מגמת הנדסת תוכנה

פרויקט גמר בתכנות ותכנון מערכות

תוכנת מנהל עסק בתחום שירותי הבריאות

שם התלמיד: דרור כהן

תעודת זהות: 214251258

כיתה: יב' 3

שם המנחה: עפרה שטרן

תאריך הגשה: מרץ 2021

מבוא

בחרתי לעשות את הפרויקט על מערכת לניהול קופת חולים מאחר שבתקופת משבר ה covid 19 נדמה היה כי מערכת הבריאות צריכה חיזוק וברצוני ללמוד על מה מבוססות מערכות הניהול של קופות החולים.

בפרויקט, ניסיתי להבין את הפעולות המרכזיות וההכרחיות של כל קופת חולים ושאפתי לבנות מערכת שתאפשר את כל הפעולות הללו ביעילות, פשטות ובצורה אינטואיטיבית לעובד.

עשיית הפרויקט עזרה לי להבין רבות אודות מורכבות המערכות של קופות החולים השונות ואני בטוח כי הידע שצברתי במהלך הפרויקט והפתרונות שידעתי ליצר עבור בעיות שונות, יעזרו לי מאוד בעתיד ויהוו ניסיון לפרויקטים עתידיים.

קריאה מהנה!

מסמך ייזום-תוכן עניינים

מסמך ייזום - תוכן עניינים

פרק ראשון: מסמך ייזום

1.הגדרת המשימה

1.1 הרקע למשימה:

שם הארגון: your best health care

תיאור הארגון: הארגון הינו קופת חולים חדשה. הממוקמת בעיר ראשל״צ.
your best health care הינה קופת חולים חדשה, המנסה להציע שירות אטרקטיבי. בקופה, צוות רופאים ואחיות מסור וחברותי הדואג לשירותי הבריאות המקצועיים ביותר

מטרת העל: רווח כספי מרבי ממתן טיפולים רפואיים.

מטרות משנה:

1. מתן שירות אדיב ללקוחות.
2. לדאוג לאיכות הטיפולים
3. למשוך לקוחות חדשים.
4. לתת לעובדים הרגשה טובה בעבודתם.
5. הפיכת תהליכי קבלת שירותים רפואיים, ניהול תורים, הצטרפות של לקוחות חדשים ופעולות נוספות למהירים וקלים, בכדי שיתאפשר שירות באיכות גבוהה.

היקפי הארגון:

כיום לעסק יש סניף אחד. סניף זה נמצא בבעלות אדם אחד.
הקופה פועלת שבעה ימים בשבוע כולל בחגים ופועלת בשיתוף פעולה עם צוות מסור של רופאים ואחיות.

המחלקה הנבדקת:

מחלקת הטיפולים והלקוחות הינה המחלקה העיקרית של העסק.

1.2 מהות המשימה:

המערכת המטפלת בכל הפרוצדורה של ניהול הליכים רפואיים מסורבלת מאד ולא נוחה לכן החליטה הנהלת העסק על הקמת מערכת ממוחשבת חדשה ומשופרת על בסיס המערכת הישנה שתעבוד באופן יעיל וממוקד יותר.

1.3 הנמקת המשימה:

- המערכת הנוכחית כעת ידנית ולא משודרגת.
- במערכת הנוכחית רישום הזמנת הטיפולים נרשם באופן מסורבל ללא קישור למאגר העסק.
- קיימות כפילויות במאגרי התורים, דבר שיוצר בלבול בהזמנות התורים השונות.
- לא קיים מאגר של רופאים ואחיות.

ועל כן באה המערכת החדשה לתקן ולשפר על ידי ריכוז פרטי הטיפולים המוצעים ופרטי הרופאים והאחיות את הניהול השוטף בשטח. כדי לדעת במה לטפל, יש להכניס הכל תחת מערכת ממוחשבת אחת ידידותית לסביבה.
קליטת הנתונים תתבצע ישירות למערכת הממוחשבת תוך בקרה של התאמת הנתונים למידע הקודם.
למעשה המערכת מיועדת לניהול מסד הנתונים שעומד מאחורי העסק בצורה נוחה וידידותית למשתמש.

1.4 מטרות ויעדי המערכת:

1.4.1 בעיות במידע

1.4.1.1 זמינות וזמן-תגובה

1.4.1.2 המידע על מלאי זמינות תורים אצל רופא אינו נגיש. הדרך היחידה לדעת מתי יש תור פנוי אצל הרופא הספציפי היא ליצור קשר עם המוקד הטלפוני. פעולה זו גוזלת זמן רב.

1.4.1.3 דוחות הציוד החסר לצוות מוגשים למנהל באיחור. במקום דו״ח יומי, מוגש לעיתים הדו״ח רק פעם בשבוע.

1.4.2 עדכניות המידע

1.4.2.1 עדכון על שינויים בשעות עבודתם של הצוות הרפואי מוגשים למנהל באיחור.

1.4.2.2 עדכון ביטול תורים על ידי מטופלים לא מועבר לצוות הרפואי בזמן.

1.4.3 שלמות המידע

1.4.3.1 המידע על התיק הרפואי של המטופלים אינו שלם. מניחים שהצוות הרפואי מכיר היטב את מצבם הרפואי של המטופלים, בודקים אותו בקביעות ומנהלים רישום סדיר. אולם עקב מחקר מצאנו כי הצוות הרפואי לא מצליחים לזכור את כל הפרטים הרפואיים על המטופלים, ולכן, לעיתים קרובות רופא אינו יכול להמשיך בהליך הרפואי מאחר שקיים סיכון לרשלנות רפואית מטעמו.

1.4.3.2 תאריכי ההזמנות של התורים חסרים לעיתים קרובות, לכן קשה לעקוב אחר מועד הגעתו של מטופל, והאם הוא אכן הגיע לקבלת הטיפול.

1.4.4 דיוק המידע

1.4.4.1 תור המיועד למטופל ספציפי מדווח לעיתים כמה פעמים: כמה אנשי צוות שונים קובעים לאותו מטופל את אותו התור במועד אחר. כתוצאה מכך, רשימת התורים מכילה תורים חסרי משמעות הדורשים טיפול שלא בצורך. בבדיקתנו נמצא כי בערך 10% מרישומי התורים שנקבעו למטופלים הינם תורים שאינם מצדיקים צורך במתן הליך רפואי נוסף. בעיה זו גורמת לבזבוז זמן יקר של הצוות הרפואי, שיכול להעניק טיפול חיוני למטופלים אחרים.

1.4.4.2 חלק מדוחות הטיפולים המוגשים למנהל, מכילים טעויות הנוצרות עקב חוסר מעקב בטיפולים שניתנו לאלה שלא.

1.4.5 יעילות

1.4.5.1 תהליך הזמנת המוצרים מכיל כפילויות. לפי דעתך, המאגר "מוצרים חסרים" מיותר. הוא קיים רק מפני שקשה לעובדי החנות למיין את רשימות ההזמנה, ולכן הם מכניסים לתיק "מוצרים חסרים" רשימות לא ממוינות של שמות מוצרים, ומאוחר יותר, כאשר המנהל מבקש זאת, הם ממיינים רשימות אלה.

1.4.6 עומסים

1.4.6.1 בשעות השיא של הפעילות בקופת החולים, הצוות הרפואי אינו עומד בעומס. במדידות שונות מצאת כי לעיתים הרופאים לא קובעים למטופלים טיפולי המשך, ובכך עלולה להיפגע בריאותם של המטופלים

1.4.7 בעיות תיאום

1.4.7.1 בתוך המערכת

1.4.7.1.1 תיאום לקוי בין אנשי הצוות הקובעים לאותו מטופל כמה תורים שונים.

1.4.7.2 בין המערכת לסביבה

1.4.7.2.1 הנהלת החשבונות מוציאה לעיתים חיובים למטופלים לפני שהמטופלים סופקו בפועל. במקרים אחדים הצוות הרפואי מקבל משכורות באיחור של כמה חודשים.

1.4.8 בעיות בתהליכים

1.4.8.1 כרגע במערכת הנוכחית חלק מן התהליכים כמו הזמנת תור ועדכון פרטים אישיים לוקח יותר מדי זמן.

2.תיחום המערכת:

2.1 גורמים מעורבים:

שם הגורם	תיאור	רמת מעורבות
מנהל העסק	• ניהול העסק • שיווק • קשר עם הצוות הרפואי • עדכון בשעות הזמינות של הצוות הרפואי	יוזם המערכת ומשתמש עיקרי
צוות רפואי	מתפעלים את העסק בפועל. הם אלו האחראיים לפעולות היומיומיות של הקופה. הצוות הרפואי אחראי על מתן הטיפולים למטופלים	משתמש משני
מטופל	מזמין שירותים רפואיים	משתמש משני

2.2 תרשים המבנה הארגוני:

2.3 תיחום ארגוני :

- המערכת תיתן תמיכה למטופלים ולצוות הרפואי.
- המערכת תופעל בידי המנהל.
- המערכת תתוכנת במראה ויזואלי נח וידידותי למשתמש כדי לאפשר הטמעה מהירה.
- המערכת תאפשר עדכון מידי של מאגרי המידע.

2.4 תיחום תהליכי:

המערכת תעסוק בתהליכים הבאים :

- עדכון שעות עבודה של הצוות הרפואי.
- רישום הצוות הרפואי למחלקות המתאימות להם.
- קביעת תורים למטופלים.

2.5 מערכות קשורות:

- תוכנת הנהלת חשבונות.
- תוכנת שכר.

2.6 המערכת לא תטפל ב:

- תחזוקה.
- שיווק ופרסום.
- בצד הפיננסי : משכורות, גביה ותמחור.

3. אפיון המשימה:

3.1 יעדים:

מטרות	יעדים	מדדים	קריטריון הצלחה
זירוז איתור של **מטופל/עובד מהצוות הרפואי/מחלקות רפואיות**	אפשרות להקיש מספר ת.ז ולקבל את כל פרטיו.	הזמן מהרגע הקשת תעודת הזהות עד לקבלת הפרטים.	2 שניות.
סיוע בטיפול בפרטי ה**מטופלים/עובדי צוות רפואי/מחלקות רפואיות**	אפשרות להוספת מטופלים/עובדי צוות רפואי/מחלקות רפואיות חדש.	הזמן מרגע הוספת מטופל/עובד צוות רפואי/מחלקה רפואית ועד לרגע עדכונו במערכת.	3 שניות.
	אפשרות לעדכן פרטי המטופלים/עובדי צוות רפואי/מחלקות רפואיות.	הזמן מרגע עדכון פרטי מטופל/עובד צוות רפואי/מחלקה רפואית ועד לרגע עדכונו במערכת.	3 שניות
	אפשרות למעבר על רשימת **ה**מטופלים/עובדי צוות רפואי/מחלקות רפואיות.	הזמן מרגע מעבר מטופל/עובד צוות רפואי/מחלקה רפואית ועד קבלת הרשומה הבאה.	3 שניות.
מניעת כפילויות במערכת של הטיפולים/מחלקות רפואיות.	למנוע מצב שבו יהיו במערכת 2 טיפולים/מחלקות רפואיות זהות.	אי הופעת כפילויות במערכת הטיפולים והמחלקות.	אי הופעת כפילויות במערכת הטיפולים והמחלקות.

3.2 אילוצים:

סיווג	שם ותאור האילוץ
אילוצי תקציב	הסכום לרכישת התוכנה לא יהיה יותר מ- 100 אלף ₪.
	הסכום לפיתוח התוכנה לא יהיה יותר מ- 5000 ₪
	תקציב התחזוקה השוטפת של המערכת לאחר הקמתה לא יהיה יותר מ- 3000 ₪ בשנה.
אילוצי כוח אדם	פיתוח המערכת ע"י תלמיד מגמת מדעי המחשב על המפתח לפתח, לעצב ולתכנן את המערכת באופן עצמאי. העזרה ניתנת על ידי מורות המגמה.
אילוצי לוח זמנים	המערכת תהיה מוכנה עד 1 ביוני 2021
אילוצי ציוד, אופי המערכת וטכנולוגיה	המערכת צריכה להיות אמינה ופשוטה כדי שלא יצטרכו לתקנה כל פעם. דרישות המערכת המינימליות של התוכנה הם: מעבד - Intel Pentium 4, מערכת הפעלה - Windows 10 בסביבת עבודה: visual C# 2013 תכתב בשפה: C# מסד נתונים: Access 2013
אילוצי המערכת	צריכה להיות ידידותית לכל מפעיליה.
	יהיה קל לשפר אותה ולהרחיבה.

3.3 הנחות יסוד:

לכל מטופל יכול להיות כמה טיפולים ספציפיים שקיבל – לכל טיפול ספציפי יש מטופל אחד.

לכל עובד צוות רפואי יש כמה טיפולים כללים שהוא מציע – לכל טיפול כללי שמוצע יש עובד צוות רפואי אחד.

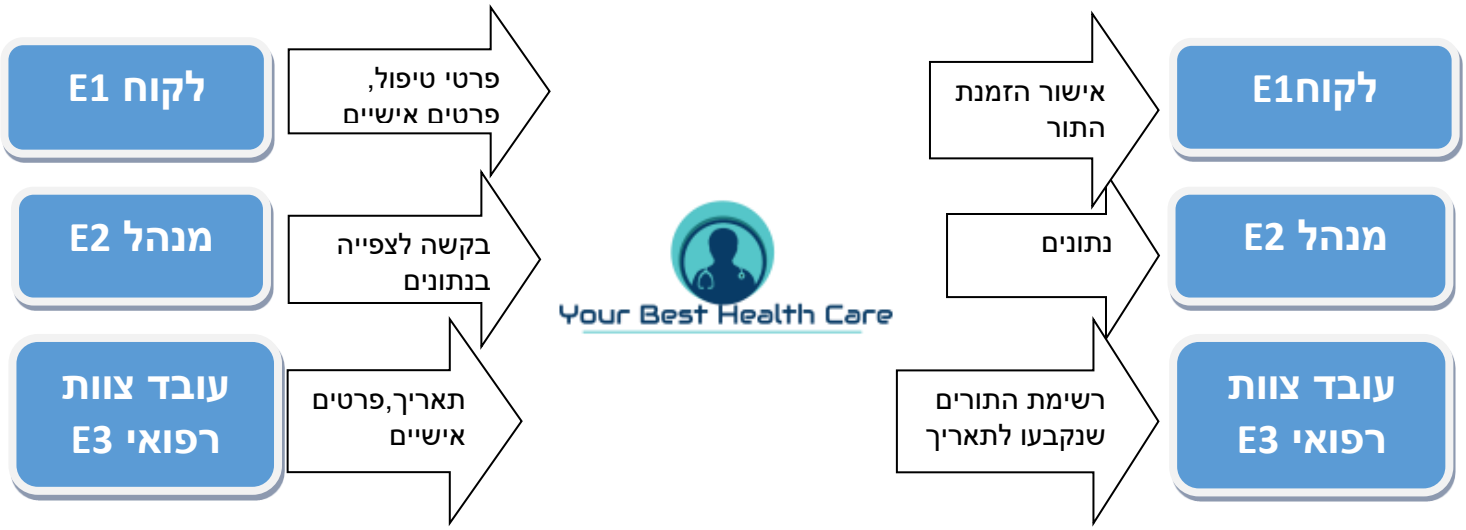
לכל טיפול כללי יכול להיות כמה טיפולים ספציפיים – לכל טיפול ספציפי יש טיפול כללי אחד.

כל עובד צוות רפואי יכול להשתייך לכמה מחלקות רפואיות – במחלקה רפואית יכולים להיות כמה עובדי צוות רפואי.

ניתוח מצב קיים- תוכן עניינים

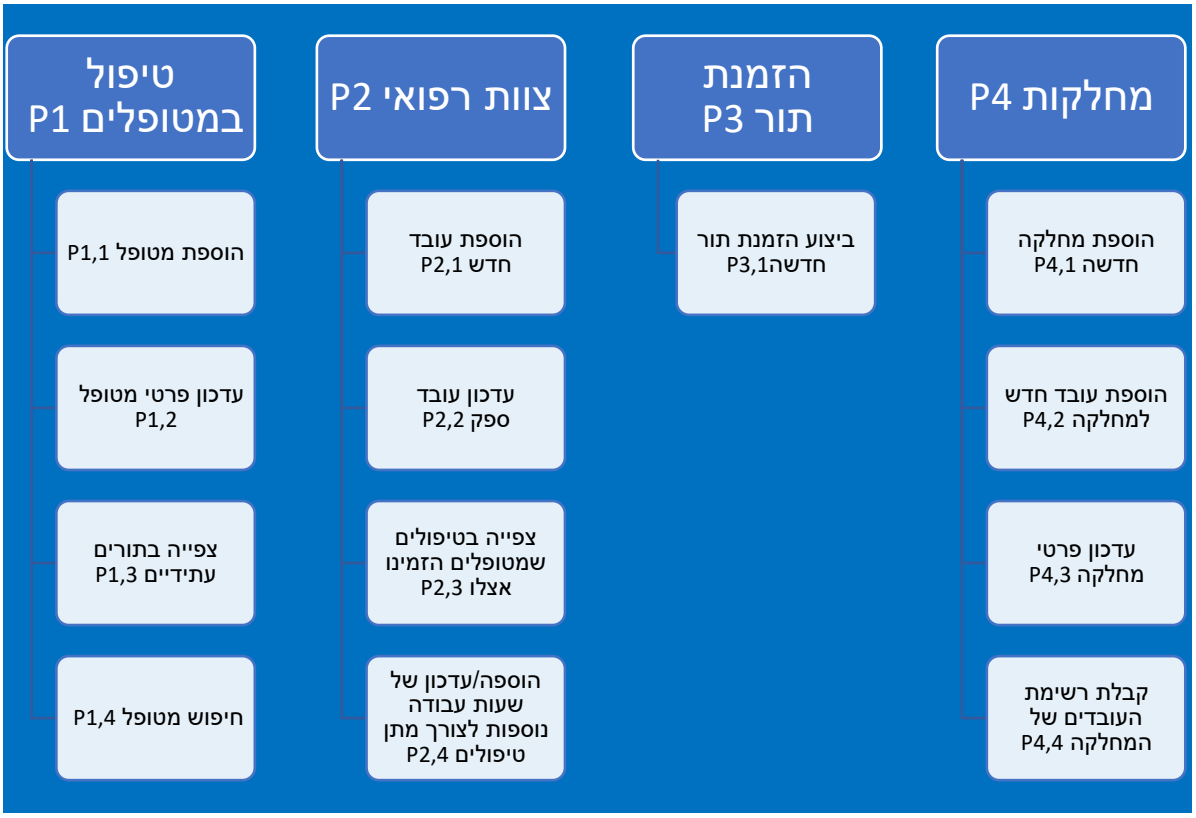
פרק שני: ניתוח מצב קיים

1.תרשים תוכן



2. תהליכים

2.1 עץ תהליכים



2.2 תיאור תהליכים:

תהליך מס'	שם התהליך	תיאור קצר	ממוחשב (כן, לא, חלקית)	בעיות במצב הקיים
1	הוספת מטופל חדש	מטופל חדש רוצה להירשם לקופה שלנו	חלקי	**במערכת יש טעויות ברישום המטופל**
2	עדכון פרטי מטופל	מטופל רוצה לעדכן אחד מפרטיו האישיים	חלקי	**לא בהכרח כל עדכון נשמר במערכת**
4	חיפוש מטופל	חיפוש פרטי מטופל ע"י שימוש בשם משפחה	חלקי	**לא כל המטופלים שמורים במערכת**
5	צפייה בטיפולים עתידיים של המטופל	בהינתן נתונים של מטופל, נציג את כל הטיפולים	חלקי	**לא כל הזמנות הטיפולים שמורות במערכת**

