKID=" + Department + ";");
        while (dr.Read())
        {
            dep = dr["Department"].ToString();
        }
        dr.Close();
        connec.closeCon();
        return dep;
    }
    /*
    update manager data
    */
    public void UpdateManagerToDB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("UPDATE Managers SET UserName = '" + UserName + "', FirstName = '" +
        FirstName + "', LastName = '" + LastName + "', Address = '" + Address + "', City= '" + City + "',
        Pic='" + Pic + "', Job='" + Job + "' WHERE PKID = " + PKID + " ");
        dr.Close();
        connec.closeCon();
    }
    /*
    update manager password
    */
    public void UpdateManagerPasswordToDB(string newpassword)
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("UPDATE Managers SET Passw ='" + newpassword + "' WHERE PKID = " +
        PKID + " ");
        dr.Close();
        connec.closeCon();
    }
    public string PrintManager => base.PrintWorker
        + string.Format(" \nJob:" + Job + " \nDepartment:" + Department);
    }
}

```

קשר לבסיס נתונים

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;

namespace IlanTalproTCB
{
    class Myconn
    {
        private OleDbConnection conn;
        /*
        open connection
        */
        public OleDbDataReader SandQuery(string str)
        {
            string strDb = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=ITDB.accdb;" + "Persist
            Security Info=False";
            conn = new OleDbConnection(strDb);
            conn.Open();
            OleDbDataReader dr;
            OleDbCommand cmd = new OleDbCommand(str, conn); //command sq;
            dr = cmd.ExecuteReader(); // pointer to table

```

```

        return dr;
    }
    /*
    empty constructor
    */
    public Myconn ()
    {

    }
    /*
    get and set
    */
    public void Setconn(OleDbConnection c)
    {
        conn = c;
    }
    public OleDbConnection Getconn()
    {
        return conn;
    }
    /*
    close connection
    */
    public void closeCon()
    {
        if (conn.State == System.Data.ConnectionState.Open)
        {
            conn.Close();
        }
    }
}
}

```

הזמנה

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    class Order
    {
        private int OrderNumber;
        private int ClientNumber;
        private int WorkerNumber;
        private int TourNumber;
        private DateTime OrderDate;
        private DateTime PaymentDate;
        private bool Payment;
        private int Quantity;
        private bool Active;
        /*
        empty constructor
        */
        public Order()
        {}
        /*
        constructor new order
        */
        public Order(int ON, int CN, int WN, int TN, bool pay, int q)
        {
            OrderNumber = ON;
            ClientNumber = CN;
            WorkerNumber = WN;
            TourNumber = TN;
            OrderDate = DateTime.Now;
            Payment = pay;
            Quantity = q;
            Active = true;
            if (Payment)
            {
                PaymentDate = DateTime.Now;
            }
            else
            {
                PaymentDate = DateTime.MinValue;
            }
        }
    }
}

```

```

}
/*
constractor
*/
public Order(int ON, int CN, int WN, int TN, bool pay, int q, DateTime od, DateTime pd, bool a)
{
    OrderNumber = ON;
    ClientNumber = CN;
    WorkerNumber = WN;
    TourNumber = TN;
    OrderDate = od;
    Payment = pay;
    Quantity = q;
    PaymentDate = pd;
    Active = a;
}
/*
get and set
*/
public void SetOrderNumber(int ON)
{
    OrderNumber = ON;
}
public void SetClientNumber(int CN)
{
    ClientNumber = CN;
}
public void SetWorkerNumber(int WN)
{
    WorkerNumber = WN;
}
public void SetTourNumber(int WN)
{
    TourNumber = WN;
}
public void SetActive(bool a)
{
    Active = a;
}
public void SetOrderDate(DateTime OD)
{
    OrderDate = OD;
}
public void SetPaymentDate(DateTime PD)
{
    PaymentDate = PD;
}
public void SetPayment(bool p)
{
    Payment = p;
}
public void SetQuantity(int q)
{
    Quantity = q;
}
public int GetOrderNumber()
{
    return OrderNumber;
}
public int GetClientNumber()
{
    return ClientNumber;
}
public int GetWorkerNumber()
{
    return WorkerNumber;
}
public int GetTourNumber()
{
    return TourNumber;
}
public DateTime GetOrderDate()
{
    return OrderDate;
}
public DateTime GetPaymentDate()
{
    return PaymentDate;
}
public bool GetPayment()

```

```

{
    return Payment;
}
public bool GetActive()
{
    return Active;
}
public int GetQuantity()
{
    return Quantity;
}
/*
print data for other uses
*/
public string PrintOrder()
{
    string str = "";
    str = "Order number:" + OrderNumber + "\nClient number:" + ClientNumber + "\nWorker number:"
+ WorkerNumber + "\nTour number" + TourNumber + "\nOrder date:" + OrderDate.ToString() + "\nPayment
date:" + PaymentDate + "\nPayment Status:" + Payment + "\nQuantity:" + Quantity;
    return str;
}
public string PrintOrderForClientPDF()
{
    string str = "";
    str = "Order number:" + OrderNumber + "\nOrder date:" + OrderDate.ToString() + "\nPayment
date:" + PaymentDate;
    return str;
}
/*
add order to database
*/
public void AddOrderToDB()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("INSERT INTO Orders VALUES ( " + OrderNumber + ", " + ClientNumber +
", " + WorkerNumber + ", " + TourNumber + ", " + OrderDate.ToString() + ", " + PaymentDate.ToString()
+ ", " + Payment + ", "+Quantity+ ", " + Active + " );");
    dr.Close();
    connec.closeCon();
}
/*
update order as paid
*/
public void PayOrder()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("UPDATE Orders SET Payment = true , PaymentDate='" + DateTime.Now +
"' WHERE PKID = " + OrderNumber + ";");
    dr.Close();
    connec.closeCon();
}
/*
update order as cancaled
*/
public void UnActiveOrder()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    Active = false;
    dr = connec.SandQuery("UPDATE Orders SET Active = False " + " WHERE PKID = " + OrderNumber
+ "; ");
    dr.Close();
    connec.closeCon();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IlanTalproTCB
{
    class Person
    {
        protected int PKID;
        protected string FirstName;
        protected string LastName;
        protected string Address;
        protected string birthdate;
        protected string City;
        protected string UserName;
        protected string Password;
        protected string Pic;
        protected string ID;
        protected bool Activity;
        /*
        empty constructor
        */
        public Person()
        {

        }
        /*
        constructor
        */
        public Person(int pid,string FN,string LN ,string Add, string BD,string icity,string UN , string
PS, string iPic,string id,bool a)
        {
            PKID = pid;
            FirstName = FN;
            LastName = LN;
            Address = Add;
            birthdate = BD;
            City = icity;
            UserName = UN;
            Password = PS;
            Pic = iPic;
            ID = id;
            Activity = a;
        }
        /*
        copy constructor
        */
        public Person(Person p)
        {
            PKID = p.PKID;
            FirstName = p.FirstName;
            LastName = p.LastName;
            Address =p.Address;
            birthdate = p.birthdate;
            City = p.City;
            UserName = p.UserName;
            Password = p.Password;
            Pic = p.Pic;
            ID = p.ID;
            Activity = p.Activity;
        }
        /*
        get and set
        */
        public void SetPKID(int pid)
        {
            PKID = pid;
        }
        public void SetActivity(bool a)
        {
            Activity = a;
        }
        public void SetFirstName(string FN)
        {
            FirstName = FN;
        }
    }
}

```

