# 

**לוגו מכללה:**

תמונה שמכילה טקסט, לבן, גופן, עיצוב

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**שם מכללה:** סמינר בית יעקב החדש אשדוד

**שם סטודנט:** אביגיל ירוחימוביץ

**ת.ז. סטודנט:** 328303920

**שם המנחה:** יעל עמר

**תאריך הגשה:** 17/06/2025

|  |
| --- |
|  |
|  |  |

# תוכן העניינים

[**תוכן העניינים 2**](#_bddcgvavt3xu)

[**1. הצעת פרויקט 4**](#_9b2ectvkoxh3)

[תיאור הפרויקט: 4](#_lo5u4pgb1bbv)

[הבעיה האלגוריתמית: 4](#_mqs9xa9a6dbf)

[עיבוד שפה טבעית (NLP) 5](#_4g0c7bp07fsu)

[ראייה ממוחשבת (computer vision) 6](#_jptbwb5iwrt)

[הצפנה ואבטחת מידע 7](#_h98juburqqb5)

[רקע תיאורטי: 7](#_fzsjgd4dfuv5)

[יתרונות GloVe 12](#_6f7qlwb79lfr)

[אופן העבודה של GloVe 12](#_beaeus3h6hjt)

[ההבדל בין Word2Vec ל-GloVe 12](#_laawu6btndif)

[PyCryptodome 15](#_uqcqzdzf9bvd)

[Cryptography 15](#_9wog5n6m7nbe)

[תיאור פרוטוקולי תקשורת: 16](#_v52pad9e3vqw)

[**2. מבוא 18**](#_fzucojd98nar)

[2.1 הרקע לפרויקט 18](#_ipo7woy60up4)

[2.2 תהליך המחקר 18](#_i8wb66dui34r)

[2.3 סקירת ספרות 19](#_q842wldwja4u)

[2.4 אתגרים מרכזיים 20](#_isvcrm3m2ek2)

[2.4.5 הצגת פתרונות לבעיה שנבחנו במחקר המקדים 21](#_92isnnszj2fa)

[3. מטרות ויעדים 21](#_ec5eme3i2nd2)

[4. אתגרים 22](#_geia2xh91d3x)

[5. מדדי הצלחה למערכת 22](#_p86qaljicla0)

[**6. רקע תיאורטי 23**](#_981hzcehd38a)

[UMLS – Unified Medical Language System 23](#_s3ot4n7uysrp)

[מבנה UMLS 23](#_zbavvxemky8l)

[UMLS Browser 27](#_sqoj0qvu22wa)

[חילוץ סימפטומים 31](#_ni49cqptry36)

[זיהוי ישויות רפואיות חכמות מטקסט באמצעות MedCAT 32](#_hgwqfvoezofr)

[סיווג תמונות רפואיות באמצעות BioMedCLIP 32](#_fzxcufb52oc)

[מסדי נתונים – PostgreSQL ו־Neo4j 33](#_dujoo5oc65hn)

[**7. תיאור מצב קיים 34**](#_5og02r752lcy)

[**8-9. ניתוח חלופות מערכתי והחלופה הנבחרת 35**](#_vvwas01gy8w2)

[חלופות ל-dataset 35](#_iohc3fug3dwj)

[חלופות למודל ניתוח טקסט רפואי 36](#_70cn6tic8v)

[חלופות למודל ניתוח תמונה 37](#_5fdnybnb5ny9)

[חלופות למסדי נתונים 37](#_48mbgmqwyq6c)

[**10. אפיון המערכת 39**](#_qbooxob5sdmp)

[10.1 ניתוח דרישות המערכת 39](#_2rsju4mfqa5)

[10.2מודולי המערכת 39](#_9v0gf3o9arjg)

[10.3 אפיון פונקציונלי 41](#_q2qsfitvlgu2)

[10.4 ביצועים עיקריים 43](#_ovje1c3bd5ca)

[10.5 אילוצים 43](#_qc8ld8ydni85)

[**11. תיאור הארכיטקטורה 43**](#_7ozk957xwp88)

[11.1 ארכיטקטורת הפתרון המוצע – Top-Down Level Design 43](#_ra5f58c0xwaw)

[11.2 תיאור הרכיבים בפתרון 44](#_tsgzjw5g4etw)

[11.3 תהליכי מערכת ההפעלה 44](#_flat8vl2q3t1)

[11.4 ארכיטקטורת רשת 44](#_3wyq1dufoemg)

[11.5 תיאור פרוטוקולי התקשורת 44](#_hd8q3eh2kyxv)

[11.6 תצורת שרת - לקוח 45](#_fe7z9rfp9tgy)

[**12. תהליכי אבטחת מידע במערכת 45**](#_k3z8532dbf3x)

[**13. למידת מכונה 45**](#_68tf6chuuchs)

[**14. ניתוח ותרשימי Use Case / UML 45**](#_k0sjbjorn0jb)

[14.1 תיאור ה-Use Cases העיקריים של המערכת 45](#_gz5jzrffk0dv)

[14.2 הצגת מקרי שימוש (Use Cases) עיקריים 46](#_21ek53p1uxhj)

[14.3 מבני נתונים בהם נעשה שימוש 47](#_tjr4p912n3hb)

[14.4 חישוב יעילות האלגוריתם 50](#_j3yxjrkhzyzy)

[14.5 הקשרים בין היחידות השונות 51](#_bezjwe1lsncj)

[14.6 עץ מודולים 51](#_k3zrhul5ehuz)

[14.7 Use Case Diagram 53](#_673059hinenr)

[14.8 רשימת Use Cases 54](#_bljkhi8vhuo8)

[14.9 תרשים UML - Class Relationships 55](#_smkp1ynn7jby)

[14.10+14.11 Design class Diagram - תרשים מחלקות 55](#_huad0ej2pl2g)

[14.12 תיאור המחלקות המשמעותיות 56](#_862jh4arsn11)

[**15. רכיבי ממשק 61**](#_xlmt14v57wc)

[**16. תיכון המערכת 61**](#_wb6ldk5a5h0)

[16.1 ארכיטקטורת המערכת 61](#_d71ls65vcibq)

[16.2 תיכון מפורט 61](#_q29hr52aetlr)

[16.3 חלופות לתיכון המערכת 62](#_d0gh66v00mic)

[**17. תיאור התוכנה 63**](#_f9jqgeo9a25n)

[17.1 סביבות עבודה 63](#_ycaxc3kflhg7)

[17.2 שפות תכנות 63](#_cgfpgudnpbd8)

[**18-20. תיאור מסכים 64**](#_o8vr2zwl421z)

[**21. תיאור מסך פתיחה 69**](#_hoznrdvhwsul)

[**22-23. מסכי האפליקציה. 69**](#_ewgt3kjyujp5)

[**24. הודעות למשתמש והתרעות 74**](#_dnqy25fq2apw)

[**25. ממשק משתמש 74**](#_fvvprccnmzma)

[**26. קוד התוכנית 74**](#_le12ajkp5zde)

[26.1 קלט (Input) 74](#_fcliqyhu1bfw)

[26.2 פלט (Output) 75](#_uo4jf9so4m94)

[26.3 פונקציות חשובות 77](#_558m46jw5ejf)

[ג. מערכת ניתוח סיכונים דינמית 84](#_hqy4t985y6oc)

[27. תיאור מסד הנתונים 86](#_5meb0evpoz3c)

[**28. מדריך ידידותי למשתמש 88**](#_wy2s784werpv)

[**29. בדיקות והערכה 93**](#_u5mcegy16fq5)

[**30. מסקנות 94**](#_hd1kljkxfyvt)

[**31. פיתוחים עתידיים 94**](#_cgakejrzlcsv)

[**31. ביבליוגרפיה (קישורים פעילים) 95**](#_cj7rb47krm9b)

[UMLS: 95](#_pzx9rknlaml5)

[NLP רפואי 96](#_3xj9741brob)

[BioMedCLIP 96](#_fp8zrez9jotm)

[MedCAT 96](#_6dc2x7xyen2k)

[גרף רפואי: 96](#_l6easaozbko1)

# הצעת פרויקט

### תיאור הפרויקט:

המערכת מיועדת לספק תמיכה ראשונית במצבי חירום רפואיים, ובכך לייעל את ההתמודדות עם מצבים דחופים שדורשים התערבות מיידית. באמצעות אפליקציה חדשנית, המערכת מאפשרת למשתמשים להעלות נתונים רפואיים כגון תוצאות בדיקות דם, תיקי מידע רפואיים, או אפילו לשתף את האפליקציה עם מאגרי הנתונים של קופת החולים שלהם, כדי לאפשר לצוותי הרפואה גישה מיידית ומדויקת למידע רלוונטי.

במהלך מצב חירום, כשהזמן הוא קריטי, האפליקציה מאפשרת למשתמשים או מיופי כוחם להזין תיאורים טקסטואליים של תסמינים או להעלות תמונות המתארות את המצב הרפואי של המטופל. המערכת תנתח את הנתונים שהוזנו, תצליב אותם עם היסטוריה רפואית קיימת ותספק הערכה ראשונית לגבי המצב, כולל הנחיות פעולה מיידיות.

המערכת עושה שימוש בטכנולוגיות של עיבוד שפה טבעית (NLP) כדי לנתח תיאורים טקסטואליים של תסמינים רפואיים וכן ניתוח ועיבוד התיק הרפואי של המשתמש בצורה חכמה ומדויקת. כמו כן, המערכת עושה שימוש בטכנולוגיות של ראייה ממוחשבת, המאפשרות לזהות תסמינים ויזואליים בתמונות שהועלו, כגון שטפי דם, תגובות אלרגיות, פציעות שונות או שינויים אחרים במצב הבריאותי של המטופל. תהליך זה מבצע התאמה אוטומטית של המידע הרפואי על מנת להנחות את המשתמשים או הצוותים הרפואיים בצורה חכמה ויעילה.

יתר על כן, האפליקציה שומרת על פרטיות המידע הרפואי של המשתמשים בצורה מיטבית, באמצעות הצפנות ואמצעי אבטחה, כך שברגע האמת רק הגורמים המורשים יוכלו לגשת למידע ולהשתמש בו. הפתרון מיועד להבטיח זמינות גבוהה של המידע הרפואי הקריטי במצבים דחופים, ולספק מענה מהיר ואיכותי בשלב הראשוני של טיפול חירום, תוך שמירה על פרטיות המשתמשים. המטרה היא להקל את תהליך קבלת ההחלטות הרפואיות, להגדיל את הסיכוי לקבלת טיפול נכון ומדויק, ולסייע להציל חיים עוד לפני הגעת אנשי מקצוע למקום האירוע.

**קלט:** המשתמשים יכולים להזין תיאורים טקסטואליים של תסמינים רפואיים, להעלות תמונות המתארות את מצבם או לשתף נתונים רפואיים קיימים לצורך ניתוח המצב הרפואי.

**פלט:** האפליקציה מספקת הערכה ראשונית של המצב הרפואי והנחיות פעולה מיידיות, בהתאם לניתוח הנתונים שהוזנו.

### הבעיה האלגוריתמית:

המערכת נוגעת בתחומי דעת רבים במדעי המחשב. נחלק את הניתוח שלהם ל-3:

* עיבוד שפה טבעית (NLP)
* ראייה ממוחשבת (computer vision)
* הצפנה ואבטחת מידע

### עיבוד שפה טבעית (NLP)

עיבוד שפה טבעית (Natural Language Processing - NLP) הוא תחום במדעי המחשב העוסק בהבנה, ניתוח ויצירה של שפה אנושית באמצעות מחשבים. מדובר בממשק בין השפה האנושית לבין מערכות מחשוב, שבו מחשבים מנסים להבין ולהגיב לשפה טבעית כמו שהיא בשפת היום-יום. השימוש ב-NLP נמצא בכל תחום בו יש צורך בהבנה אוטומטית של שפה, כולל בתחום הרפואה.

באפליקציה המוצעת, עיבוד שפה טבעית מהווה רכיב מרכזי, שמטרתו להבין תיאורים טקסטואליים של תסמינים רפואיים שהמשתמשים או בני משפחתם כותבים. זיהוי נכון של תסמינים חשוב ביותר, כי הוא מסייע לאפליקציה להפיק הבנה ראשונית על מצב המשתמש ולהכווין אותו לפעולה רפואית מתאימה בזמן אמת, ובכך לספק מענה מיידי במצבי חירום.

השלב הראשון בתהליך הוא להבין את הטקסט הכתוב בצורה אוטומטית, לאסוף ממנו את המידע הרלוונטי ולהפוך אותו למידע שימושי. NLP מאפשרת את כל שלבי העיבוד הללו, ומסייעת לאפליקציה לנתח את התסמינים ולספק ייעוץ רפואי ראשוני או הכוונה לפעולה בזמן החירום.

המערכת מתחילה בניתוח הטקסט ניתוח תחבירי שמאפשר למערכת להבין את מבנה המשפט ולזהות את הקשרים בין המילים. בשלב הבא, מתבצעת הבנה סמנטית שמפענחת את המשמעות המדויקת של התיאורים שניתנו, כמו כאב ממושך או חום גבוה. לבסוף, המערכת מסווגת את התסמינים לפי חומרה ומספקת ייעוץ רפואי ראשוני או המלצות להמשך טיפול, הכל במטרה לעזור במהירות וביעילות.

השילוב של כל שלבי עיבוד השפה הטבעית הללו יאפשר לאפליקציה לא רק להבין את הטקסטים שהמשתמש מזין, אלא גם להפיק מהם מידע שימושי ולקבל החלטות לגבי הצעד הבא במענה רפואי. האפליקציה תוכל לספק תמיכה מיידית במצבים רפואיים, לעזור למשתמשים להבין את מצבם ולהפנות אותם להכוונה רפואית מקצועית בזמן חירום, ובכך תסייע לשפר את איכות השירות הרפואי במצבים קריטיים.

בעיות נפוצות ב-NLP, כמו **מילים רבות משמעות** ו**עמימות**, עשויות להוות אתגר בפרויקט שלנו. לדוגמה, תיאורים כמו "חום" עשויים להיתקל בעמימות, מכיוון ש"חום" יכול להתפרש כטמפרטורת גוף גבוהה, תחושת חום מקומית בעור, או אפילו מצב רגשי (כמו חום אהבה). חשוב שמערכת ה-NLP תדע לפענח את המשמעות המדויקת לפי ההקשר הרפואי שבו הוא מתואר, כמו "חום מעל 38 מעלות במשך שלושה ימים" או "אזור עם תחושת חום ונפיחות".

בנוסף, יש צורך להתמודד עם אתגרים כמו **שגיאות כתיב ודקדוק**, שמיוצרים לעיתים כאשר המשתמשים מזינים תיאורים לא מושלמים. התמודדות עם שגיאות כאלו היא חלק חשוב בעיבוד הטקסט במערכת שלנו, כי שגיאות אלו יכולות להשפיע על דיוק התוצאות.

### ראייה ממוחשבת (computer vision)

מטרת האפליקציה היא לספק מענה מהיר ומדויק למצבי חירום רפואיים, תוך התאמה לצרכים הדחופים של המשתמשים. הוספת האפשרות להעלות תמונה תורמת רבות להשגת מטרה זו, שכן במצבי חירום רבים קשה לאנשים לתאר בצורה מילולית את התסמינים או התופעות שהם חווים. שילוב של עיבוד תמונה מיידי מספק פתרון יעיל המקל על המשתמשים ומאפשר לאפליקציה להפיק הערכה ראשונית מדויקת , גם בשעה בה קשה למשתמש להתנסח.

מה זה computer vision?

ראייה ממוחשבת היא תחום במדעי המחשב שמתמקד בהקניית יכולת למערכות מחשב להבין, לפרש ולעבד תמונות ווידאו באופן דומה לאופן שבו בני אדם עושים זאת. המטרה המרכזית של התחום היא לאפשר למחשב להבין את מה שהוא "רואה" — למשל, לזהות אובייקטים, תופעות או דפוסים בתמונות. בעשורים האחרונים, עם התפתחות טכנולוגיות כמו למידת מכונה וניתוח נתונים, תחום הראייה הממוחשבת צמח בצורה משמעותית ומאפשר יישומים שונים שמסייעים בפתרון בעיות בסביבה רפואית, תעשייתית, ביטחונית ועוד.

במקרה של יישומים רפואיים, ראייה ממוחשבת יכולה לעזור במגוון תחומים, כולל זיהוי בעיות רפואיות בתמונות עור, קרני רנטגן, אולטרסונוגרפיה ועוד. מטרת השימוש בטכנולוגיה כזו היא לאפשר זיהוי מוקדם ודיוק גבוה של תסמינים רפואיים מבלי להסתמך רק על בדיקות פיזיות או אמצעים פחות אובייקטיביים. כך המערכת מציעה יכולת אבחון מרחוק ללא איש רפואה זמין למענה וחוות דעת רפואית.

למשל, זיהוי פריחות או שטפי דם בעור באמצעות מערכת ראייה ממוחשבת יכול להיעשות על ידי ניתוח תמונות של עור באמצעות אלגוריתמים המיועדים לזהות שינויים צבעוניים, תבניות, וגודלים של נגעים בעור. כל זה נעשה בשיתוף פעולה עם מערכות למידת מכונה שמסוגלות לשפר את הדיוק לאורך זמן, ולזהות בעיות באופן מהיר ויעיל. טכנולוגיות כאלו גם עשויות להקל על רופאים בהמלצות על דרכי טיפול, על ידי הצגת תמונות עם הסברים מותאמים אישית.

בשלב הראשון של המערכת, התהליך כולל בדרך כלל עיבוד ראשוני של התמונה, בו המחשב מזהה את החלקים השונים בתמונה, מפריד בין הרקע לאובייקטים רלוונטיים (למשל, הפריחה או שטף הדם), ומבצע ניתוח מדויק של האזורים הללו. לאחר מכן, מתבצע זיהוי של דפוסים או שינויים בולטים, שיכולים להעיד על בעיות רפואיות כמו אלרגיה, זיהום, או מחלה עורית.

דוגמה לתהליך הרצוי של המערכת כשהיא מקבלת כקלט תמונה:  
תמונה של פריחה על העור יכולה להיות מנותחת לפי מאפיינים כמו:

* צבע: האם מדובר בגוון אדום, סגול או אחר?
* צורה: האם היא מפוזרת בצורה לא סדירה או עגולה?
* גודל: האם היא מתפשטת או ממוקדת בנקודה אחת?

המידע הזה משמש את המערכת כדי לספק הערכה ראשונית – האם מדובר באלרגיה, זיהום או מצב אחר.

### הצפנה ואבטחת מידע

המערכת המוצעת פועלת בתחום הרגיש של המידע הרפואי, ולכן יש להקפיד על שמירה קפדנית על פרטיות המידע ויכולתה להתמודד עם אתגרים של אבטחת מידע. כדי להבטיח שהמידע הרפואי של המשתמשים יהיה מוגן ויישמר בצורה מאובטחת, יש לממש אמצעי הצפנה מתקדמים ונהלים שמגנים על המידע הן במעבר והן בזמן האחסון.

המערכת תצטרך להבטיח שהמידע הרפואי, כולל נתונים רגישים כמו תיקי בריאות, תוצאות בדיקות, ותיאורים טקסטואליים של תסמינים, יישמרו מוצפנים גם בזמן השימוש וגם בזמן אחסון בשרתים. הצפנה זו תספק שכבת הגנה חשובה על המידע, כדי למנוע גישה בלתי מורשית אליו.

הצפנה היא תהליך הממיר נתונים קריאים לנתונים שאינם ניתנים להבנה מבלי להשתמש במפתח הצפנה. הצפנה מודרנית מחולקת לשני סוגים עיקריים:

* **הצפנה סימטרית (Symmetric encryption)**: בה המפתח ההצפנה והפענוח הם אותו מפתח. היתרון של הצפנה סימטרית הוא היעילות שלה, אך החיסרון הוא הצורך בהעברת המפתח בצורה מאובטחת.
* **הצפנה א-סימטרית (Asymmetric encryption)**: בה יש שני מפתחות – מפתח ציבורי ומפתח פרטי. המפתח הציבורי יכול לשמש להצפנה, והפרטי לפענוח. זוהי הצפנה אידיאלית עבור המערכת שלך, במיוחד כאשר מדובר בשיתוף מידע בין משתמשים שונים, שכן המפתח הציבורי יכול להיות זמין לכולם, אך רק בעל המפתח הפרטי יכול לפענח את המידע.

### רקע תיאורטי:

בעידן הנוכחי, טכנולוגיות רפואיות משולבות יותר ויותר בחיי היום-יום של אנשים. עם זאת, תחום העזרה הראשונה הדיגיטלית עדיין נותר מאחור במובנים מסוימים, בעיקר במצבים שבהם זמן התגובה הוא קריטי ואין גישה מיידית למידע רפואי רלוונטי.

במצבי חירום רפואיים, כמו התקפי לב, תגובות אלרגיות חמורות, או פציעות טראומטיות, מידע רפואי עדכני עשוי להיות ההבדל בין חיים למוות. תיקים רפואיים, בדיקות דם, ותיעוד היסטוריה רפואית לעיתים קרובות אינם זמינים בזמן אמת עבור אנשי מקצוע או אנשים קרובים אחרים המבקשים לעזור. יתרה מכך, יכולת האדם לתאר במדויק את מצבו במצבי לחץ מוגבלת, מה שעלול להוביל לאי-דיוק באבחון המיידי.

מחקרים מראים כי בכל דקה שעוברת במצבי חירום כמו התקפי לב או שבץ, הסיכוי להישרדות יורד בכ-10%. למשל, בישראל מדווחים מדי שנה על אלפי מקרי חירום שבהם אבחון בזמן והנחיות עזרה ראשונה יכולים לשנות משמעותית את תוצאות האירוע.

**חשיבות המידע המיידי:**

* **אבחון נכון ומהיר:**במצבים דחופים, יכולת לאסוף, לנתח ולהצליב מידע רפואי בזמן אמת מאפשרת לזהות סיבות אפשריות לתסמינים ולתת מענה מתאים במהירות. לדוגמה, ניתוח של בדיקות דם אחרונות יכול להצביע על בעיות כמו אנמיה, רמת סוכר חריגה או חוסר איזון אלקטרוליטי, שכולם עשויים להשפיע ישירות על סוג הטיפול הנדרש.
* **היסטוריה רפואית רלוונטית:**במקרים רבים, מטופל אינו מודע או שאינו מסוגל לתקשר את היסטוריית המחלות שלו במצבי לחץ. נתונים מיידיים כמו רגישות לתרופות, אלרגיות או מחלות כרוניות יכולים למנוע טיפולים שגויים או מסוכנים.

**מהם הפערים המרכזיים?**

1. **נגישות למידע רפואי בזמן אמת:** מערכות הבריאות הנוכחיות שומרות לעיתים קרובות מידע רפואי קריטי כמו היסטוריה רפואית, בדיקות דם, ותיעוד טיפולים במאגרי מידע פנימיים. חוסר הנגישות של נתונים אלו לאנשים פרטיים ולמיופי כוחם בזמן חירום מוביל לכך שהחלטות טיפוליות מבוססות על מידע חסר. גם מגישי שירות רפואי כמו אחים או פרמדיקים עשויים להיתקל בקשיים לקבלת התיק הרפואי של המטופל, מה שעשוי להיות הרסני במיוחד. לדוגמה, מחקר מראה כי שימוש במערכות המאפשרות שיתוף נתונים בזמן אמת בין גורמי חירום רפואיים (כמו SAFR בקליפורניה) שיפר את זמינות המידע ואת איכות ההחלטות הרפואיות.

באמצעות מערכות דיגיטליות המאפשרות גישה למידע מותאם אישית, לא רק שצוותי חירום יוכלו לקבל החלטות מושכלות יותר, אלא גם המטופלים ובני משפחותיהם יוכלו להרגיש בטוחים יותר במצבי לחץ. לדוגמה, באפליקציות בריאות קיימות בארה"ב, 82% מהמשתמשים דיווחו שהשימוש בטכנולוגיה העלה את תחושת הביטחון שלהם בקבלת החלטות רפואיות.

1. **היעדר ניתוח מיידי של נתונים:** אפילו כאשר המידע זמין, אין פתרונות מספקים לניתוחו באופן מיידי. טכנולוגיות כמו עיבוד שפה טבעית (NLP) וראייה ממוחשבת (Computer Vision) יכולות לזהות אנומליות רפואיות בתיאורים טקסטואליים ובתמונות, כמו תגובות אלרגיות או שטפי דם. פתרונות כאלו נמצאים בשלבי פיתוח ראשוניים, אך היישום שלהם לשימוש יום-יומי עדיין מוגבל.
2. **חוסר הדרכה מעשית:** במצבים קריטיים, אנשים רבים נמנעים ממתן סיוע מחשש לטעויות. מחקרים מראים ש-59% מהאנשים אינם מגיבים לאירועי חירום בגלל היעדר ידע או פחד מכישלון ( National Library of Medicine). אפליקציה המספקת הנחיות ברורות ופשוטות לשימוש יכולה לשפר את המענה במצבי חירום.

שילוב של טכנולוגיות מתקדמות כמו בינה מלאכותית (AI), עיבוד שפה טבעית (NLP), וראייה ממוחשבת מאפשר פיתוח מערכות שמסוגלות להעריך מצבים רפואיים באופן מיידי ומדויק. מערכת כזו יכולה לגשר על הפערים הקיימים, להנגיש מידע רפואי קריטי בזמן אמת, ולספק הנחיות מצילות חיים בהתבסס על נתונים אישיים ועדכניים של המשתמש.

**המערכות הקיימות כיום:**

המערכות הקיימות כיום בתחום הבריאות הדיגיטלית מציעות פתרונות מעניינים, אך יש להן עדיין מגבלות קריטיות במצבי חירום רפואיים, שבהם כל שנייה עשויה לשנות את התוצאה. אפליקציות כמו **K Health**, **Babylon Health**, **Symptomate**, ו-**Doc.ai** מהוות התקדמות משמעותית בתחום הבריאות הדיגיטלית, אך כל אחת מהן נתקלת באתגרים שגורמים להן לא להיות פתרון אופטימלי במצבים קריטיים.

* **K Health** מספקת חוות דעת ראשונית מבוססת AI תוך שימוש בטכנולוגיות של עיבוד שפה טבעית (NLP) שמנתחות תיאורים טקסטואליים של תסמינים. המערכת מתמקדת בעיקר בזיהוי תסמינים ושאלות טקסטואליות כדי להציע אבחנה, אך אין לה את היכולת לנתח בזמן אמת תמונות רפואיות או נתונים חיוניים אחרים כמו בדיקות דם. במצבים שבהם נדרשת החלטה רפואית מיידית, חוסר האינטגרציה עם מאגרי מידע רפואיים בזמן אמת, כמו תיקי מידע רפואי או תוצאות בדיקות אחרונות, מהווה חיסרון משמעותי. למשל, במצבי חירום כמו התקפי לב או תגובות אלרגיות חריפות, אין אפשרות להצליב את המידע עם היסטוריה רפואית קיימת ולספק ייעוץ מיידי על סמך נתונים מעודכנים.
* **Babylon Health** מציעה ייעוץ רפואי מבוסס AI ומאפשרת גישה לשירותי רופאים וירטואליים, אך גם היא לא מספקת פתרון בזמן אמת למצבי חירום. המערכת מתמקדת בהבנת התסמינים דרך שאלונים טקסטואליים והצעות טיפול בסיסיות, אך אין לה את היכולת לבצע ניתוח נתונים מיידי כמו תוצאות בדיקות דם או תמונות רפואיות שנשלחו על ידי המשתמש. במצבים קריטיים, כאשר יש צורך בהנחות טיפוליות על בסיס נתונים עדכניים, היעדר היכולת לזהות נתונים כאלה בזמן אמת הופך את המערכת לפחות אפקטיבית במצבים דחופים.
* **Symptomate** משתמשת בטכנולוגיה מתקדמת לזיהוי תסמינים בעזרת שאלות דינמיות המותאמות לתשובות שהמשתמש נותן, אך המערכת לא מציעה ניתוח בזמן אמת של נתונים רפואיים קריטיים, כמו תוצאות בדיקות דם או תמונות רפואיות. בעיה זו הופכת את המערכת לפחות יעילה כאשר מדובר במצבים שבהם הזמן הוא קריטי ויש צורך בתגובה מהירה ומדויקת. המערכת אינה מציעה הנחיות מיידיות המותאמות אישית למצב, אלא רק המלצות כלליות, דבר שיכול להיות מסוכן במקרה של בעיות רפואיות חמורות.
* **Doc.ai** מאפשרת ניתוח נתונים רפואיים ושיתוף מידע עם רופאים, אך גם היא אינה מציעה פתרונות שמנתחים תמונות רפואיות בזמן אמת, כמו תמונות של שטפי דם, פצעים או תגובות אלרגיות. מערכת כזו עשויה להחמיץ תסמינים קריטיים המובילים להבחנה לא נכונה ולחוסר תגובה במצבים בהם יש צורך בהחלטה מיידית. בנוסף, לא כל המערכות הללו מצליחות לשלב בצורה יעילה נתונים רפואיים אישיים, דבר שיכול להוביל להחלטות שגויות בזמן חירום.

המערכת המוצעת מציעה פתרון קריטי למספר בעיות קיימות במערכות הבריאות הדיגיטלית הנוכחיות. באמצעות שילוב של **עיבוד שפה טבעית (NLP)** ו-**ראייה ממוחשבת**, המערכת אינה רק מספקת חוות דעת ראשונית מדויקת על סמך תיאורים טקסטואליים של תסמינים, אלא גם מבצעת **ניתוח בזמן אמת של תמונות רפואיות**, דבר שמאפשר לאתר תסמינים כמו שטפי דם, תגובות אלרגיות או פציעות טראומטיות באופן מיידי. יתרה מכך, המערכת משלבת **נתונים רפואיים עדכניים** כמו תוצאות בדיקות דם והיסטוריה רפואית קיימת של המשתמש, ומספקת הנחיות מצילות חיים בהתבסס על מידע אישי, דבר שמאפשר תגובה רפואית מהירה ומדויקת בס"ד.

המערכת, כאמור עשויה להממשק עם 3 תחומים עיקריים במדעי המחשב: NLP, computer vision והצפנות.

**אלגוריתמים לעיבוד שפה טבעית - NLP:**

* **BERT**

Bidirectional Encoder Representations from Transformers) BERT) הוא אחד האלגוריתמים המובילים בתחום עיבוד שפה טבעית (NLP), והוא יכול לשמש במערכת כדי לנתח תיאורים טקסטואליים של תסמינים רפואיים באופן מדויק ומעמיק.

BERT הוא מודל שמבוסס על רשתות נוירונים מסוג Transformer, שפותח על ידי חוקרי גוגל. ייחודו בכך שהוא מבצע ניתוח דו-כיווני של טקסט, כלומר, הוא לוקח בחשבון הן את ההקשר שמגיע לפני מילה והן את ההקשר שמגיע אחריה, כדי להבין בצורה טובה יותר את המשמעות המדויקת של המילים בטקסט.

מה זה Transformer?

**Transformer** הוא מודל למידת מכונה בתחום עיבוד השפה הטבעית (NLP), שנוצר כדי לשפר את הביצועים של מודלים קודמים כמו RNN ו-LSTM. המודל מבוסס על מנגנון שנקרא Attention**,** שמאפשר לו להתמקד במילים רלוונטיות בטקסט ולהבין את ההקשרים ביניהן, ללא תלות בסדר שבו הן מופיעות. בניגוד למודלים קודמים שמעבדים את הטקסט בסדר לינארי, ה-Transformer מעבד את כל המילים בו זמנית, מה שמאפשר לו לעבוד בצורה הרבה יותר יעילה ומהירה. המודל מחולק לשני חלקים: **Encoder** (מקודד) שמבין את הטקסט, ו-**Decoder** (מפענח) שמייצר את הפלט. בזכות היכולת שלו להבין קשרים מורכבים בין מילים ולבצע עיבוד מקבילי, ה-Transformer הפך לאבן יסוד במודלים מתקדמים כמו BERT ו-GPT, והביא לפריצת דרך במגוון משימות עיבוד שפה כמו תרגום, סיווג טקסט, וסינתזת טקסט.

איך BERT עובד?

1. אימון דו-כיווני: בניגוד למודלים כמו Word2Vec או GloVe, שמתמקדים בהקשר חד-כיווני (קדימה או אחורה), BERT משתמש במבנה דו-כיווני. כך, הוא מסוגל להבין את המשמעות של מילה בהתאם להקשר הרחב שלה בטקסט.
2. משימות האימון המרכזיות:

Masking (מילוי מילים חסרות): במהלך האימון, חלק מהמילים במשפט מוסתרות, והמודל צריך לחזות אותן על בסיס ההקשר. זה משפר את יכולתו להבין תבניות סמנטיות מורכבות.

Next Sentence Prediction (ניבוי משפט הבא): המודל מאומן לזהות האם משפט מסוים מגיע מיד אחרי משפט אחר, מה שמחזק את הבנת ההקשר בין משפטים שונים.

1. Tokenization (פירוק טקסט): BERT משתמש בטוקניזציה מסוג Word Piece, שמפצל מילים לתת-יחידות (subwords), כך שגם מילים נדירות או לא מוכרות יקבלו ייצוג סביר.

**BioBERT** הוא גרסה מותאמת של מודל BERT, שנועדה במיוחד לעיבוד טקסטים רפואיים. המודל מאומן על נתונים רפואיים כמו מאמרים ב-PubMed והערות קליניות (Clinical Notes), מה שמאפשר לו להבין טוב יותר את המונחים הרפואיים ואת ההקשרים הסמנטיים הייחודיים לשפה הרפואית. BioBERT מציע ביצועים משופרים על משימות כמו זיהוי ישויות רפואיות, סיווג טקסטים רפואיים והבנת תיאורים רפואיים מורכבים.

