



DeTech



שם התלמיד: מעין מושהיוף

תעודת זהות: 214950941

שם המורה: ניר סליקטר

שם החלופה: תכנון ותכנות מערכות

תאריך הגשה: 28.05.2023



DETECH
PNEUMONIA DETECTION USING AI

תוכן עניינים:

5.....	מבוא
5.....	תכולת הספר
5.....	הרקע לפרויקט
6.....	תהליך המחקר
6.....	סקירת המצב הקיים בעולם
7.....	חידושים ויתרונות בפרויקט
7.....	אתגרים מרכזיים
8.....	למידה לפרויקט
8.....	מוטיבציה לעבודה
9.....	ארכיטקטורה של הפרויקט
9.....	הפתרון המוצע והסיבות לבחירתו
9.....	תרשים של היחידות השונות
10.....	הסבר כללי על היחידות השונות
10.....	אפליקציית לקוח וממשק משתמש GUI
10.....	שרת האפליקציה Application server
10.....	בסיס נתונים database
10.....	שרת אינטרנטי Web server
11.....	Flowchart Diagram
12.....	UML Sequence Diagram
13.....	הצגת המקרים (use case) עבור הפונקציות העיקריות בפרויקט
13.....	הרשמה- Sign up
13.....	התחברות- Login
14.....	צפייה או עריכה של מטופל קיים - View/Edit patient's details
14.....	דיווח על ביקור חדש- Report a visit
15.....	זיהוי תצלום רנטגן - X-ray detection
15.....	צפייה בנתונים סטטיסטיים- Watch statistical analysis
16.....	תרשים זרימה ניווט בין מסכים
17.....	תיאור מסכים
17.....	המסך הראשי- Main Window
18.....	מסך התחברות- Login dialog
19.....	מסך הרשמה- Sign up dialog
22.....	האזור האישי - Personal area
23.....	מסך דיווח על ביקור חדש- Report a visit dialog
26.....	מסך צפייה בניתוח סטטיסטי של תצלומי הרנטגן- Statistical analysis
27.....	מסך צפייה בתוצאות הניתוח של המערכת עבור תצלום רנטגן- Results dialog
28.....	פרוטוקול תקשורת
28.....	תבנית פקודה (בקשה) בין לקוח לשרת
28.....	תבנית פקודה (תשובה) בין שרת ללקוח

28.....	Commands - רשימת הפקודות
29.....	LOGIN
29.....	SIGN_UP
29.....	ADD_PATIENT
29.....	GET_SPECIFIC_PATIENT
30.....	GET_PATIENT_LIST
31.....	טכנולוגיות בהן נעשה שימוש בפרויקט
31.....	בינה מלאכותית לזיהוי תמונה- Artificial Intelligence (הסבר תיאורטי של האלגוריתם)
33.....	יישום הפתרון של הבינה המלאכותית בפרויקט זה
34.....	אבטחת מידע
34.....	הצפנות
36.....	Hashing
37.....	בסיס הנתונים
37.....	הסבר כללי על בסיס הנתונים
37.....	מבנה בסיס הנתונים בפרויקט
37.....	doctors_details_db
37.....	patients_details_db
38.....	קישור בין טבלאות הנתונים
38.....	תרשים המאגר
39.....	תכנות מתקדם (שימוש בפרדיגמות מתקדמות)
39.....	multi tasking programming תכנות בסביבה מרובת תהליכים
39.....	Signal and Slots תכנות בתצורת
39.....	שימוש בקונפיגורציה חיצונית
40.....	סקירת חולשות סיכונים ואיומים
41.....	מימוש הפרויקט
41.....	סקירת מודולים בפרויקט- טכנולוגיות
42.....	סקירת מחלקות, פעולות ומסכים בפרויקט
42.....	myClient
43.....	Ui_MainWindow
45.....	Ui_Login_Dialog
45.....	Ui_Sign_up_Dialog
46.....	Ui_Report_a_visit_Dialog
47.....	Ui_Statistical_analysis_Dialog
48.....	Ui_Results_Dialog
48.....	Worker
48.....	Worker_Dialog
49.....	AppConfig
50.....	AI_model
51.....	בדיקת המערכת
51.....	ניווט בין מסכים
51.....	פעולות בפרויקט
51.....	פעולות בבסיס הנתונים

51.....	תקשורת
51.....	חיזוי המודל
52.....	מדריך למשתמש:
52.....	הוראות התקנה
53.....	עץ קבצים
55.....	רפלקציה
55.....	תחושת מהעבודה על הפרויקט
55.....	מה קיבלתי וכלים שאקח איתי להמשך
55.....	קשיים ואתגרים שעמדו בפני
56.....	מסקנותי מהפרויקט
56.....	מה הייתי עושה אחרת אם הייתי מתחיל את הפרויקט היום
57.....	תכונות שהייתי רוצה להוסיף לפרויקט
58.....	ביבליוגרפיה:
59.....	נספחים
59.....	קישור לסרטון הסבר על הפרויקט
59.....	קישור לקוד המלא ב- GitHub
59.....	מענה על דרישות החובה
60.....	הקוד המלא
60.....	enc_server.py
63.....	enc_client.py
66.....	db.py
69.....	main_window.py
78.....	login_Dialog.py
80.....	sign_up_Dialog.py
85.....	report_a_visit_Dialog.py
94.....	date_picker.py
95.....	send_curl.py
96.....	results_Dialog.py
100.....	statistical_analysis.py
103.....	progress_bar.py
106.....	config.py
107.....	app.py
107.....	AI_model.py
110.....	pneumonia_router.py

מבוא

תכולת הספר

ספר זה מציג בפירוט את פרויקט הסיום שלי במגמת סייבר. הוא מפרט את הרקע לפרויקט, מטרותיו וקהל היעד שלו. כמו כן, הספר כולל תרשימים רבים הממחישים את אופן הפעולה של הפרויקט ומה מתרחש בו, מדריך למשתמש וצילומי מסך אשר יעזרו להמחשה. בנוסף קיים בו גם פירוט על מבנה הנתונים והטכנולוגיות השונות בהם השתמשתי לפיתוח הפרויקט, מדריך למפתח ולבסוף רפלקציה אישית וביבליוגרפיה על הנעשה.

הרקע לפרויקט

במחשבה על פרויקט סיום המגמה הצבתי כמה מטרות שחשובות בעיני שיתממשו בפרויקט. רציתי שהפרויקט שלי יוכל לענות על בעיה או צורך בעולם הנוכחי תוך שילוב חדשנות והגדלת הידע בתחומים חדשים. בתהליך החשיבה בחרתי בתחום הבינה המלאכותית. תחום זה מסקרן אותי מאוד. בימים אלו, תחום זה מאוד פופולרי ונמצא במגמת נסיקה. היכולות העצומות של מכונה לומדת (machine learning) זמינות לפיתוח אפליקציות חדשות. כמו למשל, צ'אט GPT. החלטתי לתעל את הטכנולוגיה המתקדמת כתשתית לפיתוח מערכת בריאות שתייעל ותשפר תהליך קיים בזהיו של מחלת דלקת ריאות באמצעות ניתוח צילומי רנטגן.

ערכתי סקירה אינטרנטית ומצאתי כי ישנם כלי בינה מלאכותית המסוגלים לפענח צילומי רנטגן. כמו כן ע"פ תשאול רופאים ובירור, הבנתי שלא קיים כלי תפעולי שנותן מענה ויכולת איבחון למשתמשי מערכת הבריאות.

כאמור פרויקט זה עושה שימוש במידע רפואי רגיש, ולפיכך יינתן דגש על אבטחת המידע והתקשורת, זאת על מנת להגן על פרטי הרופאים והמטופלים.

הצורך עליו עונה הפרויקט

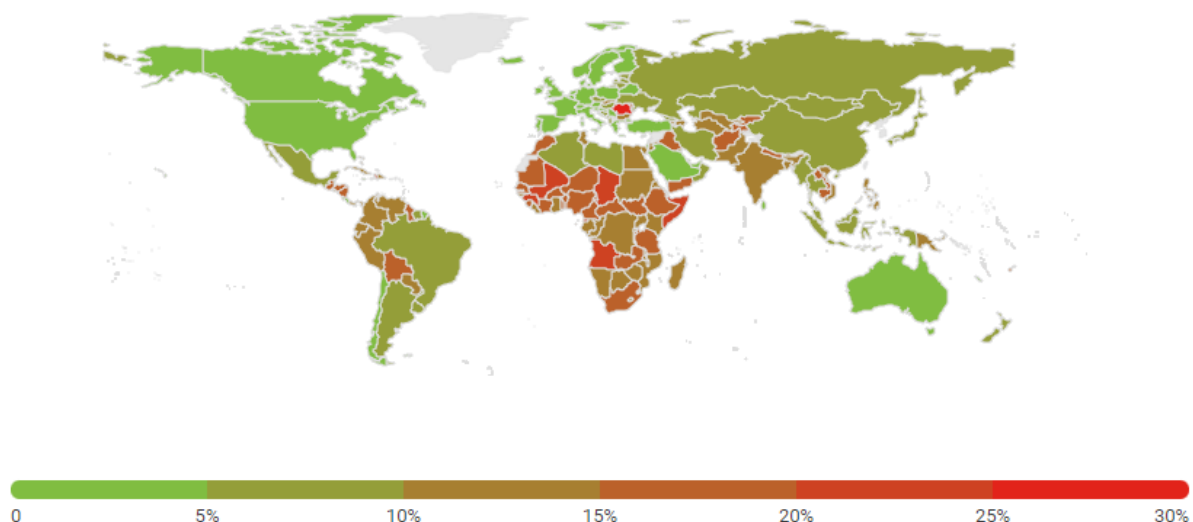
בפרויקט זה אציג פיתוח של יכולת פענוח דלקת ריאות המבוססת בינה מלאכותית, על גבי תשתית זו אפתח כלי עזר רפואי לניהול מטופלים ע"י רופא (או רדיולוג) שיאפשר שמירת נתוני מטופל ואבחון מחלה מיידי וממוחשב. המערכת מיועדת לתת מענה לצורך של ייעול התהליך הנוכחי בעולם הרפואה של פענוח תצלומי רנטגן. המערכת אינה מחליפה את שיקול דעתו של הרופא או המפענח אלא נועדה לסייע לו ולהוות כאמצעי ביקורת לשיקול דעתו. כמו כן, המערכת תתופעל על ידי אפליקציה נוחה לניהול מטופלים. נוסף על כך, היכולת של המערכת יכולה להניב יתרונות נוספים כפי שיפורט בפרק הבא [חידושים ויתרונות בפרויקט](#).

תהליך המחקר

סקירת המצב הקיים בעולם

בתחילה ערכתי מחקר בנוגע למחלת דלקת ריאות. גיליתי שהמחלה מדבקת ושכיחה מאוד, במיוחד בקרב ילדים מתחת לגיל 5. ישנו מקרה אחד לכל 71 ילדים מדי שנה, כאשר השכיחות הגדולה ביותר מתרחשת בדרום אסיה (2,500 מקרים לכל 100,000 ילדים) ובמערב ומרכז אפריקה (1,620 מקרים לכל 100,000 ילדים). המחלה יכולה להתברר גם כקטלנית. דלקת ריאות הורגת יותר ילדים מכל מחלה זיהומית אחרת, וגובה את חייהם של למעלה מ-700,000 ילדים מתחת לגיל חמש מדי שנה (כ-2,000 בכל יום). כמעט כל מקרי המוות הללו ניתנים למניעה! גיליתי שצילום חזה יכול לאשש או להפריך במידה רבה את קיומה של המחלה ברמת וודאות גבוהה מאוד. באזורים כמו אפריקה, הבעיה עלולה להחמיר עוד יותר עקב מחסור במשאבים רפואיים ובכוח אדם. עבור אוכלוסיות אלו, אבחון מדויק ומהיר יכול להבטיח גישה בזמן לטיפול, להציל חיים ולחסוך זמן וכסף. (ראה נתונים: [ourworldindata](https://ourworldindata.org), [wikipedia](https://en.wikipedia.org/wiki/COVID-19), [unicef](https://www.unicef.org)).

- שיעור אחוזי התמותה שנגרמו כתוצאה מדלקת ריאות בקרב ילדים מתחת לגיל 5 (2019):



בפרויקט זה השתמשתי במאגר צילומים שעליו מבוסס מודול הבינה המלאכותית. המאגר מכיל כ-5800 צילומי חזה שנעשו באוכלוסיית ילדים בגילאי 0 עד 5 בעיר גואנגג'ואו שבסין (ראה [Kaggle](https://www.kaggle.com/datasets/kuozhuo/covid19) [dataset here](https://www.kaggle.com/datasets/kuozhuo/covid19)).

בתהליך הנוכחי קיים זמן המתנה ארוך (אשר יכול להגיע אפילו למספר ימים) לצורך פענוח צילום רנטגן וקבלת דיאגנוזה לטיפול. בתהליך עיבוד ממוחשב המבוסס בינה מלאכותית, ניתן יהיה לפענח צילום באופן מיידי לספק מענה טיפולי מהיר ואופטימלי למטופל שממתין.

חידושים ויתרונות בפרויקט

- היכולת לפענח ממוחשב מהווה בסיס ליישום אפליקציות ושימושים שונים -
- כלי חדש ויעיל מפענח צילומי רנטגן, עבור רופאים ורדיולוגים לניהול, ואבחון מטופלים. **** יכולת זו מומשה במסגרת פרוייקט זה.**
- ניתוח סטטיסטי והפקת דוחות לצורך התייעלות תקציבית ע"י קבלת מסקנות אופרטיביות לעתיד.
- למשל: "רק 20% מכלל הצילומים ברבעון העידו על מחלה", מכאן אפשר לחשוב על התייעלות וחסכון תקציבי בצילומי שווא.
- ** יכולת זו הודגמה במסגרת פרוייקט זה.**
- סקירה של כמות גדולה מאוד של צילומים שתספק יכולת מהירה לאבחון תרחישי קיצון באוכלוסיה. (לדוגמא באיזורים כמו אפריקה).
- הימנעות מקרינת יתר עבור מטופלים (ושמירה על סביבה ירוקה).
- מתוך ניתוח ופענוח תצלומי עבר יהיה ניתן לדייק יותר בהפניה לצילום ולהימנע מצילומי שווא. כמו כן, טיפול מהיר במחלה יגרום להדבקה מופחתת ובכך להפחתת כמות צילומי רנטגן נוספים.
- קיצור תהליך קיים ומתן טיפול מהיר.

אתגרים מרכזיים

תוכן הפרויקט הציב בפני אתגרים משמעותיים רבים עימם התמודדתי בכוחות עצמי.

ראשית התמודדתי עם אתגר ההיתכנות, האם הרעיון יכול לקרום עור וגידים האם אוכל ללמוד וליישם מערכת עובדת. לשם כך השקעתי משאבים רבים כבר בתחילת הפרוייקט למחקר ומימוש מודול AI עובד וראשוני לגמרי, רק כדי להבין שזה אפשרי.

למידה של טכנולוגיה חדשה כפי שאפרט בהרחבה עוד בהמשך [הפרק הבא](#) הנושא הנלמד הינו "בינה מלאכותית" החל מתאוריה, קריאת ספרים ומאמרים ועד לשלבי יישום וקידוד בפועל של אפליקציה עובדת.

היכרות עם כליי פיתוח חדשים לעיצוב ממשק ממשתמש (ניסיון והערכה של מגוון כלים מתוכם בחרתי את המתאימים). גם בחלק זה ערכתי מחקר רב.

ארגון והקמה של סביבת ריצה הכוללת את כל המודולים הנדרשים לריצה. במהלך חלק זה נתקלתי בהמון בעיות טכניות (גרסאות לא תואמות, כישלון בהורדת מודולים וכו').

עמידה בלוח זמנים במסגרת זמן מוגבלת לביצוע (תוך תקופת בחינות בגרות אינטנסיבית).

למידה לפרויקט

לאחר התגבשות הרעיון הכללי לפרויקט הסיום התחלתי במלאכה. הפרויקט שבחרתי מכיל חומר רב מעבר למעבר לנלמד במסגרת לימודי המגמה בבית הספר. את הנושאים הבאים למדתי באופן עצמאי בקריאה, צפייה במדריכים, קוד שיתופי פתוח והרבה התנסות אישית.

בינה מלאכותית

אלגוריתם לזיהוי אובייקטים, מהי רשת עצבית neural network, בניית מודל נתונים, אימון מודל שימוש ותשאול כיצד באופן מעשי בונים יכולת זיהוי דלקת ריאות.

בניית ממשק משתמש חלונאי

זהו תחום פיתוח שלמדתי והתמודדתי עמו ללא ניסיון מקדים, נדרש לי ידע בתחום ולכן התחלתי במחקר אודות הנושא ואחר כלים מתאימים לעבודה. ניסיתי כלים כמו pygame, tkinter אך חיפשתי כלי שאוכל לעצב אותו באופן ויזואלי ולאחר מכן להפוך אותו לקוד פעיל, כך העבודה תהיה היעילה ביותר. מצאתי כלי ששמו qt designer שבו ניתן לעצב חלונות ומסכים באופן ויזואלי, יעיל וקל. לאחר עיצוב החלון ניתן להפוך את הקובץ שהופק לקובץ py באמצעות פקודה ב-cmd. הכלי עושה שימוש בספרייה PyQt5. גם במהלך העבודה עם הכלי נתקלתי בבעיות שונות. למשל, פתיחת חלון חדש לאחר סגירת הישן. פעולות אלו נראות כטריוויאליות אך דורשות המון למידה והשקעה מאחורי הקלעים. נעזרתי בסרטונים ובמחקר אינטרנטי על מנת להתגבר על קשיים אלו.

בניית אפליקציה בארכיטקטורת שרת לקוח

תקשורת מבוססת sockets על גבי פרוטוקול TCP/IP.