		העתידיים שהוזמנו לו		
6	הוספת עובד צוות רפואי	עובד צוות רפואי חדש מגיע לעבוד בקופה וברצוננו לרשום אותו במערכת	חלקי	**במערכת יש טעויות ברישום העובד**
7	עדכון פרטי עובד צוות רפואי	עובד צוות רפואי רוצה לעדכן אחד מפרטיו האישיים	חלקי	**לא בהכרח כל עדכון נשמר במערכת**
8	הוספת שעות עבודה של עובד לצורך מתן טיפולים	עובד צוות רפואי רוצה להוסיף שעות עבודה אפשריות לצורך מתן טיפולים	חלקי	**במערכת יש טעויות ברישום השעות שהזין העובד**
9	עדכון שעות עבודה של עובד לצורך מתן טיפולים	עובד צוות רפואי רוצה לעדכן שעות עבודה אפשריות לצורך מתן טיפולים	חלקי	**לא בהכרח כל עדכון נשמר במערכת**
10	ביצוע הזמנת תור	מטופל רוצה לקבוע תור אצל רופא/אחות	חלקי	**לעיתים התור לא נשמר במערכת**
11	הוספת מחלקה חדשה	יצירת מחלקה חדשה בקופת חולים	חלקי	**לא כל פרטי המחלקה נשמרים**
12	עדכון פרטי מחלקה	שינוי אחד המאפיינים של המחלקה	חלקי	**לא כל השינויים נשמרים**
13	צפייה ברשימת העובדים שעובדים באותה המחלקה	בהינתן נתוני מחלקה, נרצה להציג את כל העובדים שעובדים במחלקה זו	חלקי	**לא כל העובדים משובצים למחלקה**
14	הוספת עובד למחלקה נתונה	בהינתן נתוני מחלקה, נרצה להוסיף למחלקה עובד חדש	חלקי	**לא כל פרטי העובד נשמרים**

3. בעיות:

שם הבעיה	דרגת חומרה	תיאור	תוצאות	סיווג הבעיה	סיבות	המלצה לפתרון
כפילויות בתורים שהוזמנו	חמור מאוד	תור שהוזמן מראש נרשם כמה פעמים במערכת ולפעמים גם עם נתונים שונים	ביצוע תהליכים עם נתונים שגויים גורם לשרשרת של שגיאות בניהול מערכת המידע.	בעיה במידע - שלמות המידע	בזמן הזמנת התור אין בדיקה אם למטופל כבר הוזמן התור המבוקש	יש לנהל מאגר תורים עם מפתח "מזהה תור". בעת הוספת תור חדש שהוזמן למאגר, יש לבצע בדיקת תקינות האם התור רשום כבר במאגר.
פרטי המטופל אינם עדכניים	חמור	פרטי המטופל במציאות אינם תואמים את פרטי המטופל המופיעים במאגר	קושי באיתור המטופל בעת הצורך. לא ניתן לבצע פעולות במערכת המידע / תהליכים מתעכבים.	בעיה במידע - עדכניות המידע	אין נוהל מסודר על תהליך עדכון נתונים.	יש להוסיף תהליך שיתבצע כל פרק זמן מוגדר מראש שיבקש מידע מעודכן מהמטופל ויעדכן זאת במאגר המטופלים
פרטי מטופל חסרים	חמור מאוד	במאגר מטופלים חסרים פרטים לצורך ביצוע פעולות מערכת	קושי בביצוע פעולות הקשורות למטופל בעת הצורך .לא ניתן לבצע פעולות במערכת המידע /	בעיה במידע - שלמות המידע	על הצוות הרפואי לדעת כמה שיותר על פרטי המטופל וזאת כדי להעניק לו את הטיפול	יש לבדוק מהן הפעולות השונות המתבצעות על מטופל ותוך כדי כך להגדיר את תכונות

		המידע	תהליכים מתעכבים.		הרפואי הטוב ביותר	המטופל במאגר מטופלים
קושי באיתור מטופל	חמור מאוד	עובד צוות רפואי מתעכב כאשר הוא מנסה לאתר מידע על מטופל לצורך קבלת מידע רפואי על המטופל	קושי באיתור מטופל בעת הצורך. לא ניתן לבצע פעולות במערכת המידע / תהליכים מתעכבים.	בעיה בתהליך - יעילות	מאגר המטופלים הוא ידני, מסורבל ולא קריא בשל עדכונים רבים.	יש להגדיר במאגר המטופלים בנוסף למפתח הסדר מפתח חיפוש נוסף "שם מטופל."
כפילויות ברישום מטופלים	חמור	מטופל רשום במאגר מספר פעמים ולפעמים גם עם נתונים שונים	ביצוע תהליכים עם נתונים שגויים גורם לשרשרת של שגיאות ניהול מערכת המידע.	בעיה בתהליך - יעילות	בזמן רישום מטופל חדש במאגר המידע, אין בדיקה האם קיים רישום קודם לאותו מטופל	הגדרת מפתח סדר "ת.ז". בעת ההוספה של מטופל יש לבדוק האם אינו מופיע כבר במאגר.
פרטי עובד אינם עדכניים	חמור	פרטי העובד במצאות אינם תואמים את פרטי העובד המופיעים במאגר	קושי באיתור העובד בעת הצורך. לא ניתן לבצע פעולות במערכת המידע / תהליכים מתעכבים.	בעיה במידע - עדכניות המידע	אין נוהל מסודר על תהליך עדכון נתונים.	יש להוסיף תהליך שיתבצע כל פרק זמן מוגדר מראש שיבקש מידע מעודכן מהעובד ויעדכן זאת במאגר עובדי הצוות
פרטי עובד	חמור	במאגר עובדי	קושי בביצוע פעולות	בעיה במידע -	פרטי הצוות	יש לבדוק מהן

חסרים	מאוד	צוות רפואי חסרים פרטים לצורך ביצוע פעולות מערכת המידע	הקשורות לעבוד בעת הצורך .לא ניתן לבצע פעולות במערכת המידע / תהליכים מתעכבים.	שלמות המידע	הרפואי מוכרחים להימצא במאגר הנתונים כדי שלהנהלה תהיה היכולת לאתר את עובדיהם וכדי שהמטופלי ם יוכלו לבחור ברופא/ אחות המתאים לדרישותיה ם	הפעולות השונות המתבצעות על עובד הצוות הרפואי ותוך כדי כך להגדיר את תכונות העובד צוות רפואי במאגר מעובדי צוות רפואי
קושי באיתור עובד צוות רפואי	חמור מאוד	תהליך הזמנת תור מתעכב כאשר המטופל/ מנהל מנסה לאתר מידע על העובד לצורך קבלת מידע מתאים	קושי באיתור עובד בעת הצורך. לא ניתן לבצע פעולות במערכת המידע / תהליכים מתעכבים.	בעיה בתהליך - יעילות	מאגר עובדי הצוות הוא ידני, מסורבל ולא קריא בשל עדכונים רבים.	יש להגדיר במאגר עובדי הצוות הרפואי בנוסף למפתח הסדר מפתח חיפוש נוסף "שם עובד."
כפילויות ברישום עובדים	חמור	עובד צוות רפואי רשום במאגר מספר פעמים ולפעמים גם עם נתונים שונים	ביצוע תהליכים עם נתונים שגויים גורם לשרשרת של שגיאות ניהול מערכת המידע.	בעיה בתהליך – יעילות	בזמן רישום עובד חדש במאגר המידע, אין בדיקה האם קיים רישום קודם לאותו עובד	הגדרת מפתח סדר "ת.ז". בעת ההוספה של עובד צוות רפואי יש לבדוק האם אינו מופיע כבר במאגר.

אין מספיק מידע על טיפולים שניתנו	חמור	למנהל יש קושי לקבל בזמן נתון אינדקציה לגבי יעילות הטיפולים שמוצעים	ביצוע כפול של טיפולים או אי ביצוע של טיפולים בכלל	בעיה תהליך-יעילות	אין תהליך מסודר של ביצוע טיפולים למטופלים. אין מעקב מסודר אחר טיפולים שטרם בוצעו	כל טיפול שבוצע יש לעדכן במאגר בהתאם

מסמך הגדרת דרישות - תוכן עניינים

פרק שלוש- מסמך הגדרת דרישות

1. דרישות

1.1 דרישות מול משתמשים

שם משתמש	רשימת דרישות מהמערכת
מנהל קופת חולים	• צפייה בפרטי מטופל, עובד צוות רפואי, ומחלקה • אפשרות לעדכון פרטי, מטופל, עובד צוות רפואי, ומחלקה • אפשרות להוספת פרטי מטופל, עובד צוות רפואי, ומחלקה • אפשרות לחיפוש פרטי מטופל • בירור לוח הטיפולים של כל עובד • קביעת תורים למטופלים • צפייה בפרטי תורים • צפייה בלוחות הזמנים של כל עובד • צפייה ברשימת התורים שהוזמנו למטופל
עובד צוות רפואי	• צפייה בפרטי מטופלים • עדכון פרטים אישיים • קביעת תורים למטופלים • חיפוש מטופל • הצגת כל הטיפולים שנקבעו על העובד לתת • הצגת כל הטיפולים שהוזמנו למטופל
מטופל	• עדכון פרטים אישיים • הזמנת תור • צפייה בפרטי תורים שהוזמנו לו

1.2 דרישות פונקציונאליות

מספר	שם הדרישה	משתמש	תגובת המערכת	הערות
.1	אפשרות להוסיף מטופל/ עובד/ מחלקה חדש	מנהל	מוסיפה מטופל/ עובד/ מחלקה בהתאמה	
.2	אפשרות לעדכן פרטי מטופל	מנהל, עובד צוות רפואי, מטופל	מעדכנת פרטי מטופל	
.3	אפשרות לצפייה בטיפולים עתידיים שנקבעו למטופל	מנהל, עובד צוות רפואי, מטופל	המערכת מציגה את רשימת הטיפולים העתידיים של המטופל	
.4	אפשרות לעדכן פרטי עובד צוות רפואי	מנהל, עובד צוות רפואי	מעדכנת פרטי עובד צוות רפואי	
.5	אפשרות לעדכון/ הוספת שעות עבודה של העובד צוות רפואי	מנהל, עובד צוות רפואי	מעדכנת/ מוסיפה שעות עבודה של העובד לצורך מתן טיפולים	
.6	אפשרות לחפש מטופל	מנהל, עובד צוות רפואי	מחפשת מטופל	
.7	אפשרות להצגת נתונים על מטופל	מנהל, עובד צוות רפואי	מציגה פרטי מטופל	

8.	אפשרות לחפש מטופל	מנהל, עובד צוות רפואי	מחפש מטופל	
9.	אפשרות להצגת נתונים על עובד/ מחלקה מסוימת	מנהל	מציג נתונים של עובד/ מחלקה מסוימת בהתאמה	
10.	אפשרות להוספת עובד צוות רפואי למחלקה מסוימת	מנהל	מוסיפה עובד חדש למחלקה הנתונה מראש	
11.	אפשרות להצגת נתונים על מטופל	מנהל, עובד צוות רפואי	מציג נתונים של מטופל מסוים	
12.	אפשרות לקביעת תור למטופל	מנהל, עובד צוות רפואי, מטופל	קובע תור למטופל	
13.	צפייה בלוח הטיפולים של עובד צוות רפואי	מנהל, עובד צוות רפואי	מציג לוח טיפולים של כל רופא	
14.	צפייה בפרטי תורים	מנהל, עובד צוות רפואי, מטופל	מציג פרטי תור	

2. אפיון המערכת העתידית:

2.1 תרשים שחקנים

2.2 תרשים Use Case

2.2.1 שם המודול ניהול מטופלים

אפשרות להוסיף מטופל חדש

שם הפעולה	הוספת מטופל חדש.
מטרה	לאפשר הוספה של מטופל חדש.
תנאי קדם	המטופל אינו קיים במערכת.
שחקנים	מנהל.
אירוע מזניק	מטופל חדש הגיע לקופת החולים.
מהלך	המנהל מזין את פרטי המטופל ושומר אותם במערכת.
חריגים	נתונים לא תקינים. (בדיקת תקינות ת.ז).
תנאי סיום	מטופל חדש התווסף למאגר המטופלים.

אפשרות להציג פרטי מטופל מסוים.

שם הפעולה	הצגת נתונים על מטופל מסוים.
מטרה	לאפשר צפייה בפרטיו האישיים של המטופל.
תנאי קדם	קיים מטופל שרוצים לצפות בפרטיו.
שחקנים	מנהל, עובד צוות רפואי, מטופל
אירוע מזניק	רצון לצפות בפרטיו האישיים של המטופל.
מהלך	בדיקת הנתונים של חיפוש צפייה במטופלים.
חריגים	אין.
תנאי סיום	מטופל נמצא בהצלחה.

אפשרות לעדכן פרטי מטופל

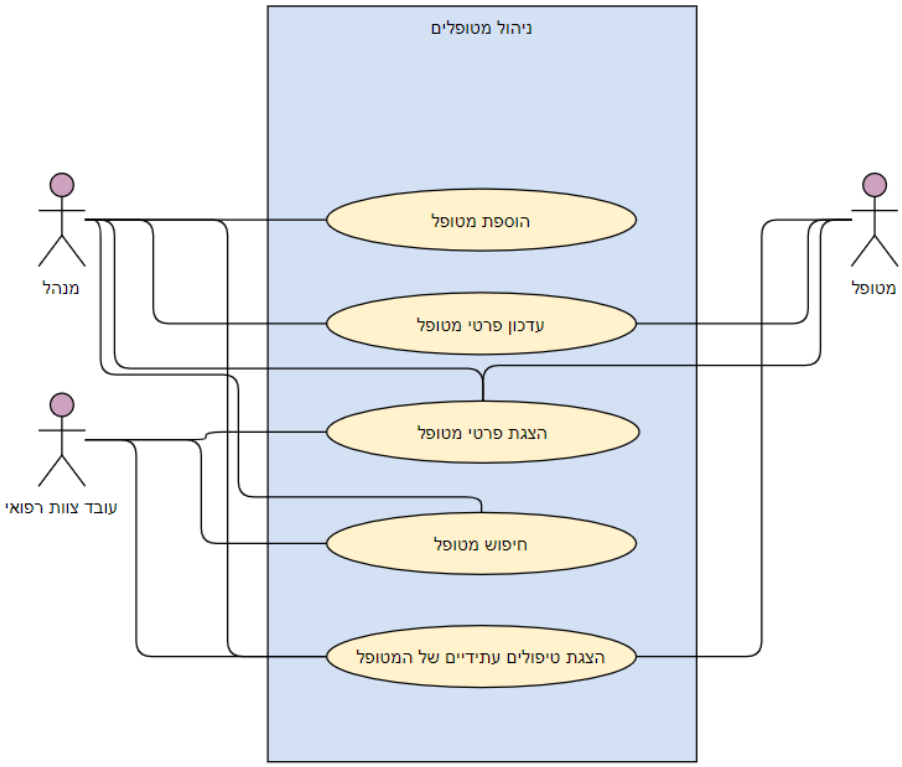
שם הפעולה	עדכון פרטי מטופל.
מטרה	לאפשר עדכון פרטים של מטופל מסוים.
תנאי קדם	אחד מפרטי המטופל שונו או שלא הוקלדו כראוי.
שחקנים	מנהל, מטופל
אירוע מזניק	מטופל ביקש לעדכן את פרטיו.
מהלך	בדיקת הנתונים ועדכון.
חריגים	אין.
תנאי סיום	פרטי המטופל המעודכנים נשמרו במערכת.

אפשרות לחפש מטופל

שם הפעולה	אפשרות לחיפוש מטופל במערכת קופת החולים.
מטרה	לאפשר צפייה בפרטיו האישיים של מטופל מסוים.
תנאי קדם	קיים מטופל שרוצים לצפות בו.
שחקנים	מנהל, עובד צוות רפואי.
אירוע מזניק	חיפוש לצורך הצגת פרטי מטופל/ עדכון.
מהלך	המנהל או העובד מזין את שם המטופל הרצוי והמערכת מציגה את פרטי המטופל.
חריגים	המטופל הרצוי אינו קיים במערכת/ הוקש שם שגוי.
תנאי סיום	הצגת פרטי המטופל אותו חיפשו.

הצגת טיפולים עתידיים של מטופל

שם הפעולה	אפשרות להצגת טיפולים עתידיים של מטופל במערכת קופת החולים.
מטרה	לאפשר צפייה בטיפולים העתידיים שהוזמנו למטופל מסוים.
תנאי קדם	קיים מטופל עם טיפולים עתידיים שרוצים לצפות בו.
שחקנים	מנהל, עובד צוות רפואי, מטופל
אירוע מזניק	הצגת טיפולים עתידיים של מטופל לצורך בקרה
מהלך	כל אחד מסוגי המשתמשים צופה בפרטי המטופל ונחשף גם לרשימת הטיפולים העתידיים שנקבעו לו
חריגים	אין
תנאי סיום	הצגת רשימת הטיפולים העתידיים של המטופל בו צופים



שם המודול: ניהול עובדים

אפשרות להוסיף עובד חדש.

שם הפעולה	הוספת עובד חדש למאגר העובדים.
מטרה	לאפשר הוספה של עובד חדש.
תנאי קדם	העובד אינו קיים במערכת.
שחקנים	מנהל.
אירוע מזניק	עובד חדש הצטרף לצוות הרפואי של קופת החולים.
מהלך	המנהל מזין את פרטי העובד ושומר אותם במערכת.
חריגים	נתונים לא תקינים. (בדיקת תקינות ת.ז).
תנאי סיום	עובד חדש התווסף למאגר העובדים.

אפשרות להציג פרטי עובד מסוים.

שם הפעולה	הצגת פרטים אישיים של עובד מסוים.
מטרה	לאפשר צפייה על פרטיו האישיים של העובד.
תנאי קדם	קיים עובד שרוצים לצפות בפרטיו.
שחקנים	מנהל.
אירוע מזניק	רצון לצפות בנתוני העובד.
מהלך	בדיקת הנתונים של חיפוש צפייה בעובדים.
חריגים	אין.
תנאי סיום	עובד נמצא בהצלחה.

אפשרות לעדכן פרטי עובד.

שם הפעולה	עדכון פרטי עובד.
מטרה	לאפשר עדכון פרטים של עובד מסוים.
תנאי קדם	אחד מפרטי העובד שונו או שלא הוקלדו כראוי.
שחקנים	מנהל, עובד צוות רפואי
אירוע מזניק	עובד ביקש לעדכן את פרטיו.
מהלך	בדיקת הנתונים ועדכון.
חריגים	אין.
תנאי סיום	פרטי העובד המעודכנים נשמרו במערכת.