BERT מתאים לפרויקט בזכות יכולתו להבין ולהתמקד בהקשרים רפואיים מורכבים, כמו תיאורים של תסמינים רפואיים וזיהוי ישויות רפואיות. יחד עם זאת, הוא דורש משאבים חישוביים גבוהים ואימון מותאם למאגר נתונים רפואי, מה שיכול להוות אתגר במצבי חירום בזמן אמת. יתרון משמעותי נוסף הוא BioBERT שמותאם במיוחד למושגים רפואיים.

* **Word2Vec**אלגוריתם בתחום עיבוד שפה טבעית (NLP) שמטרתו לייצר ייצוגים וקטוריים למילים במרחב רב-ממדי. אלגוריתם זה ממפה מילים לווקטורים מספריים, כך שמילים בעלות משמעות דומה ממוקמות קרוב זו לזו במרחב הווקטורי.

**סוגי המודלים של Word2Vec:**ל-Word2Vec יש שני סוגים עיקריים של מודלים:

**:Continuous Bag of Words (CBOW).1**במודל זה, האלגוריתם מנבא מילה על סמך ההקשר שלה. היתרון שלו הוא יעילות ומהירות, במיוחד באימון על נתונים קטנים.

**Skip-gram.2:**מודל זה פועל באופן הפוך - מנבא הקשרים על בסיס מילה מרכזית. הוא מתאים במיוחד לעבודה עם נתונים גדולים ומסוגל לזהות הקשרים נדירים בין מילים.

**כיצד Word2Vec פועל?**התהליך מתחיל בשלב ה-**Preprocessing**, שבו מנקים ומכינים את הטקסט לעיבוד.  
בשלב הבא, יוצרים זוגות אימון שמורכבים ממילות מטרה והקשר.  
לבסוף, מאמנים את המודל באמצעות רשת נוירונים קטנה. במהלך האימון, מתעדכנים משקלי הרשת כדי להפחית את שגיאת הניבוי. לאחר מכן, הווקטורים המייצגים את המילים מופקים מהשכבה הנסתרת של הרשת.

**יתרונות Word2Vec:**

* יכולת ליצור הקשרים סמנטיים בין מילים, כך שמילים דומות יהיו קרובות במרחב הווקטורי.
* מהיר ויעיל במיוחד באימון על כמויות נתונים גדולות.
* פשוט יחסית להבנה ולשימוש.

**יישומים של Word2Vec:**השימושים כוללים מציאת מילים נרדפות, ניבוי מילים והשלמת טקסט, ושיפור מנועי חיפוש. אלגוריתם זה מסייע לשפר את ההבנה של שפה טבעית ואת תהליכי עיבוד הטקסט.

* **GloVe** מודל מתקדם ללמידת ייצוגים וקטוריים של מילים, שתכליתו לשמר את המשמעות והקשרים הסמנטיים בין מילים. המודל מבוסס על מטריצת הופעות משתופות המייצגת את תדירות המילים שמופיעות יחד בקורפוס. באמצעות פונקציית מטרה, GloVe מתאים את הוקטורים כך שהמכפלה הפנימית שלהם תתאים ליחסי ההופעה במטריצה, תוך התחשבות בתדירות המילים.

### יתרונות GloVe

* שמירה על יחסים מספריים בין מילים. לדוגמה, ההבדל בין "מלך" ל"מלכה" מדמה את ההבדל בין "איש" ל"אישה".
* יעילות ופשטות בשימוש במשימות שונות בעיבוד שפה טבעית כמו תרגום מכונה וניתוח טקסטים.

### אופן העבודה של GloVe

1. איסוף קורפוס.
2. יצירת מטריצת הופעות משותפות.
3. ביצוע אופטימיזציה לוקטורים על בסיס מטריצה זו.

### ההבדל בין Word2Vec ל-GloVe

* **Word2Vec** מתמקד ביצירת קשרים בהקשרים מיידיים בין מילים ומבוסס על קרבת מילים בטקסט.
* **GloVe** מתחשב בתמונה הכוללת של הופעות משותפות בקורפוס, ומשמר קשרים סמנטיים לאורך זמן ובמגוון רחב יותר של נתונים.

**אלגוריתמים בתחום עיבוד התמונה:**

* **CNN**

רשתות נוירונים קונבולוציוניות (Convolutional Neural Networks, או בקיצור CNN) הן טכנולוגיה פורצת דרך בתחום הראייה הממוחשבת. מדובר במערכות מתקדמות המדמות את פעולת המוח האנושי בניתוח ועיבוד תמונות. הייחוד של רשתות אלו הוא היכולת שלהן לזהות תבניות מורכבות בתמונות באופן אוטומטי, על ידי שימוש במבנה היררכי שבו תכונות פשוטות מובילות לזיהוי תכונות מורכבות יותר.

באמצעות שכבות קונבולוציה ייחודיות, הרשתות מתמקדות באזורים קטנים בתמונה ומאתרות בהם תכונות בולטות כמו צבע, צורה, ודפוסים. לאחר מכן, הן מעבדות את המידע שנאסף כדי להפיק תובנות מדויקות. זהו תהליך קריטי במיוחד עבור אפליקציות רפואיות שבהן דיוק ומהירות הם גורמים מכריעים.

רשת CNN בנויה מכמה שכבות שעובדות יחד:

1. **שכבת הקונבולוציה:** השכבה הראשונה לוקחת את התמונה ומחפשת תבניות בסיסיות – נגיד, קו אדום או מרקם ייחודי.
2. **שכבת הצמצום (Pooling):** שכבה זו מקטינה את כמות המידע שהמערכת צריכה לנתח, תוך שמירה על הדברים החשובים באמת.
3. **שכבת הסיווג:** אחרי שהמידע עובר עיבוד, המערכת מחליטה – מה אנחנו רואים בתמונה?

מלבד הדיוק, אחד היתרונות המרכזיים של רשתות אלו הוא ההתאמה לזמן אמת. במצבי חירום רפואיים, כאשר למטופל קשה לתאר את הסימפטומים במילים, האפשרות להעלות תמונה ולקבל תובנה מיידית עשויה להציל חיים. האפליקציה תוכל לזהות מיידית תסמינים בעייתיים ולהמליץ על פעולות דחופות, כמו פנייה למיון או התייעצות עם רופא מומחה.

בנוסף, רשתות CNN מציעות יכולת למידה מתמשכת. ככל שהאפליקציה תנתח יותר תמונות, כך היא תלמד ותשתפר, מה שיאפשר לה לזהות מגוון רחב יותר של תסמינים ולספק מענה מדויק עוד יותר. זו לא רק מערכת אבחון, אלא כלי שתורם לשיפור מתמיד של שירותי הרפואה ולסיוע באבחון מוקדם של מחלות.

* **U-Net**

U-Net היא רשת נוירונים ייחודית שפותחה במיוחד לצורך עיבוד תמונות, בעיקר בתחומי הרפואה. הרשת נבנתה במקור עבור משימות של סגמנטציה (חלוקת תמונה לאזורים) – כלומר, זיהוי מדויק של אזורים שונים בתמונה, כמו רקמות, איברים או תופעות חריגות. בניגוד לרשתות אחרות שמתמקדות בזיהוי אובייקטים כלליים, U-Net מותאמת לעבוד עם תמונות מורכבות שבהן נדרש דיוק גבוה, כמו תמונות רפואיות.

המבנה של U-Net דומה לאות "U" (ומכאן שמה), והוא מורכב משני שלבים עיקריים:

1. **שלב ה"דחיסה" (Encoder):** בשלב הזה, הרשת מנתחת את התמונה ומצמצמת את המידע, תוך זיהוי מאפיינים חשובים כמו צבעים, מרקמים ותבניות. זהו שלב שבו המידע הופך לתמציתי יותר ומובנה.
2. **שלב ה"שחזור" (Decoder):** בשלב הזה, הרשת משחזרת את המידע המכווץ וממפה אותו בחזרה לתמונה בגודל המקורי, תוך זיהוי מדויק של האזורים הרלוונטיים.

מה שמייחד את U-Net הוא השימוש בחיבורים מיוחדים (skip connections) בין שלב הדחיסה לשלב השחזור. חיבורים אלה שומרים על מידע חיוני (כמו גבולות ברורים או פרטים עדינים) שאחרת עלול ללכת לאיבוד.

**יתרונות U-Net במערכות רפואיות**

U-Net הפכה לכלי מוביל בתחום הרפואה בזכות מספר יתרונות מרכזיים:

1. **דיוק גבוה:** היכולת לזהות גבולות מדויקים בין אזורים בתמונה הופכת אותה לאידיאלית למשימות כמו זיהוי פריחות, שטפי דם או שינויים עוריים.
2. **התאמה לתמונות רפואיות:** רשתות U-Net מותאמות לתמונות ברזולוציה גבוהה, מה שמאפשר עבודה עם תמונות עור, צילומי רנטגן או MRI.
3. **עבודה עם נתונים מוגבלים:** בשונה מרשתות אחרות, U-Net מסוגלת לספק תוצאות מצוינות גם כאשר יש מעט נתוני אימון, וזה יתרון עצום בתחום הרפואה שבו הדאטה מוגבל ויקר.
4. **סגמנטציה מדויקת:** U-Net לא רק אומרת "יש פריחה בתמונה", אלא גם מסמנת בדיוק היכן היא נמצאת ובאיזה גודל.

U-Net היא אחת מהטכנולוגיות המובילות כיום בתחום עיבוד התמונה הרפואית, ובזכות יכולות הסגמנטציה שלה, היא מאפשרת זיהוי מדויק וממוקד של בעיות רפואיות בתמונות. השילוב שלה במערכת כמו זו המוצעת, שמיועדת לספק עזרה רפואית בזמן אמת, יכול לשדרג את תהליך האבחון, לאפשר ניתוח תמונות באופן מיידי, ולספק למטופלים מענה מהיר ומדויק במצבי חירום.

* **YOLO**

(YOLO (You Only Look Once הוא אלגוריתם מתקדם לזיהוי אובייקטים, שמאפשר למחשבים לנתח תמונות בזמן אמת ולזהות אזורים רלוונטיים במהירות ובדיוק גבוה. ייחודו של YOLO טמון בגישה החדשנית שלו – במקום לבחון את התמונה אזור אחרי אזור (כפי שעושים אלגוריתמים מסורתיים יותר), YOLO מנתח את כל התמונה בפעם אחת, מה שמוביל לזיהוי מהיר ואפקטיבי.

אלגוריתם YOLO מחלק את התמונה לרשת (grid) של תאים. כל תא אחראי להערכת האם קיים אובייקט מרכזי בתוכו ואם כן – מהו אותו אובייקט. האלגוריתם מייצר "קופסאות גבול" (bounding boxes) סביב אובייקטים שנמצאו רלוונטיים, ומסווג אותם לפי הסוג (למשל, פריחה, שטף דם, או נגע עור אחר).

בניגוד לשיטות קודמות שמסתמכות על מספר שלבים נפרדים כמו הצעת אזורים (region proposals) וסיווג, YOLO מבצע את כל המשימות האלו באופן משולב. התוצאה היא אלגוריתם שמספק תגובה מיידית, מהירה, ויעילה מאוד.

השילוב של YOLO במערכת יכול להקפיץ את היכולת לזיהוי בעיות עוריות בזכות המהירות והדיוק שלו. היכולת לנתח תמונות בצורה רחבה, הופכת את YOLO לכלי עוצמתי ורלוונטי במיוחד למערכות רפואיות מתקדמות.

**אלגוריתמים להצפנה:**

* **AES -** Advanced Encryption Standard

**תיאור ומנגנון פעולה:**AES הוא אלגוריתם הצפנה סימטרי העובד על נתונים בגושים (blocks) בגודל 128 ביט. תהליך ההצפנה כולל סדרת פעולות מחזוריות הנקראות סבבים (rounds), שמספרם תלוי באורך המפתח: 128, 192, או 256 ביט. כל סבב כולל פעולות ערבוב והחלפת מפתחות, מה שמבטיח רמת אבטחה גבוהה. האלגוריתם ידוע ביעילותו מבחינת ביצועים, במיוחד כאשר נעשה שימוש בתמיכה בחומרה (כגון מעבדי Intel ו-AMD).

* **RSA -**Rivest–Shamir–Adleman

**תיאור ומנגנון פעולה:**RSA הוא אלגוריתם הצפנה אסימטרי המבוסס על הקושי לפרק מכפלה של שני מספרים ראשוניים גדולים. האלגוריתם עושה שימוש בזוג מפתחות:

* **מפתח ציבורי (public key):** משמש להצפנת מידע.
* **מפתח פרטי (private key):** משמש לפענוח המידע.  
  RSA מתאים במיוחד להצפנת נתונים קצרים, כמו מפתחות של AES, ומהווה בסיס להעברת מפתחות מאובטחת וחתימות דיגיטליות.
* **ECC -** Elliptic Curve Cryptography

**תיאור ומנגנון פעולה:**ECC הוא אלגוריתם המבוסס על אלגברה של עקומים אליפטיים, המאפשר רמת אבטחה גבוהה באמצעות מפתחות קצרים בהרבה מאלה של RSA. לדוגמה, מפתח ECC באורך 256 ביט מספק רמת אבטחה שוות ערך למפתח RSA באורך 3072 ביט.  
השימושים המרכזיים של ECC כוללים:

1. **ECDH:** לחילופי מפתחות מאובטחים.
2. **ECDSA:** ליצירת חתימות דיגיטליות.  
   ECC

מתאים במיוחד למכשירים מוגבלים במשאבים, כמו סמארטפונים ואפליקציות מובייל.

* **TLS -**Transport Layer Security

**תיאור ומנגנון פעולה:**TLS הוא פרוטוקול המשלב הצפנה סימטרית (AES) והצפנה אסימטרית (RSA או ECC) על מנת להבטיח תקשורת מאובטחת. TLS מספק:

* **אבטחת העברת נתונים:** שמירה על סודיות ושלמות הנתונים המועברים בין אפליקציה לשרת.
* **פרטיות בתקשורת בזמן אמת (real-time):** קריטי במיוחד עבור מערכות רפואיות שבהן מועברים נתונים רגישים.

**ספריות המתאימות להצפנה:**

### PyCryptodome

PyCryptodome היא ספריית הצפנה חופשית בפייתון, המספקת כלים להצפנה סימטרית ואסימטרית. היא תומכת במגוון רחב של אלגוריתמים כמו AES, RSA, DES ו-3DES, ומאפשרת למפתחי תוכנה להטמיע פתרונות אבטחה חזקים ויעילים. הספרייה גם כוללת כלים נוספים כמו חתימה דיגיטלית, יצירת מפתחות והדרכה למימוש אלגוריתמים מתקדמים כמו HMAC ו-DH. היתרון הגדול של PyCryptodome הוא בכך שהיא מאוד קלה לשימוש ומתאימה למפתחים שרוצים לשלב הצפנה באפליקציות שלהם מבלי להסתבך עם ספריות מורכבות. למרות שהיא תומכת במגוון רחב של אלגוריתמים, חסרונה המרכזי הוא שהיא לא תומכת בגרסאות ישנות של Python, כלומר יש צורך להשתמש בגרסאות חדשות יותר של השפה. בנוסף, הספרייה דורשת התקנה חיצונית, דבר שעלול להיות מאתגר לפיתוחים שמיועדים לפלטפורמות בהן לא ניתן להתקין תוספים חיצוניים.

### Cryptography

ספריית Cryptography היא פתרון הצפנה מודרני ומאובטח לפייתון, המיועדת למפתחים הזקוקים לכלים חזקים אך גמישים עבור הצפנה סימטרית, אסימטרית ו-hashing. הספרייה תומכת במגוון אלגוריתמים כמו AES, RSA, ECC (עקומות אליפטיות) ו-SHA, ומספקת תמיכה מלאה בעבודה עם SSL/TLS לצורך אבטחת תקשורת. Cryptography מתמקדת בהפשטת השימוש באלגוריתמים הצפנה מתקדמים, תוך שמירה על אבטחה גבוהה, מה שהופך אותה לפתרון מצוין עבור פיתוחי תוכנה מאובטחים. יתרונה העיקרי הוא שהיא תומכת בפרקטיקות הצפנה מודרניות, מה שמבטיח רמת אבטחה גבוהה מאוד. כמו כן, הספרייה היא קוד פתוח, כך שניתן להתאים אותה לצרכים אישיים או תרומות קהילתיות. אולם, חסרונה הוא שמפתחים חדשים עלולים למצוא את הספרייה מסובכת לשימוש, במיוחד בהשוואה לספריות פשוטות יותר. זה עשוי לדרוש זמן למידה נוסף למי שאין לו רקע קודם בהצפנה.

**הליכים עיקריים בפתרון בעיה בטכנולוגיות הנדסה מתקדמות:**

* אפליקציה לייעול מצבי חירום רפואיים על ידי ניתוח נתונים רפואיים בזמן אמת.
* המשתמשים מעלים בדיקות, תיקי מידע רפואיים, או תיאורי תסמינים טקסטואליים ותמונות.
* המערכת משתמשת ב-NLP לניתוח תיאורים טקסטואליים ובמחשב ראייה לניתוח תמונות.
* שילוב נתונים עם היסטוריה רפואית ליצירת הערכה ראשונית של מצב המטופל.
* האפליקציה מספקת הנחיות פעולה מיידיות ותומכת בהחלטות רפואיות מהירות.

**הליכים עיקריים בתחום למידת מכונה:**

* עיבוד שפה טבעית.
* ניתוח תמונה באמצעות מודל.

**הליכים עיקריים בתחום מערכות הפעלה:**

לא קיימת התייחסות בפרויקט

**הליכים עיקריים בתחום רשתות מחשבים / תקשורת נתונים:**

לא קיימת התייחסות בפרויקט

**הליכים עיקריים בתחום אבטחת מידע:**

שימוש באלגוריתמי הצפנה ע"מ להבטיח את פרטיות המשתמשים.

### תיאור פרוטוקולי תקשורת:

בפרויקט זה נעשה שימוש בפרוטוקולים מאובטחים כגון:

* HTTPS: להעברת מידע בצורה מאובטחת ומוצפנת בין המשתמשים לשרתים.
* REST API: לגישה לממשקים דרך פרוטוקולי HTTP, המספקים תקשורת בין רכיבים שונים במערכת.
* WebSocket: במקרה של צורך בעדכון בזמן אמת, ניתן להשתמש בפרוטוקול זה לשמירה על תקשורת דו-כיוונית.

**פיתוחים עתידיים:**

בעתיד, האפליקציה צפויה להתפתח לכיוונים מתקדמים שיגדילו את ערכה ואת יכולותיה. בין היתר, מתוכננת אינטגרציה עם מכשירים לבישים ושעונים חכמים לצורך ניטור מתמשך של מדדים רפואיים, לצד הרחבת יכולות ניתוח התמונות לזיהוי בעיות רפואיות מורכבות יותר, כמו פגיעות עוריות ושינויים פיזיולוגיים עדינים. בנוסף, תתווסף תמיכה רב-לשונית שתאפשר שימוש במגוון רחב של שפות, ותשולב מערכת ייעוץ רפואי בזמן אמת לחיבור עם צוותים רפואיים. פיתוחים נוספים יכללו שיפור מודלים מבוססי AI לאבחון מדויק יותר, חיבור לשירותי חירום מקומיים לשיתוף מיידי של מידע חיוני, ותוספת מודול הדרכה להעלאת מודעות הציבור להתנהלות במצבי חירום. לבסוף, יתווספו אפשרויות לשיתוף נתונים אנונימיים לצרכי מחקר רפואי, מה שיתרום לקידום המדע והרפואה.

**תיאור טכנולוגיה הנדסה:**

**צד הלקוח (Client-Side):** המשתמש מזין בתחילת השימוש את התיק הרפואי שלו או נותן הרשאת שיתוף לקופת החולים. בעת חירום, המשתמש יזין תיאורים טקסטואליים או תמונות המתארות את התסמינים ממנו הוא סובל. המשתמש יקבל הנחיות מותאמות למצבו הרפואי.

**צד השרת (Server-Side):** השרת מקבל את התיאורים מהאפליקציה, מנתח אותם בעזרת ניתוח שפה טבעית או ניתוח תמונה, מעבד את הנתונים לכדי הנחיות מותאמות למשתמש ושולח את התוצאות.

**מסד נתונים:**

הפרויקט יכלול מסד נתונים לניהול ואחסון נתוני משתמשים, היסטוריה רפואית, קבצי תמונות ותוצאות ניתוחים. המסד יתמוך במבנה רלציוני לנתונים מובנים ובאחסון נתונים לא מובנים במידת הצורך.

**לוחות זמנים:**

חקר המצב הקיים-ספטמבר

הגדרת הדרישות-ספטמבר

אפיון המערכת-אוקטובר

אפיון בסיס נתונים-נובמבר

עיצוב המערכת-דצמבר

בניית תוכנה-ינואר, פברואר

בדיקות-מרץ

הכנת תיק פרויקט-אפריל

הטמעת המערכת-מאי

הגשת פרויקט סופי-מאי

**מנחת פרויקט:**

תמונה שמכילה גופן, קליגרפיה, לבן, טיפוגרפיה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.****

**חתימת הסטודנט :**

****

**חתימת רכז המגמה :**



## 2. מבוא

### 2.1 הרקע לפרויקט

בשנים האחרונות חלה עלייה משמעותית בזמינות הטכנולוגיות הדיגיטליות בשירותי הבריאות, אך יחד עם זאת, במצבי חירום רפואיים רבים – הן בזירה הציבורית והן בזירה הביתית – קיימת פער משמעותי בין התרחשות האירוע לבין קבלת סיוע רפואי מקצועי. פער זה, אשר נובע לעיתים מהיעדר מידע רפואי זמין, מחוסר יכולת של הנפגע או של הסובבים אותו להעריך את חומרת המצב, או מפערי תקשורת מול גורמי רפואה, עלול להוביל להחמרה מיותרת במצב הבריאותי ואף לאובדן חיים.

בניסיון להתמודד עם בעיה זו, פותח פרויקט Mediaid – מערכת חכמה אשר מאפשרת למשתמשים להזין או להעלות מידע רפואי ראשוני (כגון בדיקות מעבדה, תיאורי תסמינים, ותמונות קליניות), ומבצעת ניתוח ראשוני של הנתונים באמצעות עיבוד שפה טבעית (NLP), ראיית מחשב (Computer Vision) וטכנולוגיות למידת מכונה (Machine Learning). המערכת מספקת למשתמש הערכת מצב מיידית והמלצות לפעולה, תוך הקפדה על שמירה קפדנית של פרטיות המידע באמצעות אמצעי אבטחה מתקדמים.

הצורך שזוהה הינו צורך אוניברסלי: בכל מצב בו יש עיכוב בזיהוי תסמינים רפואיים משמעותיים – קיים סיכון ממשי. קיצור הזמן בין הופעת התסמינים לבין קבלת מענה מקצועי, ולוּ בצורה ראשונית ומיידית, עשוי להציל חיים, למנוע סיבוכים רפואיים ולשפר את סיכויי ההחלמה של החולה. Mediaid אינה נועדה להחליף אנשי מקצוע רפואיים, אלא לשמש ככלי עזר ראשוני קריטי, במיוחד בזמנים בהם עזרה מקצועית אינה נגישה מיידית.

פיתוח הפרויקט נשען על הבנה עמוקה של האתגרים הכרוכים בזיהוי עצמי של בעיות רפואיות, וכן על הכרה במגבלות הקיימות כיום בפתרונות הקיימים בתחום האבחון מרחוק. שילוב טכנולוגיות מתקדמות בניתוח מידע רפואי שואף להעניק למשתמשים כוח וכלים להתמודד בצורה טובה יותר עם מקרי חירום – עד לקבלת טיפול מקצועי.

### 2.2 תהליך המחקר

תהליך המחקר שביסס את פרויקט Mediaid התבצע כולו באמצעות חקירה מקיפה של מקורות אינטרנטיים, מאמרים מדעיים, בלוגים טכנולוגיים, תיעוד פרויקטים דומים, ודוחות של גופים רפואיים בינלאומיים. מטרת המחקר הייתה להבין את המצב הקיים, לזהות את הצרכים האמיתיים של משתמשים בעיתות חירום רפואיות, ולבחון את היתכנות השילוב בין טכנולוגיות NLP, ראייה ממוחשבת וטלרפואה למתן מענה אוטומטי ראשוני.

המחקר התבצע בשלבים:

1. **חקירה טכנולוגית**

בשלב זה התמקדתי בזיהוי טכנולוגיות הליבה שיכולות לשמש את הפרויקט:

* חיפשתי מידע על כלים קיימים בתחום **עיבוד שפה טבעית רפואית**, דוגמת MedCAT ו־scispaCy.
* חקרתי פלטפורמות ומודלים לניתוח תמונות רפואיות, בדגש על נגעים, פריחות ופציעות (למשל ResNet, EfficientNet בתחום Medical Imaging).
* בדקתי מסמכים העוסקים ב**טלרפואה בזמן אמת**, כולל אפליקציות קיימות כמו Ada, Babylon Health ודומותיהן, מתוך מטרה להבין את גבולות היכולת שלהן.

1. **בעיות במצב הקיים**

בשלב זה קראתי ביקורות משתמשים, חוות דעת מקצועיות ומאמרים המדגישים את החסמים שקיימים במערכות דיגיטליות רפואיות:

* חוסר הבנה של משתמשים את השפה הרפואית המקצועית.
* מערכות שמתקשות להתמודד עם טקסט חופשי לא מובנה (למשל ניסוחים עממיים של תסמינים).
* חוסר תמיכה בהעלאת תמונות ובניתוחן בזמן אמת.
* פערים משמעותיים בשמירה על פרטיות מידע רפואי.

1. **שילוב והכנת הפתרונות**

בשלב זה שילבתי את כל המידע שנאסף והתחלתי לבחון אילו פתרונות ניתן להציע. בין היתר:

* זיהיתי צורך בכלי **המאפשר קלט חופשי מצד המשתמש** (טקסט או תמונה) ומחזיר באופן מיידי פלט רפואי פשוט ומובן.
* הבנתי שהשימוש במודלים קיימים ידרוש התאמה מיוחדת לשפה רפואית עממית.
* למדתי מהכישלונות של מערכות קיימות, במיוחד אלה שהתבססו על שאלונים סגורים, שאינם מתאימים למצבי חירום משתנים ודינמיים.

**כלים עיקריים בהם נעשה שימוש במחקר:**

* גישה למאמרים דרך Google Scholar, IEEE Xplore, PubMed ו־arXiv.
* תיעוד של מודלים וקוד פתוח בפלטפורמות כמו GitHub, Hugging Face ו־Papers With Code.
* פורומים מקצועיים כגון StackOverflow, Reddit (r/HealthTech) ואתרי סקירה של מוצרי טלרפואה.

### 2.3 סקירת ספרות

סקירת הספרות שבוצעה לצורך פיתוח פרויקט Mediaid התמקדה בשלושה תחומי ידע מרכזיים: טכנולוגיות עיבוד שפה טבעית רפואית (Medical NLP), ראיית מחשב לאבחון רפואי (Medical Computer Vision), ויישומי טלרפואה להערכת מצב חירום.

**1. עיבוד שפה טבעית רפואית (Medical NLP)** מחקרים רבים הצביעו על פוטנציאל השימוש בעיבוד שפה טבעית לניתוח טקסטים רפואיים חופשיים. לדוגמה, מחקר של **(Wang et al. (2018** הדגים כיצד מערכות NLP יכולות לאתר תסמינים מתוך רשומות רפואיות לא מובנות, ולמפות אותם לסטנדרטים רפואיים דוגמת UMLS (Unified Medical Language System). מחקר נוסף של **(Rajkomar et al. (2019** הראה כי שילוב מודלים של עיבוד שפה רפואית יכול לשפר משמעותית את זיהוי מחלות בשלב מוקדם.

**2. ראיית מחשב קלינית (Medical Computer Vision)** בתחום ניתוח תמונות רפואיות, נרשמה התקדמות מרשימה בעשור האחרון, בעיקר בזכות התפתחות רשתות עצביות עמוקות (CNNs). מחקר של **(Esteva et al. (2017** הדגים זיהוי מדויק של סרטן עור ברמת מומחה באמצעות עיבוד תמונות קליניות. ממצאים אלו מדגישים את האפשרות להשתמש בטכנולוגיה דומה גם לניתוח נגעים, פצעים או תסמינים חיצוניים בתמונות המועלות על ידי משתמשים פרטיים.

**3. טלרפואה והערכת חירום מרחוק** תחום הטלרפואה התפתח בעיקר בעקבות מגפת הקורונה, עם דגש על מענה רפואי מהיר במצבים בהם הגעה פיזית לרופא אינה אפשרית. סקירה של **(Keesara et al. (2020** מראה כיצד מערכות דיגיטליות שיפרו את זמינות הטיפול, אך גם מצביעה על מגבלות – בעיקר בהקשר של הערכה עצמאית של חומרת מצב רפואי ללא תמיכת איש מקצוע.

**4. סיכום מגמות מרכזיות** מהסקירה עלה כי קיימים כלים בעלי יכולות מרשימות לניתוח נתונים רפואיים, אך רובם מתמקדים במשתמשים מקצועיים (רופאים, קלינאים) ואינם מותאמים למשתמש הקצה בעת חירום. כמו כן, נמצא כי אבטחת המידע היא גורם קריטי – במיוחד כאשר מדובר במידע רפואי אישי המועבר ומנותח באינטרנט.

בהתבסס על תובנות אלו, הוגדר כי Mediaid תשלב עיבוד טקסט חופשי ותמונות רפואיות תוך שימוש במודלים ייעודיים המותאמים לאוכלוסייה כללית, עם דגש חזק על אבטחת מידע, קלות שימוש, ומתן פידבק מיידי.

### 2.4 אתגרים מרכזיים

**2.4.1 הבעיה איתה התמודדתי**

בעת פיתוח הפרויקט Mediaid, האתגר המרכזי היה השילוב בין מספר תחומי טכנולוגיה מתקדמים – עיבוד שפה טבעית, ראייה ממוחשבת, אבטחת מידע רפואי, ואפליקציה הנוחה וידידותית למשתמשים במצבי חירום – לכלי אחד אפקטיבי, פשוט, וזמין לשימוש בזמן אמת. רוב הפלטפורמות הקיימות לא מצליחות לנתח גם טקסט רפואי חופשי וגם תמונות בצורה מדויקת ובזמן קצר. כמו כן, היה קושי למצוא מודלים שמתאימים לעברית רפואית או לשפה רפואית לא־מקצועית שנכתבת בידי משתמשים ללא רקע.

**2.4.2 הסיבות לבחירת הנושא**

הבחירה בנושא נבעה מהבנה עמוקה של חוסר האונים שחווים אנשים ברגעים קריטיים שבהם אין להם גישה מיידית לרופא, או שאין להם מושג כיצד לפרש בדיקות, תסמינים או תמונות של פצעים. המגמה הגוברת של שימוש בטלרפואה, יחד עם נגישות גבוהה לאמצעים דיגיטליים, פתחה הזדמנות לבנות פתרון שיגשר בין הצורך האנושי לתגובה רפואית מיידית לבין הכלים הטכנולוגיים הקיימים.

**2.4.3 המוטיבציה לעבודה**

המוטיבציה נבעה מרצון ליצור ערך מיידי שיש לו פוטנציאל להציל חיים או לכל הפחות להקטין את הפאניקה וחוסר הוודאות שמאפיינים מצבי חירום רפואיים. בנוסף, השילוב בין ידע טכנולוגי לבין צורך אנושי מוחשי העניק לעבודה תחושת שליחות. המחשבה שאדם ללא רקע רפואי יוכל לקבל תוך שניות ניתוח חכם של תמונה או תיאור תסמינים – הייתה הדלק המניע לפרויקט.

**2.4.4 הפתרון שנותנת המערכת**

הצורך המרכזי הוא מענה ראשוני מהיר ומדויק למי שנמצא במצוקה רפואית ואינו יודע כיצד לפעול. הפרויקט נותן פתרון בשלושה רבדים:

1. **פענוח תסמינים מטקסט חופשי** – כולל שפה יומיומית, ניסוחים עממיים, תיאורים אנושיים ולא רפואיים.
2. **ניתוח תמונה רפואית** – לדוגמה: זיהוי נגע, אדמומיות, שטפי דם, פריחה וכדומה.
3. **הצגת מידע רפואי ראשוני, מסודר ומנורמל** – כולל סיכום תמציתי, רשימת חשדות רפואיים, ושאלות המשך שכדאי לשאול את הרופא.

### 2.4.5 הצגת פתרונות לבעיה שנבחנו במחקר המקדים

במהלך המחקר נבחנו מספר פתרונות קיימים:

* **Ada Health** – אפליקציה מבוססת שאלונים סגורים. אינה תומכת בטקסט חופשי או תמונה.
* **Symptomate** – מנוע חיפוש תסמינים, מדויק אך מתבסס על תפריטים סגורים וללא הבנה של שפה טבעית.
* **K Health** – מערכת חזקה אך סגורה, עם גישה חלקית בלבד ל־AI רפואי, ודגש בעיקר על צ'אט עם רופא אנושי.

מסקנה מהשוואת הפתרונות הייתה כי אף אחד מהם אינו תומך בשילוב מלא של **טקסט חופשי**, **תמונה**, **פלט מובן למשתמש**, **קוד פתוח**, **ללא צורך בגישה ישירה לרופא** – ולכן יש מקום לפיתוח כלי חדשני שנותן מענה מהיר ומדויק, מבלי לוותר על אבחנה אחראית וזהירה.

## 3. מטרות ויעדים

מטרת העל של מערכת Mediaid היא לאפשר **תגובה רפואית ראשונית מהירה, נגישה, ומבוססת בינה מלאכותית**, למשתמשים שאינם אנשי רפואה, באמצעות ניתוח טקסט חופשי, תמונות רפואיות ואיחוד המידע לכדי תובנות פרקטיות.

היעדים המרכזיים של הפרויקט כוללים:

* פיתוח מנוע עיבוד שפה טבעית (NLP) המזהה תסמינים מתוך תיאור חופשי של המשתמש.
* בניית מערכת ניתוח תמונה רפואית (CV) שמזהה סממנים חיצוניים כגון פריחה, פצעים, דלקות ועוד.
* נירמול (normalization) הפלטים לטרמינולוגיה רפואית מוכרת (כגון SNOMED, ICD או UMLS).
* הצגת מידע תמציתי, ברור, עם המלצות או פעולות אפשריות להמשך (לדוגמה: "פנה לרופא", "עקוב אחר תסמינים").
* שמירה על פרטיות ועמידה בסטנדרטים מחמירים של אבטחת מידע רפואי.