שימוש בבסיס נתונים

כתיבה קריאה ועידכון רשומות באמצעות שפת SQL.

Multitasking programming

תכנות בריצה מקבילית לצורך פתרון בעיות של המתנה ארוכה לביצוע פעולה (שימוש ב-threads).

מוטיבציה לעבודה

ביצוע הפרויקט היה כרוך בהמון אנרגיה וזמן. לכן, לא הייתי מצליח לסיים את הפרויקט לולא המוטיבציה הרבה שהייתה לי. את המוטיבציה לעשיית הפרויקט שאבתי מתוך העניין הרב בתחום הנלמד החדש, הבינה המלאכותית. נושא זה ריתק אותי ונשאבתי לשעות ארוכות של למידה על התחום החדש וכן לעבודה על ביצוע הפרויקט. כמו כן, הפרויקט שבחרתי יכול בעתיד להתפתח ולהוות כלי ממשל לסיוע בעולם הרפואה. בשל כך, לאורך ביצוע הפרויקט חשתי שהעבודה שאני מבצע הינה חשובה מאוד ומועילה.

ארכיטקטורה של הפרויקט

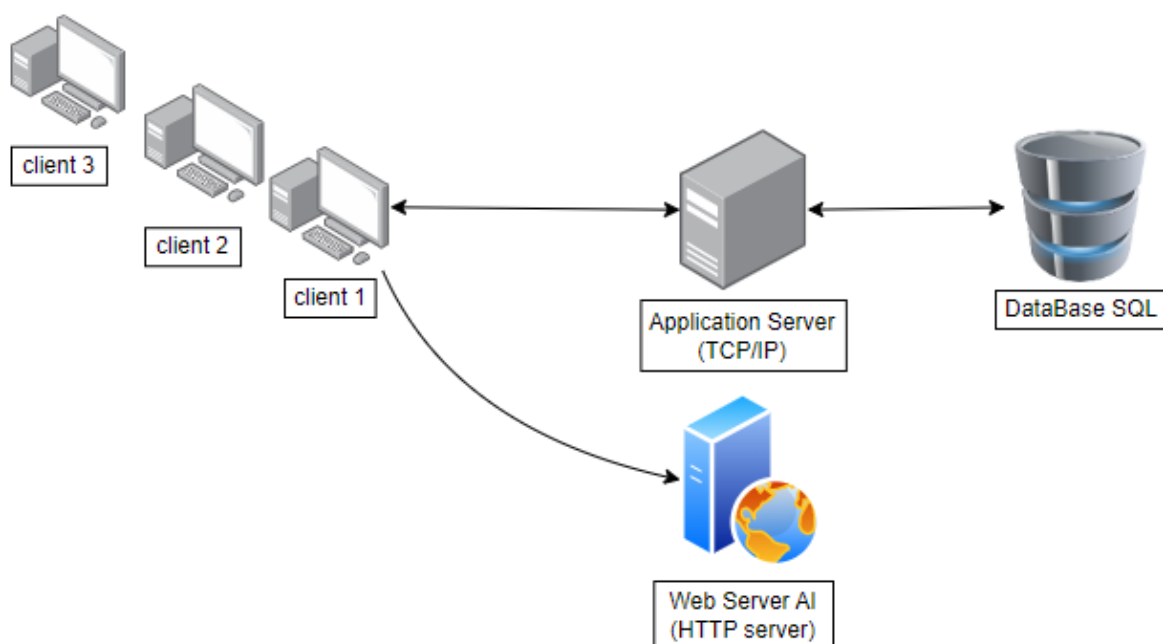
הפתרון המוצע והסיבות לבחירתו

- הפתרון כולל מערכת עם מספר רכיבים
- המערכת בנויה בתצורת שרת לקוח
 - אפליקציית לקוח עם ממשק חלונאי דרכה המשתמש יכול לבצע פעולות.
 - קיימת תמיכה במספר מערכות הפעלה multi platform support האפליקציה תוכל לרוץ במחשב לקוח Windows או Mac (נבדק).
 - שרת אפליקציה TCP/IP שמקבל את הבקשות ומבצע את הפעולות הנדרשות
 - מאגר נתונים המכיל את פרטי המשתמשים במערכת ואת פרטי המטופלים
 - שרת Web HTTP שמספק גישה אל מודל הבינה המלאכותית מקבל תצלום ומבצע פענוח רנטגן של דלקת ריאות.

- בחרתי במודל שרת-לקוח כדי לאפשר -
- גישה וזמינות** לשירות מכל מקום.
- אבטחת מידע** מאגר הנתונים נמצא בשרת אחד שאותו ניתן לאבטח.
- תעבורת מידע מאובטחת** התקשורת בין הרכיבים מוצפנת ובטוחה.
- פיתוח תחזוקה** שירותים נוספים שינויים או תיקונים יכולים להתבצע בשרת מבלי לשנות את אפליקציית המשתמש.

תרשים של היחידות השונות

במבט העל יש ניתן לראות את שני השרתים, database וכמה לקוחות.



הסבר כללי על היחידות השונות

אפליקציית לקוח וממשק משתמש GUI

אחריות על האינטראקציה של המשתמש עם התוכנה והשרתים. היא בנויה ממסכים שעוצבו באמצעות PyQt5. הפיתוח בשפת פייתון מאפשר תמיכה במספר מערכות הפעלה multi platform support האפליקציה תוכל לרוץ במחשב לקוח Windows או Mac (נבדק).

המשתמש נרשם או מתחבר למערכת באמצעות פרטים אישיים ולאחר מכן הוא נכנס לאזור האישי שלו. הממשק מציג עבור המשתמש (רופא/רדיולוג) את רשימת המטופלים שלו, אותם הוא יכול למיין ולסווג לפי מספר פרמטרים. בנוסף, הממשק מאפשר למשתמש לבצע פעולות באמצעות תקשורת עם השרתים. הוספת מטופל, עריכת פרטי מטופל, בקשה לפענוח זיהוי צילום רנטגן וצפייה בפרטי מטופל קיימים.

שרת האפליקציה Application server

יש לו כמה תפקידים עיקריים:

1. מבצע בפועל את הקישוריות למאגר הנתונים. מבצע שמירה, איחזור ועריכה של נתונים.
2. תומך בפעולות Log in ו-Sign up.
3. תומך בפעולות הוספת מטופל, קבלת רשימת המטופלים, וקבלה של מטופל ספציפי.
4. אבטחת המידע. כאמור, פרטיהם של הרופאים והמטופלים מכילים מידע רגיש. (למשל, סיסמאות ומידע רפואי), השרת מיישם הצפנה. אחראי על אבטחת התקשורת בין השרת ללקוח ושומר סיסמאות בצורה מאובטחת (מוצפנת).

בסיס נתונים database

בסיס נתונים מסוג SQLite.

שומר בתוכו מידע:

- על המשתמשים: אימייל, שם מלא וסיסמה.
 - על המטופלים: שם מלא, מספר תעודת זהות, מין, תאריך לידה, כתובת אימייל, תאריך ביקור אחרון אצל הרופא, תיאור המקרה (המחלה) ונתונים לגבי זיהוי דלקת ריאות.
- הגישה לבסיס הנתונים מתבצעת ע"י שרת האפליקציה והפעולות על בסיס הנתונים מתבצעות ע"י פקודות SQL.

שרת אינטרנטי Web server

שרת זה מכיל את המודל של הבינה המלאכותית.

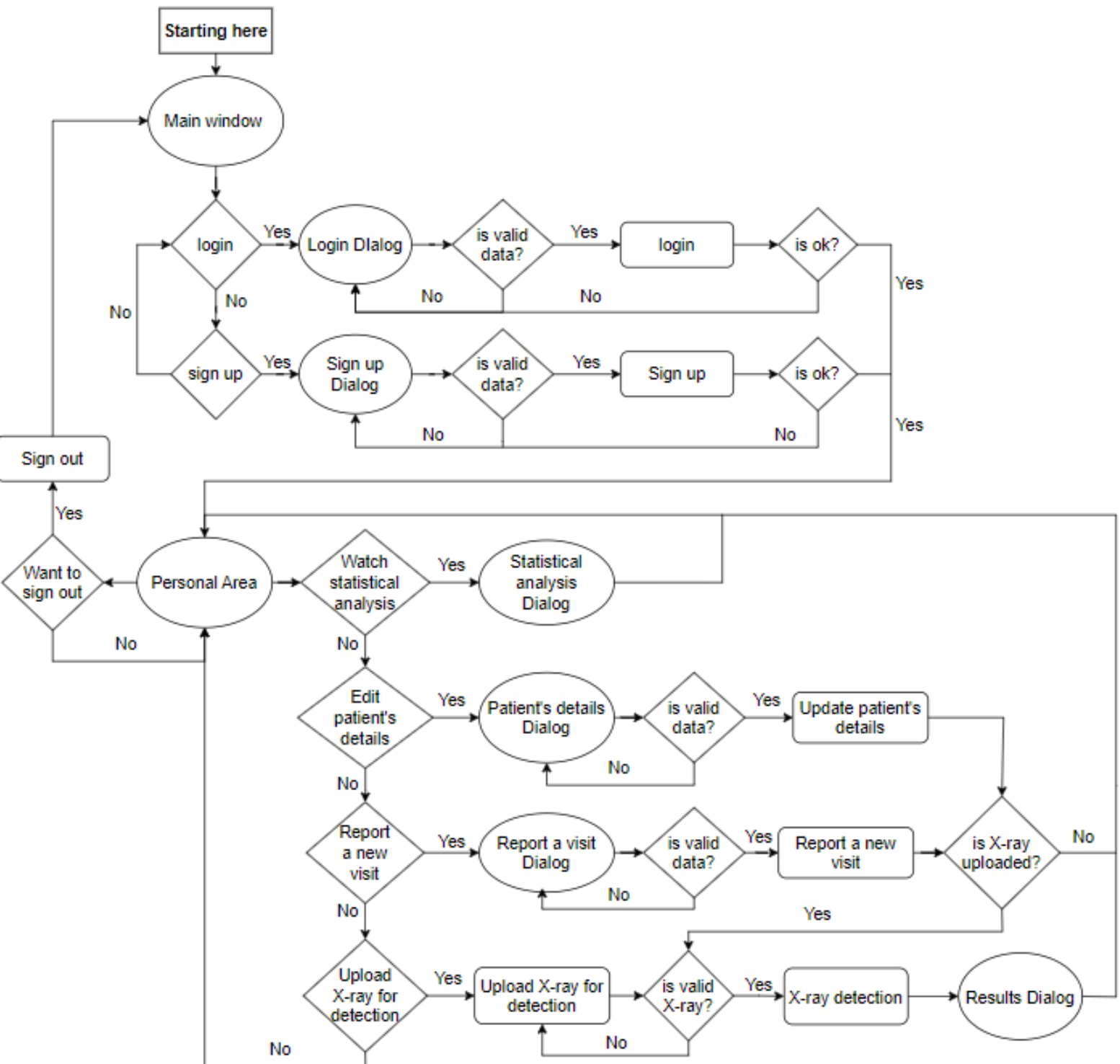
הוא מכיל שכבת תקשורת שמספקת גישה חיצונית למודל באמצעות HTTP POST Request

****אותה בפועל יוזם המשתמש באפליקציה הלקוח**

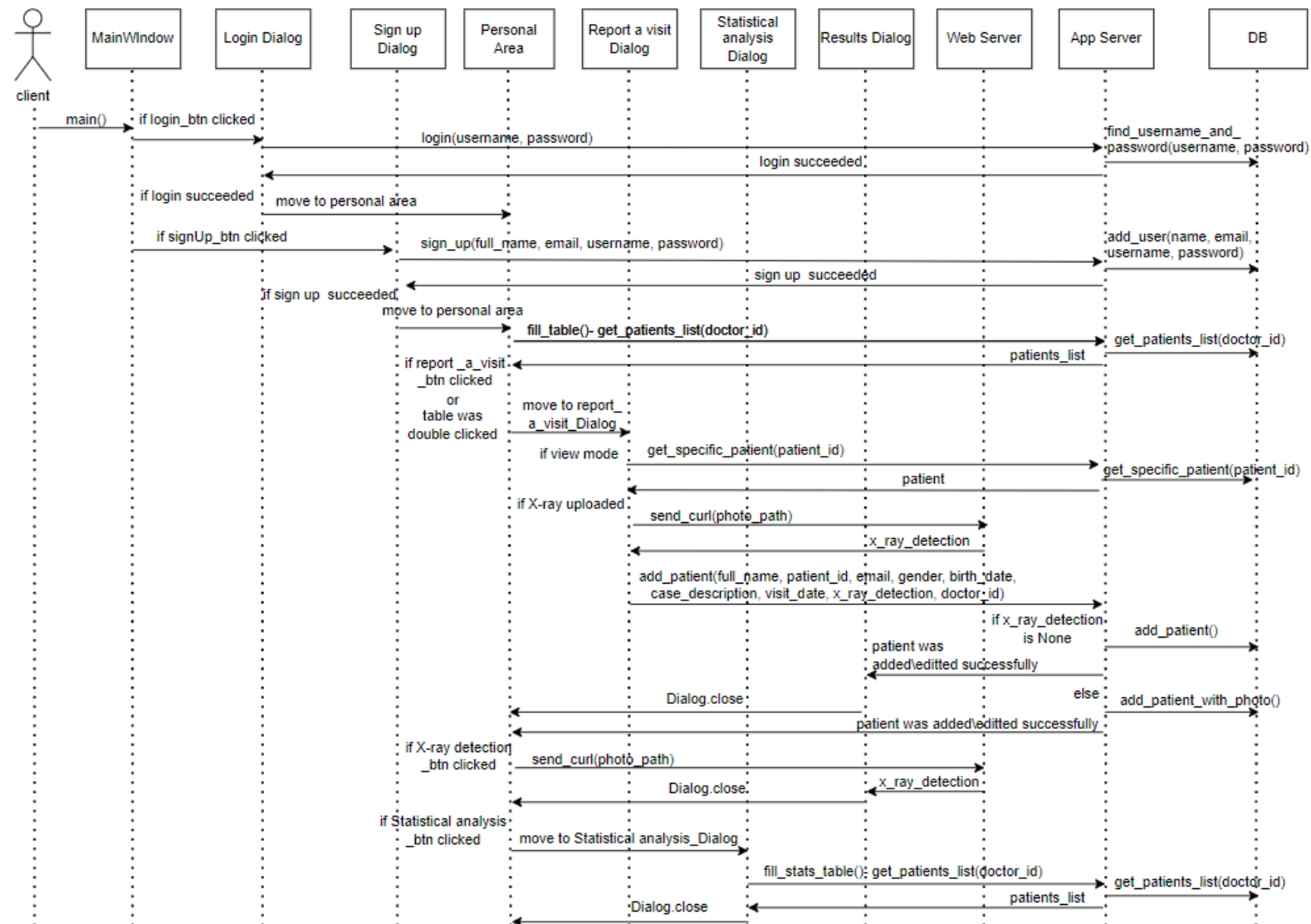
באמצעות הקריאה, ניתן להעלות קובץ צילום רנטגן שמעובד בשכבה הקולטת (שימוש בספריות pillow and keras)

ממנה מתבצעת גישה למודל ע"י ספריית פיתוח keras over tensorflow לשם פענוח דלקת ריאות.

Flowchart Diagram



UML Sequence Diagram



הצגת המקרים (use case) עבור הפונקציות העיקריות בפרויקט

הרשמה - Sign up

תיאור הפעולה: המשתמש מנסה להירשם למערכת.

תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה והוא נכנס למסך הראשי וממנו אל מסך ה – Sign up
תנאי סיום: המשתמש רשום במערכת.

שלבי פעולה:

1. המשתמש נכנס למסך ההרשמה.
2. המשתמש ממלא את השדות הריקים (שם מלא, אימייל, שם משתמש, סיסמא, אישור סיסמא), בהתאם לכתוב ליד השדות.
3. לאחר לחיצה על כפתור ה – Sign up. המערכת מוודאת שפרטי המשתמש תקינים ועושה מספר בדיקות.
 - i כל השדות מלאים
 - ii אימייל תקין
 - iii סיסמא בעלת אורך 6 אותיות ומכילה קומבינציה של אות גדולה אות קטנה ומספר
 - iv סיסמאות זהות
4. לאחר הכנסת הפרטים תקינים, פרטי המשתמש ישלחו לשרת האפליקציה. השרת יבצע בדיקה האם קיים משתמש בעל שם משתמש זהה, אם כן הוא יבקש מהמשתמש להזין שם משתמש שעוד לא קיים במערכת. במקרה ששם המשתמש לא קיים במערכת, השרת יזין את נתוני המשתמש ל-database.
5. לאחר הזנת הפרטים של הלקוח בבסיס הנתונים בהצלחה, הלקוח יועבר למסך הבית של האפליקציה והאזור אישי.

התחברות - Login

תיאור הפעולה: המשתמש מנסה להתחבר למערכת.

תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה והוא נכנס הראשי וממנו אל מסך ה – Login.
תנאי סיום: המשתמש מתחבר למערכת.

שלבי פעולה:

1. המשתמש נכנס למסך ההתחברות.
2. המשתמש ממלא את השדות הריקים (שם משתמש וסיסמא), בהתאם לכתוב ליד השדות.
3. לאחר לחיצה על כפתור ה – Login מתבצעת בדיקה בצד של הלקוח לגבי הקלטים. כלומר, בדיקה האם אחד השדות ריקים. במקרה כזה, המשתמש יתבקש למלא את הפרטים החסרים שוב.
4. לאחר בדיקה אצל הלקוח, המידע נשלח לשרת האפליקציה. שם מתבצעת בדיקה האם שם המשתמש והסיסמא אכן קיימים בבסיס הנתונים. אם לא, השרת ישלח הודעה ללקוח על כך ובמסך ה-Login תופיע הודעה על כך ששם המשתמש או הסיסמא אינם נכונים, ושצריך להכניסם שוב.

