בס"ד

Sched to surg'

חיה שבה אבוחצירא
322419276

תוכן

1. הצעת פרויקט תשפ"ב

סמל מוסד: 560342

שם מכללה: סמינר תורני בית יעקב.

שם הסטודנט: חיה שבה אבוחצירא.

ת.ז. הסטודנט: 322419276

שם הפרויקט: שיבוץ חדרי ניתוח.

תיאור הפרויקט:

המערכת תתכנן לוח זמנים לביצוע מקביל של מספר ניתוחים.
המערכת תתחשב בחשיבות ובנחיצות הניתוחים, בלוחות הזמנים של הרופאים הנמצאים בבית החולים, בזמני החיטוי בין ניתוח לניתוח, ותבנה לוח זמנים אידיאלי להספק הרב ביותר בזמן הקצר ביותר.
כדי להיכנס למערכת יש להזין תעודת זהות של אחות האחראית על חדרי ניתוחים, כאשר הכניסה התבצעה בהצלחה תינתן האפשרות ליצור לוח זמנים חדש, לצפות בלוח זמנים קיים או להכניס ניתוח חירום.
במקרה של יצירת לו"ז על האחות להכניס את פרטי הניתוח:
רופא, מחלקה, רמת עדיפות וסיכון והאם הוא זקוק למכשירים מיוחדים, כך לכל הניתוחים והמערכת תשבץ אותם בצורה האופטימלית הכי אפשרית.
במקרה של ניתוח חירום על האחות להכניס את פרטי הניתוח כמו לעיל במקרה כזה יתכנו שתי אפשרויות:
1.יש חדר פנוי - הניתוח ישתבץ בחדר הפנוי.
2. אין חדר פנוי - המערכת תעדכן את לו"ז מחדש ותשבץ את כולם בהתאם.
כמובן שתינתן גם אפשרות לצפות בלוח הזמנים הנוכחי של השבוע ולשנותו במידת הצורך.

הגדרת הבעיה האלגוריתמית:
שיבוץ לוח זמנים האידיאלי ביותר תוך התחשבות באילוצים השונים (כגון: זמני הניתוח, חדרי ניתוח עם מכשירים מיוחדים שאינם יכולים להיות בשימוש מקביל, ניתוחים בחשיבות עליונה שנכנסים מחדרי מיון וחדרי טיפול נמרץ וכדו').
כמובן שבעיית שיבוץ היא בעיה בלתי פתירה, והמטרה היא להגיע לתוצאה הטובה בקירוב.
כיוון שיש מספר חדרי ניתוח ומספר ניתוחים וכל ניתוח צריך חדר ניתוח כדי להתבצע, לכן אני אשתמש לפתרון הבעיה בשיבוץ הונגרי שתפקידו לשבץ את הדברים בצורה הנכונה ביותר כדי להגיע אל התוצאה הטובה ביותר המתואמת והמשובצת על הצד הכי טובה.
הרעיון בשיבוץ הונגרי המקורי שיש כמה אפשרויות בחירה ומכל אחד מהם בוחרים את האפשרות הטובה ביותר כדי להגיע לפתרון הזול ביותר גם כאן מתוך כמה אפשרויות תבחר האפשרות בעלת הסיכון הגבוה ביותר קודם כל ולאחר מכן יסונכרן בין רמת הסיכון והעדיפות כדי להציל כמה שיותר חיים ולהועיל לכמה שיותר אנשים.

רקע תאורטי בתחום הפרויקט:

כשמשבצים ניתוחים צריך להתחשב בכמות חדרי הניתוח שיש בבית החולים, בלוח הזמנים של הרופאים שנמצאים בבית החולים כרגע ויכולים לנתח, בחשיבות הניתוח וכן במקרים שחדרי ניתוח נלקחים לטובת טיפול נמרץ וחדרי מיון, חדרי הניתוח מלאים וצריך לשלוח לבית חולים אחר או שחדרי הניתוח ריקים ואפשר לשבץ ניתוחים של מטופלי חוץ ומטופלים מהמחלקות תוך כדי שימת לב לא ליצור מקרים שבהם ימתינו לניתוח זמן רב.

בכל בית חולים קיימים מספר רב של חדרי ניתוחים. לכל מחלקה כמה חדרים שבהם יש מכונות ומכשירים המותאמים למחלקה. בכל בית חולים יש צורך לבצע מספר גדול של ניתוחים ואין מספיק חדרים, לכן תמיד יש צורך לתעדף את הניתוחים לפי רמת סיכון ורמת עדיפות, כדי שלא יקרה מצב שבו יכנסו ניתוחים בעלי סיכון ושאר הניתוחים יוזנחו או שיכניסו סתם ניתוחים בלי להתחשב ברמת הסיכון ויתכנו סיכוני חיים. כדי למנוע מצב שניתוח שלא מסכן חיים לעולם לא יקבל את תורו, כל פעם שניתוח יתעכב בשבוע רמת העדיפות תעלה (כדי לא ליצור הרעבה).

כמובן כאשר נכנסים ניתוחי חירום או מקרי חירום אין זה משנה באיזה חדר הניתוח יתבצע (מבחינת מחלקה) והוא ישתבץ מידית תוך עדכון כל הלוח הזמנים בהתאם.

הפתרון לכך הוא תכנון ושילוב נכון של חדרי הניתוח תוך התחשבות בנתונים השונים.

תהליכים עיקריים בפרויקט:

1. המערכת תקבל את לוחות הזמנים של הרופאים, חדרי הניתוח, מטופלים וכו'.
2. המערכת תשבץ את סדר ביצוע שלבי הניתוח, החיטוי כך שיסתיימו בזמן הקצר האפשרי.
3. המערכת תתחשב באפשרות שיכנסו ניתוחים לא מתוכננים וידחו את כל הלוח זמנים המתוכנן ותעדכן אותו מחדש.

תיאור הטכנולוגיה:

צד שרת:

שפת תכנות בצד השרת: #C

צד לקוח:

שפת תכנות בצד לקוח: angular .

מסד נתונים: SQL Server .

פרוטוקולי תקשורת: אין.

לוחות זמנים:

1. חקר המצב הקיים – ספטמבר
2. הגדרת הדרישות – ספטמבר
3. אפיון המערכת – אוקטובר
4. אפיון בסיס הנתונים – נובמבר
5. עיצוב המערכת – דצמבר
6. בנית התוכנה – ינואר, פברואר
7. בדיקות – מרץ
8. הכנת תיק פרויקט – אפריל
9. הטמעת המערכת – מאי
10. הגשת פרויקט סופי - מאי

חתימת הסטודנט: חיה שבה אבוחצירא

חתימת הרכז המגמה:

אישור משרד החינוך:

2. מבוא / תקציר

2.1. הרקע לפרויקט

מהרגע שנכנסים למגמת הנדסת תוכנה ישנו דבר אחד שתמיד מרחף באוויר ומחכים לו פרויקט ההגשה, פרויקט שיתרום לנו כל כך הרבה התנסות ובעצם יעשה לנו הכנה לעבודה בחברות עם התמודדויות שלא נתקלים בהם במשך הלימודים, באגים שצריך לפתור לבד, חיפוש מידע שנצרך לפרויקט, חשיבה עצמאית של איך לפתח את הרעיון מבחינה שיהיה יעיל ויעזור לאנשים וגם שיהיה עם אלגוריתם טוב וחזק שיביא למימוש ומיצוי הלימודים שלנו.
המחשבה שקדמה לבחירת הרעיון שלי היא שילוב של שני תחומים שמעניינים אותי הנדסת תוכנה- המקצוע שלמדתי ורפואה-תחום שתמיד ענין אותי ואז עלה לי הרעיון על שיבוץ ניתוחים, במקום לשבץ ניתוחים באופן ידני ולבדוק שיש את כל המכשירים הנצרכים לניתוח ובמקרה חירום להזיז את כל הניתוחים, ולבדוק שכל השינויים שנעשו יתאימו לכולם ולא נשכח אף אחד דבר שמסובך יותר אם מדובר בכמה שינויים שנעשים בטווח קרוב, ולא נוצר מקרה שבו יהיו כפילויות וטעויות אדם, חשבתי על שיבוץ הניתוחים על ידי אלגוריתם כך שיבדוק את ההתאמה הטובה ביותר בין חדרי הניתוח לניתוח המתבצע, המכשירים הנצרכים שייתן עדיפות לניתוחי חירום ומצד שני לא ירעיב את שאר הממתינים , שהשיבוץ יעשה במהירות ולא יצטרכו להשקיע זמן ומחשבה בכל שינוי .
האלגוריתם שמתאים לי לפרויקט הוא צריך להיות אחד שבודק את כל האפשרויות שמתאימות ובוחר את הטובה מבין כולם- בדיקת התאמה של הניתוחים וחדרי הניתוח ציון שיינתן לכל אחד מהם ובחירת ההתאמה הטובה ביותר לכן האלגוריתם שבחרתי הוא שיבוץ הונגרי.
הפרויקט נקרא בשם "scudge and surge" משחק מילים על המילים שיבוץ וניתוח באנגלית.
- "scudge and surge"אפליקציה שתסייע לבתי חולים לשיבוץ ניתוחים בדרך הטובה ביותר ותסייע להצלת חיי אנשים בהמשך הספר אפשר יהיה לראות את השילוב הלוגי ביחד עם עיצוב נעים לעין, מידע נרחב על האלגוריתם ועל דרך הפיתוח.

2.2 תהליך המחקר

בתור התחלה שאלתי הרבה שאלות על הנושא כדי לפשט אותו כמה שיותר ושיהיה לי ברור על מ אני צריכה לעבוד

1. איך קובעים באיזה חדר מנתחים?
2. לפי מה קובעים איזה ניתוח מתבצע לפני?
3. מה קורה במקרי חירום?
4. כיצד כיום משבצים ניתוחים?

ועוד שאלות שעלו לי במהלך המחקר ואז התחלתי לחפש להם תשובות אם זה היה בחיפוש באתרים שונים או בהתייעצות עם אנשי מקצוע שונים אחיות, מזכירות וכדו'

לאחר שקיבלתי את התשובות לכל השאלות שהן:

1. קביעת חדר נעשית לרוב על ידי החדר הפנוי במקרים בסיסים אך במקרה שיש

ידעתי מה אני צריכה לעשות אבל היה חסר לי איך לפעול מלבד ידע בסיסי שמדובר בשיבוץ.

ערכתי חיפוש על אלגוריתמים של שיבוץ וקיבלתי כמה אפשרויות

שיבוץ גנטי- שיבוץ שנעשה על ידי שלוקחים שני פרטים מתאימים ביניהם ומהם יוצרים מוטציה ומתאימים שוב כך עד שמגיעים לתוצאה האפשרית המתאימה ביותר.

שיבוץ הונגרי- שיבוץ שנעשה על ידי כך שיש שני סוגי דברים שצריך להתאים ביניהם ומתאימים כל פריט עם כל פריט לדוגמא ישנם שלושה אחים: אפרים, מאיר וישראל. אחד מהם צריך לנקות את חדר השינה, שני לטאטא את החצר ושלישי לשטוף את החלונות. כל אחד מהם דורש תשלום שונה עבור כל משימה. המטרה היא למצוא את השיבוץ שייתן את העלות הנמוכה ביותר לביצוע שלוש המשימות

ואז הבנתי שהאלגוריתם המתאים לי ביותר הוא אלגוריתם הונגרי רק שבפרויקט הזה צריך ציון מקסימלי.

2.3 סקירת ספרות

מידע רב על הנושא והאלגוריתם מצאתי באתרים ונעזרתי בהם רבות

חומר לימודי-

המכלול

ויקיפדיה

אתרי קופ"ח ובתי חולים שונים

וכן באנשי מקצוע

אלגוריתם-ויקיפדיה

stackoverflwe

github

צד שרת ולקוח- angular io

Microsoft

stackoverflew

github

לעיצוב השתמשתי בbootstrep

3. מטרות ויעדים

מטרת בחירת הפרויקט נבעה מכמה שיקולים כמו התנסות מקצועית- בפרויקט שבביצוע שלו דומה לעבודה בשוק עם ההתמודדות עצמית של קשיים ובאגים.
מטרת ידע- להרחיב את האופקים על ידי בחירת הנושא שעוסק ברפואה נושא שמאד מעניין אותי אישית וחקירה עליו ואיך הוא פועל
וכמובן שהמטרה הייתה לפתח לא סתם עוד אפליקציה אלא כזו שתתרום ותעזור לאנשים ואולי אף תציל חיים.

מטרת העל היא אפליקציה שתעשה שיבוץ באופן מהיר קל ומזמין למשתמש בלי להתפשר על מקצועיות ובחירת התוצאות הטובות ביותר.
המערכת תקבל רשימת ניתוחים או אחד בכל פעם לפי הצורך ותשבץ אותם בהתאם באופן הטוב ביותר ותוציא למשתמש לוח זמנים מאורגן ונעים לעין .

4. אתגרים

תחילה מאד התפזרתי עם הרעיון של מה האפליקציה תעשה וכל הזמן זה לא היה לי עם קווים ברורים עד שישבתי עם עצמי לכמה שעות להיסגר בדיוק על מה הוא עושה,
ואז היה צורך לנתח את האפליקציה לרמה של טבלאות- כלומר איזה טבלאות יהיו בדאטה בייס שעליו היא תרוץ, ולא לשכוח לעבוד לפיהם.
לעשות מראה אפליקציה- איך אני רוצה שהאפליקציה שלי תראה באופן הסופי דברים אלו היו טיפה קשים בכללי כיוון שזהו סוג של חיזוי העתיד וקרו כמה וכמה מקרים שטבלאות מסוימות היו מיותרות או להפך היה צורך בלהוסיף טבלאות נוספות דבר שגורם שוב לחישוב מסלול מחדש.
בנוגע לאלגוריתם הייתי צריכה להתייעץ איזה מהם הוא הטוב לי ביותר עם מנחת הפרויקטים אבל המחשבה העיקרית שהושקעה זה איך לבצע בפועל את האלגוריתם זה בהחלט היה מאתגר איך לחשב ולהגיע לציון הסופי של כל אחד.
תחילה חשבתי שהפרויקט יהיה קל אך נתקלתי בקשיים ובבאגים שלקחו כמה שעות עד לפתירתם הסופית, בעיקר בבניית המטריצה עליה ירוץ האלגוריתם ומה הפרמטרים שלפיהם היא תבנה, הבנת האלגוריתם בו אני משתמשת עד למצב שהכל נאור וברור לי והתאמת האלגוריתם לפרויקט שלי.