אפשרות להוספת שעות עבודה לצורך מתן טיפולים

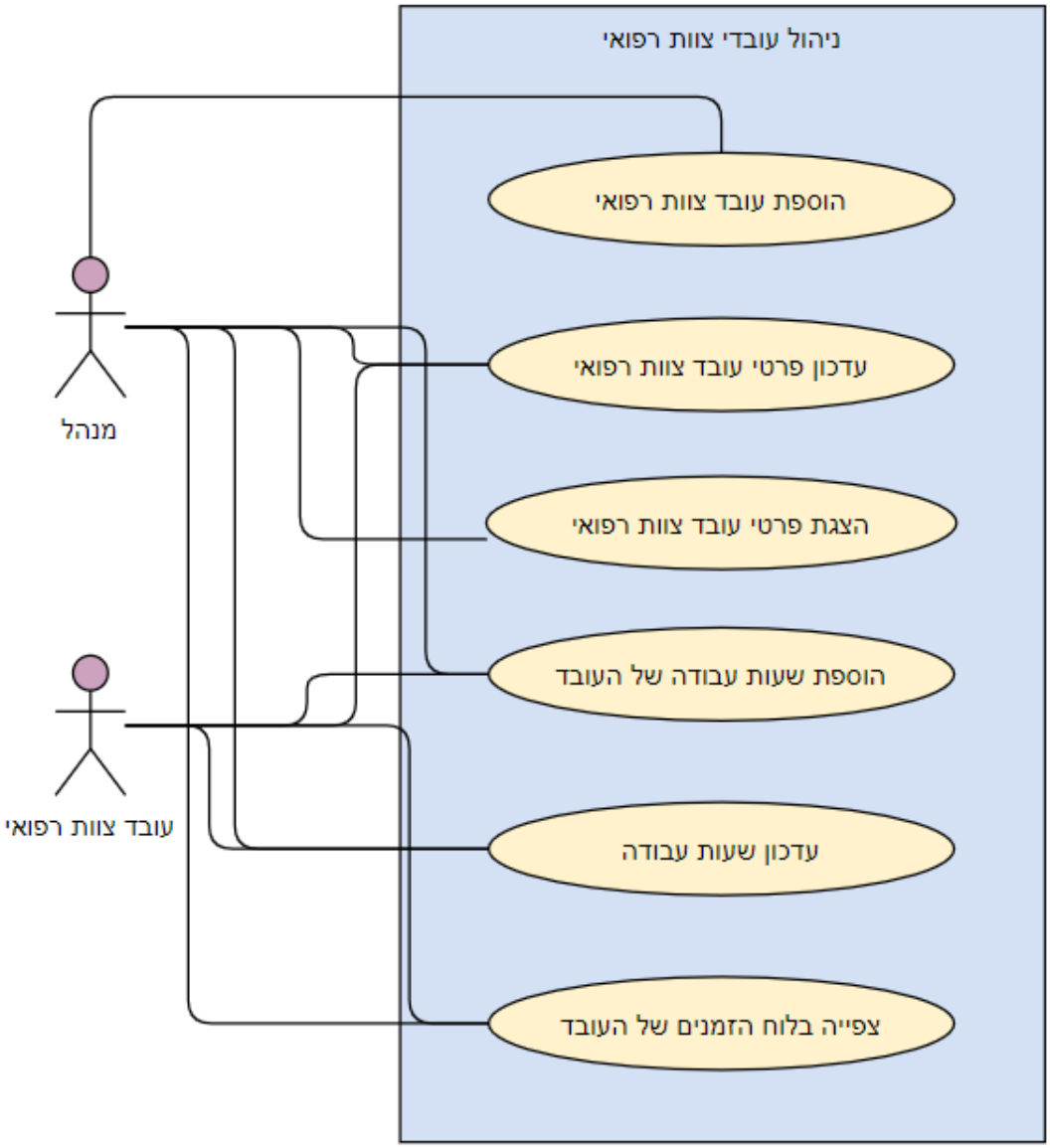
שם הפעולה	הוספת שעות עבודה לצורך מתן טיפולים
מטרה	לאפשר הוספה של שעות בהן עובד מסוים יכול לתת טיפולים רפואיים.
תנאי קדם	העובד נמצא במערכת, שעות העבודה החדשות שלו לא קיימות במערכת
שחקנים	מנהל, עובד צוות רפואי
אירוע מזניק	עובד ביקש להוסיף שעות עבודה לצורך מתן טיפולים רפואיים
מהלך	בדיקת תקינות הנתונים והוספתם.
חריגים	אין.
תנאי סיום	שעות העבודה החדשות נשמרו במערכת.

אפשרות לעדכון שעות עבודה לצורך מתן טיפולים

שם הפעולה	עדכון שעות עבודה לצורך מתן טיפולים
מטרה	לאפשר עדכון של שעות בהן עובד מסוים יכול לתת טיפולים רפואיים.
תנאי קדם	העובד נמצא במערכת, קיימות לעובד שעות עבודה במערכת
שחקנים	מנהל, עובד צוות רפואי
אירוע מזניק	עובד ביקש לעדכן שעות עבודה לצורך מתן טיפולים רפואיים
מהלך	בדיקת תקינות הנתונים ועדכונם.
חריגים	אין.
תנאי סיום	שעות העבודה מתעדכנות ונשמרות במערכת.

אפשרות לצפייה בלוח הזמנים של עובד

שם הפעולה	צפייה בלוח הזמנים של עובד
מטרה	לאפשר לעובד לצפות במצב הזמנות הטיפולים אצלו
תנאי קדם	העובד נמצא במערכת
שחקנים	מנהל, עובד צוות רפואי
אירוע מזניק	עובד ביקש לצפות בלוח הטיפולים שלו
מהלך	בדיקת תקינות הנתונים והצגתם.
חריגים	אין.
תנאי סיום	לוח הזמנים של העובד מוצג


ניהול עובדי צוות רפואי
הוספת עובד צוות רפואי
עדכון פרטי עובד צוות רפואי
הצגת פרטי עובד צוות רפואי
הוספת שעות עבודה של העובד
עדכון שעות עבודה
צפייה בלוח הזמנים של העובד
מנהל
עובד צוות רפואי

שם המודול: ניהול מחלקות

אפשרות להוסיף מחלקה חדשה לקופת החולים.

שם הפעולה	אפשרות להוסיף מחלקה חדשה לקופת החולים.
מטרה	לאפשר הוספה של מחלקה חדשה.
תנאי קדם	לא קיימת מחלקה עם אותו קוד או שהמחלקה לא קיימת.
שחקנים	מנהל.
אירוע מזניק	רצון להרחיב את שירותי המרפאה, ולספק שירות איכותי וטוב לכמה שיותר מטופלים.
מהלך	המנהל מזין את פרטי המחלקה החדשה, ושומר אותם בתוך המערכת.
חריגים	נתונים לא תקינים, כמו: שם מחלקה, קוד מחלקה.
תנאי סיום	מחלקה חדשה התווספה למאגר המחלקות של קופת החולים.

אפשרות להציג פרטי מחלקה מסוימת.

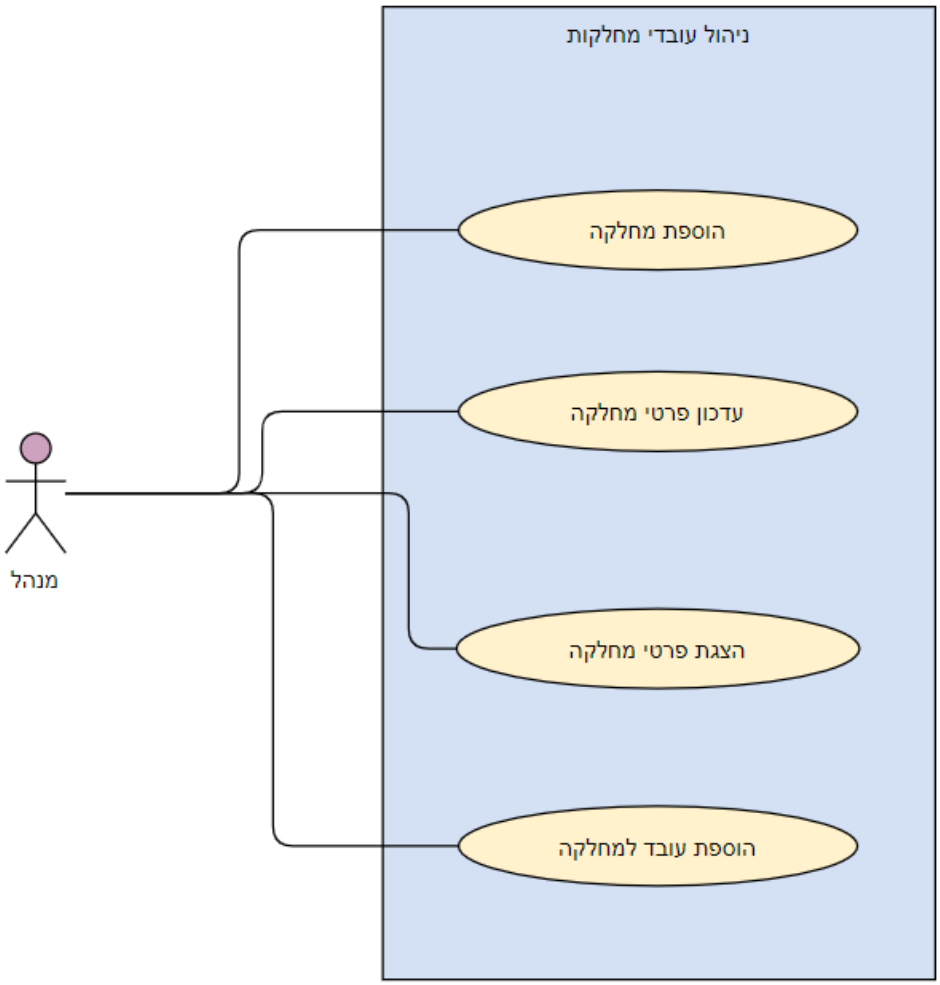
שם הפעולה	הצגת נתונים על מחלקה מסוימת.
מטרה	לאפשר צפייה בפרטי המחלקה.
תנאי קדם	קיימת מחלקה שרוצים לצפות בפרטיה.
שחקנים	מנהל.
אירוע מזניק	רצון לצפות בפרטיה של המחלקה.
מהלך	בדיקת הנתונים של חיפוש צפייה במחלקות.
חריגים	אין.
תנאי סיום	פרטי המחלקה נמצאו בהצלחה.

אפשרות לעדכן פרטי מחלקה.

שם הפעולה	עדכון פרטי מחלקה.
מטרה	לאפשר עדכון פרטים של מחלקה מסוימת.
תנאי קדם	אחד מפרטי המחלקה שונו או שלא הוקלדו כראוי.
שחקנים	מנהל.
אירוע מזניק	המנהל הבחין כי באחד השדות של המחלקה קיימת טעות.
מהלך	בדיקת הנתונים ועדכון.
חריגים	אין.
תנאי סיום	פרטי המחלקה המעודכנים נשמרו במערכת.

אפשרות להוספת עובד למחלקה

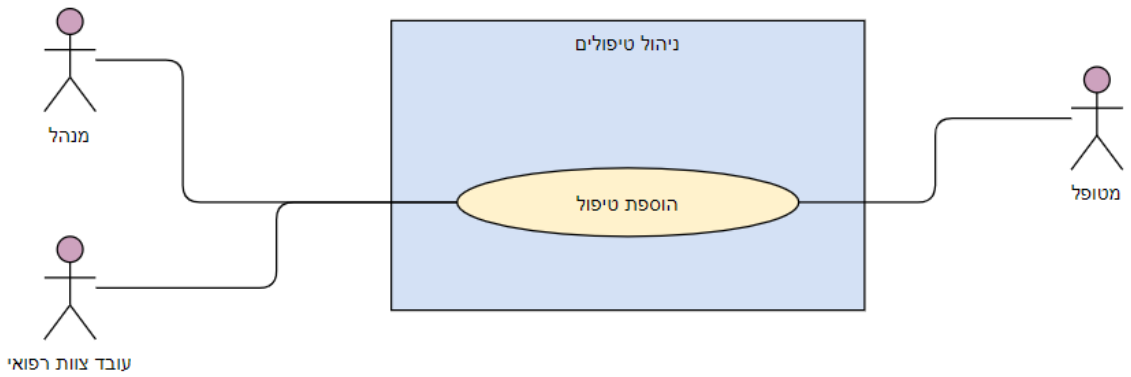
שם הפעולה	הוספת עובד למחלקה
מטרה	לאפשר להוסיף עובדים למחלקות קיימות
תנאי קדם	העובד כבר קיים במערכת
שחקנים	מנהל.
אירוע מזניק	המנהל הבחין כי יש צורך בלהוסיף עוד עובדים למחלקה
מהלך	בדיקת הנתונים והוספתם.
חריגים	אין.
תנאי סיום	פרטי העובד נשמרים במחלקה הספציפית


ניהול עובדי מחלקות
הוספת מחלקה
עדכון פרטי מחלקה
הצגת פרטי מחלקה
הוספת עובד למחלקה
מנהל

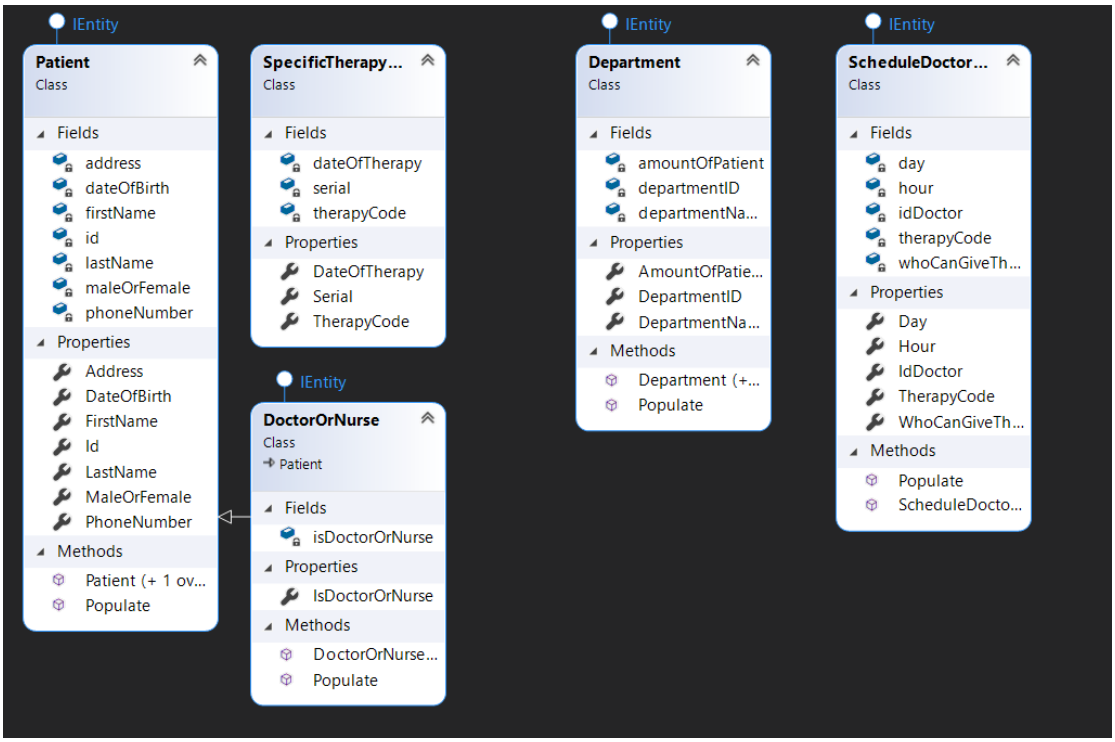
שם המודול: ניהול טיפולים

אפשרות להוסיף טיפול חדש.

שם הפעולה	הוספת טיפול חדש למאגר הטיפולים.
מטרה	לאפשר הוספה של טיפול חדש.
תנאי קדם	הטיפול אינו קיים במערכת.
שחקנים	מנהל, עובד צוות רפואי, מטופל
אירוע מזניק	מטופל רוצה להזמין טיפול אצל אחד מרופאי המרפאה.
מהלך	המנהל מזין את פרטי הטיפול הרצוי ושומר את הנתונים במערכת.
חריגים	נתונים לא תקינים (בדיקת תקינות קוד טיפול).
תנאי סיום	טיפול חדש התווסף למאגר הטיפולים של קופת החולים.



class diagram תרשים



3. תיאור מסד הנתונים

3.1 תיאור טבלאות:

מטופלים:

שם שדה	סוג נתונים
ID	טקסט קצר
FirstName	טקסט קצר
LastName	טקסט קצר
Address	טקסט קצר
PhoneNumber	טקסט קצר
DateOfBirth	תאריך/שעה
MaleFemale	טקסט קצר

ID	FirstName	LastName	Address	PhoneNuml	DateOfBirth	MaleFemale
123456789	Rom	Cohen	Alanbi 65	0523045678	17/11/2000	Male
210980762	Yaniv	Hacker	Balfur 12	0507632208	13/05/2004	Male
214393647	Jonathan	Langer	Rotchikd 3	0506578902	12/07/2003	Male
307865432	Dror	Cohen	Alanbi 5	0549203721	12/09/2003	Male
314278903	Shani	Maor	Shenkin 70	0525123461	27/10/2020	Female

עובד צוות רפואי:

שם שדה	סוג נתונים
ID	טקסט קצר
FirstName	טקסט קצר
LastName	טקסט קצר
PhoneNumber	טקסט קצר
DateOfBirth	תאריך/שעה
Address	טקסט קצר
MaleFemale	טקסט קצר
IsDoctorOrNurse	טקסט קצר

ID	FirstName	LastName	PhoneNuml	DateOfBirth	Address	MaleFemale	IsDoctorOrN
111111111	admin	admin	admin	20/05/1992	admin	admin	Doctor
213982305	Moshe	Butman	0506565201	14/08/1968	Shenkin 76	Male	Doctor
305637891	Sarit	Natan	0507655309	22/05/1980	Rotchild 45	Female	Nurse
305672144	Kfir	Bar	0523408921	16/05/1970	Rotchild 90	Male	Doctor
310562349	Galit	Haim	0549827821	06/04/1978	Alanbi 56	Female	Doctor

טיפול שנקבע במערכת בין מטופל לרופא:

שם שדה	סוג נתונים
Serial	מספר
TherapyCode	טקסט קצר
DateOfTherapy	תאריך/שעה
IdPatient	טקסט קצר

DepartmentID	DepartmentName	AmountOfPatient
1	Trauma	10
2	Plastics	10
3	Cardiology	5
4	Orthopedics	10
5	Orthodontics	5

קשר בין מחלקה לעובדי צוות רפואי:

שם שדה	סוג נתונים
DepartmentId	טקסט קצר
DoctorOrNurseId	טקסט קצר

Serial	TherapyCod	DateOfTher	IdPatient
1	1	19/08/2021	123456789
2	2	10/07/2021	214393647
3	2	16/09/2021	307865432
4	3	23/09/2021	314278903
5	4	09/12/2021	210980762

מחלקות:

שם שדה	סוג נתונים
DepartmentID	טקסט קצר
DepartmentName	טקסט קצר
AmountOfPatient	מספר