## 4. אתגרים

הפרויקט התמודד עם מספר אתגרים עיקריים:

* **שפה לא רפואית**: המשתמש כותב או מדבר בשפה עממית, מעורפלת, ולעיתים רגשית. האתגר היה לתרגם את התיאורים למונחים רפואיים מדויקים.
* **עיבוד תמונה**: אין סטנדרט אחיד לצילום בעיות רפואיות (תאורה, זווית, חדות), ולכן היה צורך בפיתוח מערכת שמצליחה לפעול גם תחת תנאים לא אידיאליים.
* **אינטגרציה בין מקורות מידע**: חיבור בין ניתוח טקסט לניתוח תמונה ליצירת תובנה אחת ברורה.
* **אחריות רפואית**: מציאת האיזון בין מתן מידע שימושי לבין הימנעות מהגדרות אבחנתיות נחרצות שאינן בסמכות המערכת.
* **היעדר דאטה בעברית רפואית**: נדרש תהליך של איסוף מידע, תרגום, והתאמה של מודלים קיימים.

## 5. מדדי הצלחה למערכת

כדי להעריך את הצלחת Mediaid, הוגדרו מספר מדדים ברורים:

* **דיוק בזיהוי תסמינים מטקסט**: המערכת תיחשב מוצלחת אם תגיע לדיוק של 85% לפחות בזיהוי נכון של תסמינים מתוך טקסטים חופשיים.
* **דיוק בזיהוי ויזואלי**: המערכת תיחשב מדויקת אם תזהה נכונה לפחות 80% מהקטגוריות הוויזואליות שנבדקו.
* **מהירות תגובה**: זמן תגובה מרגע קלט (תמונה או טקסט) ועד הצגת פלט – פחות מ־30 שניות.
* **שביעות רצון משתמשים**: סקר משתמשים לאחר השימוש – מעל 85% דיווחו שהמידע שסופק היה מועיל, ברור וניתן ליישום.
* **שימוש בפועל**: מדד הצלחה נוסף הוא שימוש בפועל על ידי קבוצות בקרה או ציבור יעד – דגימת משתמשים שנעזרו במערכת תוך כדי תהליך הפיתוח.

# 6. רקע תיאורטי

המערכת המפותחת בפרויקט זה נועדה לספק ניתוח ראשוני ואינטליגנטי של מידע רפואי לא מובנה – הן טקסטואלי והן חזותי – באמצעות שילוב של טכנולוגיות מתקדמות מתחום הבינה המלאכותית. בפרט, המערכת מאפשרת למשתמשים להעלות תיאורים חופשיים של תסמינים רפואיים, וכן תמונות של נגעים או פצעים חיצוניים, ולקבל פלט מובנה של זיהויים רפואיים ראשוניים, עם הנחיות מיידיות לטיפול. בכך, היא יוצרת תשתית למענה מהיר, חכם ומבוסס־מידע במצבי חירום רפואיים או לצורכי תיעוד והכוונה ראשונית.

השאיפה לפענוח אוטומטי של מידע רפואי בלתי מובנה מציבה אתגרים ייחודיים: טקסטים רפואיים נוטים להיות לא תקניים, עמוסי קיצורים וראשי תיבות, ולעיתים קרובות נכתבים בשפה טבעית שאינה פורמלית. תמונות רפואיות, במיוחד כאלו המגיעות ממטופלים ולא מאנשי מקצוע, סובלות מאיכות משתנה, זוויות לא מבוקרות, ורקע חזותי לא אחיד. על כן, נדרש שילוב של מודלים חזקים ל־NLP רפואי, לצד מודלים חזותיים מותאמים לתחום הביו־רפואי.

המערכת מבוססת ומושתתת על data set רפואי רחב היקף שנקרא UMLS.

### UMLS – Unified Medical Language System

**UMLS (Unified Medical Language System)j** הוא פרויקט רחב היקף שפותח על ידי הספרייה הלאומית לרפואה של ארצות הברית (U.S. National Library of Medicine), במטרה לאחד ולגשר בין מערכות מונחים רפואיים שונות. מדובר במערכת שמהווה תשתית סמנטית־קונספטואלית המקיפה מאות אלפי מושגים רפואיים, מונחים נרדפים, מזהים סטנדרטיים, וסוגי קשרים בין מושגים – הכל תחת מסגרת אחת אחידה ומקושרת.

המטרה של UMLS היא לאפשר **אינטגרציה של מידע רפואי** ממקורות שונים – בין אם מדובר במסדי נתונים, מערכות מידע קליניות, או מסמכים חופשיים – באמצעות **זיהוי אחיד של מושגים רפואיים** ויצירת מיפוי ביניהם. בכך, הוא מהווה תשתית קריטית למערכות NLP רפואיות, מנועי חיפוש קליניים, מערכות החלטה תומכות (CDSS), ועוד.

UMLS Metathesaurus הוא המטאתזאורוס המרכזי של עולם הרפואה.

**מה זה Metathesaurus?**

המילה **Metathesaurus** באה מהמילה היוונית "meta" – כלומר "מעבר ל־", ו־"thesaurus" – אוסף של מונחים. במילים פשוטות:

Metathesaurus הוא תזאורוס־על, שמאגד, ממפה ומקשר בין עשרות תזאורוסים רפואיים קיימים ממקורות שונים – למאגר אחיד, מקושר ומסונכרן.

במקום שכל מערכת תשתמש במינוח רפואי שונה (למשל: ICD, SNOMED, MeSH), ה־Metathesaurus מייצר מזהה אחיד (CUI) למושגים שווי־ערך, גם אם הם מגיעים ממקורות שונים ובניסוחים שונים.

#### מבנה UMLS

המערכת מורכבת משלושה רכיבים עיקריים:

**1. Metathesaurus**

זהו הרכיב המרכזי ב־UMLS, והוא כולל:

* **מושגים רפואיים (Concepts)** – כל מושג מזוהה לפי מזהה ייחודי בשם **CUI – Concept Unique Identifier**.
* **מונחים נרדפים (Synonyms)** – לכל מושג ייתכנו עשרות ניסוחים שונים בשפות שונות או מונחים שונים המתייחסים לאותו הקונספט.
* **מקורות חיצוניים** – כל קונספט מקושר למקורות שבהם הוא מופיע (כגון SNOMED CT, ICD-10, MeSH, LOINC, RxNorm, ועוד).
* **יחסים סמנטיים בין קונספטים** – לדוגמה: קשרים היררכיים ("is-a"), קשרים של חלקים ("part-of"), קשרים סיבתיים ("causes", "complicates") וקשרים סמנטיים כלליים ("associated-with").

**2. Semantic Network**

זוהי רשת סמנטית שבה כל קונספט במטא־תזאורוס משויך ל־**Semantic Type** אחד או יותר, כגון:

* Disease or Syndrome
* Sign or Symptom
* Body Location or Region
* Therapeutic or Preventive Procedure
* Pharmacologic Substance

בנוסף, מוגדרים **קשרים אפשריים** בין סוגי ישויות. לדוגמה:

* "Disease or Syndrome" **may cause** "Sign or Symptom"
* "Procedure" **treats** "Disease"
* "Drug" **prevents** "Condition"

קשרים אלה מאפשרים ניתוח סמנטי עמוק של מידע רפואי ויצירת מסקנות לוגיות אוטומטיות.

**3. SPECIALIST Lexicon**

מרכיב זה כולל מילון תחבירי־בלשני עשיר שמסייע בכלים כמו MedCAT לבצע ניתוח שפה טבעית מתקדם:

* ניתוח נטיות (lemmatization)
* זיהוי יחסי תחביר
* הרחבת תבניות זיהוי

**קבצי UMLS ותפקידם בפרויקט**

ה־**UMLS Metathesaurus** כולל עשרות קבצי טקסט בפורמט (RRF (Rich Release Format, שכל אחד מהם אחראי על אספקט אחר של המידע הרפואי. בפרויקט הנוכחי נעשה שימוש בשלושה קבצים עיקריים: MRCONSO.RRF, MRSTY.RRF ו־MRREL.RRF. הקבצים מאפשרים גישה ישירה לנתוני UMLS לצורך ניתוח, מיפוי, ויצירת גרף סמנטי של קשרים בין מושגים רפואיים.

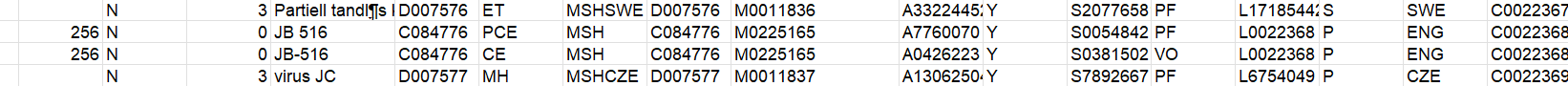
**1. MRCONSO.RRF – מושגים ומונחים (Concepts & Synonyms)**

זהו הקובץ המרכזי במטאתזאורוס, והוא מכיל את רשימת כל המונחים והמושגים הקיימים ב־UMLS. כל שורה מייצגת מופע של מונח אחד במקור אחד.

**שימושים עיקריים בפרויקט:**

* זיהוי שם קונספט לפי CUI.
* חיפוש מונחים מקבילים (synonyms) בשפות שונות.
* פילוח לפי מקור (SNOMED, MeSH, ICD וכו').

**מבנה - דוגמה לרשומות:**

****

**עמודות חשובות:**

|  |  |  |
| --- | --- | --- |
| **אינדקס** | **שם עמודה** | **תיאור** |
| 0 | CUI | מזהה ייחודי של הקונספט |
| 14 | STR | המחרוזת המקורית של המונח (השם) |
| 1 | LAT | שפת המונח (למשל ENG) |
| 11 | SAB | מקור המידע (למשל SNOMEDCT\_US, MSH, ICD10) |
| 12 | TTY | סוג המונח (לדוג' PT – preferred term, SY – synonym) |

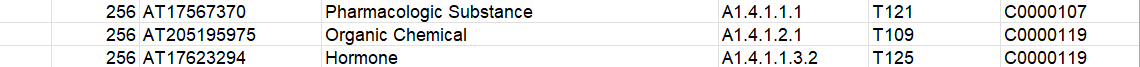
**2. MRSTY.RRF – סוגים סמנטיים (Semantic Types)**

קובץ זה קובע לאיזה **סוג סמנטי** שייך כל קונספט – לפי המערכת הסמנטית של UMLS. כל CUI עשוי להשתייך ל־1 או יותר סוגים.

**שימושים עיקריים בפרויקט:**

* סינון של סימפטומים בלבד לצורך המודלים.
* שיוך קונספטים לקבוצות רפואיות: מחלות, טיפולים, תרופות, אנטומיה וכו'.
* יצירת גרף סמנטי מקובץ MRREL, תוך התייחסות לסוגי המושגים.

**מבנה - דוגמה לרשומות:**



**עמודות חשובות:**

|  |  |  |
| --- | --- | --- |
| **אינדקס** | **שם עמודה** | **תיאור** |
| 0 | CUI | מזהה ייחודי של הקונספט |
| 1 | TUI | מזהה סוג סמנטי (Type Unique Identifier) |
| 3 | STY | תיאור סוג הקונספט |

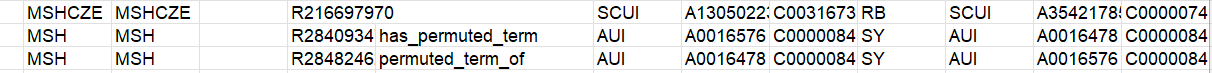
**3. MRREL.RRF – קשרים בין מושגים (Relationships)**

קובץ זה מכיל את הקשרים הסמנטיים בין קונספטים ב־UMLS. כל שורה מייצגת קשר בין שני מושגים.

**שימושים עיקריים בפרויקט:**

* בניית גרף של קשרים בין מושגים רפואיים (ב־Neo4j).  
  חיפוש מחלות קשורות לסימפטומים.
* זיהוי מונחים נרדפים, היררכיות (is-a), קשרים סיבתיים (causes, complicates) ועוד.

**מבנה - דוגמה לרשומות:**



**עמודות חשובות:**

|  |  |  |
| --- | --- | --- |
| **אינדקס** | **שם עמודה** | **תיאור** |
| 0 | CUI1 | קונספט מקור |
| 2 | CUI2 | קונספט יעד |
| 4 | REL | סוג הקשר (למשל PAR – מושג־אב, SY – נרדף) |
| 5 | RELA | קשר מורחב יותר (RO, RB, RN וכו') |

שלושת הקבצים האלה – MRCONSO, MRSTY, MRREL – מרכיבים את הליבה הפונקציונלית של השימוש ב־UMLS בפרויקט:

* MRCONSO משמש לזיהוי שמות וקודדים רפואיים,
* MRSTY מאפשר לסווג ולסנן לפי תחום רפואי,
* MRREL מאפשר להפיק קשרים ולהבין הקשרים סמנטיים בין מונחים.

שילוב מידע מכל שלושת הקבצים הללו מאפשר **זיהוי, סיווג וקישור סמנטי של סימפטומים**, תוך יצירת בסיס לניתוח גרפי, המלצות רפואיות, או העשרה של נתונים קליניים לא־מובנים.

#### UMLS Browser

במהלך הפיתוח והניסוי של הפרויקט, נעשה שימוש ב־**UMLS Metathesaurus Browser** – מערכת אינטראקטיבית מבית National Library of Medicine (NLM), המאפשרת גישה נוחה ומיידית לתוכן של המטאתזאורוס ללא צורך בעיבוד ישיר של קבצי .RRF.

באמצעות הדפדפן ניתן לבצע:

* חיפוש לפי מונח רפואי (למשל: *headache*) ולקבל את כל ה־CUI הרלוונטיים.
* חיפוש לפי מזהה (CUI) ולראות אילו שמות נרדפים, מקורות, וקודים אחרים משויכים אליו.
* הצגת הקשרים הסמנטיים של המושג, כולל היררכיות (parent, child), נרדפים, קשרים סיבתיים ועוד.
* צפייה בסוג הסמנטי של כל קונספט (למשל "Sign or Symptom", "Disease", "Procedure").
* מעבר ישיר למקורות חיצוניים (כגון SNOMED, MeSH) עם קישורים חיצוניים.

**שימוש בפרויקט**

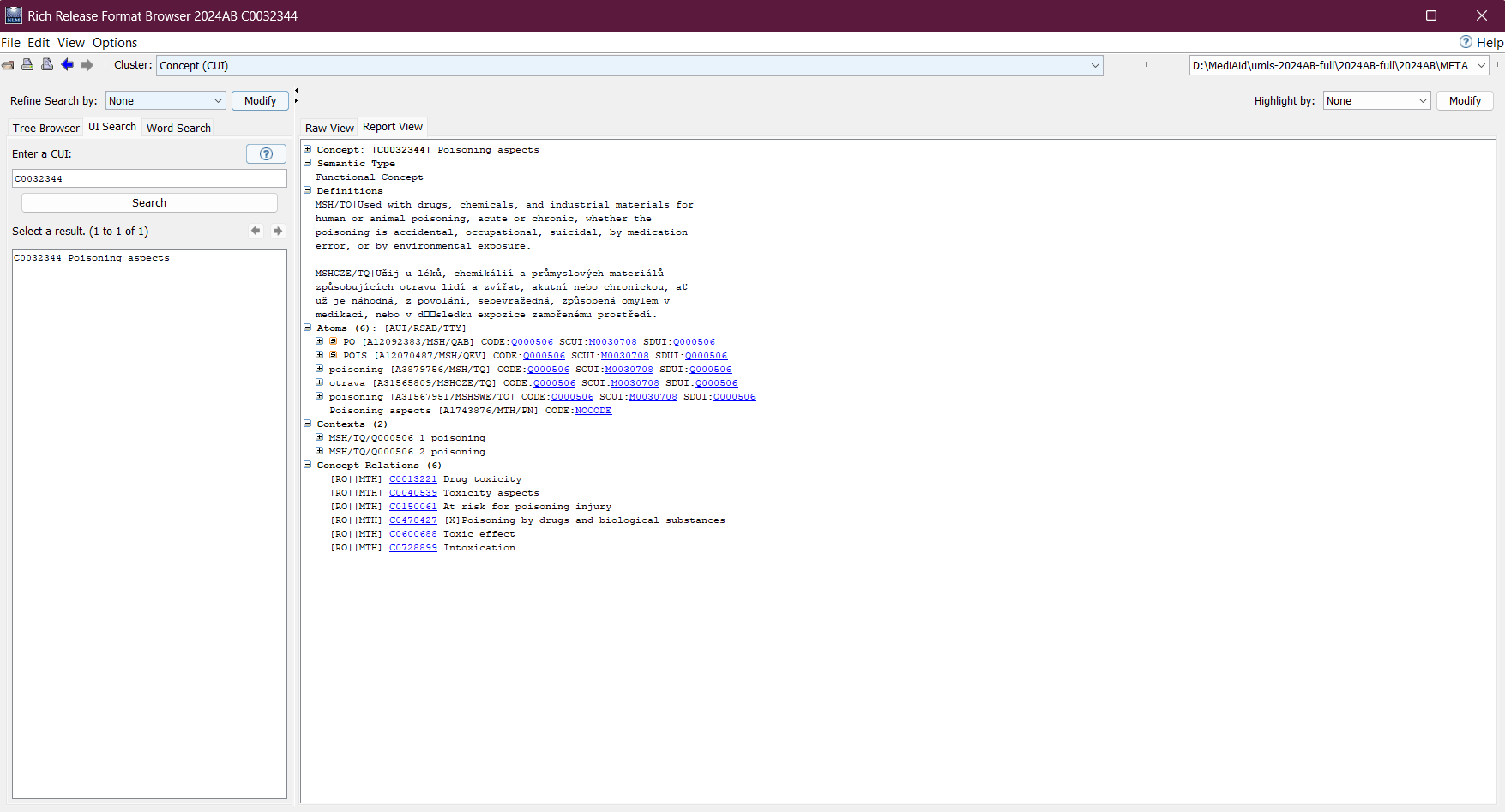
ה־Browser שימש בפרויקט לצורך:

* **וידוא נכונות תוצאות** שהתקבלו מה־MedCAT.
* **בחינה ידנית של קשרים** בין סימפטומים שזוהו לבין מחלות רלוונטיות.
* **איתור תיאורים מקוצרים** (למשל מתוך MeSH) עבור תצוגה למשתמש.
* **בדיקה חוצה מקורות** של קונספטים.

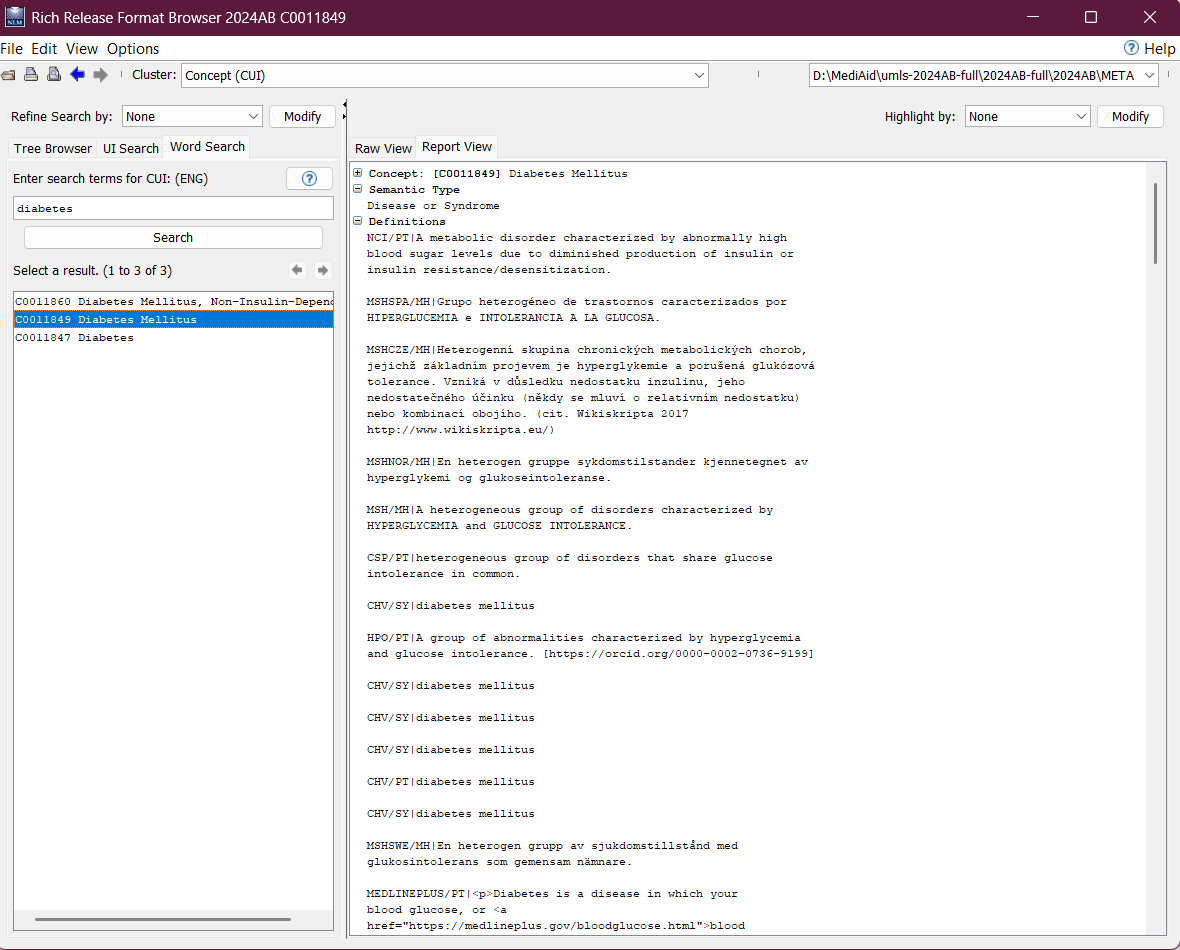
**דוגמאות:**

מתוך הBrowser המותקן יחד עם הUMLS על המחשב:

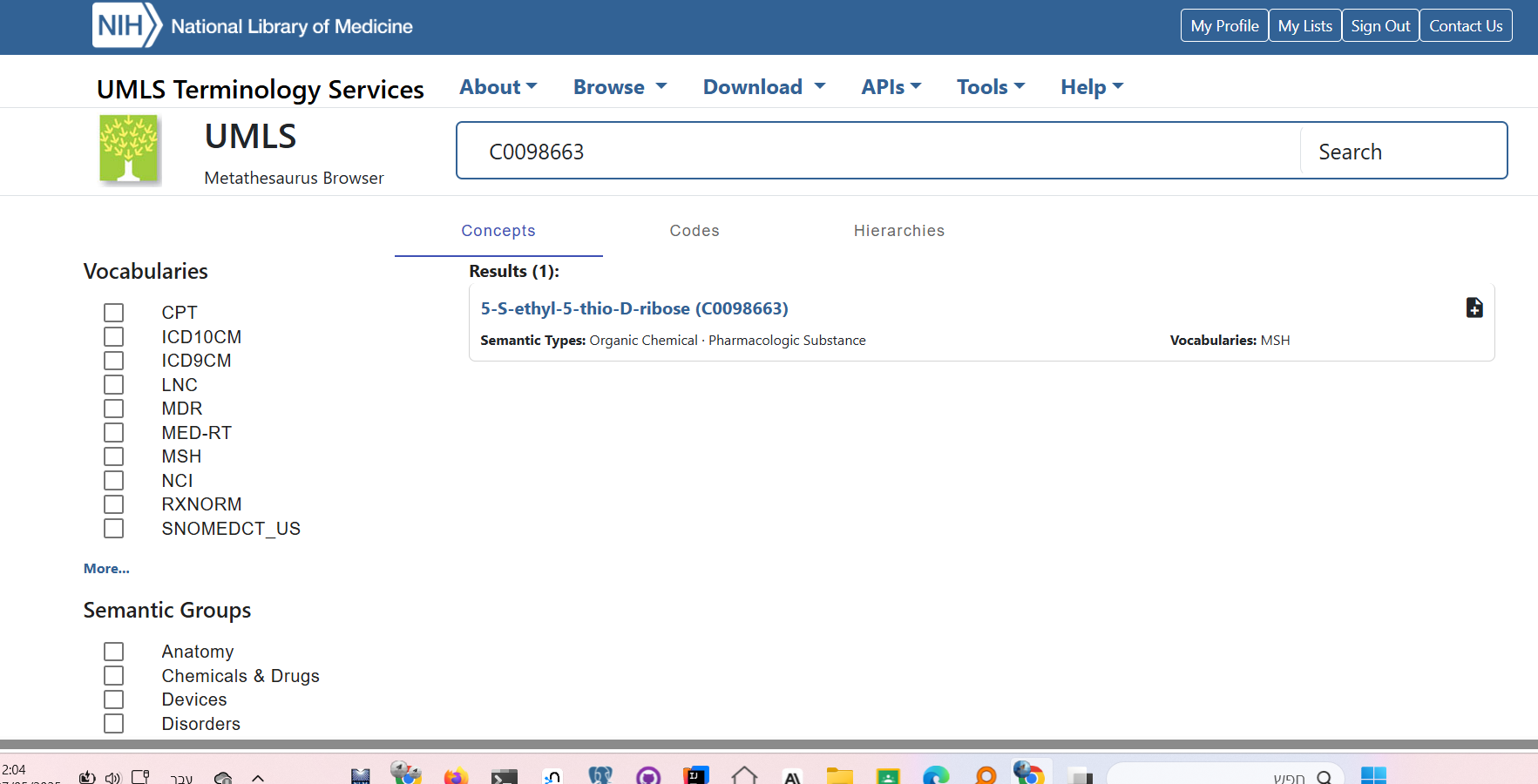
* חיפוש לפי CUI:

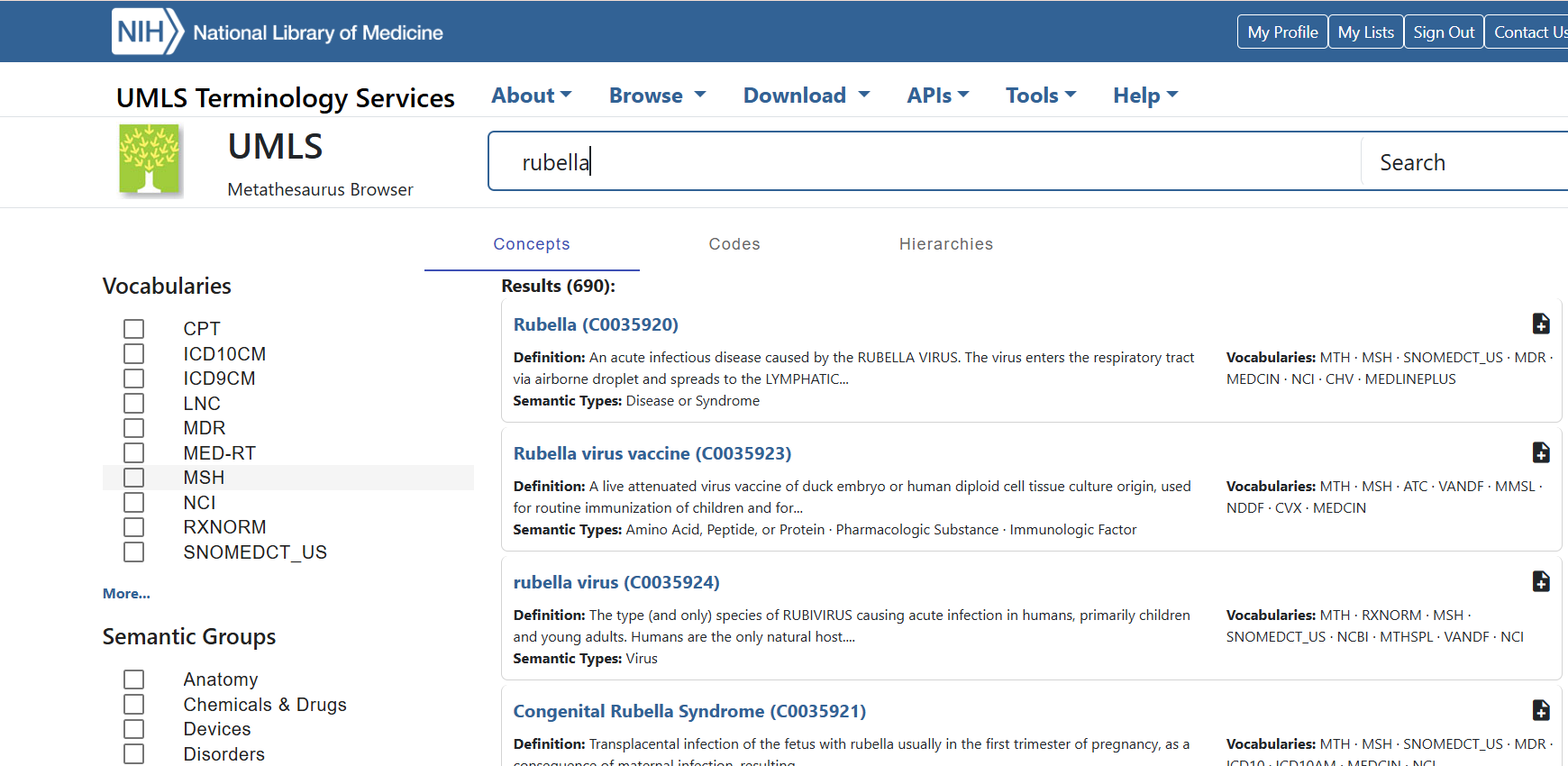


* חיפוש לפי מילת מפתח:



מתוך UMLS Metathesaurus Browser בכתובת: <https://uts.nlm.nih.gov/uts/umls/home>





**תפקיד מערכת ה־UMLS בליבת המערכת**

מערכת MediAid עושה שימוש עמוק ומקיף ב־UMLS – לא ככלי עזר נקודתי, אלא כבסיס סמנטי רחב שעליו מושתת כל עיבוד הידע הרפואי בטקסטים ובתמונות. UMLS מספק את השפה האחידה, את המושגים, את הקשרים הסמנטיים ואת המבנה המושגי שמאפשר למערכת "להבין רפואה" באופן שיטתי, מבוסס ומדויק.

**למה דווקא UMLS הוא הבסיס?**

מערכת MediAid מיועדת להתמודד עם קלטים חופשיים, לא־אחידים ומרובי־משמעויות – כמו תיאורי תסמינים, מונחים עממיים, שמות תרופות מקומיים או קיצורים קליניים. כדי לפענח בצורה תקפה את המידע הזה ולמפות אותו למושגים רפואיים פורמליים, דרושה שכבת ידע אחידה, עשירה, רחבת־היקף, ובעלת היררכיה ברורה וקשרים סמנטיים בין ישויות. כאן בדיוק נכנס UMLS.

**השימושים המרכזיים ב־UMLS במערכת**

1. **מיפוי אוניברסלי של ישויות רפואיות** כל טקסט חופשי ממופה ל־Concept Unique Identifiers (CUIs), שמייצגים מושגים רפואיים אחידים מתוך בסיס ידע רחב הכולל מיליוני מונחים. כך ניתן לגשת לכל מושג רפואי – סימפטום, מחלה, תרופה, בדיקה – בצורה פורמלית, ולא רק טקסטואלית.
2. **אינטגרציה בין מודולים שונים** ה־CUIs משמשים כ"שפת תווך" בין רכיבי המערכת השונים: זיהוי תמונה, גרף הידע הרפואי, מודול המלצות, מחשוב גורמי סיכון – כולם עובדים מול אותם מזהים רפואיים מוסכמים.
3. **הבנת קשרים רפואיים מורכבים** UMLS מספק גם את הקשרים בין מושגים (כגון "מחלה גורמת לסימפטום", "תרופה מטפלת במחלה", "סימפטום קשור למערכת גופנית") – המידע הזה מוטמע בגרף הידע ובאלגוריתמי הסקת המסקנות.
4. **סטנדרטיזציה של מידע ממקורות שונים** המערכת קולטת נתונים ממקורות מגוונים: טקסט חופשי, מסמכים רפואיים, תמונות נגעים חיצוניים, ומידע מובנה מהמשתמש. בזכות UMLS, כל הקלטים האלה מקודדים לתוך פורמט אחיד שניתן לנתח ולהצליב.
5. **בסיס למודול ה־Explainability** כאשר המערכת מציגה למשתמש המלצה – למשל, לפנות לרופא עור – היא יכולה להסביר זאת דרך שרשרת מושגים UMLS: מהתסמין → למחלה אפשרית → לבדיקה נדרשת → לגורם מטפל.

**לסיכום**

UMLS אינו רכיב צדדי בפרויקט MediAid – הוא **עמוד שדרה סמנטי** שמאפשר למערכת להבין, לנתח, להסביר ולחבר מידע רפואי בצורה אחידה, שקופה ורב־ממדית. כל שכבת ניתוח או המלצה נשענת עליו – מטקסט, דרך תמונה, ועד גרף ידע מותאם אישית למשתמש.

**דוגמה: סוכרת (Diabetes)**

**מושג עיקרי:**

* CUI: C0011849
* שם: Diabetes Mellitus (סוכרת)

**קשרים:**

* Type 1 Diabetes (C0011854) **IS\_A** Diabetes Mellitus (C0011849 - סוכרת סוג 1 היא סוג של סוכרת
* Type 2 Diabetes (C0011860) **IS\_A** Diabetes Mellitus (C0011849 - סוכרת סוג 2 היא סוג של סוכרת
* Insulin (C0021641) **TREATS** Diabetes Mellitus (C0011849- אינסולין מטפל בסוכרת
* Hyperglycemia (C0020456) **ASSOCIATED\_WITH** Diabetes Mellitus (C0011849 - רמת סוכר גבוהה בדם קשורה לסוכרת
* Diabetic Retinopathy (C0011884) **COMPLICATION\_OF** Diabetes Mellitus (C0011849 - רטינופתיה סוכרתית היא סיבוך של סוכרת.