5. מדדי הצלחה

האפליקציה שלי הצליחה אם אכן בסופו של דבר השיבוץ יעבוד- כלומר כל הניתוחים משובצים בחדרים עם ההתאמה המקסימאלית הכי אפשרית.
כמובן שיש כמהה הישגים נוספים שארצה להגיע אליהם אך הם פחות עיקריים לתלות בהם את הצלחת האפליקציה:
שיהיה גישה אך ורק למורשים,
שבסופו של דבר ייווצר לוח שנה מסודר עם הניתוחים המשובצים,
שבניתוח חירום יהיה אפשרות כניסה מידית לניתוח והתאמת כל השינויים שנעשו.

6. תיאור המצב הקיים

כיום לאחר בירורי שערכתי עם אנשים שעובדים בתחום נודע לי שיש מקומות שעובדים בשיבוץ ידני כלומר יש מישהו שמכניס את הניתוחים למערכת ומוודא שאכן אין כפילויות והכל מתאים,
בחיפוש אחר החומר מצאתי אפליקציה שככל הנראה עושה בדיוק את אותו הרעיון כמו שלי אך אי אפשר לקבל עליה שום מידע כיוון שהיא מוצעת לבתי חולים בתשלום

7. רקע תאורטי

האלגוריתם המרכזי בפרויקט הוא שיבוץ הונגרי וכיצד הוא בעצם עובד?
אלגוריתם ששיבוץ הונגרי עובד על מטריצה שבה יש שני מאפיינים וכל ריבוע במטריצה מהווה ניקוד מסוים של התאמה בין שני המאפיינים באופן שכל אחד מושווה עם כל אחד
כדי להבין זאת היטב הנה דוגמא פשוטה:

	ניתוח 1	ניתוח 2	ניתוח 3
חדר 1	2	3	3
חדר 2	3	2	3
חדר 3	3	3	2

כאשר נריץ את האלגוריתם ההונגרי על מטריצה זו הוא ייתן לנו את התוצאה המינימאלית הטובה ביותר או המקסימלית לפי הצורך
במקרה שאנו מחפשים את העלות המקסימאלית הטובה יותר במטריצה זו השלב הבא יהיה לחסר מכל שורה את הסכום המקסימאלי פחות מה שיש בעמודה כלומר:

	ניתוח 1	ניתוח 2	ניתוח 3
חדר 1	1	0	0
חדר 2	0	1	0
חדר 3	0	0	1

לאחר מכן מחסרים מכל עמודה את הסכום המקסימאלי במקרה לעיל אפשר למצוא כבר מהשלב השני שני פתרונות אופטימאליים,
אך במקרה שלא מוצאים:

- מסמנים את השורות שללא פתרון
- עמודות שנמצאו בהם פתרון שהוא גם בשורות-כלומר עמודה שיש בה 0
- שורות עם שיבוץ בעמודות מסומנות
- מסמנים בקו את השורות הלא מסומנות ואת העמודות המסומנות
- מוצאים מינימאלי/מקסימאלי שלא מכוסה ומפחיתים מכל הלא מכוסים ומוסיפים לצמתים ואז ישנו שיבוץ

בשתי האפשרות לעיל התוצאה המקסימאלית היא 9,
כך בעצם האלגוריתם ההונגרי עובד הוא רץ על מטריצת ציונים של התאמה בין ניתוח לחדר שנקבעת ממספר פרמטרים כמו רמת סיכון, עדיפות, מחלקה, מכשירים וכו' ומשבצת ניתוחים בחדרים על פי ההתאמה הטובה ביותר.

8. ניתוח חלופות מערכתי

ישנן אפשרויות שונות לפתרון הבעיה כמו:
שיבוץ ידני באופן שרירותי על ידי גורם מוסמך-לא מחייב התאמה בין חדר לניתוח וחוסר במכשירים מסוימים.
שיבוץ כל האפשרויות- הגיוני כאשר מדובר בכמה ניתוחים אך כשאר מדובר בעשרות אן זה הגיוני.
שיבוץ גנטי- שמוצא את אפשרות השיבוץ האפשריות הכי טובות ומהם מוצא עוד אפשרויות עד לקבלת האפשרות האופטימאלית הטובה ביותר.
המצאת אלגוריתם פשוט שיעבור בלולאה על כל החדרים הריקים והניתוחים שיש ופשוט ישבץ אחד לשני,
אני בחרתי באפשרות של מעבר בלולאה.

9. תיאור החלופה הנבחרת והנימוקים לבחירה

המערכת שפיתחתי מקבלת רשימת חדים ורשימת ניתוחים ומשבצת ניתוחים בחדרים על פי ההתאמה ביותר שנבדקת על ידיי מכשירים, מחלקה, ועוד תוך התחשבות במקרי חירום.

הרעיונות הקודמים שהוצעו לא רלוונטיים כיוון שהאפשרות של שיבוץ באופן שרירותי לא מתחשבת בהתאמה ולכן עדיף לעשות זאת כבר באופן ממוחשב כיוון שייקח הרבה פחות זמן

והאפשרות השנייה של בדיקת כל האופציות לא הגיונית כיון שאם מדובר בכמה חדרים וניתוחים הדבר מתאפשר אך כאשר מדובר בדאטה בייס של בית חולים שיש בו כמה עשרות ניתוחים אין זה הגיוני לעשות זאת באופן ידני.
אפשרות נוספת היא השיבוץ הגנטי אבל איננה מתאימה למערכת מסוג זה כיוון שלוקחת זמן ואינה יכולה לעסוק בניתוחים ובייחוד מקרי חירום שצריכים מענה בזמן קצר.
לכן האפשרות שבחרתי גם אינה טובה אך הכי מועילה מבין השאר מקבלים רשימת חדרים ריקים ורשימת ניתוחים עוברים ומאימים אחד לשני באופן שרירותי.

10. אפיון המערכת

סביבת פיתוח:

חומרה:

מעבד RAM 16.0 GB

Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60 GHz

עמדת פיתוח:

מחשב DALL

מערכת הפעלה:

windows 10

שפות תוכנה:

#C, תוך שימוש בטכנולוגית WebApi, Angular.

כלי תוכנה לפיתוח המערכת:

vs code, Microsoft Visual Studio 2019.

מסד נתונים: SQL server.

עמדת משתמש מינימאלית:

- **חומרה:**
  מעבד i5 4GB RAM
- **מערכת הפעלה:**
  Windows 7 ומעלה.
- **חיבור לרשת:**
  נדרש.
- **תוכנות:**
  Chrom.

10.1. ניתוח דרישות המערכת

דרישות בהן המערכת צריכה לעמוד:

- כתיבה בסטנדרטים מקצועיים .
- מחשוב השרות ללקוח .
- כתיבת הקוד בסיבוכיות היעילה ביותר .
- ממשק נוח וידידותי למשתמש .
- תגובה מהירה ככל שניתן למשתמש .

10.2. מודול המערכת

- הכנסת ניתוחים ופרטיהם על ידי צד לקוח
- חיבור לצד שרת והזנת הנתונים בדאטה בייס
- קריאה למערכת לערוך שיבוץ
- מציאת השיבוץ האופטימאלי על ידי המערכת
- הזנת פרטי השיבוץ בדאטה בייס
- הצגת פרטי השיבוץ ללקוח

10.3. אפיון פונקציונאלי

GetSurgeryFromSpecificDate()-

```
public static List<SurgeryDTO> GetSurgeryFromSpecificDate(DateTime specificDate)
{
    List<surgery> surgeriesFromTable = db.GetDbSet<surgery>().Where(S => S.surgeryDate == specificDate).ToList();
    List<SurgeryDTO> CreateSurgeryDtoList = SurgeryDTO.CreateSurgeryDtoList(surgeriesFromTable);
    return CreateSurgeryDtoList;

}
```

פונקציה זו מקבלת מפונקציה נוספת שנמצאת בצד לקוח שכתובה באנגולר:

shoeSurgeryFromSpecifictDate()-

```
shoeSurgeryFromSpecifictDate(today:Date){
  this.db.getSurgeryFromSpecificDate(today).subscribe(res =>{
    this.surgery=res;
  })
}
```

פונקציות אלו פועלות על ידי כך שמקבלים מהלקוח תאריך מסוים ששלפיו מוצגים כל הניתוחים שהוכנסו לדאטה בייס בתאריך זה ברשימה לפעמים הניתוחים מוצגים ללקוח ולפעמים נשלחים לפונקציה נוספת לפי הצורך.

בדומה לפונקציות למעלה נוצרות רשימות של חדרים, מכשירים הנצרכים לכל ניתוח ורשימת המכשירים הזמינים בבית החולים
כל הרשימות לעיל נשלחות לפונקציה:

FillMatrix()-

```
class HungrienScudling
{
    public double[,] FillMatrix(List<SurgeryDTO> listOfSurgery, List<RoomDTO> listOfRoom, List<DeviceForSurgeryDTO> D, List<SpecialDeviceDTO> S)
    {
        PreHungrien preMat = new PreHungrien();
        double[,] gradeMat = preMat.CalculateScore(listOfSurgery, listOfRoom, D, S);
```

פונקציה זזו נמצאת ב **HungrienScudling** class שבו מתבצע השיבוץ בפועל אבל לפני הפונקציה בונה מטריצת התאמות כפי שהסברנו בסעיף 7 שעליה תרוץ ובעצם שולחת לפונקציה:

CalculateScore()-

```
public double [,] CalculateScore(List<SurgeryDTO> listOfSurgery, List<RoomDTO> listOfRoom, List<DeviceForSurgeryDTO> D, List<SpecialDeviceDTO>S )
{
  surgeryMatrix=new double [listOfSurgery.Count, listOfRoom.Count];
    IDictionary<double, SurgeryDTO> surgeryWithPriority =CalculatePriority(listOfSurgery);
    foreach (var item in surgeryWithPriority)
    {
        int i = 0;
        for(int j=0; j < listOfRoom.Count();j++)
        {
            double squereGrade=Grade(item, listOfRoom[j], D, S);
            surgeryMatrix[i,j] = squereGrade;
            i++;

        }
    }

    return surgeryMatrix;
}
```

פונקציה זו עוברת על רשימת הניתוחים ונותנת התאמה של כל אחד מהניתוחים לכל אחד מהחדרים
וכיצד היא נותנת ציון עפ"י הפונקציה:

Grade()-

```
public double Grade(KeyValuePair<double, SurgeryDTO >surgery, RoomDTO Room, List<DeviceForSurgeryDTO> D, List<SpecialDeviceDTO> S)
{
    double grade;
    grade = surgery.Key + MatchRoom(surgery.Value, Room)+MatchDevice(D,S,surgery.Value);
    return int.MaxValue -  grade;
}
```

פונקציה זו מחשבת התאמה על פי רמת עדיפות וסיכון שנקבעה על ידי רופא, חדר תואם ומכשירים זמינים הנצרכים לניתוח כל אחד מהפונקציות האלו מחזירה ציון שמשתכלל לציון סופי ונכנס לריבוע המתאים של החדר והניתוח, לאחר שיש מטריצה מלאה היא חוזרת בחזרה לפונקציה הראשונה שקראה לה FillMatrix() ואז האלגוריתם מתחיל לרוץ על המטריצה ולעשות שיבוץ בפועל.

10.4. ביצועים עיקריים

אחות בכירה רשאית להוסיף את שאר המשתמשים למערכת
אחות בכירה רשאית להוסיף ניתוחים למערכת
אחות רגילה יכולה לצפות בלו"ז השיבוץ בלבד
המשתמש יכול לקבל רשימת חדרים, ניתוחים, מכשירים עפ"י סינון
למשתמש מותג לוח שיבוץ עפ"י תאריכים מסודר

10.5. אילוצים

המערכת פועלת על פי נתונים ואינה יכולה לערוך את השיבוץ בלעדיהם.
יהיה מספר מוגבל של ניתוחים ביום בחדר לפי יכולת בית החולים.

11. תיאור הארכיטקטורה

11.1. הארכיטקטורה של הפיתרון המוצע בפורמט של Design level Down-Top

צד השרת - server side פותח במודל 3 השכבות ומתחלק ל-4 פרויקטים החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין הנתונים עצמם.
הפרדה זו מאפשרת לבצע שינויים בכל אחת מהשכבות בלי תלות ובלי זעזועים בשכבות האחרות.

– API שכבת ה – Controller חיבור בין צד השרת והלקוח.

– BL הלוגיקה של המערכת.

– DAL מכיל את הפונקציונאליות הנדרשת לכל התקשורת עם ה. Data Base

– Models מכילה מחלקות המתארות את הנתונים ובמבנה זה מעבירים את הנתונים בין השכבות.

מטרת שכבה זו היא למנוע תלות של שכבת ה BL במבנה בסיס הנתונים. שכבת ה BL מכילה פונקציות המרה מטיפוס הנתונים של בסיס הנתונים לטיפוס הנתונים של שכבת ה Models ולהיפך, וכך מיוצגים הנתונים בכל הפרויקט.

11.2. תיאור הרכיבים בפתרון

הפרויקט מחולק ל-2 חלקים :

- צד שרת - הנכתב בשפת C# ובטכנולוגיית .WebApi
- צד לקוח - נכתב בשפת Angular ובטכנולוגיית .Html, TypeScript

בחרתי לכתוב צד לקוח ב- אנגולר שהינה שפה מתקדמת ועדכנית בעלת מאפייני Angular8 חדשניים ופונקציונאלית ביותר.
אנגולר הינה סביבת עבודה שפותחה על ידי גוגל. מאפשרת לפתח אפליקציות Framework אינטרנט בקלות ומהירות. במקור היא באה לתת מענה לבניית Applications Page Single בצורה מושלמת ומהירה.
מהיתרונות הבולטים והעיקריים של אנגולר אפשר למנות: חיסכון במשאבים, מהירות ביצוע, קוד קצר יותר, רוב העבודה מתבצעת בצד הלקוח ופחות בשרת ויכולת התמודדות טובה) סינון מהיר ופשוט לביצוע (של תוכן המתקבל מהשרת לפי מספר רב של פרמטרים.
צד שרת בחרתי לכתוב ב .#C
C #היא שפת תכנות עילית מרובת- פרדיגמות, מונחית עצמים בעיקרה המשלבת רעיונו ת כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיות, פונקציונאליות פרוצדוראליות וגנריות.