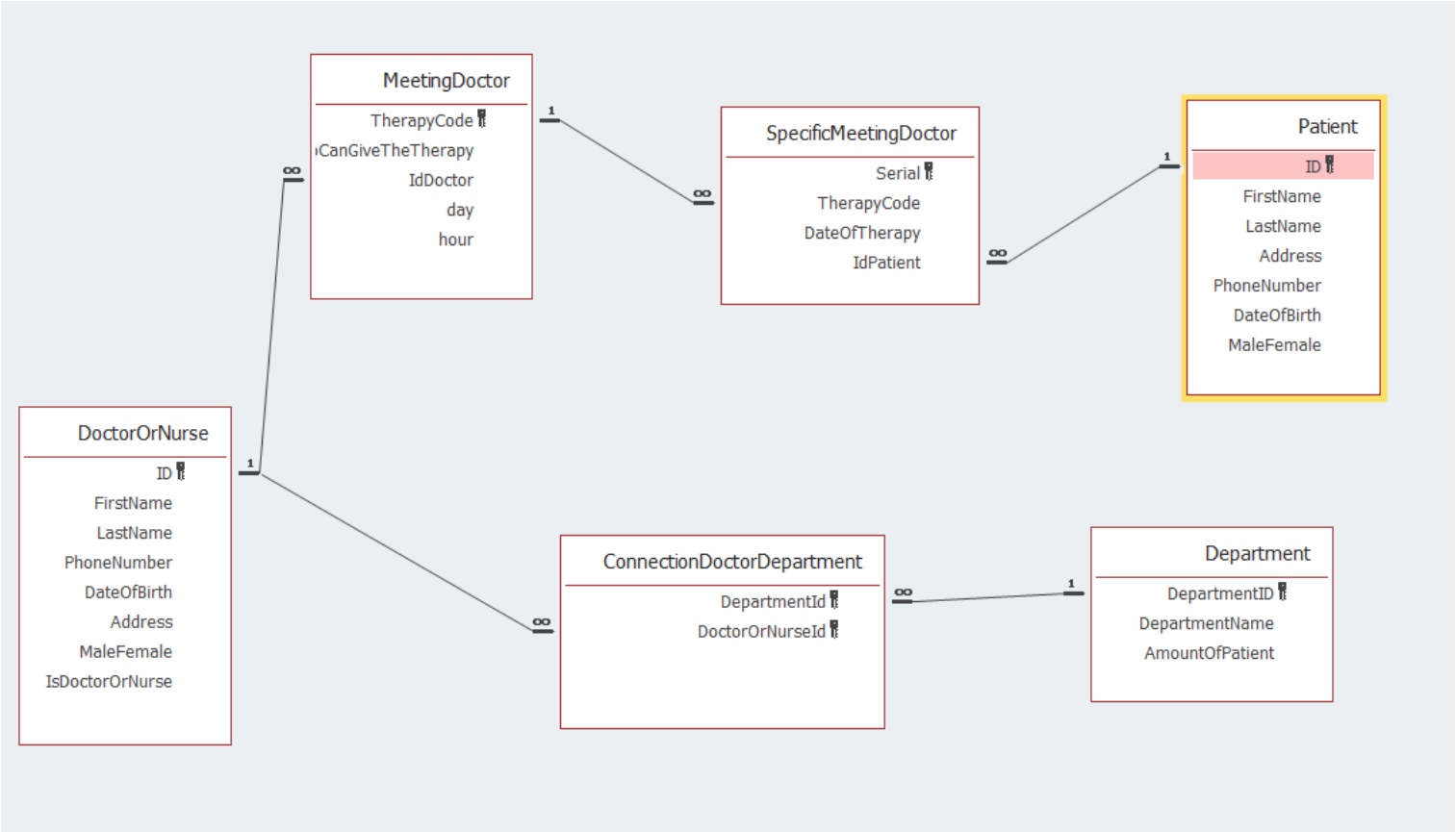
DepartmentID	DepartmentName	AmountOfPatient
1	Trauma	10
2	Plastics	10
3	Cardiology	5
4	Orthopedics	10
5	Orthodontics	5

שעת טיפול המיועדת לפגישה של עובדים עם מטופלים (באופן כללי):

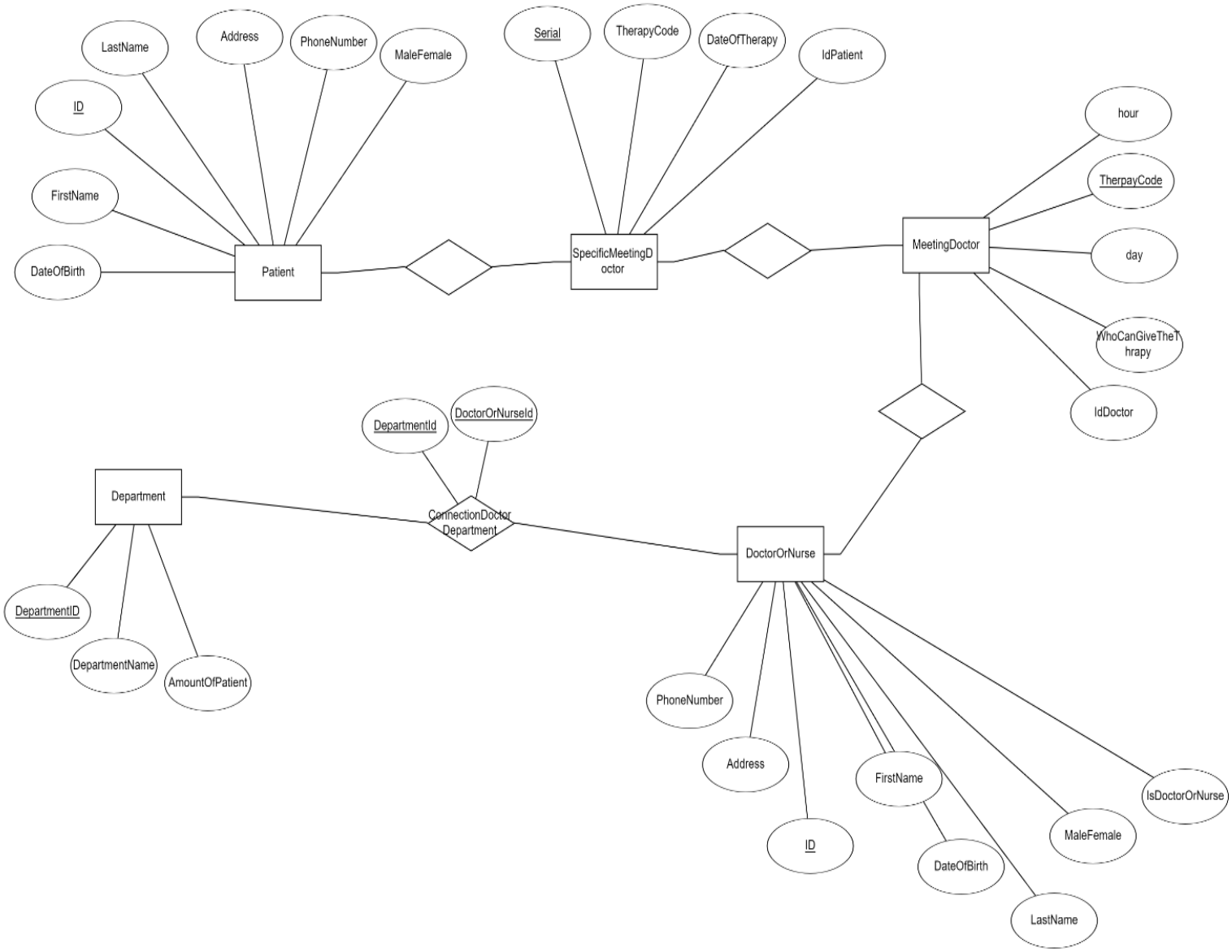
שם שדה	סוג נתונים
TherapyCode	טקסט קצר
WhoCanGiveTheTherapy	טקסט קצר
IdDoctor	טקסט קצר
day	מספר
hour	טקסט קצר

TherapyCod	WhoCanGiv	IdDoctor	day	hour
1	Doctor	213982305	1	12:00
2	Nurse	305637891	3	12:45
3	Nurse	305637891	4	16:00
4	Doctor	305672144	5	09:00
5	Doctor	310562349	6	08:30

3.2 קשרי גומלין ו ERD:



ERD


LastName
Address
PhoneNumber
MaleFemale
ID
FirstName
DateOfBirth
Patient
Serial
TherapyCode
DateOfTherapy
IdPatient
SpecificMeetingDoctor
MeetingDoctor
hour
TherpayCode
day
WhoCanGiveTheTherapy
IdDoctor
DepartmentId
DoctorOrNurseId
Department
ConnectionDoctorDepartment
DoctorOrNurse
DepartmentID
DepartmentName
AmountOfPatient
PhoneNumber
Address
ID
FirstName
DateOfBirth
LastName
MaleFemale
IsDoctorOrNurse

עיצוב

שלב העיצוב:

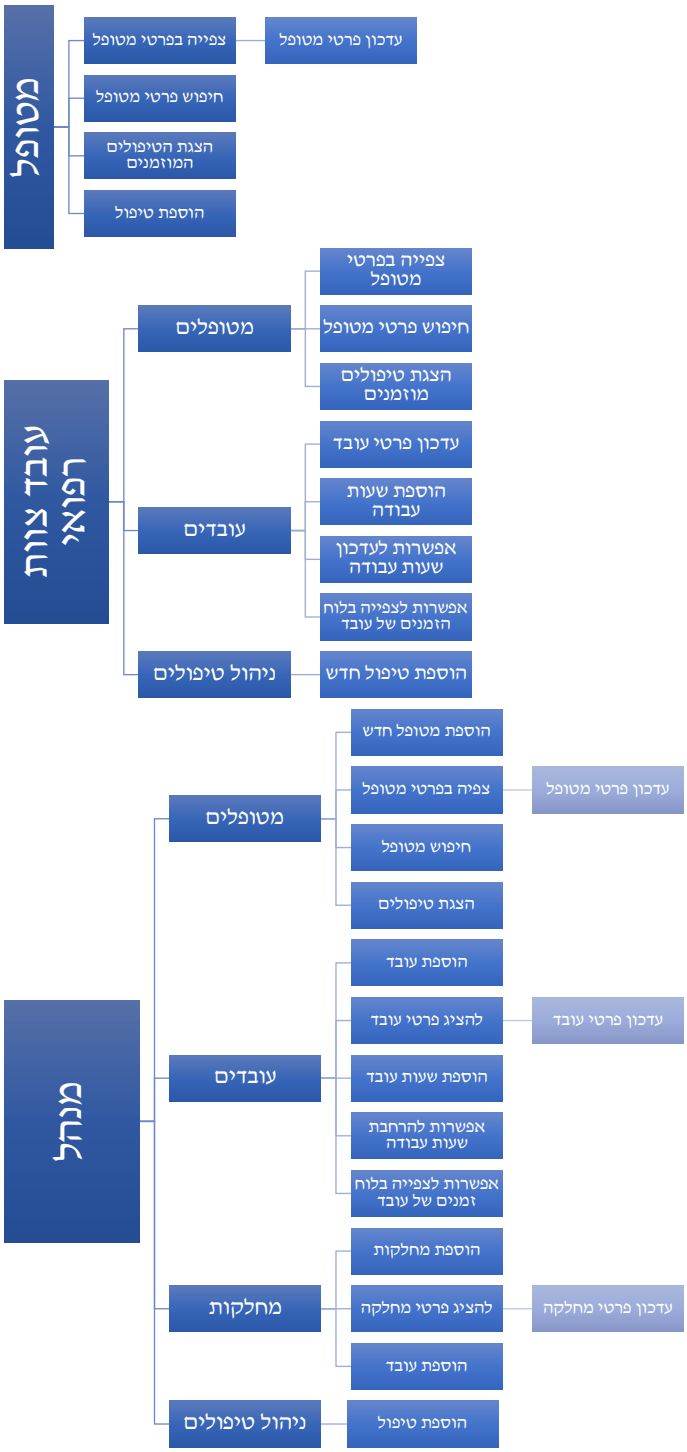
הנחיות להפעלת המערכת:

על מנת שהתוכנה תפעל כנדרש, יש לבצע מספר פעולות פשוטות:

1. היכנס לדיסק המצורף.
2. היכנס לקובץ – DrorCohen.sln
3. התחל את השימוש בעבודה.

התוכנה נכתבה על מערכת Microsoft Visual Studio בשפת C#

4.1 עץ תפריטים



תוכנית:

שם התוכנית	שם פיזי	סיוג	תיאור כללי
עדכון	btnUpdate	פעולה	כפתור המאפשר עדכון הנתונים הקיימים.
שמור	btnSave	פעולה	כפתור השומר במאגר את העדכונים שבוצעו.
קדימה	btnNext	ניווט	כפתור המציג את הרשומה הבאה.
אחורה	btnPrev	ניווט	כפתור המציג את הרשומה הקודמת.
ביטול	btnCancel	פעולה	כפתור המבטל את השינוי שהתבצע ברשומה.
חדש	btnCreate	פעולה	כפתור המאפשר יצירת רשומה חדשה.
כניסה למערכת	btnSubmit	פעולה	בדיקה האם התז שהוכנס זהה לקוד הקיים במערכת המידע, אם כן מעבירה לתופס הראשי, אחרת לא מעבירה.
חיפוש	Search	פעולה	כפתור המאפשר להציג את פרטי מטופל מסוים לפי שמו.
יציאה	button1	פעולה	כפתור המאשר יציאה מהמערכת.

עץ תפריטים:

מהתפריט הראשי מגיעים לכל הטפסים בתוכנה.

מסך ראשי מהתוכנה

שם הטופס	תיאור
טופס כניסה **frmLogIn**	טופס טעינת המערכת.
טופס ראשי **frmOpen**	הטופס הראשי לבחירה בין הטפסים השונים בהם מטפלת המערכת. הטופס הזה הוא התבנית העיקרית של המערכת כאשר הטפסים המשניים מופיעים בתוך container (Panel) שנמצא במרכז התבנית בהתאם לבקשת המשתמש
טופס מטופלים **frmPatient**	טופס המציג את נתוני כל המטופלים הקיימים במערכת ומאפשר את עריכת פרטים והוספת מטופלים חדשים.
טופס עובדים **frmDoctorOrNurse**	טופס המציג את כל נתוני העובדים הקיימים במערכת ומאפשר את עריכתם והוספת עובדים חדשים.
טופס מחלקות **frmDepartment**	טופס המציג את כל נתוני המחלקות הקיימות במערכת ומאפשר את עריכתן והוספת מחלקות חדשות.
טופס הזמנת טיפול **frmOrderTherapyFinal**	טופס המאפשר למטופל להזמין טיפול אצל אחד העובדים
טופס שעות עבודה של עובד **frmSchduleDoctor**	טופס המציג את נתוני כל השעות עבודה הקיימות במערכת ומאפשר את עריכתן והוספת שעות עבודה חדשות חדשות.

התחברות

- אימות

תפריט ראשי

- מטופלים
- עובדי צוות רפואי
- מחלקות
- יציאה

מטופל

- קדימה
- אחורה
- חדש
- ביטול
- שמור
- עדכן
- חיפוש
- הצגת לוח טיפולים עתידיים שנקבעו למטופל
- הזמנת טיפול חדש (פתיחת טופס חדש)

עובדי צוות רפואי

- קדימה
- אחורה
- חדש
- ביטול
- שמור
- עדכן
- הצגת לוח טיפולים של עובד
- שעות עבודה של העובד (פתיחת טופס חדש)

מחלקות

- קדימה
- אחורה
- חדש
- ביטול
- שמור
- עדכן
- הצגת רופאים המשובצים למחלקה
- הוספת עובד קיים למחלקה