לסיכום, ה־UMLS הוא תשתית קריטית לסטנדרטיזציה של מידע רפואי, ומהווה מרכיב מרכזי בכל מערכת NLP רפואית מודרנית. שילובו בפרויקט הנוכחי מאפשר לתרגם טקסטים חופשיים של משתמשים לרשימות סימפטומים אחידות, מדויקות, ניתנות לניתוח והצלבה – ובכך מגשר בין עולם השפה האנושית ובין עולם המידע הקליני הדיגיטלי.

### חילוץ סימפטומים

בעת חירום המשתמש מזין מידע באמצעות טקסט חופשי או תמונה. המטרה בשלב זה היא להגיע לרשימה של סימפטומים אותה מתאר המשתמש, כשהם מנורמלים למזהי CUI ע"פ UMLS.

### זיהוי ישויות רפואיות חכמות מטקסט באמצעות MedCAT

בפרויקט זה נעשה שימוש בספריית **MedCAT (Medical Concept Annotation Tool)** – כלי קוד פתוח מתקדם לכריית טקסטים רפואיים וחילוץ מושגים רפואיים סטנדרטיים מתוך טקסט חופשי. הספרייה פותחה כדי לתת מענה לצורך הגובר בעיבוד שפה טבעית (NLP) בתחומי הרפואה, במיוחד בעבודה עם מסמכים קליניים לא־מובנים, כגון תיאורי תסמינים חופשיים של מטופלים, סיכומי אשפוז, או תיעוד קליני.

MedCAT עושה שימוש בשיטות מתקדמות של למידת מכונה ולמידה עמוקה, ומשלבת בין גישות מבוססות־כללים ובין למידת הקשר סטטיסטית, כדי לזהות ישויות רפואיות בטקסט ולשייך אותן ל־**UMLS (Unified Medical Language System)h** – מסד נתונים רפואי מקיף הכולל מאות אלפי מושגים רפואיים בתצורה היררכית וסטנדרטית.

במסגרת הפרויקט, הוטען מודל קיים בשם umls\_sm\_pt2ch\_533bab5115c6c2d6, והוא שימש לניתוח משפטים רפואיים חופשיים והמרתם לרשימת סימפטומים מזוהים. לדוגמה, משפט כמו:

"Patient complains of headache and fever"

נותח באמצעות המודל, וכתוצאה מכך זוהו הישויות **headache** ו־**fever** – שתיהן קיבלו מזהי CUI, שמות סטנדרטיים מתוך UMLS, וכן דירוגי דיוק (accuracy), ניתוח הקשר (contextual similarity), וסיווג לפי סוגים (TUI – Type Unique Identifier).

השימוש ב־MedCAT מאפשר להפיק מידע רפואי משמעותי מתוך טקסטים לא־מובנים, באופן שמקדם סטנדרטיזציה, אוטומציה ויכולת שילוב עם מערכות רפואיות אחרות (כגון מערכות תיוג, סיווג, חיזוי או ניתוח אפידמיולוגי).

יתרונו המרכזי של MedCAT הוא ביכולתו להתחשב בהקשר הלשוני ובשגיאות כתיב, ולהפיק ישויות גם מתוך טקסטים מורכבים – תכונה קריטית במיוחד בעבודה עם טקסטים שנכתבו בידי מטופלים או תיעוד חופשי של אנשי צוות רפואי.

### סיווג תמונות רפואיות באמצעות BioMedCLIP

לצורך ניתוח אוטומטי של תמונות נגעים עוריים בפרויקט, נעשה שימוש במודל בשם **BioMedCLIP** – מודל שפותח על ידי Microsoft ומהווה התאמה רפואית למודל CLIP המקורי של OpenAI. מטרת המודל היא לקשר בין תיאורים טקסטואליים למידע חזותי רפואי, כך שניתן יהיה לזהות ממצאים רפואיים מתמונה גם ללא צורך באימון ספציפי.

BioMedCLIP מבוסס על גישת **Zero-Shot Classification**, כלומר: מקבלים תמונה רפואית, ורשימת תיאורים מילוליים (labels), וללא אימון נוסף – המודל מדרג את הסבירות שכל תיאור מתאים לתמונה. השוואה זו נעשית במרחב וקטורי משותף לתמונה ולטקסט, ומחזירה הסתברויות לכל תיאור.

בפרויקט הנוכחי, נעשה שימוש בתיאורים המבוססים על **מושגים עוריים מ־UMLS** – מערכת מונחים רפואיים אחידה. כל תיאור ממופה אוטומטית למזהה רפואי (CUI), כך שהתוצאה היא לא רק "label" אלא גם ממצא רפואי סטנדרטי, הניתן לשילוב עם טקסטים שנכתבו על ידי המטופל (שנותחו באמצעות MedCAT).

היישום כולל:

* טעינת המודל מתיקייה מקומית שמכילה משקלים מוקפאים של HuggingFace.
* עיבוד ראשוני של תמונה שהועלתה.
* המרה של רשימת מושגים רפואיים למשפטים כגון:  
   *“this is a photo of cellulitis”*
* חישוב הסתברויות להתאמה בין התמונה לכל תיאור.

יתרונות המודל: אין צורך באימון ייעודי, מאפשר עבודה עם קטגוריות רפואיות מגוונות, ומספק פלט שניתן להצליב עם מידע טקסטואלי. החיסרון המרכזי הוא בכך שהמודל אינו מקבל מידע קליני נלווה (למשל תסמינים או רקע רפואי), ולכן הוא מתאים בעיקר לשלב הסינון הראשוני או לאימות חזותי של תיאורים טקסטואליים.

### מסדי נתונים – PostgreSQL ו־Neo4j

מערכות ניתוח מידע רפואי מחייבות תשתית נתונים אמינה, גמישה ומבוססת סטנדרטים, המאפשרת שילוב בין נתונים טבלאיים (structured data) לבין ידע סמנטי מורכב המייצג קשרים בין ישויות רפואיות. לצורך כך, נעשה בפרויקט שימוש בשני מסדי נתונים משלימים: **PostgreSQL**, כמסד רלציוני מרכזי, ו־**Neo4j**, כמסד גרפי לייצוג קשרים סמנטיים מתוך UMLS ומקורות נוספים.

**PostgreSQL – מסד נתונים רלציוני**

PostgreSQL הוא מסד נתונים רלציוני בקוד פתוח, מהמתקדמים והנפוצים בעולם. המסד מתאפיין ביכולת לבנות סכמות מורכבות, לתמוך בשלמות הנתונים (data integrity) באמצעות הגדרות של primary keys, foreign keys, constraints, וכן לנהל עסקאות (ACID transactions) באופן מלא.

בפרויקט, מסד PostgreSQL משמש לאחסון מידע מובנה שנוצר בתהליך העבודה, לרבות:

* פרטי משתמשים, בקשות ופעולות שביצעו.
* פלטים מובנים של סימפטומים שנחולצו מטקסט או תמונה.
* מזהי CUI של סימפטומים מזוהים.
* תאריכים, סטטיסטיקות, ומידע טכני עבור תיעוד וניתוח שימוש.

השימוש במסד רלציוני מאפשר שאילתות יעילות, ניתוחים סטטיסטיים פשוטים, ושילוב עתידי עם מערכות רפואיות סטנדרטיות (כגון FHIR או מערכות HIS).

**Neo4j – מסד גרפי לקשרים סמנטיים**

Neo4j הוא מסד נתונים גרפי (Graph Database) אשר מבוסס על ייצוג מידע בצורת קודקודים (Nodes) וקשרים (Edges). להבדיל ממסד רלציוני שבו הנתונים מאורגנים בטבלאות, מסד גרפי נועד במיוחד לייצוג ידע מרובה־קשרים, היררכי, ובלתי צפוי – תכונות אשר מאפיינות היטב את תחום הרפואה, ובמיוחד את UMLS.

בפרויקט נעשה שימוש ב־Neo4j לשמירה וניתוח של:

* קשרים סמנטיים בין סימפטומים, מחלות, תרופות, גורמי סיכון ועוד.
* מבנה היררכי של ישויות רפואיות (למשל: "Headache" הוא תת־מקרה של "Pain").
* קשרים של סיבה ותוצאה, קשרים אנטומיים, וטיפולים רלוונטיים.

מימוש הקשרים מאפשר:

* חיפוש מושגים לפי הקשר רפואי.
* הצעת סימפטומים או מחלות קשורות.
* הצלבת פלטים של מודלים שונים (למשל: חיבור בין תיאור טקסטואלי ותמונה) דרך הגרף.

השפה בה נעשה שימוש היא **Cypher**, שפת שאילתות דקלרטיבית המותאמת למסדי גרפים.

**אינטגרציה בין המסדים**

שני המסדים פועלים במקביל ומשלימים זה את זה:

* PostgreSQL מטפל בניהול ואחסון של הנתונים הגולמיים, קלטים ופלטים מובנים.
* Neo4j מאפשר ניתוח סמנטי מורכב והסקת קשרים.

השימוש הכפול במסד רלציוני ובמסד גרפי מעניק גמישות רבה למערכת: מצד אחד ניהול מדויק של מידע מובנה, ומצד שני הבנה עמוקה של הקשרים המהותיים בין ישויות רפואיות. שילוב זה מהווה תשתית יציבה וסקלאבילית להרחבה עתידית של המערכת.

# 7. תיאור מצב קיים

מערכות תמיכה רפואית דיגיטליות מתפתחות בעשור האחרון במגוון תחומים, אולם מרבית הפתרונות הזמינים כיום ממוקדים באנשי מקצוע רפואיים – רופאים, אחיות ומטפלים מוסמכים – ופחות מותאמים לאדם מן השורה הנמצא במצוקה רפואית ראשונית. ברוב המקרים, אבחנה ראשונית או המלצה לפעולה דורשות הגעה פיזית למרפאה, או שימוש במערכות סגורות שאינן מאפשרות גישה ישירה ואוטונומית מצד המשתמש.

קיימות אמנם אפליקציות בריאות לציבור (כגון מערכות triage דיגיטליות או בוטים רפואיים), אך רובן מתבססות על שאלונים סגורים, בחירה מתוך תפריטים, או הזנה של נתונים מובנים מראש. מערכות אלו אינן מסוגלות לעבד תיאור מילולי חופשי של סימפטומים, ולעיתים אינן מקבלות כלל קלט חזותי (כגון תמונה של פריחה, פצע או נפיחות). בנוסף, הפלט שהן מספקות אינו תמיד מותאם אישית, שכן הוא אינו מבצע הצלבה עם המידע הרפואי האישי של המשתמש (למשל: רקע רפואי, אלרגיות, תרופות נוכחיות).

בתחום עיבוד תמונות רפואיות – במיוחד תמונות שצולמו על ידי מטופלים עצמם – קיימים פתרונות מבוססי בינה מלאכותית, אך רבים מהם חסרים שקיפות, סטנדרטיזציה, או הסבר קליני אמיתי. ברוב המקרים, הפלט הוא קטגוריה כללית או דירוג סיכון, ולא הנחיית פעולה רפואית מבוססת הקשר אישי.

נכון להיום, אין פתרונות מבוזרים, גמישים ומיידיים, אשר מאפשרים עיבוד בזמן אמת של טקסט חופשי או תמונה, הצלבת מידע עם פרופיל רפואי קיים, והפקת המלצה רפואית מותאמת אישית. על כן, קיים צורך ממשי בפיתוח מערכת המסוגלת לגשר בין עולם המידע הראשוני שנמסר על ידי המשתמש, לבין פרשנות קלינית מיידית, אחראית וסטנדרטית – כפי שמטרתו של פרויקט זה.

# 8-9. ניתוח חלופות מערכתי והחלופה הנבחרת

בפרויקט זה נבנתה מערכת מורכבת המשלבת רכיבי בינה מלאכותית לעיבוד שפה טבעית (NLP), עיבוד תמונה רפואית (Computer Vision), ניהול מסדי נתונים, ותשתית צד שרת ולקוח. כל רכיב נבחר לאחר בחינה של מספר חלופות, תוך שקלול שיקולים כגון דיוק, זמינות, קוד פתוח, קלות אינטגרציה, ביצועים בזמן אמת, ותמיכה במונחים רפואיים סטנדרטיים.

בפרק זה ינותחו מספר חלופות אפשריות לרכיבי המערכת השונים, עם פירוט היתרונות והחסרונות של כל גישה או טכנולוגיה. הבחירות שבוצעו בפרויקט ינמקו מתוך מענה מיטבי לדרישות הפרויקט: עבודה עם קלט טקסטואלי וחזותי לא מובנה, הפקת פלט רפואי תקני (UMLS), והצלבה עם מידע אישי רפואי של המשתמש.

### חלופות ל-dataset

כדי לתרגם תיאורים חופשיים של תסמינים רפואיים למידע קליני מדויק, יש צורך במערכת מונחים רפואית סטנדרטית – כזו שמאפשרת למחשב "להבין" למה המשתמש מתכוון כשהוא כותב מונחים כמו "כאב ראש", "עור מגרד", או "כאב בחזה". מערכות כאלה מספקות לא רק שמות קבועים למושגים, אלא גם מזהים ייחודיים (כגון CUI), היררכיה, קשרים בין תסמינים ומחלות, ולעיתים גם תרגום בין שפות או קיצורים.

קיימות מספר מערכות מונחים מוכרות בעולם הרפואה, ביניהן:

SNOMED CT – מערכת היררכית גדולה, נפוצה במדינות רבות, ומתאימה ל־EHRs.

ICD-10 – מערכת קידוד אבחנות בשימוש בעיקר לצרכים סטטיסטיים, רישום וביטוח.

MeSH – אוצר מונחים רפואיים המשמש בעיקר למיון וחיפוש ספרות רפואית.

RxNorm – לקידוד תרופות ויחסי גומלין ביניהן.

LOINC – לציוני בדיקות מעבדה ונתונים קליניים מספריים.

למרות שלכל אחת יתרונות בתחומה, רוב המערכות האלו מתמחות רק בתחום מסוים, ואינן כוללות מידע סמנטי רחב על סימפטומים, תסמונות או מונחים מהשפה הרפואית המדוברת. בנוסף, בחלק מהן יש מגבלות רישוי או צורך בהתאמה למדינות מסוימות.

לעומת זאת, UMLS – Unified Medical Language System נבנה מראש כאיחוד של עשרות מערכות מונחים, כולל כולן (SNOMED, ICD, MeSH ועוד), תחת מסד אחד עם מזהים אחידים (CUIs). הוא כולל:

מאות אלפי מונחים רפואיים מתורגמים, מקושרים ונרדפים.

קשרים היררכיים וסמנטיים בין מושגים.

שיוך לכל מושג לפי סוג (סימפטום, איבר, מצב, הליך).

קוד פתוח וזמין לשימוש במחקר ובפיתוח.

הבחירה ב־UMLS לפרויקט הייתה ברורה: מדובר במערכת עשירה, פתוחה, סטנדרטית, ומקושרת היטב, שתואמת בדיוק לצורך של המערכת – תרגום של טקסט או תמונה לפלט אחיד, שמקושר לעולם הרפואה הקלינית. גם המודלים שבהם נעשה שימוש (MedCAT, BioMedCLIP) עובדים ישירות מול UMLS, מה שמאפשר אינטגרציה קלה, והצלחה גבוהה בזיהוי סימפטומים במונחים מדויקים.

### חלופות למודל ניתוח טקסט רפואי

במרכזו של כל תהליך עיבוד טקסט רפואי עומדת השאלה – האם המערכת באמת "מבינה" על מה המטופל מדבר? לא מספיק לזהות מילים רפואיות. כדי להוציא תועלת קלינית מהטקסט, נדרש לזהות את המשמעות המדויקת של כל ישות – ובעיקר לדעת אם מדובר בסימפטום, איבר, מחלה, תהליך רפואי או דבר אחר. בפרויקט הזה, המטרה היא לחלץ אך ורק סימפטומים מתוך הטקסט, ולכן היכולת לדעת אם מילה מסוימת היא Sign\_or\_Symptom היא קריטית.

במהלך העבודה נבחנו כמה ספריות נפוצות בעולם ה־NLP הרפואי. אחת הבולטות היא SciSpacy – כלי מהיר ונוח שמתבסס על SpaCy, עם תמיכה חלקית בזיהוי ישויות רפואיות. עם זאת, כבר בשלב מוקדם עלתה בעיה מהותית: SciSpacy אמנם מזהה מילים רפואיות ומחזיר את קוד ה־UMLS שלהן, אבל לא מחזיר שום אינדיקציה לסוג הישויות. כלומר, אין לדעת אם מונח מזוהה מתאר סימפטום, איבר או הליך רפואי. הדבר הפך את SciSpacy לבלתי שמיש למטרת הפרויקט, כי לא ניתן לסנן מתוך כל ה־CUIs את אלו שמייצגים רק תסמינים.

נשקל גם השימוש ב־BioBERT, מודל עוצמתי מבוסס Transformers שעבר אימון על ספרות רפואית וקלינית. מדובר במודל גמיש ובעל פוטנציאל גבוה, אך בפועל הוא לא מספק זיהוי ישויות מוכן לשימוש (NER), ולא תומך בנורמליזציה ל־UMLS או בהחזרת סוג רפואי. השימוש בו היה מחייב תהליך מלא של fine-tuning ודירוג תיוגים – תהליך שאינו מעשי בפרויקט שמבוסס על עבודה בזמן אמת ובמשאבים מצומצמים.

כלי נוסף שנבחן היה MetaMap, פרויקט ותיק מבית NLM, אשר דווקא כן תומך בזיהוי סוגים רפואיים ומחזיר גם את תגית ה־TUI של כל מושג. עם זאת, הוא איטי, מורכב מאוד להטמעה, ומותאם בעיקר למסמכים קליניים רשמיים – לא לטקסטים חופשיים ולא־תקניים של משתמשים רגילים. בפועל, התוצאה הייתה שהמערכת לא הצליחה להתמודד עם ניסוחים יומיומיים כמו "כואב לי הראש" או "יש לי פריחה בגב", דבר שפגע בתפקוד המבוקש.

מתוך כל האפשרויות שנבדקו, MedCAT התבלט ככלי היחיד שנותן מענה שלם ומדויק לדרישות הפרויקט. הוא מזהה ישויות רפואיות מתוך טקסט חופשי בשפה טבעית, ממפה אותן למזהי UMLS (CUIs), ומחזיר לכל ישות את סוגה הרפואי – כולל Sign\_or\_Symptom, Body\_Part, Disease\_or\_Syndrome, ועוד. מעבר לכך, הוא גם מבצע ניתוח של הקשר: האם מדובר בהצהרה חיובית, בהכחשה, או בתיאור של אדם אחר. זה מאפשר סינון מדויק של תסמינים, ותיאום מלא עם המודול הוויזואלי של המערכת.

הבחירה ב־MedCAT נבעה לא מהנוחות אלא מהצורך: זו הייתה האפשרות היחידה שהתאימה באמת לאתגר – לתרגם טקסט חופשי של מטופל לפלט רפואי תקני, שמסומן כסימפטום, וניתן לעיבוד נוסף.

### חלופות למודל ניתוח תמונה

אחד האתגרים המשמעותיים בפרויקט היה עיבוד של תמונות רפואיות לא מובנות – כאלו שהמשתמש מצלם בעצמו: פריחה, פצע, נגע, נפיחות, כוויה ועוד. מדובר בתמונות שאינן בהכרח ברזולוציה רפואית גבוהה, ומכילות רעש רקע, זוויות לא אחידות ולעיתים גם אור וחשיפה לא עקביים. נדרש היה מודל שמסוגל להבין את תוכן התמונה גם בלי הדרכה מראש, ולקשר אותו למונחים רפואיים מתוך רשימה פתוחה.

בנקודה זו עלתה השאלה: האם נכון לבחור מודל קלאסי לזיהוי תמונה רפואית – או דווקא גישה חדשנית של התאמה בין תמונה לטקסט?

נבדקו כמה אפשרויות:

מודלים מסורתיים כמו **ResNet** או **EfficientNet** נחשבים לאמינים ומהירים, אך הם דורשים דבר מהותי – אימון מפורש על סט תמונות מתויגות מראש. כלומר, כל מחלה או תסמין שברצונך לזהות חייב להופיע בדאטהסט האימון, עם אלפי דוגמאות מתויגות. מאחר והפרויקט עוסק בהתאמה גמישה של תמונה לתיאור רפואי, ולא בזיהוי מתוך מספר קטגוריות סגור, הגישה הזו לא התאימה.

נשקל גם שימוש ב־**ViT (Vision Transformer)h** רגיל, שמציע תוצאות מצוינות במגוון משימות חזותיות. אבל גם כאן נדרש fine-tuning, ואין תמיכה ישירה בהשוואת תמונה לתיאור טקסטואלי – כלומר, אין דרך "לשאול" את המודל האם התמונה נראית כמו cellulitis, rash או wound. כל שאלה כזו מחייבת אימון מחדש או מודול עליון מותאם.

לעומת זאת, המודל **BioMedCLIP** בנוי מראש בדיוק לתרחיש הזה: הוא מקבל תמונה, ומערך של תיאורים טקסטואליים (labels), ומשווה ביניהם באופן zero-shot – בלי צורך באימון מחדש. כלומר, ניתן לשלוח תמונה חדשה של נגע בעור, יחד עם 100 תיאורים של מחלות אפשריות (למשל: "this is a photo of eczema", "this is a photo of herpes", וכו') – והמודל יחזיר הסתברות לכל אחד מהם, לפי מידת ההתאמה בין התמונה לתיאור.

בנוסף, BioMedCLIP עבר אימון משולב על טקסטים רפואיים (PubMed, UMLS) ותמונות קליניות, ולכן הוא יודע "לזהות" מונחים רפואיים גם ברמה סמנטית, לא רק ויזואלית. הוא תומך בתוויות המבוססות על מזהי UMLS, כך שהפלט של המודל מתיישר עם שאר רכיבי המערכת (למשל הצלבה עם טקסט שנכתב על ידי המשתמש).

לצד הדיוק, הגמישות והחיבור ל־UMLS, יש כאן יתרון חשוב נוסף: אין צורך לאמן את המודל או לאסוף דאטהסט חדש. זה חוסך זמן, מאמץ רגולטורי, והתמודדות עם סוגיות פרטיות. כל תוספת של מחלה או מצב חדש פשוט דורשת הוספת משפט נוסף לרשימת התיאורים – ללא צורך בשינוי המודל.

לסיכום, בעוד שמודלים כמו ResNet או ViT רגיל מתאימים למשימות סגורות של סיווג חזותי, הבחירה ב־**BioMedCLIP** אפשרה גישה גמישה, חכמה וממוקדת־רפואה, שמתאימה במיוחד לסביבת משתמשים שמעלים תמונה ספונטנית מתוך מצב חירום רפואי.

### חלופות למסדי נתונים

במערכת מורכבת כמו זו שנבנתה בפרויקט, מסד הנתונים הוא הרבה מעבר למקום אחסון – הוא תשתית שמכתיבה את אופן החשיבה, השאילתות והקשרים שבין חלקי המידע. במקרה שלנו, יש שני סוגי מידע שונים לחלוטין: מצד אחד נתונים טבלאיים, כמו משתמשים, תוצאות ניתוח, תאריכים והיסטוריית בקשות – ומצד שני ידע רפואי סמנטי מקושר, כמו הקשרים בין סימפטומים למחלות, או בין תסמינים לתרופות. לא סביר ולא יעיל לנסות לנהל את שניהם באותו מסד.

בהתאם לכך התקבלה ההחלטה להשתמש בשני מסדי נתונים משלימים:

* **PostgreSQL** עבור הנתונים הרלציוניים.
* **Neo4j** עבור ניהול הגרף הסמנטי של הידע הרפואי.

לפני כן נבחנו מספר חלופות.

**PostgreSQL מול MySQL או MongoDB**

בבחירת מסד רלציוני, נשקלה גם האפשרות להשתמש ב־MySQL. זהו מסד ותיק, נפוץ ומבוסס קהילה, אך חסר תכונות מתקדמות ש־PostgreSQL תומך בהן כברירת מחדל – כמו שאילתות מורכבות יותר, תמיכה טובה יותר ב־JSON, עבודה עם enums, טרנזקציות חכמות, ויכולת להרחיב טיפוסי נתונים. כל אלו היו רלוונטיים בדיוק לסוג המידע שנשמר – במיוחד כשמדובר בתשומות דינמיות של משתמשים, תוויות רפואיות, ותיעוד פלטים.

MongoDB, שנחשב למסד NoSQL מבוסס־מסמכים, גם הוא נשקל – בעיקר בשל הפשטות והגמישות שלו. הוא טוב מאוד לשמירה של נתונים חופשיים ולא מוגדרים, אך בפרויקט הזה נדרשה שליטה קשיחה על מבני הנתונים, מפתחות זרים, טבלאות קשר, ויכולת להריץ שאילתות מורכבות ומבוקרות. MongoDB פשוט לא התאים לדרישה לרמות גבוהות של עקביות ושלמות.

**לכן** PostgreSQL נבחר – בזכות האיזון בין ביצועים, יציבות, גמישות טכנית, ויכולת לעבוד עם מידע רפואי סטנדרטי, כולל מזהי CUI, enums רפואיים ומבני מידע מורכבים.

**Neo4j מול מסדי גרף אחרים או פתרונות חלופיים**

כשנדרש היה לנהל את כל הידע הסמנטי של המערכת – כלומר את הקשרים בין סימפטומים, מחלות, תרופות, ואפילו תתי־קשרים כמו "מוחמר על ידי", "מתרחש יחד עם", או "נפוץ יותר בגברים" – התברר שמסד רלציוני פשוט לא יספיק. מודל של טבלאות וקשרים אינו מסוגל לייצג ביעילות רשתות של קשרים מסוגים שונים בין אלפי מושגים רפואיים.

מסדי גרף כמו ArangoDB, OrientDB או אפילו פתרונות על בסיס ElasticSearch נשקלו, אך כולם דרשו רמות תחזוקה גבוהות יותר או לא סיפקו תמיכה מתקדמת בשאילתות סמנטיות. לעומתם, Neo4j נחשב לסטנדרט בתחום הגרפים – הוא קל להטמעה, חזק מאוד בביצועים, ומספק שפת שאילתות (Cypher) שהיא טבעית וקריאה, דבר שהפך את שילובו לקל ופשוט גם בצד הקוד וגם בצד הלוגיקה הרפואית.

בעזרת Neo4j ניתן היה לבנות רשת קשרים שבה כל מושג רפואי (צומת בגרף) מחובר למושגים אחרים לפי הקשרים מובנים – כמו קשר היררכי (is-a), סיבתי (causes), סמיכות (associated\_with) או משלים (treated\_by). כך למשל, ניתן לשאול:  
 "מהם כל הסימפטומים הקשורים למצב רפואי מסוים?" או  
 "איזה מושגים סמוכים לסימפטום X לפי UMLS?" – ולקבל תשובות מיידיות במבנה גרפי טבעי.

# 10. אפיון המערכת

### 10.1 ניתוח דרישות המערכת

מערכת MediAid נועדה לספק תמיכה ראשונית במצבי חירום רפואיים באמצעות ניתוח טקסטים רפואיים, תמונות, נתונים אישיים וגרף ידע רפואי. דרישות המערכת מחולקות לדרישות פונקציונליות ודרישות לא-פונקציונליות:

**דרישות פונקציונליות**

* קבלת מידע רפואי מהמשתמש ע"י חיפוש במאגר המסונן מתוך הUMLS שמאוחסן בPostgreSQL.
* שמירה על פרטיות באמצעות הצפנה דו-שלבית ואבטחת מידע בהתאם לתקנים מחמירים.
* זיהוי סימפטומים מתוך טקסט חופשי באמצעות עיבוד שפה טבעית (NLP באמצעות MedCAT).
* המרת סימפטומים לקונספטים רפואיים תקניים במבנה אחיד.
* טיפול בטקסטים באנגלית בשלב הראשון של הפיתוח.
* ניתוח תמונות של נגעים או פצעים חיצוניים והמרת המאפיינים הוויזואליים לרשימת סימפטומים ע"י BioMedCLIP.
* המרה לקונספטים רפואיים אחידים והתאמתם לקטגוריות גרפולוגיות ידועות.
* ייצוג הקשרים בין סימפטומים, מחלות, טיפולים וגורמים נוספים באמצעות גרף רפואי המבוסס על Neo4j.
* יבוא ישויות רפואיות מתוך בסיס נתונים רלציוני והמרתן לצמתים וקשתות בהתאם לקשרים לוגיים מוגדרים.
* שילוב אוטומטי של הסימפטומים שאותרו מתוך טקסטים ותמונות בגרף.
* חישוב ציון סיכון מבוסס על משקל של גורמי סיכון רפואיים אישיים.
* ניתוח מצב המשתמש ומתן המלצות לפעולה בהתאם לרמת הדחיפות.
* הצגת הסבר ברור לכל המלצה לצד חיזוק מבוסס-נתונים מתוך הגרף הרפואי.

**דרישות לא פונקציונליות**

* כל המידע הרפואי שנשמר ומעובד במערכת יוצפן ברמה גבוהה, כולל הצפנת מידע במנוחה ובמעבר.
* גישת משתמשים למידע תאובטח באמצעות טוקני JWT.
* המערכת תעבד שאילתות טקסט ותמונות רפואיות בזמן אמת, תוך פחות מ־5 שניות לתגובה.
* המערכת תספק זמני תגובה מהירים גם תחת עומס של משתמשים מרובים.
* המערכת תפותח בארכיטקטורה מודולרית, כך שניתן יהיה להוסיף רכיבים או לשפרם מבלי להשפיע על שאר המערכת.
* הממשק יהיה ידידותי למשתמש, רספונסיבי ונתמך באנגלית.
* המערכת תפותח בטכנולוגיות עדכניות: Spring Boot (Java)gבצד השרת, PostgreSQL ו־Neo4j למסדי נתונים, Python למודולי הבינה המלאכותית.

### 10.2מודולי המערכת

**מודול חילוץ סימפטומים (NLP ו־Computer Vision)**

מודול זה ממומש בPython, והוא אחראי על עיבוד ראשוני של הקלטים שהמשתמש מספק — טקסט חופשי המתאר תסמינים, או תמונה של נגע חיצוני. המטרה המרכזית היא לזהות ולחלץ סימפטומים רפואיים, תוך נורמליזציה למונחים רפואיים תקניים מאונטולוגיית UMLS.

עבור טקסטים, המודול משתמש במודלים רפואיים (MedCAT) המיועדים לאנליזה סמנטית של תיעוד קליני. התהליך כולל:

* **Tokenization** ו־**NER** ,תוך סינון סימפטומים בלבדלזיהוי ישויות רפואיות כגון תסמינים, תרופות ומחלות.
* נורמליזציה של הישויות לקונספטים מתוך **UMLS** (Unified Medical Language System).

עבור תמונות של נגעים חיצוניים (כגון פריחה, כוויות או דלקת), המודול משתמש ברשתות נוירונים מאומנות לזהות תסמינים חזותיים. הפלט כולל רשימת תסמינים מנורמלים (כגון "erythema", "ulceration"), עם confidence score לכל זיהוי.

מתבצע נסיון התאמה בין רשימת סימפטומים אפשרית לבין התמונה המתקבלת.

התוצאה היא מבנה חצי־מעובד של סימפטומים מנורמלים, הנשלח למודולים הבאים להמשך עיבוד.

**מודול ניהול מידע רפואי**

מודול זה מרכז את כלל המידע הקבוע על המשתמש, אשר נאסף מראש:

* מחלות עבר ודיאגנוזות קיימות
* תרופות שנלקחות באופן קבוע
* גורמי סיכון קבועים (BMI, עישון, סוכרת, היסטוריה משפחתית וכו')

המידע מסונן ראשית מתוך מסמכי הUMLS לקטכוריות באמצעות מזהיי הקטגוריה - TUI, ונשמר בצורה מסוננת בPostgreSQL. המידע שמור ביחס של many to many ביחס למשתמשים.

**מודול גרף הידע הרפואי (Medical Knowledge Graph)**

מודול זה מממש את הלב האנליטי של המערכת: גרף רפואי מבוסס Neo4j שמחבר בין מחלות, תסמינים, תרופות, תופעות לוואי, בדיקות והנחיות טיפוליות. הקשרים מיובאים ממסדי נתונים קליניים ונתמכים על ידי ההיסטוריה הרפואית של המשתמש.

למשל:  
 Chest pain ← side effect of ← Nitroglycerin  
 Chest pain ← symptom of ← Angina  
 Angina → treated by → Beta blockers

באמצעות אלגוריתמים גרפיים (כגון shortest path, pattern matching ו־subgraph traversal), המערכת מסוגלת:

* לזהות האם תסמין חדש הוא תופעת לוואי של תרופה קיימת
* למצוא קשר בין סימפטום למחלה קודמת
* להציע מסלול טיפול מותאם

**מודול המלצות טיפוליות**

בהתבסס על הקלט המעובד מהמודולים הקודמים, מודול זה אחראי להפקת ההנחיות. התהליך כולל:

* ניתוח מסלול הגרף בין המידע הרפואי של המשתמש לבין התסמין שזוהה
* זיהוי קונפליקטים (למשל: תופעת לוואי מול מחלה קיימת)
* הפקת המלצה בהתבסס על מדריכים רפואיים, אלגוריתמים קליניים וניסיון מצטבר

הפלט כולל המלצות טיפול כמו:

* תרופה מוצעת (כולל מינון ותדירות)
* בדיקת מעבדה רלוונטית.
* המלצה לפנייה לרופא, כולל תחום מומחיות (קרדיולוגיה, עור וכו')
* טיפולים לא תרופתיים (לדוגמה: שתייה מרובה, מנוחה)

**מודול ממשק משתמש**

מודול זה מציג למשתמש את המידע המעובד וההמלצות בפורמט ברור ואינטואיטיבי. תכונות עיקריות:

* יכולת הצגת **גרף חזותי** של הקשרים: מהיסטוריית המשתמש דרך התסמינים ועד לטיפול
* **חיווי ויזואלי** של רמת דחיפות (למשל: סימון באדום במקרה חירום רפואי)

### 10.3 אפיון פונקציונלי

המערכת כוללת מספר פונקציות מרכזיות, שכל אחת מהן תורמת לשלב שונה בתהליך קבלת מידע רפואי, זיהוי סימפטומים, ויצירת המלצות טיפוליות מותאמות אישית. להלן תיאור הפונקציות העיקריות, כולל פירוט של הקלט, העיבוד והפלט בכל שלב:

**עיבוד טקסט רפואי (NLP)**

**תיאור:** פונקציה זו מופעלת כאשר המשתמש מזין תיאור מילולי של תסמינים או מצב רפואי כללי. המערכת מבצעת ניתוח שפה טבעית (NLP) כדי לזהות רכיבים רפואיים חשובים, כולל סימפטומים, מונחים רפואיים, תרופות, והקשרים ביניהם.