```

public void SetLastName(string LN)
{
    LastName = LN;
}
public void SetAddress(string Add)
{
    Address = Add;
}
public void Setbirthdate(string BD)
{
    birthdate = BD;
}
public void SetCity(string city)
{
    City = city;
}
public void SetUserName(string UN)
{
    UserName = UN;
}
public void SetPassword(string PS)
{
    Password = PS;
}
public void SetPic(string pic)
{
    Pic = pic;
}
public void SetID(string id)
{
    ID = id;
}
public string GetFirstName()
{
    return FirstName;
}
public string GetLastName()
{
    return LastName;
}
public string GeteAddress()
{
    return Address;
}
public string GetBirthdate()
{
    return birthdate;
}
public string GetCity()
{
    return City;
}
public string GetUserName()
{
    return UserName;
}
public string GetPassword()
{
    return Password;
}
public string GetPic()
{
    return Pic;
}
public string GetID()
{
    return ID;
}
public bool GetActivity()
{
    return Activity;
}
public int GetPKID()
{
    return PKID;
}
/*
print data
*/
public string PrintPerson()

```

```

    {
        string str = "";
        str = "PKID: " + PKID.ToString()+"\nID: " + ID + "\nName: " + FirstName+" "+ LastName +
        "\nAddress: " + Address + "\nBirthdate: " + birthdate + "\nCity: " + City + "\nUsername:"+
        UserName+"\nActivity="+Activity;
        return str;
    }
    /*
    show age
    */
    public int ShowAge()
    {
        int now = int.Parse(DateTime.Now.ToString("yyyyMMdd"));
        int dob = int.Parse(DateTime.Parse(birthdate).ToString("yyyyMMdd"));
        int age = (now - dob) / 10000;
        return age;
    }
}
}

```

רשימת תמונות לטיול

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class PicsTourList
    {
        public List<PicsTours> PicTourl = new List<PicsTours>();
        /*
        constructor
        */
        public PicsTourList()
        {
            PicTourl.Clear();
            PicsTours t = new PicsTours();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Pics_Tours");
            while (dr.Read())
            {
                t = new PicsTours(int.Parse(dr["PKID"].ToString()), int.Parse(dr["TourID"].ToString()),
                dr["Pic_Name"].ToString());
                PicTourl.Add(t);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        constructor by tour id
        */
        public PicsTourList(string TourID)
        {
            PicTourl.Clear();
            PicsTours t = new PicsTours();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Pics_Tours where TourID="+TourID);
            while (dr.Read())
            {
                t = new PicsTours(int.Parse(dr["PKID"].ToString()), int.Parse(dr["TourID"].ToString()),
                dr["Pic_Name"].ToString());
                PicTourl.Add(t);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        get and set
        */
        public void SetTourPicList(List<PicsTours> t)
        {
            PicTourl = t;
        }
        public List<PicsTours> GetTourPicList()
        {
            return PicTourl;
        }
        /*

```



```

change list by tour id
*/
public List<PicsTours> FindPicByTourID(int tourID)
{
    List<PicsTours> t = new List<PicsTours>();
    var result = from s in PicTourl where s.GetTourID() == tourID select s;
    foreach (var Tours in result)
    {
        t.Add(Tours);
    }
    return t;
}
}
}

```

תמונה לטיול

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class PicsTours
    {
        int PKID;
        int TourID;
        string PicName;
        /*
        empty constractor
        */
        public PicsTours()
        {
        }
        /*
        constractor
        */
        public PicsTours(int pid,int tid,string pic)
        {
            PKID = pid;
            TourID = tid;
            PicName = pic;
        }
        /*
        get and set
        */
        public void SetPKID(int pk)
        {
            PKID = pk;
        }
        public void SetTourID(int tid)
        {
            TourID = tid;
        }
        public void SetPicName(string pic)
        {
            PicName = pic;
        }
        public int GetPKID()
        {
            return PKID;
        }
        public int GetTourID()
        {
            return TourID;
        }
        public string GetPicName()
        {
            return PicName;
        }
        /*
        add pic to data base
        */
        public void AddPicToDB()
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            sha1ceypto sh = new sha1ceypto();

```

```

        dr = connec.SandQuery("INSERT INTO Pics_Tours VALUES ( '" + PKID + "', '" + PicName + "', "
+ TourID + " );");
        dr.Close();
        connec.closeCon();
    }
    /*
    delete pic from database
    */
    public void DeletePicToDB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("Delete from Pics_Tours where PKID = " + PKID + " ; ");
        dr.Close();
        connec.closeCon();
    }
}

```

קבלה

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    class Receipt
    {
        private Order o;
        private Client c;
        private Tour t;
        private Worker w;
        /*
        empty constractor
        */
        public Receipt() { }
        /*
        function for flight ticket pdf to put data in pdf
        */
        public static void filldataticet(PdfContentByte cb, int x, int y, string text)
        {
            cb.BeginText();
            cb.MoveText(x, y);
            cb.ShowText(text);
            cb.EndText();
        }
        /*
        constractor from database
        */
        public Receipt (int Orderpkid)
        {
            OleDbDataReader dr;
            Myconn connec = new Myconn();
            dr = connec.SandQuery("SELECT *FROM((Orders INNER JOIN Clients ON Orders.ClientPKID =
Clients.PKID) INNER JOIN Tours ON Orders.TourPKID = Tours.PKID) INNER JOIN Workers ON Orders.WorkerPKID
= Workers.PKID WHERE Orders.PKID="+Orderpkid+" );");
            while (dr.Read())
            {
                o = new Order(int.Parse(dr["Orders.PKID"].ToString()),
int.Parse(dr["ClientPKID"].ToString()), int.Parse(dr["WorkerPKID"].ToString()),
int.Parse(dr["TourPKID"].ToString()), bool.Parse(dr["Payment"].ToString()),
int.Parse(dr["Quantity"].ToString()), DateTime.Parse(dr["OrderDate"].ToString()),
DateTime.Parse(dr["PaymentDate"].ToString()), bool.Parse(dr["Orders.Active"].ToString()));
                c = new Client(int.Parse(dr["ClientPKID"].ToString()),
dr["Clients.FirstName"].ToString(), dr["Clients.LastName"].ToString(), dr["Clients.Address"].ToString(),
dr["Clients.Birthdate"].ToString(), dr["Clients.City"].ToString(), dr["Clients.UserName"].ToString(),
dr["Clients.Passw"].ToString(), dr["Clients.Pic"].ToString(), dr["Clients.ID"].ToString(),
dr["Email"].ToString(), dr["PhoneNumber"].ToString(), bool.Parse(dr["Clients.active"].ToString()));
                w = new Worker(int.Parse(dr["WorkerPKID"].ToString()),
dr["Workers.FirstName"].ToString(), dr["Workers.LastName"].ToString(), dr["Workers.Address"].ToString(),
dr["Workers.Birthdate"].ToString(), dr["Workers.City"].ToString(), dr["Workers.UserName"].ToString(),
dr["Workers.Passw"].ToString(), dr["Workers.Pic"].ToString(), dr["Workers.ID"].ToString(),
int.Parse(dr["Salary"].ToString()),
bool.Parse(dr["Workers.active"].ToString()),DateTime.Parse(dr["DateAbsorption"].ToString()));
            }
        }
    }
}