#Cהיא שפה מעניינת, נוחה ומלאה פונקציונאליות למתכנת. שימוש בשפה זו נפוץ כיום, וכתוצאה מכך, ניתן היה למצוא בה קודים שונים שנדרשו לפיתוח.
בנוסף, בחרתי להשתמש ב-EntityFrameworkטכנולוגית עבודה מתקדמת של מיקרוסופט .

ה- EntityFramework מאפשר לטעון את הנתונים מה- DB ולעשות להם השמה בצורה ישירה ואוטומטית לתוך אובייקטים בקוד הממפים את מאגר הנתונים בצורה מידית.
ה- EntityFramework קורא נתונים מה- DataBase שנכתב בשפת ה- .Sql Server

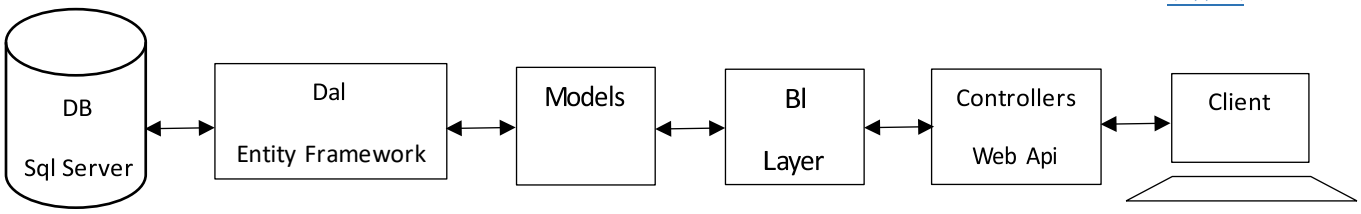
למסד הנתונים של ה- Sql Server
יש כלים נרחבים לגיבוי כל המידע המערכת, כולל מערכת ההפעלה, חשבונות המשתמשים והרשאותיהם, הגדרות ההתקנים, תוכניות וכן של שאר הרכיבים המסופקים עם השרת ואובייקטי המשתמש .

דוגמא לזרימת מידע במערכת

שליפת כל הניתוחים מתאריך מסוים-
ברצוננו לקבל את כל הניתוחים בתאריך מסוים מ ה DBולכן יתבצעו השלבים הנ"ל :

- **המשתמש יחפוץ לראות את כל הניתוחים של תאריך מסוים הוא ילחץ על כפתור מסוים בתצוגה,(html) ישלח את התאריך שהוא חפץ ובקשתו תפנה ל .TypeScript**
- **תתבצע קריאה לפונקציה ()shoeSurgeryFromSpecifictDate ב - Type scriptאשר תפנה לשרת url ותתבצע בקשת services**
- **השרת מקבל את הבקשה ומנווט ל Controller שנמצא ב .API**
- **ה Controller-יזמן את הפונקציה ()GetSurgeryFromCurrentDay שנמצאת ב -SurgeryManager ב BL-הוא מעונין לקבל נתונים מה DB-ולכן הוא פונה ל - DAL דרך ה - Entity framework ה DAL-שואב את הנתונים הרצויים ממסד הנתונים וכעת מתבצע שלב החזרה .**
- **ה DAL מחזיר את רשימת הניתוחים לשכבת ה BLבה מתבצעת פונקצית הסינון של הבאת ניתוחים מתאריך מסוים .**
- **הפונקציה GetSurgeryFromSpecificDate() מחזירה את הנתונים ל BL מה-.controller**
- **הנתונים מוחזרים ל service מה -.controller**
- **מה service-חוזרת הרשימה ל typeScript-הרשימה מוצגת ב .HTML**

איור :



1. **מסד הנתונים הבנוי מטבלאות וקשרי גומלין ביניהם .**
2. **שכבת הגישה לנתונים באמצעות.Entity Framework**

3. שכבת הישויות .
4. שכבת ה - BL בה כתובים האלגוריתמים.
5. Web Apiפרוטוקול התקשורת בין צד הלקוח וצד השרת.
6. צד לקוח.angular, TypeScript-

11.3. ארכיטקטורת רשת (לא רלוונטי)

11.4. <u>תיאור פרוטוקולי התקשורת</u>

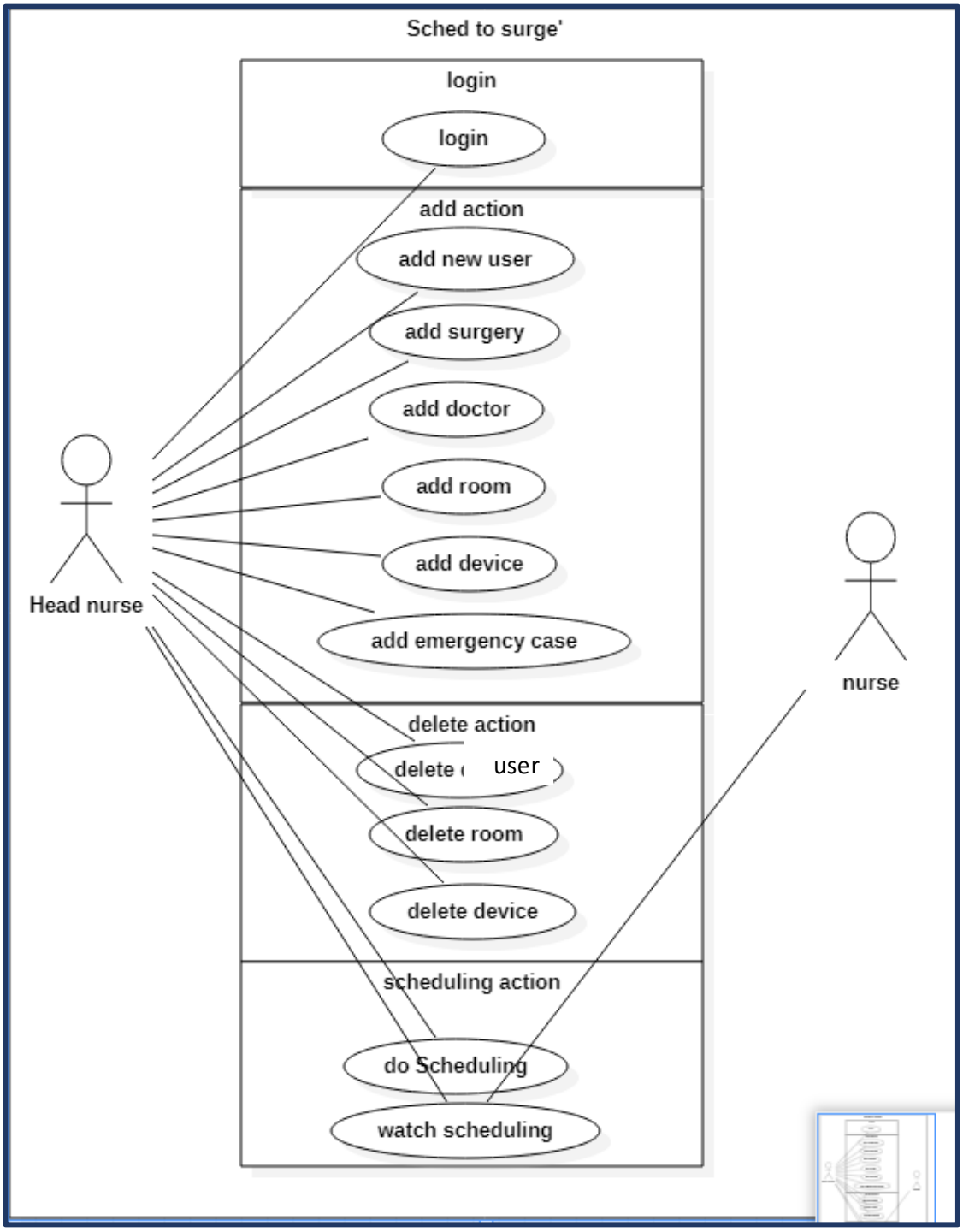
- http הוא פרוטוקול תקשורת שנועד להעברת דפי HTML ואובייקטים שהם מכילים (כמו תמונות, קובצי קול, סרטוני פלאש וכו') ברשת אינטרנט וברשתות אינטארנט.

11.5. <u>שרת – לקוח</u>

צד השרת נכתב בטכנולוגית WebApi ובשפת .c #
צד הלקוח נכתב בטכנולוגית angular
בשפות . Html, css ,typescript-

11.6. תיאור הצפנות (לא רלוונטי)

12.ניתוח ותרשים use case של המערכת המוצעת



12.1. רשימת use case

המשתמש מתחבר על ידי הזנת שם וסיסמא
במקרה שזוהי אחות רגילה:
אחות רגילה רשאית לצפות בשיבוץ

במקרה שזוהי אחות ראשית:
מוסיפה משתמש חדש על ידי הכנסת פרטיו תעודת זהות, שם, סיסמא, מה התפקיד....

מוסיפה ניתוח חדש ומכניסה תאריך, מחלקה, מכשירים הנצרכים לניתוח, רמת הסיכון והעדיפות שנקבעו על ידי הרופא, מי הרופא שקבע.

הוספת חדר איזה מחלקה ומה המכשירים הנמצאים בו

הוספת מכשיר שם, מחלקה, כמות, נייד

הוספת מקרה חירום

מחיקת משתמש על ידי הכנסת שם

מחיקת חדר הכנסת ID של חדר ומחיקתו

מחיקת מכשיר הכנסת ID של מכשיר ומחיקתו

הפעלת השיבוץ על ידי שליחת רשימת חדרים, רשימת ניתוחים, רשימת מכשירים, לפעולת השיבוץ.

12.2. תיאור ה use case-העיקריים של המערכת

UC1

- Uc1 :Identifie
- Login :Name
- Description: המשתמש מכניס שם משתמש וסיסמא על מנת להתחבר למערכת.
- Actors: משתמש שרשום וחפץ להיכנס למערכת.
- Frequency: בכניסה לאתר.
- pre-condition: רשום באתר.
- post-condition: נתוני המשתמש נקלטים במערכת ונבדקים האם הם קיימים במערכת.
- Extended use cases: בהרשמה לאתר.
- Assumptions: הנתונים שהוזנו נכונים ותקינים.
- basic action of cours: המערכת קלטה עובד מסוים של בית החולים והיא מאפשרת לו את הפעולות המורשות לו.
- Alternate action of course: אם המשתמש אינו קיים במערכת תוצג שגיאה
- history Change: גרסה ראשונה, חיה אבוחצירא.
- Issues: הכנסת נתונים של בית החולים החדש.
- הכנסת נתוני משתמשים/עובדים.
- Decisions: אנו מסתמכים על נכונות הפרטים

UC2

- Identifie: uc2
- Name: Add action
- Description: המשתמש מוסיף למערכת מידע מסוג- ניתוחים, משתמש חדש, מחלקות, חדרים, רופאים, מכשירים ומקרי חירום.
- Actors: אחות ראשית הרשומה במערכת.
- Frequency: בכל פעם שמעונין בהוספת מידע לאתר.
- pre-condition: רשום באתר.
- post-condition: נתוני המידע נקלטים במערכת ומוספים ל -DataBase
- Extended use cases:
- Assumptions: הנתונים שהוזנו נכונים ותקינים.
- basic action of cours: המערכת קלטה מידע מעובד מורשה של בית החולים.
- Alternate action of course: אם המידע לא נוסף תוצג שגיאה.
- history Change: גרסה ראשונה, חיה אבוחצירא.
- Issues: הכנסת המידע על ידי המשתמש.
- Decisions: אנו מסתמכים על נכונות הפרטים

UC3

- Identifi: Uc3
- Name: Delete action
- Description: המשתמש מסיר מהמערכת מידע מסוג- ניתוחים, משתמש חדש, מחלקות, חדרים, רופאים, מכשירים ומקרי חירום.
- Actors: אחות ראשית הרשומה במערכת.
- Frequency: בכל פעם שמעונין בהסרת מידע מהמערכת.
- pre-condition: רשום באתר.
- post-condition: נתוני המידע נקלטים במערכת ומוסרים מ-DataBase
- Extended use cases:
- Assumptions: הנתונים שהוזנו קיימים במערכת.
- basic action of cours: המערכת קלטה מידע מעובד מורשה של בית החולים.
- Alternate action of course: אם המידע לא הוסר תוצג שגיאה.
- history Change:.
- Issues: הסרת מידע על ידי המשתמש.
- Decisions: אנו מסתמכים על נכונות הפרטים