* **קלט:** טקסט חופשי המתאר תסמינים, תיאור מצב או שאלות רפואיות.
* **עיבוד:** זיהוי ישויות רפואיות (NER), נרמול למונחים תקניים מתוך אונטולוגיית UMLS, ניתוח הקשרים סמנטיים בין הישויות.
* **פלט:** אובייקט נתונים המכיל רשימת סימפטומים מנורמלים, תרופות, מחלות ורכיבי מידע נוספים לזיהוי בעיה רפואית.

**עיבוד תמונה (Computer Vision)**

**תיאור:** פונקציה זו מאפשרת למשתמש להעלות תמונה (כגון צילום של מסמך רפואי או נגע עור) לצורך חילוץ מידע רלוונטי.

* **קלט:** קובץ תמונה (JPG, PNG וכדומה).
* **עיבוד:** זיהוי תסמינים חזותיים בעזרת מודלים של רשתות נוירונים.  
  תוך נסיון מציאת התאמה לרשימת סימפטומים אפשריים שהוכנה מראש.
* **פלט:** רשימת סימפטומים חזותיים מנורמלים עם ציון רמת ודאות.

**ניהול מידע רפואי קבוע**

**תיאור:** פונקציה זו מנהלת את המידע הקבוע של המשתמש, כגון מחלות עבר, תרופות, אלרגיות וגורמי סיכון ומאפשרת עדכון פרטים ושימוש במידע הרפואי על המשתמש.

* **קלט:** פרטי המשתמש מתוך מאגר נתונים (PostgreSQL ו־Neo4j).
* **עיבוד:** שליפת המידע, סידור והכנתו לשימוש במודולים האנליטיים.
* **פלט:** אובייקט מידע רפואי מלא המכיל פרופיל רפואי מעודכן.

**ניתוח והסקת מסקנות בגרף הידע הרפואי**

**תיאור:** פונקציה זו מקבלת את הסימפטומים ואת המידע הרפואי הקיים, ומנתחת אותם בתוך גרף הידע הרפואי במערכת.

* **קלט:** רשימת סימפטומים מנורמלים ואובייקט פרופיל רפואי.
* **עיבוד:**
  + איתור קשרים בין סימפטומים, מחלות ותרופות בגרף Neo4j.
  + זיהוי תופעות לוואי, מחלות רקע, ואפשרויות טיפול רלוונטיות.
  + הפעלת אלגוריתמים גרפיים (כגון חיפוש מסלולים וקלאסיפיקציות).
* **פלט:** רשימת תובנות רפואיות והמלצות ראשוניות.

**הפקת המלצות טיפוליות**

**תיאור:** פונקציה זו ממירה את המידע המעובד להמלצת טיפול מותאמת אישית.

* **קלט:** תובנות רפואיות ומידע מהגרף הרפואי.
* **עיבוד:**
  + ניתוח תנאי סף והקשר בין נתונים.
  + בחירת המלצות טיפוליות, בדיקות נוספות, הפנייה לרופא או הוראות טיפול.
* **פלט:** המלצות מפורטות בפורמט טקסטואלי ומכין נתונים להצגה בממשק המשתמש.

**ממשק משתמש להצגת תוצאות**

**תיאור:** הצגת המידע המעובד למשתמש בצורה נגישה וברורה.

* **קלט:** המלצות טיפוליות, פירוט סימפטומים, תובנות נוספות.
* **עיבוד:** עיצוב המידע לשפה פשוטה, הצגת גרפים, חיווי רמות דחיפות.
* **פלט:** ממשק ויזואלי עם טקסט ברור, דיאגרמות והודעות.

### 10.4 ביצועים עיקריים

* דיוק בזיהוי וסיווג סימפטומים מתוך טקסט חופשי: מעל 85% (במקרים רפואיים מורכבים ובהתבסס על מודלים רפואיים מתקדמים).
* דיוק בזיהוי מאפיינים רפואיים מתמונה (נגעים חיצוניים): מעל 80% בממוצע, תלוי באיכות התמונה.
* זמן תגובה ממוצע לניתוח טקסט וסימפטומים: עד 2 שניות.
* זמן תגובה ממוצע לניתוח תמונה והפקת סימפטומים: עד 5 שניות, כולל עיבוד רפואי.
* דיוק בהתאמת המלצות טיפול אישיות לפי נתוני המשתמש (מחלות ותרופות קיימות): מעל 90%.
* יכולת זיהוי מצבי חירום רפואיים קריטיים מתוך קלט המשתמש: מעל 95% (בהתבסס על כלל הנתונים הזמינים).
* עמידות למקרים בהם הנתונים חלקיים או סותרים: המערכת מזהה ומדווחת על אי-בהירות עם המלצה למעקב רפואי.
* שמירת פרטיות המשתמש: אימות התחברות באמצעות JWT עיבוד נתונים מוצפן ואנונימי לאורך כל התהליך.

### 10.5 אילוצים

* המערכת פועלת כרגע באנגלית בלבד.
* טקסט חופשי חייב להיות כתוב בשפה ברורה יחסית, עם מינימום טעויות כתיב שמונעות עיבוד תקין.
* תמונות שהמשתמש מעלה צריכות להיות באיכות מספקת (רזולוציה מינימלית, תאורה סבירה) כדי לאפשר זיהוי מדויק של נגעים או פצעים.
* המערכת אינה מחליפה ייעוץ רפואי מקצועי, ומספקת המלצות בלבד. יש להבהיר זאת למשתמש בממשק.

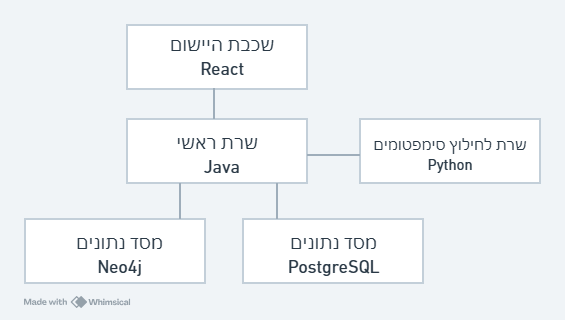
# 11. תיאור הארכיטקטורה

#### 11.1 ארכיטקטורת הפתרון המוצע – Top-Down Level Design

המערכת בנויה במודל היררכי לפי עיקרון ה-Top-Down, המאפשר חלוקה לרמות שונות של רכיבים ותהליכים, מהכללי לפרטני, באופן שמבטיח גמישות, ניהוליות ויכולת הרחבה עתידית:

* **רמת יישום כוללת (מערכת על)** זוהי השכבה העליונה הכוללת את כל תפקודי המערכת: קבלת קלט (טקסט ותמונה), עיבוד וניתוח, חיבור לגרף ידע, הפקת המלצות, ממשק משתמש להצגה.
* **רמת שירותי עיבוד והסקת מסקנות** תת-מערכת זו כוללת מודולים של NLP לעיבוד טקסט, OCR לזיהוי תווים בתמונה, מנוע ניתוח תמונה, ומודול AI המנתח ומסיק מסקנות רפואיות בהתבסס על גרף הידע.
* **רמת אחסון וניהול מידע** בסיסי נתונים רלציוני (למשל PostgreSQL) לאחסון מידע משתמש, תיעוד ותוצאות, וגרף ידע (למשל Neo4j) לייצוג הידע הרפואי והקשרים בין ישויות.
* **רמת תקשורת והצגה** רכיבי שרת-לקוח המאפשרים תקשורת בין ממשק המשתמש (לדפדפן או אפליקציה) לבין השרת, כולל APIs להזרמת נתונים ושמירת תוצאות.

#### 11.2 תיאור הרכיבים בפתרון



פירוט בסעיף 16.

#### 11.3 תהליכי מערכת ההפעלה

1. תהליך קבלת תמונות ונתוני טקסט מהמשתמש.
2. תהליך הרצת מודול ניתוח תמונה (Computer Vision) או עיבוד NLP.
3. תהליך עדכון מסדי הנתונים - המערכת מפעילה תהליכים לכתיבה וקריאה למסדי PostgreSQL ו-Neo4j, כולל ניהול טרנזקציות, אינדוקס ותקשורת בין שרתים.
4. תהליך הפקת המלצות ופתרון בעיות רפואיות

#### 11.4 ארכיטקטורת רשת

המערכת בנויה בתצורת Web App - תקשורת בין שרת ללקוח באמצעות REST API.

#### 11.5 תיאור פרוטוקולי התקשורת

פרוטוקולי התקשורת בפרויקט מבוססים על HTTP/HTTPS לתקשורת בין הלקוח לשרת, ומערכת REST API להעברת נתונים. הפרוטוקולים מבטיחים אמינות, אבטחה ותגובה מהירה במערכת.

#### 11.6 תצורת שרת - לקוח

התקשורת בין השרת ללקוח מתבצעת באמצעות webAPI כאשר הגישה לאפליקציה זמינה מכל מכשיר web. הלקוח שולח בקשות, השרת מעבד את הנתונים, מעדכן מסד נתונים וכו', ומחזיר את תוצאות העיבוד ללקוח. צד הלקוח מציג את התוצאה בצורה ידידותית.

# 12. תהליכי אבטחת מידע במערכת

מערכת MediAid נועדה לטפל במידע רפואי רגיש, ולכן נדרש בה מערך אבטחה מקיף ורב-שכבתי. האבטחה במערכת מבוססת על שילוב של הצפנת מידע, אימות משתמשים, ניהול גישות, והגנה על תקשורת, תוך הקפדה על עקרונות אבטחת מידע מודרניים ותקניים.

קיימת תשתית להצפנה באמצעות אלגוריתם AES-256, המספק רמת הגנה גבוהה במיוחד. כל נתון רגיש עובר תהליך של הצפנה עם IV אקראי, תוך שימוש בשירותים ייעודיים להצפנה ופענוח. מפתחות ההצפנה נשמרים בצורה מאובטחת בקובץ keystore מוצפן, ונמצאים בנפרד מהמידע עצמו. המערכת אף תומכת ביצירה, ייצוא, ייבוא ומחיקה מאובטחת של מפתחות לפי צורך.

אימות המשתמשים מתבצע באמצעות טוקנים מסוג JWT, שמאפשרים למערכת לאמת כל בקשה בצורה בטוחה ומהירה. הטוקנים מכילים מידע מוצפן על זהות המשתמש ותוקפם מוגבל בזמן. לצד זאת, נעשה שימוש באלגוריתם BCrypt עם salt אקראי להגנה על סיסמאות, והמערכת שומרת רק את הגיבוב המוצפן, ולא את הסיסמה עצמה.

גם מסד הנתונים בנוי תוך שימת דגש על אבטחה. מזהים ייחודיים מנוהלים באמצעות UUID במקום מזהים רציפים ושדות מסוימים עשויים להיות מוצפנים ברמת שדה באופן אוטומטי ושקוף לקוד היישום. בכך נשמרת ההגנה גם במקרה של פריצה למסד הנתונים.

# 13. למידת מכונה

בפרויקט אין התייחסות ישירה ללמידת מכונה, אם כי יש שימוש במודלים MedCAT ו-BioMedCLIP המבוססים על למידת מכונה.

# 14. ניתוח ותרשימי Use Case / UML

#### 14.1 תיאור ה-Use Cases העיקריים של המערכת

**אלגוריתם ראשי - ניתוח רפואי משולב**

המערכת מבוססת על אלגוריתם משולב לניתוח נתונים רפואיים:

1. קבלת קלט מהמשתמש (טקסט/תמונה/נתונים אישיים)

2. עיבוד הנתונים באמצעות AI מתמחה:

- טקסט: MedCAT model עם UMLS

- תמונות: BiomedCLIP model

3. חילוץ סימפטומים ומונחים רפואיים

4. השוואה עם בסיס הנתונים הרפואי (Neo4j)

5. חישוב גורמי סיכון אישיים

6. החזרת המלצות מותאמות אישית

**אלגוריתמים עיקריים:**

1. אלגוריתם ניתוח טקסט:

* שימוש במודל MedCAT לזיהוי מושגים רפואיים
* מיפוי ל-UMLS codes - CUI
* סינון לפי semantic types) T184 = סימפטומים)

2. אלגוריתם ניתוח תמונות:

* שימוש במודל BiomedCLIP
* השוואת מאפיינים חזותיים לתוויות רפואיות
* חישוב confidence scores

3. אלגוריתם חישוב סיכונים:

* איסוף גורמי סיכון מהמשתמש
* חישוב משקל לכל גורם
* יצירת ציון סיכון כולל

4. אלגוריתם המלצות טיפול (גרף מסלולים):

**Input**: סימפטומים מחולצים + היסטוריה רפואית

**Process**:

1. יצירת גרף רפואי אישי למשתמש

2. מציאת מסלולים אפשריים בגרף

3. חישוב הסתברויות למסלולי טיפול

4. דירוג המלצות לפי משקל ורלוונטיות

**Output**: המלצות טיפול מותאמות אישית

5. אלגוריתם Graph Traversal לחיפוש מסלולי טיפול:

* שימוש באלגוריתם Dijkstra עם משקלים רפואיים
* חיפוש shortest path ממחלה/תרופה → סימפטום → טיפול
* דירוג מסלולים לפי רלוונטיות ובטיחות

#### 14.2 הצגת מקרי שימוש (Use Cases) עיקריים

UC1: הרשמה

* **Actor:** משתמש חדש
* **מטרה:** יצירת חשבון וגישה למערכת
* **תהליך:** הזנת פרטים ← אימות ← יצירת JWT token← העלאת מידע רפואי ע"י שאלון אורח חיים כללי ובחירה במחלות רקע ותרופות קבועות מתוך המאגר השמור בDB

UC2: התחברות

* **Actor**: משתמש רשום
* **מטרה**: גישה למערכת
* **תהליך**: הזנת פרטים ← אימות ← יצירת JWT ← גישה למערכת

UC3: עדכון פרופיל

* **Actor:** משתמש רשום
* **מטרה:** עדכון מצב רפואי
* **תהליך:** הזנת מילות חיפוש ← חיפוש ב-DB ← החזרת תוצאות מדויקות← עדכון מידע רפואי על המשתמש

UC4: ניהול גורמי סיכון

* **Actor:** משתמש רשום
* **מטרה:** מעקב אחר בריאות אישית
* **תהליך:** הזנת נתונים ← חישוב BMI ← הערכת סיכונים

UC5: ניתוח סימפטומים מטקסט או תמונה

* **Actor:** מערכת
* **מטרה:** זיהוי סימפטומים מתיאור רפואי בטקסט חופשי או מתמונות עור
* **תהליך:** העלאת טקסט ← עיבוד MedCAT או BioMedCLIP בהתאם למה שמועלה ← החזרת סימפטומים + CUI codes

UC6: שימוש בעת חירום

* **Actor:** משתמש מחובר למערכת
* **מטרה:** קבלת הנחיות בעת חירום
* **תהליך:** העלאת תמונה או טקסט← UC5 ← השוואה עם בסיס הנתונים הרפואי← הנחיות לטיפול

UC6: ייבוא נתונים רפואיים (אדמין)

* **Actor:** מנהל מערכת
* **מטרה:** עדכון בסיס הנתונים הרפואי
* **תהליך:** טעינת קבצי UMLS ← עיבוד ← ייבוא ל-Neo4j

#### 14.3 מבני נתונים בהם נעשה שימוש

**Lists (רשימות)**

List<Map<String, Object>> validRelationships = new ArrayList<>();

List<UmlsRelationship> batch = new ArrayList<>();

List<EntityImportConfig<?>> importConfig = Arrays.asList(...)

private static final List<String> PREFERRED\_SOURCES = Arrays.asList("SNOMEDCT\_US", "MSH", "ICD10CM");

* **שימוש:** אחסון קשרים לעיבוד באצווה, רשימות קבועים, תצורות ייבוא
* **יתרון:** גישה רציפה למניעת fragment memory, גישה באינדקס (O(1

**Sets (קבוצות)**

Set<String> existingCuis = new HashSet<>();

Set<String> diseaseCuis = new HashSet<>();

Set<String> medicationCuis = new HashSet<>();

private static final Set<String> EXCLUDED\_RELATIONSHIP\_TYPES = Set.of("translation\_of", "mapped\_to");

Set<String> existingRelationships = loadExistingRelationships();

* **שימוש:** מניעת כפילויות CUI, בדיקת קיום מהירה, סינון מקורות
* **יתרון:** חיפוש/הוספה/מחיקה O(1, ייחודיות אוטומטית

**Maps**

Map<String, Object> response = new HashMap<>();

Map<String, String> diseaseTerms = new HashMap<>();

private static final Map<String, String> UMLS\_TO\_NEO4J\_RELATIONSHIPS = new HashMap<>();

Map<String, Integer> debugRelationCounts = new HashMap<>();

Map<String, Long> stats = new HashMap<>();

* **שימוש:** מיפוי מפתח-ערך, תגובות API, סטטיסטיקות, קאש
* **יתרון:** חיפוש מהיר O(1), מיפוי לוגי

**Arrays (מערכים)**

String[] fields = line.split("\\|");

String[] entityTypes = {EntityTypes.DISEASE, EntityTypes.MEDICATION, EntityTypes.SYMPTOM};

String[] relationshipTypes = {RelationshipTypes.TREATS, RelationshipTypes.INDICATES};

* **שימוש:** עיבוד קבצי UMLS, רשימות קבועים, פיצול מחרוזות
* **יתרון:** זיכרון קבוע, גישה מהירה O(1)

**StringBuilder**

StringBuilder responseMessage = new StringBuilder("Data uploaded successfully.\n");

StringBuilder queryBuilder = new StringBuilder();

queryBuilder.append("MERGE (n:").append(entityType).append(" {cui: $cui}) ");

* **שימוש:** בניית מחרוזות ארוכות, שאילתות דינמיות
* **יתרון:** יעיל לשרשור מחרוזות, מניעת יצירת אובייקטים מיותרים

**Python Collections**

**Lists**

symptoms = []

symptoms.append(symptom)

symptoms.sort(key=lambda x: x['confidence'], reverse=True)

**Dictionaries**

verified\_skin\_conditions\_umls = {

"C0041834": "Erythema",

"C0151908": "Edema"

}

* **שימוש:** ניתוח AI, מיפוי תנאים רפואיים
* **יתרון:** תחביר פשוט, גמישות גבוהה

**Enums**

public enum SmokingStatus {

NEVER(0.0, "Never smoked"),

FORMER\_LIGHT(0.2, "Former smoker - light"),

CURRENT\_HEAVY(1.0, "Current smoker - heavy");

private final double weight;

private final String description;

}

* **שימוש:** גורמי סיכון עם משקלים, סטטוסים קבועים
* **יתרון:** type safety, אין magic numbers

#### 14.4 חישוב יעילות האלגוריתם

**ניתוח טקסט:**

O(n\*m)g

* n = אורך הטקסט
* m = מספר מושגים במאגר UMLS

**ניתוח תמונות:**

O(h*w*c + L)k

* h,w,c = רזולוציה ומספר ערוצי התמונה
* L = מספר התוויות הרפואיות

**חיפוש במסד נתונים:**

* **עם Indexes:** O(log n
* **ללא Indexes:** O(n
* **Batch Processing:** O(n/b)g כאשר b = batch size

**אלגוריתם חיפוש מסלולי טיפול:**

O(V + E\*log V)h

* V = מספר הצמתים הרפואיים (מחלות, תרופות, סימפטומים)
* E = מספר הקשרים ביניהם

**מערכת המלצות מבוססת גרף:**

* **Path Finding:** O(k\*log V כאשר k = מספר המסלולים המבוקשים
* **Ranking Algorithm:** O(p\*log p כאשר p = מספר ההמלצות
* **Real-time Response:** O(1 עם caching של מסלולים נפוצים

#### 14.5 הקשרים בין היחידות השונות

משתמש → Controllers → Services → Repositories → Database

↓

Python API ← Business Logic ← Data Processing ← AI Models

#### 14.6 עץ מודולים

MediAid System

├── mediAid-server (Spring Boot)

│ ├── API Layer

│ │ ├── APIController (authentication)

│ │ ├── UserProfileController (user management)

│ │ ├── RiskFactorController (health metrics)

│ │ ├── TermsController (medical search)

│ │ ├── RecommendationController ( המלצות טיפול)

│ │ └── ImportManagementController (admin)

│ ├── Business Logic Layer

│ │ ├── UserService

│ │ ├── RiskFactorService

│ │ ├── RelationshipProcessor

│ │ ├── UmlsEntityImporter

│ │ ├── PathfindingService ( חיפוש מסלולים)

│ │ ├── RecommendationEngine (מנוע המלצות)

│ │ └── GraphAnalysisService (ניתוח גרף רפואי)

│ ├── Data Access Layer

│ │ ├── Entities (User, Disease, Medication, etc.)

│ │ ├── TreatmentPath ( מסלולי טיפול)

│ │ ├── Recommendation (המלצות)

│ │ └── Repositories (JPA + Custom queries)

│ ├── Security Layer

│ │ ├── JWT Authentication

│ │ └── Encryption Services

│ └── Database Layer

│ ├── PostgreSQL (user data, UMLS terms)

│ └── Neo4j (medical relationships + pathfinding)

└── python-project (AI Analysis)

├── text\_analyzer.py (MedCAT)

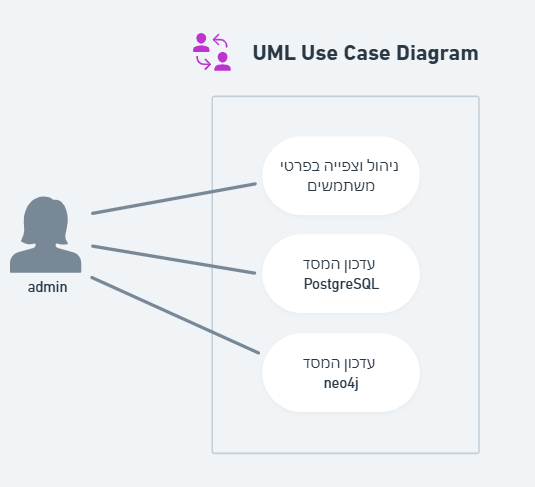
├── image\_analyzer.py (BiomedCLIP)

├── app.py (Flask API)

└── skin\_umls\_codes.py (medical labels)

#### 14.7 Use Case Diagram

# 

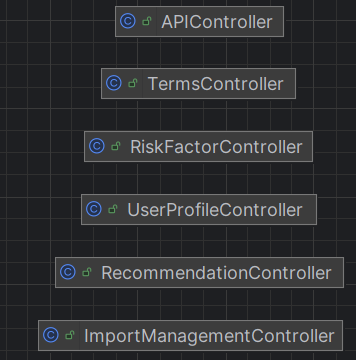


#### 14.8 רשימת Use Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **שם Use Case** | **תיאור קצר** | **Actor** | **עדיפות** |
| **UC1** | הרשמה למערכת | יצירת חשבון משתמש חדש | משתמש חדש | גבוהה |
| **UC2** | התחברות למערכת | כניסה לחשבון קיים | משתמש רשום | גבוהה |
| **UC3** | יציאה | יציאת משתמש מהמערכת | משתמש רשום | גבוהה |
| **UC4** | העלאת מידע בעת חירום | תיאור תחושות בעזרת טקסט חופשי או תמונה | משתמש רשום | גבוהה |
| **UC5** | קבלת המלצות טיפול | המלצות מותאמות אישית במצב חירום | משתמש במצב חירום | קריטית |
| **UC6** | עדכון גורמי סיכון | הזנת נתוני בריאות אישיים | משתמש רשום | בינונית |
| **UC7** | צפייה בפרופיל | הצגת נתוני המשתמש | משתמש רשום | בינונית |
| **UC8** | חיפוש מחלות | חיפוש במאגר המחלות לעדכון פרטים | משתמש רשום | בינונית |
| **UC9** | חיפוש תרופות | חיפוש במאגר התרופות לעדכון פרטים | משתמש רשום | בינונית |
| **UC10** | ייבוא נתוני UMLS | עדכון בסיס הנתונים | מנהל מערכת | נמוכה |
| **UC11** | ניהול מערכת | ניטור וניתוח מצב המערכת | מנהל מערכת | נמוכה |

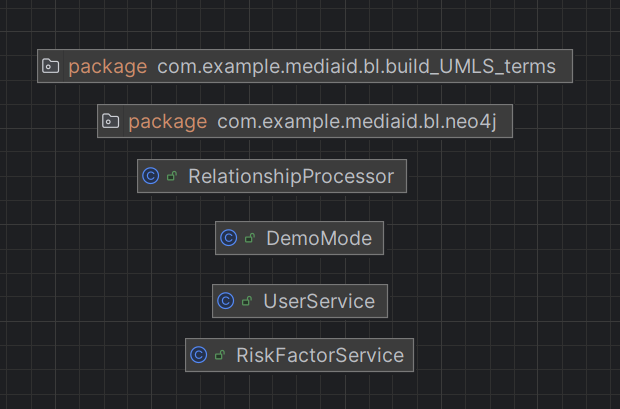
#### 14.9 תרשים UML - Class Relationships

תרשים כללי ומינימלי בשרת הראשי.

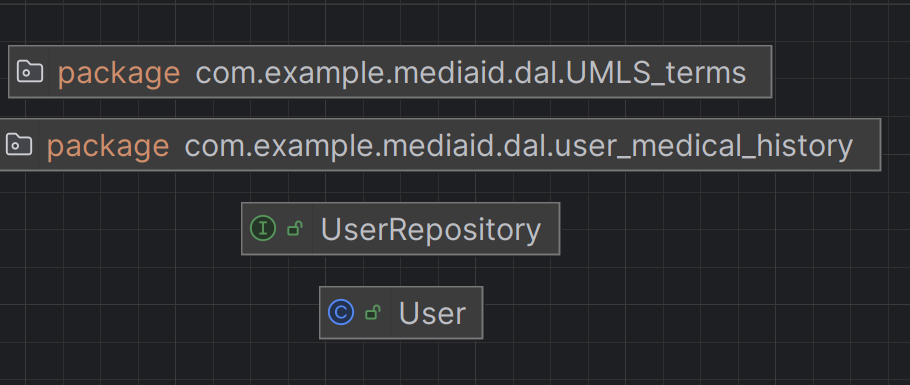


API

BL



DAL



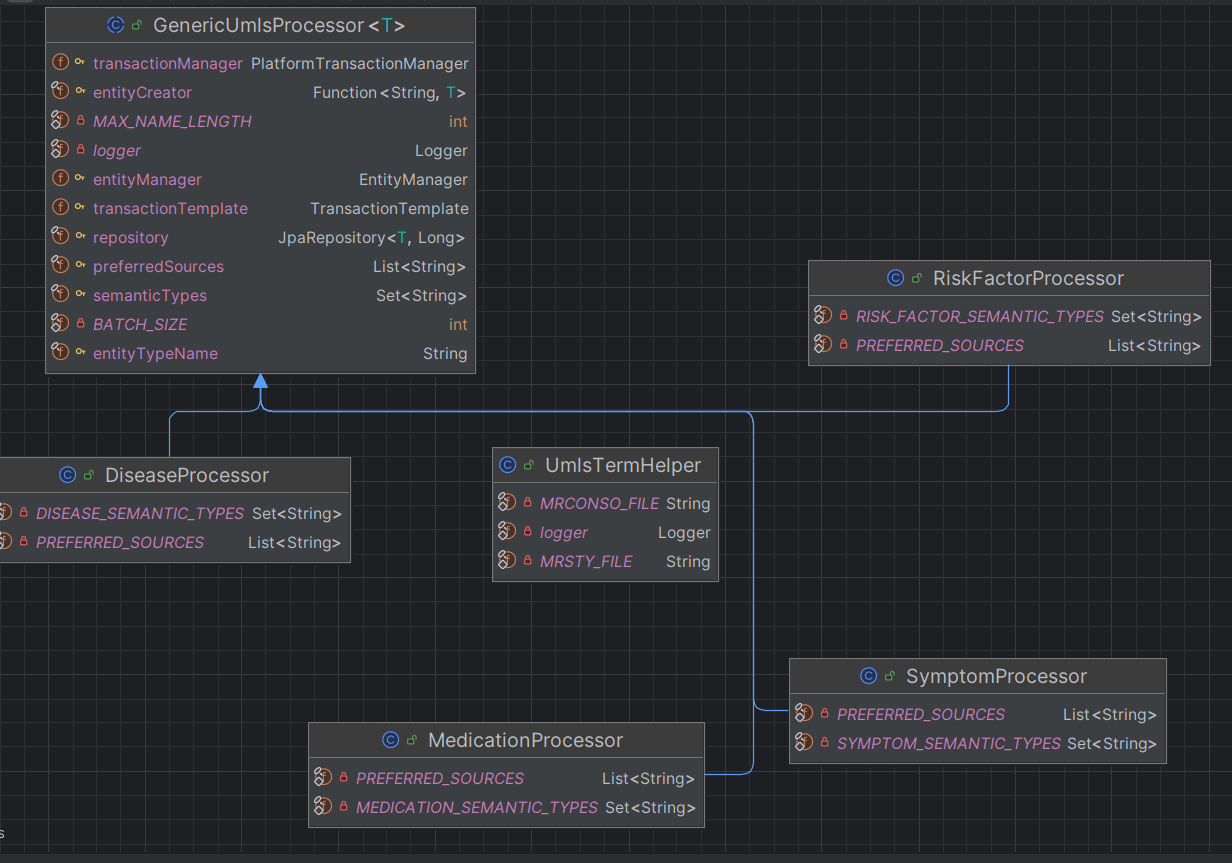
#### 14.10+14.11 Design class Diagram - תרשים מחלקות

פירוט שדות וירושות - בתיאור המחלקות



#### 14.12 תיאור המחלקות המשמעותיות

**סינון ועיבוד קבצי הUMLS לקטגוריות**

****

1. GenericUmlsProcessor<T>

* **תפקיד**: מחלקת עיבוד כללית, אב מופשט לכל שאר מחלקות ה־Processor. מבצעת תהליך עיבוד של מונחי UMLS בהתאם לסוג היישות (T).
* **קלטים**:  
  + entityCreator: פונקציה ליצירת יישות מסוג T מתוך מחרוזת.
  + preferredSources: רשימת מקורות מועדפים (כמו "MSH", "SNOMEDCT\_US").
  + semanticTypes: טיפוסי סמנטיקה של המונחים (Set<String>).
  + repository: גישה ל־DB דרך JPA.
* **פלטים**: ישויות מעובדות מסוג T, נשמרות במסד הנתונים.
* **קלטים נוספים**:  
  + משתמש בקבצי UMLS דרך UmlsTermHelper
  + קלטים נוספים דרך TransactionManager, EntityManager ו־TransactionTemplate.

2. DiseaseProcessor

* **תפקיד**: מעבד מונחים רפואיים המקוטלגים כמחלות.
* **קלטים**:  
  + DISEASE\_SEMANTIC\_TYPES: טיפוסים סמנטיים רלוונטיים למחלות.
  + PREFERRED\_SOURCES: מקורות מועדפים לזיהוי מונחי מחלה.
* **פלטים**: ישויות Disease שנשמרות בבסיס הנתונים.
* **מימוש**: יורש מ־GenericUmlsProcessor<Disease>.

3. SymptomProcessor

* **תפקיד**: מזהה ומעבד מונחים שהם תסמינים רפואיים.
* **קלטים**:  
  + SYMPTOM\_SEMANTIC\_TYPES
  + PREFERRED\_SOURCES
* **פלטים**: ישויות מסוג Symptom.
* **מימוש**: יורש מ־GenericUmlsProcessor<Symptom>.

4. MedicationProcessor

* **תפקיד**: עיבוד מונחים של תרופות מתוך קבצי ה־UMLS.
* **קלטים**:  
  + MEDICATION\_SEMANTIC\_TYPES
  + PREFERRED\_SOURCES
* **פלטים**: ישויות Medication.
* **מימוש**: יורש מ־GenericUmlsProcessor<Medication>.

5. RiskFactorProcessor

* **תפקיד**: מזהה גורמי סיכון רפואיים (Risk Factors).
* **קלטים**:  
  + RISK\_FACTOR\_SEMANTIC\_TYPES
  + PREFERRED\_SOURCES
* **פלטים**: ישויות RiskFactor.
* **מימוש**: יורש מ־GenericUmlsProcessor<RiskFactor>.