```

```

        t = new Tour(int.Parse(dr["Tours.PKID"].ToString()), dr["Tour_Name"].ToString(),
dr["Flight_number"].ToString(), int.Parse(dr["price"].ToString()), dr["Country"].ToString(),
dr["Disdescription"].ToString(), int.Parse(dr["Days"].ToString()), dr["Tour_Date"].ToString(),
int.Parse(dr["Capacity"].ToString()), dr["HandM"].ToString(),int.Parse(dr["Gate"].ToString()),
dr["HandMReturn"].ToString(), dr["Flight_number_Return"].ToString(),
int.Parse(dr["Gate_Return"].ToString()));

    }
    dr.Close();
    connec.closeCon();
}
/*
print for other uses
*/
public string printData()
{
    return (o.PrintOrder() + "\n" + c.PrintClient + "\n" + t.PrintTour() + "\n" +
w.PrintWorker);
}
public string printDataPDF()
{
    string workerData;
    if (w.GetPKID() ==4)
    {
        workerData = "Client Orderd the Tour";
    }
    else
    {
        workerData = w.PrintWorkerForClietPDF();
    }
    return ("Order:\n"+o.PrintOrderForClientPDF() + "\nTour:\n" + t.PrintTourForClientPDF() +
"\nWorker:\n" + workerData);
}
public string PrintDataForclientMenu()
{
    return ("Order num: " + o.GetOrderNumber() + " OrderDate: " + o.GetOrderDate() + " Tour
Date:" + t.GetDate() + " tour Price: " + o.GetQuantity() * t.GetPrice() + "$");
}
public string PrintDataForWorkerMenu()
{
    return ("Order num: " + o.GetOrderNumber() + " Tour Date:" + t.GetDate() + " tour Price: " +
o.GetQuantity() * t.GetPrice() + "$ Client name:" + c.GetFirstName()+ " "+c.GetLastName()+ " Client Phone:
"+c.GetPhoneNumber()+ " Client Mail: " + c.GetMail());
}
/*
get and set
*/
public Order GetOrder()
{
    return o;
}
public Client GetClient()
{
    return c;
}
public Tour GetTour()
{
    return t;
}
public Worker GetWorker()
{
    return w;
}
/*
print pdf Receipt for clint
*/
public void PrintReciptDataPDF(string fileName, string FileLocation)
{
    ReceiptList rl = new ReceiptList();
    Document doc = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
    PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation + "/" + fileName +
".pdf", FileMode.Create));
    doc.Open();

    iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance("pic/plane.png");
    image.SetAbsolutePosition(doc.LeftMargin, wri.PageSize.GetTop(doc.TopMargin) - 15f);
    image.ScaleAbsolute(40f, 40f);
    doc.Add(image);

    image = iTextSharp.text.Image.GetInstance("pic/IT.png");

```

```

        image.ScaleAbsolute(50f, 50f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2,
wri.PageSize.GetTop(doc.TopMargin) - 15f);
        doc.Add(image);

        Paragraph p = new Paragraph(new Chunk(new iTextSharp.text.pdf.draw.LineSeparator(0.0F,
100.0F, BaseColor.BLACK, Element.ALIGN_LEFT, 1)));
        doc.Add(p);

        Paragraph paragraph1 = new Paragraph("Receipt For Tour:" + t.GetTourID() + "\nfor: " +
c.GetFirstName() + " " + c.GetLastName() + " ID:" + c.GetID());
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);

        Paragraph paragraph2 = new Paragraph(printDataPDF());
        doc.Add(paragraph2);

        paragraph1 = new Paragraph("Total payment:" + o.GetQuantity()*t.GetPrice()+"$ For
"+o.GetQuantity()+" Places.");
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);
        doc.Close();
    }
    /*
    print pdf flight ticket for client for clint
    */
    public void PrintTicketPDF(string PName, string FClass, string seat, string fileName, string
FileLocation)
    {
        Country c = new Country(t.GetCountry());
        iTextSharp.text.Image imageFilePath =
iTextSharp.text.Image.GetInstance("pic/AirTicket.jpg");

        iTextSharp.text.Image jpg = iTextSharp.text.Image.GetInstance(imageFilePath);
        Document doc = new Document(iTextSharp.text.PageSize.A6.Rotate(), 0, 0, 0, 0);

        jpg.ScaleToFit(420, 620);

        jpg.Alignment = iTextSharp.text.Image.UNDERLYING;

        PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation + "/" + fileName +
".pdf", FileMode.Create));

        doc.Open();

        doc.NewPage();

        doc.Add(jpg);
        PdfContentByte cb = wri.DirectContent;
        BaseFont bf = BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.CP1252,
BaseFont.NOT_EMBEDDED);
        cb.SaveState();
        cb.SetFontAndSize(bf, 8);
        filldataticet(cb, 35, 190, PName);
        filldataticet(cb, 35, 166, FClass);
        filldataticet(cb, 35, 142, "Israel");
        filldataticet(cb, 35, 118, c.GetCountryName());
        filldataticet(cb, 170, 190, t.GetDate());
        filldataticet(cb, 170, 166, t.GetGate().ToString());
        filldataticet(cb, 170, 142, t.GetFlight_numberID());
        filldataticet(cb, 235, 190, t.GetTimeHM());
        filldataticet(cb, 235, 166, seat);
        filldataticet(cb, 320, 201, PName);
        filldataticet(cb, 320, 188, "Israel");
        filldataticet(cb, 320, 174, c.GetCountryName());
        filldataticet(cb, 312, 159, t.GetDate());
        filldataticet(cb, 320, 145, t.GetGate().ToString());
        filldataticet(cb, 320, 131, t.GetFlight_numberID());
        filldataticet(cb, 365, 159, t.GetTimeHM());
        filldataticet(cb, 366, 143, seat);
        cb.RestoreState();
        doc.NewPage();
        cb = wri.DirectContent;
        cb.SaveState();
        cb.SetFontAndSize(bf, 8);
        doc.Add(jpg);
        filldataticet(cb, 35, 190, PName);
        filldataticet(cb, 35, 166, FClass);
        filldataticet(cb, 35, 142, c.GetCountryName());
        filldataticet(cb, 35, 118, "Israel");

```

```

        filldataticet(cb, 170, 190,
(DateTime.Parse(t.GetDate()).AddDays(t.GetDay()).ToString("dd/MM/yyyy")).ToString());
        filldataticet(cb, 170, 166, t.GetGateReturn().ToString());
        filldataticet(cb, 170, 142, t.getFlight_number_Return());
        filldataticet(cb, 235, 190, t.GetTimeHMReturn().ToString());
        filldataticet(cb, 235, 166, seat);
        filldataticet(cb, 320, 201, PName);
        filldataticet(cb, 320, 188, c.GetCountryName());
        filldataticet(cb, 320, 174, "Israel");
        filldataticet(cb, 312, 159,
(DateTime.Parse(t.GetDate()).AddDays(t.GetDay()).ToString("dd/MM/yyyy")).ToString());
        filldataticet(cb, 320, 145, t.GetGateReturn().ToString());
        filldataticet(cb, 320, 131, t.getFlight_number_Return());
        filldataticet(cb, 365, 159, t.GetTimeHMReturn().ToString());
        filldataticet(cb, 366, 143, seat);
        cb.RestoreState();
        doc.Close();
    }
}
}