UC4

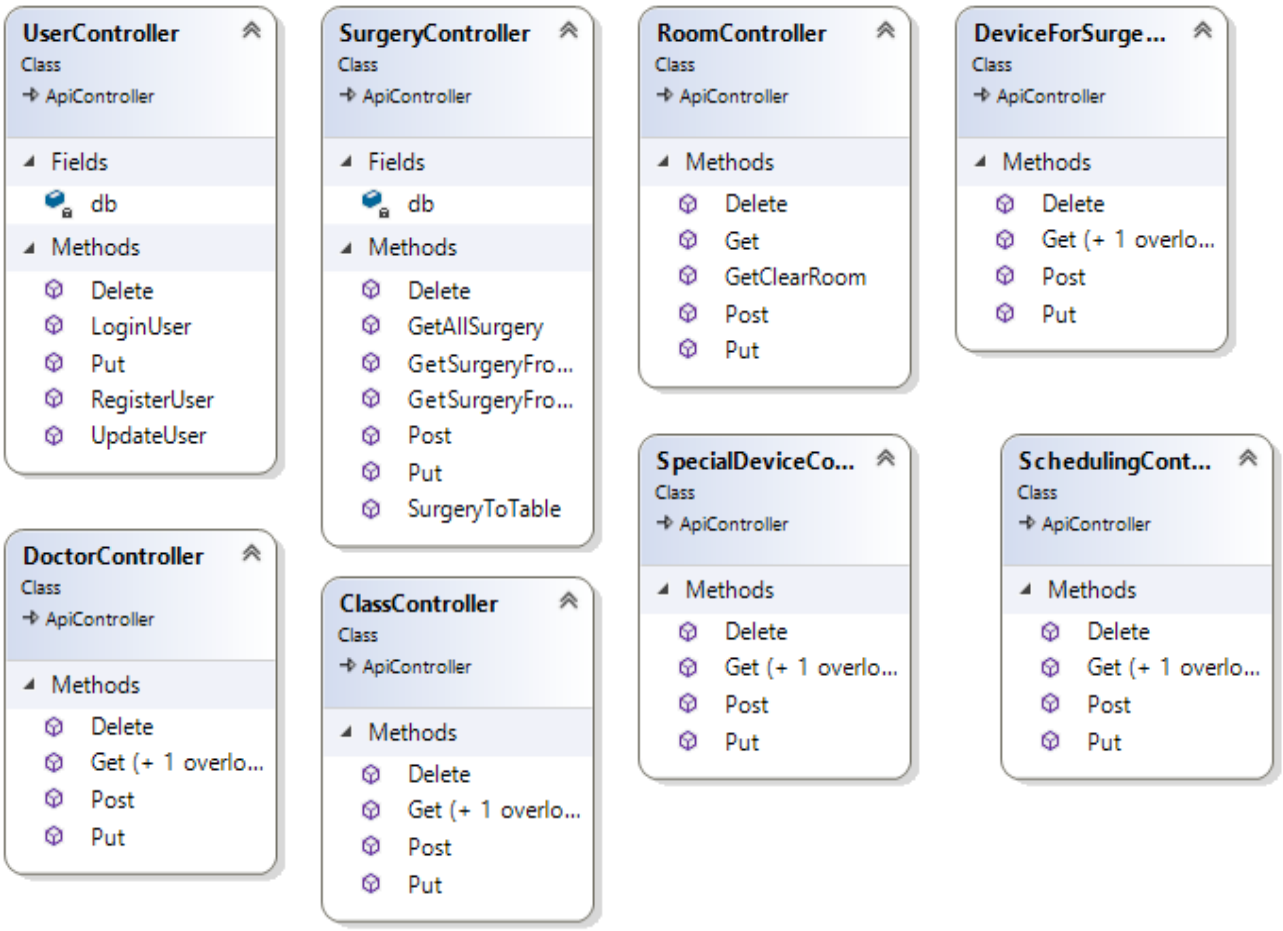
- Uc4 :Identifi
- Scheduling action :Name
- Description: המשתמש צופה בתוצאת השיבוץ שנעשה או מפעיל את פעולת השיבוץ.
- Actors: אחות או אחות ראשית הרשומה במערכת.
- Frequency: בכל פעם שמעונין בצפייה בתוצאות השיבוץ או בהכנסת נתונים חדשים אם חפץ.
- condition-pre: קיימים נתונים בבמערכת.
- condition-post: המשתמש יכול לראות את השיבוץ
- Extended use cases:
- Assumptions: תוצאות השיבוץ הן ההתאמה הטובה ביותר.
- basic action of cours: המשתמש יצפה בשיבוץ.
- Alternate action of course: אם נוספו נתונים חדשים למערכת השיבוץ יופעל שוב.
- history Change:גרסה ראשונה, חיה אבוחצירא.
- Issues: *הפעלת שיבוץ
  *בחירת אופן התצוגה של השיבוץ
- Decisions: אנו מסתמכים על נכונות הפרטים

12.3. מבני נתונים בהם משתמשים בפרויקט

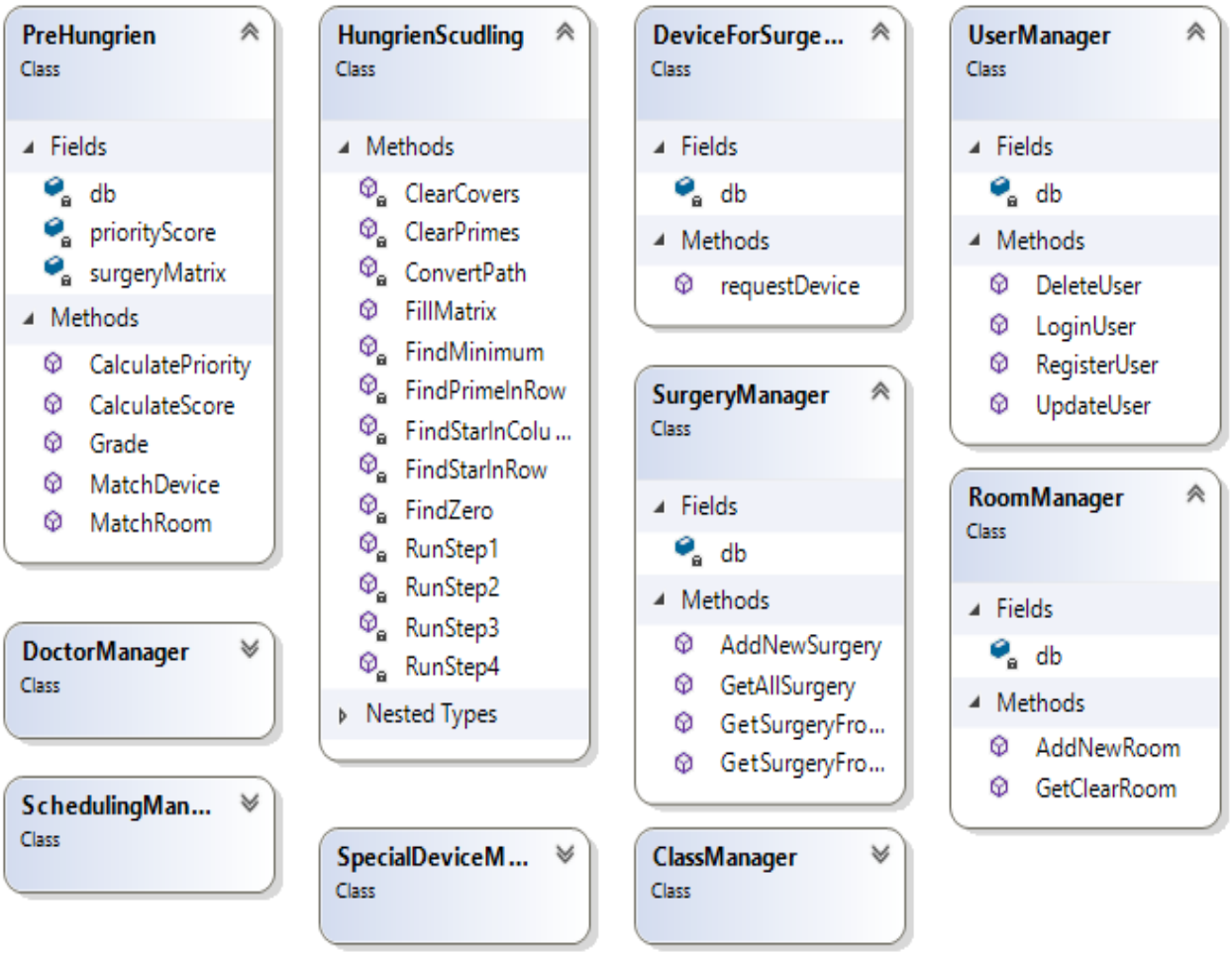
מילון- היה לי צורך במבנה נתונים שישמור לכל ניתוח את קוד הניתוח ואת רמת העדיפות שחישבו לו בתור הכנה למטריצת השיבוץ וזהו המבנה המתאים ביותר רשימה- יש צורך בקבלת מספר אוביקטים ,המרת מספר אוביקטים ושליחתם

12.4. תרשים מחלקות

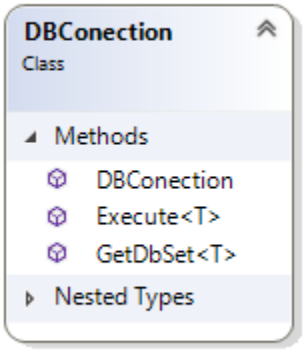
שכבת ה-API



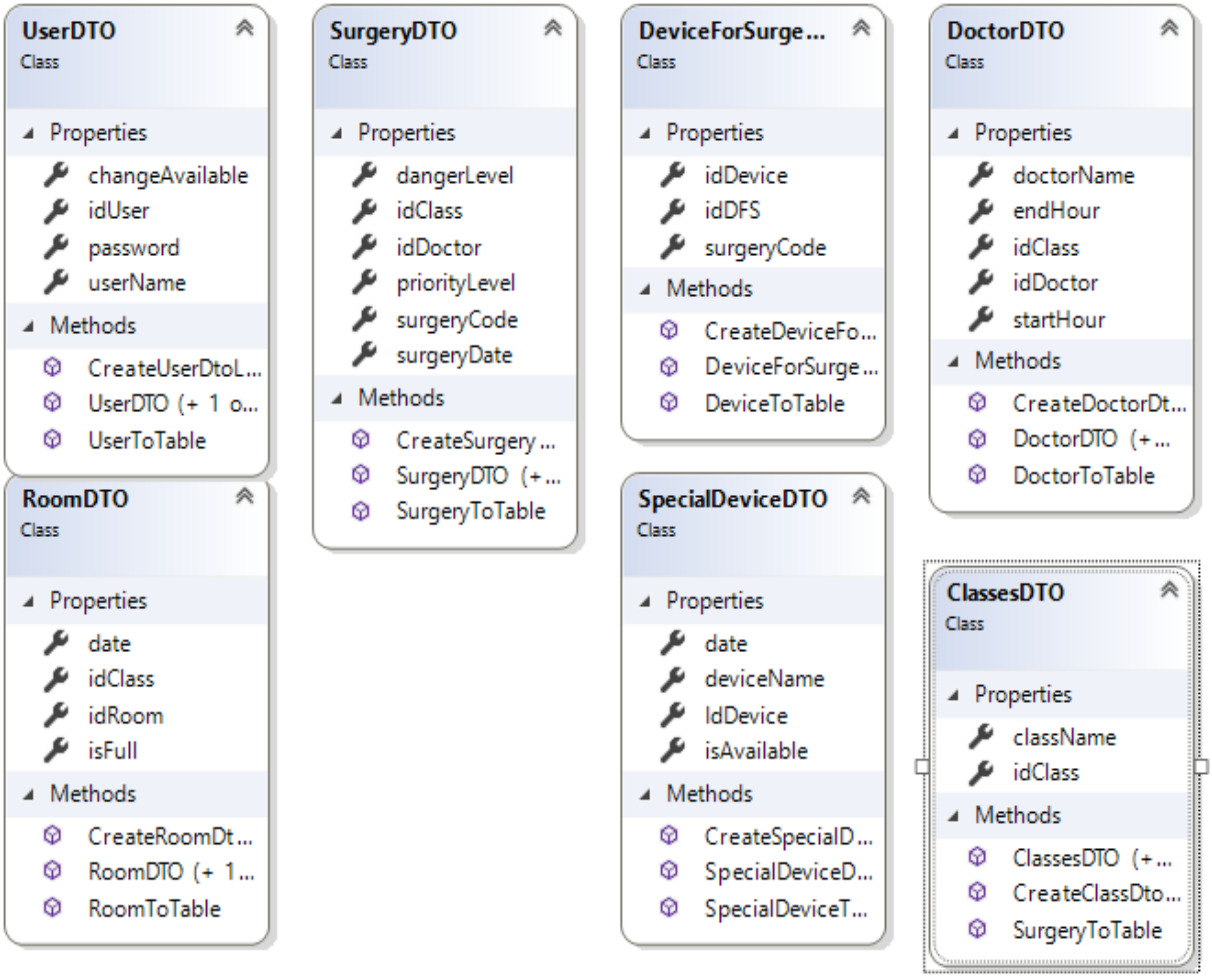
שכבת ה-BL



שכבת ה-DAL



שכבת ה-DTO


UserDTO
Class
Properties
changeAvailable
idUser
password
userName
Methods
CreateUserDtoL...
UserDTO (+ 1 o...
UserToTable
SurgeryDTO
Class
Properties
dangerLevel
idClass
idDoctor
priorityLevel
surgeryCode
surgeryDate
Methods
CreateSurgery...
SurgeryDTO (+...
SurgeryToTable
DeviceForSurge...
Class
Properties
idDevice
idDFS
surgeryCode
Methods
CreateDeviceFo...
DeviceForSurge...
DeviceToTable
DoctorDTO
Class
Properties
doctorName
endHour
idClass
idDoctor
startHour
Methods
CreateDoctorDt...
DoctorDTO (+...
DoctorToTable
RoomDTO
Class
Properties
date
idClass
idRoom
isFull
Methods
CreateRoomDt...
RoomDTO (+ 1...
RoomToTable
SpecialDeviceDTO
Class
Properties
date
deviceName
IdDevice
isAvailable
Methods
CreateSpecialD...
SpecialDeviceD...
SpecialDeviceT...
ClassesDTO
Class
Properties
className
idClass
Methods
ClassesDTO (+...
CreateClassDto...
SurgeryToTable

12.5. תיאור המחלקות

-API

שכבה זו מתקשרת עם צד הלקוח, באמצעות קונטרולרים – בקרים, המקבלים פניות מהלקוח ומאחזרים מידע בהתאם.

בשיטה זו קיימת הפרדת ישויות מוחלטת וכל שכבה עומדת באופן עצמאי לחלוטין ומתקשרת רק עם השכבה שמעליה.

בשכבה זו נמצאים הקונטרולים:

-RoomController

קונטרולר שמקבל פניות מהלקוח בנוגע לחדרים ומפעיל את הפונקציות הבאות:

-GetClearRoom()

פונקציה השולפת את כל החדרים הריקים במאגר הנתונים.

-GetAllRoom()

פונקציה השולפת את כל החדרים הנמצאים במערכת.

-SurgeryController

קונטרולר שמקבל פניות מהלקוח בנוגע לניתוחים ומפעיל את הפונקציות הבאות:

-GetSurgeryFromCurrentDate()

פונקציה המפעילה ב-SurgeryManager פונקציה השולפת את כל הניתוחים מהתאריך הנוכחי.

-GetSurgeryFromSpecificDate()

פונקציה המפעילה ב-SurgeryManager פונקציה השולפת ניתוחים מתאריך מסוים שהתקבל מהמשתמש.

-GetAllSurgery()

פונקציה המפעילה ב-SurgeryManager פונקציה השולפת את כל הניתוחים הנמצאים במערכת.

-AddSurgery()

```
[Route("api/Surgery/SurgeryToTable")]
[HttpPost]
public string AddSurgery(SurgeryDTO newSurgery)
{
    surgery s = newSurgery.SurgeryToTable();
    db.surgery.Add(s);
    db.SaveChanges();
    return "succes";
}
```

פונקציה המוסיפה ניתוח חדש למערכת.

- UserController

קונטרולר שמקבל פניות מהלקוח בנוגע למשתמשים ומפעיל את הפונקציות הבאות:

-LoginUser()

פונקציה המפעילה ב- UserManager פונקציה המקבלת את פרטי המשתמש ובודקת האם משתמש זה קיים במערכת ופרטי ההזדהות נכונים.