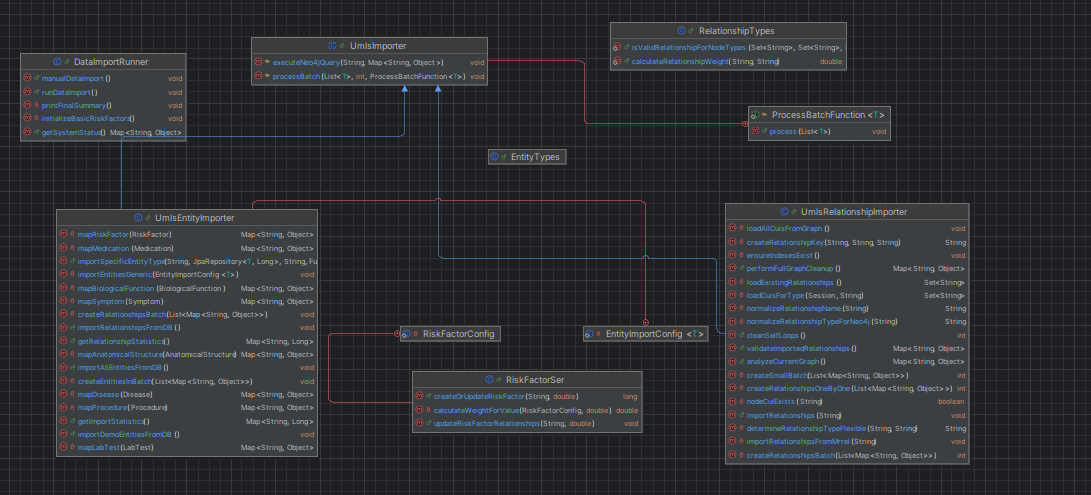
6. UmlsTermHelper

* **תפקיד**: מחלקת עזר לקריאה ועיבוד של קבצי UMLS.
* **קלטים**:  
  + MRCONSO\_FILE: נתיב לקובץ קונספטים רפואיים.
  + MRSTY\_FILE: נתיב לקובץ טיפוסים סמנטיים.
* **פלטים**: נתונים גולמיים או מסוננים לפלטפורמת העיבוד.

**זרימת מידע בין היחידות**

1. מחלקת UmlsTermHelper מספקת קבצים ונתונים למחלקות העיבוד.
2. כל מחלקת Processor יורשת מ־GenericUmlsProcessor, מקבלת את המידע הרלוונטי עבורה (עבור טיפוס סמנטי מסוים), ומבצעת עיבוד של המונחים.
3. הנתונים המעובדים נשמרים במסד הנתונים דרך JpaRepository.

**neo4j**

****

הpackage neo4j עוסקת באינטראקציה עם מסד הנתונים הגרפי Neo4j, שמכיל את רשת הקשרים בין המושגים הרפואיים מתוך ה־UMLS. מחלקות אלה מספקות פונקציונליות לטעינה, עיבוד, ולניתוח קשרים על גבי הגרף.

1. UmlsImporter

* **תפקיד**: ממשק גנרי לפעולה מול מסד Neo4j.
* **פונקציות עיקריות**:
  + executeNeo4jQuery(String, Map<String, Object>): הרצת שאילתות גמישות ב־Cypher.
  + processBatch(List<T>, int, ProcessBatchFunction<T>): עיבוד קבוצתי של תוצאות.
* **משמשת את**: UmlsEntityImporter, UmlsRelationshipImporter.

2. UmlsRelationshipImporter

* **תפקיד**: עיבוד, ניתוח וייבוא של קשרים מתוך מסד הנתונים הגרפי Neo4j.
* **פונקציות חשובות**:  
  + loadAllCUIsFromGraph(): טוענת את כל המושגים הקיימים בגרף.
  + normalizeRelationshipName(String): מנרמל שמות קשרים.
  + importRelationshipsFromMrrel(...): מייבאת קשרים מתוך קבצי MRREL.
  + validateImportedRelationships(...): בודקת תקינות לוגית של קשרים.
  + createRelationshipKey(String, String): מזהה ייחודי לקשר.
* **אחראית על**:
  + ייבוא הקשרים הגרפיים בין ישויות רפואיות.
  + בדיקה אם קשרים קיימים (nodeExists, relationshipExists).
  + בניית מבנה הגרף בפועל על סמך המידע הגולמי.

3. RelationshipTypes

* **תפקיד**: הגדרת טיפוסי קשרים מותרים במערכת בין ישויות.
* **לוגיקה עיקרית**:
  + isValidRelationshipForNodeTypes(Set<String>, Set<String>, String): מוודא אם קשר אפשרי בין שני סוגי צמתים.
  + calculateRelationshipWeight(String, String): מחשב את המשקל של קשר מסוים בין ישויות, כנראה לפי תדירות או רלוונטיות.
* **שימושים**: במהלך ייבוא הקשרים, משמשת לסיווג וסינון.

4. ProcessBatchFunction<T>

* **ממשק פונקציונלי**:
  + מגדיר את הפעולה שיש לבצע על כל קבוצת ישויות ב־batch.
  + מכיל פונקציה אחת: process(List<T>).
* **שימוש עיקרי**: מועבר למחלקת UmlsImporter.processBatch() כדי לבצע עיבוד מותאם אישית על קבוצות נתונים.

**סיכום זרימת עבודה בתוך neo4j:**

1. UmlsImporter מבצע שאילתות Cypher מול Neo4j, ומעביר את התוצאות באצוות.
2. UmlsRelationshipImporter משתמש בתוצאות כדי:
   * לנתח את הגרף
   * לייבא קשרים
   * לוודא תקינותם
3. RelationshipTypes מוודא שהקשרים בין סוגי הישויות חוקיים ומחשב את משקלם.
4. ProcessBatchFunction משמש כ Callback לעיבוד מותאם אישית לפי אצוות נתונים.

# 15. רכיבי ממשק

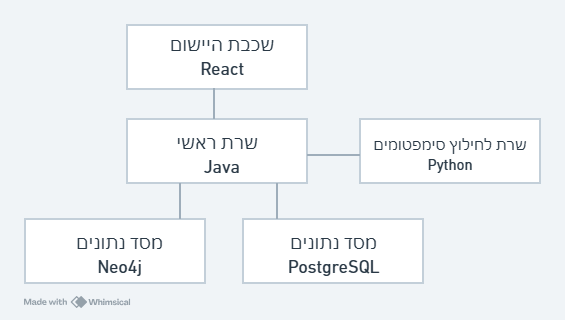
המערכת מכילה ממשק משתמש מודרני ואינטואיטיבי המבוסס על React ו-Tailwind CSS. הממשק כולל מסך התחברות אלגנטי עם אפשרות יצירת חשבון חדש, תפריט ראשי מקיף המציג סיכום בריאותי אישי וכרטיסי מידע דינמיים. אשף הגדרת משתמש מנחה את המשתמשים החדשים בארבעה שלבים: מידע בסיסי, גורמי סיכון, תרופות והיסטוריה רפואית.

מרכיב העלאת הנתונים הרפואיים תומך בקלט טקסטואלי, העלאת תמונות, צילום במצלמה והקלטת קול. מסך הפרופיל האישי מציג מידע מקיף על הבריאות כולל BMI, ציון סיכון והערכת שלמות הפרופיל. המערכת כוללת גם רכיב חיפוש תרופות ומחלות מתקדם עם השלמה אוטומטית, ומסך הנחיות טיפול מותאמות אישית.

העיצוב מדגיש נגישות וחוויית משתמש איכותית עם אייקונים ברורים, צבעים אינטואיטיביים לסטטוסי בריאות, ותגובתיות מלאה למכשירים ניידים. כל הרכיבים משולבים בצורה חלקה עם הודעות משוב מיידיות ואנימציות עדינות.

# 16. תיכון המערכת

#### 16.1 ארכיטקטורת המערכת



#### 16.2 תיכון מפורט

1. **REACT - שכבת היישום (Frontend)** זו השכבה שבה המשתמשים מקלידים את הנתונים הרפואיים (תיאורי תסמינים, העלאת תמונות וכו׳) ומקבלים את התוצאות וההמלצות. הממשק בנוי ב־React ומספק חוויית משתמש אינטראקטיבית ודינמית. שכבה זו מתקשרת עם השרת הראשי לקבלת מידע ועיבוד.
2. **שרת ראשי - (Backend מרכזי)** שרת ה-Java אחראי על ניהול הלוגיקה העסקית של המערכת, תיאום בין רכיבים שונים, אימות נתונים וניהול סשנים. השרת מטפל בבקשות ה־API מה־Frontend, מפעיל שאילתות למסדי הנתונים ומעביר את המידע לשרת עיבוד הסימפטומים.
3. **שרת חילוץ סימפטומים - PYTHON** מודול זה אחראי על עיבוד מתקדם של הנתונים שהתקבלו, בעיקר ניתוח טקסטים (NLP) וזיהוי תסמינים מתמונות (Computer Vision). השרת מבצע המרות והפשטות של התיאורים החופשיים לפורמט אחיד של סימפטומים ומחזיר את התוצאה לשרת הראשי.
4. **מסד נתונים POSTGRESQL** מסד הנתונים הרלציוני בו נשמרים כל סוגי המושגים והיישויות שחולצו ומוינו מתוך UMLS, נתוני המשתמשים, ההיסטוריה הרפואית, והנתונים המנהלתיים של המערכת. PostgreSQL משמש כבסיס לאחסון מובנה עם יכולת לבצע שאילתות מורכבות.
5. **מסד נתונים NEO4J** מסד הנתונים הגרפי בו נשמר גרף הידע הרפואי. גרף זה כולל קשרים בין סימפטומים, מחלות, תרופות וגורמי סיכון, ומאפשר ביצוע חיפושים וניתוחים מבוססי גרפים להתאמת המלצות טיפוליות מותאמות אישית.
6. **תקשורת בין הרכיבים** התקשורת בין ה־Frontend לשרת הראשי מתבצעת באמצעות REST API. בין שרת ה-Java לשרת ה-Python התקשורת מתבצעת דרך ממשק API פנימי, ומאפשרת העברת מידע לעיבוד סימפטומים בזמן אמת. מסדי הנתונים מתקשרים עם השרתים דרך פרוטוקולי גישה סטנדרטיים (JDBC עבור PostgreSQL, ו־Bolt עבור Neo4j).

#### 16.3 חלופות לתיכון המערכת

בעת תכנון מערכת מבוססת ניתוח טקסט ותמונות רפואיות, קיימות מספר גישות אפשריות לארכיטקטורה.

**1. יישום מונוליתי:** בגישה זו, כל הרכיבים (ממשק המשתמש, עיבוד, גישה למסדי נתונים) מאוחדים בשרת אחד. היתרון הוא פשטות בפיתוח והטמעה, אך עם הזמן התחזוקה נעשית קשה יותר, והרחבת המערכת מוגבלת.

**2. ארכיטקטורת מיקרו־שירותים:** כאן כל רכיב (Frontend, Backend, עיבוד NLP וכו׳) הוא שירות עצמאי המתקשר עם השאר דרך API. היתרונות כוללים גמישות טכנולוגית, תחזוקה נוחה וסקלאביליות, אך מדובר בגישה מורכבת יותר שדורשת תיאום וניהול טוב בין השירותים.

**3. ארכיטקטורת Serverless:** גישה זו מבוססת על פונקציות בענן שפועלות לפי הצורך (למשל, AWS Lambda). היא מתאימה למערכות דינמיות עם שימוש משתנה, חוסכת בעלויות תשתית ודורשת פחות ניהול, אך עלולה להיות מוגבלת מבחינת זמן ביצוע וניטור תקלות.

לסיכום, המערכת מבוססת על שילוב גמיש בין רכיבי Frontend, Backend, ו־Python – תצורה שמאזנת בין פשטות לפונקציונליות, ומתאימה היטב לפרויקט עם ניתוח רפואי מתקדם.

# 17. תיאור התוכנה

#### 17.1 סביבות עבודה

* IntelliJ - שרת Backend ראשי בJAVA.
* PyCharm - שרת Backend בPython
* Visual Studio Code - שרת Fronend בReact
* PGAdmin - PostgreSQL
* Neo4j Desktop

#### 17.2 שפות תכנות

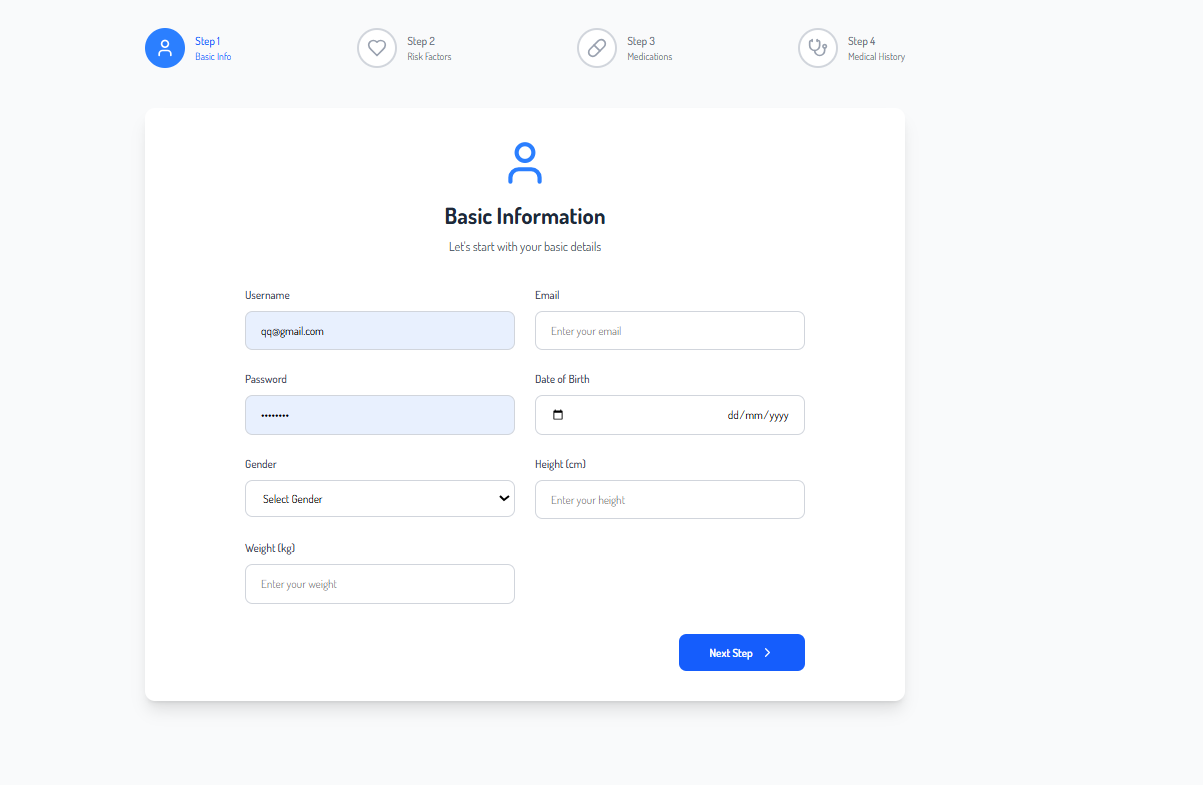
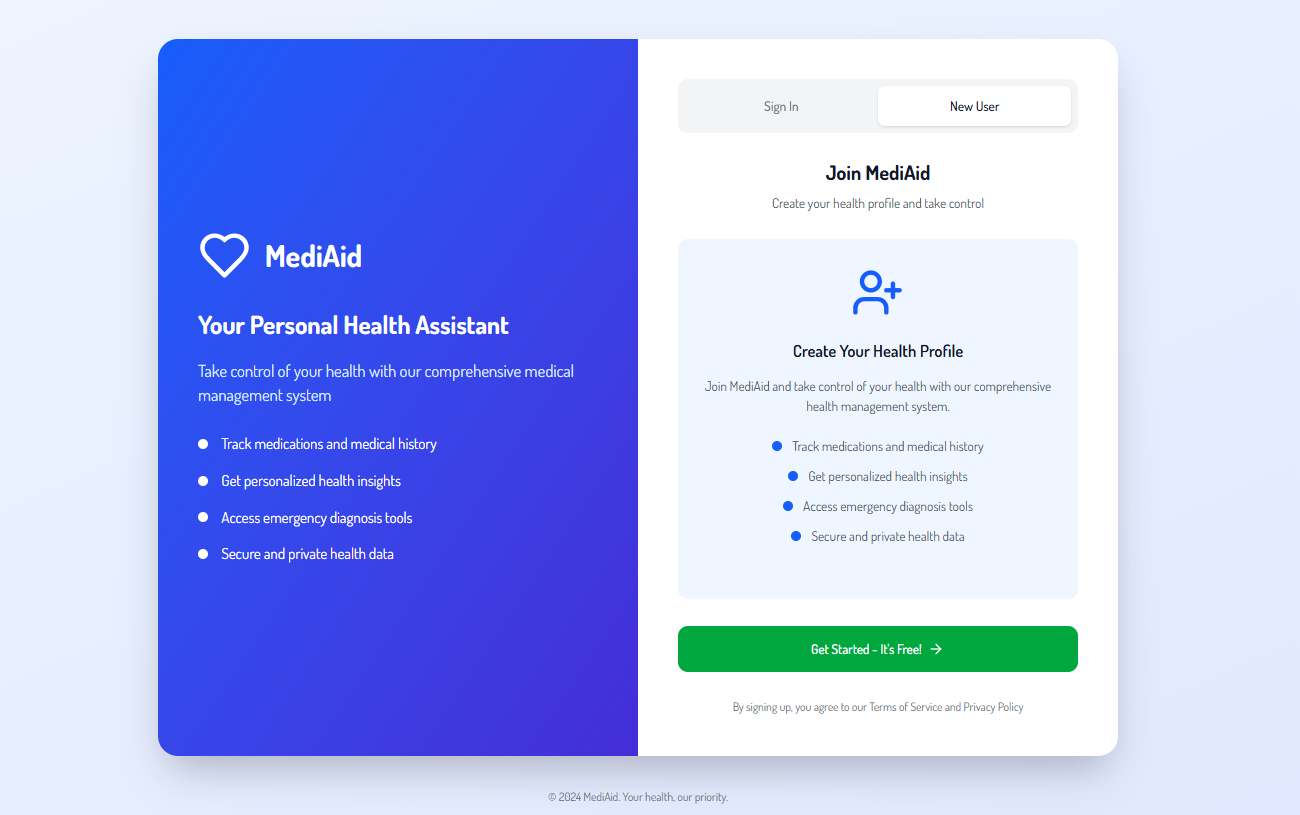
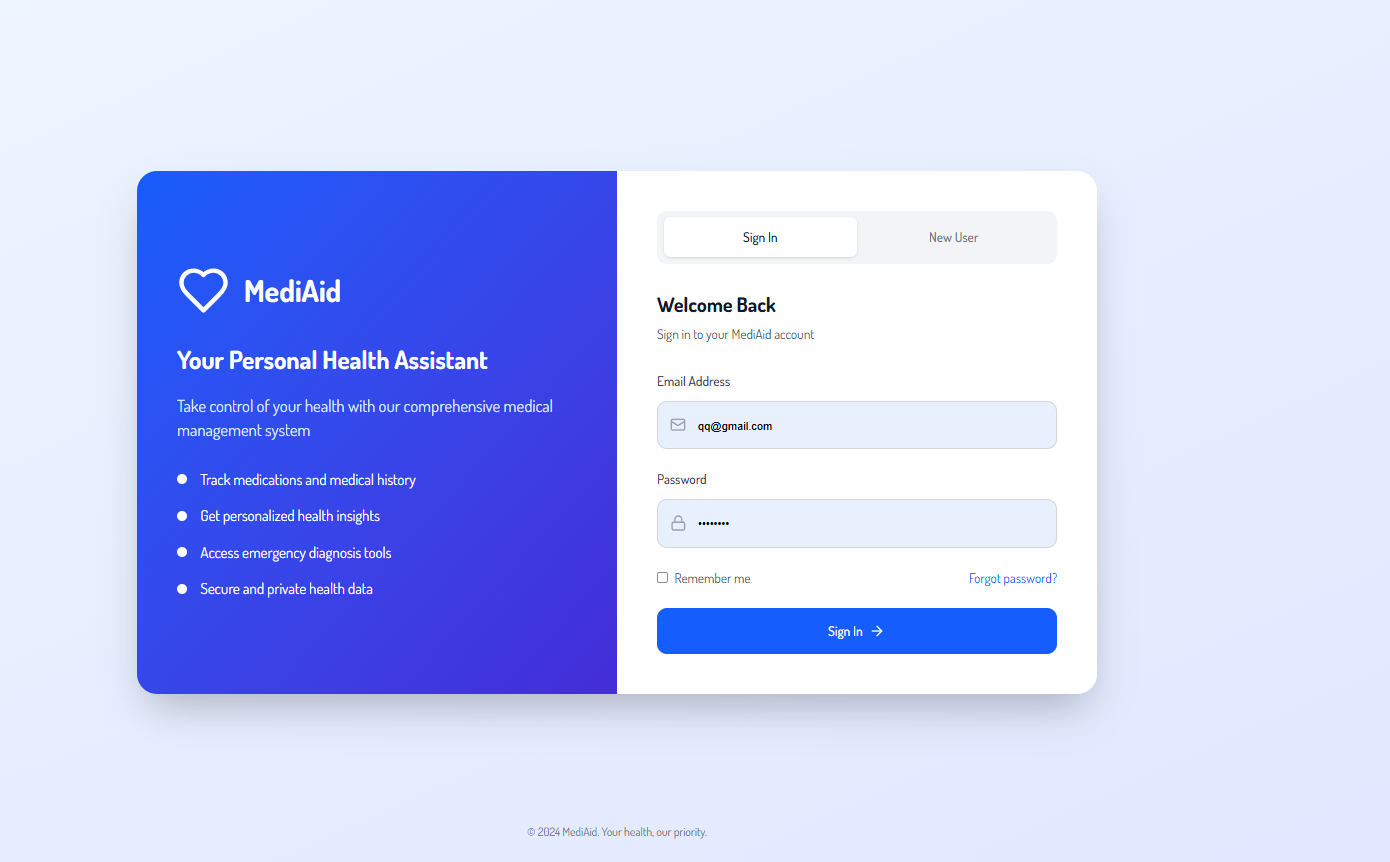
Backend: Java, Python

Frontend: React

**DataBase**: PostgreSQL, Neo4j

# 18-20. תיאור מסכים

מסך כניסה

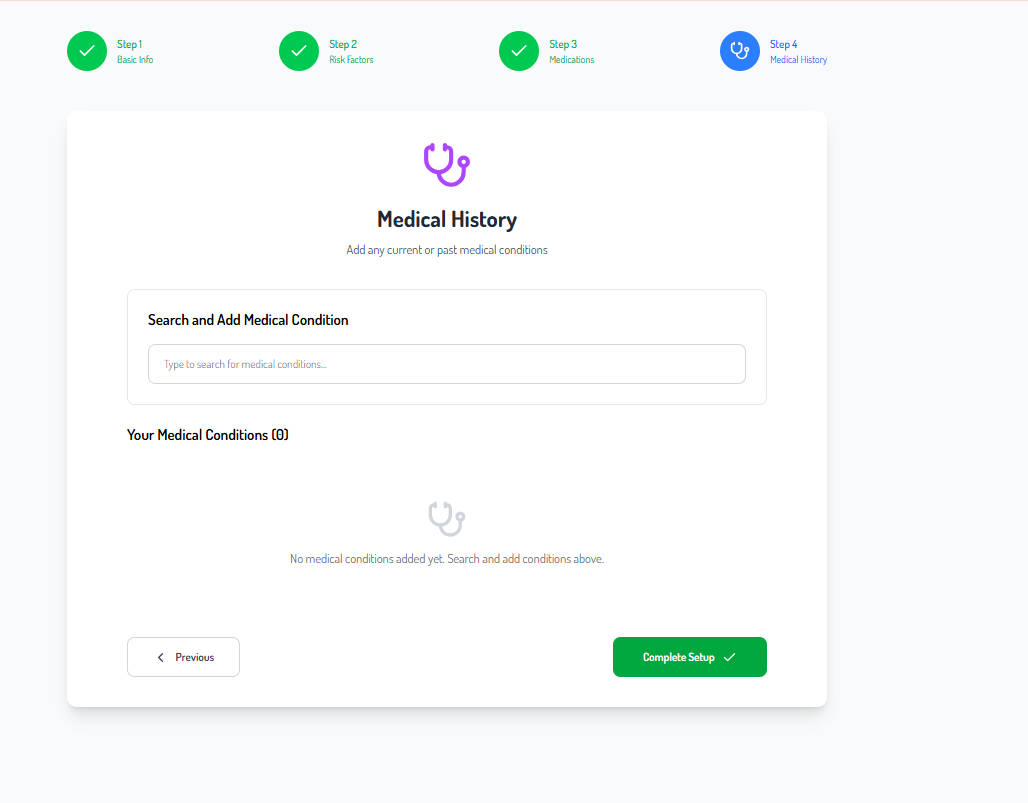
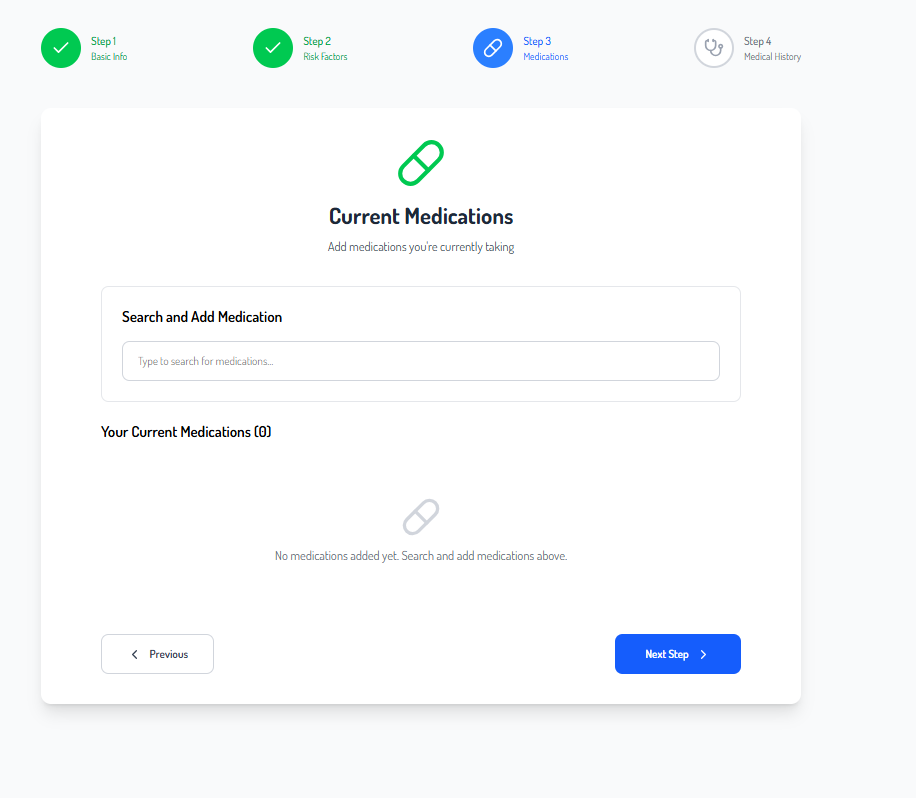
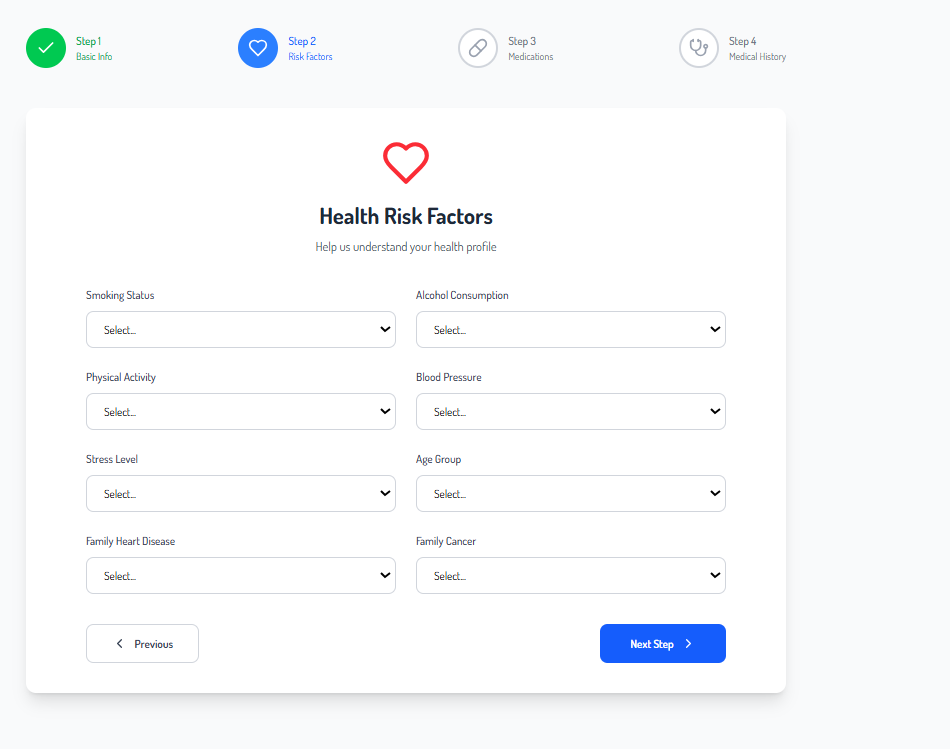