```

רשימת קבלות

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class ReceiptList
    {
        public List<Receipt> Receipts = new List<Receipt>();
        /*
        empty constructor
        */
        public ReceiptList()
        {
        }
        /*
        constructor by client id unpaid orders
        */
        public void BuildReceiptsByClientUnpaid(int CID)
        {
            Receipts.Clear();
            Receipt r = new Receipt();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= "+CID+ "and
Payment= false and Active= True ORDER BY PKID;");
            while (dr.Read())
            {
                r = new Receipt(int.Parse(dr["PKID"].ToString()));
                Receipts.Add(r);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        constructor by client id paid orders and in the Future
        */
        public void BuildReceiptsByClientForFlightTicket(int CID)
        {
            Receipts.Clear();
            Receipt r = new Receipt();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= " + CID + "and
Payment= True and Active= True ORDER BY PKID;");
            while (dr.Read())
            {
                r = new Receipt(int.Parse(dr["PKID"].ToString()));
                if (DateTime.Parse(r.GetTour().GetDate()) > DateTime.Now)
                {
                    Receipts.Add(r);
                }
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        constructor by worker id paid orders

```

```

/*
public void BuildReceiptsByWorkerpaidActive(int WID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where WorkerPKID= " + WID + "and
Payment= true and Active= True ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
constractor by tour id paid orders
*/
public void BuildReceiptsByTourPaid(int TID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where TourPKID= " + TID + "and
Payment= True and Active= True ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
constractor by tour id unpaid orders
*/
public void BuildReceiptsByTourUnPaidActive(int TID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where TourPKID= " + TID + "and
Payment= false and Active= True ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
constractor by tour id unactive orders
*/
public void BuildReceiptsByTourUnPaidUnActive(int TID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where TourPKID= " + TID + "and
Payment= false and Active= false ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
constractor by client id unpaid orders
*/
public void BuildReceiptsByClientUnpaidActive(int CID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= " + CID + "and
Payment= false and Active= True ORDER BY PKID;");

```

```

        while (dr.Read())
        {
            r = new Receipt(int.Parse(dr["PKID"].ToString()));
            Receipts.Add(r);
        }
        dr.Close();
        connec.closeCon();
    }
    /*
    constructor by worker id unpaid orders
    */
    public void BuildReceiptsByWorkerUnpaidActive(int WID)
    {
        Receipts.Clear();
        Receipt r = new Receipt();
        Myconn connec = new Myconn();
        OleDbDataReader dr = connec.SandQuery("Select * from Orders where WorkerPKID= " + WID + "and
Payment= false and Active= True ORDER BY PKID;");
        while (dr.Read())
        {
            r = new Receipt(int.Parse(dr["PKID"].ToString()));
            Receipts.Add(r);
        }
        dr.Close();
        connec.closeCon();
    }
    /*
    constructor by client id unactive orders
    */
    public void BuildReceiptsByClientUnpaidUnActive(int CID)
    {
        Receipts.Clear();
        Receipt r = new Receipt();
        Myconn connec = new Myconn();
        OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= " + CID + "and
Payment= false and Active= false ORDER BY PKID;");
        while (dr.Read())
        {
            r = new Receipt(int.Parse(dr["PKID"].ToString()));
            Receipts.Add(r);
        }
        dr.Close();
        connec.closeCon();
    }
    /*
    constructor by worker id unactive orders
    */
    public void BuildReceiptsByWorkerUnpaidUnActive(int WID)
    {
        Receipts.Clear();
        Receipt r = new Receipt();
        Myconn connec = new Myconn();
        OleDbDataReader dr = connec.SandQuery("Select * from Orders where WorkerPKID= " + WID + "and
Payment= false and Active= false ORDER BY PKID;");
        while (dr.Read())
        {
            r = new Receipt(int.Parse(dr["PKID"].ToString()));
            Receipts.Add(r);
        }
        dr.Close();
        connec.closeCon();
    }
    /*
    constructor by client id paid orders
    */
    public void BuildReceiptsByClientpaidActive(int CID)
    {
        Receipts.Clear();
        Receipt r = new Receipt();
        Myconn connec = new Myconn();
        OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= " + CID + "and
Payment= true and Active= True ORDER BY PKID;");
        while (dr.Read())
        {
            r = new Receipt(int.Parse(dr["PKID"].ToString()));
            Receipts.Add(r);
        }
        dr.Close();
        connec.closeCon();
    }
}

```

```

/*
constructor by client id
*/
public void BuildReceiptsByClient(int CID)
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where ClientPKID= " + CID + "
ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
constructor unpaid orders
*/
public void BuildReceiptsByUnpaid()
{
    Receipts.Clear();
    Receipt r = new Receipt();
    Myconn connec = new Myconn();
    OleDbDataReader dr = connec.SandQuery("Select * from Orders where Payment= false and Active=
True ORDER BY PKID;");
    while (dr.Read())
    {
        r = new Receipt(int.Parse(dr["PKID"].ToString()));
        Receipts.Add(r);
    }
    dr.Close();
    connec.closeCon();
}
/*
get and set
*/
public List<Receipt> GetReceiptsList()
{
    return Receipts;
}
/*
print data
*/
public string PrintReceiptsData()
{
    string s = "There is a " + Receipts.Count + " of unpaid orders please pay them or the
travaling office will charge you before I.T Tours will Charge you\n";
    foreach (Receipt i in Receipts)
    {
        s += i.PrintDataForclientMenu()+ "\n";
    }
    return s;
}
/*
print data for worker and manager show all unpaid order for tour in the next 14 days
*/
public string PrintReceiptsDataWorker()
{
    string s1 = "", s2 = "";
    DateTime d = DateTime.Now;
    d = d.AddDays(14);
    int counter1 = 0, counter2 = 0;
    foreach (Receipt i in Receipts )
    {
        if (DateTime.Parse(i.GetTour().GetDate()) > DateTime.Now &&
DateTime.Parse(i.GetTour().GetDate()) < d)
        {
            s1 += i.PrintDataForWorkerMenu() + "\n";
            counter1++;
        }
        else if((DateTime.Parse(i.GetTour().GetDate()) < DateTime.Now))
        {
            s2 += i.PrintDataForWorkerMenu() + "\n";
            counter2++;
        }
    }
}

```



```

        return "There is a:" + counter1 + " of unpaid orders for tour going in the next 2 weeks
please contact the client! \n" + s1+ "There is a:" + counter2 + " of unpaid orders for tour that passed
please contact the client please contact the client! \n"+s2;
    }
}
}

```

הצפנה

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
namespace IlanTalproTCB
{
    class sha1ceypto
    {
        SHA1CryptoServiceProvider sh = new SHA1CryptoServiceProvider();
        /*
        empty constructor
        */
        public sha1ceypto()
        {
        }
        /*
        encode to sha1
        */
        public string GetSHA1(string str)
        {
            SHA1CryptoServiceProvider sh = new SHA1CryptoServiceProvider();
            sh.ComputeHash(ASCIIEncoding.ASCII.GetBytes(str));
            byte[] re = sh.Hash;
            StringBuilder sb = new StringBuilder();
            foreach (byte b in re)
            {
                sb.Append(b.ToString("X2"));
            }
            return sb.ToString();
        }
    }
}