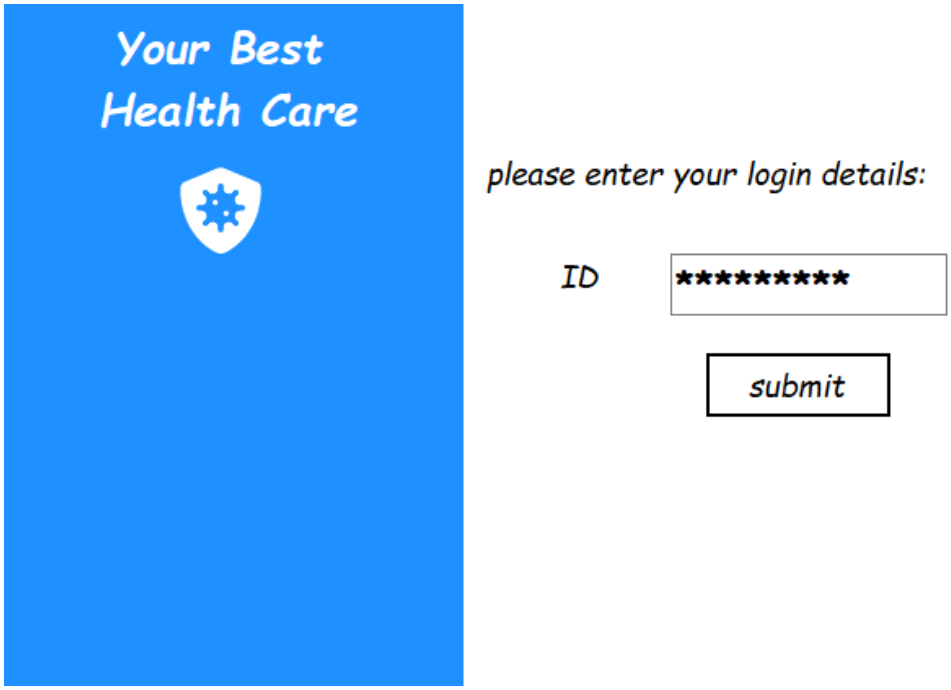
הזמנת טיפול חדש

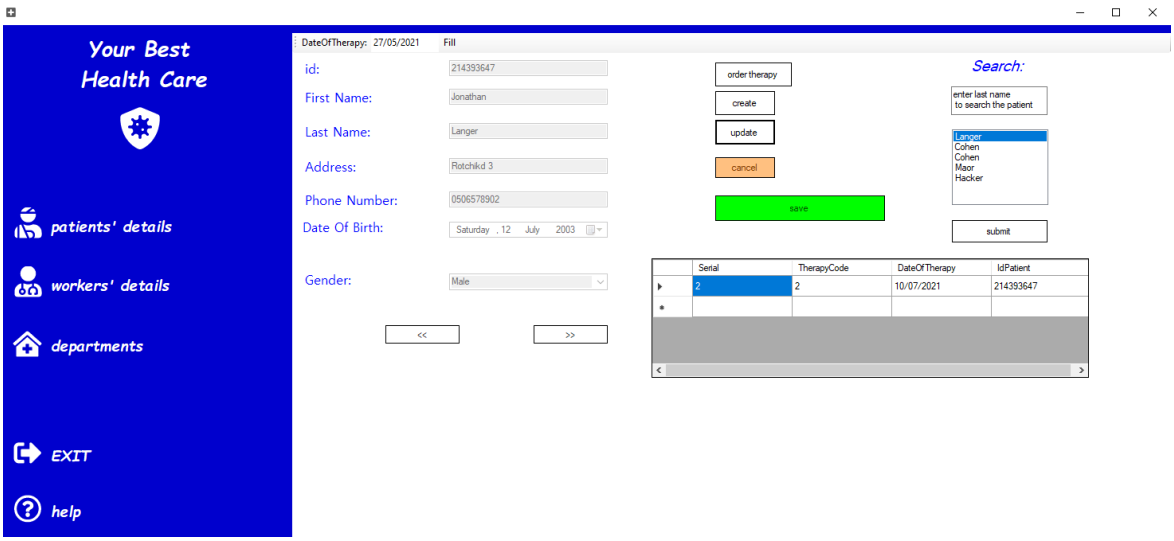
- ביטול
- שמור

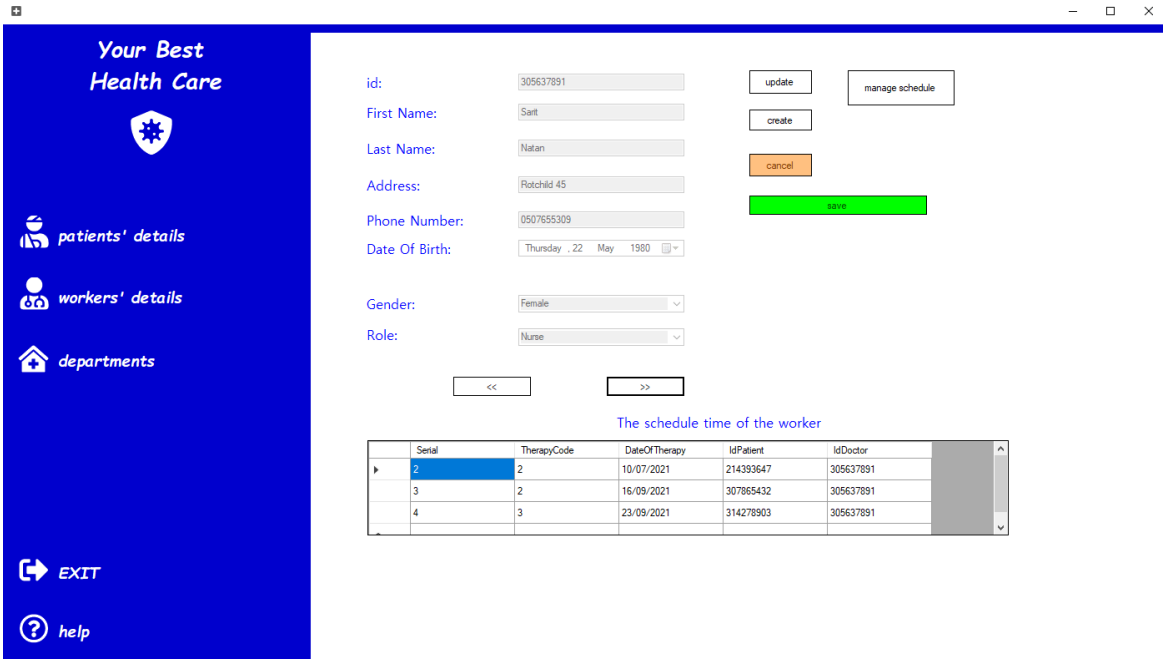
שעות עבודה של העובד

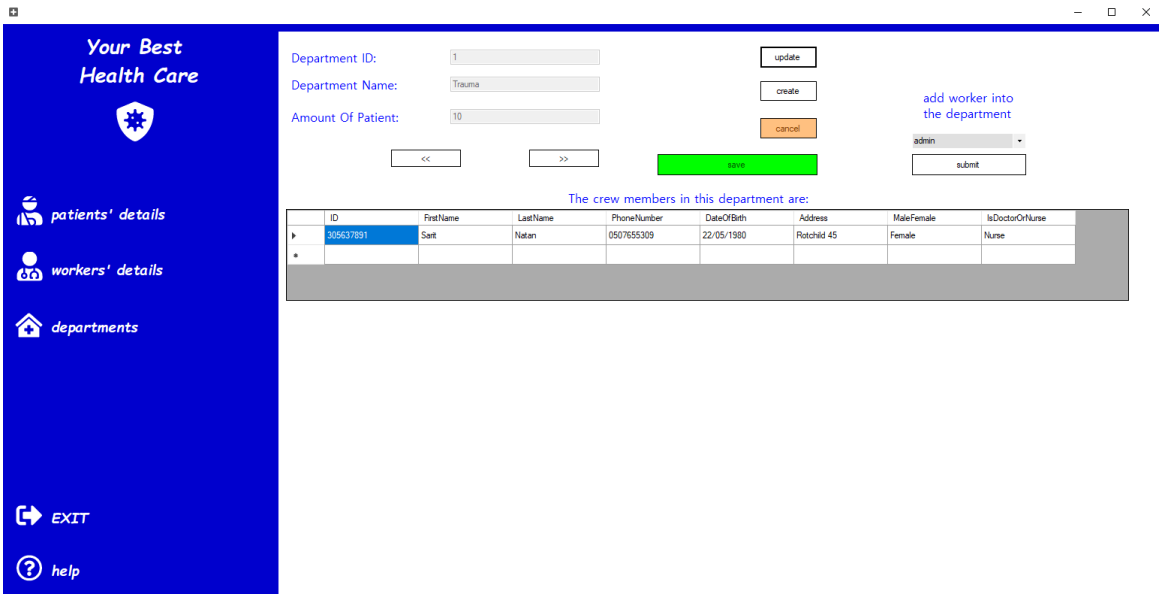
- קדימה
- אחורה
- חדש
- ביטול
- שמור
- עדכן

4.2 עיצוב מסכי קלט




Your Best
Health Care
patients' details
workers' details
departments
EXIT
help
DateOfTherapy: 27/05/2021
Fill
id:
214393647
First Name:
Jonathan
Last Name:
Langer
Address:
Rotchikd 3
Phone Number:
0506578902
Date Of Birth:
Saturday , 12 July 2003
Gender:
Male
<<
>>
order therapy
create
update
cancel
save
Search:
enter last name
to search the patient
Langer
Cohen
Cohen
Maor
Hacker
submit
Serial
TherapyCode
DateOfTherapy
IdPatient
2
2
10/07/2021
214393647


Your Best
Health Care
patients' details
workers' details
departments
EXIT
help
id:
305637891
First Name:
Sarit
Last Name:
Natan
Address:
Rotchild 45
Phone Number:
0507655309
Date Of Birth:
Thursday , 22 May 1980
Gender:
Female
Role:
Nurse
<<
>>
update
create
cancel
save
manage schedule
The schedule time of the worker
Serial
TherapyCode
DateOfTherapy
IdPatient
IdDoctor
2
2
10/07/2021
214393647
305637891
3
2
16/09/2021
307865432
305637891
4
3
23/09/2021
314278903
305637891


Your Best
Health Care
patients' details
workers' details
departments
EXIT
help
Department ID:
1
Department Name:
Trauma
Amount Of Patient:
10
<<
>>
update
create
cancel
save
add worker into
the department
admin
submit
The crew members in this department are:
ID
FirstName
LastName
PhoneNumber
DateOfBirth
Address
MaleFemale
IsDoctorOrNurse
305637891
Sarit
Natan
0507655309
22/05/1980
Rotchild 45
Female
Nurse

קוד הפרויקט

Models. Department.cs

```
using DrorCohen.DB;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DrorCohen.Models
{
    public class Department: IEntity
    {
        private string departmentID;
        private string departmentName;
        private int amountOfPatient;

        public Department(DataRow dr)
        {
            this.departmentID = dr["DepartmentID"].ToString();
            this.departmentName = dr["DepartmentName"].ToString();
            this.amountOfPatient = (int)dr["AmountOfPatient"];
        }
        public Department() { }
        public string DepartmentID
        {
            set { this.departmentID = value; }
            get { return this.departmentID; }
        }
        public string DepartmentName
        {
            set { this.departmentName = value; }
            get { return this.departmentName; }
        }
        public int AmountOfPatient
        {
            set { this.amountOfPatient = value; }
            get { return this.amountOfPatient; }
        }
        public void Populate(DataRow dr)
        {
            dr["DepartmentID"] = DepartmentID;
            dr["DepartmentName"] = DepartmentName;
            dr["AmountOfPatient"] = AmountOfPatient;
        }
    }
}
```

Models

DoctorOrNurse.cs

```
using DrorCohen.Utility;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DrorCohen.DB;
namespace DrorCohen.Models
{
    public class DoctorOrNurse: Patient,IEntity
    {
        private string isDoctorOrNurse;
        public string IsDoctorOrNurse
        {
            set
            {
                if (ValidationUtilites.IsLegalJob(value))
                    this.isDoctorOrNurse = value;
            }
            get { return this.isDoctorOrNurse; }
        }

        public DoctorOrNurse() { }
        public DoctorOrNurse(DataRow dr)
        {
            this.Id = dr["ID"].ToString();
            this.FirstName = dr["FirstName"].ToString();
            this.LastName = dr["LastName"].ToString();
            this.Address = dr["Address"].ToString();
            this.PhoneNumber = dr["PhoneNumber"].ToString();
            this.DateOfBirth = Convert.ToDateTime(dr["DateOfBirth"]);
            this.MaleOrFemale = dr["MaleFemale"].ToString();
            this.IsDoctorOrNurse = dr["IsDoctorOrNurse"].ToString();
        }
        public override void Populate(DataRow dr)
        {
            dr["ID"] = Id;
            dr["FirstName"] = FirstName;
            dr["LastName"] = LastName;
            dr["Address"] = Address;
            dr["PhoneNumber"] = PhoneNumber;
            dr["DateOfBirth"] = DateOfBirth;
            dr["MaleFemale"] = MaleOrFemale;
            dr["IsDoctorOrNurse"] = IsDoctorOrNurse;
        }
    }
}
```

Patient.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DrorCohen.DB;
using DrorCohen.Utility;
namespace DrorCohen.Models
{
    public class Patient: IEntity
    {
        private string id;
        private string firstName;
        private string lastName;
        private string address;
        private string phoneNumber;
        private DateTime dateOfBirth;//string or dateTime
        private string maleOrFemale;
        public string Id
        {
            set
            {
                if (ValidationUtilites.IsLegalId(value))
                    this.id = value;
            }
            get { return this.id; }
        }
        public string FirstName
        {
            set
            {
                if(ValidationUtilites.IsLegalName(value))
                    this.firstName = value;
            }
            get { return this.firstName; }
        }
        public string LastName
        {
            set
            {
                if (ValidationUtilites.IsLegalName(value))
                    this.lastName = value;
            }
            get { return this.lastName; }
        }
        public string Address
        {
            set
            {
                if (ValidationUtilites.IsLegalAddress(value))
                    this.address = value;
            }
            get { return this.address; }
        }
        public string PhoneNumber
        {
            set
```

```
            {
                //if(ValidationUtilites.IsLegalPhoneNumber(value))
                    this.phoneNumber = value;
            }
            get { return this.phoneNumber; }
        }
        public DateTime DateOfBirth
        {
            set { this.dateOfBirth = value; }
            get { return this.dateOfBirth; }
        }
        public string MaleOrFemale
        {
            set
            {
               if(ValidationUtilites.IsLegalSex(value))
                   this.maleOrFemale = value;
            }
            get { return this.maleOrFemale; }
        }
        public Patient() { }
        public Patient(DataRow dr)
        {
            this.Id = dr["ID"].ToString();
            this.FirstName = dr["FirstName"].ToString();
            this.LastName = dr["LastName"].ToString();
            this.Address = dr["Address"].ToString();
            this.PhoneNumber = dr["PhoneNumber"].ToString();
            this.DateOfBirth =  Convert.ToDateTime(  dr["DateOfBirth"]);
            this.MaleOrFemale = dr["MaleFemale"].ToString();
        }
        public virtual void Populate(DataRow dr)
        {
            dr["ID"] = Id;
            dr["FirstName"] = FirstName;
            dr["LastName"] = LastName;
            dr["Address"] = Address;
            dr["PhoneNumber"] = PhoneNumber;
            dr["DateOfBirth"] = DateOfBirth;
            dr["MaleFemale"] = MaleOrFemale;
        }
    }
}
```

ScheduleDoctorMeeting.cs

```
using DrorCohen.DB;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DrorCohen.Models
{
    public class ScheduleDoctorMeeting:IEntity
    {
        private string therapyCode;
        private string whoCanGiveTheTherapy;
        private string idDoctor;
        private int day;
        private string hour;

        public ScheduleDoctorMeeting() { }
        public ScheduleDoctorMeeting(DataRow dr)
        {
            this.TherapyCode = dr["TherapyCode"].ToString();
            this.whoCanGiveTheTherapy = dr["WhoCanGiveTheTherapy"].ToString();
            this.idDoctor = dr["IdDoctor"].ToString();
            this.day =(int)(dr["day"]);
            this.hour = dr["hour"].ToString();
        }

        public string TherapyCode
        {
            set { this.therapyCode = value; }
            get { return this.therapyCode; }
        }
        public string WhoCanGiveTheTherapy
        {
            set { this.whoCanGiveTheTherapy = value; }
            get { return this.whoCanGiveTheTherapy; }
        }
        public string IdDoctor
        {
            set { this.idDoctor = value; }
            get { return this.idDoctor; }
        }
        public int Day
        {
            set { this.day = value; }
            get { return this.day; }
        }
        public string Hour
        {
            set { this.hour = value; }
            get { return this.hour; }
        }

        public void Populate(DataRow dr)
        {
            dr["TherapyCode"] = TherapyCode;
            dr["WhoCanGiveTheTherapy"] = WhoCanGiveTheTherapy;
            dr["IdDoctor"] = idDoctor;
            dr["day"] = day;
```

```
            dr["hour"] = hour;
        }
    }
}
```

SpecificTherapyForPatient.cs

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

namespace DrorCohen.Models
{
    class SpecificTherapyForPatient
    {
        private string serial;
        private string therapyCode;
        private DateTime dateOfTherapy;
        //private string wasTheTherapyGiven;
        public string Serial
        {
            set
            {
                this.serial = value;
            }
            get
            {
                return this.serial;
            }
        }
        public string TherapyCode
        {
            set
            {
                this.therapyCode = value;
            }
            get
            {
                return this.therapyCode;
            }
        }
        public DateTime DateOfTherapy
        {
            set
            {
                this.dateOfTherapy = value;
            }
            get
            {
                return this.dateOfTherapy;
            }
        }
        //public string WasTheTherapyGiven
        //{
        //    set
        //    {
        //        this.wasTheTherapyGiven = value;
        //    }
        //    get
        //    {
        //        return this.wasTheTherapyGiven;
        //    }
        //}
    }
}
```

Utility

AddState.cs

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

namespace DrorCohen.Utility
{
    enum AddState
    {
        ADDNEW,UPDATE,NAVIGATE
    };
}
```

StatusMessage.cs

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

namespace DrorCohen.Utility
{
    enum StatusMessage
    {
        APPROVED=1,ERROR=2,INFO=3
    };
}
```

ValidationUtilities.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace DrorCohen.Utility
{
    public static class ValidationUtilites
    {
        public static bool isHour(string s)
        {
            if (s.Length != 5)
                return false;
            if (s[2] != ':')
                return false;
            for(int i = 0; i < 5; i++)
            {
                if (i == 2)
                    continue;
                if (s[i] < '0' || s[i] > '9')
                    return false;
            }
            int x = int.Parse(s[0] + "" + s[1]);
            if (x < 0 || x > 23)
```

```
            return false;
        x = int.Parse(s[3] + "" + s[4]);
        if (x < 0 || x > 59)
            return false;
        return true;
    }
    public static bool isDay(string s)
    {
        return s.Equals("Sun") ||
            s.Equals("Mon") ||
            s.Equals("Tue") ||
            s.Equals("Wed") ||
            s.Equals("Thu") ||
            s.Equals("Fri") ||
            s.Equals("Sat");
    }
    public static bool isNumber(string s)
    {
        foreach (char c in s)
            if (c < '0' || c > '9')
                return false;
        return true;
    }
    public static bool IsLegalHour(string hour)
    {
        if (hour.Length != 5)
            return false;
        if (hour[2] != ':')
            return false;
        int x;
        string s = hour[0] + "" + hour[1];
        x = int.Parse(s);
        if (x < 0 || x > 23)
            return false;
        s = hour[3] + "" + hour[4];
        x = int.Parse(s);
        if (x < 0 || x > 59)
            return false;
        return true;
    }
    public static bool LegalId(string s)
    {
        /*if (s.Length == 0)
            return false;*/
        int x;
        if (!int.TryParse(s, out x))
            return false;
        if (s.Length < 5 || s.Length > 9)
            return false;
        for (int i = s.Length; i < 9; i++)
            s = "0" + s;
        int sum = 0;
        for (int i = 0; i < 9; i++)
        {
            char c = s[i]; ;
            int k = ((i % 2) + 1) * (c - '0');
            if (k > 9)
                k -= 9;
            sum += k;

        }
        return sum % 10 == 0;
```

```
        }

        public static bool CheckIdNumber(String s)
        {
            return s.Length == 9;
        }

        public static bool PhoneNumber(string num)
        {
            string pattern = @"\b0[2-4 7-9]-[2-9]\d{6}";
            Regex r = new Regex(pattern);
            return r.IsMatch(num);
        }
        public static bool LegalName(string name)
        {
            string pattern = @"\b[א-ת- ]+";
            Regex r = new Regex(pattern);
            return r.IsMatch(name);
        }
        public static int GetAge(DateTime d)
        {
            DateTime t = DateTime.Today;
            int age = t.Year - d.Year;
            if (t < d.AddYears(age)) age--;
            return age;
        }

        public static bool GreaterThanZero(int num)
        {
            return num > 0;
        }

        public static bool IsHebrewLetter(char c)
        {
            string otiyot = "'אבגדהוזחטיכלמנסעפצקרשתדסוף";
            if (otiyot.IndexOf(c) == -1)
                return false;
            return true;
        }

        public static bool IsEnglishLetter(char c)
        {
            string otiyot =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'";
            if (otiyot.IndexOf(c) == -1)
                return false;
            return true;
        }

        public static bool IsDigits(char c)
        {
            string digits = "0123456789";
            if (digits.IndexOf(c) == -1)
                return false;
            return true;
        }

        public static bool IsLegalItemName(string word)
        {
            foreach (char c in word)
                if (IsDigits(c) == false && IsHebrewLetter(c) == false && c !=
'-' && c != ' ' && IsEnglishLetter(c) == false)
```

```
                return false;
            return true;
        }

        public static bool IsLegalName(string word)
        {
            foreach (char c in word)
                if (IsHebrewLetter(c) == false && IsEnglishLetter(c) == false
&& (c != '-'||c!= ' '))
                return false;
            return true;
        }
        public static bool IsLegalAddress(string word)
        {
            foreach (char c in word)
                if (IsHebrewLetter(c) == false && IsEnglishLetter(c) == false
&& !(c<='9'&&c>='0') && (c != '-' && c != ' '))
                return false;
            return true;
        }

        public static bool IsLegalCity(string word)
        {
            foreach (char c in word)
                if (IsHebrewLetter(c) == false && IsEnglishLetter(c) == false
&& c != '-' && c != ' ')
                return false;
            return true;
        }

        public static bool IsLegalId(string id)
        {
            string word = id;
            if (word.Length != 9)
                return false;
            foreach (char c in word)
                if (IsDigits(c) == false)
                return false;
            return true;
        }
        public static bool IsLegalDigit(string dig)
        {
            string digit = dig;
            foreach (char c in digit)
                if (digit.IndexOf(c) == -1)
                return false;
            return true;
        }
        public static bool IsLegalZipcode(string zip)
        {
            string zipcode = zip;
            if (zipcode.Length != 5)
                return false;
            foreach (char c in zipcode)
                if (IsDigits(c) == false)
                return false;
            return true;
        }
        public static bool IsLegalCNumberVisa(string cnum)
        {
            string creditnumber = cnum;
            if (creditnumber.Length != 16)
```