5. לאחר הכנסת פרטים נכונים ותקינים, הלקוח יתחבר למערכת והוא יועבר למסך הבית של האפליקציה ולאזור האישי.

צפייה או עריכה של מטופל קיים - View/Edit patient's details

תיאור הפעולה: המשתמש רוצה לצפות בפרטים של אחד ממטופליו.
תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה, הוא מחובר למערכת והוא לחץ לחיצה כפולה על שורה מסויימת בטבלה.
תנאי סיום: המשתמש מגיע למסך הצפייה במטופל קיים.
שלבי פעולה:

1. המערכת מזהה את השורה המבוקשת שהמשתמש לחץ עליה פעמיים.
2. המערכת ניגשת למאגר הנתונים, מאתרת את המידע בטבלה, פותחת חלון שבו מוזנים כל פרטי המטופל המבוקש
3. את הפרטים של המטופל ניתן לערוך, למעט השם המלא שלו, מספר תעודת הזהות שלו ותאריך הלידה שלו.
4. לכל ביקור של מטופל ניתן להעלות תמונה אחת בלבד של דלקת ריאות למערכת. אם עוד לא צורפה תמונה למטופל, ניתן להוסיף בשלב זה תמונה.
5. לאחר לחיצה על כפתור Submit, פרטי המטופל נשלחים אל שרת האפליקציה. השרת מקבל אותם ומעדכן את פרטיו החדשים של המטופל ב-database.
6. במקרה שהמשתמש אכן הוסיף נתיב של שם קובץ תמונת רנטגן, המערכת תבצע קריאה אל שרת ה AI ותציג עבורו מסך בו יוצגו לו תוצאות הניתוח של התמונה שהזין

דיווח על ביקור חדש - Report a visit

תיאור הפעולה: המשתמש רוצה לדווח על ביקור חדש של מטופל.
תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה, הוא מחובר למערכת ונמצא במסך הראשי ממנו הוא לוחץ על כפתור report a visit.
תנאי סיום: המשתמש דיווח על ביקור של מטופל ופרטיו נשמרו במערכת
שלבי פעולה:

1. המערכת פותחת חלון בו ניתן למלא את פרטי המטופל שהגיע לביקור. הפרטים: שם מלא, תעודת זהות, אימייל, תאריך לידה, תיאור הביקור, תאריך הביקור וניתוב לתמונת רנטגן (אופציונלי).
2. המערכת מוודאת שפרטי המטופל תקינים ועושה מספר בדיקות.
 (i) כל השדות מלאים (למעט שדה הניתוב לתמונת רנטגן שיכול להישאר ריק)
 (ii) תעודת זהות שעשויה ממספרים בלבד ואורכה תקין
 (iii) אימייל תקין
3. במקרה שהוזן שם קובץ של תמונת רנטגן המערכת תשלח אותו אל שרת ה-web, שם התמונה תנותח. השרת יחזיר תשובה האם קיימת בתמונה שהוזנה דלקת ריאות או לא. במקרה שהמשתמש לא הזין תמונה, לא תשלח תמונה לשרת.
4. הפרטים של הביקור של המטופל שהוזן יישלחו לשרת האפליקציה והוא ישמור אותם ב-database.
5. לאחר שמירת הפרטים, במקרה שהוזן שם קובץ של תמונת רנטגן המערכת תציג עבור המשתמש מסך בו יוצגו לו תוצאות הניתוח של התמונה שהזין.

זיהוי תצלום רנטגן - X-ray detection

תיאור הפעולה: המשתמש רוצה לקבל אבחנה רפואית בנוגע לתצלום רנטגן של בית חזה.
תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה, הוא מחובר למערכת טנמצא במסך הראשי ממנו הוא לחץ על כפתור X-ray detection .
תנאי סיום: המשתמש מקבל את אבחנת שרת ה-web בנוגע לתצלום.
שלבי פעולה:

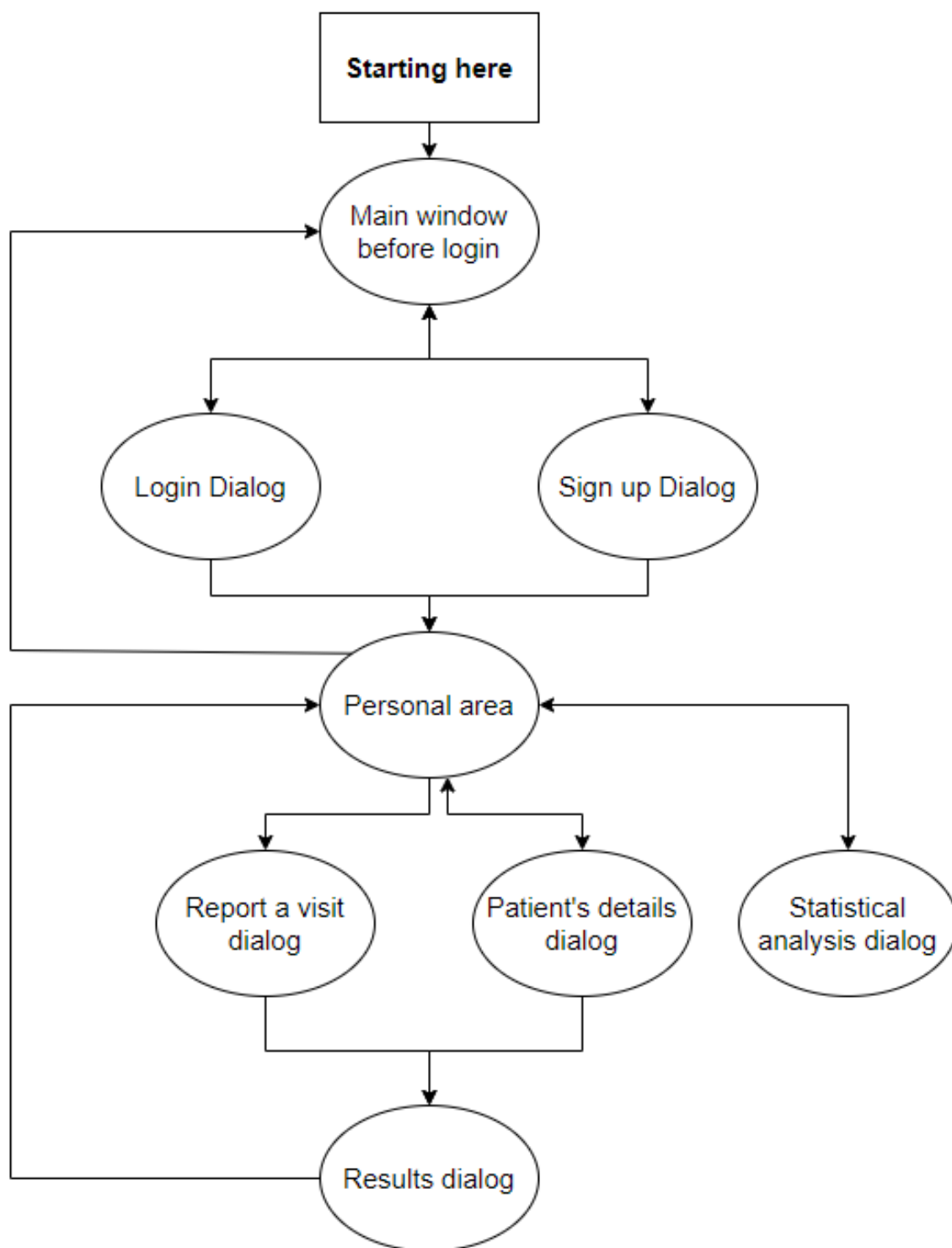
1. המשתמש נדרש לבחור מן המחשב את תצלום הרנטגן.
2. המערכת תשלח את הניתוב אל הקובץ הקובץ שהוזן אל שרת ה-web, שם התמונה תנותח. השרת יחזיר תשובה האם קיימת בתמונה שהוזנה דלקת ריאות או לא. במקרה שהמשתמש לא הזין תמונה, לא תשלח תמונה לשרת.
3. המערכת תציג עבור המשתמש מסך בו יוצגו לו תוצאות הניתוח של התמונה שהזין.

צפייה בנתונים סטטיסטיים - Watch statistical analysis

תיאור הפעולה: המשתמש רוצה לצפות בנתונים סטטיסטיים אודות כלל תצלומי הרטנגן שהועלו עד כה על ידי אל המערכת.
תנאים מקדימים: המשתמש הפעיל את האפליקציה, יש לו חיבור פעיל עם שרת האפליקציה, הוא מחובר למערכת ונמצא במסך הראשי ממנו הוא לחץ על כפתור Statistical analysis .
תנאי סיום: מסך הניתוח הסטטיסטי פתוח.
שלבי פעולה:

1. על המשתמש להיכנס למסך התצוגה.
2. כאשר המסך נפתח יוצגו לפניו תוצאות הניתוח הסטטיסטי וכן טבלת כל תצלומי הרנטגן שהועלו עד כה לצד תוצאותיהם.
3. המשתמש יוכל לצפות בניתוח הסטטיסטי ולמיין את הטבלה לפי הפרמטרים. (אחוזי דיוק, תוצאה).

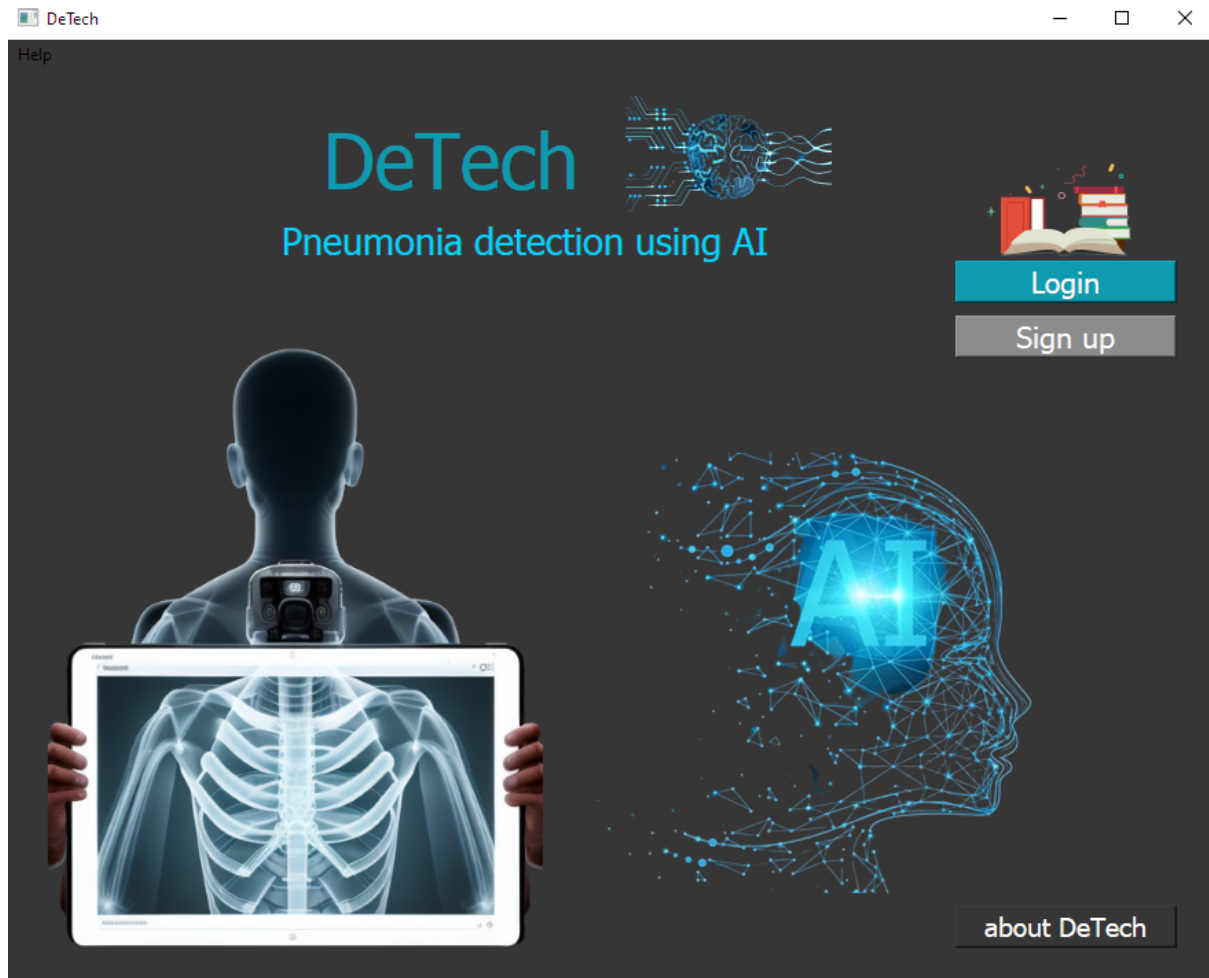
תרשים זרימה ניווט בין מסכים



תיאור מסכים

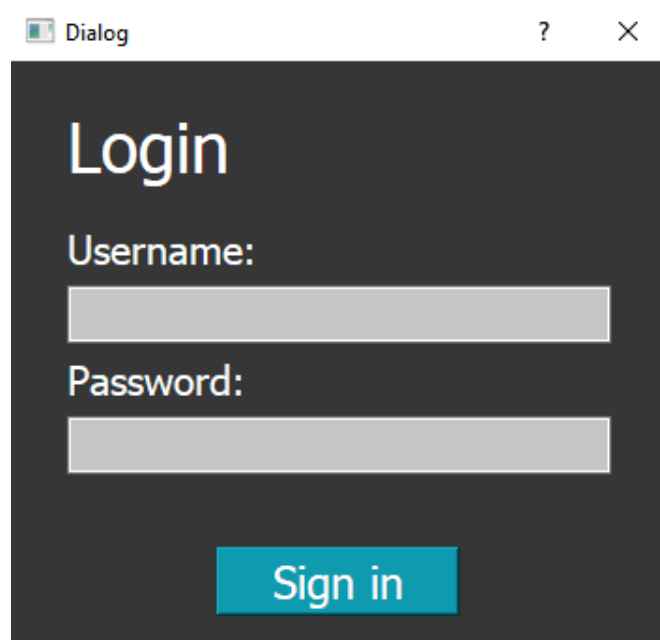
המסך הראשי- Main Window

דרך מסך זה ניתן להירשם לאפליקציה או להיכנס כמשתמש רשום. בנוסף, ישנה אפשרות לקרוא אודות האפליקציה ולהגיע לספר הפרויקט באמצעות לחיצה על כפתור about DeTech.



מסך התחברות- Login dialog

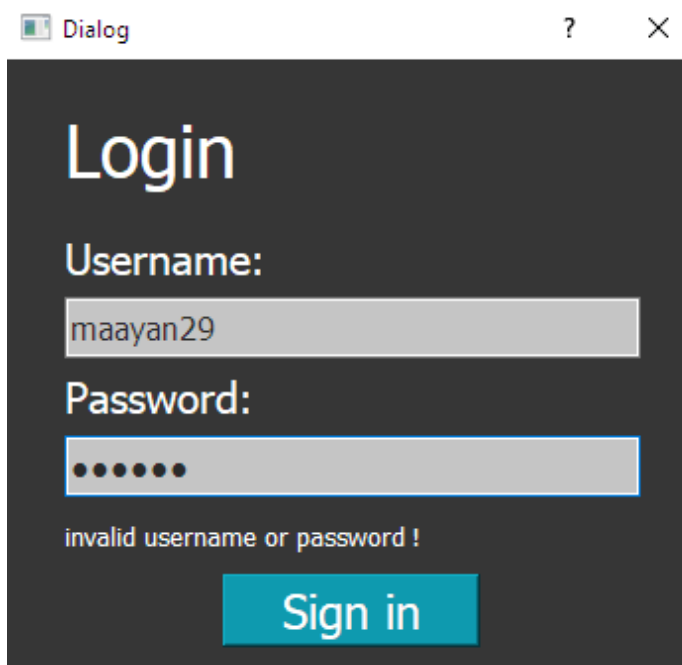
המסך שהמשתמש רואה כאשר הוא לוחץ על כפתור ה-Login במסך הראשי. המשתמש נדרש בחלון זה להזין שם משתמש וסיסמא. במקרה של שגיאה, בעקבות הוולידציה, תוצג הודעה מתאימה למשתמש. במקרה של הצלחה, יועבר המשתמש למסך הראשי (האזור האישי של המשתמש).



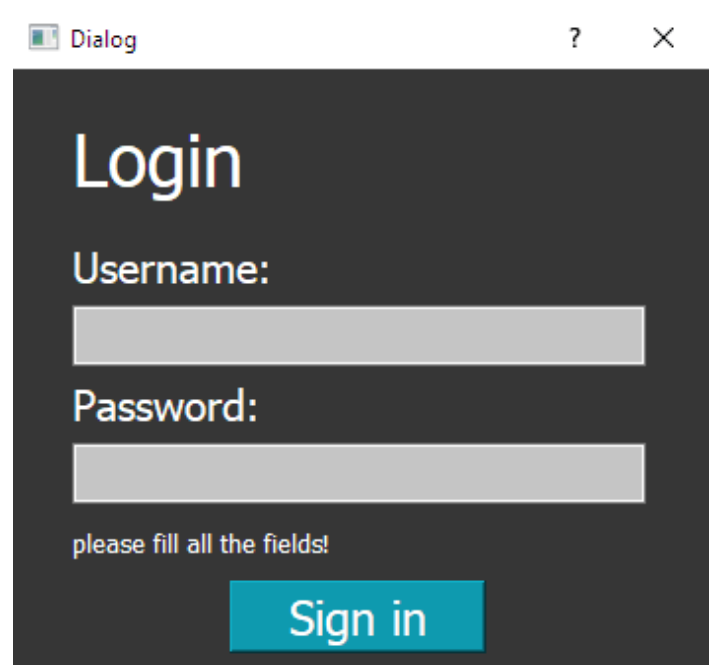
A screenshot of the initial 'Login' dialog box. It has a dark gray background with the title 'Login' in white. Below the title are two input fields: 'Username:' and 'Password:'. At the bottom is a blue button labeled 'Sign in'.