```

טיוט

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    class Tour
    {
        private int TourID;
        private string Tour_Name;
        private string Flight_number;
        private int Price;
        private string Country;
        private string disdescription;
        private int Days;
        private string Date;
        private int Capacity;
        private string TimeHM;
        private string TimeHMReturn;
        private int Gate;
        private string Flight_number_Return;
        private int Gate_Return;
        /*
        empty constructor
        */
        public Tour()
        {
        }
        /*
        constructor
        */
    }
}

```

```

    */
    public Tour (int TID,string tn,string fn,int Pric,string c,string dis,int daytime,string d, int
cap,string t, int g,string tReturn,string FNR,int GR)
    {
        TourID = TID;
        Tour_Name = tn;
        Flight_number = fn;
        Price = Pric;
        Country = c;
        disdescription = dis;
        Days = daytime;
        Date = d;
        Capacity = cap;
        TimeHM = t;
        Gate = g;
        TimeHMReturn = tReturn;
        Flight_number_Return = FNR;
        Gate_Return = GR;
    }
    /*
    add to the database
    */
    public void AddTourToDB()
    {
        OleDbDataReader dr;
        Flight_Ticket FT = new Flight_Ticket();
        Myconn connec = new Myconn();
        int m,i,j=1;
        char c='A';
        Function f1 = new Function();
        dr = connec.SandQuery("INSERT INTO Tours VALUES ( '" + TourID + "', '" + Tour_Name + "', '"
+ Country + "', '" + Flight_number + "', " + Price + ", " + Days + ", " + Capacity + ", '" + Date + "',
'" + disdescription + "', '" + TimeHM + "', " + Gate + ", '" + TimeHMReturn + "', '" + Flight_number_Return + "',
"+ Gate_Return + "');"");
        dr.Close();
        connec.closeCon();
        //connec = new Myconn();
        m = f1.MakePKID("Flight_Ticket") + 1;
        for (i=0; i< Capacity;i++)
        {
            FT = new Flight_Ticket(m,0, c + j.ToString(),TourID);
            FT.Add_Flight_Ticket_TODB();
            j++;
            m++;
            if (j > 12)
            {
                c++;
                j = 1;
            }
        }
    }
    /*
    update cap in the database
    */
    public void UpdateCapacity()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("UPDATE Tours SET Capacity = "+Capacity+ " WHERE PKID = "+TourID+";
");
        dr.Close();
        connec.closeCon();
    }
    /*
    set and get
    */
    public void SetTourID(int TID)
    {
        TourID = TID;
    }
    public void SetTourName(string tN)
    {
        Tour_Name = tN;
    }
    public void SetFlight_numberID(string FN)
    {
        Flight_number = FN;
    }
    public void SetPrice(int p)

```

```

{
    Price = p;
}
public void SetCountry(string c)
{
    Country = c;
}
public void Setdisdescription(string d)
{
    disdescription = d;
}
public void SetDays(int n)
{
    Days = n;
}
public void SetDate(string d)
{
    Date = d;
}
public void SetCapacity(int c)
{
    Capacity = c;
}
public void SetTimeHM(string t)
{
    TimeHM = t;
}
public void SetTimeHMReturn(string t)
{
    TimeHMReturn = t;
}
public void SetFlight_number_Return(string fnr)
{
    Flight_number_Return = fnr;
}
public void SetGate(int g)
{
    Gate = g;
}
public void SetGateReturn(int g)
{
    Gate_Return = g;
}
public int GetTourID()
{
    return TourID;
}
public string GetTourname()
{
    return Tour_Name;
}
public string GetFlight_numberID()
{
    return Flight_number;
}
public int GetPrice()
{
    return Price;
}
public string GetCountry()
{
    return Country;
}
public string GetCountryName()
{
    Country c = new Country(Country);
    return c.GetCountryName();
}
public string Getdisdescription()
{
    return disdescription;
}
public int GetDay()
{
    return Days;
}
public string GetDate()
{
    return Date;
}
}

```

```

public int GetCapacity()
{
    return Capacity;
}
public string GetTimeHM()
{
    return TimeHM;
}
public string GetTimeHMReturn()
{
    return TimeHMReturn;
}
public int GetGate()
{
    return Gate;
}
public string getFlight_number_Return()
{
    return Flight_number_Return;
}
public int GetGateReturn()
{
    return Gate_Return;
}
/*
print data for other uses
*/
public string PrintTour()
{
    string str = "";
    Country c = new Country(Country);
    Continent co=new Continent(c.GetContinentPKID().ToString());
    str = "Tour PKID is: " + TourID.ToString() + "\nTour Name is: " + Tour_Name + " \nFlight
number: " + Flight_number + " \nPrice: " + Price.ToString() + "$\nContinent:" +co.GetContinentName() + "
\nCountry: " + c.GetCountryName() + " \nDays:" + Days + " \nDate:" + Date.ToString() + " \nTime:" +
TimeHM + "\nGate:" + Gate + "\nReturn Flight number:" + Flight_number_Return + "\nreturn time:" +
TimeHMReturn + "\nreturn Gate:" + Gate_Return;
    return str;
}
public string PrintTourForClientPDF()
{
    string str = "";
    Country c = new Country(Country);
    str = "Tour PKID is: " + TourID.ToString() + "\nTour Name is: " + Tour_Name + " \nFlight
number: " + Flight_number + " \nPrice for a seat: " + Price.ToString() + "$ \nCountry: " +
c.GetCountryName() + " \nDays:" + Days + " \nDate:" + Date.ToString() + " \nTime:" + TimeHM + "\nGate:"
+ Gate + "\nReturn Flight number:" + Flight_number_Return + "\nreturn time:" + TimeHMReturn + "\nreturn
Gate:" + Gate_Return;
    return str;
}
public string PrintTourForticket()
{
    string str = "";
    Country c = new Country(Country);
    str = "PKID: " + TourID.ToString() + " Name: " + Tour_Name + " Flight number: " +
Flight_number + " Country: " + c.GetCountryName() + " Date:" + Date.ToString()+ " " + TimeHM + " Gate:" +
Gate + " Return Flight number:" + Flight_number_Return + " return time:" + TimeHMReturn + " return
Gate:" + Gate_Return;
    return str;
}
/*
get tour taken places
*/
public int getTourTakenSeats()
{
    int Count = 0;
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    try
    {
        dr = connec.SandQuery("SELECT sum(Quantity) AS [counter] FROM Orders Where TourPKID = "
+ TourID + " and Active=True;");
        while (dr.Read())
        {
            Count = int.Parse(dr["counter"].ToString());
        }
        dr.Close();
        connec.closeCon();
        return Count;
    }
}

```

```

        catch
        {
            return 0;
        }
    }/*
    make PDF file for client
    */
    public void PrintTourDataPDFClient(string fileName, string FileLocation)
    {
        int i;
        Document doc = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
        PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation + "/" + fileName +
        ".pdf", FileMode.Create));
        doc.Open();

        iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance("pic/plane.png");
        image.SetAbsolutePosition(doc.LeftMargin, wri.PageSize.GetTop(doc.TopMargin) - 15f);
        image.ScaleAbsolute(40f, 40f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/IT.png");
        image.ScaleAbsolute(50f, 50f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2,
        wri.PageSize.GetTop(doc.TopMargin) - 15f);
        doc.Add(image);