```
                return false;
            foreach (char c in creditnumber)
                if (IsDigits(c) == false)
                    return false;
            return true;
        }
        public static bool IsLegalCNumberAmericanexpress(string cnum)
        {
            string creditnumber = cnum;
            if (creditnumber.Length != 15)
                return false;
            foreach (char c in creditnumber)
                if (IsDigits(c) == false)
                    return false;
            return true;
        }
        public static bool IsLegalThreeDig(string tdig)
        {
            string threedig = tdig;
            if (threedig.Length != 3)
                return false;
            foreach (char c in threedig)
                if (IsDigits(c) == false)
                    return false;
            return true;
        }
        public static bool IsLegalItemId(string id)
        {
            string word = id;
            if (word.Length != 5)
                return false;
            foreach (char c in word)
                if (IsDigits(c) == false)
                    return false;
            return true;
        }

        public static bool IsLegalSex(string sex)
        {
            return(sex=="Male"||sex=="Female");
        }
        public static bool IsLegalJob(string job)
        {
            return (job == "Doctor" || job == "Nurse" || job == "Other");
        }

        internal static bool IsPhoneNumber(string text)
        {
            if (text.Length != 10)
                return false;
            foreach (char c in text)
                if (!(c >= '0' && c <= '9'))
                    return false;
            return true;
        }
    }
}
```

Data

DAL.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.OleDb;
using System.Collections;
using System.Windows;
using System.Windows.Forms;

namespace DrorCohen.DATA
{
    public class DAL
    {//הגדרת משתנים
        private static DAL instance;
        private OleDbConnection con;
        private DataSet ds;
        private Hashtable adapters;

        //בנאי המחלקה
        private DAL(string connectionString)
        {
            con = new OleDbConnection(connectionString);
            ds = new DataSet();
            adapters = new Hashtable();
        }
        //בניית עצם של  וחיבור למאגרDAL
        public static DAL GetInstance()
        {

            if (instance == null)
            {
                string path = System.IO.Directory.GetCurrentDirectory();
                int x = path.IndexOf("\\bin");
                path = path.Substring(0, x) + "\\bin\\Debug\\HMO.accdb";
                //Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=E:\WindowsFormsApplication2\WindowsFormsApplication1\Data\projects.accdb
                instance = new DAL(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source='" + path + "';Persist Security Info=True");
            }
            return instance;
        }

        /// פעולה המוסיפה טבלה ל- Dataset
        /// ומכינה את האדפטר לביצוע כל פעולות העידכון
        /// "tableName">שם הטבלה<
        ///"sqlStat">שאילתת שליפה<
        public bool AddTable(string tableName, string sqlStat)
        {
            if (!ds.Tables.Contains(tableName))
            {
                OleDbDataAdapter adapter = new OleDbDataAdapter(sqlStat, con);
                OleDbCommandBuilder builder = new OleDbCommandBuilder(adapter);
                adapter.InsertCommand = builder.GetInsertCommand();
                adapter.UpdateCommand = builder.GetUpdateCommand();
                adapter.DeleteCommand = builder.GetDeleteCommand();
                adapter.Fill(ds, tableName);
                adapters.Add(tableName, adapter);
```

```
            return true;
        }
        return false;
    }

    //בנית פעולת השאילתא ב select

    public bool AddTable(string tableName)
    {
        return AddTable(tableName, "Select * from " + tableName);
    }
    //קבלת נתוני טבלה
    public DataTable GetTable(string tableName)
    {
        return ds.Tables[tableName];
    }
    //מחיקת טבלה
    public bool RemoveTable(string tableName)
    {
        bool suceed = true;
        try
        {
            ds.Tables.Remove(tableName);
            adapters.Remove(tableName);
        }
        catch
        {
            suceed = false;
        }
        return suceed;
    }
    //עדכון נתונים במסד הנתונים
    public int ExecuteNonQuery(string sqlQry)
    {
        int x;
         OleDbCommand command = con.CreateCommand();
        command.CommandText = sqlQry;
        con.Open();

        try
        {
            x = command.ExecuteNonQuery();
            if (x > 0)

                MessageBox.Show("Insert succeed");

        }

        catch (Exception ex)
        {
            x = 0;
            MessageBox.Show(ex.Message);
        }





con.Close();
        return x;
    }
```

```
        //שליפת הנתונים מהמאגר לפי השאילתא
        public object ExecuteScalarQuery(string sqlStr)
        {
            OleDbCommand command = con.CreateCommand();
            command.CommandText = sqlStr;
            con.Open();
            object obj = command.ExecuteScalar();
            con.Close();
            return obj;

        }
        //עדכון נתונים למאגר
        public void Update(string tableName)
        {
           OleDbDataAdapter adapter = (OleDbDataAdapter)adapters[tableName];
           adapter.Update(ds, tableName);
        }
        //עדכון לכל הטבלאות
        public void Update()
        {
            foreach (DataTable table in ds.Tables)
            {
                Update(table.TableName);
            }
        }
        //הוספת קשרי גומלין
        public void AddRelation(string relName, DataColumn primaryKey,
DataColumn foreignKey)
        {
            try
            {
                ds.Relations.Add(relName, primaryKey, foreignKey);
            }
            catch { }
        }
        //הסרת קשרי גומלין
        public void RemoveRelation(string relName)
        {
            try
            {
                ds.Relations.Remove(relName);
            }
            catch { }
        }
        //הצגת טבלה לפי שאילתה

        public DataTable GetDisplayTable(string name, string sqlStat)
        {
            DataTable dt = new DataTable();
            OleDbDataAdapter ad = new OleDbDataAdapter(sqlStat, con);
            ad.Fill(dt);

            return dt;
        }



    }
}
```

DB

DepartmentDB.cs

```
using DrorCohen.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DrorCohen.DATA;
using System.Data;

namespace DrorCohen.DB
{
    public class DepartmentDB:GeneralDB
    {
        public DepartmentDB() : base("Department", "DepartmentID") { }
        public new Department GetCurrentRow()
        {
            return new Department(base.GetCurrentRow());
        }
        public DataView GetDataView()
        {
            DAL d = DAL.GetInstance();
            DataTable a = d.GetDisplayTable("Department", "Select * From
Department");
            return new DataView(table);
        }
        //החזרת מפתח ראשי האחרון
        public /*int*/ string GetKey()
        {
            int x = currentRow;
            GoToLast();
            //int key = Convert.ToInt32(base.GetCurrentRow()[primaryKey]) + 1;
            string key =
Convert.ToString(Convert.ToInt32(base.GetCurrentRow()[primaryKey]) + 1);
            currentRow = x;
            return key;
        }
        public string GetKeyName()
        {
            return "DepartmentID";
        }
        public void Update(Patient cos)
        {
            DataRow dr = base.GetCurrentRow();
            cos.Populate(dr);
        }
        //public void Add(Customer cus)
        //{
        //    DataRow dr = base.Add(cus);
        //    cus.Populate(dr);
        //    //base.Add(dr);
        //    Find(cus.Id);
        //}
        //public static DataTable GetAllCustomer()
        //{
        //    DataTable tb;
        //    string sqlStat = "select customer_ID, last_name + ' ' +
first_name as [fullName] From customers ";
        //    tb = DAL.GetInstance().GetDisplayTable("customer", sqlStat);
```

```
        //    return tb;
        //}
        //public DataView GetDataView()
        //{
        //    DAL d = DAL.GetInstance();
        //    DataTable a = d.GetDisplayTable("customers", "Select customer_ID,
first_name,last_name From customers");
        //    return new DataView(table);

        //}



    }
}
```

DoctorOrNurseDB.cs

```
using DrorCohen.Models;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DrorCohen.DB
{
    public class DoctorOrNurseDB:GeneralDB
    {
        public DoctorOrNurseDB() : base("DoctorOrNurse", "ID") { }
        public new DoctorOrNurse GetCurrentRow()
        {
            return new DoctorOrNurse(base.GetCurrentRow());
        }
        //החזרת מפתח ראשי האחרון
        //public /*int*/ string GetKey()
        //{
        //    int x = currentRow;
        //    GoToLast();
        //    //int key = Convert.ToInt32(base.GetCurrentRow()[primaryKey]) +
1;
        //    string key =
Convert.ToString(Convert.ToString(base.GetCurrentRow()[primaryKey]) + 1);
        //    currentRow = x;
        //    return key;
        //}
        public string GetKeyName()
        {
            return "ID";
        }
        public void Update(DoctorOrNurse cos)
        {
            DataRow dr = base.GetCurrentRow();
            cos.Populate(dr);
        }
        //public void Add(Customer cus)
        //{
        //    DataRow dr = base.Add(cus);
        //    cus.Populate(dr);
        //    //base.Add(dr);
```

```
        //    Find(cus.Id);
        //}
        //public static DataTable GetAllCustomer()
        //{
        //    DataTable tb;
        //    string sqlStat = "select customer_ID, last_name + ' ' +
first_name as [fullName] From customers ";
        //    tb = DAL.GetInstance().GetDisplayTable("customer", sqlStat);
        //    return tb;
        //}
        //public DataView GetDataView()
        //{
        //    DAL d = DAL.GetInstance();
        //    DataTable a = d.GetDisplayTable("customers", "Select customer_ID,
first_name,last_name From customers");
        //    return new DataView(table);

        //}
    }
}
```

GeneralDB.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DrorCohen.Models;
using System.Data;
using DrorCohen.DATA;
namespace DrorCohen.DB
{
    public abstract class GeneralDB
    {//המאפיינים
        protected DataTable table;
        protected int currentRow;
        protected string primaryKey;

        //  הבנאי פונה לדאל כדי להביא את נתוני הטבלה מהמאדר לזכרון. שיטה בונה
        public GeneralDB(string tableName, string primaryKey)
        {
            DAL.GetInstance().AddTable(tableName);
            table = DAL.GetInstance().GetTable(tableName);
            this.primaryKey = primaryKey;
            if (IsEmpty())
                currentRow = -1;
            else
                currentRow = 0;
        }
        //פעולות ניווט
        #region NAVIGATION

        /// מעדכן את השרה הנוכחית לשורה הראשונה

        public void GoToFirst()
        {
            if (IsEmpty())
                throw new Exception("ניווט על טבלה ריקה");
            currentRow = 0;
        }
```

```
/// מעדכן את השורה הנוכחית לשורה האחרונה

public void GoToLast()
{
    if (IsEmpty())
        throw new Exception("ניווט על טבלה ריקה");
    currentRow = table.Rows.Count - 1;
}

/// עובר לשורה הבאה בטבלה. אם אנחנו בסוף חוזרים להתחלה

public void MoveNext()
{
    if (IsEmpty())
        throw new Exception("ניווט על טבלה ריקה");
    currentRow = (currentRow + 1) % table.Rows.Count;
}

/// moves to the previous object. If reaches the beginning, goes back
/// to the end

public void MovePrev()
{
    if (IsEmpty())
        throw new Exception("ניווט על טבלה ריקה");
    if (this.currentRow == 0)
        currentRow = table.Rows.Count - 1;
    else
        --currentRow;
}


/// חיפוש האובייקט באמצעות המפתח
   ///"key">the key being looked for
/// <returns>true if found and false if no such row  exists
public bool Find(object key)
{
    int r = 0;
    foreach (DataRow dr in table.Rows)
    {
        if (dr[primaryKey].Equals(key))
        {
            currentRow = r;
            return true;
        }
        else
            r++;
    }
    return false;
}
public bool FindString(object key)
{
    int r = 0;
    foreach (DataRow dr in table.Rows)
    {
        if (dr[primaryKey].ToString()==(key).ToString())
        {
            currentRow = r;
            return true;
        }
        else
```

```
            r++;
    }
    return false;
}

#endregion

#region GENERAL OPERATIONS

/// מחזיר מספר השורות בטבלה

/// <returns>number of rows</returns>
public int Size()
{
    return table.Rows.Count;
}

/// בודק האם הטבלה ריקה

/// <returns> true if empty, false if not empty<

public bool IsEmpty()
{
    return table.Rows.Count == 0;
}
public virtual void Save()
{
   DAL.GetInstance().Update(table.TableName);
}
#endregion

public DataRow[] Filter(string filterString)
{
    if (filterString.Trim().Length == 0)
        return table.Select();
    return table.Select(filterString);
}
#region CRUD

//הוספה קריאה עדכוון ומחיקה
//מוסיפה שורה לטבלה עם שימוש בממשק
public void AddRow(IEntity obj)
{
    DataRow dr = table.NewRow();
    obj.Populate(dr);
    table.Rows.Add(dr);
   Find(dr[primaryKey]);

}
//עדכון נתונים עם ממשק באובייקט
public void UpdateRow(IEntity obj)
{
    DataRow dr = GetCurrentRow();
    obj.Populate(dr);

}
public virtual void DeleteCurrentRow()
{
    DataRow dr = GetCurrentRow();
    table.Rows.Remove(dr);
    dr.Delete();
    if (IsEmpty())
```

```
                this.currentRow = -1;
            if (this.currentRow == Size())
                this.currentRow = 0;
        }
        //מאחזר שורה נוכחית בטבלה
        public virtual DataRow GetCurrentRow()
        {
            return table.Rows[currentRow];
        }
        #endregion

        public DataTable GetTable() { return this.table; }

    }

}
```

IEntity.cs

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

namespace DrorCohen.DB
{
    public interface IEntity
    {
        void Populate(DataRow dr);
    }
}
```

PatientDB.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DrorCohen.Models;
using System.Data;
using DrorCohen.DATA;

namespace DrorCohen.DB
{
    public class PatientDB : GeneralDB
    {
        public PatientDB() : base("Patient", "ID") { }
        public new Patient GetCurrentRow()
        {
            return new Patient(base.GetCurrentRow());
        }
        public DataView GetDataView()
        {
            DAL d = DAL.GetInstance();
            DataTable a = d.GetDisplayTable("Patient", "Select * From
Patient");
```

```
            return new DataView(table);

        }
        //החזרת מפתח ראשי האחרון
        //public /*int*/ string GetKey()
        //{
        //    int x = currentRow;
        //    GoToLast();
        //    //int key = Convert.ToInt32(base.GetCurrentRow()[primaryKey]) +
1;
        //    string key =
Convert.ToString(Convert.ToString(base.GetCurrentRow()[primaryKey]) + 1);
        //    currentRow = x;
        //    return key;
        //}
        public string GetKeyName()
        {
            return "ID";
        }
        public void Update(Patient cos)
        {
            DataRow dr = base.GetCurrentRow();
            cos.Populate(dr);
        }
        //public void Add(Customer cus)
        //{
        //    DataRow dr = base.Add(cus);
        //    cus.Populate(dr);
        //    //base.Add(dr);
        //    Find(cus.Id);
        //}
        //public static DataTable GetAllCustomer()
        //{
        //    DataTable tb;
        //    string sqlStat = "select customer_ID, last_name + ' ' +
first_name as [fullName] From customers ";
        //    tb = DAL.GetInstance().GetDisplayTable("customer", sqlStat);
        //    return tb;
        //}
        //public DataView GetDataView()
        //{
        //    DAL d = DAL.GetInstance();
        //    DataTable a = d.GetDisplayTable("customers", "Select customer_ID,
first_name,last_name From customers");
        //    return new DataView(table);
    }
}
```

ScheduleDoctorMeeting.cs

```
using DrorCohen.DATA;
using DrorCohen.Models;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DrorCohen.DB
{
    public class ScheduleDoctorMeetingDB : GeneralDB
    {
        public ScheduleDoctorMeetingDB() : base("MeetingDoctor", "TherapyCode")
{ }
        public new ScheduleDoctorMeeting GetCurrentRow()
        {
           // DataRow dr = base.GetCurrentRow();
            return new ScheduleDoctorMeeting(base.GetCurrentRow());
        }
        public DataView GetDataView()
        {
            DAL d = DAL.GetInstance();
            DataTable a = d.GetDisplayTable("MeetingDoctor", "Select * From
MeetingDoctor");
            return new DataView(table);
        }
        //החזרת מפתח ראשי האחרון
        public /*int*/ string GetKey()
        {
            int x = currentRow;
            GoToLast();
            //int key = Convert.ToInt32(base.GetCurrentRow()[primaryKey]) + 1;
            string key =
Convert.ToString(Convert.ToInt32(base.GetCurrentRow()[primaryKey]) + 1);
            currentRow = x;
            return key;
        }
        public string GetKeyName()
        {
            return "TherapyCode";
        }
        public void Update(ScheduleDoctorMeeting cos)
        {
            DataRow dr = base.GetCurrentRow();
            cos.Populate(dr);
        }
        //public void Add(Customer cus)
        //{
        //    DataRow dr = base.Add(cus);
        //    cus.Populate(dr);
        //    //base.Add(dr);
        //    Find(cus.Id);
        //}
        //public static DataTable GetAllCustomer()
        //{
        //    DataTable tb;
        //    string sqlStat = "select customer_ID, last_name + ' ' +
first_name as [fullName] From customers ";
        //    tb = DAL.GetInstance().GetDisplayTable("customer", sqlStat);
```

```
        //    return tb;
        //}
        //public DataView GetDataView()
        //{
        //    DAL d = DAL.GetInstance();
        //    DataTable a = d.GetDisplayTable("customers", "Select customer_ID,
first_name,last_name From customers");
        //    return new DataView(table);

        //}
    }
}
```

GUI

frmDepartment.cs

```
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 DrorCohen.DB;
using DrorCohen.Models;
using DrorCohen.Utility;
namespace DrorCohen.Gui
{
    public partial class frmDepartment : Form
    {
        private Form parent;
        private AddState state;
        private DepartmentDB departments;
        public frmDepartment(Form frmmain)
        {
            this.parent = frmmain;
            InitializeComponent();
            departments = new DepartmentDB();
            state = AddState.NAVIGATE;
            Populate(departments.GetCurrentRow());
            SetButtonStates(true);
        }
        private void Populate(Department d)
        {
            inputId.Text = d.DepartmentID;
            inputDepartmentName.Text = d.DepartmentName;
            inputAmountOfPatient.Text = d.AmountOfPatient.ToString();
        }
        private bool UpdateObject(Department d)
        {
            bool ok = true;
            try
            {
                d.DepartmentID = inputId.Text;
                errorProvider1.SetError(inputId, "");
            }
            catch (Exception ex)
            {
                ok = false;
                errorProvider1.SetError(inputId, ex.Message);
            }
            try
            {
                d.DepartmentName = inputDepartmentName.Text;
                errorProvider1.SetError(inputId, "");
            }
            catch (Exception ex)
            {
                ok = false;
                errorProvider1.SetError(inputId, ex.Message);
            }
            try
```

```
            {
                d.AmountOfPatient = Convert.ToInt32(inputAmountOfPatient.Text);
                errorProvider1.SetError(inputId, "");
            }
            catch (Exception ex)
            {
                ok = false;
                errorProvider1.SetError(inputId, ex.Message);
            }
            return ok;
        }
        private void SetButtonStates(bool b)
        {
            next.Enabled = b;
            prev.Enabled = b;
            Add.Enabled = b;
            update.Enabled = b;
            cancel.Enabled = !b;
            save.Enabled = !b;
            inputId.Enabled = !b;
            inputDepartmentName.Enabled = !b;
            inputAmountOfPatient.Enabled = !b;
        }
        private void inputId_TextChanged(object sender, EventArgs e)
        {

this.doctorOrNurseTableAdapter.Show(this.doctorAndDepartmentConnection1.DoctorO
rNurse,inputId.Text);

//this.doctorOrNurseTableAdapter.FillBy(this.hMODataSet.DoctorOrNurse,
inputId.Text);
        }

        private void save_Click(object sender, EventArgs e)
        {
            Department d = new Department();
            if (UpdateObject(d))
            {
                if (state == AddState.ADDNEW)
                    departments.AddRow(d);
                else
                    departments.UpdateRow(d);
            }
            SetButtonStates(true);
            state = AddState.NAVIGATE;
            departments.Save();
        }

        private void next_Click(object sender, EventArgs e)
        {
            departments.MoveNext();
            Populate(departments.GetCurrentRow());
        }

        private void prev_Click(object sender, EventArgs e)
        {
            departments.MovePrev();
            Populate(departments.GetCurrentRow());
        }

        private void update_Click(object sender, EventArgs e)
        {
```

```
            SetButtonStates(false);
            state = AddState.UPDATE;
        }

        private void Add_Click(object sender, EventArgs e)
        {
            Clear();
            inputId.Text = departments.GetKey();
            state = AddState.ADDNEW;
            SetButtonStates(false);
        }
        private void Clear()
        {
            foreach (Control c in Controls)
            {
                if (c is TextBox)
                    (c as TextBox).Text = "";
                if (c is ComboBox)
                    (c as ComboBox).Text = "";
                if (c is DateTimePicker)
                    (c as DateTimePicker).Value = DateTime.Today;
            }
        }

        private void cancel_Click(object sender, EventArgs e)
        {
            Populate(departments.GetCurrentRow());
            SetButtonStates(true);
        }

        private void frmDepartment_FormClosing(object sender,
FormClosingEventArgs e)
        {
            this.Hide();
            this.Dispose();
            parent.Show();
        }

        private void doctorOrNurseBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            /*this.doctorOrNurseBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.hMODataSet);*/

        }

        private void frmDepartment_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'createConnection.ConnectionDoctorDepartment' table. You can move, or remove
it, as needed.
   ////////
this.connectionDoctorDepartmentTableAdapter1.Fill(this.createConnection.Connect
ionDoctorDepartment);
            // TODO: This line of code loads data into the 's.DoctorOrNurse'
table. You can move, or remove it, as needed.
            this.doctorOrNurseTableAdapter1.Fill(this.s.DoctorOrNurse);
            // TODO: This line of code loads data into the
'doctorAndDepartmentConnection1.ConnectionDoctorDepartment' table. You can
move, or remove it, as needed.
```

```
this.connectionDoctorDepartmentTableAdapter.Fill(this.doctorAndDepartmentConnec
tion1.ConnectionDoctorDepartment);
            // TODO: This line of code loads data into the
'doctorAndDepartmentConnection1.DoctorOrNurse' table. You can move, or remove
it, as needed.