בדיקת האימייל והסיסמא מול השרת ובסיס הנתונים. במקרה זה הנתונים אינם נכונים.

בדיקת תקינות האימייל והסיסמא אצל הלקוח.



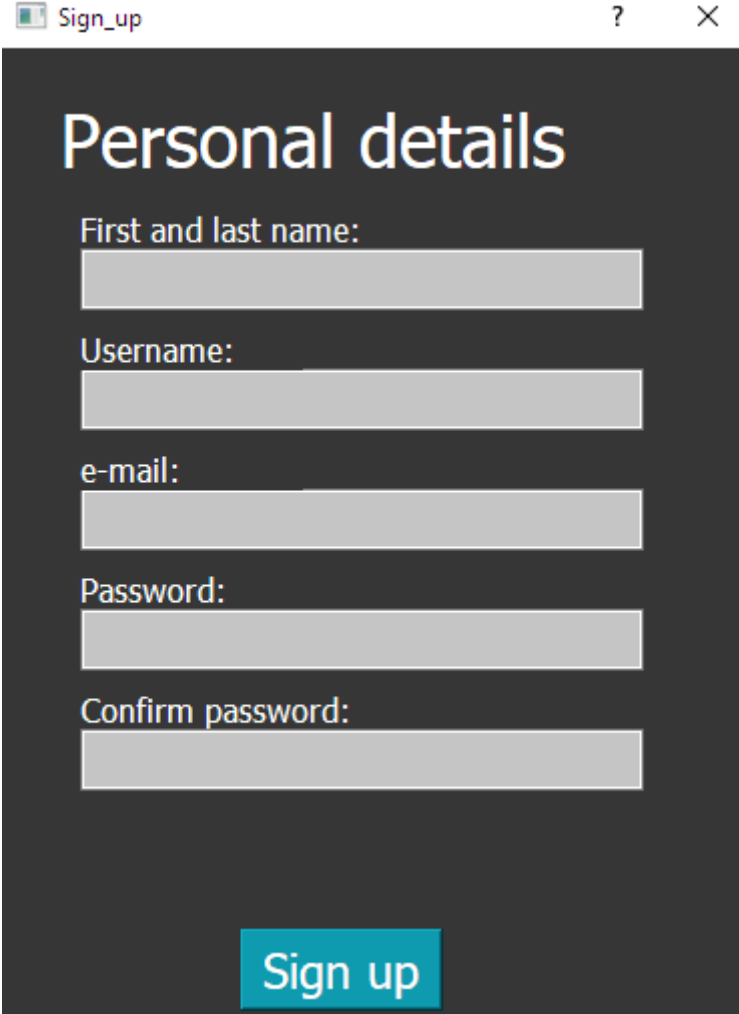
A screenshot of the 'Login' dialog box after an invalid login attempt. The 'Username:' field contains the text 'maayan29'. The 'Password:' field is filled with dots. Below the fields, the text 'invalid username or password !' is displayed. The 'Sign in' button remains at the bottom.



A screenshot of the 'Login' dialog box after an attempt with empty fields. The 'Username:' and 'Password:' fields are empty. Below the fields, the text 'please fill all the fields!' is displayed. The 'Sign in' button remains at the bottom.

מסך הרשמה- Sign up dialog

מסך ליצירת משתמש חדש באפליקציה. קלט: שם מלא, שם משתמש, אימייל סיסמא ואישור סיסמא. אני מבקש אימייל משום שבאמצעותו ניתן (בעתיד) לאפס סיסמא ולוודא שבעל כתובת האימייל הוא זה שמנסה להירשם. במקרה של שגיאה, בעקבות הוולידציה, תוצג הודעה מתאימה למשתמש. במקרה של הצלחה, ירשם המשתמש בהצלחה במערכת ונתוניו יוזנו ל-database. לאחר מכן הוא יועבר למסך הראשי (האזור האישי של המשתמש).

A screenshot of a 'Sign up' dialog box. The window title is 'Sign_up' with a question mark and a close button. The dialog has a dark background and is titled 'Personal details' in large white text. It contains five input fields with labels: 'First and last name:', 'Username:', 'e-mail:', 'Password:', and 'Confirm password:'. At the bottom right is a blue button with the text 'Sign up' in white.

Sign_up

Personal details

First and last name:

Username:

e-mail:

Password:

Confirm password:

Sign up

בדיקת תקינות האימייל אצל
הלקוח.

בדיקת תקינות הפרטים אצל
הלקוח.

Sign_up

? X

Personal details

First and last name:

Username:

e-mail:

Password:

Confirm password:

please enter a valid email!

Sign_up

? X

Personal details

First and last name:

Username:

e-mail:

Password:

Confirm password:

please fill all the fields!

בדיקת שם המשתמש מול
השרת ובסיס הנתונים.
במקרה זה קיים כבר שם
משתמש בעל שם משתמש
זהה.

בדיקת תקינות הסיסמא:

- אורכה לפחות 6 אותיות
- מכילה אות קטנה באנגלית
- מכילה אות גדולה באנגלית
- מכילה ספרה

Sign_up ? X

Personal details

First and last name:
maayan moshayov

Username:
maayan29

e-mail:
mmay@gmail.com

Password:
●●●

Confirm password:
●●●

not a valid password. enter a password that at least 6 characters long with a combination of uppercase letters, lowercase letters, numbers

Sign up

Sign_up ? X

Personal details

First and last name:
maayan moshayov

Username:
maayan29

e-mail:
mmay@gmail.com

Password:
●●●●●●

Confirm password:
●●●●●●

username already exists!

Sign up

האזור האישי - Personal area

האזור האישי של האפליקציה הוא המסך המוצג לאחר הכניסה למערכת.
דרך מסך זה המשתמש יכול לבצע את כל הפעולות שקיימות באפליקציה. למסך זה פונקציונליות רבה:

- ניתן לחזור אל המסך הראשי ולהתנתק מהמשתמש באמצעות לחיצה על כפתור Sign out.
- דרכו ניתן לצפות ברשימת המטופלים של המשתמש.
- באמצעות לחיצה כפולה על אחת משורות טבלת המטופלים ניתן להגיע לחלון צפייה בפרטיו-Patient's details.
- דרך מסך זה ניתן להגיע למסך דיווח על ביקור חדש של מטופל-Report a visit.
- ניתן להעלות דרך מסך זה תצלום רנטגן אותו המערכת תנתח באופן מיידי ותציג עבור המשתמש את התוצאות בהתאם.
- ניתן להגיע למסך צפייה של ניתוח סטטיסטי אודות כל תצלומי הרנטגן שהמשתמש הזין למערכת.
- ישנה אפשרות לקרוא אודות האפליקציה ולהגיע לספר הפרויקט באמצעות לחיצה על כפתור about DeTech.

	patient name	ID	email	age
1	Ulysses Wright	593827416	ulysses.wright9697@hotmail.com	46
2	Victoria Bailey	265197483	victoria.bailey9899@gmail.com	28
3	William Garcia	439518276	william.garcia0001@yahoo.com	39
4	Xander Collins	718263495	xander.collins0103@hotmail.com	29
5	Isla Watson	385972641	isla.watson7071@hotmail.com	41
6	Kaitlyn Ramirez	921375486	kaitlyn.ramirez7475@yahoo.com	48
7	Liam Cooper	748216593	liam.cooper7677@hotmail.com	24
8	Hannah Lee	427638915	hannah.lee2021@yahoo.com	43
9	Isaac Martinez	582731964	isaac.martinez2223@hotmail.com	31
10	Jacob Davis	654893172	jacob.davis2425@gmail.com	37
11	Katherine Hernandez	793216584	katherine.hernandez2627@yahoo.com	28
12	Lucy Brown	218574936	lucy.brown2829@hotmail.com	42
13	Michael Adams	376429815	michael.adams3031@gmail.com	50
14	Nicole Taylor	961247538	nicole.taylor3233@yahoo.com	31

מסך דיווח על ביקור חדש - Report a visit dialog

מסך לצורך דיווח על ביקור חדש של מטופל. קלט: שם מלא של המטופל, מספר תעודת זהות, אימייל, מין, תאריך לידה, תיאור הביקור וניתוב לתמונת רנטגן (אופציונלי). במקרה של שגיאה, בעקבות הוולידציה, תוצג הודעה מתאימה למשתמש. במקרה של הצלחה, הפרטים של הביקור של המטופל שהוזן יישלחו לשרת האפליקציה והוא ישמור אותם ב-database. לאחר מכן הוא יוחזר למסך הראשי (האזור האישי של המשתמש). במקרה שהוזן שם קובץ של תמונת רנטגן המערכת תשלח אותו אל שרת ה-web שם התמונה תנותח. השרת יחזיר תשובה האם קיימת בתמונה שהוזנה דלקת ריאות או לא. המערכת תציג עבור המשתמש מסך בו יוצגו לו תוצאות הניתוח של התמונה שהזין.

Dialog ? X

Report a visit

Full name:

ID:

E-mail:

Gender:

Male

Birth date:

10-05-2023

Description of the case:

Visit date:

10-05-2023

X-ray photo (Optional): add

Submit

בדיקת תקינות האימייל אצל
הלקוח.

בדיקת תקינות הפרטים אצל
הלקוח.

Dialog ? X

Report a visit

Full name:
maayan moshayov

ID:
123456789

E-mail:
mya@t

Gender:
Male

Birth date:
02-05-2000

Description of the case:
Headaches and fever

Visit date:
10-05-2023

X-ray photo (Optional): [add](#)
D:/Projects/finale/pneumonia_3.jpeg

please enter a valid email!

[Submit](#)

Dialog ? X

Report a visit

Full name:

ID:

E-mail:

Gender:
Male

Birth date:
10-05-2023

Description of the case:

Visit date:
10-05-2023

X-ray photo (Optional): [add](#)

please fill all the fields!

[Submit](#)

בדיקת תקינות תעודת
הזהות אצל הלקוח.

Dialog ? X

Report a visit

Full name:
maayan moshayov

ID:
1234

E-mail:
mya@gmail.com

Gender:
Male

Birth date:
02-05-2000

Description of the case:
Headaches and fever

Visit date:
10-05-2023

X-ray photo (Optional): add
D:/Projects/final/pneumonia_3.jpeg

please enter a valid ID!

Submit






מסך צפייה בניתוח סטטיסטי של תצלומי הרנטגן- Statistical analysis

מסך לצורך צפייה בנתונים סטטיסטיים אודות כל תצלומי הרנטגן שהמשתמש העלה למערכת אי פעם. המערכת מציגה למשתמש בטבלה את כל תצלומי הרנטגן שהוא העלה לצד תוצאת הניתוח של התצלום על ידי המערכת. את התצלומים ניתן למיין לפי סוג התוצאה ולפי רמת הדיוק של המערכת. כמו כן, יוצג סיכום עבור המשתמש שיכלול את מספר תצלומי הרנטגן שהוא העלה, את מספר התצלומים שזוהו כמכילים דלקת ריאות ואת רמת הדיוק הממוצעת של המערכת.

Dialog ? X

Statistical analysis:

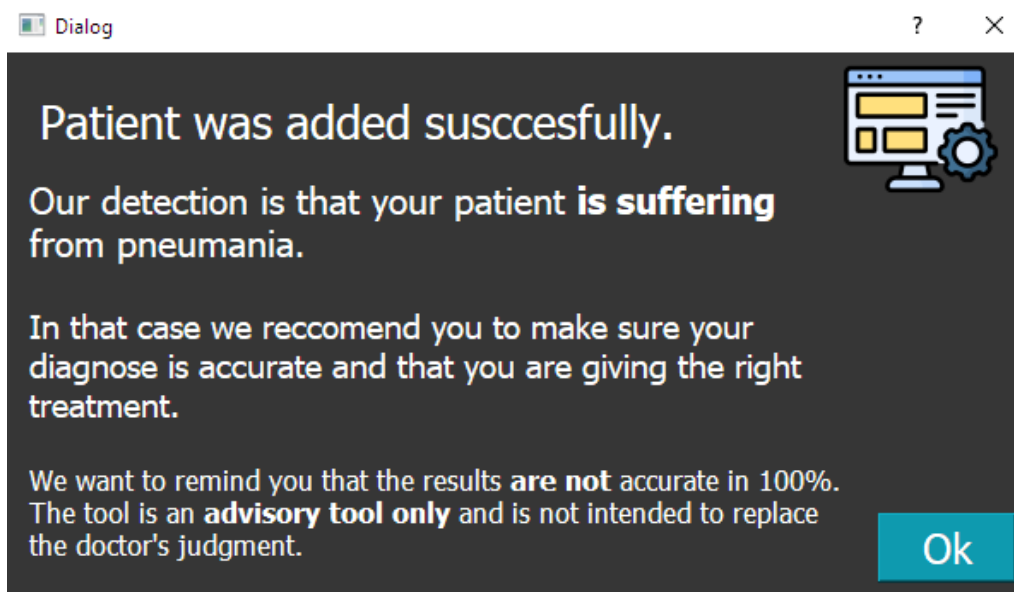
Below are presented some statistical data regarding the x-rays you uploaded to the system.
 X-rays uploaded: 17
 X-rays that were detected ad pneumonia: 12
 Overall accuracy: 88.5%

	X-ray	Detection	Accuracy
1		pneumonia	97.1%
2		pneumonia	99.7%
3		pneumonia	99.9%
4		normal	98.96%
			

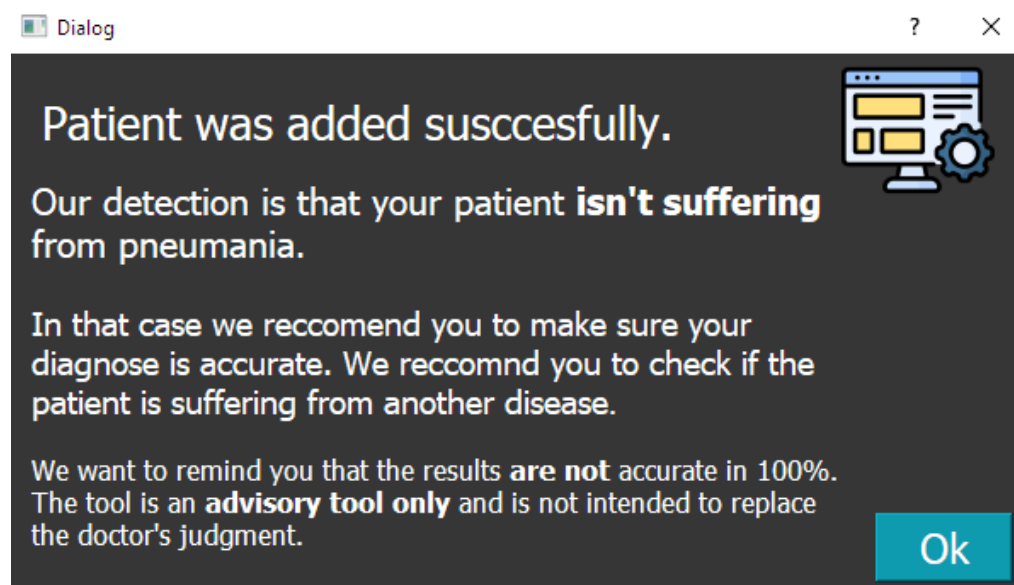
מסך צפייה בתוצאות הניתוח של המערכת עבור תצלום רנטגן- Results dialog

מסך לצורך צפייה בתוצאות הניתוח של המערכת עבור תצלום רנטגן. מסך זה כולל את תוצאות הניתוח ואת המלצת המערכת כיצד כדאי לרופא לפעול. מסך זה יוצג למשתמש לאחר כל פעם שהוא יזין תצלום רנטגן למערכת לצורך ניתוחה.

עבור תצלום שמכיל דלקת
ריאות



עבור תצלום שאינו מכיל
דלקת ריאות



פרוטוקול תקשורת

- התקשורת בין הלקוח לשרת ה AI מתבצעת בפרוטוקול HTTP.
 - התקשורת בין הלקוח לשרת האפליקציה מתבצעת על פי פרוטוקול TCP/IP.
- כל הודעה בין הלקוח לשרת האפליקציה בנויה על פי תבנית שאני הגדרתי באופן הבא -

תבנית פקודה (בקשה) בין לקוח לשרת

[Parameters] \n\n [Command]

בשורה הראשונה מופיעה הפקודה (אחת מתוך מאגר הפקודות).
בשאר השורות מופיעים הפרמטרים של ההודעה לפי מספר הפרמטרים הדרושים בפקודה, כל פרמטר בשורה. למשל, שם מלא, אימייל, סיסמא

תבנית פקודה (תשובה) בין שרת ללקוח

Command + "_ans" \n\n [True/False] \n\n [Parameters]

בשורה הראשונה מופיעה הפקודה (אותה אחת אשר קיבל מהלקוח + המחרוזת "_ans" מלשון "answer").
בשורה השנייה מופיע פרמטר בוליאני של True/False. השרת ישלח True כאשר הפעולה שהלקוח ביקש לבצע בוצעה בהצלחה ו-False אם אינה בוצעה בהצלחה.
בשאר השורות מופיעים הפרמטרים המוחזרים ללקוח לפי מספר הפרמטרים הדרושים בפקודה, כל פרמטר בשורה. למשל, רשימת הלקוחות.