-RegisterUser()

פונקציה המפעילה ב- UserManager פונקציה המקבלת את פרטי המשתמש ומוסיפה את המשתמש החדש למערכת.

-UpdateUser()

פונקציה המפעילה ב- UserManager פונקציה המקבלת שינויים מהמשתמש ומעדכנת אותם במערכת.

-ClassController

קונטרולר שמקבל פניות מהלקוח בנוגע למחלקות כמו הוספה ומחיקה.

-DeviceForSurgeryController

קונטרולר שמקבל פניות מהלקוח בנוגע לבקשות מכשירים לניתוח ומפעיל את הפונקציות הבאות:

-GetAllRequest()

פונקציה המפעילה ב- DeviceForSurgeryManager פונקציה השולפת את כל הבקשות למשירים הקיימות במערכת.

-AddRequest()

```
[Route("api/DeviceForSurgery/AddRequest")]
[HttpPost]
public string AddRequest (DeviceForSurgeryDTO newdeviceForSurgery)
{
    deviceForSurgery d = newdeviceForSurgery.DeviceToTable();
    db.deviceForSurgery.Add(d);
    db.SaveChanges();
    return "succes";
}
```

פונקציה המוסיפה בקשת מכשיר חדשה למערכת.

DoctorController - קונטרולר שישלוף את רשימות הרופאים הקיימים במערכת.

-SchedulingController

קונטרולר שמקבל מהמשתמש פניות לצפייה בשיבוץ והפעלתו ומפעיל את הפונקציות הבאות:

-GetSchedulng()

פונקציה המפעילה ב-HungrienScudling פונקציה המפעילה את אלגוריתם השיבוץ וההתאמה הכי אופטימאלית בין חדר לניתוח.

-SpecialDeviceController

קונטרולר שמקבל מהמשתמש פניות בנוגע למכשירים ומפעיל את הפונקציות הבאות:

-GetAllSpecialDevice()

פונקציה המפעילה ב-SpecialDeviceManager פונקציה השולפת את כל המכשירים שקיימים במערכת.

-AddSpecialDevice()

```
[Route("api/SpecialDevice/AddRequest")]
[HttpPost]
public string AddSpecialDevice(SpecialDeviceDTO newSpecialDevice)
{
    specialDevice s = newSpecialDevice.SpecialDeviceToTable();
    db.specialDevice.Add(s);
    db.SaveChanges();
    return "succes";
}
```

פונקציה המוסיפה מכשיר חדש למערכת.

-BL

שכבה זו מכילה את הלוגיקה של הפרויקט. כמו כן, משמשת כמתווכת בין שכבת ה-DAL לשכבת ה-WebApi בשליפת ובעדכון הנתונים.

-RoomManager

מחלקה שמכילה את כל הפעולות הקשורות לחדרים כמו:

-GetClearRoom()

פונקציה השולפת את כל החדרים הריקים במאגר הנתונים.

-GetAllRoom()

פונקציה השולפת את כל החדרים הנמצאים במערכת.

-AddNewRoom()

פונקציה המוסיפה חדר חדש למערכת.

המחלקה מקבלת פניות מהלקוח על ידי ה-RoomController ועושה שינויים בהתאם בדאטה בייס או מציגה מידע.

-SurgeryManager

מחלקה שמכילה את כל הפעולות הקשורות לניתוחים כמו:

-GetSurgeryFromCurrentDate()

פונקציה השולפת את כל הניתוחים מהתאריך הנוכחי.

-GetSurgeryFromSpecificDate()

פונקציה השולפת ניתוחים מתאריך מסוים שהתקבל מהמשתמש.

-GetAllSurgery()

פונקציה השולפת את כל הניתוחים הנמצאים במערכת.

-AddNewSurgery()

```
public static SurgeryDTO AddNewSurgery(SurgeryDTO AddSurgery)
{
surgery newSurgery = AddSurgery.SurgeryToTable();
db.Execute<surgery>(newSurgery, DBConection.ExecuteActions.Insert);
AddSurgery.surgeryCode = newSurgery.surgeryCode;
return AddSurgery;

}
```

פונקציה המוסיפה ניתוח חדש למערכת.

המחלקה מקבלת פניות מהלקוח על ידי -SurgeryController ועושה שינויים בהתאם בדאטה בייס או מציגה מידע.

-UserManager

מחלקה שמכילה את כל הפעולות הקשורות לניתוחים כמו:

-LoginUser()

```
public static UserDTO LoginUser(UserDTO checkUser)
{
    user userFromTable = db.GetDbSet<user>().FirstOrDefault(U => U.userName== checkUser.userName && U.password== checkUser.password);
    if (userFromTable == null)
        return null;
    else
        return new UserDTO(userFromTable);

}
```

פונקציה המקבלת את פרטי המשתמש ובודקת האם משתמש זה קיים במערכת ופרטי ההזדהות נכונים.

-RegisterUser()

```
public static UserDTO RegisterUser(UserDTO AddUser)
{
    user newLogin = AddUser.UserToTable();
    db.Execute<user>(newLogin, DBConection.ExecuteActions.Insert);
    AddUser.idUser = newLogin.idUser;
    return AddUser;


}
```

פונקציה המקבלת את פרטי המשתמש ומוסיפה את המשתמש החדש למערכת.

-UpdateUser()

פונקציה המקבלת שינויים מהמשתמש ומעדכנת אותם במערכת.

-DeleteUser()

פונקציה המקבלת משתמש ומסירה אותו מהמערכת.

המחלקה מקבלת פניות מהלקוח על ידי ה -UserController ועושה שינויים בהתאם בדאטה בייס.

-SpecialDeviceManager

מחלקה שמכילה את כל הפעולות הקשורות למכשירים כמו:

-GetAllSpecialDevice()

פונקציה השולפת את כל המכשירים שקיימים במערכת.

-AddSpecialDevice()

פונקציה המוסיפה מכשיר חדש למערכת.

המחלקה מקבלת פניות מהלקוח על ידי ה- SpecialDeviceController ועושה שינויים בהתאם בדאטה בייס או מציגה מידע.

-ClassManager

מחלקה שמכילה את כל הפעולות הקשורות למחלקות כמו הוספה, מחיקה, עדכון על ידי ה- ClassController ועושה שינויים בהתאם בדאטה בייס.

-DeviceForSurgeryManager

מחלקה שמכילה את כל הפעולות הקשורות לבקשות מכשירים לניתוח כמו:

-GetAllRequest()

פונקציה השולפת את כל הבקשות למשירים הקיימות במערכת.

-AddRequest()

פונקציה המוסיפה בקשת מכשיר חדשה למערכת.

המחלקה מקבלת פניות מהלקוח על ידי ה- DeviceForSurgeryController ועושה שינויים בהתאם בדאטה בייס או מציגה מידע.

-DoctorManager

מחלקה שמכילה את כל הפעולות הקשורות לרופאים כמו הוספה, מחיקה, עדכון על ידי ה- DoctorController ועושה שינויים בהתאם בדאטה בייס.

-SchedulingManager

מחלקה שמכילה את כל הפעולות הקשורות לשיבוץ כמו הוספה, מחיקה, עדכון על ידי ה- SchedulingController ועושה שינויים בהתאם בדאטה בייס.

-PreHungrien

מחלקה המכילה את כל ההכנות המקדימות לאלגוריתם כמו:

-CalculateScore()

פוונקציה זו יוצרת מטריצה עם ציון התאמה בין חדר לניתוח (אפיון פונקציונאלי).

-CalculatePriority()

```
public IDictionary<double,SurgeryDTO> CalculatePriority(List<SurgeryDTO> listOfSurgery)
{
    IDictionary<double,SurgeryDTO> priorityList=new Dictionary<double,SurgeryDTO>();
    foreach (var ls in listOfSurgery)
    {
        priorityScore = (ls.dangerLevel * 0.85) + (ls.priorityLevel * 0.15);
        priorityList.Add(priorityScore, ls);
    }
    priorityList.OrderBy(p => p.Key);

    return priorityList;
}
```

פונקציה המחשבת לכל ניתוח את רמת הדחיפות שלו.

-MatchRoom()

פונקציה הבודקת האם החדר והניתוח ממחלקה דומה.

-MatchDevice()

```
public double MatchDevice(List<DeviceForSurgeryDTO> D, List<SpecialDeviceDTO>S,SurgeryDTO surg)
{
    double sumMatchDavice = 0;
    List<DeviceForSurgeryDTO> surgeryDevices = D.Where(sd => sd.surgeryCode == surg.surgeryCode).ToList();
    foreach(var x in surgeryDevices)
    {
        foreach(var y in S)
        {
            if ((x.deviceName == y.deviceName) &&( y.isAvailable == false) && (y.amount>x.amount))
            {
                y.date = surg.surgeryDate;
                y.isAvailable = true;
                y.amount -= x.amount;
                sumMatchDavice += 2;
            }
        }
    }
    return sumMatchDavice;
}
```

פונקציה המעניקה מכשיר לניתוח על פי בקשה.

-Grade()

פונקציה המפעילה את הפונקציות לעיל ומכמת ציון סופי לכל התאמה בין חדר לניתוח (אפיון פונקציונאלי).

-HungrienScudling

מחלקה שבה נמצא האלגוריתם ההונגרי ומתבצע השיבוץ בפועל (פירוט על פונקציות המחלקה בקוד האלגוריתם).

-DAL

בשכבה זו קיים מודל שנבנה ע"י טכנולוגיית EntityFrameWork, ובו מחלקה מקבילה לכל טבלה במסד הנתונים.

-DTO

ספריה זו מכילה את מחלקות מקבילות למבנה הנתונים כדי לקשר בין שכבת ה-DAL וה-WebApi.

-DeviceForSurgeryDTO

מחלקת מכשירים השמורים לניתוח:

-CreateDeviceForSurgeryDtoList()

```
public static List<DeviceForSurgeryDTO> CreateDeviceForSurgeryDtoList(List<deviceForSurgery> deviceForSurgeryList)
{
    List<DeviceForSurgeryDTO> dtoDeviceForSurgeryList = new List<DeviceForSurgeryDTO>();
    foreach (var c in deviceForSurgeryList)
    {
        DeviceForSurgeryDTO dtoDeviceForSurgery = new DeviceForSurgeryDTO(c);
        dtoDeviceForSurgeryList.Add(dtoDeviceForSurgery);
    }
    return dtoDeviceForSurgeryList;
}
```

משמש להמרת רשימה מטיפוס טבלה ל-DTO.

-DeviceToTable()

```
public deviceForSurgery DeviceToTable()
{
    deviceForSurgery dfs = new deviceForSurgery();
    dfs.idDFS = idDFS;
    dfs.idDevice = idDevice;
    dfs.surgeryCode = surgeryCode;
    dfs.deviceName = deviceName;
    dfs.amount = amount;
    return dfs;
}
```

משמש להמרה מטיפוס DTO לטבלה.

בדומה למחלקה לעיל יש פונקציות זהות בשאר המחלקות העושות את אותן הפעולות.

-DoctorDTO

מחלקת רופאים משמש להמרה לטבלה וממנה.

-RoomDTO

מחלקת חדרים משמש להמרה לטבלה וממנה.

-SpecialDeviceDTO

מחלקת מכשירים משמש להמרה לטבלה וממנה.

-SurgeryDTO

מחלקת ניתוחים משמש להמרה לטבלה וממנה.

-UserDTO

מחלקת משתמשים משמש להמרה לטבלה וממנה.

-ClassDTO

מחלקת סוגי מחלקות משמש להמרה לטבלה וממנה.

-SchedulingDTO

מחלקת שיבוץ משמשת להמרה לטבלה וממנה.

13. תיאור התוכנה

- **סביבת עבודה :**
  Visual Studio Code ו Visual Studio
- **שפות תכנות:**
  צד השרת נכתב בטכנולוגית WebApi ובשפת #c .
  צד הלקוח נכתב בטכנולוגית angular ובשפות typescript, Html, css.

14. אלגוריתמים מרכזיים

פונקציית האלגוריתם העיקרי מקבלת רשימת חדרים, ניתוחים, מכשירים, ומכשירים שצריך כל ניתוח אותם היא שולחת לפונקציה שבונה מטריצה ונותנת ציון לכל התאמה בין חדר לניתוח על פי רמת עדיפות וסיכון שנתן הרופא, מחלקה תואמת, ומכשירים מתאימים לניתוח את הציון הסופי משבצים בריבוע המתאים במטריצה ואת המטריצה שולחים בחזרה למחלקת השיבוץ

על המטריצה המוכנה מפעילים את פעולת השיבוץ שתמצא את ההתאמה הטובה ביותר עבור כל ניתוח היכן יתבצע, האלגוריתם עובר על המטריצה מוצא ציון מקסימאלי בכל שורה ולאחר מכן בכל עמודה ואז בודק אם יש ציון חד חד ערכי במקרה שלא ממשיך בפעולות נוספות כפי שאפשר לראות בקוד עד אשר ימצא הפתרון הסופי.