//this.doctorOrNurseTableAdapter.Fill(this.doctorAndDepartmentConnection1.Docto
rOrNurse);
            // TODO: This line of code loads data into the
'hMODataSet.DoctorOrNurse' table. You can move, or remove it, as needed.
        }

        private void doctorOrNurseDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            //int rowIndex = doctorOrNurseDataGridView.CurrentCell.RowIndex;
            //string id =
doctorOrNurseDataGridView.Rows[rowIndex].Cells[0].Value.ToString();
            //frmDoctorOrNurse f = new frmDoctorOrNurse(id);
            //f.Show();
        }

        private void id_Click(object sender, EventArgs e)
        {

        }

        private void fillByToolStripButton_Click(object sender, EventArgs e)
        {
            try
            {

this.doctorOrNurseTableAdapter.FillBy(this.doctorAndDepartmentConnection1.Docto
rOrNurse);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }

        private void
doctorAndDepartmentConnection1BindingSource_CurrentChanged(object sender,
EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {

//doctorAndDepartmentConnection1.ConnectionDoctorDepartment.AddConnectionDoctor
DepartmentRow(inputId.Text, comboBox1.SelectedItem.ToString())

//doctorAndDepartmentConnection1.ConnectionDoctorDepartment.AddConnectionDoctor
DepartmentRow
            //(inputId.Text, comboBox1.SelectedItem.ToString());
```

```
this.connectionDoctorDepartmentTableAdapter.Update(doctorAndDepartmentConnectio
n1.ConnectionDoctorDepartment);
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
    }
}
```

frmDoctorOrNurse.cs

```
using DrorCohen.DB;
using DrorCohen.Models;
using DrorCohen.Utility;
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;

namespace DrorCohen.Gui
{
    public partial class frmDoctorOrNurse : Form
    {
        private AddState state;
        private DoctorOrNurseDB doctorsOrNurses;
        private Form parent;
        private string idDoc;
        private static bool flag;

        public frmDoctorOrNurse(string id)
        {
            InitializeComponent();
            this.idDoc = id;
            doctorsOrNurses = new DoctorOrNurseDB();
            state = AddState.NAVIGATE;
            SetButtonStates(true);
            doctorsOrNurses.Find(id);
            Populate(doctorsOrNurses.GetCurrentRow());
            flag = true;
        }
```

```
public frmDoctorOrNurse(Form f)
{
    this.parent = f;
    InitializeComponent();
    doctorsOrNurses = new DoctorOrNurseDB();
    state = AddState.NAVIGATE;
    Populate(doctorsOrNurses.GetCurrentRow());
    SetButtonStates(true);
}

private void Populate(DoctorOrNurse p)
{
    inputId.Text = p.Id;
    inputFirstName.Text = p.FirstName;
    inputLastName.Text = p.LastName;
    inputAddress.Text = p.Address;
    inputPhoneNumber.Text = p.PhoneNumber;
    inputGender.Text = p.MaleOrFemale;
    inputDateBirth.Value = p.DateOfBirth;
    inputDoctorOrNurse.Text = p.IsDoctorOrNurse;
}
private bool UpdateObject(DoctorOrNurse p)
{
    bool ok = true;
    try
    {
        p.Id = inputId.Text;
        errorProvider1.SetError(inputId, "");
    }
    catch (Exception ex)
    {
        ok = false;
        errorProvider1.SetError(inputId, ex.Message);
    }
```

```
try
{
    p.FirstName = inputFirstName.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.LastName = inputLastName.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.Address = inputAddress.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.PhoneNumber = inputPhoneNumber.Text;
    errorProvider1.SetError(inputId, "");
}
```

```
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.MaleOrFemale = inputGender.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.DateOfBirth = inputDateBirth.Value;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.IsDoctorOrNurse = inputDoctorOrNurse.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
```

```
        return ok;
    }
    private void SetButtonStates(bool b)
    {
        next.Enabled = b;
        prev.Enabled = b;
        Add.Enabled = b;
        update.Enabled = b;
        cancel.Enabled = !b;
        save.Enabled = !b;
        inputId.Enabled = !b;
        inputFirstName.Enabled = !b;
        inputLastName.Enabled = !b;
        inputAddress.Enabled = !b;
        inputPhoneNumber.Enabled = !b;
        inputGender.Enabled = !b;
        inputDateBirth.Enabled = !b;
        inputDoctorOrNurse.Enabled = !b;
    }
    private void frmPatient_Load(object sender, EventArgs e)
    {

    }

    private void inputId_TextChanged(object sender, EventArgs e)
    {

    }



    private void update_Click(object sender, EventArgs e)
    {
        SetButtonStates(false);
        state = AddState.UPDATE;
    }
```

```
private void Clear()
{
    foreach (Control c in Controls)
    {
        if (c is TextBox)
            (c as TextBox).Text = "";
        if (c is ComboBox)
            (c as ComboBox).Text = "";
        if (c is DateTimePicker)
            (c as DateTimePicker).Value = DateTime.Today;
    }
}
private void save_Click(object sender, EventArgs e)
{
    DoctorOrNurse dn = new DoctorOrNurse();
    if (UpdateObject(dn))
    {
        if (state == AddState.ADDNEW)
            doctorsOrNurses.AddRow(dn);
        else
            doctorsOrNurses.UpdateRow(dn);
    }
    SetButtonStates(true);
    state = AddState.NAVIGATE;
    doctorsOrNurses.Save();

}

private void Add_Click(object sender, EventArgs e)
{
    Clear();
    state = AddState.ADDNEW;
    SetButtonStates(false);
}
```

```
        private void cancel_Click(object sender, EventArgs e)
        {
            Populate(doctorsOrNurses.GetCurrentRow());
            errorProvider1.SetError(inputId, null);
            errorProvider1.SetError(inputFirstName, null);
            errorProvider1.SetError(inputLastName, null);
            errorProvider1.SetError(inputAddress, null);
            errorProvider1.SetError(inputPhoneNumber, null);
            errorProvider1.SetError(inputDateBirth, null);
            errorProvider1.SetError(inputGender, null);
            errorProvider1.SetError(inputDoctorOrNurse, null);
            SetButtonStates(true);
        }
        private void prev_Click(object sender, EventArgs e)
        {
            doctorsOrNurses.MovePrev();
            Populate(doctorsOrNurses.GetCurrentRow());
        }

        private void next_Click(object sender, EventArgs e)
        {
            doctorsOrNurses.MoveNext();
            Populate(doctorsOrNurses.GetCurrentRow());
        }

        private void frmDoctorOrNurse_FormClosing(object sender,
FormClosingEventArgs e)
        {
            flag = false;
            this.Hide();
            this.Dispose();
            parent.Show();
        }

        private void departmentBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
```

```
        {
            this.Validate();
            this.departmentBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.hMODataSet);

        }

        private void frmDoctorOrNurse_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'hMODataSet.DoctorOrNurse' table. You can move, or remove it, as needed.
            this.doctorOrNurseTableAdapter.Fill(this.hMODataSet.DoctorOrNurse);
            // TODO: This line of code loads data into the
'hMODataSet.Department' table. You can move, or remove it, as needed.
            this.departmentTableAdapter.Fill(this.hMODataSet.Department);

        }





        private void inputId_TextChanged_1(object sender, EventArgs e)
        {

this.specificMeetingDoctorTableAdapter.Fill(this.meetingForToday.SpecificMeetin
gDoctor, inputId.Text);
        }

        private void inputId_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.CheckIdNumber(inputId.Text))
            {
                e.Cancel = true;
                inputId.Focus();
                errorProvider1.SetError(inputId, "this id number is not valid
id");
            }
            else
```

```
            {
                e.Cancel = false;
                errorProvider1.SetError(inputId, null);
            }
        }

        private void inputFirstName_Validating(object sender, CancelEventArgs
e)
        {
            if (!Utility.ValidationUtilites.IsLegalName(inputFirstName.Text))
            {
                e.Cancel = true;
                inputFirstName.Focus();
                errorProvider1.SetError(inputFirstName, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputFirstName, null);
            }
        }

        private void inputLastName_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.IsLegalName(inputLastName.Text))
            {
                e.Cancel = true;
                inputLastName.Focus();
                errorProvider1.SetError(inputLastName, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputLastName, null);
```

```
            }
        }

        private void inputAddress_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.IsLegalAddress(inputAddress.Text))
            {
                e.Cancel = true;
                inputAddress.Focus();
                errorProvider1.SetError(inputAddress, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputAddress, null);
            }
        }

        private void inputPhoneNumber_Validating(object sender, CancelEventArgs
e)
        {
            if
(!Utility.ValidationUtilites.IsPhoneNumber(inputPhoneNumber.Text))
            {
                e.Cancel = true;
                inputPhoneNumber.Focus();
                errorProvider1.SetError(inputPhoneNumber, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputPhoneNumber, null);
            }
        }
```

```
        private void inputDateBirth_Validating(object sender, CancelEventArgs
e)
        {
            if (inputDateBirth.Value > DateTime.Now)
            {
                e.Cancel = true;
                inputDateBirth.Focus();
                errorProvider1.SetError(inputDateBirth, "this date isnt
happened yet");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputDateBirth, null);
            }
        }

        private void inputGender_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.IsLegalSex(inputGender.Text))
            {
                e.Cancel = true;
                inputGender.Focus();
                errorProvider1.SetError(inputGender, "this option does not
exist");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputGender, null);
            }
        }

        private void inputDoctorOrNurse_Validating(object sender,
CancelEventArgs e)
        {
```

```
            if
(!Utility.ValidationUtilites.IsLegalJob(inputDoctorOrNurse.Text))
            {
                e.Cancel = true;
                inputDoctorOrNurse.Focus();
                errorProvider1.SetError(inputDoctorOrNurse, "this option does
not exist");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputDoctorOrNurse, null);
            }
        }

        private void comboBox1_Validating(object sender, CancelEventArgs e)
        {
            //bool flag = false;
            //foreach(string s in comboBox1.Items.IndexOf(comboBox1.Text))
            //    if (comboBox1.Text.Equals(s))
            //    {
            //        flag = true;
            //        break;
            //    }
            //int a = comboBox1.Items.IndexOf(0);
            //if (comboBox1.Items.IndexOf(comboBox1.Text)!=-1)
            //{
            //    e.Cancel = true;
            //    comboBox1.Focus();
            //    errorProvider1.SetError(comboBox1, "this department does not
exist");
            //}
        }

        private void inputFirstName_TextChanged(object sender, EventArgs e)
        {
```

```
        }

        private void specificMeetingDoctorDataGridView_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            frmScheduleDoctor f = new frmScheduleDoctor(inputId.Text);
            f.Width = 1100;
            f.ShowDialog();
        }
    }
}
```

frmLogIn.cs

```
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;

namespace DrorCohen.Gui
{
    public partial class frmLogIn : Form
    {
        public frmLogIn()
        {
            InitializeComponent();
        }

        private void password_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            int x = (int)this.doctorOrNurseTableAdapter.isAdmin(id.Text,
"admin");
            //Frmmain f = new Frmmain();
            if (x>0)
            {
                frmOpen f = new frmOpen(this);
                f.Show();
                this.Hide();
            }
            else
            {
                MyMessage m = new MyMessage("your are not identify as manger\n
please try again later", 2);
                m.applyCustomChange();
                m.ShowDialog();
            }
        }

        private void id_TextChanged(object sender, EventArgs e)
        {

        }

        private void doctorOrNurseBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            this.doctorOrNurseBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.isAdmin1);

        }

        private void frmLogIn_Load(object sender, EventArgs e)
        {
```

```
            // TODO: This line of code loads data into the
'isAdmin1.DoctorOrNurse' table. You can move, or remove it, as needed.
            this.doctorOrNurseTableAdapter.Fill(this.isAdmin1.DoctorOrNurse);

        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }
    }
}
```

frmOpem.cs

```
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;

namespace DrorCohen.Gui
{
    public partial class frmOpen : Form
    {
        frmLogIn parent;
        public frmOpen(frmLogIn logIn)
        {
            parent = logIn;
            InitializeComponent();
            vaccine f = new vaccine () { Dock = DockStyle.Fill };
            this.pContainer.Controls.Clear();
            this.pContainer.Controls.Add(f);
            this.pContainer.BringToFront();
            f.Show();
        }

        private void imageLogo_Click(object sender, EventArgs e)
        {

        }

        private void Patient_Click(object sender, EventArgs e)
        {
            frmPatient f = new frmPatient(this) { Dock = DockStyle.Fill,
TopLevel = false, TopMost = true };
            this.pContainer.Controls.Clear();
            this.pContainer.Controls.Add(f);
            this.pContainer.BringToFront();
            f.Show();
            /*frmPatient f = new frmPatient(this);
            f.Show();
            this.Hide();*/
        }

        private void doctor_Click(object sender, EventArgs e)
        {
            frmDoctorOrNurse f = new frmDoctorOrNurse(this) { Dock =
DockStyle.Fill, TopLevel = false, TopMost = true };
            this.pContainer.Controls.Clear();
            this.pContainer.Controls.Add(f);
            this.pContainer.BringToFront();
            f.Show();
            /*frmDoctorOrNurse f = new frmDoctorOrNurse(this);
            f.Show();
            this.Hide();*/
        }

        private void department_Click(object sender, EventArgs e)
        {
```

```
            frmDepartment f = new frmDepartment(this) { Dock = DockStyle.Fill,
TopLevel = false, TopMost = true };
            this.pContainer.Controls.Clear();
            this.pContainer.Controls.Add(f);
            this.pContainer.BringToFront();
            f.Show();
            /*frmDepartment f = new frmDepartment(this);
            f.Show();
            this.Hide();*/
        }

        private void therapys_Click(object sender, EventArgs e)
        {

        }

        private void panelLogo_Paint(object sender, PaintEventArgs e)
        {

        }

        private void frmOpen_Load(object sender, EventArgs e)
        {

        }

        private void iconButton1_Click(object sender, EventArgs e)
        {
            MyMessage m = new MyMessage("please call to our call center: 1111",
(int)Utility.StatusMessage.INFO);
            //MessageBox.Show("please call to our call center: 1111");
            m.applyCustomChange();
            m.ShowDialog();
        }

        private void vaccine1_Load(object sender, EventArgs e)
        {

        }

        private void iconButton2_Click(object sender, EventArgs e)
        {

        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Dispose();
            parent.Dispose();
        }

        private void frmOpen_FormClosing(object sender, FormClosingEventArgs e)
        {
            parent.Dispose();
        }
    }
}
```

frmOrderTherapyFinal

```
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;

namespace DrorCohen.Gui
{
    public partial class frmOrderTherapyFinal : Form
    {
        private string idDoctor, d;
        private int idTherapy;
        private frmPatient parent;
        private string patientId;
        public frmOrderTherapyFinal(string patientId, frmPatient parent)
        {
            InitializeComponent();
            this.parent = parent;
            this.patientId = patientId;
        }

        private void doctorOrNurseBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            this.doctorOrNurseBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.theDoctorMeeting);

        }

        private void frmOrderTherapyFinal_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'meeting.SpecificMeetingDoctor' table. You can move, or remove it, as needed.

            // TODO: This line of code loads data into the
'meetingByDoctorId.MeetingDoctor' table. You can move, or remove it, as needed.
            // TODO: This line of code loads data into the
'theDoctorMeeting.DoctorOrNurse' table. You can move, or remove it, as needed.

this.doctorOrNurseTableAdapter.Fill(this.theDoctorMeeting.DoctorOrNurse);
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

this.meetingDoctorTableAdapter.FillBy(this.meetingByDoctorId.MeetingDoctor,
comboBox1.SelectedValue.ToString());
        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
```

```
        {

specificMeetingDoctorTableAdapter.Fill(meeting.SpecificMeetingDoctor);
            //
this.doctorOrNurseTableAdapter.Fill(this.theDoctorMeeting.DoctorOrNurse);
            string x = idTherapy.ToString();

            int serial = this.meeting.SpecificMeetingDoctor.Count + 1;

            string dd = dateTimePicker1.Value.DayOfWeek.ToString();
            switch (dd)
            {
                case "Sunday":
                    dd = "1";
                    break;
                case "Monday":
                    dd = "2";
                    break;
                case "Tuesday":
                    dd = "3";
                    break;
                case "Wednesday":
                    dd = "4";
                    break;
                case "Thursday":
                    dd = "5";
                    break;
                case "Friday":
                    dd = "6";
                    break;
                default:
                    dd = "7";
                    break;
            }
            if (dd == d)
            {
                DateTime d =
DateTime.Parse(dateTimePicker1.Value.ToShortDateString());



this.specificMeetingDoctorTableAdapter2.Fill(this.f.SpecificMeetingDoctor,
idTherapy.ToString(), new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
, typeof(System.DateTime))))));
                if
(Convert.ToInt32(f.SpecificMeetingDoctor.Rows[0][0].ToString()) == 0) {
                    meeting.SpecificMeetingDoctor.
                        AddSpecificMeetingDoctorRow
                        (serial,idTherapy.ToString(),d,patientId);
                    specificMeetingDoctorTableAdapter.
                        Update(meeting.SpecificMeetingDoctor);
                    MyMessage m = new MyMessage("your request to get therapy
has been accepted and aprroved :)\n see you at "+d.ToShortDateString(),1);
                    m.applyCustomChange();
                    this.Close();
                    parent.MyRefresh();
                    m.ShowDialog();
                }
                else
                {
```

```
                    MyMessage message = new MyMessage("your rerquest doesn't
aprooved from some reasons \n please try another date or another doctor", 2);
                    message.applyCustomChange();
                    message.ShowDialog();
                }
                //f.SpecificMeetingDoctor.Rows[0][0] = 5;
                //f.SpecificMeetingDoctor.AcceptChanges();
            }
            else
            {
                MyMessage message = new MyMessage("your rerquest doesn't
aprooved from some reasons \n please try another date or another doctor", 2);
                message.applyCustomChange();
                message.ShowDialog();
            }

            //)

            //meeting.SpecificMeetingDoctor.AddSpecificMeetingDoctorRow
            //  (serial, x, d, patientId);

            specificMeetingDoctorTableAdapter.Fill
                (meeting.SpecificMeetingDoctor);
            //meeting.SpecificMeetingDoctor.AddSpecificMeetingDoctorRow
            //    (serial, x, d, patientId);
            this.specificMeetingDoctorTableAdapter.
            Update(meeting.SpecificMeetingDoctor);
        }


        private void comboBox2_Click(object sender, EventArgs e)
        {

this.meetingDoctorTableAdapter.FillBy(this.meetingByDoctorId.MeetingDoctor,
comboBox1.SelectedValue.ToString());
        }

        private void fillToolStripButton_Click(object sender, EventArgs e)
        {
            //try
            //{
            //
this.specificMeetingDoctorTableAdapter2.Fill(this.f.SpecificMeetingDoctor,
therapyCodeToolStripTextBox.Text, new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
ateOfTherapyToolStripTextBox.Text, typeof(System.DateTime))))));
            //}
            //catch (System.Exception ex)
            //{
            //    System.Windows.Forms.MessageBox.Show(ex.Message);
            //}