רשימת הפקודות- Commands

- LOGIN
- SIGN_UP
- ADD_PATIENT
- GET_SPECIFIC_PATIENT
- GET_PATIENT_LIST

LOGIN

השרת יבדוק אם שם המשתמש והסיסמה רשומים במערכת, אם כן ישלח True ואת מספר הזהות הייחודי של הרופא אחרת יחזיר False.

- הודעת לקוח (בקשה)

LOGIN \r\n username \r\n password

- הודעת שרת (תשובה)

LOGIN_ans \r\n True\FALSE \r\n doctor_id\None

SIGN_UP

השרת יבדוק קיים משתמש בעל שם משתמש זהה לזה שהלקוח שלח. אם כן, השרת יחזיר False. אם לא השרת יחזיר True, ישמור את פרטי המשתמש בבסיס הנתונים וישלח את מספר הזהות הייחודי של הרופא.

- הודעת לקוח (בקשה)

SIGN_UP \r\n full_name \r\n email \r\n username \r\n password

- הודעת שרת (תשובה)

SIGN_UP_ans \r\n True\FALSE \r\n doctor_id\None

ADD_PATIENT

השרת יוסיף את המטופל אל בסיס הנתונים. אם המטופל כבר קיים השרת ימחק אותו ויוסיף אותו מחדש עם פרטיו החדשים. השרת יחזיר True אם הפעולה הצליחה ו-False אם לא.

- הודעת לקוח (בקשה)

ADD_PATIENT \r\n full_name \r\n ID \r\n email \r\n gender \r\n birth_date \r\n case_description \r\n visit_date \r\n doctor_id \r\n chance\None

- הודעת שרת (תשובה)

ADD_PATIENT_ans \r\n True\FALSE

GET_SPECIFIC_PATIENT

השרת ימצא את המטופל המבוקש בבסיס הנתונים לפי תעודת הזהות שלו. השרת יחזיר True ואת פרטי המטופל מופרדים ב-"#" אם המטופל נמצא והפעולה הצליחה. השרת יחזיר False אחרת.

- הודעת לקוח (בקשה)

GET_SPECIFIC_PATIENT \r\n patient_id

- הודעת שרת (תשובה)

GET_SPECIFIC_PATIENT_ans \r\n True\False \r\n patient\None

GET_PATIENT_LIST

השרת ימצא את כל המטופלים של רופא מסויים בבסיס הנתונים לפי מספר הזהות הייחודי של הרופא. השרת יחזיר True ואת פרטי כל המטופלים אם הפעולה הצליחה (כל מטופל נמצא בשורה נפרדת ופרטיו מופרדים ב- "#"). השרת יחזיר False אחרת.

- הודעת לקוח (בקשה)

GET_PATIENT_LIST \r\n doctor_id

- הודעת שרת (תשובה)

GET_PATIENT_LIST_ans \r\n True\False \r\n patient_list\None

טכנולוגיות בהן נעשה שימוש בפרויקט

בינה מלאכותית לזיהוי תמונה - Artificial Intelligence (הסבר תיאורטי של האלגוריתם)

בפרויקט שלי עשיתי שימוש במודל בינה מלאכותית. לצורך הבנה על כיצד עובד המודל אסביר תחילה כמה מושגים בסיסיים ולאחר מכן אסביר כיצד מתרחש הקסם.

CNN - Convolutional Neural Network. הינה רשת נוירונים הבנויה בשכבות כל יחידה מקושרת לאחרת. הקשר הוא בעצם תוצאת חישוב של יחידה (נוירון). לקשרים שונים יש משקלים (תוצאות שונות) המבנה הזה משמש ומתאים בעיקר לעיבוד תמונה וראייה ממוחשבת.

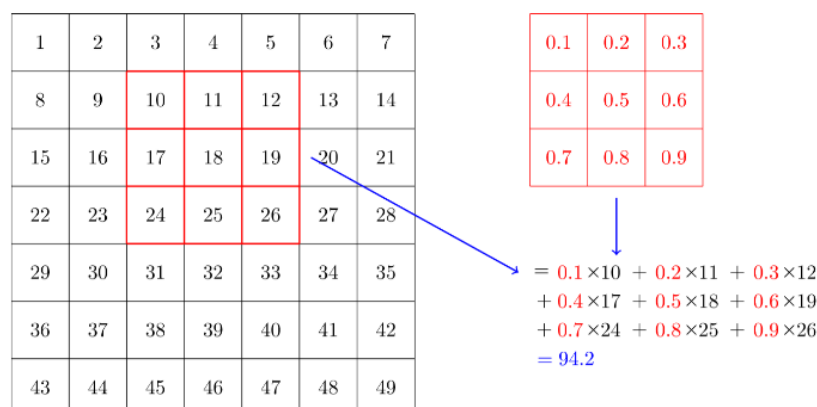
Convolution היא פעולה מתמטית אשר מחשבת תוצאה עבור שתי תופעות שהיא קולטת לחישוב, התוצאה הסופית היא סכום של סדרת התוצאות של חישוב בודד בין כל 2 יחידות

באמצעות אלגוריתם קונבולוציה ומבנה הרשת ה-CNN בונים מודל נתונים שבאמצעותו ניתן לחשב תוצאה ע"פ קלט שונה ומשתנה

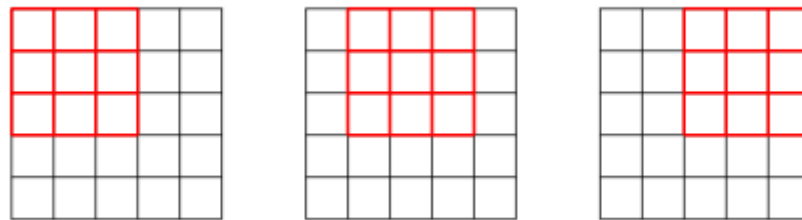
אימון training את המודל הבנוי ניתן לאמן ע"י מתן קלט של אובייקט מזוהה שאת תוצאת החישוב שלו אנו רוצים שהמודל יחזה

הסבר התהליך לפי שלבים:

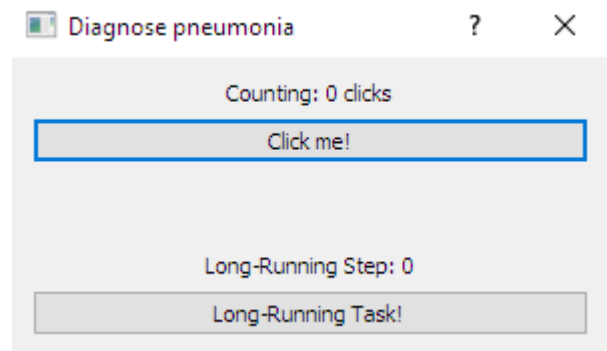
1. **המטרה:** לקלוט תמונה ולזהות האם קיים בתוכה אובייקט או תבנית שאותה אנו מחפשים במקרה שלנו אנו מספקים תמונת רנטגן של צילום חזה ורוצים לדעת אם התמונה משקפת דלקת ריאות
2. **בניית המודל** הקלט הוא תמונה ובעצם היא רצף של ערכים בינאריים (של כל פיקסל) שערכו הוא מ-0 עד 255, במילים תמונת שחור לבן.
3. **הגדרת טנסור**, טנסור הוא מטריצת ערכים רב מימדית המתארת גודל פיזיקלי שניתן להחיל בחישוב על גודל אחר ולקבל ערך מומר במקרה שלנו אנו מגדירים טנסור שהוא מטריצה בגודל 3×3 עם ערכים/משקלים:



- עבור המטריצה המסומנת (מימין באדום) נקבע הטנסור שמכיל את המשקלים 0.1, 0.2 וכו'.
4. חישוב: אנחנו סורקים את התמונה ממקמים את הטנסור כל פעם באיזור שעדיין לא נסרק עבור כל מיקום מתבצע חישוב מכפלה וסיכום של ערכי הטנסור עם הערכים האמיתיים שמופיעים בתמונה (ערך כל פיקסל). בסיום החישוב, מתקבל ערך מספרי אחד שמייצג חלק בתמונה בגודל 3×3 מתבצע כפל בין הערך במטריצה לבין המיקום שלו על הטנסור. מחברים את כל התוצאות ומגיעים לתוצאת ערך הטנסור.
 את המשקלים בטנסור כמובן ניתן לשנות, משקל מסויים יכול להיות יעיל עבור זיהוי של אובייקט מסויים אך משקל אחר יהיה יותר יעיל עבור זיהוי של אובייקט אחר.



- 5.** לאחר סריקה ראשונה של התמונה מתקבלים ערכים של מכפלה וסיכום טנסור עם כל חלקי התמונה נוצר מצב שבו ישנם פחות ערכים מאשר בתחילת התהליך. זאת בגלל שערכיהם של 9 פיקסלים נהפכו לערך אחד. לכן למעשה, אנו מצמצמים את התמונה תוך שמירה על ערכיה. חוזרים על התהליך שוב ושוב עד אשר נותר עבורנו ערך אחד, זהו הערך של התמונה לפי המשקלים שבחרנו.
**** תהליך החישוב הזה הוא בעצם הקונבולוציה**
6. בסיום התהליך התמונה כולה מיוצגת ע"י ערך מספרי אחד
לצורך המחשה: אם ניתן קלט זהה (אותה תמונה) לחישוב, יתקבל אותו מספר בתוצאה. לחילופין אם ניתן תמונה דומה יתקבל ערך קרוב. וכך לגבי תמונות זהות רבות יתקבל תחום ערכים שמייצג את התמונות האלו
7. Model training בתהליך האימון אנו מספקים לאלגוריתם החישוב תמונות רבות של צילומי דלקת ריאות. לכל תמונה יתקבל ערך מספרי ולכל התמונות יתקבל תחום ערכים שבתוכו הם מצויים. בתהליך האימון עלינו לתת למערכת כמה שיותר תצלומים של דלקת ריאות **וגם** תצלומים **שאינם** מכילים דלקת ריאות. זאת, על מנת שלמודל יהיה מספר גדול ככל הניתן של ערכים ולכן יהיה מדויק יותר.
8. בתהליך הבחינה **prediction** של המודל, אנו מביאים עבורו תצלום אותו הוא לא פגש בעבר, הוא מפענח את התצלום ומתקבל ערך. את הערך שהתקבל הוא משווה עם ערכיהם של תצלומים אחרים שאת כותרתם הוא כבר יודע. אם הערך שהתקבל נמצא בטווח הערכים של צילומי רנטגן של דלקת ריאות הוא יכול להסיק שמדובר בתצלום של דלקת ריאות. במידה והערך שהתקבל תואם דווקא את טווח הערכים של תצלומים שאינם מכילים דלקת ריאות הוא יסיק שמדובר בתצלום שאינו מכיל דלקת ריאות.



יישום הפתרון של הבינה המלאכותית בפרויקט זה

הלמידה

את הלימוד בנושא הבינה המלאכותית התחלתי לראשונה באתר [kaggle](https://www.kaggle.com). האתר מציע סביבת עבודה notebook עבור פיתוח והרצת קוד online, שיתוף פרויקטים ובניית מודלים חכמים באמצעות kaggle - :

1. מצאתי מאגר צילומים שעליו מבוסס מודול הבינה המלאכותית. המאגר מכיל כ-5800 צילומי חזה שנעשו באוכלוסיית ילדים בגילאי 0 עד 5 בעיר גואנגג'ואו שבסין (ראה [Kaggle dataset here](#)).
2. איתרתי פרויקטים וקטעי קוד שונים, התנסיתי בהרצתם ולמדתי כיצד בונים מודל ראה לדוגמא: [project 1](#), [project 2](#) המודל המתקבל מפרוייקטים אלו הוא ראשוני ועדיין לא מדויק בחיזוי המבוקש יש לבצע תהליך אימון עליו אפרט בהמשך

סביבת kaggle טובה מאוד ללימוד והתנסות אך פחות מתאימה לפיתוח רציף, זמני ההרצה איטיים ומוגבלים (למספר שעות), העבודה מסורבלת ומסובכת. הבנתי שעליי לחפש אלטרנטיבה ..

הפיתוח

בהתבסס על ההבנה והצרכים, חיפשתי מדריכים וקוד שיתופי פתוח שאוכל לבסס באמצעותו יישום עצמאי ומקומי על גבי המחשב האישי שלי ולא על גבי שרתי online. מצאתי הרבה חומר. פרויקט זה כולל קטעי קוד שמייצר ומאמן מודל (ראה נספח) הקוד מבוסס על ספריות פיתוח AI של Keras ו Tensorflow נעזרתי הרבה במדריכים של ספריות אלו כדי להבין את הדרך בה מגדירים מודל ואימון ראה: [keras](#), [tensorflow](#)

הגישה אל המודל

לאחר שלב זה שבו התקבל מודל מאומן, יש צורך לאפשר גישה אליו כדי לשאול אותו לגבי חיזוי של תמונת רנטגן. לשם כך זה עשיתי שימוש בכלי פיתוח מאוד יעיל לסביבת פייתון (ראה: [FastAPI](#)) באמצעותו הגדרתי API המאפשר גישה RESTful אל המודל באמצעות בקשת HTTP POST, נעזרתי במדריך טוב זה, קטע הקוד מצוי בנספחים.

דוגמא לבקשה

```
http://host ip:port/pneumonia/predict
```

דוגמא לתשובה שמתקבלת בפורמט json

```
{"predicted_class": "pneumonia", "pneumonia_probability":  
  {" "[0.97447765
```

את ה API, מארח שרת אינטרנט [uvicorn](#) (הוא מימוש שרת אינטרנט עבור Python).

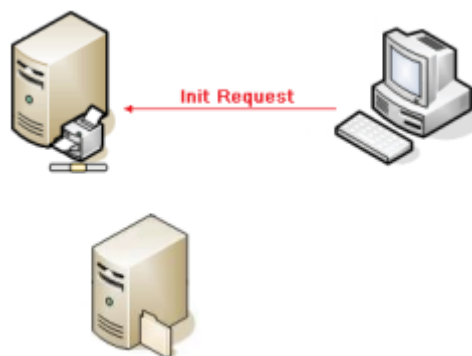
אבטחת מידע

בפרויקט זה הושם דגש גדול על אבטחת המידע. כידוע בעולם הנוכחי ישנם שיטות רבות ומתוחכמות לפריצה, האזנה ולקחת מידע ממערכות ובסיסי נתונים. ובכך מידע רגיש חשוף לסכנות אלו. בפרויקט שלי ישנה התייחסות ויכולת להתמודד עם מידע רגיש במיוחד בשל העובדה שהכלי שפיתחתי מכיל מידע רפואי רגיש אשר דורש אבטחה. את אבטחת המידע בפרויקט יישמתי בשני אופנים ע"י הצפנות ו- hashing.

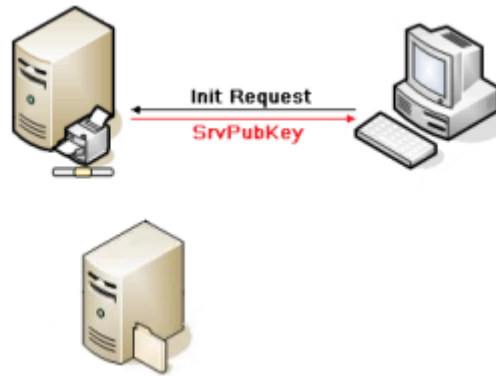
הצפנות

התקשורת בין השרת והלקוחות מוצפנת כולה. הצפנתי את התקשורת באמצעות SSL. כלומר, Transport Layer Security Protocol. הפרוטוקול מהווה שכבה נוספת מתחת לפרוטוקול הקיים. ההצפנה משתמשת באמצעות העברת מפתחות ובאמצעות גורם חיצוני (צד שלישי אמין) בכדי לאמת את נתוני ההצפנה לפני השימוש הראשוני בהם. מנגנון לחיצת היד (handshake) בין שני הצדדים מתרחש באופן הבא

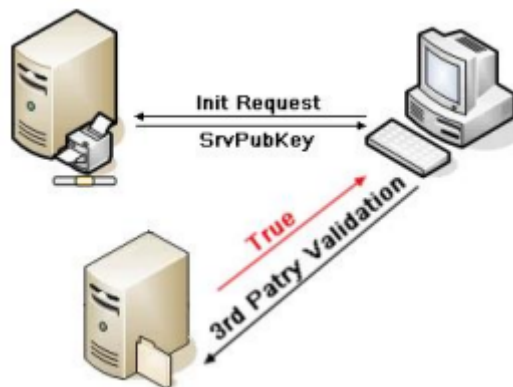
- **שלב ראשון** - הלקוח שולח בקשה ליצירת קשר עם השרת.



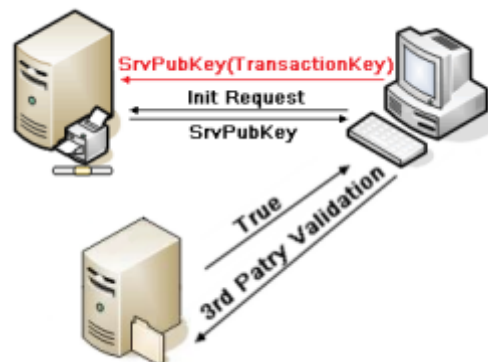
- **שלב שני** - השרת שולח Certificate ייחודי לו, אשר מכיל בין היתר את המפתח הציבורי שלו. את הסרטיפיקט ייצרתי בעצמי (לצורכי פיתוח, אף גוף שלישי מאומת רציני לא חתום עליו).