פרטים בסיסיים

הרשמה

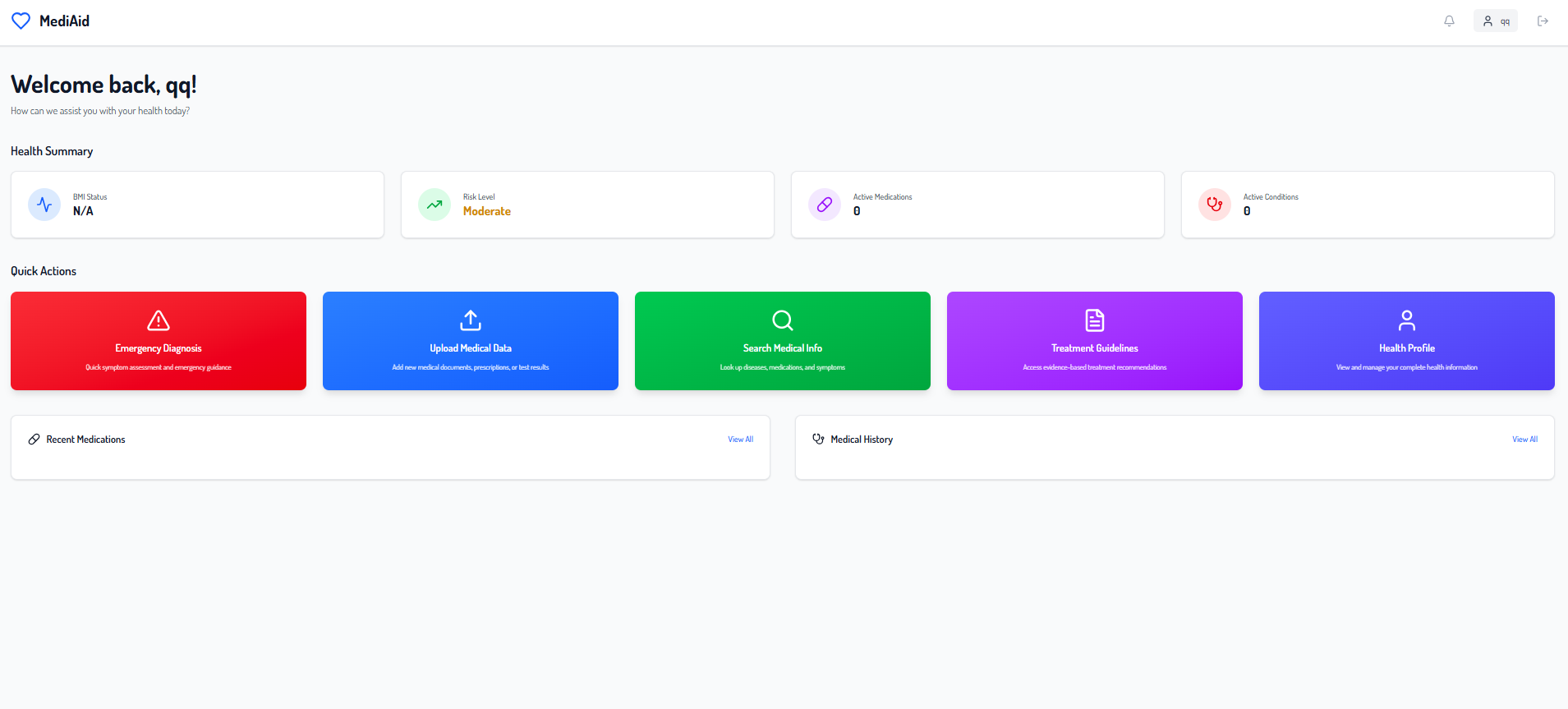
תרופות

מחלות

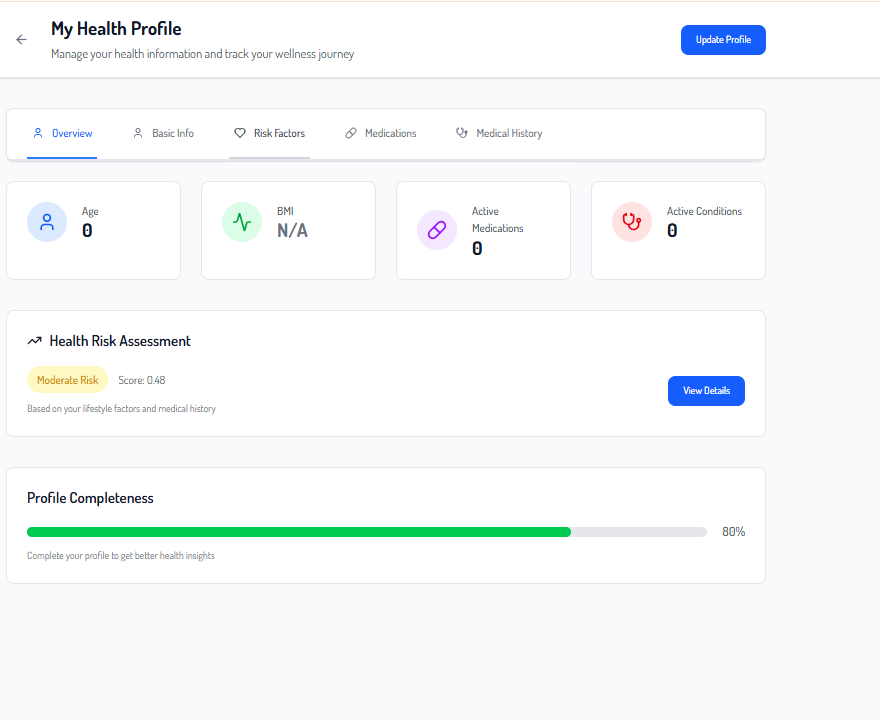


שאלון אורח חיים

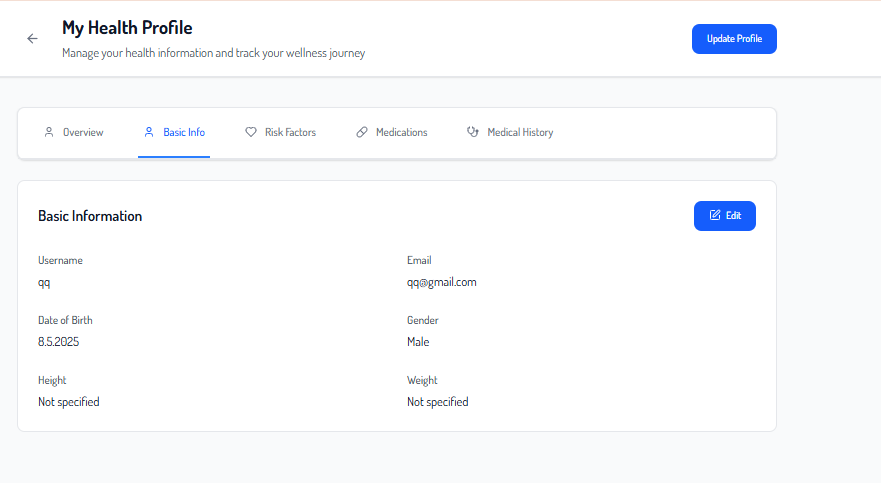
תפריט ראשי



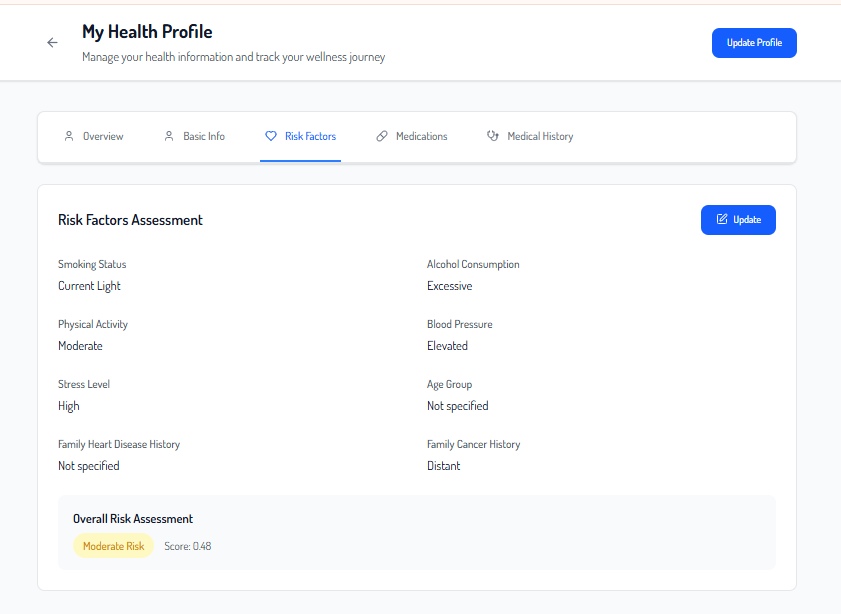
צפייה בפרופיל הכללי



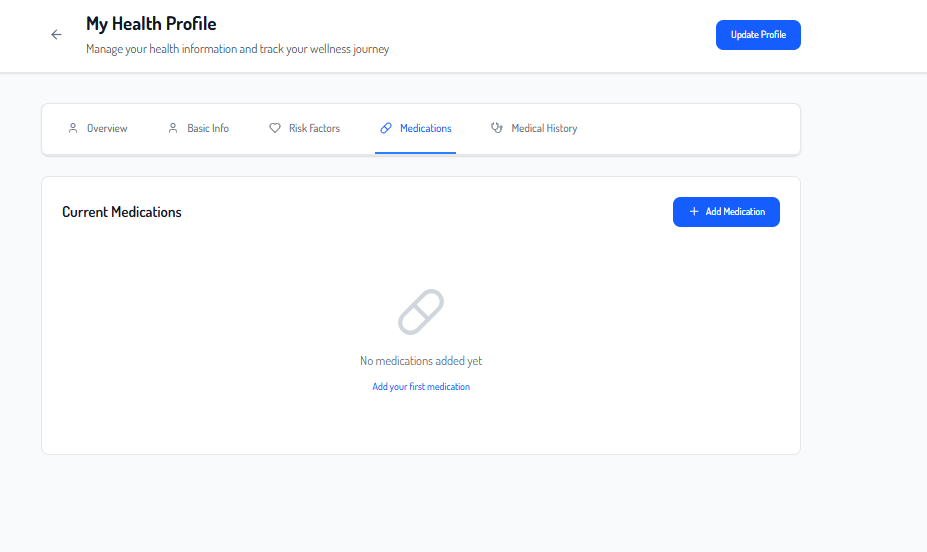
צפיה במידע אישי



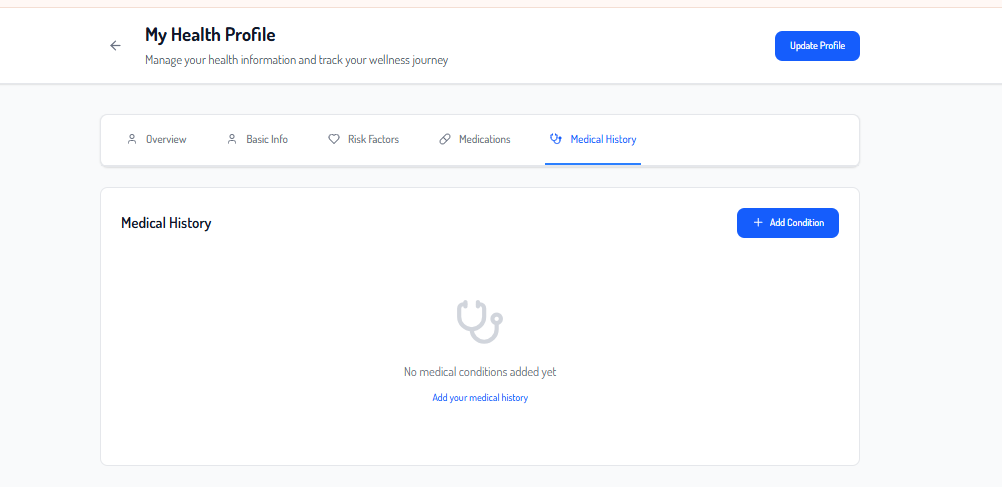
צפייה בגורמי הסיכון



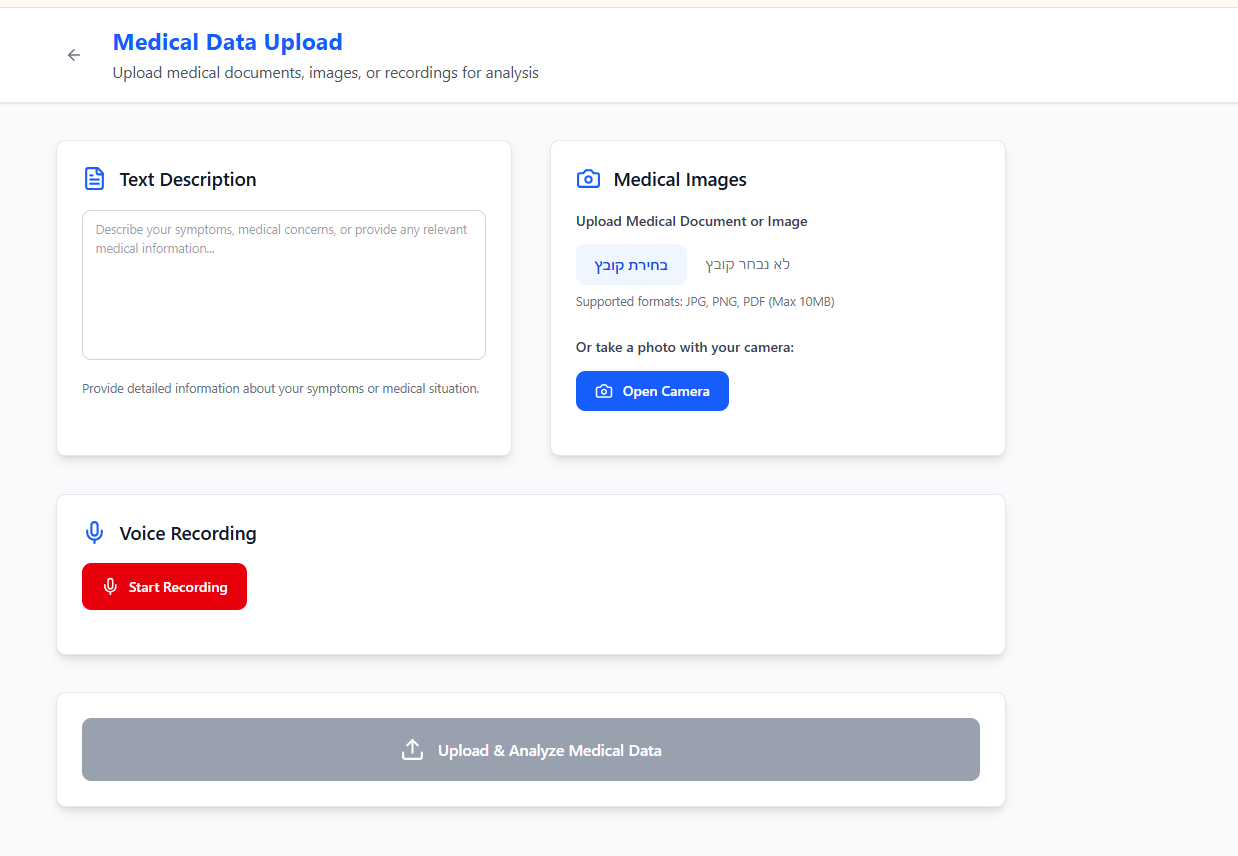
צפייה בתרופות הנלקחות

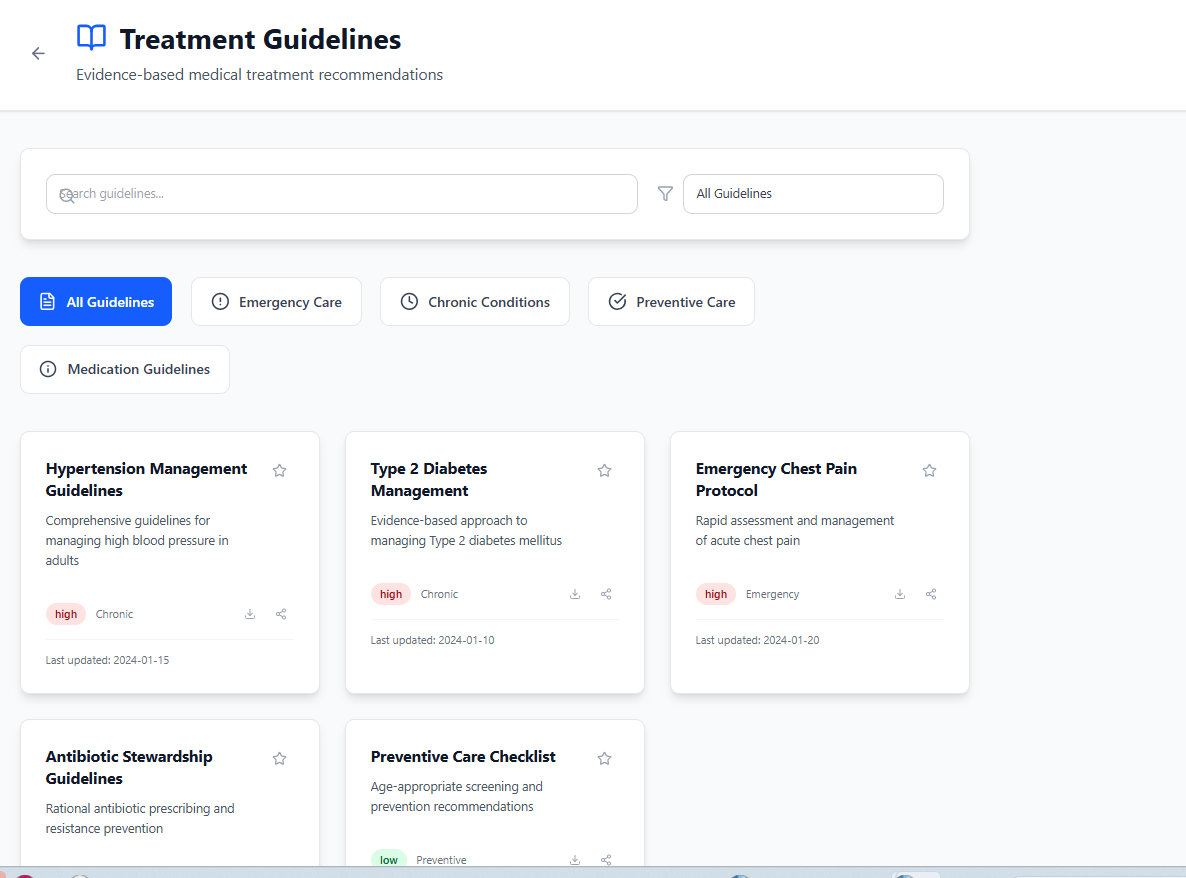


צפייה במחלות שמוגדרות



תיאור תחושות בעת חירום



קבלת הנחיות ע"פ תוצאות העיבוד

# 21. תיאור מסך פתיחה

מסך הפתיחה של מערכת MediAid מציג ממשק התחברות ידידותי למשתמש, המחולק לשני חלקים עיקריים. בצדו השמאלי מוצגת תמצית של יתרונות המערכת – מעקב אחר תרופות והיסטוריה רפואית, קבלת תובנות בריאות מותאמות אישית, גישה לכלי אבחון חירום, ושמירה על פרטיות הנתונים. בצדו הימני של המסך נמצא טופס ההתחברות הכולל שדות להזנת כתובת דוא"ל וסיסמה, אפשרות לסמן "זכור אותי", קישור לשחזור סיסמה, וכפתור "Sign In". בראש המסך קיימת לשונית למעבר בין התחברות לבין רישום משתמש חדש.

המסך מהווה נקודת ניווט ראשונית למשתמשים קיימים וחדשים, ומוביל לגישה למערכת ולשירותיה המרכזיים.

# 22-23. מסכי האפליקציה.

* **מסך הרשמה**

מסך זה מאפשר הרשמה לאפליקציית MediAid. הוא כולל מידע שיווקי על יתרונות האפליקציה, ומאפשר התחלת תהליך יצירת פרופיל בריאות אישי.

* New User (כפתור): מעבר לטופס הרשמה.
* Join MediAid (כותרת): הצגת מטרת ההרשמה.
* אייקון משתמש + טקסט: המחשת תהליך יצירת הפרופיל.
* רשימת יתרונות: מדגישה את הפונקציונליות – מעקב תרופות, תובנות, אבחון חירום, פרטיות.
* Get Started – It’s Free! (כפתור): התחלת ההרשמה.
* טקסט משפטי: הסכמה לתנאי שימוש ופרטיות.
* **פרטים אישיים**

**מטרה**:

איסוף פרטי זיהוי ובריאות בסיסיים של המשתמש לצורך יצירת פרופיל רפואי מותאם.

**רכיבי תצוגה עיקריים:**

* Username – שדה להזנת שם משתמש ייחודי.
* Password – שדה להזנת סיסמה להגנה על פרטיות.
* Email – שדה להזנת כתובת אימייל ליצירת קשר ולאימות.
* Date of Birth – בחירת תאריך לידה (באמצעות בורר תאריכים).
* Gender – תפריט נפתח לבחירת מין.
* Height – שדה להזנת גובה בס"מ.
* Weight– שדה להזנת משקל בק"ג.
* Next Step (כפתור) – מעבר לשלב הבא בתהליך (Risk Factors).
* **שאלון אורח חיים**

מסך זה הוא השלב השני בתהליך יצירת פרופיל הבריאות באפליקציית MediAid. מטרת המסך היא לאסוף מידע אישי על גורמי סיכון בריאותיים, אשר ישמשו להתאמת המלצות רפואיות ומעקב מותאם אישית.

**הסבר על אלמנטי התצוגה:**

Step Indicator (סרגל שלבים): מציין את שלבי ההרשמה – שלב נוכחי מודגש (Step 2: Risk Factors), שלבים קודמים מסומנים כשלמים (Step 1: Basic Info), ושלבים עתידיים באפור (Medications, Medical History).

Health Risk Factors (כותרת ראשית): מבהירה את מטרת המסך – איסוף מידע אישי לצורך ניתוח בריאותי.

טקסט הסבר (Help us understand your health profile): משפט מדריך המנחה את המשתמש למלא את השדות.

**טופס איסוף מידע (9 שדות נפתחים):**

* Smoking Status (סטטוס עישון)
* Alcohol Consumption (צריכת אלכוהול)
* Physical Activity (רמת פעילות גופנית)
* Blood Pressure (לחץ דם)
* Stress Level (רמת לחץ)
* Age Group (קבוצת גיל)
* Family Heart Disease (היסטוריה משפחתית של מחלות לב)
* Family Cancer (היסטוריה משפחתית של סרטן)

Previous (כפתור): חזרה למסך הקודם (Basic Info).

Next Step (כפתור): מעבר לשלב הבא – Medications.

אייקון לב עליון: תוספת גרפית המדגישה את נושא הבריאות.

* **תרופות**

מסך זה הוא השלב השלישי בתהליך יצירת פרופיל הבריאות באפליקציית MediAid. הוא מאפשר למשתמש לבחור מתוך המאגר את רשימת התרופות שהוא נוטל כיום, לצורך קבלת תובנות מותאמות אישית ומניעת אינטראקציות תרופתיות.

**הסבר על אלמנטי התצוגה:**

Current Medications (כותרת ראשית): מציינת את מטרת המסך – הזנת תרופות.

טקסט תיאורי (Add medications you're currently taking): מנחה את המשתמש לגבי הפעולה הרצויה.

Search and Add Medication (שדה חיפוש):  
 מאפשר חיפוש תרופות לפי שם והוספתן לרשימה.  
 שדה טקסט עם הצעה: "Type to search for medications..."

Your Current Medications (0) (רשימת תרופות):  
 רשימה שתציג את התרופות שנוספו. כרגע מופיעה הודעה:  
 "No medications added yet. Search and add medications above."  
 עם אייקון של קפסולה להמחשה ויזואלית.

Previous (כפתור): חזרה לשלב הקודם (Risk Factors).

Next Step (כפתור): המשך לשלב הבא (Medical History).

אייקון קפסולה עליון: תוספת גרפית המשקפת את נושא התרופות.

* **מחלות**

אותו דבר כמו בתרופות, במקביל במחלות.

* **תפריט ראשי**

מסך הבית לאחר התחברות לאפליקציית MediAid. מרכז את כל המידע הבריאותי של המשתמש, כולל התראות, סטטוס בריאותי נוכחי, וקישורים לפעולות נוספות.

* ברכת ברוך הבא: מוצגת בראש המסך עם כתובת המייל של המשתמש.
* כפתור התנתקות: ממוקם בפינה הימנית העליונה ומאפשר יציאה מהמערכת.
* אייקון התראות: מציג מספר התראות חדשות במידת הצורך.
* התראה על השלמת פרופיל: הודעה על כך שהפרופיל הושלם רק ב-40%, עם קריאה להשלמת מידע לשם שיפור תובנות בריאותיות.
* Health Summary (סיכום בריאותי):
  + מצב BMI: —
  + רמת סיכון:---
  + תרופות פעילות:---
  + מצבים רפואיים פעילים:---
* Quick Actions (פעולות מהירות):
  + Emergency Diagnosis: אבחון חירום מהיר והכוונה ראשונית.
  + Upload Medical Data: העלאת מסמכים רפואיים, מרשמים ותוצאות בדיקות.
  + Search Medical Info: חיפוש מחלות, תרופות ותסמינים.
  + Treatment Guidelines: גישה להמלצות טיפול מבוססות-ראיות.
  + Health Profile: צפייה ועריכה של פרטי הבריאות האישיים.
* Recent Medications: הצגת תרופות אחרונות עם קישור לצפייה מלאה.
* Medical History: גישה להיסטוריה הרפואית עם קישור לצפייה מלאה.
* **צפייה בפרופיל האישי**

מטרת המסך היא לאפשר למשתמש לעקוב אחר מצב הבריאות הכללי שלו, לצפות בנתונים שהוזנו בשלבים קודמים בתהליך ההרשמה ולקבל הערכות סיכון מותאמות אישית.

**הסבר על אלמנטי התצוגה:**

כותרת עליונה (My Health Profile):  
 מציינת את מטרת המסך – ניהול מידע בריאותי אישי ומעקב אחר מסע הבריאות של המשתמש.

סרגל ניווט עליון (Tabs) כולל לשוניות מעבר בין קטגוריות שונות בפרופיל.

כרטיסי מידע - מציגים נתונים עיקריים על המשתמש:

* Age (גיל): —
* BMI (מדד מסת גוף): —
* Active Medications (תרופות פעילות): —
* Active Conditions (מצבים רפואיים פעילים): —

Health Risk Assessment (הערכת סיכון בריאותי):

* מציג סיכון בריאותי כללי : —
* מתבסס על גורמי סיכון ואירועים רפואיים קיימים.
* כפתור View Details מאפשר להעמיק בנתוני ההערכה.

Profile Completeness (השלמת פרופיל):

* מדד גרפי המראה את אחוז השלמת פרטי הפרופיל – כרגע 40%.
* נועד לעודד את המשתמש להמשיך לעדכן פרטים לשיפור הדיוק של תובנות הבריאות.

כפתור עליון (Update Profile):  
 מאפשר למשתמש לעדכן את המידע האישי ולשפר את תמונת המצב הבריאותית שלו.

* **צפיה במידע אישי / צפייה בגורמי הסיכון / צפייה בתרופות הנלקחות / צפייה במחלות שמוגדרות**

כל המסכים דומים במראם וייעודם:

* סרגל לניווט בין כרטיסיות הפרופיל
* הצגת המידע הרפואי הקיים של המשתמש.
* כפתור לעריכת הפרופיל
* **העלאת תחושות בעת חירום**

מסך זה הוא שלב העלאת מידע רפואי באפליקציית MediAid.  
 מטרתו לאפשר למשתמש להעביר מידע רפואי בצורת טקסט, תמונה או הקלטה קולית לצורך ניתוח ואבחון מדויק.

**הסבר על אלמנטי התצוגה:**

* **Text Description (תיאור טקסטואלי):** שדה פתוח להזנת תיאור תסמינים, דאגות רפואיות או כל מידע רלוונטי.
* **Medical Images (העלאת תמונה/מסמך):** מאפשר העלאת קובץ רפואי (JPG, PNG, PDF עד 10MB) או צילום דרך המצלמה (כפתור "Open Camera").
* **Voice Recording (הקלטה קולית):** כפתור להקלטת תיאור קולי של המצב הרפואי ("Start Recording").
* **Upload & Analyze Medical Data (כפתור פעולה):** שולח את הנתונים לניתוח רפואי חכם.
* **קבלת הנחיות**

המטרה היא להנגיש המלצות טיפול מבוססות על העיבוד שנעשה במערכת, לפי קטגוריות רפואיות שונות, לתמיכה בקבלת החלטות קליניות.

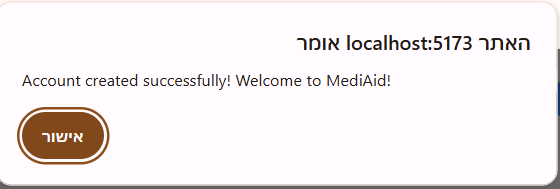
**הסבר על אלמנטי התצוגה:**

* כותרת עליונה (Treatment Guidelines):  
   מסמנת שהמסך עוסק בהנחיות לטיפול רפואי.
* שורת חיפוש וסינון:  
   מאפשרת חיפוש לפי מילות מפתח או סינון לפי סוג הנחיות (All Guidelines, Emergency, Chronic, Preventive וכו').
* כפתורי סינון קטגוריות:  
   מאפשרים מעבר מהיר בין קטגוריות:
  + All Guidelines (ברירת מחדל)
  + Emergency Care
  + Chronic Conditions
  + Preventive Care
  + Medication Guidelines
* כרטיסי הנחיות (Guideline Cards):  
   כל כרטיס מציג שם ההנחיה, תיאור קצר, רמת דחיפות (High / Medium / Low), קטגוריה (כגון Chronic, Emergency וכו'), תאריך עדכון אחרון ואייקונים לפעולות (שמירה, שיתוף).  
   דוגמאות בולטים מהמסך:
  + Hypertension Management Guidelines – לטיפול בלחץ דם גבוה.
  + Emergency Chest Pain Protocol – לניהול מהיר של כאבים בחזה.
  + Preventive Care Checklist – להמלצות סינון מותאמות גיל.

# 24. הודעות למשתמש והתרעות

כאשר מתרחשת תקלה שקשורה ישירות לממשק עם המשתמש, המערכת מתריעה על כך.

כך בעת בעיית התחברות עם השרת, שדות שאינם יכולים להישאר ריקים ונשארו כאלו וכדו'.

כמו"כ, המערכת מודיעה למשתמש גם על פעולות מוצלחות, כגון:  


# 25. ממשק משתמש

ממשק המשתמש של אפליקציית MediAid בנוי בצורה ברורה, מודולרית ונוחה לניווט בין שלבי התהליך הרפואי. כל מסך מתמקד בפעולה אחת מרכזית, עם עיצוב נקי, כפתורים מודגשים והסברים מלווים שמנחים את המשתמש באופן אינטואיטיבי לאורך כל הדרך.

# 26. קוד התוכנית

#### 26.1 קלט (Input)

המערכת מטפלת במגוון סוגי קלט הנדרשים לפונקציונליות רפואית מקיפה:

קלט מידע משתמש

* **נתוני רישום וזיהוי**: שם משתמש, דואר אלקטרוני, סיסמה מוצפנת
* **מידע דמוגרפי**: תאריך לידה, מין, גובה, משקל
* **גורמי סיכון**: סטטוס עישון, צריכת אלכוהול, פעילות גופנית, לחץ דם, רמת מתח

קלט מידע רפואי היסטורי

* **מחלות**: רשימת מחלות נוכחיות וקודמות כולל תאריכי אבחון, רמת חומרה וסטטוס
* **תרופות**: תרופות נוכחיות עם מינון, תדירות ודרך מתן
* **בדיקות מעבדה**: תוצאות בדיקות עם ערכי ייחוס ותאריכים
* **הליכים רפואיים**: ניתוחים ופרוצדורות שבוצעו

קלט בזמן אמת לניתוח סימפטומים

**טקסט חופשי**: תיאור סימפטומים בשפה טבעית  
 { "text": "יש לי כאב ראש חד ולחץ בחזה כבר שעתיים"}

**תמונות רפואיות**: תמונות של נגעים או שינויים חיצוניים  
 { "image": "base64\_encoded\_image\_data", "min\_confidence": 0.1}

#### 26.2 פלט (Output)

ניתוח סימפטומים מטקסט

{

"success": true,

"data": {

"original\_text": "I have a severe headache.",

"symptoms\_found": 1,

"symptoms": [

{

"cui": "C0018681",

"name": "Headache",

"detected\_name": "headache",

"accuracy": 0.94,

"confidence": 0.91,

"status": "Present"

}

],

"analyzer\_type": "MedCAT"

}

}

ניתוח סימפטומים מתמונות

{

"success": true,

"data": {

"symptoms\_found": 2,

"symptoms": [

{

"cui": "C0041834",

"name": "Erythema",

"confidence": 0.87,

"probability": 0.87

}

],

"analyzer\_type": "BiomedCLIP"

}

}

המלצות טיפול

{

"treatment\_recommendations": [

{

"type": "medication",

"recommendation": "Acetaminophen 500mg",

"reason": "For headache relief",

"urgency": "low",

"confidence": 0.85

},

{

"type": "medical\_consultation",

"recommendation": "See primary care physician",

"reason": "Persistent chest pressure with cardiac risk factors",

"urgency": "high",

"confidence": 0.92

}

]

}

מסלולי אבחון גרפיים

{

"diagnostic\_path": {

"nodes": [

{"id": "user\_condition", "type": "disease", "name": "Hypertension"},

{"id": "symptom\_detected", "type": "symptom", "name": "Chest pressure"},

{"id": "treatment\_node", "type": "treatment", "name": "Dose adjustment"}

],

"edges": [

{"from": "user\_condition", "to": "symptom\_detected", "relationship": "HAS\_SYMPTOM"},

{"from": "symptom\_detected", "to": "treatment\_node", "relationship": "TREATED\_BY"}

]

}

}

#### 26.3 פונקציות חשובות

מערכת השרת הראשית (Java - Spring Boot)

א. ניהול אבטחה ואימות משתמשים

**generateToken() - יצירת JWT Token**

public String generateToken(String username, String email, UUID userId) {

return Jwts.builder()

.setSubject(username)

.claim("email", email)

.claim("userId", userId.toString())

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + expiration))

.signWith(getKey())

.compact();

}

**הסבר**: פונקציה המייצרת JWT token מאובטח עבור משתמש מחובר, כולל שם משתמש, אימייל ומזהה ייחודי עם תוקף זמן מוגדר.

**isValid() - אימות תקינות Token**

public boolean isValid(String token) {

try {

Claims claims = getClaimsFromToken(token);

return !claims.getExpiration().before(new Date());

} catch (JwtException | IllegalArgumentException e) {

return false;

}

}

**הסבר**: בודקת האם JWT token תקין ולא פג תוקפו, מבטיחה אבטחת הגישה למערכת.

ב. ניהול מידע רפואי אישי

**updateUserRiskFactors() - עדכון גורמי סיכון**

public RiskFactorResponseDTO updateUserRiskFactors(UUID userId, RiskFactorUpdateDTO dto) {

User user = userRepository.findById(userId).orElse(null);

if (user == null) {

throw new RuntimeException("User not found");

}

if (dto.getSmokingStatus() != null) {

user.setSmokingStatus(dto.getSmokingStatus());

}

if (dto.getAlcoholConsumption() != null) {

user.setAlcoholConsumption(dto.getAlcoholConsumption());

}

User savedUser = userRepository.save(user);

double overallRiskScore = savedUser.calculateOverallRiskScore();

return new RiskFactorResponseDTO(

"Risk factors updated successfully",

overallRiskScore,

calculateRiskLevel(overallRiskScore)

);

}

**הסבר**: מעדכנת גורמי סיכון של המשתמש (עישון, אלכוהול וכד') ומחשבת ציון סיכון כולל מעודכן.

**calculateOverallRiskScore() - חישוב ציון סיכון כולל**

public double calculateOverallRiskScore() {

double totalRisk = 0.0;

int factorCount = 0;

if (smokingStatus != null) {

totalRisk += smokingStatus.getWeight();

factorCount++;

}

if (alcoholConsumption != null) {

totalRisk += alcoholConsumption.getWeight();

factorCount++;

}

// המשך עם גורמי סיכון נוספים...

return factorCount > 0 ? totalRisk / factorCount : 0.0;

}

**הסבר**: מחשבת ציון סיכון כולל על בסיס כל גורמי הסיכון הרפואיים של המשתמש עם משקלים שונים לכל גורם.

ג. ניהול מסד נתוני גרף Neo4j

**importAllEntitiesFromDB() - ייבוא ישויות רפואיות לגרף**

public void importAllEntitiesFromDB() {

List<EntityImportConfig<?>> importConfig = Arrays.asList(

new EntityImportConfig<>("Diseases", diseaseRepository, EntityTypes.DISEASE, this::mapDisease),

new EntityImportConfig<>("Medications", medicationRepository, EntityTypes.MEDICATION, this::mapMedication),

new EntityImportConfig<>("Symptoms", symptomRepository, EntityTypes.SYMPTOM, this::mapSymptom)

);

for (EntityImportConfig<?> config : importConfig) {

importEntitiesGeneric(config);

}

}

**הסבר**: מייבאת את כל הישויות הרפואיות (מחלות, תרופות, סימפטומים) ממסד הנתונים היחסי לגרף Neo4j ליצירת מבנה קשרים מתקדם.