14.1. חלק מהאלגוריתם עובר על המטריצה מוצא מינימום

```
//מעבר זה על המטריצה מבטיח שבכל שורה יש לפחות תא אחד שהוא אפס
for (var i = 0; i < h; i++)
{
    var min = double.MaxValue;

    for (var j = 0; j < w; j++)
    {
        min = Math.Min(min, gradeMat[i, j]);
    }

    for (var j = 0; j < w; j++)
    {
        gradeMat[i, j] -= min;//מחסיר את המינימום בכל שורה מכל התאים בשורה
    }
}

//יצירת מסיכה המשמשת להבנת מצב שיבוץ החדרים והניתוחים
var masks = new byte[h, w];
//רשימה המייצגת את מצב שיבוץ החדרים
var rowsCovered = new bool[h];
//רשימה המייצגת את מצב שיבוץ הניתוחים
var colsCovered = new bool[w];

//מעבר זה על המטריצה ממלא את המסיכה ב-1 במקומות שבמטריצה המקורים הם הציון המינימאלי-כלומר ישנה התאמה טובה בין חדר לניתוח
//במקרה שיש 2 בעמודה או בשורה יסומן הראשון מבינהם
for (var i = 0; i < h; i++)
{
    for (var j = 0; j < w; j++)
    {
        if (gradeMat[i, j] == 0 && !rowsCovered[i] && !colsCovered[j])
        {
            masks[i, j] = 1;
            rowsCovered[i] = true;
            colsCovered[j] = true;
        }
    }
}
```

14.2. באלגוריתם השיבוץ ישנם ארבעה צעדים שעוברים עליהם וחוזרים עד שמגיעים לתוצאת השיבוץ וזו הפעולה העיקרית באלגוריתם זה

```
while (step != -1)
{
    switch (step)
    {
        case 1:
            step = HungrienScudling.RunStep1(masks, colsCovered, w, h);
            break;

        case 2:
            step = HungrienScudling.RunStep2(gradeMat, masks, rowsCovered, colsCovered, w, h, ref pathStart);
            break;

        case 3:
            step = HungrienScudling.RunStep3(masks, rowsCovered, colsCovered, w, h, path, pathStart);
            break;

        case 4:
            step = HungrienScudling.RunStep4(gradeMat, rowsCovered, colsCovered, w, h);
            break;
    }
}
```

14.3. חלק זה באלגוריתם זהו החלק שמסמן את אופציית ההתאמה הראשונית בין ניתוח לחדר

```
//מחפשת את המיקום של התא הראשון במטריצת הציונים שבו יש אפס - אם לא קיים תא מחזיר מיקום -1-1//
private static Location FindZero(double[,] gradeMat, bool[] rowsCovered, bool[] colsCovered, int w, int h)
{
    if (gradeMat == null)
        throw new ArgumentNullException(nameof(gradeMat));

    if (rowsCovered == null)
        throw new ArgumentNullException(nameof(rowsCovered));

    if (colsCovered == null)
        throw new ArgumentNullException(nameof(colsCovered));

    for (var i = 0; i < h; i++)
    {
        for (var j = 0; j < w; j++)
        {
            if (gradeMat[i, j] == 0 && !rowsCovered[i] && !colsCovered[j])
                return new Location(i, j);
        }
    }

    return new Location(-1, -1);
}
```

15.קוד האלגוריתם

הפונקציות הבאות הן ארבעת הצעדים של אלגוריתם השיבוץ:

צעד ראשון -

```
//מחזירה -1 אם השיבוץ הסתיים ו-2 אם לא - מסמל שלא כל הניתוחים משובצים
private static int RunStep1(byte[,] masks, bool[] colsCovered, int w, int h)
{
    if (masks == null)
        throw new ArgumentNullException(nameof(masks));

    if (colsCovered == null)
        throw new ArgumentNullException(nameof(colsCovered));
    //מסמן באיזה עמודות יש תאים שסומנו
    for (var i = 0; i < h; i++)
    {
        for (var j = 0; j < w; j++)
        {
            if (masks[i, j] == 1)
                colsCovered[j] = true;
        }
    }

    //סופר כמה עמודות סומנו סה''כ
    var colsCoveredCount = 0;

    for (var j = 0; j < w; j++)
    {
        if (colsCovered[j])
            colsCoveredCount++;
    }

    // אם כל העמודות נבחרו - אם כל דבר שרצינו לשבץ מצא משבצת
    if (colsCoveredCount == h)
        return -1;

    return 2;
```

צעד שני-

```
// ????הפונקציה
//מחזירה:
//4 - אם אין תא מאופס במטריצת הציונים - אם אין עוד שום ניתוח שמתאים לשום חדר
//3 - אם מצאנו שורה שבא רק אין עוד תא מאופס בשורה - אם מצאנו חדר שבו אין עוד ניתוח מתאים
private static int RunStep2(double[,] gradeMat, byte[,] masks, bool[] rowsCovered, bool[] colsCovered, int w, int h, ref Location pathStart)
{
    if (gradeMat == null)
        throw new ArgumentNullException(nameof(gradeMat));

    if (masks == null)
        throw new ArgumentNullException(nameof(masks));

    if (rowsCovered == null)
        throw new ArgumentNullException(nameof(rowsCovered));

    if (colsCovered == null)
        throw new ArgumentNullException(nameof(colsCovered));

    while (true)
    {
        //מחפשת ניתוח ראשון שמשובץ בכל חדר
        var loc = HungrienScudling.FindZero(gradeMat, rowsCovered, colsCovered, w, h);
        if (loc.row == -1) //אם אין במטריצה שום ניתוח שמתאים לאיזשהוא חדר
            return 4; //יש לעבור לצעד 4

        //משנה את התא במסיכה שהיה 1 להיות 2
        //שיבוץ החדר והניתוח נבחרו
        masks[loc.row, loc.column] = 2;

        var starCol = HungrienScudling.FindStarInRow(masks, w, loc.row);
        //אם אכן יש עוד תא שערכו 1
        if (starCol != -1)
        {
            //מבחינת השורה הוא לא מעניין - כימצאנו תא  ??????????????????????????
```

```
            //מבחינת השורה הוא לא מעניין - כימצאנו תא  ??????????????????????????
            //אץ החדר סידרנו וסימנו - כי מצאנו ניתוח שמשתבץ בו
            rowsCovered[loc.row] = true;
            //אבל בעמודות אני אסמן שהעמודה לא נבדקה - כי אולי זה התא היחיד בעמודה     ??????????????
            //אבל לגבי הניתוח האחר שמצאנו שמתאים לחדר הזה - לא נסמן אותו - כי החדר הזה תפוס
            colsCovered[starCol] = false;
        }
        //אם אכן אין עוד תא שערכו 1 - כלומר אין עוד אופציות בחדר הנוכחי

        else
        {
            pathStart = loc; //חייב להיות שנתחיל מהמיקום הזה  - התא הראשון שנמצא
            return 3;
        }
    }
}
```

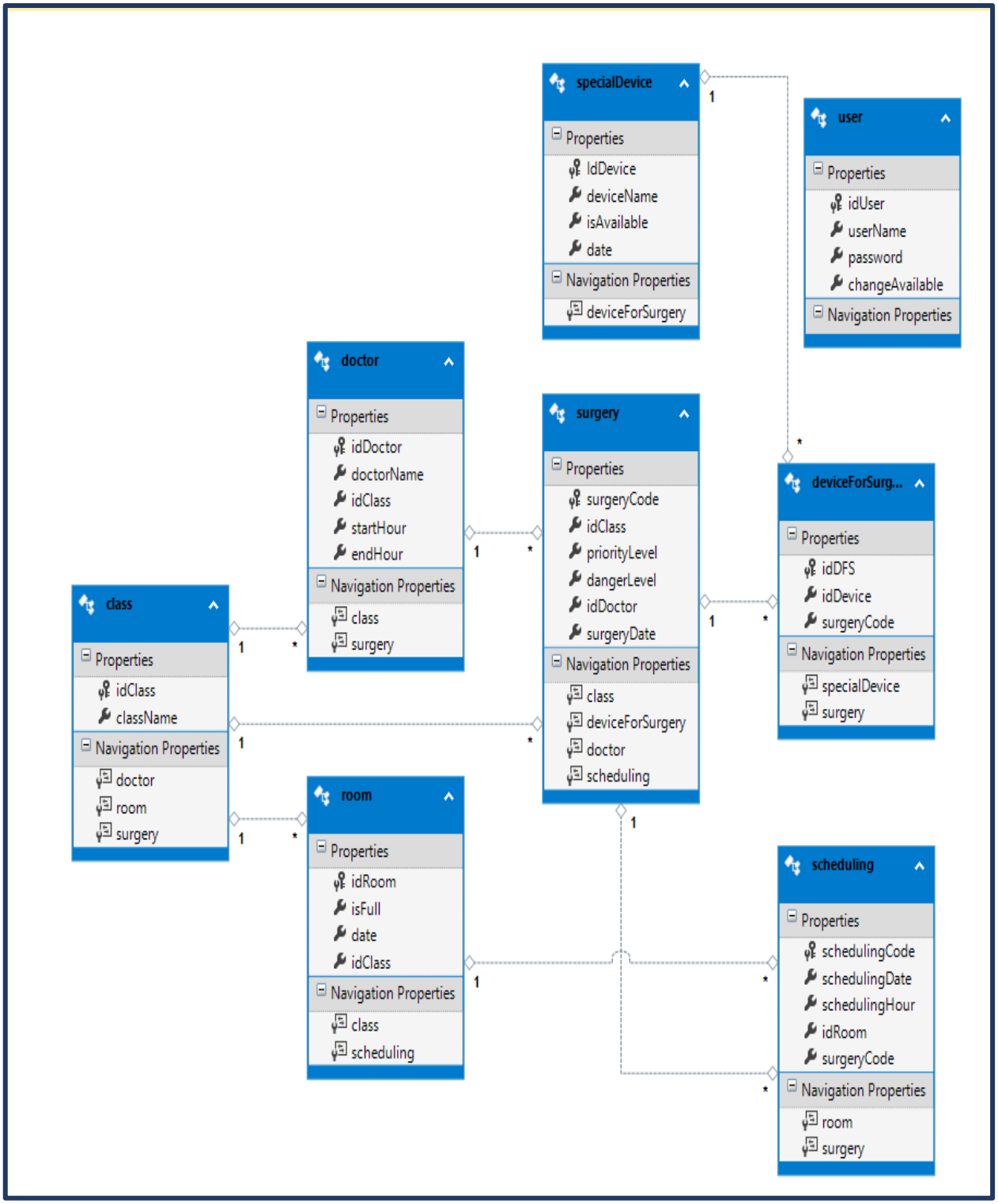
צעד שלישי-

```
//מסמנת במערך המיקומים את מקום השיבוץ שנבחר במידה ואין עוד התאמה מנקה את  מערכי העזר
//וממשיך לבדוק במקרה ויש??????
private static int RunStep3(byte[,] masks, bool[] rowsCovered, bool[] colsCovered, int w, int h, Location[] path, Location pathStart)
{
    if (masks == null)
        throw new ArgumentNullException(nameof(masks));

    if (rowsCovered == null)
        throw new ArgumentNullException(nameof(rowsCovered));

    if (colsCovered == null)
        throw new ArgumentNullException(nameof(colsCovered));

    var pathIndex = 0;
    path[0] = pathStart;

    while (true)
    {
        var row = HungrienScudling.FindStarInColumn(masks, h, path[pathIndex].column);
        if (row == -1)
            break;

        pathIndex++;
        path[pathIndex] = new Location(row, path[pathIndex - 1].column);

        var col = HungrienScudling.FindPrimeInRow(masks, w, path[pathIndex].row);

        pathIndex++;
        path[pathIndex] = new Location(path[pathIndex - 1].row, col);
    }

    HungrienScudling.ConvertPath(masks, path, pathIndex + 1);
    HungrienScudling.ClearCovers(rowsCovered, colsCovered, w, h);
    HungrienScudling.ClearPrimes(masks, w, h);

    return 1;
}
```

צעד רביעי-

```
private static int RunStep4(double[,] gradeMat, bool[] rowsCovered, bool[] colsCovered, int w, int h)
{
    if (gradeMat == null)
        throw new ArgumentNullException(nameof(gradeMat));

    if (rowsCovered == null)
        throw new ArgumentNullException(nameof(rowsCovered));

    if (colsCovered == null)
        throw new ArgumentNullException(nameof(colsCovered));

    var minValue = HungrienScudling.FindMinimum(gradeMat, rowsCovered, colsCovered, w, h);

    for (var i = 0; i < h; i++)
    {
        for (var j = 0; j < w; j++)
        {
            if (rowsCovered[i])
                gradeMat[i, j] += minValue;
            if (!colsCovered[j])
                gradeMat[i, j] -= minValue;
        }
    }
    return 2;
}
```

16.תיאור מסד הנתונים



16.1. פירוט הטבלאות ב-Data Base

User-

טיפוס	תיאור	שם שדה	מפתח
int	קוד משתמש	idUser	pk
varchar	שם משתמש	userName	
int	סיסמא	password	
bit	מורשה/לא	changeAvailable	

מכילה את רשימת המשתמשים הנמצאים בדאטה קוד משתמש זהו תעודת זהות של המשתמש ועמודה זו היא גם מפתח ראשי.

Surgery-

טיפוס	תיאור	שם שדה	מפתח
int	קוד ניתוח	surgeryCode	pk
int	קוד מחלקה	idClass	fk
int	רמת עדיפות	priorityLevel	
int	רמת סיכון	dangerLevel	
int	קוד רופא	idDoctor	fk
datetime	תאריך כניסה למערכת	surgeryDate	

טבלת ניתוחים- טבלה המכילה את רשימת הניתוחים שנכנסו למערכת מפתח ראשי של טבלה זו הוא קוד ניתוח וקוד מחלקה וקוד רופא אלו מפתחות זרים המקושרים לטבלאות מחלקות ורופאים בהתאמה.

SpecialDevice-

מפתח	שם שדה	תיאור	טיפוס
pk	IdDevice	קוד מכשיר	int
	deviceName	שם מכשיר	varchar
	isAvailable	זמין/לא	bit
	date	תאריך	Datetime

טבלת מכשירים- טבלה זו מכילה רשימת מכשירים שיש בבית החולים שזמינים או תפוסים, במקרה שפנוי התאריך משתנה לתאריך הניתוח שתפס.

Scheduling-

מפתח	שם שדה	תיאור	טיפוס
pk	schedulingCode	קוד שיבוץ	int
	schedulingDate	תאריך שיבוץ	date
	schedulingHour	שעת שיבוץ	time
fk	idRoom	קוד חדר	int
fk	surgeryCode	קוד ניתוח	int

טבלת שיבוץ- מכילה את כל השיבוצים בין חדר לניתוח שעששו אחרי הפעלת האלגוריתם המפתח הראשי הוא קוד השיבוץ והמפתחות הזרים הם קוד ניתוח וקוד חדר שמקורם לטבלת חדרים וניתוחים

Room-

מפתח	שם שדה	תיא ר	טיפוס
pk	idRoom	קוד חדר	int
	isFull	מלא/לא	bit
	date	תאריך	date
fk	idClass	מחלקה	int

טבלת חדרים- מכילה את רשימת החדרים בין אם פנויים ובין אם לא ההמפתח הראשי הוא קוד חדר והמפתח הזר הוא קוד מחלקה שמקורו בטבלת מחלקות.

Doctor-

מפתח	שם שדה	תיאור	טיפוס
pk	idDoctor	קוד רופא	int
	doctorName	שם רופא	varchar
fk	idClass	קוד מחלקה	int
	startHour	שעת התחלה	datetime
	endHour	שעת סיום	datetime

טבלת רופאים המכילה את רשימת הרופאים הקיימים במערכת ואת שעות עבודתם המפתח הראשי הוא קוד רופא ומתח זר קוד מחלקה שמקורו בטבלת מחלקות.