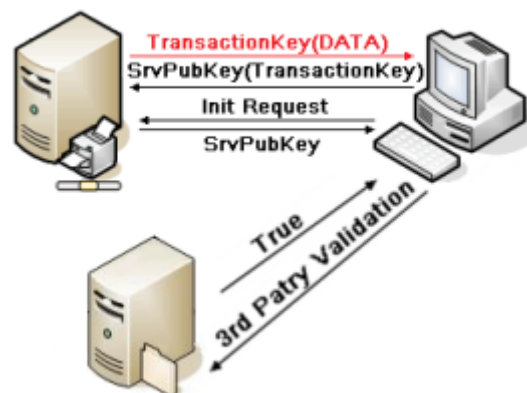
- **שלב שלישי** - הלקוח יחליץ את הסיסמא מתוך הסרטיפיקט (בתוך הסרטיפיקט יש את הסיסמא של ה-RSA).
- **שלב רביעי** - הלקוח מקבל תשובה חיובית (במידה ואכן ה-Certificate אמיתי) מהגורם השלישי.



- **שלב חמישי** - הלקוח מצפין מספר רנדומלי (Key Transaction) בעזרת המפתח הציבורי שאותו הוא שלף מתוך ה-Certificate המאושר ושולח אותו לשרת.



- **שלב שישי - השרת מפענח את ההודעה שהגיע מהלקוח בעזרת המפתח הפרטי שברשותו, שולף מתוכה את המספר הרנדומאלי שאותו יצר הלקוח, ומצפין בעזרתו את המידע הרגיש שאותו הלקוח ביקש.**



- המפתח הרנדומאלי הזה מהווה את מפתח ההצפנה של השיחה, ומכאן- שכבת האבטחה תצפין בעזרתו כל חבילת מידע שתעבור דרכה ותדאג לפיענוחה (בעזרת אותו מספר) כאשר החבילה הגיעה ליעדה.

Hashing

על כל הסיסמאות שנשמרו בבסיס הנתונים הופעלה פונקציית hash. פונקצייה אשר הופכת מחרוזת רגילה למחרוזת מסובכת אשר לא ניתן להבין ממנה את המחרוזת המקורית. בפרויקט השתמשתי בפונקציית hash מסוג md5. להלן אפרט את היתרונות של השימוש ב-hashing:

- המחרוזת החדשה שנוצרה הינה ייחודית, אין עוד מחרוזת שתביא ליצירת מחרוזת זהה לזו של מחרוזת אחרת.
- גם אם בסיס הנתונים ייפרץ והסיסמאות ייגנבו, לא יהיה ניתן לעשות בהם שימוש, שכן לא ניתן להבין דבר מהסיסמה השמורה.

דוגמה לשימוש בפונקציה

```
hashlib.md5('b'.encode()).hexdigest()
```

בעבור הפעלת הפונקצייה על האות 'b' תיווצר המחרוזת הבאה

```
92eb5ffee6ae2fec3ad71c777531578f
```

בסיס הנתונים

הסבר כללי על בסיס הנתונים

למימוש מאגר הנתונים בחרתי להשתמש במנוע בסיס נתונים מסוג SQLite, המאגר הכולל טבלאות שבהן שלכל אחת מהן מוגדרות עמודות מסוימות. כל טבלה משמשת לשמירת נתונים בנושאים שונים. בכל טבלה ישנן שדות ועמודות המכילות ערכים מסוגים שונים (למשל, string, int, float). ניתן לבצע מגוון פעולות על בסיס נתונים זה:

- הכנסה של מידע חדש.
- עדכון מידע קיים.
- קריאה של מידע.

לביצוע פעולות עם בבסיס הנתונים ניתן לתקשר באמצעות c, python, ו-SQL statements. בפרויקט שלי אני השתמשתי ב SQL Statements. אלו הן הצהרות בשפת SQL שבאמצעותן ניתן לבצע את מגוון הפעולות שבבסיס הנתונים מציע. למשל:

```
cur.execute("SELECT rowid FROM user_details_db WHERE username = ?", (username,))
```

מבנה בסיס הנתונים בפרויקט

בבסיס הנתונים בפרויקט שלי ישנן שתי טבלאות. טבלאות אלו שומרות מידע בנוגע למשתמשי האפליקציה ובנוגע למטופליהם.

doctors_details_db

בטבלה זו שמורים כל הנתונים אודות משתמשי האפליקציה (הרופאים). טבלה זו מאפשרת את אחסון פרטי המשתמשים והגנה על פרטיהם האישיים. עמודות הטבלה:

- שם מלא- full name
- אימייל- email
- סיסמא- password. סיסמא זו נשמרת בבסיס הנתונים כ-hash (פעולה שמופעלת על מחרוזת ומתקבלת מחרוזת ייחודית שלא ניתן לפענח אותה). באמצעות השימוש ב-hash הסיסמה מוגנת גם אם בסיס הנתונים ייפרץ.
- מספר זהות ייחודי- doctor id. משמש כאות זיהוי לרופא. כל מטופל משויך לרופא מסוים באמצעות מספר זה.

patients_details_db

בטבלה זו נשמרים כל הנתונים אודות המטופלים. עמודות הטבלה:

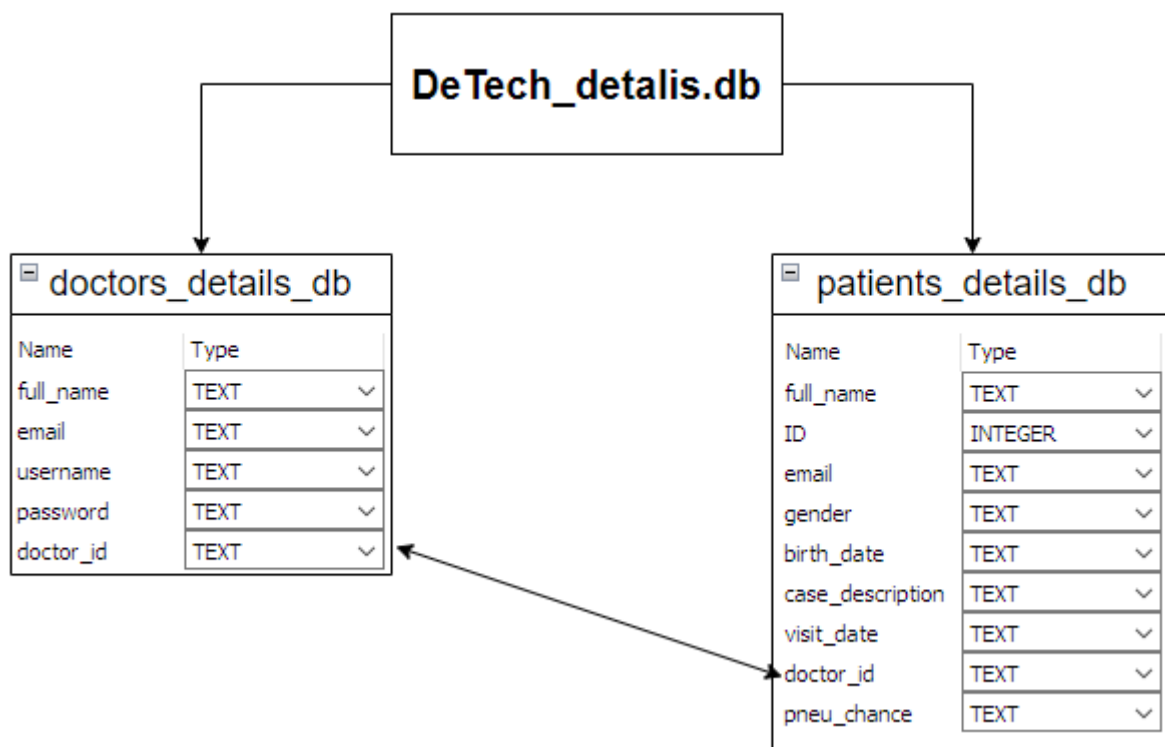
- שם מלא- full name
- תעודת זהות- ID
- אימייל- email
- מין - Gender
- תאריך לידה- birth date
- תיאור הביקור- case description. מהווה תיאור של ביקור המטופל אצל הרופא.
- תאריך ביקור- visit date. תאריך הביקור של המטופל אצל הרופא.
- מספר זהות של הרופא- doctor id. כאשר רופא מזין מטופל חדש למערכת המערכת שומרת את מספר הזהות הייחודי של הרופא גם אצל המטופל ובכך משייכת אותו אליו.

- תוצאת תצלום הרנטגן- pneu chance. אם הוזן תצלום רנטגן בעבור הביקור של המטופל תוצאותיו ישמרו בבסיס הנתונים לצורך ניתוח סטטיסטי.

קישור בין טבלאות הנתונים

קישור בין טבלאות הנתונים נעשה באמצעות השדה doctor id. שדה זה מהווה כ-PRIMARY KEY בטבלת doctors_details_db ומהווה כ-FOREIGN KEY בטבלת patients_details_db. כל מטופל בעל doctor id זהה לזה של רופא מסוים למעשה משויך אליו. השדה הינו ייחודי לרופא, משמע אין שני רופאים בעלי doctor id זהים.

תרשים המאגר

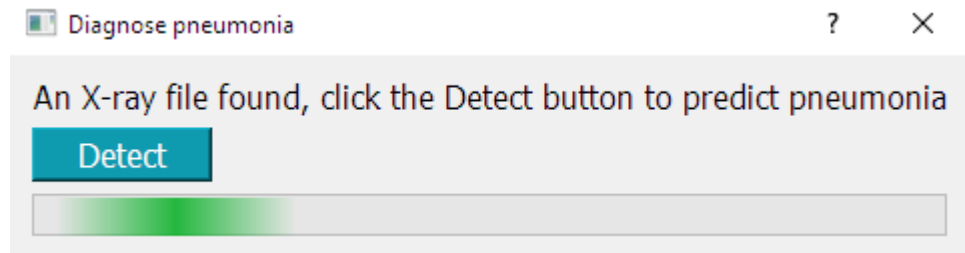


תכנות מתקדם (שימוש בפרדיגמות מתקדמות)

תכנות בסביבה מרובת תהליכים multi tasking programming

בפרוייקט זה עשיתי שימוש בהרצת תהליכים ברקע עבור אופרציות שיש להם זמן ריצה ארוך ואני רוצה שהאפליקציה תמשיך להגיב ולא תקפא.
לדוגמא: כאשר נשלחת בקשה לזיהוי תמונה היא מבוצעת ב task נפרד, בזמן ההמתנה מוצג חלון עם progress bar עד לסיום הפעולה.

ראה תמונת מסך -



תכנות בתצורת Signal and Slots

שימוש ביכולת זו, מקנה אפשרות גמישה לפונקציה מסוימת להודיע על אירוע 'signal' במערכת ובמקום אחר ישנו מאזין שקולט את האירוע ומגיב.
לדוגמא: במחלקה `Worker` שכתבתי, יש הודעה על סיום אירוע (בו חוזרת תשובה משרת ה AI) יש צורך לעבד אותה ולהציגה למשתמש, קיים ומוגדר מאזין שקולט את הסיגנל ומגיב בהפעלת פונקציה אחרת שמציגה את התשובה ושומרת את הנתונים שהגיעו במאגר הנתונים.

שימוש בקונפיגורציה חיצונית

בפרוייקט זה עשיתי שימוש בקריאת קובץ קונפיגורציה חיצוני בפורמט INI באמצעותו אני מגדיר פרמטרים גלובליים שניתנים לגישה בכל מקום בקוד, כמו כן אין צורך בשינוי קוד כאשר משתנה ערך של פרמטר.

לדוגמא הפרמטרים host, port מוגדרים בקובץ ואני יכול לשנותם בקלות
 ראה מקטע מקובץ `app_config.ini`

```
[server]
port: 8080
host: 192.168.2.23
```

סקירת חולשות סיכונים ואיומים

גורם סיכון/רכיב במערכת	חומרת הסיכון	רמת הסתברות	תיאור	פתרון והתמודדות
לוח זמנים	2	2	חריגה מלוח הזמנים, הספק מועט.	חלוקת הפרויקט לספרינטים. לכל ספרינט מטרת מוגדרת. בדרך זו ניתן להתמודד עם קשיים בלוח הזמנים
מודול הבינה המלאכותית	4	3	מודל בינה מלאכותית לא עובד. חיזוי דלקת הריאות שגוי או חיזוי באחוזי דיוק נמוכים	אימון המודל. הרצת סדרות של אימון על המודל על מנת להגדיל את אחוזי הדיוק שלו ואת כושר החיזוי שלו.
למידת חומר חדש	5	4	אי הבנה של החומר הדרוש. חוסר הצלחה בניסיון ללמוד את החומר ולממש אותו. סיכון זה יכול להיות גורם לכישלון הפרויקט.	הקדשת משאבים וזמן רב לצורך הלמידה. עיקר העבודה בתחילת הפרויקט הייתה מושקעת לצורך למידה. צפייה בסרטונים, קריאת מאמרים ופודקאסטים באינטרנט סייעו ללמידת החומר הדרוש.
עבודה עם בסיס נתונים	2	5	שמירה ואחזור של מידע לבסיס הנתונים עובד בצורה לא תקינה. לא ניתן לעשות שימוש רציף בבסיס הנתונים.	למדתי את השפה עמה מתקשרים עם בסיס הנתונים (SQL). גם אם בסיס הנתונים לא היה עובד בכלל, היה ניתן לעשות שימוש בקובץ טקסט פשוט לצורך שמירת הנתונים. דרך זו הרבה פחות יעילה ונכונה אך אפשרית במקרה קיצון.
תקשורת	4	4	תקשורת לא תקינה. אין חיבור עובד בין הלקוח לבין שרת האפליקציה או לשרת ה-web.	למדתי כיצד ניתן להרים חיבור על גבי פרוטוקול TCP/IP באמצעות סוקטים. בנוסף, עשיתי שימוש בפקודות POST בפרוטוקול HTTP.
אבטחת מידע	2	3	מצב בו התקשורת בין הלקוח לשרתים אינה מוצפנת. בנוסף, פרטי המטופלים והמשתמשים אינם מוצפנים בבסיס הנתונים	למדתי את האופן בו מצפינים את התקשורת. בפרויקט זה יישמתי אבטחה באמצעות ssl ומפתחות AES. כמו כן השתמשתי בפונקציית hash.
פיתוח ממשק משתמש	3	4	קושי ביישום ופיתוח של ממשק חלונאי. לא ניתן לעבור כמו שצריך בין מסכי האפליקציה והפעולות השונות.	למדתי לעבוד עם הספרייה PyQt5. למדתי לעצב חלונות, לקבוע פעולות לאחר לחיצת כפתור ולשנות תצוגה. בנוסף, למדתי לעבוד עם הכלי qt designer שהפך את תהליך עיצוב החלון לפשוט יותר. ניתן לתת חלופה של ממשק לא חלונאי. למשל, command line.

חומרת סיכון: 1 חסרת משמעות, 2 מצומצמת, 3 חמורה, 4 חמורה מאוד, 5 אסונית
רמת הסתברות: 1 כמעט ודאי, 2 ייתכן, 3 אפשרי, 4 לא סביר, 5 נדיר

מימוש הפרויקט

סקירת מודולים בפרויקט-טכנולוגיות

- **Socket** - לצורך תקשורת מבוססת סוקטים ב-TCP
- **Ssl** - לצורך הצפנת התקשורת בפרויקט
- **Threading** - על מנת לאפשר ריבוי לקוחות
- **PyQt5** - לצורך בניית ממשק המשתמש ה-GUI
- **Sys** - לצורך הרצת ממשק המשתמש כאפליקציה
- **Re** - השתמשתי בפונקציה של המודל לצורך ואלידציה של אימייל
- **Os** - לצורך עבודה עם מערכת הקבצים של המחשב
- **PIL** - לצורך עיבוד תמונה
- **Uuid** - לצורך יצירת מחרוזת מיוחדת
- **Hashlib** - לצורך אבטחת הסיסמא הפעלתי פונקציית hash עליה
- **lo** - לפתיחת תמונה והפיכתה לביטים
- **Fastapi** - לצורך הנגשת הפנייה אל שרת ה-web
- **Tensorflow** - ספריית התשתית לפיתוח יכולות AI.
- **Keras** - שכבת API של python על גבי tensorflow. באמצעותה מייצרים את המודל, מבצעים את החישוב והחיזוי ומאמנים את המודל.
- **Webbrowser** - לצורך פתיחת כתובת URL בזמן ריצה.
- **Date** - לצורך עבודה עם תאריכים (תאריכי לידה ותאריכי ביקור של מטופל).
- **Json** - שימש כפורמט של קבלת מידע משרת ה-HTTP.
- **Requests** - לצורך שליחת בקשות HTTP ב-python.
- **Sqlite3** - מספק ממשק עם בסיס הנתונים sqlite.

סקירת מחלקות, פעולות ומסכים בפרויקט

myClient

תפקיד המחלקה

לכל לקוח שמתחבר אל הסרבר נוצר מופע משלו אשר יש לו תכונות ייחודיות שימשו אותו בעת התקשורת.