        Paragraph p = new Paragraph(new Chunk(new iTextSharp.text.pdf.draw.LineSeparator(0.0F,
        100.0F, BaseColor.BLACK, Element.ALIGN_LEFT, 1)));
        doc.Add(p);

        Paragraph paragraph1 = new Paragraph("Tour Data From \nTour: " +Tour_Name + "\nPrint Time:"
        + DateTime.Now);
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);

        Paragraph paragraph2 = new Paragraph(PrintTour());
        doc.Add(paragraph2);
        paragraph1 = new Paragraph(disdescription+"\n\n\n");
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);
        PdfPTable table = new PdfPTable(3);
        table.DefaultCell.Border = Rectangle.NO_BORDER;
        PdfPCell cell = new PdfPCell();
        cell.Colspan = 3;
        cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
        table.AddCell(cell);
        PicsTourList ptl = new PicsTourList(TourID.ToString());
        int counter = ptl.GetTourPicList().Count;
        for (i = 0; i < counter; i++)
        {
            image = iTextSharp.text.Image.GetInstance(@"pic/Tours/" +
            ptl.GetTourPicList()[i].GetPicName());
            table.AddCell(image);
        }
        table.CompleteRow();
        doc.Add(table);
        doc.Close();
    }
    /*
    make PDF file for worker and manager
    */
    public void PrintTourDataPDF(string fileName, string ChartLocation, string FileLocation)
    {
        Document doc = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
        PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation + "/" + fileName +
        ".pdf", FileMode.Create));
        int i,sum=0;
        doc.Open();

        iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance("pic/plane.png");
        image.SetAbsolutePosition(doc.LeftMargin, wri.PageSize.GetTop(doc.TopMargin) - 15f);
        image.ScaleAbsolute(40f, 40f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/IT.png");
        image.ScaleAbsolute(50f, 50f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2,
        wri.PageSize.GetTop(doc.TopMargin) - 15f);
        doc.Add(image);
    }

```

```

Paragraph p = new Paragraph(new Chunk(new iTextSharp.text.pdf.draw.LineSeparator(0.0F,
100.0F, BaseColor.BLACK, Element.ALIGN_LEFT, 1)));
doc.Add(p);

Paragraph paragraph1 = new Paragraph("Tour Data From \nTour: " + Tour_Name+"\nPrint
Time:"+DateTime.Now);
paragraph1.Alignment = Element.ALIGN_CENTER;
doc.Add(paragraph1);

Paragraph paragraph2 = new Paragraph(PrintTour());
doc.Add(paragraph2);
image = iTextSharp.text.Image.GetInstance(ChartLocation);
image.ScaleAbsolute(250f, 250f);
image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2, (PageSize.A4.Height -
image.ScaledHeight) / 2);
doc.Add(image);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
ReceiptList r1 = new ReceiptList();
PdfPTable table = new PdfPTable(8);
PdfPCell cell = new PdfPCell(new Phrase(TourID+" "+Tour_Name + " paid Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Client Name");
table.AddCell("Worker Name");
table.AddCell("Tour Date");
table.AddCell("Order Date");
table.AddCell("Payment");
table.AddCell("Quantity");
table.AddCell("Total");
r1.BuildReceiptsByTourPaid(TourID);
int counter = r1.GetReceiptsList().Count;
for (i = 0; i < counter; i++)
{
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetClient().GetFirstName().ToString()+" "+
r1.GetReceiptsList()[i].GetClient().GetLastName().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetWorker().GetFirstName()+" "+
r1.GetReceiptsList()[i].GetWorker().GetLastName());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetPaymentDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += (r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
r1.BuildReceiptsByTourUnPaidActive(TourID);
table = new PdfPTable(8);
cell = new PdfPCell(new Phrase(TourID + " " + Tour_Name + " unpaid Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Client Name");
table.AddCell("Worker Name");
table.AddCell("Tour Date");
table.AddCell("Order Date");
table.AddCell("Payment");
table.AddCell("Quantity");
table.AddCell("Total");
counter = r1.GetReceiptsList().Count;

```

```

sum = 0;
for (i = 0; i < counter; i++)
{
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetClient().GetFirstName().ToString() + " " +
r1.GetReceiptsList()[i].GetClient().GetLastName().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetWorker().GetFirstName() + " " +
r1.GetReceiptsList()[i].GetWorker().GetLastName());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetPaymentDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += (r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
r1.BuildReceiptsByClientUnpaidUnActive(TourID);
table = new PdfPTable(8);
cell = new PdfPCell(new Phrase(TourID + " " + Tour_Name + " Canceled Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Client Name");
table.AddCell("Worker Name");
table.AddCell("Tour Date");
table.AddCell("Order Date");
table.AddCell("Payment");
table.AddCell("Quantity");
table.AddCell("Total");
counter = r1.GetReceiptsList().Count;
sum = 0;
for (i = 0; i < counter; i++)
{
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetClient().GetFirstName().ToString() + " " +
r1.GetReceiptsList()[i].GetClient().GetLastName().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetWorker().GetFirstName() + " " +
r1.GetReceiptsList()[i].GetWorker().GetLastName());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetPaymentDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += (r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.Close();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
namespace IlanTalproTCB
{
    class TourList
    {
        public List<Tour> Tours = new List<Tour>();
        /*
        constructor
        */
        public TourList()
        {
            Tours.Clear();
            Tour t = new Tour();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Tours ORDER BY PKID");
            while (dr.Read())
            {
                t = new Tour(int.Parse(dr["PKID"].ToString()), dr["Tour_Name"].ToString(),
                dr["Flight_number"].ToString(), int.Parse(dr["price"].ToString()), dr["Country"].ToString(),
                dr["Disdescription"].ToString(), int.Parse(dr["Days"].ToString()), dr["Tour_Date"].ToString(),
                int.Parse(dr["Capacity"].ToString()), dr["HandM"].ToString(), int.Parse(dr["Gate"].ToString()),
                dr["HandMReturn"].ToString(), dr["Flight_number_Return"].ToString(), int.Parse(dr["Gate_Return"].ToString()));
                Tours.Add(t);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        copy constructor
        */
        public TourList(List<Tour> t1)
        {
            Tours.AddRange(t1);
        }
        /*
        put all tours in the list
        */
        public void BuildTourtlist()
        {
            Tours.Clear();
            Tour t = new Tour();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Tours ORDER BY PKID");
            while (dr.Read())
            {
                t = new Tour(int.Parse(dr["PKID"].ToString()), dr["Tour_Name"].ToString(),
                dr["Flight_number"].ToString(), int.Parse(dr["price"].ToString()), dr["Country"].ToString(),
                dr["Disdescription"].ToString(), int.Parse(dr["Days"].ToString()), dr["Tour_Date"].ToString(),
                int.Parse(dr["Capacity"].ToString()), dr["HandM"].ToString(), int.Parse(dr["Gate"].ToString()),
                dr["HandMReturn"].ToString(), dr["Flight_number_Return"].ToString(),
                int.Parse(dr["Gate_Return"].ToString()));
                Tours.Add(t);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        only Future tours
        */
        public void GetToursByDate()
        {
            TourList temp = new TourList(Tours);
            Tours.Clear();
            DateTime d = DateTime.Now;
            var result = from s in temp.GetTourList() where DateTime.Parse(s.GetDate()) > d select s;
            foreach (var tour in result)
            {
                Tours.Add(tour);
            }
        }
        /*
        only Continent tours

```