        }

        private void meetingDoctorDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            int rowIndex = meetingDoctorDataGridView.CurrentCell.RowIndex;
            idTherapy =
Convert.ToInt32(meetingDoctorDataGridView.Rows[rowIndex].Cells[0].Value);
            button1.Visible = true;
```

```
            d =
(meetingDoctorDataGridView.Rows[rowIndex].Cells[3].Value).ToString();// לשנותן
באמת יום
            idDoctor =
(meetingDoctorDataGridView.Rows[rowIndex].Cells[2].Value).ToString();
        }
    }
}
```

frmPatient.cs

```
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 DrorCohen.DB;
using DrorCohen.Models;
using DrorCohen.Utility;

namespace DrorCohen.Gui
{
    public partial class frmPatient : Form
    {
        private Form parent;
        private AddState state;
        private PatientDB patients;
        private DataView dv;
        //public frmPatient()
        //{
        //    InitializeComponent();
        //    patients = new PatientDB();
        //    state = AddState.NAVIGATE;
        //    Populate(patients.GetCurrentRow());
        //    SetButtonStates(true);
        //}

        public frmPatient(Form frmmain)
```

```
        {
            this.parent = frmmain;
            InitializeComponent();
            patients = new PatientDB();
            state = AddState.NAVIGATE;
            Populate(patients.GetCurrentRow());
            SetButtonStates(true);
            dv = patients.GetDataView();
            listBox1.DataSource = dv;
            listBox1.ValueMember = "ID";
            listBox1.DisplayMember = "LastName";
        }
        public void MyRefresh()
        {
            idPatientToolStripTextBox.Text = inputId.Text;
            dateOfTherapyToolStripTextBox.Text =
DateTime.Today.ToShortDateString();

this.specificMeetingDoctorTableAdapter.Fill(this.futurePatient.SpecificMeetingD
octor, idPatientToolStripTextBox.Text, new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
ateOfTherapyToolStripTextBox.Text, typeof(System.DateTime))))));
        }
        private void Populate(Patient p)
        {
            inputId.Text = p.Id;
            inputFirstName.Text = p.FirstName;
            inputLastName.Text = p.LastName;
            inputAddress.Text = p.Address;
            inputPhoneNumber.Text = p.PhoneNumber;
            inputGender.Text = p.MaleOrFemale;
            inputDateBirth.Value = p.DateOfBirth;
        }
        private bool UpdateObject(Patient p)
        {
            bool ok = true;
            try
```

```
{
    p.Id = inputId.Text;
    errorProvider1.SetError(inputId, "");
}
catch(Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.FirstName = inputFirstName.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.LastName = inputLastName.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.Address = inputAddress.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
```

```
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.PhoneNumber = inputPhoneNumber.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.MaleOrFemale = inputGender.Text;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
try
{
    p.DateOfBirth = inputDateBirth.Value;
    errorProvider1.SetError(inputId, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(inputId, ex.Message);
}
return ok;
```

```
        }
        private void SetButtonStates(bool b)
        {
            next.Enabled = b;
            prev.Enabled = b;
            Add.Enabled = b;
            update.Enabled = b;
            cancel.Enabled = !b;
            save.Enabled = !b;
            inputId.Enabled = !b;
            inputFirstName.Enabled = !b;
            inputLastName.Enabled = !b;
            inputAddress.Enabled = !b;
            inputPhoneNumber.Enabled = !b;
            inputGender.Enabled = !b;
            inputDateBirth.Enabled = !b;
        }
        private void frmPatient_Load(object sender, EventArgs e)
        {

        }

        private void inputId_TextChanged(object sender, EventArgs e)
        {
            idPatientToolStripTextBox.Text = inputId.Text;
            dateOfTherapyToolStripTextBox.Text =
DateTime.Today.ToShortDateString();

this.specificMeetingDoctorTableAdapter.Fill(this.futurePatient.SpecificMeetingD
octor, idPatientToolStripTextBox.Text, new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
ateOfTherapyToolStripTextBox.Text, typeof(System.DateTime))))));
        }

        private void save_Click(object sender, EventArgs e)
        {
            Patient p = new Patient();
```

```
    if (UpdateObject(p))
    {
        if (state == AddState.ADDNEW)
            patients.AddRow(p);
        else
            patients.UpdateRow(p);
    }
    SetButtonStates(true);
    state = AddState.NAVIGATE;
    try
    {
        patients.Save();
    }
    catch
    {

    }
}

private void next_Click(object sender, EventArgs e)
{
    patients.MoveNext();
    Populate(patients.GetCurrentRow());
}

private void prev_Click(object sender, EventArgs e)
{
    patients.MovePrev();
    Populate(patients.GetCurrentRow());
}

private void update_Click(object sender, EventArgs e)
{
    SetButtonStates(false);
    state = AddState.UPDATE;
```

```
        }

        private void Add_Click(object sender, EventArgs e)
        {
            Clear();
            state = AddState.ADDNEW;
            SetButtonStates(false);
        }
        private void Clear()
        {
            foreach (Control c in Controls)
            {
                if (c is TextBox)
                    (c as TextBox).Text = "";
                if (c is ComboBox)
                    (c as ComboBox).Text = "";
                if (c is DateTimePicker)
                    (c as DateTimePicker).Value = DateTime.Today;
            }
        }

        private void cancel_Click(object sender, EventArgs e)
        {
            Populate(patients.GetCurrentRow());
            SetButtonStates(true);
            errorProvider1.SetError(inputId, null);
            errorProvider1.SetError(inputFirstName, null);
            errorProvider1.SetError(inputLastName, null);
            errorProvider1.SetError(inputAddress,null);
            errorProvider1.SetError(inputPhoneNumber, null);
            errorProvider1.SetError(inputDateBirth, null);
            errorProvider1.SetError(inputGender, null);
        }

        private void frmPatient_FormClosing(object sender, FormClosingEventArgs
e)
```

```
        {
            this.Hide();
            this.Dispose();
            parent.Show();
        }

        private void lastNameSearch_TextChanged(object sender, EventArgs e)
        {
            dv.RowFilter = "LastName Like '" + lastNameSearch.Text.Trim() +
"*'";
        }
        public void GetChoice(int choice)
        {
            if (patients.FindString(choice))
                Populate(patients.GetCurrentRow());
        }

        private void search_Click(object sender, EventArgs e)
        {
            GetChoice(Convert.ToInt32(listBox1.SelectedValue));
        }

        private void orderTherapy_Click(object sender, EventArgs e)
        {
            frmOrderTherapyFinal f = new frmOrderTherapyFinal(inputId.Text,
this);
            f.ShowDialog();
        }

        private void specificMeetingDoctorBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
        {
            this.Validate();
            this.specificMeetingDoctorBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.futurePatient);

        }
```

```
        private void fillToolStripButton_Click(object sender, EventArgs e)
        {
            try
            {
this.specificMeetingDoctorTableAdapter.Fill(this.futurePatient.SpecificMeetingD
octor, idPatientToolStripTextBox.Text, new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
ateOfTherapyToolStripTextBox.Text, typeof(System.DateTime))))));
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }

        private void specificMeetingDoctorDataGridView_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {

        }

        private void inputId_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.CheckIdNumber(inputId.Text))
            {
                e.Cancel = true;
                inputId.Focus();
                errorProvider1.SetError(inputId, "this id number is not valid
id");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputId, null);
            }
```

```
        }

        private void inputFirstName_Validating(object sender, CancelEventArgs
e)
        {
            if (!Utility.ValidationUtilites.IsLegalName(inputFirstName.Text))
            {
                e.Cancel = true;
                inputFirstName.Focus();
                errorProvider1.SetError(inputFirstName, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputFirstName, null);
            }
        }

        private void inputLastName_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.IsLegalName(inputLastName.Text))
            {
                e.Cancel = true;
                inputLastName.Focus();
                errorProvider1.SetError(inputLastName, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputLastName, null);
            }
        }

        private void inputAddress_Validating(object sender, CancelEventArgs e)
```

```
        {
            if (!Utility.ValidationUtilites.IsLegalAddress(inputAddress.Text))
            {
                e.Cancel = true;
                inputAddress.Focus();
                errorProvider1.SetError(inputAddress, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputAddress, null);
            }
        }

        private void inputPhoneNumber_Validating(object sender, CancelEventArgs
e)
        {
            if
(!Utility.ValidationUtilites.IsPhoneNumber(inputPhoneNumber.Text))
            {
                e.Cancel = true;
                inputPhoneNumber.Focus();
                errorProvider1.SetError(inputPhoneNumber, "you have entered non
valid charchters");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputPhoneNumber, null);
            }
        }

        private void inputDateBirth_Validating(object sender, CancelEventArgs
e)
        {
            if (inputDateBirth.Value>DateTime.Now)
```

```
            {
                e.Cancel = true;
                inputDateBirth.Focus();
                errorProvider1.SetError(inputDateBirth, "this date isnt
happened yet");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputDateBirth, null);
            }
        }

        private void inputGender_Validating(object sender, CancelEventArgs e)
        {
            if (!Utility.ValidationUtilites.IsLegalSex(inputGender.Text))
            {
                e.Cancel = true;
                inputGender.Focus();
                errorProvider1.SetError(inputGender, "this option does not
exist");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(inputGender, null);
            }
        }



        private void lastNameSearch_Click(object sender, EventArgs e)
        {
            lastNameSearch.SelectAll();
        }
```

```
        private void inputFirstName_TextChanged(object sender, EventArgs e)
        {

        }

        private void orderTherapy_Validated(object sender, EventArgs e)
        {
        }

        private void frmPatient_Activated(object sender, EventArgs e)
        {
            idPatientToolStripTextBox.Text = inputId.Text;
            dateOfTherapyToolStripTextBox.Text =
DateTime.Today.ToShortDateString();

this.specificMeetingDoctorTableAdapter.Fill(this.futurePatient.SpecificMeetingD
octor, idPatientToolStripTextBox.Text, new
System.Nullable<System.DateTime>(((System.DateTime)(System.Convert.ChangeType(d
ateOfTherapyToolStripTextBox.Text, typeof(System.DateTime))))));
        }
    }
}
```

frmSchduleDoctor.cs

```
using DrorCohen.Utility;
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 DrorCohen.DB;
using DrorCohen.Models;
using DrorCohen.DATA;

namespace DrorCohen.Gui
{
    public partial class frmScheduleDoctor : Form
    {
        private string doctorId;
        private AddState state;
        private ScheduleDoctorMeetingDB meetings;
        public frmScheduleDoctor(string doctorId)
        {
            this.doctorId = doctorId;
            InitializeComponent();
            meetings = new ScheduleDoctorMeetingDB();
            Populate(meetings.GetCurrentRow());
            state = AddState.NAVIGATE;
            SetButtonStates(true);
        }
        private void Populate(ScheduleDoctorMeeting meeting)
        {
            textBox1.Text = meeting.TherapyCode;
            comboBox2.Text = meeting.WhoCanGiveTheTherapy;
```

```
            textBox2.Text = meeting.IdDoctor;
            comboBox1.SelectedIndex = meeting.Day-1;
            textBox3.Text = meeting.Hour;
        }
        private void SetButtonStates(bool b)
        {
            next.Enabled = b;
            prev.Enabled = b;
            Add.Enabled = b;
            update.Enabled = b;
            cancel.Enabled = !b;
            save.Enabled = !b;
            textBox1.Enabled = !b;
            textBox2.Enabled = !b;
            textBox3.Enabled = !b;
            comboBox1.Enabled = !b;
            comboBox2.Enabled = !b;
        }
        private void meetingDoctorBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            /*this.meetingDoctorBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.hMODataSet1);*/
        }

        private void frmScheduleDoctor_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'scedule.MeetingDoctor' table. You can move, or remove it, as needed.
            // TODO: This line of code loads data into the
'hMODataSet1.MeetingDoctor' table. You can move, or remove it, as needed.

//this.meetingDoctorTableAdapter.Fill(this.hMODataSet1.MeetingDoctor);
            string firstTherapyCode = textBox1.Text;
            while (!doctorId.Equals(textBox2.Text))
            {
```

```
                meetings.MoveNext();
                Populate(meetings.GetCurrentRow());
                if(firstTherapyCode.Equals(textBox1.Text))
                {
                    Clear();
                    textBox1.Text = meetings.GetKey();
                    textBox2.Text = doctorId;
                    state = AddState.ADDNEW;
                    SetButtonStates(false);
                    cancel.Enabled = false;
                    label6.Visible = true;
                    return;
                }
            }
        }

        private void bindingNavigatorAddNewItem_Click(object sender, EventArgs
e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {

        }
        private bool UpdateObject(ScheduleDoctorMeeting s)
        {
            bool ok = true;
            try
            {
```

```
    s.TherapyCode= textBox1.Text;
    errorProvider1.SetError(textBox1, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(textBox1, ex.Message);
}
try
{
    s.IdDoctor= textBox2.Text;
    errorProvider1.SetError(textBox2, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(textBox2, ex.Message);
}
try
{
    s.WhoCanGiveTheTherapy= comboBox2.Text;
    errorProvider1.SetError(comboBox2, "");
}
catch (Exception ex)
{
    ok = false;
    errorProvider1.SetError(comboBox2, ex.Message);
}
try
{
    s.Day = comboBox1.SelectedIndex+1;
    errorProvider1.SetError(comboBox1, "");
}
catch (Exception ex)
{
```

```
                ok = false;
                errorProvider1.SetError(comboBox1, ex.Message);
            }
            try
            {
                s.Hour = textBox3.Text;
                errorProvider1.SetError(textBox3, "");
            }
            catch (Exception ex)
            {
                ok = false;
                errorProvider1.SetError(textBox3, ex.Message);
            }
            return ok;
        }
        private void save_Click(object sender, EventArgs e)
        {
            ScheduleDoctorMeeting s = new ScheduleDoctorMeeting();
            //s.Day = comboBox1.SelectedIndex + 1;
            if (UpdateObject(s))
            {
                if (state == AddState.ADDNEW)
                {
                    string SQLadd = "INSERT INTO MeetingDoctor ( TherapyCode,
WhoCanGiveTheTherapy, IdDoctor, [day], [hour] ) VALUES('" + s.TherapyCode+
"','" + s.WhoCanGiveTheTherapy+ "','" + s.IdDoctor+ "',[" + s.Day+ "],['" +
s.Hour+ "'])";
                    //string path = System.IO.Directory.GetCurrentDirectory();
                    //int x = path.IndexOf("\\bin");
                    //path = path.Substring(0, x) + "\\Data\\try.accdb";


                    //string myConnectionStr =
(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source='" + path + "';Persist
Security Info=True");
                    //con = new OleDbConnection(myConnectionStr);
                    //con.Open();
                    //OleDbCommand SQLCommand = new OleDbCommand();
```

```
                //SQLCommand.CommandText = SQLadd;
                //SQLCommand.Connection = con;
                int response1 = -1;
                //  response1 = SQLCommand.ExecuteNonQuery();
                response1 = DAL.GetInstance().ExecuteNonQuery(SQLadd);



            }
            //meetings.AddRow(s);

            else
                meetings.UpdateRow(s);
        }
        SetButtonStates(true);
        state = AddState.NAVIGATE;
//..          meetings.Save();
        label6.Visible = false;
    }

    private void next_Click(object sender, EventArgs e)
    {
        meetings.MoveNext();
        Populate(meetings.GetCurrentRow());
    }

    private void prev_Click(object sender, EventArgs e)
    {
        meetings.MovePrev();
        Populate(meetings.GetCurrentRow());
    }

    private void update_Click(object sender, EventArgs e)
    {
        SetButtonStates(false);
        state = AddState.UPDATE;
```

```
}

private void Add_Click(object sender, EventArgs e)
{
    Clear();
    textBox1.Text = meetings.GetKey();
    textBox2.Text = this.doctorId;
    state = AddState.ADDNEW;
    SetButtonStates(false);
}
private void Clear()
{
    foreach (Control c in Controls)
    {
        if (c is TextBox)
            (c as TextBox).Text = "";
        if (c is ComboBox)
            (c as ComboBox).Text = "";
        if (c is DateTimePicker)
            (c as DateTimePicker).Value = DateTime.Today;
    }
}

private void cancel_Click(object sender, EventArgs e)
{
    Populate(meetings.GetCurrentRow());
    SetButtonStates(true);
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void textBox1_Validating(object sender, CancelEventArgs e)
```

```
        {
            if (!ValidationUtilites.isNumber(textBox1.Text))
            {
                e.Cancel = true;
                textBox1.Focus();
                errorProvider1.SetError(textBox1, "not valid code");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(textBox1, null);
            }
        }

        private void comboBox2_Validating(object sender, CancelEventArgs e)
        {
            if (!(comboBox2.Text.Equals("Doctor") ||
comboBox2.Equals("Nurse")))
            {
                e.Cancel = true;
                comboBox2.Focus();
                errorProvider1.SetError(comboBox2, "not valid role");
            }
            else
            {
                e.Cancel = false;
                errorProvider1.SetError(comboBox2, null);
            }
        }

        private void textBox2_Validating(object sender, CancelEventArgs e)
        {
            if (!ValidationUtilites.CheckIdNumber(textBox2.Text))
            {
                e.Cancel = true;
```

```
        textBox2.Focus();
        errorProvider1.SetError(textBox2, "not valid id");
    }
    else
    {
        e.Cancel = false;
        errorProvider1.SetError(textBox2, null);
    }
}

private void comboBox1_Validating(object sender, CancelEventArgs e)
{
    if (!ValidationUtilites.isDay(comboBox1.Text))
    {
        e.Cancel = true;
        comboBox1.Focus();
        errorProvider1.SetError(comboBox1, "this day format is wrong");
    }
    else
    {
        e.Cancel = true;
        errorProvider1.SetError(comboBox1, null);
    }
}

private void textBox3_Validating(object sender, CancelEventArgs e)
{
    if (!ValidationUtilites.isHour(textBox3.Text))
    {
        e.Cancel = false;
        textBox3.Focus();
        errorProvider1.SetError(textBox3, "not valid hour!" +
            " please adhere to the format");
    }
    else
```

```
            {
                e.Cancel = true;
                errorProvider1.SetError(textBox3, null);
            }
        }
    }
}
```

MyMessage.cs

```
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 DrorCohen.Utility;

namespace DrorCohen.Gui
{
    public partial class MyMessage : Form
    {
        private string textMessage;
        private StatusMessage statusMessage;
        public MyMessage(string txt,int status)
        {
            this.textMessage = txt;
            this.statusMessage = (StatusMessage)status;
            InitializeComponent();
        }
        public void applyCustomChange()
        {
            label1.Text = textMessage;
            iconPictureBox1.IconFont = FontAwesome.Sharp.IconFont.Solid;
            iconPictureBox1.IconColor = Color.FromArgb(255, 255, 255);
            button1.Text = "Got it";
            button1.FlatStyle = FlatStyle.Flat;
            if (this.statusMessage == StatusMessage.APPROVED)
            {
                panel1.BackColor = Color.FromArgb(139, 210, 63);
                iconPictureBox1.IconChar =
FontAwesome.Sharp.IconChar.CheckCircle;
                button1.BackColor = Color.FromArgb(139, 210, 63);
            }
            if (this.statusMessage == StatusMessage.ERROR)
            {
                panel1.BackColor = Color.Crimson;
                iconPictureBox1.IconChar =
FontAwesome.Sharp.IconChar.TimesCircle;
                button1.BackColor = Color.Crimson;
            }
            if (this.statusMessage == StatusMessage.INFO)
            {
                panel1.BackColor = Color.FromArgb(53, 169, 228);
                iconPictureBox1.IconChar =
FontAwesome.Sharp.IconChar.InfoCircle;
                button1.BackColor = Color.FromArgb(53, 169, 228);
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void label1_Click(object sender, EventArgs e)
        {
```

```
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }
    }
}
```