תכונות המחלקה

- Client_socket - הסוקט שדרכו מתבצעת התקשורת עם השרת
- The_config_parser - מכיל את קובץ הקונפיגורציות של הריצה

פעולות המחלקה

```
# A function that starts the run and connects to the server with socket
def run(self):
```

```
# A function that starts the run of the GUI
def run_app(self):
```

```
# A function that receives username and password, organizes it into a login msg
and sends it to the server.
# The function returns if the login was successful
def login(self, username, password):
```

```
# A function that receives a name, email, username, and password,
organizes it into a sign up msg and sends it to the server.
# The function returns if the sign up was successful
def sign_up(self, full_name, email, username, password):
```

```
# A function that receives a full_name, patient_id, email, gender,
birth_date, case_description, visit_date, x_ray_detection, doctor_id,
organizes it into a add_patient msg and sending it to the server.
# The function returns the details of the requested patient as a list
def add_patient(self, full_name, patient_id, email, gender, birth_date,
case_description, visit_date, x_ray_detection, doctor_id):
```

```
# A function that receives a patient_id, organizes it into a
get_specific_patient msg and sending it to the server.
# The function returns the details of the requested patient as a list
def get_specific_patient(self, patient_id):
```

```
# A function that receives a doctor_id, organizes it into a
get_patients_list msg and sends it to the server.
# The function returns all the patients of the requested doctor as lists
def get_patients_list(self, doctor_id):
```

```
# A function that receives a msg and command. The function checks if the
message matches the protocol structure
# If it does, and the server returns True, the function returns the data
of the msg
def handle_res(self, res, command):
```

```
# A function that receives an email and checks if it is valid
def check_email(self, email):
```

Ui_MainWindow

תפקיד המחלקה

מהווה את החלון הראשי בממשק המשתמש. מעבר להצגה הויזואלית ישנם פעולות ותכונות חשובות במחלקה זו שתורמים לזרימת הפרויקט.

תכונות המחלקה

- Doctor_id - מהווה את מספר הזהות המיוחד של המשתמש. כאשר עוד לא התחבר משתמש לאפליקציה התכונה תהיה שווה ל-None אך כאשר יתחבר משתמש, תשתנה תצוגת המסך ותתחלף להיות לאזורו האישי של המשתמש וכמו כן תשתנה התכונה doctor_id למספר הזהות הייחודי של המשתמש שהתחבר.

פעולות עיקריות במחלקה

```
#A function that receives MainWindow, client, config_parser. The function
builds the dialog with its details
def setupUi(self, MainWindow, client, config_parser):
```

```
#The function opens the project book from a URL.
def open_project_book(self):
```

```
#A function that always updates the row in the table that is selected
def selected_row(self, selected):
```

```
# A function that receives a client. The function opens the
report_a_visit_Dialog and changes the display into view mode.
def watch_patient_details(self, client):
```

```
#A function that receives a client. The function opens the login dialog
def open_login_Dialog(self, client):
```

```
#A function that receives a client. The function opens the sign up dialog
def open_sign_up_Dialog(self, client):
```

```
#A function that receives the username and its unique id. The function
changes the view of the main window. It turns it into a screen that shows
the personal area of the logged in user
def turn_into_logged_in_mode(self, username, doctor_id):
```

```
#A function that receives a client. The function gets the patient list from
the server and fills the table of patients
def fill_table(self, client):
```

```
#A function that clears the table widget. The row number becomes 0 and the
contents are empty.
def clear_table(self):
```

```
#A function that receives birth date and calculates the age of the person.
Returns the age.
def calculate_age(self, birth_date):
```

```
#A function that asks the user to choose an x-ray photo from the
computer(using another function) and then the function sends it to the web
server. In the end (if everything went well) the function opens the results
Dialog with the detection of the x-ray (using another function).
def select_photo_and_send_curl(self):
```

```
# A function that asks the user to choose an x-ray photo from the computer.
The function allows only 'jpeg' files. The function returns the file
location.
def get_file_name(self):
```

```
#A function that receives tuple_response. The function returns if the
detection is pneumonia and returns the exact chance.
def get_pneumonia_probability(self, tuple_response):
```

```
# A function that receives res_tuple_pneumonia, client. The function is
called after the send curl function is finished. The function opens the
results dialog.
def after_send_curl(self, res_tuple_pneumonia, client):
```

```
# A function that receives a client and detection of an x-ray. The function
opens the results dialog in accordance with the detection.
def open_results_Dialog(self, is_pneu):
```

```
#A function that receives a client. The function opens the report_a_visit
dialog
def open_report_a_visit_Dialog(self, client):
```

```
#A function that changes the display to the design of the home page. The
user exits the personal area
def sign_out(self):
```

```
# A function that receives a client. The function opens the
statistical_analysis Dialog.
def open_statistical_analysis_Dialog(self, client):
```

Ui_Login_Dialog

תפקיד המחלקה

מהווה חלון התחברות אל האפליקציה

פעולות עיקריות במחלקה

```
# A function that receives a Dialog, client, main_win. The function builds
the dialog with its details
def setupUi(self, Dialog, client, main_win):
```

```
# A function that receives Dialog, client, main_win. The function gets the
fields from the lines, does a validation, and send the details of the user
to the server to check if the details are correct. The function would raise
error labels when necessary
def try_login(self, Dialog, client, main_win):
```

Ui_Sign_up_Dialog

תפקיד המחלקה

מהווה חלון ההרשמה אל האפליקציה

פעולות עיקריות במחלקה

```
# A function that receives a Dialog, client, main_win. The function builds
the dialog with its details
def setupUi(self, Dialog, client, main_win):
```

```
# A function that receives Dialog, client, main_win. The function gets the
fields from the lines, does a validation, and send the details of the user
to the server to check if the username already exists. If not the server
would add the user to the database The function would raise error labels
when necessary
def try_sign_up(self, Dialog, client, main_win):
```

```
#The function receives a password and checks if it's strong enough. Returns
True of False
def strong_password_checker(self, password):
```

Ui_Report_a_visit_Dialog

תפקיד המחלקה

לחלון זה שני תפקידים.

- מהווה כחלון דיווח על ביקור חדש של מטופל
- מהווה כחלון צפייה ועריכה של מטופל קיים.

תכונות המחלקה

- Can_add_photo - מכיל ערך בוליאני, האם ניתן להוסיף תמונה למטופל.
- View_mode - מכיל ערך בוליאני, האם החלון נמצא במצב צפייה.

פעולות עיקריות במחלקה

```
# A function that receives a Dialog, client, main_win, doctor_id. The
function builds the dialog with its details
def setupUi(self, Dialog, client, main_win, doctor_id):
```

```
# A function that receives Dialog, client, main_win, doctor_id. The
function gets the fields from the lines, does a validation, and send the
details of the patient to the server. If an x-ray was uploaded the function
would send it to the web server. The function will close the Dialog at the
end of the function.
def report(self, Dialog, client, main_win, doctor_id):
```

```
#A function that receives a photo path and a patient_IDpatient_id. The
function resizes the image and saves it in the "x_ray_stats" folder.
def resize_and_save_photo(self, photo_path, patient_id):
```

```
#A function that receives a photo path, and sends an HTTP request to the
web server. The function returns the detection of the photo by the AI
model.
def send_curl(self, photo_path):
```

```
#A function that receives client, ID, doctor_id. The
def check_ID(self, client, patient_id, doctor_id):
```

```
#A function that opens the results dialog
def open_results_Dialog(self):
```

```
#A function that receives the main_window. The function call to another
function that ask the user to choose x-ray photo from the computer. The
function updates the QLineEdit of the photo path with the path the user
selected.
def select_photo(self, main_win):
```

```
#A function that receives client, patient_id. The function changes the
display into view mode. It puts the details of the selected patient in the
appropriate fields.
def turn_into_show_patient_mode(self, client, patient_id):
```

```
#A function that receives a date(string) splitted in a list. The function
returns the date as a QDate type.
def from_string_to_date(self, splitted_date):
```

Ui_Statistical_analysis_Dialog

תפקיד המחלקה

מהווה כמסך המציג נתונים סטטיסטיים אודות כל תצלומי הרנטגן שהמשתמש העלה למערכת אי פעם.

פעולות עיקריות במחלקה

```
# A function that receives a Dialog. The function builds the dialog with
its details
def setupUi(self, Dialog):
```

```
#A function that receives directory_path, client, dialog, doctor_id. The
function fills a table with x-ray images and it's results.
def fill_stats_table(self, directory_path, client, dialog, doctor_id):
```

```
#A function that counts the number of x-rays that were detected as
pneumonia
def count_pneumonia_among_table(self):
```

```
# A function that calculates the average accuracy of detection.
def check_accuracy(self):
```

Ui_Results_Dialog

תפקיד המחלקה

מהווה כמסך המציג את תוצאות החיזוי של מודל הבינה המלאכותית.

פעולות עיקריות במחלקה

```
# A function that receives a Dialog. The function builds the dialog with
its details
def setupUi(self, Dialog):
```

```
#A function that receives a detection of an x-ray and a reason to call
the results dialog. The function changes the display of the dialog
according to these parameters.
def turn(self, is_pneu, call_reason):
```

```
#A function that receives a Dialog and close it.
def close_Dialog(self, Dialog):
```

Worker

תפקיד המחלקה

משמש ככלי להרצת פונקצייה ברקע מבלי לפגוע בהרצה הנוכחית

פעולות עיקריות במחלקה

```
#A function that runs the caller function. It informs when the function
is finished.
def run(self):
```

Worker_Dialog

תפקיד המחלקה

מהווה כמסך טעינה. בזמן שהפונקצייה רצה באופן מקבילי על ה-worker במסך מוצג progress bar.

פעולות עיקריות במחלקה

```
# A function that receives Dialog, client, func_for_background_worker,
parameters_tuple, func_for_callback. The function builds the dialog with
its details
def setupUi(self, Dialog, client, func_for_background_worker,
parameters_tuple, func_for_callback):
```

```
#A function that starts a new thread, runs the worker, shows the progress
and the calls after_worker_finished when the function is finished
def run_long_task(self):
```



```
#A function that is called after the run_long_task is finished. The
function is calling to the callback function.
def after_worker_finished(self):
```

AppConfig

תפקיד המחלקה

משמש לצורך גישה לקובץ הקונפיגורציות

תכונות המחלקה

• the_config_parser - מכיל את הניתוב לקובץ הקונפיגורציות.

פעולות עיקריות במחלקה

```
#A function that receives config_file_path. The function prints details
about the config files.
def read_config(self, config_file_path):
```

```
# A function that receives category, name. The function returns the value
from the config file in accordance to the parameters. (Strings only)
def get_string_value(self, category, name):
```

```
# A function that receives category, name. The function returns the value
from the config file in accordance to the parameters. (ints only)
def get_int_value(self, category, name):
```

```
# A function that receives category, name. The function returns the value
from the config file in accordance to the parameters. (booleans only)
def get_bool_value(self, category, name):
```

```
# A function that returns if the current run is debug values in
accordance to the config file.
def is_using_debug_values(self):
```

```
# A function that returns the username value from the config file.
def get_user_name(self):
```

```
# A function that returns the password value from the config file.
def get_password(self):
```

AI_model

תפקיד המחלקה

משמש לצורך גישה לקובץ הקונפיגורציות

פעולות עיקריות במחלקה

```
#A function that loads the x-ray files.  
def load_data(self):
```

```
#A function that defines the AI model. The function returns the model.  
def define_model(self):
```

```
#A function that trains the model.  
def train_model(self):
```

```
#A function that calls the train_model function and save the new trained  
model to the directory.  
def deploy_model(self):
```

בדיקת המערכת

ניווט בין מסכים

מטרת הבדיקה: בדיקת הניווט בין המסכים. האם עובד בהתאם לתרשים הזרימה
ביצוע הבדיקה: ניסיון לעבור בין מסכים בניגוד לתרשים הזרימה. למשל, לקפוץ מהמסך הראשי למסך דיווח על ביקור חדש.
תוצאות הבדיקה: לא ניתן לעבור בין המסכים בניגוד למתואר בתרשים הזרימה.

פעולות בפרויקט

מטרת הבדיקה: בדיקת הפעולות בפרויקט. האם כל פעולה עובדת בהתאם למצופה ממנה.
ביצוע הבדיקה: ניסיון "להפיל" את הפעולה. למשל, לחיצה על כפתור ה-Login לפני מילוי כל השדות, מילוי שדות שגויים.
תוצאות הבדיקה: מצאתי בחלק מהפעולות בעיות בעבור מקרי קצה. למשל, הזנת תעודת זהות אשר מכילה תווים שאינם מספרים.
פתרון הבעיות: טיפלתי בכל מקרה קצה בעייתי בצורה אינדיבידואלית. למשל, בדיקה האם כל התווים שהוזנו הינם מסוג integer. לאחר ביצוע הפתרון, בדקתי את הפעולה שתוקנה שוב לבדוק אם ישנם מקרי קצה נוספים.

פעולות בבסיס הנתונים

מטרת הבדיקה: בדיקת שמירה ואחזור של נתונים מבסיס הנתונים.
ביצוע הבדיקה: ניסיון להזין מידע אל בסיס הנתונים, קריאה שלו לאחר מכן וניסיון לעדכן אותו.
תוצאות הבדיקה: שמירה, אחזור ועדכון של מידע בבסיס הנתונים בהצלחה.

תקשורת

מטרת הבדיקה: בדיקת התקשורת בין הלקוח לשרת האפליקציה ובין הלקוח לשרת ה-web.
ביצוע הבדיקה: ניסיון לפתיחת חיבור באמצעות סוקט ולאחר מכן ניסיון לשלוח הודעות. ניסיון לשליחת בקשות HTTP לשרת ה-web
תוצאות הבדיקה: לא נתקלתי בבעיות בתקשורת עם שרת ה-web. נוצרו בעיות בתקשורת מול שרת האפליקציה כאשר התחברו מספר לקוחות בו זמנית אל השרת.
פתרון הבעיות: שימוש בת'רדים. כל לקוח שמתחבר אל השרת רץ על Thread נפרד. בכך השרת מבצע ריצה מקבילית ומטפל בכל הלקוחות.

חיזוי המודל

מטרת הבדיקה: בדיקת החיזוי של המודל. האם המודל מסוגל לזהות דלקת ריאות באחוזים גבוהים.
ביצוע הבדיקה: מתן תצלומי רנטגן למודל ובדיקת תוצאות המודל.
תוצאות הבדיקה: נתקלתי בזיהויים שגויים או בזיהויים בעלי אחוזי דיוק נמוכים.
פתרון הבעיות: הרצה נוספת של אימון המודל. האימון גורם להגדלת אחוזי הדיוק של המודל ולהפחתת החיזויים השגויים.

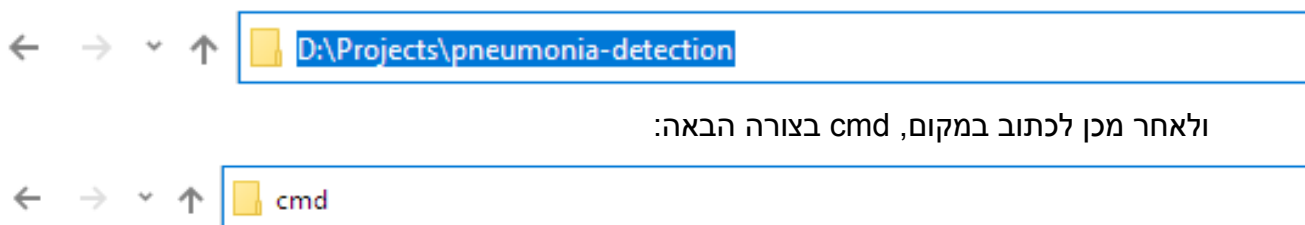
מדריך למשתמש:

הוראות התקנה

אפליקצית הלקוח תומכת במספר מערכות הפעלה multi platform support האפליקציה תוכל לרוץ במחשב Windows או Mac.

השלבים בהתקנת האפליקציה:

1. יש לוודא או להתקין סביבת Python 3.11 [מהאתר הזה](#)
2. להוריד את הקוד של הפרויקט שלי מ- GITHUB ולשמור אותו בספרייה נפרדת במחשב <https://github.com/maayanmg/DeTech.git>
3. בספרייה שבו נשמר הקוד יש לפתוח Command line.
ניתן לעשות זאת באמצעות הקשה על נתיב הספרייה:



ולאחר מכן לכתוב במקום, cmd בצורה הבאה:

ולהקיש Enter.