```

    */
    public TourList GetToursByContinent(int con)
    {
        TourList temp = new TourList(Tours);
        temp.GetTourList().Clear();
        DateTime d = DateTime.Now;
        CountryList cl = new CountryList(con);
        for (int i = 0; i < cl.getCountryList().Count; i++)
        {
            var result = from s in Tours where int.Parse(s.GetCountry()) ==
cl.getCountryList()[i].GetPKID() select s ;
            foreach (var tour in result)
            {
                temp.GetTourList().Add(tour);
            }
        }
        temp.SetTourList(temp.GetTourList().OrderBy(q => q.GetTourID()).ToList());
        return temp;
    }
    /*
    find tourby id
    */
    public Tour FindTourbyID(string ID)
    {
        Tour t = new Tour();
        var result = from s in Tours where s.GetTourID().ToString() == ID select s;
        foreach (var tours in result)
        {
            t = tours;
        }
        return t;
    }
    /*
    get and set
    */
    public void SetTourList(List<Tour> t)
    {
        Tours = t;
    }
    public List<Tour> GetTourList()
    {
        return Tours;
    }
}
}

```

עובד

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
namespace IlanTalproTCB
{
    class Worker : Person
    {
        protected int Salary;
        protected DateTime DateAbsorption;
        /*
        empty constructor
        */
        public Worker()
        {
        }
        /*
        constructor
        */
        public Worker(int pid, string FN, string LN, string Add, string BD, string icity, string UN,
string PS, string iPic, string id, int s, bool a)
            : base (pid, FN, LN, Add, BD, icity, UN, PS, iPic, id, a)
        {
            Salary = s;

```

```

        DateAbsorption = DateTime.Now;
    }
    public Worker(int pid, string FN, string LN, string Add, string BD, string icity, string UN,
string PS, string iPic, string id, int s, bool a,DateTime dd)
    : base(pid, FN, LN, Add, BD, icity, UN, PS, iPic, id, a)
    {
        Salary = s;
        DateAbsorption = dd;
    }
    /*
    constructor from pkid
    */
    public Worker(string CPKID)
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("Select * from Workers where PKID=" + CPKID + ";");
        while (dr.Read())
        {
            PKID = int.Parse(dr["PKID"].ToString());
            FirstName = dr["FirstName"].ToString();
            LastName = dr["LastName"].ToString();
            Address = dr["Address"].ToString();
            birthdate = dr["Birthdate"].ToString();
            City = dr["City"].ToString();
            UserName = dr["UserName"].ToString();
            Password = dr["Passw"].ToString();
            Pic = dr["Pic"].ToString();
            ID = dr["ID"].ToString();
            Salary = int.Parse(dr["Salary"].ToString());
            DateAbsorption= Convert.ToDateTime(dr["DateAbsorption"]);
            Activity = (bool)dr["active"];
        }
        dr.Close();
        connec.closeCon();
    }
    /*
    get and set
    */
    public void SetSalary(int s)
    {
        Salary = s;
    }
    public void SetDateAbs(DateTime D)
    {
        DateAbsorption = D;
    }
    public double GetSalary()
    {
        return Salary;
    }
    public DateTime GetADate()
    {
        return DateAbsorption;
    }
    /*
    print data
    */
    public string PrintWorker => base.PrintPerson()
        + string.Format("\nSalary:" + Salary+ "\nAbsorption
Date:"+DateAbsorption.ToString());
    public string PrintWorkerForClietPDF()
    {
        return "PKID:"+PKID+"\n"+"UserName:"+UserName;
    }
    /*
    update from database
    */
    public void UpdateWorkerToDB()
    {
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        sha1ceypto sh = new sha1ceypto();
        dr = connec.SandQuery("UPDATE Workers SET UserName = '" + UserName + "', FirstName = '" +
FirstName + "' , LastName = '" + LastName + "' , Address = '" + Address + "' , City= '" + City + "' ,
Pic='" + Pic + "' , Salary=" + Salary + " WHERE PKID = " + PKID + " ");
        dr.Close();
        connec.closeCon();
    }
    /*

```

```

add to database
*/
public void AddWorkerToDB()
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    sha1ceypto sh = new sha1ceypto();
    dr = connec.SandQuery("INSERT INTO Workers VALUES('"+PKID+"', '"+UserName+"', '"+Password+"',
"+FirstName+"', '"+LastName+"', '"+Address+"', '"+ birthdate + "', '"+City+"', '"+Pic+"', '"+ID+"',
"+Salary+", Now(),'+ Activity+'");");
    dr.Close();
    connec.closeCon();
}
/*
update password from database
*/
public void UpdateWorkerPasswordToDB(string newpassword)
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    sha1ceypto sh = new sha1ceypto();
    dr = connec.SandQuery("UPDATE Workers SET Passw =' " + newpassword + "' WHERE PKID = " + PKID
+ "; ");
    dr.Close();
    connec.closeCon();
}
/*
activate/deactivate
*/
public void UpdateWorkerActivityToDB(bool Active)
{
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    sha1ceypto sh = new sha1ceypto();
    dr = connec.SandQuery("UPDATE Workers SET active =" + Active.ToString() + " WHERE PKID = " +
PKID + "; ");
    dr.Close();
    connec.closeCon();
    Activity = Active;
}
/*
show age by Birthday
*/
public new int ShowAge()
{
    return base.ShowAge();
}
/*
get unpaid orders count
*/
public int GetOrderCount()
{
    int OrderCount = 0;
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("SELECT COUNT(WorkerPKID) AS [counter] FROM Orders Where WorkerPKID =
" + PKID + " and Payment=false and Active= True;");
    while (dr.Read())
    {
        OrderCount = int.Parse(dr["counter"].ToString());
    }
    dr.Close();
    connec.closeCon();
    return OrderCount;
}
/*
get paid orders count
*/
public int GetOrderCount2()
{
    int OrderCount = 0;
    OleDbDataReader dr;
    Myconn connec = new Myconn();
    dr = connec.SandQuery("SELECT COUNT(WorkerPKID) AS [counter] FROM Orders Where WorkerPKID =
" + PKID + " and Payment=True and Active= True;");
    while (dr.Read())
    {
        OrderCount = int.Parse(dr["counter"].ToString());
    }
}