DeviceForSurgery-

מפתח	שם שדה	תיאור	טיפוס
pk	idDFS	קוד בקשה	int
fk	idDevice	קוד מכשיר	int
fk	surgeryCode	קוד ניתוח	int

טבלה המכילה בקשות מכשירים לניתוח המפתח הראשי הוא קוד הבקשה מפתחות זרים קוד מכשיר וקוד ניתוח שמקורם מטבלת מכשירים וניתוחים בהתאמה.

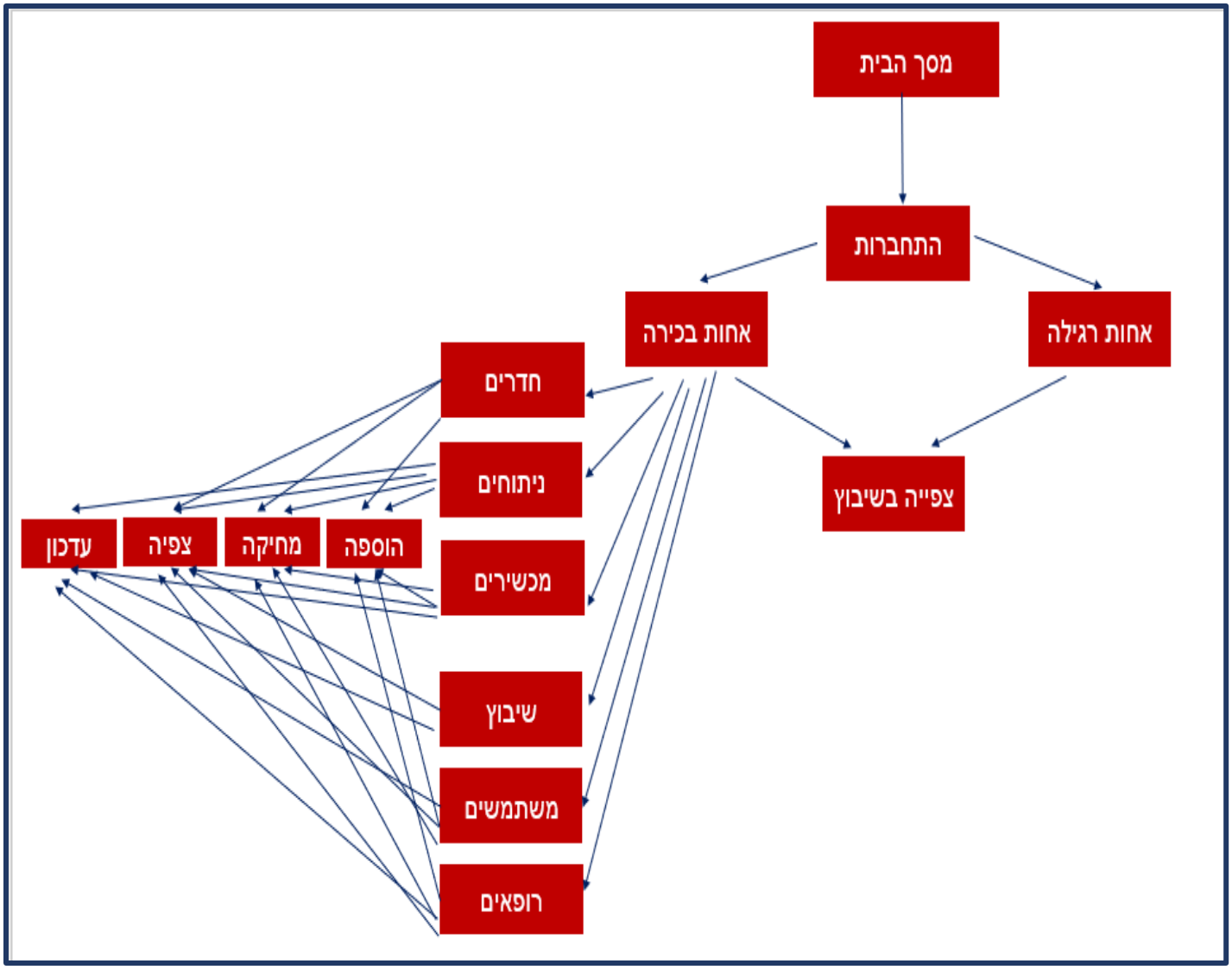
Class-

מפתח	שם שדה	תיאור	טיפוס
pk	idClass	קוד מחלקה	int
	className	שם מחלקה	varchar

טבלה המכילה את שמות המחלקות המפתח הראשי הוא קוד המחלקה.

17. מדריך למשתמש

17.1. תיאור המסכים



17.2. מדריך למשתמש

האפליקציה נפתחת על דף הבית על מנת לגשת לפעולות ולהפעיל את האפליקציה יש להירשם תחילה, על המשתמש להכניס שם משתמש וסיסמא.

כיוון שזוהי אפליקציה רפואית לא כל אחד יכול להוסיף משתמש חדש אלא רק משתמש קיים, ולכן תחילה עליו להיכנס למערכת ואז להוסיף את המשתמש החדש שיכול אף הוא להתחבר מעתה.

הפעולות האפשריות שיהיו זמינות למערכת תלויות בסוג המשתמש שהתחבר:

במקרה שזו אחות רגילה היא תוכל לגשת לחלונית המשתמשים ולהוסיף משתמש חדש ולחלונית השיבוץ ולצפות בשיבוצי הניתוחים שהתקיימו וכבר מסודרים במערכת בתור לוח זמנים.

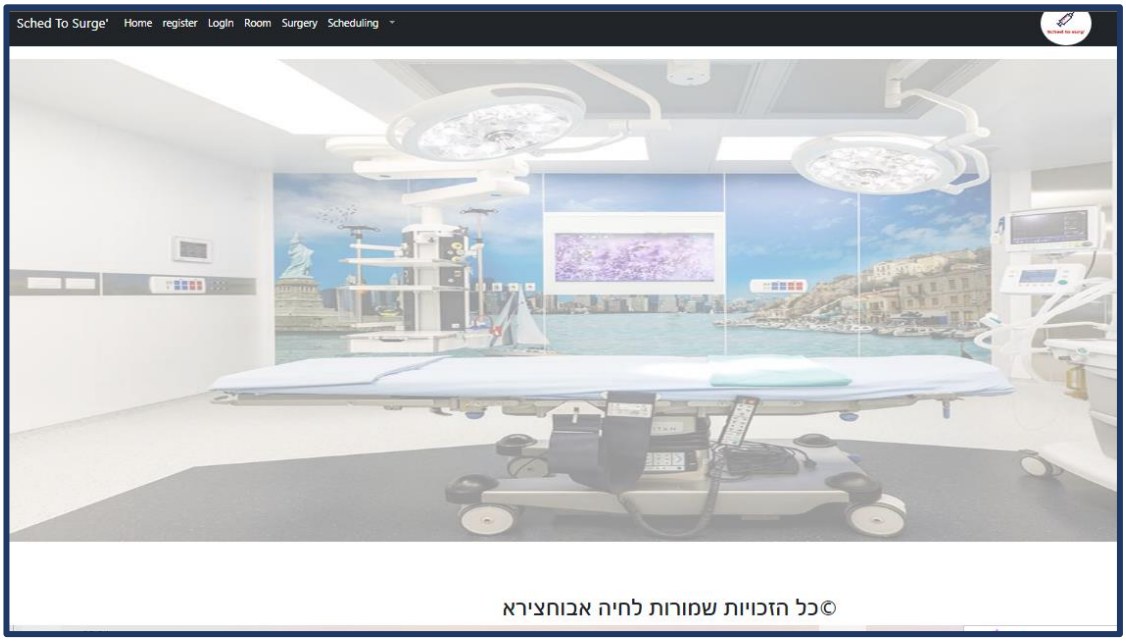
במקרה שזו אחות בכירה היא יכולה לגשת ליותר פעולות ושאר החלוניות במערכת יהיו זמינים לה כמו חדרים, מחלקות, ניתוחים, רופאים, משתמשים, מכשירים, שיבוץ.

בחדרים יש מספר פעולות כמו שליפת רשימת חדרים פנויים, שליפה על ידי סינון לפי מחלקה, הוספת חדרים או מחיקתם ועדכון האם פנויים או לא.
במחלקות יש אפשרות להוסיף, למחוק או לעדכן כמו כן אפשר גם לצפות בכל המחלקות הקיימות בבית החולים.
ניתוחים- בחלונית זו אפשר להוסיף ניתוחים, לבקש מכשיר מסוים לניתוח, לעדכן, למחוק ולצפות בניתוחים הקיימים במערכת.
בחלונית הרופאים אפשר להוסיף למערכת רופאים, לצפות ברופאים שעובדים היום או בכללי לצפות ברופאי בית החולים, לצפות באלו שעובדים בתאריך מסוים, יהיה אפשר גם לצפות בניתוחים ששובצו רופא מסוים, למחוק או לעדכן.
במשתמשים בדומה לאחות רגילה האחות הבכירה יכולה להוסיף משתמש חדש ולעדכן אך בשונה מאחות רגילה רק אחות בכירה רשאית למחוק, לעדכן או לצפות ברשימת המשתמשים.
מכשירים- בחלונית זו ניתן להוסיף מכשירים, לצפות ברשימת המכשירים הקיימים, לעדכן או למחוק.
שיבוץ- בחלונית זו אפשר להפעיל את השיבוץ בפועל וכיצד?
לאחר שהוספנו מספר ניתוחים למערכת שעדיין אינם משובצים אפשר ללחוץ על שיבוץ, בפעולה זו ישלחו לפעולת השיבוץ רשימת ניתוחים שעדין לא שובצו, המכשירים שהם זקוקים להם, וחדרים ריקים, המערכת תשבץ את הניתוחים בחדר המתאים ביותר ותציג את לוח הזמנים למשתמש.

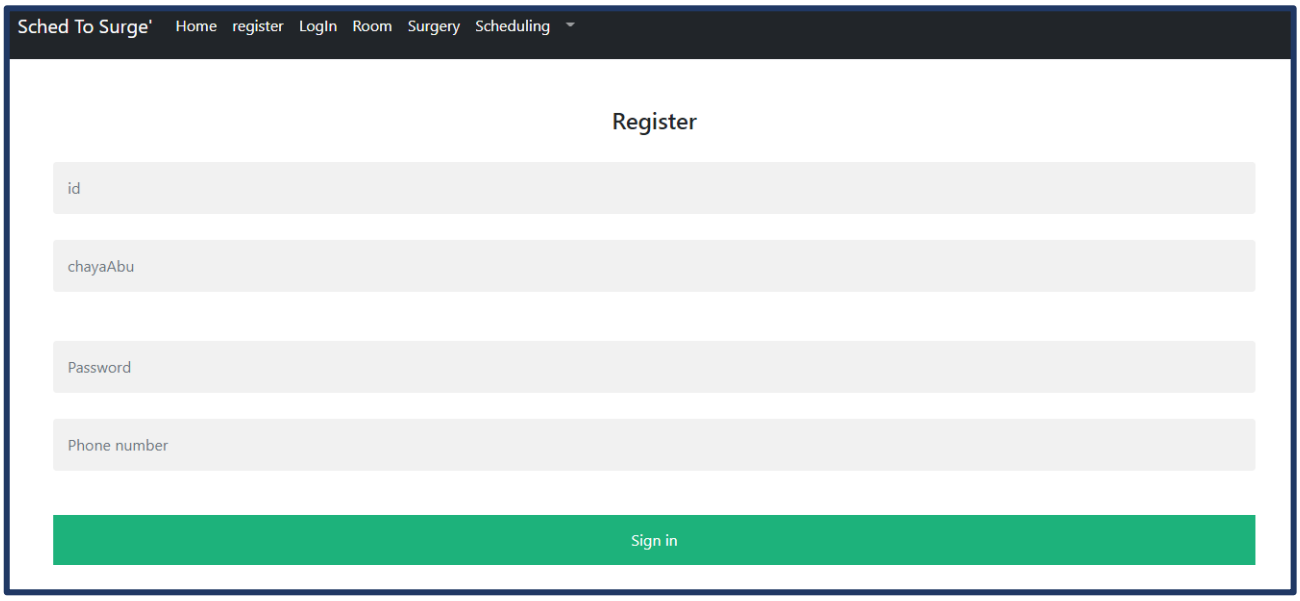
בכל רגע נתון יהיה אפשר להוסיף ניתוח חירום למערכת בפעולה זו הניתוח ישובץ באופן מידי ושאר הניתוחים יידחו ויעודכנו במערכת.