**createRelationshipsBatch() - יצירת קשרים בגרף**

private int createRelationshipsBatch(List<Map<String, Object>> relationships) {

try (Session session = neo4jDriver.session()) {

return session.writeTransaction(tx -> {

int successCount = 0;

for (Map<String, Object> rel : relationships) {

String query = "MATCH (n1 {cui: $cui1}), (n2 {cui: $cui2}) " +

"CREATE (n1)-[:" + rel.get("relType") +

" {weight: $weight, source: $source}]->(n2)";

tx.run(query, rel);

successCount++;

}

return successCount;

});

}

}

**הסבר**: יוצרת קשרים רפואיים בין ישויות בגרף (כמו "תרופה מטפלת במחלה") עם משקלים ומקורות מידע.

מערכת ניתוח סימפטומים (Python)

א. ניתוח טקסט רפואי

**extract\_symptoms() - חילוץ סימפטומים מטקסט**

def extract\_symptoms(self, text):

if not self.is\_loaded:

raise Exception("Text model not loaded")

entities = self.cat.get\_entities(text)

symptom\_entities = []

for ent\_id, data in entities.get("entities", {}).items():

type\_ids = data.get("type\_ids", [])

if "T184" in type\_ids: # T184 = Sign or Symptom לפי UMLS

symptom = {

"cui": data.get("cui"),

"name": data.get("pretty\_name"),

"detected\_name": data.get("detected\_name"),

"accuracy": round(data.get("acc", 0), 4),

"confidence": round(data.get("context\_similarity", 0), 4)

}

symptom\_entities.append(symptom)

return symptom\_entities

**הסבר**: מנתחת טקסט חופשי ומחלצת סימפטומים רפואיים מזוהים עם קודי UMLS, רמת דיוק וביטחון.

**analyze\_text() - ניתוח מלא של טקסט**

def analyze\_text(self, text):

symptoms = self.extract\_symptoms(text)

return {

"original\_text": text,

"symptoms\_found": len(symptoms),

"symptoms": symptoms,

"analysis\_status": "success",

"analyzer\_type": "MedCAT"

}

**הסבר**: מבצעת ניתוח מקיף של טקסט רפואי ומחזירה תוצאות מפורטות כולל הטקסט המקורי וכל הסימפטומים שזוהו.

ב. ניתוח תמונות רפואיות

**\_preprocess\_image() - עיבוד מקדים של תמונות**

def \_preprocess\_image(self, image\_input):

try:

if isinstance(image\_input, str):

if image\_input.startswith('data:image') or len(image\_input) > 1000:

# Base64 string

if image\_input.startswith('data:image'):

image\_input = image\_input.split(',')[1]

image\_bytes = base64.b64decode(image\_input)

image = Image.open(io.BytesIO(image\_bytes)).convert('RGB')

else:

# File path

image = Image.open(image\_input).convert('RGB')

elif isinstance(image\_input, bytes):

image = Image.open(io.BytesIO(image\_input)).convert('RGB')

processed\_image = self.preprocess(image).unsqueeze(0).to(self.device)

return processed\_image

except Exception as e:

raise Exception(f"Failed to preprocess image: {e}")

**הסבר**: מעבדת תמונות מפורמטים שונים (נתיב קובץ, base64, bytes) ומכינה אותן לניתוח במודל.

**extract\_symptoms() - חילוץ סימפטומים מתמונות**

def extract\_symptoms(self, image\_input, min\_confidence=None):

confidence\_threshold = min\_confidence or self.min\_confidence

image = self.\_preprocess\_image(image\_input)

template = 'this is a photo of '

text\_inputs = self.tokenizer(

[template + label for label in self.labels],

context\_length=256

).to(self.device)

with torch.no\_grad():

image\_features, text\_features, logit\_scale = self.model(image, text\_inputs)

logits = (logit\_scale \* image\_features @ text\_features.T).softmax(dim=-1)

probs = logits[0].cpu().numpy()

symptoms = []

for i, prob in enumerate(probs):

if prob >= confidence\_threshold:

cui = self.\_get\_cui\_for\_label(self.labels[i])

symptom = {

"cui": cui,

"name": self.labels[i],

"confidence": round(float(prob), 4),

"probability": round(float(prob), 4)

}

symptoms.append(symptom)

return sorted(symptoms, key=lambda x: x['confidence'], reverse=True)

**הסבר**: מנתחת תמונה רפואית ומזהה סימפטומים חזותיים (כמו אדמומיות, נפיחות) עם רמת ביטחון לכל זיהוי.

פונקציות ניתוח גרף עיקריות (מתוכנן)

א. מנוע חיפוש מסלולים רפואיים (Java)

**findTreatmentPaths() - חיפוש מסלולי טיפול**

public List<DiagnosticPath> findTreatmentPaths(

UUID userId,

List<String> detectedSymptomCuis,

PathFindingOptions options) {

// טעינת פרופיל רפואי של המשתמש

UserMedicalProfile profile = loadUserMedicalProfile(userId);

// יצירת נקודות התחלה בגרף

List<GraphNode> startNodes = createStartNodes(profile);

List<GraphNode> symptomNodes = createSymptomNodes(detectedSymptomCuis);

// חיפוש מסלולים בגרף Neo4j

List<DiagnosticPath> paths = neo4jPathService.findPaths(

startNodes, symptomNodes, options.getMaxDepth(), options.getMinWeight()

);

// דירוג לפי רלוונטיות

return rankPaths(paths, profile, options);

}

**הסבר**: תחפש במסד הגרף מסלולים המקשרים בין המצב הרפואי הקיים של המשתמש לסימפטומים שזוהו, ותמצא דרכי טיפול אפשריות.

**rankPaths() - דירוג מסלולי טיפול**

private List<DiagnosticPath> rankPaths(List<DiagnosticPath> paths,

UserMedicalProfile profile,

PathFindingOptions options) {

return paths.stream()

.map(path -> {

double relevanceScore = calculateRelevanceScore(path, profile);

double confidenceScore = calculateConfidenceScore(path);

double urgencyScore = calculateUrgencyScore(path, profile);

path.setOverallScore(

relevanceScore \* 0.4 + confidenceScore \* 0.4 + urgencyScore \* 0.2

);

return path;

})

.sorted((p1, p2) -> Double.compare(p2.getOverallScore(), p1.getOverallScore()))

.collect(Collectors.toList());

}

**הסבר**: תדרג את מסלולי הטיפול לפי רלוונטיות למשתמש, רמת אמינות מדעית ודחיפות רפואית.

ב. מנוע המלצות טיפול

**generateRecommendations() - יצירת המלצות טיפול**

public TreatmentRecommendations generateRecommendations(

UUID userId, List<DiagnosticPath> paths, List<DetectedSymptom> symptoms) {

TreatmentRecommendations recommendations = new TreatmentRecommendations();

for (DiagnosticPath path : paths) {

UserCompatibility compatibility = checkUserCompatibility(userId, path);

if (compatibility.isCompatible()) {

List<TreatmentOption> options = generateTreatmentOptions(path, compatibility);

for (TreatmentOption option : options) {

// בדיקת אינטראקציות תרופתיות

DrugInteractionCheck interactionCheck = checkDrugInteractions(userId, option);

ContraindicationCheck contraindicationCheck = checkContraindications(userId, option);

if (interactionCheck.isSafe() && contraindicationCheck.isSafe()) {

option.setSafetyScore(calculateSafetyScore(interactionCheck, contraindicationCheck));

recommendations.addOption(option);

}

}

}

}

return prioritizeRecommendations(recommendations);

}

**הסבר**: תייצר המלצות טיפול מותאמות אישית, תבדוק התאמה למשתמש, אינטראקציות תרופתיות והתוויות נגד.

#### ג. מערכת ניתוח סיכונים דינמית

**assessRisk() - ניתוח סיכונים מקיף**

public RiskAssessment assessRisk(UUID userId, List<DetectedSymptom> symptoms, List<DiagnosticPath> paths) {

UserMedicalProfile profile = userProfileService.getProfile(userId);

RiskAssessment assessment = new RiskAssessment();

// חישוב סיכון מהסימפטומים

double symptomRisk = calculateSymptomBasedRisk(symptoms, profile);

double historyRisk = calculateHistoryBasedRisk(profile);

double pathRisk = calculatePathBasedRisk(paths, profile);

double overallRisk = combineRiskScores(symptomRisk, historyRisk, pathRisk);

assessment.setOverallRiskScore(overallRisk);

assessment.setRiskLevel(determineRiskLevel(overallRisk));

assessment.setRecommendedAction(determineRecommendedAction(overallRisk));

// זיהוי סימני אזהרה קריטיים

List<RedFlag> redFlags = identifyRedFlags(symptoms, profile, paths);

assessment.setRedFlags(redFlags);

return assessment;

}

**הסבר**: תנתח את רמת הסיכון הכוללת על בסיס הסימפטומים, ההיסטוריה הרפואית והמסלולים שנמצאו, ותזהה סימני אזהרה קריטיים.

**identifyRedFlags() - זיהוי סימני אזהרה**

private List<RedFlag> identifyRedFlags(List<DetectedSymptom> symptoms, UserMedicalProfile profile, List<DiagnosticPath> paths) {

List<RedFlag> redFlags = new ArrayList<>();

// בדיקת סימפטומים מסכני חיים

for (DetectedSymptom symptom : symptoms) {

if (isLifeThreateningSymptom(symptom.getCui())) {

redFlags.add(new RedFlag(

"CRITICAL\_SYMPTOM",

"Life-threatening symptom detected: " + symptom.getName(),

RedFlag.Severity.CRITICAL

));

}

}

// בדיקת שילובים מסוכנים

for (DiagnosticPath path : paths) {

if (indicatesMedicalEmergency(path, profile)) {

redFlags.add(new RedFlag(

"EMERGENCY\_PATTERN",

"Symptom pattern indicates possible emergency",

RedFlag.Severity.HIGH

));

}

}

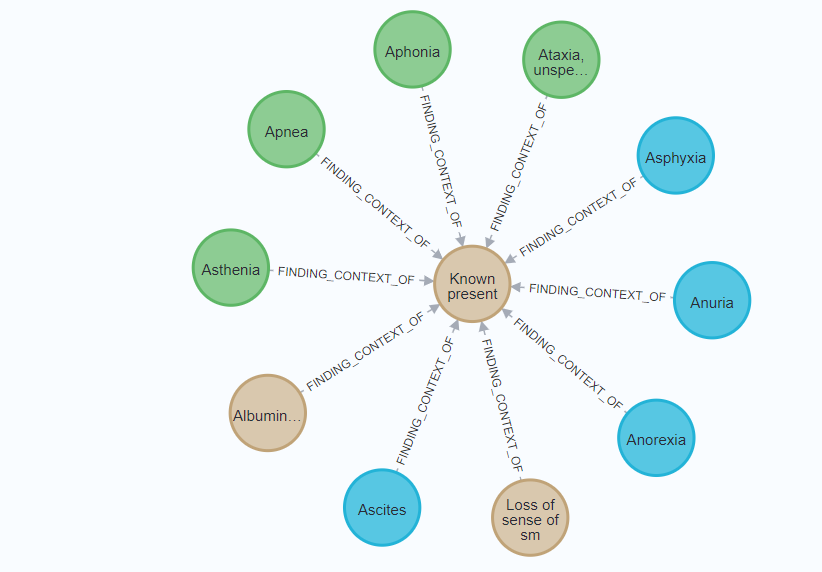
return redFlags;

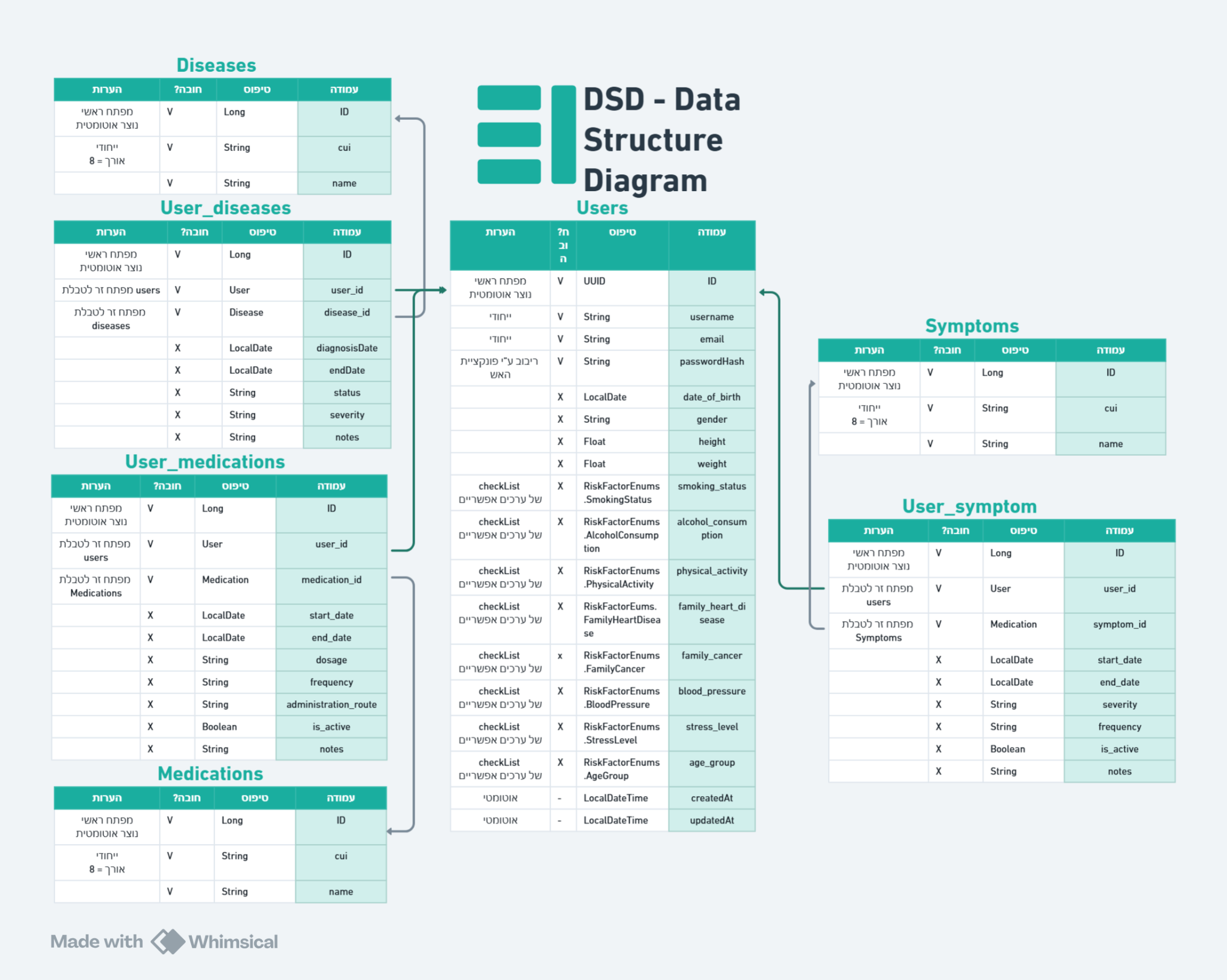
}

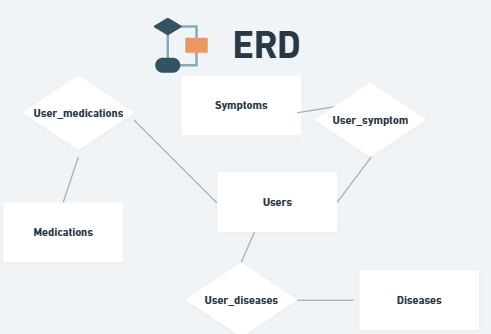
**הסבר**: תזהה סימפטומים או שילובי סימפטומים המצביעים על מצב חירום רפואי ותפיק התראות מתאימות.

### 27. תיאור מסד הנתונים

מסד הנתונים של Neo4j מורכב מישויות עם תוויות מזהות - לפי קטגוריות וקשרים מסוגים רבים.

דוגמה:  
לכל קטגוריה יש צבע שונה.

מסד הנתונים של PostgreSQL מורכב מטלת משתמשים, טבלאות של מונחים רפואיים ממויינים מתוך UMLS וטבלאות קישור בין המשתמשים למונחים בתוספת שדות מידע.



# 28. מדריך ידידותי למשתמש

**🎯 מהי מערכת MediAid?**

MediAid היא מערכת בריאות חכמה שעוזרת לך:

* **לקבל המלצות טיפול** מותאמות אישית
* **לעקוב אחר בריאותך** ולזהות גורמי סיכון
* **לחפש מידע רפואי** מהימן ומדויק

⚠️ **חשוב לזכור:** המערכת היא כלי עזר בלבד ואינה מחליפה ייעוץ רפואי מקצועי!

**🚀 התחלה מהירה**

**שלב 1: הרשמה למערכת**

1. לחץ על **"הרשמה"** בעמוד הראשי
2. מלא את הפרטים הבסיסיים:
   * שם משתמש (יחודי)
   * כתובת אימייל
   * סיסמה חזקה (לפחות 8 תווים)
   * מגדר ותאריך לידה

המערכת תיצור עבורך token אבטחה אוטומטית

**👤 יצירת פרופיל בריאות**

לאחר ההרשמה, חשוב למלא את הפרופיל הרפואי שלך:

**נתונים פיזיים:**

* **גובה:** (בסנטימטרים) - דוגמה: 175
* **משקל:** (בקילוגרמים) - דוגמה: 70
* **BMI:** יחושב אוטומטית

**למה זה חשוב?** המערכת משתמשת בנתונים אלה כדי לחשב גורמי סיכון ולהתאים המלצות אישיות.

**מחלות קיימות**

הוסף מחלות שאתה סובל מהן:

1. לחץ על **"הוסף מחלה"**
2. התחל להקליד את שם המחלה
3. בחר מהרשימה שמופיעה (המערכת מחפשת במאגר UMLS)
4. הוסף תאריך אבחון ופרטים נוספים

**תרופות נוכחיות**

1. לחץ על **"הוסף תרופה"**
2. הקלד שם התרופה
3. הוסף מינון ותדירות נטילה

**🔍 ניתוח סימפטומים**

**ניתוח באמצעות טקסט חופשי**

**מתי להשתמש?** כשאתה מרגיש לא טוב ורוצה להבין מה עשוי לקרות.

**איך זה עובד:**

1. לחץ על **"נתח סימפטומים מטקסט"**
2. כתב בחופשיות מה אתה מרגיש

**דוגמאות לתיאורים טובים:**

✅ "אני מרגיש כאב חד בחזה כבר שעתיים, בעיקר כשאני נושם עמוק"

✅ "יש לי חום 38.5 מעלות, כאב ראש וחולשה כללית מאתמול"

✅ "הרגליים שלי נפוחות בערב והן כואבות כשאני הולך"

**דוגמאות לתיאורים שלא יעזרו:**

❌ "אני לא מרגיש טוב"

❌ "משהו כואב לי"

❌ "יש לי בעיה"

ניתוח באמצעות תמונה

**מתי להשתמש?** עבור בעיות עור, נגעים חיצוניים או שינויים חזותיים.

**איך זה עובד:**

1. לחץ על **"נתח סימפטומים מתמונה"**
2. צלם תמונה ברורה או העלה מהגלריה
3. המערכת תנתח את התמונה באמצעות AI רפואי

**טיפים לתמונות טובות:**

* ✅ תאורה טובה ובהירה
* ✅ מקרוב על האזור הפגוע
* ✅ חדות מקסימלית
* ❌ אל תשתמש בפלאש
* ❌ אל תצלם מרחוק מדי

**דוגמאות מצבים מתאימים:**

* פצעים או חתכים
* נגעי עור
* שינויי צבע בעור
* נפיחות או דלקות

**💡 קבלת המלצות טיפול**

המלצות אוטומטיות

לאחר ניתוח הסימפטומים, המערכת תציג:

**סוגי המלצות:**

1. **📋 בדיקות מומלצות**
   * בדיקות דם
   * בדיקות הדמיה
   * בדיקות מומחה
2. **💊 טיפול ראשוני**
   * תרופות ללא מרשם
   * שינויי אורח חיים
   * הקלה ביתית
3. **🏥 רמת דחיפות**
   * 🟢 **נמוך:** אפשר להמתין לרופא משפחה
   * 🟡 **בינוני:** מומלץ לפנות לרופא תוך יום-יומיים
   * 🟠 **גבוה:** פנה לרופא היום
   * 🔴 **קריטי:** פנה למיון מיד!

**📊 ניהול גורמי סיכון**

**מה זה גורמי סיכון?**

גורמי סיכון הם מאפיינים שמגדילים את הסיכון למחלות מסוימות.

**הגורמים שהמערכת עוקבת אחריהם:**

**🚬 עישון**

* אף פעם לא עישנתי
* עישנתי בעבר (קל/כבד)
* מעשן כיום (קל/כבד)

**🍺 אלכוהול**

* לא שותה
* שתייה קלה (1-2 כוסות בשבוע)
* שתייה בינונית (3-7 כוסות בשבוע)
* שתייה כבדה (יותר מ-7 כוסות בשבוע)

**🏃‍♂️ פעילות גופנית**

* מאוד פעיל (ספורט מקצועי)
* פעיל (ספורט קבוע)
* בינוני (פעילות מדי פעם)
* נמוך (הליכה בסיסית)
* יושבני (ללא פעילות)

**💓 לחץ דם**

* תקין (מתחת ל-120/80)
* מוגבר קלות (120-139/80-89)
* שלב 1 (140-159/90-99)
* שלב 2 (160-179/100-109)
* משבר (מעל 180/110)

**חישוב ציון סיכון אישי**

המערכת מחשבת עבורך ציון סיכון כולל:

**דוגמה:**

📊 הציון שלך: 0.35 (35%)

📈 רמת סיכון: בינונית

**🆘 מצבי חירום**

**מתי להשתמש במצב חירום?**

* כאב חזה או קוצר נשימה
* חום גבוה מ-39 מעלות
* כאב חד ופתאומי
* שינוי פתאומי במצב

**איך המערכת עוזרת במצב חירום?**

1. ניתוח מהיר של הסימפטומים
2. העדפה לבטיחות - אם יש ספק, המערכת תמליץ על פנייה לעזרה רפואית
3. זיהוי קשרים עם התרופות והמחלות שלך
4. הנחיות ברורות למה לעשות עכשיו

**בעיות נפוצות ופתרונות**

**🔧 המערכת לא מזהה סימפטומים:**

* וודא שכתבת בעברית או אנגלית
* השתמש במונחים רפואיים פשוטים
* תאר כמה שיותר פרטים

**🔧 התמונה לא עולה:**

* בדוק שהקובץ לא גדול מ-16MB
* ודא שזה קובץ תמונה (JPG/PNG)
* נסה לצלם תמונה חדשה

**🔧 שכחתי סיסמה:**

* לחץ על "שכחתי סיסמה"
* הכנס את האימייל שלך
* בדוק את תיבת הדואר לקישור איפוס

**למצבי חירום רפואיים:**

* 🚑 מגן דוד אדום: 101
* 🏥 מוקד בריאות כללית: \*6101

💙 **המערכת פותחה מתוך רצון לעזור לך לנהל את בריאותך טוב יותר.** **זכור: אתה הכי מכיר את הגוף שלך, והמערכת היא רק כלי עזר.** **תמיד פנה לייעוץ רפואי מקצועי כשיש ספק!**

# 29. בדיקות והערכה

לאורך תהליך הפיתוח של מערכת MediAid ביצענו בדיקות יסודיות שבחנו את איכות המערכת, אמינותה ויכולתה להתמודד עם מקרים רפואיים אמיתיים. הבדיקות התבצעו הן ברמה הפונקציונלית של כל רכיב, והן ברמה כוללת של חוויית המשתמש ודיוק התוצאה.

**בדיקות פונקציונליות**

תחילה נבדקו כל הפונקציות העיקריות של המערכת: זיהוי תסמינים מתוך טקסט חופשי, ניתוח תסמינים מתמונה, חישוב גורמי סיכון רפואיים, ומתן המלצות מותאמות אישית. בכל אחד מהמקרים בדקנו אם הקלטים מנותחים נכון ואם הפלטים שהמערכת מספקת תואמים לציפיות רפואיות בסיסיות.

למשל, כשהוזן טקסט רפואי הכולל תיאור של כאבי ראש, חולשה וחום – המערכת הצליחה לזהות את התסמינים העיקריים ולסווג אותם כראוי. גם בניתוח תמונות של נגעים עוריים, הצלחנו לזהות פרמטרים כמו אדמומיות, פריחה או נפיחות ברמה מספקת לאבחנה ראשונית.

**בדיקות קצה (Edge Cases)**

אחת ממטרות הבדיקה הייתה לוודא שהמערכת שורדת גם מקרים קיצוניים. בדקנו כיצד היא מגיבה להזנת טקסטים חסרי משמעות, תמונות מטושטשות, קלטים חלקיים או חסרי נתונים חשובים. המערכת הגיבה היטב לרוב המקרים, עם הודעות שגיאה ברורות או טיפול סביר בחריגים.

**מדדי ביצועים**

מדדנו את מהירות עיבוד הקלטים (גם טקסט וגם תמונות), ואת זמן התגובה הכולל של המערכת למשתמש. המערכת עיבדה תשובות בתוך שניות בודדות – זמן שמתאים לאפליקציה רפואית לשימוש בזמן אמת, במיוחד בתרחישים של חירום רפואי.

**משוב משתמשים**

לבסוף, ביצענו בדיקות עם קבוצת משתמשים מצומצמת, כדי לקבל חוות דעת מהשטח. המשתמשים דיווחו על ממשק נעים וקל להבנה, ותיארו תחושת אמון מסוימת כלפי התשובות שקיבלו. בהתאם למשוב, שיפרנו ניסוחים, זרימת תהליך העבודה והדגשנו נתונים חשובים יותר בממשק.

# 30. מסקנות

פיתוח מערכת MediAid אפשר לי לבחון מקרוב את הפוטנציאל הגלום בשילוב בין בינה מלאכותית, ניתוח שפה טבעית (NLP), ראייה ממוחשבת, ומודלים גרפיים לצורך פתרון בעיות רפואיות מורכבות. כמפתחת יחידה, נדרש ממני לא רק ידע טכני רחב, אלא גם תכנון קפדני, קבלת החלטות עצמאית, ושמירה על איזון בין עומק טכנולוגי לבין נגישות וקלות שימוש למשתמש הקצה.

אחת המסקנות המרכזיות שעלו לאורך הדרך היא החשיבות הרבה שיש לנירמול המידע הרפואי. גיליתי שהמעבר מטקסט חופשי של משתמשים לרשימה מובנית של תסמינים (למשל בהתבסס על מערכת UMLS) הוא שלב מהותי בניתוח מידע רפואי אמין. לצורך כך, פיתחתי מנגנון ייחודי המזהה תסמינים רלוונטיים מתוך תיאורים כלליים או תמונות של נגעים, וממיר אותם לייצוג פורמלי שאפשר להצליב מול גרף הידע הרפואי.

שימוש ב־Neo4j כגרף ידע רפואי התברר כבחירה אסטרטגית מוצלחת. הוא איפשר לי לתכנן לוגיקת טיפול אינטואיטיבית אך חכמה, שמבוססת על הקשרים בין סימפטומים, מצבים רפואיים והמלצות טיפוליות. במקביל, PostgreSQL שימש כמאגר מידע מובנה, והקפדתי לבנות אינטגרציה הדוקה בין שני בסיסי הנתונים באמצעות Spring Boot ב־Java.

עבודה עם מספר טכנולוגיות – Python למודולים של AI, Java לממשק עם מסדי הנתונים וה־Backend, וספריות מתקדמות לעיבוד טקסט ותמונה – חייבה אותי לבנות ארכיטקטורה נקייה, מודולרית וברורה. כל מודול תוכנן כך שיהיה ניתן לבדיקה עצמאית, עם ממשק ברור, תחום אחריות אחד, ותיעוד פנימי מדויק. זה איפשר לי לנהל את הפרויקט בצורה שיטתית ולהבטיח שכל רכיב תורם ישירות לתוצאה הסופית.

מהבחינה המעשית, אחת התובנות העיקריות הייתה החשיבות של חוויית משתמש – במיוחד כשמדובר בתחום רפואי. השקעתי רבות בעיצוב הממשק, בהנגשת ההודעות ובבהירות ההמלצות, מתוך הבנה שהמערכת צריכה לשרת משתמשים ללא רקע רפואי – במצבי לחץ, ובאופן עצמאי.

לסיכום, תהליך הפיתוח של MediAid חיזק אצלי את ההבנה שלא מספיק להפעיל אלגוריתם חכם – יש חשיבות מכרעת לארגון נכון של זרימת המידע, לתכנון גרפי אינטואיטיבי של מהלך קבלת ההחלטות, ולבדיקות שיטתיות שמבטיחות אמינות לאורך זמן. אני רואה בפרויקט הזה אבן דרך משמעותית במסלול ההתמקצעות שלי כמהנדסת תוכנה, ובוודאי בסיס להמשך פיתוח והעמקה בעתיד.

# 31. פיתוחים עתידיים

במהלך פיתוח מערכת MediAid, עלו לא מעט רעיונות ואפשרויות להרחבה עתידית של הפונקציונליות והיכולות שלה, הן מהפן הטכנולוגי והן מהפן הרפואי־יישומי. מאחר והמערכת נבנתה בצורה מודולרית, קיימת תשתית טובה להוסיף פיצ’רים נוספים בצורה הדרגתית, מבלי לשבור את המבנה הקיים.

אחד מהכיוונים המרכזיים שאני שוקלת להוסיף בעתיד הוא שילוב **מודול המלצות בזמן אמת לרופאים**, שיספק לא רק תובנות ראשוניות למטופל, אלא גם ממשק מקצועי לרופא, עם ניתוחים סטטיסטיים, גרפים של מגמות, והשוואות למקרים דומים. פיתוח כזה ידרוש גם גישה למידע אנונימי נוסף ולמודלים מתקדמים יותר של למידת מכונה, אך יש לו פוטנציאל משמעותי לחולל שינוי מהותי בקבלת החלטות רפואיות.

כיוון נוסף הוא **שיפור התמיכה הרב־לשונית**, כך שהמערכת תוכל לקבל טקסטים חופשיים גם בשפות נוספות (כגון עברית, צרפתית, ערבית), ולתרגם אותם באופן סמנטי לאנגלית רפואית סטנדרטית, תוך שימוש במודלי תרגום רפואיים מותאמים. דבר זה יאפשר שימוש רחב יותר במדינות נוספות, תוך שמירה על דיוק רפואי.

בנוסף, אני מתכננת להוסיף **אינטגרציה חכמה עם מערכות לבישה (wearables)** כמו שעונים חכמים, צמידים רפואיים וחיישני תנועה. המטרה היא לשלב מדדים פיזיולוגיים (כגון דופק, חום גוף, תזוזות חריגות) כחלק מהקלט למערכת, וכך לייצר תמונת מצב רחבה ואמינה יותר. שיפור כזה יוכל לשפר את זיהוי מצבי חירום באופן יזום – עוד לפני שהמשתמש בעצמו מבקש עזרה.

היבט נוסף שנמצא על הפרק הוא **שיפור המודלים הגרפיים**, כך שיכללו גם הקשרים סטטיסטיים מבוססי עשרות אלפי תיקים רפואיים, לא רק על סמך ידע רפואי פורמלי, אלא גם על סמך Pattern Recognition מתוך דאטה אמיתי. זה ידרוש אינטגרציה עם מאגרי מידע קליניים אנונימיים והרחבת הידע של הגרף הקיים.

לבסוף, ברמה המערכתית, אני מתכננת לאפשר **אופציית "היסטוריה רפואית אישית"** מבוססת הרשאות, כך שהמערכת תוכל ללמוד לאורך זמן את ההיסטוריה של המשתמש (מחלות רקע, תרופות, בדיקות) ולתת מענה מותאם ומדויק יותר עם כל אינטראקציה. בכך, המערכת תהפוך למלווה רפואית חכמה ורציפה.

לסיכום, הפיתוח הנוכחי מהווה רק שלב ראשון במימוש הפוטנציאל של MediAid. באמצעות הרחבות מדורגות ומתוכננות היטב, ניתן להפוך את המערכת מכלי תמיכה ראשוני – לפלטפורמה רפואית רחבה, מהימנה, מותאמת אישית – ובעתיד גם מחוברת למערכות בריאות ארציות או עולמיות.

# 31. ביבליוגרפיה (קישורים פעילים)

#### UMLS:

[Unified Medical Language System (UMLS)](https://www.nlm.nih.gov/research/umls/index.html)

[National Institutes of Health (NIH) (.gov)](https://uts.nlm.nih.gov/)

[wikipedia](https://en.wikipedia.org/wiki/Unified_Medical_Language_System)

[ScienceDirect](https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/unified-medical-language-system)

[Papers With Code](https://paperswithcode.com/dataset/umls)

[Oxford Academic](https://academic.oup.com/nar/article/32/suppl_1/D267/2505235)

#### NLP רפואי

[National Library of Medicine](https://pmc.ncbi.nlm.nih.gov/articles/PMC9074854/)

[Analytics Vidhya](https://www.analyticsvidhya.com/blog/2023/02/extracting-medical-information-from-clinical-text-with-nlp)

[MDClone](https://mdclone.com/3-ways-natural-language-processing-nlp-can-impact-healthcare/)

[Orecal](https://docs.oracle.com/iaas/language/using/healthcare-nlp-models.htm)

[Google Cloud](https://cloud.google.com/healthcare-api/docs/concepts/nlp)

[Inovalon](https://www.inovalon.com/blog/natural-language-processing-in-healthcare/)

#### BioMedCLIP

[hugging face](https://huggingface.co/microsoft/BiomedCLIP-PubMedBERT_256-vit_base_patch16_224)

[github/microsoft](https://github.com/microsoft/BiomedCLIP_data_pipeline)

[github](https://github.com/mlfoundations/open_clip/issues/772)

[NEJM](https://ai.nejm.org/doi/full/10.1056/AIoa2400640)

[gitee](https://gitee.com/modelee/BiomedCLIP-PubMedBERT_256-vit_base_patch16_224)

[Microsoft](https://www.microsoft.com/en-us/research/publication/large-scale-domain-specific-pretraining-for-biomedical-vision-language-processing/)

[Connected Papers](https://www.connectedpapers.com/main/5814bd146b37e13115af4330caf3a751159a156f/BiomedCLIP%3A-a-multimodal-biomedical-foundation-model-pretrained-from-fifteen-million-scientific-image%20text-pairs/graph)

#### MedCAT

[github](https://github.com/CogStack/MedCAT)

[medcat](https://medcat.readthedocs.io/en/latest/)

[pypi](https://pypi.org/project/medcat/0.3.7.6/)

[PubMed](https://pubmed.ncbi.nlm.nih.gov/34127232/)

[ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0933365721000762)

[arxiv](https://arxiv.org/abs/2010.01165)

[The SAO Astrophysics Data System](https://ui.adsabs.harvard.edu/abs/2019arXiv191210166K/abstract)

#### גרף רפואי:

[Neo4j](https://neo4j.com/developer/life-sciences-and-healthcare/)

[arxiv](https://arxiv.org/abs/2408.04187)

[neo4j](https://neo4j.com/blog/healthcare/transforming-health-information-technology-knowledge-graph)

[arxiv](https://arxiv.org/html/2306.04802v4)

[read the docs](https://ckg.readthedocs.io/en/latest/INTRO.html)

[ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1532046423001247)