```

```

        dr.Close();
        connec.closeCon();
        return OrderCount;
    } /*
    get canacelsed orders count
    */
    public int GetOrderCount3()
    {
        int OrderCount = 0;
        OleDbDataReader dr;
        Myconn connec = new Myconn();
        dr = connec.SandQuery("SELECT COUNT(WorkerPKID) AS [counter] FROM Orders Where WorkerPKID = " + PKID + " and Active= false;");
        while (dr.Read())
        {
            OrderCount = int.Parse(dr["counter"].ToString());
        }
        dr.Close();
        connec.closeCon();
        return OrderCount;
    }
    /*
    make worker pdf data
    */
    public void PrintWorkerDataPDF(string fileName, string ChartLocation, string FileLocation)
    {
        ReceiptList rl = new ReceiptList();
        Document doc = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
        PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(FileLocation + "/" + fileName + ".pdf", FileMode.Create));
        int i, sum=0;
        doc.Open();

        iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance("pic/plane.png");
        image.SetAbsolutePosition(doc.LeftMargin, wri.PageSize.GetTop(doc.TopMargin) - 15f);
        image.ScaleAbsolute(40f, 40f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/Workers/" + Pic);
        image.SetAbsolutePosition(wri.PageSize.GetRight(doc.RightMargin) - 60f, wri.PageSize.GetTop(doc.TopMargin) - 15f);
        image.ScaleAbsolute(50f, 50f);
        doc.Add(image);

        image = iTextSharp.text.Image.GetInstance("pic/IT.png");
        image.ScaleAbsolute(50f, 50f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2, wri.PageSize.GetTop(doc.TopMargin) - 15f);
        doc.Add(image);

        Paragraph p = new Paragraph(new Chunk(new iTextSharp.text.pdf.draw.LineSeparator(0.0F, 100.0F, BaseColor.BLACK, Element.ALIGN_LEFT, 1)));
        doc.Add(p);

        Paragraph paragraph1 = new Paragraph("Worker Data From \nWorker: " + FirstName + " " + LastName + "\nPrint Time:" + DateTime.Now);
        paragraph1.Alignment = Element.ALIGN_CENTER;
        doc.Add(paragraph1);

        Worker w = new Worker(PKID.ToString());
        Paragraph paragraph2 = new Paragraph(w.PrintWorker);
        doc.Add(paragraph2);
        image = iTextSharp.text.Image.GetInstance(ChartLocation);
        image.ScaleAbsolute(250f, 250f);
        image.SetAbsolutePosition((PageSize.A4.Width - image.ScaledWidth) / 2, (PageSize.A4.Height - image.ScaledHeight) / 2);
        doc.Add(image);
        doc.NewPage();
        doc.Add(paragraph1);
        doc.Add(p);
        PdfPTable table = new PdfPTable(8);
        PdfPCell cell = new PdfPCell(new Phrase(FirstName + " " + LastName + " paid Orders list"));
        cell.Colspan = 9;
        cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
        table.AddCell(cell);
        table.AddCell("Order ID");
        table.AddCell("Tour ID");
        table.AddCell("Tour Name");
        table.AddCell("Tour Date");
        table.AddCell("Payment");
    }

```

```

table.AddCell("Price");
table.AddCell("Quantity");
table.AddCell("Total");
r1.BuildReceiptsByWorkerpaidActive(PKID);
int counter = r1.GetReceiptsList().Count;
for (i = 0; i < counter; i++)
{
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourID().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourname());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetPaymentDate().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetPrice().ToString() + "$");
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum+= r1.GetReceiptsList()[i].GetOrder().GetQuantity()
*(r1.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString()+"$");
doc.Add(table);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
r1.BuildReceiptsByWorkerUnpaidActive(PKID);
table = new PdfPTable(8);
cell = new PdfPCell(new Phrase(FirstName + " " + LastName + " unpaid Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");
table.AddCell("Tour ID");
table.AddCell("Tour Name");
table.AddCell("Tour Date");
table.AddCell("Client ID");
table.AddCell("Price");
table.AddCell("Quantity");
table.AddCell("Total");
counter = r1.GetReceiptsList().Count;
sum = 0;
for (i = 0; i < counter; i++)
{
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourID().ToString());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourname());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
    table.AddCell(r1.GetReceiptsList()[i].GetClient().GetID());
    table.AddCell(r1.GetReceiptsList()[i].GetTour().GetPrice().ToString() + "$");
    table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
    table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
    sum += r1.GetReceiptsList()[i].GetOrder().GetQuantity() *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
}
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell("");
table.AddCell(sum.ToString() + "$");
doc.Add(table);
doc.NewPage();
doc.Add(paragraph1);
doc.Add(p);
r1.BuildReceiptsByWorkerUnpaidUnActive(PKID);
table = new PdfPTable(8);
cell = new PdfPCell(new Phrase(FirstName + " " + LastName + " Canceled Orders list"));
cell.Colspan = 9;
cell.HorizontalAlignment = 1; //0=Left, 1=Centre, 2=Right
table.AddCell(cell);
table.AddCell("Order ID");

```

```

        table.AddCell("Tour ID");
        table.AddCell("Tour Name");
        table.AddCell("Tour Date");
        table.AddCell("Client ID");
        table.AddCell("Price");
        table.AddCell("Quantity");
        table.AddCell("Total");
        counter = r1.GetReceiptsList().Count;
        sum = 0;
        for (i = 0; i < counter; i++)
        {
            table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetOrderNumber().ToString());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourID().ToString());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetTourname());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetDate());
            table.AddCell(r1.GetReceiptsList()[i].GetClient().GetID());
            table.AddCell(r1.GetReceiptsList()[i].GetTour().GetPrice().ToString() + "$");
            table.AddCell(r1.GetReceiptsList()[i].GetOrder().GetQuantity().ToString());
            table.AddCell(((r1.GetReceiptsList()[i].GetOrder().GetQuantity()) *
(r1.GetReceiptsList()[i].GetTour().GetPrice()).ToString() + "$");
            sum += r1.GetReceiptsList()[i].GetOrder().GetQuantity() *
(r1.GetReceiptsList()[i].GetTour().GetPrice());
        }
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell("");
        table.AddCell(sum.ToString() + "$");
        doc.Add(table);

        doc.Close();
    }
}
}

```

רשימת עובדים

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using System.Security.Cryptography;

namespace IlanTalproTCB
{
    class WorkerList
    {
        public List<Worker> Workers = new List<Worker>();
        /*
        constructor
        */
        public WorkerList()
        {
            Workers.Clear();
            Worker w = new Worker();
            Myconn connec = new Myconn();
            OleDbDataReader dr = connec.SandQuery("Select * from Workers where NOT PKID=8 ORDER BY
PKID");
            while (dr.Read())
            {
                w = new Worker(int.Parse(dr["PKID"].ToString()), dr["FirstName"].ToString(),
dr["LastName"].ToString(), dr["Address"].ToString(), dr["Birthdate"].ToString(), dr["City"].ToString(),
dr["UserName"].ToString(), dr["Passw"].ToString(), dr["Pic"].ToString(),
dr["ID"].ToString(),int.Parse(dr["Salary"].ToString()), bool.Parse(dr["active"].ToString()));
                Workers.Add(w);
            }
            dr.Close();
            connec.closeCon();
        }
        /*
        set and get
        */
        public void SetWorkerList(List<Worker> w)
        {
            Workers = w;
        }
    }
}

```

```

    }
    public List<Worker> GetWorkerList()
    {
        return Workers;
    }
    /*
    find worker by id
    */
    public Worker FindWorkerbyID(string ID)
    {
        Worker w = new Worker();
        var result = from s in Workers where s.GetID() == ID select s;
        foreach (var Workers in result)
        {
            w = Workers;
        }
        return w;
    }
    /*
    find worker by username
    */
    public Worker FindWorkerbyUN(string UN)
    {
        Worker w = new Worker();
        var result = from s in Workers where s.GetUserName() == UN select s;
        foreach (var Workers in result)
        {
            w = Workers;
        }
        return w;
    }
}
}
}

```