17.3. צילומי מסכים

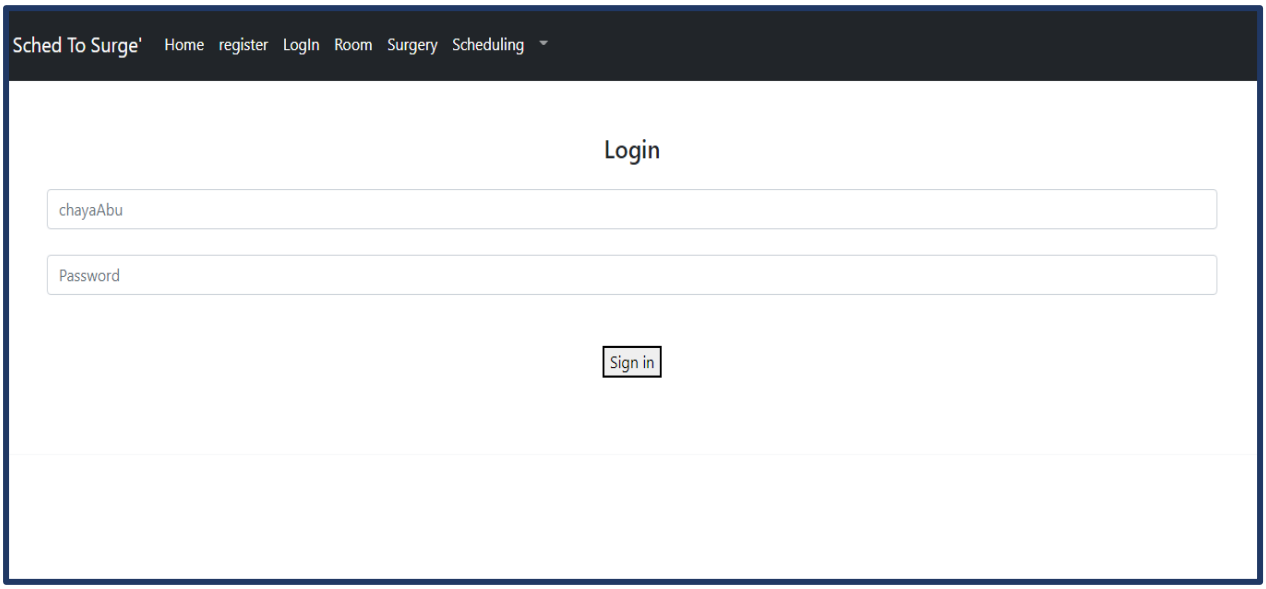
Home-



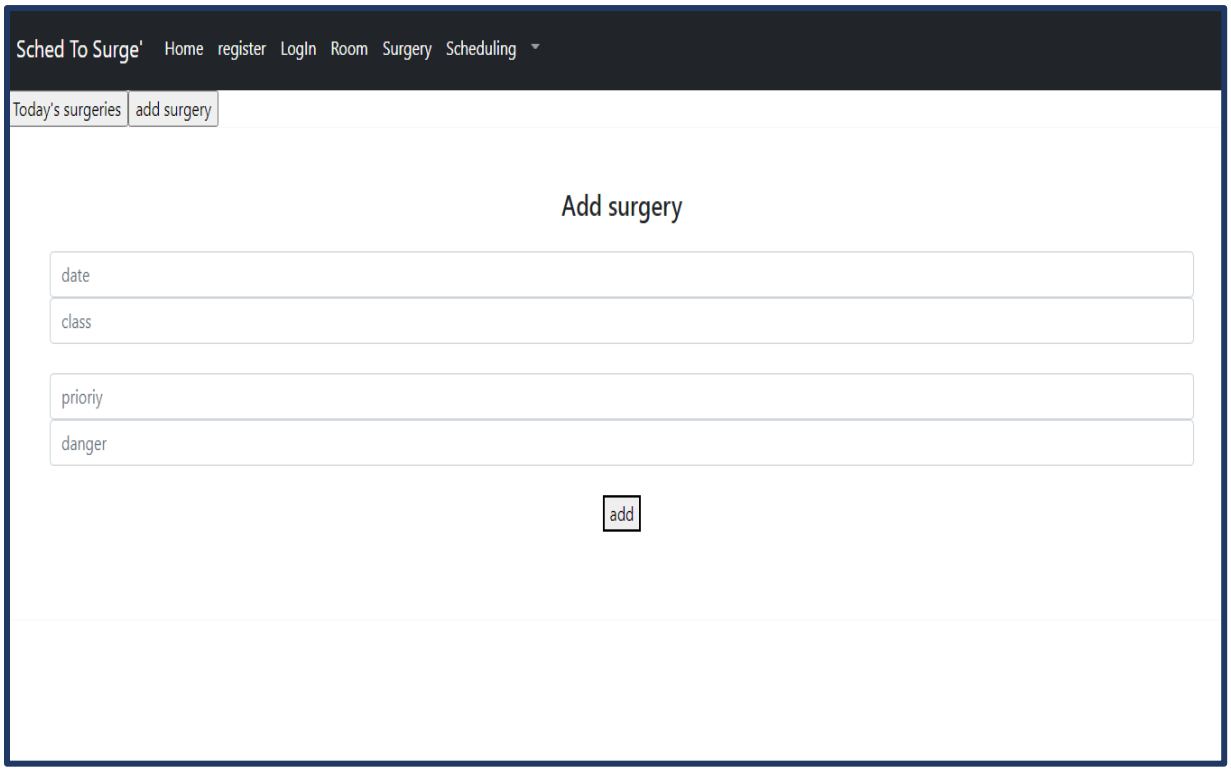
-Register



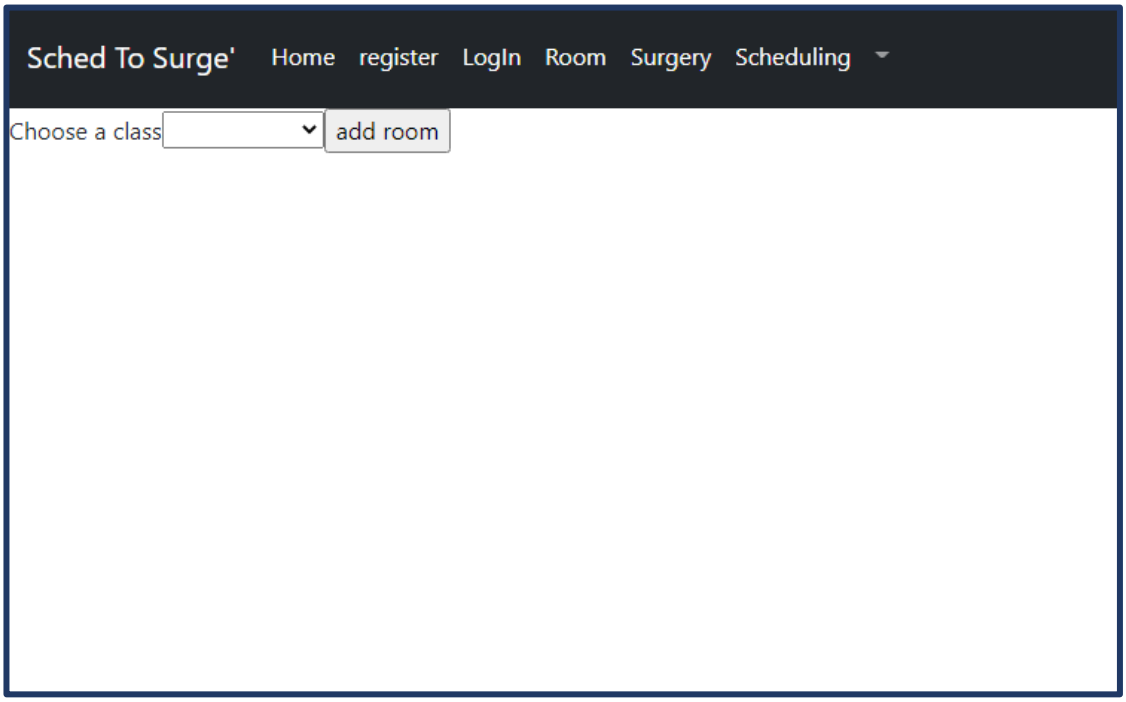
-Login



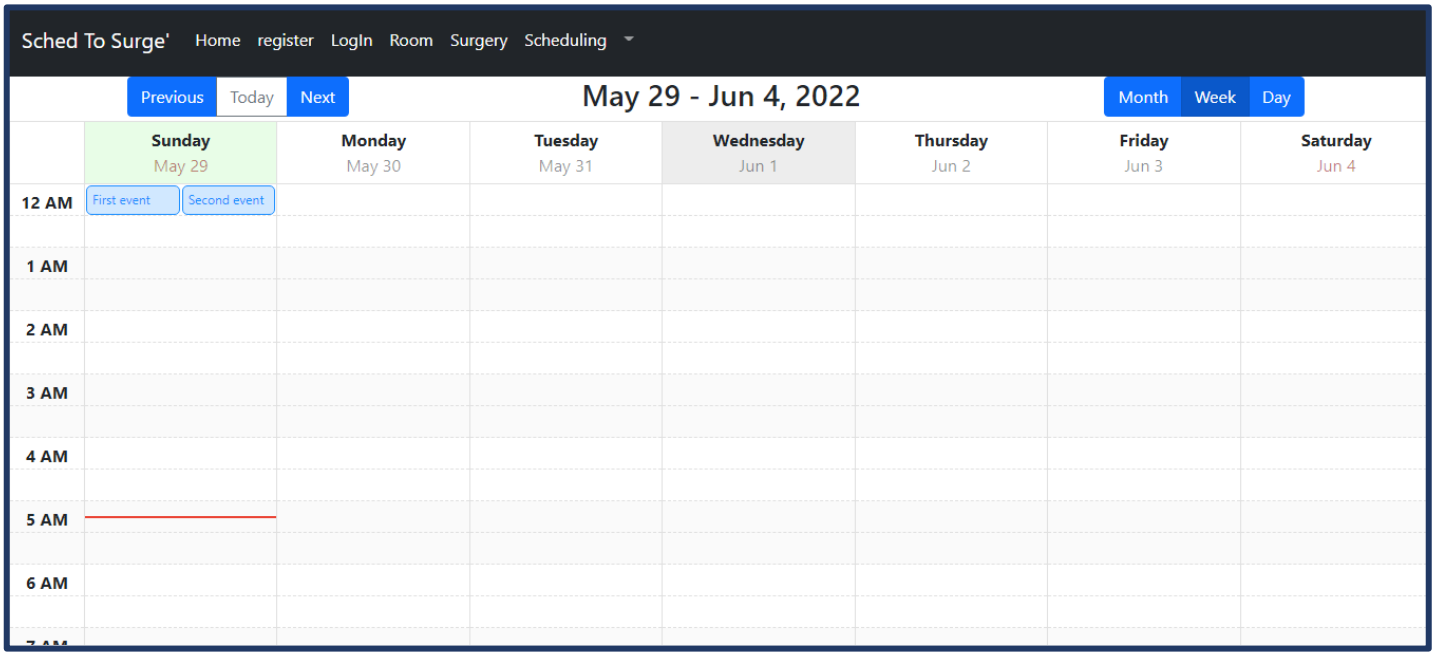
-Surgery



-Room



Scheduling-



18.בדיקות והערכה

לאחר הפעלת אלגוריתם השיבוץ, ונתינת ציון על כל מיני פרמטרים כדי להגיע לשיבוץ האופטימלי האפשרי אך כאשר הופיעו טעויות ובאגים של האלגוריתם נבדק הקוד שוב ושוב עד שתוקנו הבעיות והקוד רץ תקין, לאחר תיקונים ודיוקים שנעשו באלגוריתם ובדיקת מקרים אפשריים שלא תמיד חושבים עליהם, האלגוריתם הגיע לתוצאה האופטימאלית לוח זמנים משובץ עם ניתוחים וחדרים ושעות לכל אחד

19.ניתוח יעילות

יעילות של האלגוריתם היא זמן פולינמי של P(n) כמובן שחשוב מאד זמן ריצתו של האלגוריתם בייחוד שמדובר על אפליקציות ותוכנות שצריכות תשובה מיידית צריך כמה שפחות סיבוכיות.

20.אבטחת מידע

כיוון שזוהי אפליקציה רפואית חייב להיות לה אבטחת מידע ולא כל אחד יכול לגשת אליה ולכן בכניסה למערכת יש הרשמה ורק משתמשים שנמצאים במערכת יכולים להתחבר, אפילו הרשמת משתמש חדש לא כל אד יכול לעשות רק משתמש רשום.

21.מסקנות

כתיבת הפרויקט זה אחד הדברים החשובים שעשיתי קיבלתי מבט על צורת העבודה ששנעית באמת, היינו צריכות לחפש נושא בעצמנו לחקור עליו למצוא לו פיתרון לכתוב ולהתאים את האלגוריתם לפרויקט שלנו.
כאשר עובדים על פרויקט בסדר גודל כזה לומדים שלא יר כל מה ששחושבים כותבים אלא יושבים חושבחים על דרך פיתרון על כל האופציות האפשריות ומקרי הקצה שיעלו ולפי כך כותבים את הקוד.
בתהליך הפרויקט צברתי ידע רב גם בנושא עצמו,בפתירת באגים והתמודדות אישית איתם, חיפוש באינטרנט על דרך הפתירה שלהם והניסיון לפתור אותם בעצמנו וז שמתי לב שכשנתקלתי ששוב בבאג דומה כבר ידעתי כיצד לפעול ולטפל בו.
לפעמים זה הי נדמה כאילו הבאג במקום מסוים אך בעצם מה ששגורם לו נמצא במקום אחר ואפשר לעלות על זה רק על ידי דבאגינג ובדיקה לאט לאט , גם משהו שהפרויקט לימד אותי,
אני מקווה שאכן האפליקציה תרוץ כמו שצריך והאתר יהיה כפי שדמיינתי, שיהיה בו שימוש נרחב ומועיל ,הפרויקט הזה נכתב עסהמון השקעה ושעות מרובות מול המחשב ושל מחשבה שהביאו לתוצר הסופי הזה ובתקווה שאכן כל זה היה משתלם, והפרויקט הזה יצליח וימצא חן ויקבל ציון טוב.

22.פיתוח עתידי

אפשרות של שיבוץ מנתחים גם לפי ההתאמה הטובה ביותר ולפי בקשה, שליחת אימייל אוטומטית למנותח בנוגע לתאריך הניתוח ולשינויים וכמובן שליחת לו"ז לרופא המנתח.

23. ביבליוגרפיה

המכלול-

https://www.hamichlol.org.il/%D7%A2%D7%9E%D7%95%D7%93_%D7%A8%D7%90%D7%A9%D7%99

ויקיפדיה-

https://www.google.com/search?q=%D7%95%D7%99%D7%A7%D7%99%D7%A4%D7%93%D7%99%D7%94&oq=%D7%95%D7%99%D7%A7%D7%99%D7%A4%D7%93%D7%99%D7%94&aqs=chrome..69i57j0i512l5j69i61l2.2339j0j4&sourceid=chrome&ie=UTF-8

כללית-

https://www.clalit.co.il/he/Pages/default.aspx

הדסה עין כרם-

/https://www.hadassah.org.il

איכילוב-

https://www.google.com/search?q=%D7%90%D7%99%D7%9B%D7%99%D7%9C%D7%95%D7%91&oq=%D7%90%D7%99%D7%9B%D7%99%D7%9C%D7%95%D7%91&aqs=chrome..69i57j46i67i175i199i433j0i512l8.1839j0j4&sourceid=chrome&ie=UTF-8

-github

/https://github.com

-stckoverflow

/https://stackoverflow.com

-microsoftt visual studio

/https://visualstudio.microsoft.com

-anguler.io

/https://angular.io

-mdbootstrep

/https://mdbootstrap.com

-wix logo maker

https://placeit.net/placeit-online-logo-maker?gclid=CjwKCAjwkMeUBhBuEiwA4hpqEPv01wADMdxtli6ydkw8w-okYcZ2DhCop1J0pFL3xZjNffqxQjAoyRoCjqgQAvD_BwE

ועוד.