4. לאחר מכן יש להריץ את שתי הפקודות הבאות, לצורך יצירה של סביבה וירטואלית והרצתה:
python -m venv venv
venv\Scripts\activate

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

D:\Projects\final_project>python -m venv venv

D:\Projects\final_project>venv\Scripts\activate

(venv) D:\Projects\final_project>
```

5. לאחר מכן יש להוריד את קובץ ה-requirements. לשם כך נשתמש בפקודה הבאה ב-cmd:
pip install -r requirements_venv.txt --trusted-host files.pythonhosted.org
6. לאחר הורדת כל ההתקנות הדרושות ניתן להתחיל בהרצה.

תחילה נריץ את שרת ה-web אשר עליו יושב מודל הבינה המלאכותית. נעשה זאת באמצעות הפקודה הבאה ב-cmd:

```
uvicorn --host 0.0.0.0 --port 5000 classifier.app:app
```

7. כעת ניתן להריץ את שרת האפליקציה ואת הלקוח. ניתן לעשות זאת בשני דרכים:
(i) לפתוח סביבת עבודה שמסוגלת להריץ קבצי python. למשל, Visual, PyCharm, Studio Code.
(ii) להריץ את הקבצים באמצעות ה-cmd. ניתן לעשות זאת באמצעות הרצת הפקודה:
שם הקובץ + "python"
8. כעת, לאחר שהן שרת האפליקציה, שרת ה-web והלקוח רצים ניתן לומר כי האפליקציה מוכנה לשימוש!! כמובן שניתן להתחבר כמה משתמשים בו זמנית ממקומות שונים.

עץ קבצים

```

Folder PATH listing
Volume serial number is BA33-1B14
D:.
|   .dockerignore
|   .envrc.example
|   .gitignore
|   Dockerfile
|   README.md
|   README_EnvSetting.txt
|   requirements.txt
|   requirements_venv.txt
|   requirements_venv.txt.orig
|   tree.txt
+---app_code
|   AI2.png
|   ai_human.png
|   app_config.ini
|   books.png
|   books3.png
|   cacert.pem
|   config.py
|   date_picker.py
|   db.py
|   enc_client.py
|   enc_server.py
|   localhost.pem
|   login_Dialog.py
|   logo1.png
|   main_logo.png
|   main_window.py
|   pic_2.png
|   progress_bar.py
|   progress_bar.ui
|   report_a_visit_Dialog.py
|   results_Dialog.py
|   results_logo.png
|   send_curl.py
|   sign_up_Dialog.py
|   statistical_analysis_Dialog.py
|   user_details.db
|   xray1.png
|   x_ray.png
\---classifier
|   app.py
|   sendCurl.py
|   train.py
+---models
|   weights.h5
\---routers
    pneumonia_router.py

```

רפלקציה

תחושת מהעבודה על הפרויקט

העבודה על הפרויקט הייתה מאוד חווייתית עבורי. נהנתי מתהליך הלמידה, נהנתי לחקור חומרים חדשים ולהרחיב את אופקיי בתחום התוכנה ובטכנולוגיות חדשות שאותן למדתי במהלך הדרך. חשתי תחושת סיפוק גדולה מהתוצר המוגמר שהוא שילוב טכנולוגי בין החומר הנלמד בנושא אבטחת מידע והטמעתו לתוך אפליקציה רגישת נתונים. נהניתי לשפר ולקדם את התוצר שייחלתי לו. כמו כן, אני שמח מאוד שהצלחתי ליצור פרויקט שנותן מענה על צורך שקיים היום ואולי בעתיד יהיה ניתן לעשות בו שימוש.

מה קיבלתי וכלים שאקח איתי להמשך

במהלך העבודה על הפרויקט, נחשפתי לטכנולוגיות חדשניות ומרתקות ולכלים שימושיים ומעניינים. טעמתי **מעולם הבינה המלאכותית**. למדתי איך בונים מודל של בינה מלאכותית, איך מאמנים אותו, איך בוחנים אותו ואיך בכלל הוא עובד. כמו כן, רכשתי כלים בתחום **ממשק המשתמש (GUI)**. למדתי כיצד ניתן ליצור ממשק משתמש ברמה גבוהה, לעצב מסכים, להזין נתונים לטבלאות תוך הוספת תמונות ועוד. את כלים אלו רכשתי כאשר השתמשתי בספרייה `pyqt5` וחקרתי אודותיה. אין לי ספק שכלים אלה עשויים לשמש אותי בעתיד במיוחד בעולם הנוכחי בו ישנו גובר השימוש בטכנולוגיה זו. זאת ועוד, למדתי רבות במהלך הפרויקט אודות נושא **אבטחת המידע**. למדתי כיצד מצפינים תקשורת (בפרויקט זה למדתי בעיקר אודות הצפנת SSL) וכיצד שומרים מידע באופן מאובטח (למשל, שימוש ב-hash).

אני מרגיש שתהליך הלמידה העצמאית בפרויקט חיזקה אצלי את תחושת המסוגלות. למדתי חומרים רבים, חדשים ומאתגרים בהם עשיתי שימוש בפרויקט שלי ואני מרגיש שעשיתי זאת בהצלחה. נוסף על כך, אני מרגיש שתהליך העבודה על הפרויקט שיפר אצלי את יכולות התכנון והיצירתיות. מתוקף הצורך לתכנן את הפרויקט מאפס, לחשוב על מרכיביו השונים, לתכנן לו "עבודה ולתעד הכל לספר פרויקט. אני שמח שהתנסיתי ביכולות אלו ואני בטוח שיכולות אלו עוד עשויות לשמש אותי בעתיד.

קשיים ואתגרים שעמדו בפני

בפרויקט זה ניצבתי בפני מספר אתגרים כפי שתיארתי בפרק [המבוא](#) אזכיר אותם כאן שוב בנקודות

אתגר ההיתכנות האם הרעיון יכול לקרום עור וגידים.

למידה של טכנולוגיה חדשה האם אוכל ללמוד וליישם מערכת עובדת בטכנולוגיה שאינה מוכרת לי.

היכרות עם **כלי פיתוח** חדשים לעיצוב ממשק ממשתמש.

ארגון והקמה של **סביבת ריצה** הכוללת את כל המודולים הנדרשים לריצה.

עמידה בלוח זמנים במסגרת זמן מוגבלת לביצוע (תוך תקופת בחינות בגרות אינטנסיבית).

תכנון כולל לפרויקט חלוקה למקטעים וארגון זמנים ברצף של שנת לימודים שלמה. הקושי במתן עדיפויות למשימות והקצאת זמן למשימה אינו קל

פיתוח יחידני עמידה בכל המשימות באופן עצמאי היא מאוד מאתגרת

מסקנות מהפרויקט

היקף הפרויקט כפי שנשקף בעיני (כבר בהתחלה בשלב התגבשות הרעיון) הוא גדול מבחינת תחומי טכנולוגיות וארכיטקטורת רבת רכיבים.

מסקנות העיקריות -

- לא להירתע ממעמסה של מורכבות בכדי להשיג מטרה.
- למידה של חומר חדש היא הכרחית כמעט באופן וודאי אם רוצים להשיג מטרה טכנולוגית
- אם שלב ההיתכנות צולח, המטרה היא ברת השגה ויש לחתור למימושה.
- מוטיבציה היא אלמנט הכרחי להשגת מטרה, אתה חייב להיות מחובר ונלהב מהרעיון בכדי ליישמו.
- אני משוכנע שפרויקט מסוג זה ימצא שימוש גובר בעידן של בינה מלאכותית.
- היכולת והצורך בזיהוי דלקת ריאות שנחשבת לשכיחה באוכלוסייה מגביר את הפיתוח של אפליקציות מסוג זה.

מה הייתי עושה אחרת אם הייתי מתחיל את הפרויקט היום

באופן כללי, אני מרוצה מהאופן שבו עבדתי להשגת המטרות. לאורך הפרויקט הגעתי למספר תובנות שאני יכול היום להגדירם כהמלצות עבור פרויקט חדש -

לימוד חומר חדש: דורש זמן רב ראשית באיתור וחיפוש של מאמרים ומדריכים, הלמידה עצמה וכמובן היישום. בכל שלב יתכן קושי שיהווה מכשול בהתקדמות יש להיערך לזה ולהקצות זמן ולאפיין חלופות. חוותי את האתגר הזה ובראייה לאחור ברור לי נדרש זמן רב יותר ממה ששיערת.

שימוש בכלי פיתוח חדשים: בפרויקט זה עשיתי שימוש בכלים חדשים שלא הכרתי לפני כן. השימוש בהם התבצע בשלב היישום ולכן אני ממליץ לארגן סביבת עבודה מוקדם ככל האפשר ולערוך היכרות מקדימה, דבר היכול לחסוך זמן יקר.

משימות לא ידועות: למדתי שתיחום חומר חדש בפרויקט כולל הרבה מאוד משימות ובעיות לפתרון. הן כמובן אינן ידועות בשלב התכנון אבל מחייבות התייחסות ופתרון. למדתי שיש לקחת אלמנט זה בחשבון על מנת לאפשר מסגרת זמן ומרווחי ביטחון.

פיתוח יכולות בדיקה: מובנות בתוך המערכת, כך שניתן יהיה לבצע בדיקה נקודתית של פעולה בודדת. נוכחתי לדעת תוך כדי עבודה שבדיקת מערכת שכל רכיביה מחוברים כבר יחד, הינה משימה קשה יותר. לעיתים דרושות בדיקות באיזור או רכיב מסוים מבלי לעבור דרך נתיב הפעולה המלא.

תכונות שהייתי רוצה להוסיף לפרויקט

הפרויקט השיג את כל מטרותיו על פי התכנון, אני חושב שהייתי מוסיף את התכונות הבאות להמשך פיתוח האפליקציה:

- שמירת התמונות שהמשתמש מעלה למערכת בבסיס הנתונים ולא מקומית אצלו במחשב.
 - כך תתאפשר צפייה מכל מחשב על גבי הרשת כמו כן התמונות ישארו חסויות ומאובטחות.
 - תמיכה בפיצ'ר של two factor authentication לצורך הגברת האבטחה בכניסה אל המערכת, היות ומדובר במידע רפואי רגיש.
 - הוספת יכולת/פיצ'ר שיחזור סיסמא ("שכחתי סיסמא").
 - אימות כתובת אימייל בעת ההרשמה למערכת.
 - הגנה על השירות שנותן שרת האפליקציה מפני מתקפה של בקשות רבות שעלולות ליצור תור המתנה ארוך ותחושת קיפאון למשתמש.
 - הצפנת בסיס הנתונים כולו לצורך אבטחת מידע בצורה חזקה יותר.
- כיום קיימת הגנה על הפרטים האישיים אך אם ייוספו פרטים רגישים נוספים בעתיד יש לתת על כך את הדעת.

ביבליוגרפיה:

Stack Overflow

<https://stackoverflow.com/>

GeekForGeeks

<https://www.geeksforgeeks.org/>

GitHub

<https://github.com/>

W3Schools

<https://www.w3schools.com/>

Draw.io

<https://drawio-app.com/>

Kaggle

<https://www.kaggle.com/>

dataset Chest X-Ray Images (Pneumonia)

<https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia>

ChatGPT

<https://chat.openai.com/>

Python

<https://www.python.org/>

Removebg

<https://www.remove.bg/upload>

Looka

https://looka.com/logo-maker/?gclid=Cj0KCQjwiryjBhD0ARIsAMLvnF-5Ney0rULOXxaxpZ8ypvJyyfL6M1d-HR1EW7iCVM79BDTc6sJg5NUaAqScEALw_wcB

Real Python

<https://realpython.com/qt-designer-python/>

Starryai

<https://starryai.com/app/create/photography?project=>

נספחים

קישור לסרטון הסבר על הפרויקט

[DeTech_vid.mp4](#)

קישור לקוד המלא ב- GitHub

<https://github.com/maayanmg/DeTech.git>

מענה על דרישות החובה

נושא	מימוש בפרויקט
תכנות מונחה עצמים	1. שימוש במחלקות (Ui_MainWindow, Ui_Login, Ui_Sign_up, Ui_Report_case, Ui_statistical_analysis).
תקשורת	1. מימוש תקשורת על גבי פרוטוקול TCP/IP באמצעות סוקטים לצורך תקשורת בין שרת האפליקציה ללקוח. כל הודעה בין הלקוח לשרת האפליקציה בנויה על פי תבנית שאני הגדרתי. 2. שרת האפליקציה תומך בריבוי לקוחות, תקשורת מול מספר לקוחות באופן מקבילי. 3. התקשורת בין הלקוח לשרת ה-AI (שרת web) מתבצעת בפרוטוקול HTTP.
מערכות הפעלה	1. Multitasking programming - תכנות בריצה מקבילית לצורך פתרון בעיות של המתנה ארוכה לביצוע פעולה (שימוש ב-threads). 2. גישה למערכת הקבצים של מערכת ההפעלה. שמירה ואחזור של קבצים (תמונות). 3. תמיכה ב multi platform, אפליקציית הלקוח יכולה לרוץ ב Windows וגם ב MacOS.
אבטחה	בפרויקט זה הושם דגש גדול על אבטחת המידע. בפרויקט שלי יש שימוש במידע רפואי רגיש אשר דורש אבטחה. 1. אבטחת התקשורת בין שרת האפליקציה ללקוח על ידי הצפנת ssl. 2. אבטחת המידע באמצעות hashing. שמירה בטוחה של סיסמאות למאגר הנתונים (database).
ממשק משתמש	1. בפרויקט יש שימוש רציני בממשק המשתמש. ישנם מסכים שונים שמכילים פונקציונליות רבה (main_window, login, sign_up, statistical_analysis). 2. שימוש בכלי ויזואלי לצורך עיצוב ממשק המשתמש (qt designer). הכלי עושה שימוש בספרייה pyqt5.

הקוד המלא

enc_server.py

```
import ssl
import threading as th
import socket
import db
import config

certfile=r"localhost.pem"
cafile = r"cacert.pem"
purpose = ssl.Purpose.CLIENT_AUTH
context = ssl.create_default_context(purpose, cafile=cafile)
context.load_cert_chain(certfile)

#A function that receives a msg of login and returns the details of the
user
def handle_login_msg(msg):
    username = msg[0]
    password = msg[1]
    return username, password

#A function that receives a msg of sign_up and returns the details of the
user
def handle_sign_up_msg(msg):
    full_name = msg[0]
    email = msg[1]
    username = msg[2]
    password = msg[3]
    return full_name, email, username, password

#A function that receives a msg of add_patient and returns the details of
the patient
def handle_add_patient_msg(msg):
    full_name = msg[0]
    patient_id = msg[1]
    email = msg[2]
    gender = msg[3]
    birth_date = msg[4]
    case_description = msg[5]
    visit_date = msg[6]
    doctor_id = msg[7]
    if len(msg) < 9:
        return full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id, None
    else:
        pneu_chance = msg[8]
        return full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id, pneu_chance
```

```

#A function that receives the client_socket and the client_address. The
function handles the client request and send him back a response
def handle_client(client_socket, client_address):
    print(f'[*] Connection from {client_address}')
    while True:
        try:
            data = client_socket.recv(1024)
            data = data.decode()
            print(f'[*] Received message from {client_address}: {data}')
            split_data = data.splitlines()
            bret = 0
            #handle with login requests
            if split_data[0] == 'LOGIN':
                username, password = handle_login_msg(split_data[1:])
                bret, doctor_id = db.find_username_and_password(username,
password)
                if bret:
                    res = str(bret) + "\r\n" + doctor_id
                else:
                    res = str(bret)

            #handle with sign up requests
            if split_data[0] == 'SIGN_UP':
                full_name, email, username, password =
handle_sign_up_msg(split_data[1:])
                bret, doctor_id = db.add_user(full_name, email, username,
password)
                if bret:
                    res = str(bret) + "\r\n" + doctor_id
                else:
                    res = str(bret)

            if split_data[0] == 'ADD_PATIENT':
                full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id, pneu_chance =
handle_add_patient_msg(split_data[1:])
                if pneu_chance is None:
                    bret = db.add_patient(full_name, patient_id, email,
gender, birth_date, case_description, visit_date, doctor_id)
                else:
                    bret = db.add_patient_with_photo(full_name,
patient_id, email, gender, birth_date, case_description, visit_date,
doctor_id, pneu_chance)
                if bret:
                    res = str(bret) + "\r\n" + username
                else:
                    res = str(bret)

            if split_data[0] == 'GET_SPECIFIC_PATIENT':
                bret, patient = db.get_specific_patient(split_data[1])
                res = str(bret) + "\r\n" + patient

```

```

        if split_data[0] == 'GET_PATIENT_LIST':
            bret, patient_list = db.get_patients_list(split_data[1])
            if patient_list is None:
                res = str(bret) + "\r\n" + "None"
            else:
                res = str(bret) + "\r\n" + patient_list

        print(bret)
        # Send encrypted message to client
        response = split_data[0] + "_ans\r\n" + res
        response = response.encode()
        client_socket.sendall(response)
    except Exception as e:
        None
        client_socket.close()

# A function that start the run and connects the server with its clients
using socket
def run():
    theAppConfig = config.AppConfig('app_config.ini')
    host = theAppConfig.get_string_value('server', 'host')
    port = theAppConfig.get_int_value('server', 'port')
    ThreadCount = 0

    ServerSideSocket = socket.socket()

    try:
        ServerSideSocket.bind((host, port))
    except socket.error as e:
        print(str(e))
    ServerSideSocket = context.wrap_socket(ServerSideSocket,
server_side=True)
    print('Socket is listening..')
    ServerSideSocket.listen(1)
    b = th.Thread(target=db.build_DB())
    b.start()
    while True:
        client_socket, client_address = ServerSideSocket.accept()
        print(f'[*] Accepted connection from {client_address}')
        a = th.Thread(target=handle_client, args=(client_socket,
client_address,))
        a.start()
        ThreadCount += 1
        print('Thread Number: ' + str(ThreadCount))
    ServerSideSocket.close()

if __name__ == '__main__':
    run()

```

enc_client.py

```
import hashlib
import socket
import ssl
import re
from PyQt5 import QtCore, QtGui, QtWidgets
import sys
import main_window
import config

context = ssl.create_default_context()
# Set the context to not verify the server's SSL/TLS certificate
context.check_hostname = False
context.verify_mode = ssl.CERT_NONE

class myClient():
    client_socket = None
    the_config_parser = None

    # A function that starts the run and connects to the server with
    socket
    def run(self):
        self.client_socket = socket.socket()
        self.the_config_parser = config.AppConfig('app_config.ini')
        host = self.the_config_parser.get_string_value('server', 'host')
        port = self.the_config_parser.get_int_value('server', 'port')

        # host = '127.0.0.1'
        # port = 8080
        print('Waiting for connection response')
        try:
            self.client_socket.connect((host, port))
            self.client_socket = context.wrap_socket(self.client_socket,
server_hostname=host)
        except socket.error as e:
            print(str(e))

        self.run_app()

    # A function that starts the run of the GUI
    def run_app(self):
        app = QtWidgets.QApplication(sys.argv)
        Window = QtWidgets.QMainWindow()
        ui = main_window.Ui_MainWindow()
        ui.setupUi(Window, self, self.the_config_parser)
        Window.show()
        sys.exit(app.exec_())
```

```

# A function that receives username and password, organizes it into a
login msg and sends it to the server.
# The function returns if the login was successful
def login(self, username, password):
    password = hashlib.md5(password.encode()).hexdigest()
    message = 'LOGIN\r\n' + username + '\r\n' + password

    if self.the_config_parser.is_using_debug_values():
        message = 'LOGIN\r\n' + self.the_config_parser.get_user_name()
+ '\r\n' + self.the_config_parser.get_password()

    self.client_socket.send(message.encode())
    res = self.client_socket.recv(1024).decode()
    return self.handle_res(res, "LOGIN")

# A function that receives a name, email, username, and password,
organizes it into a sign up msg and sends it to the server.
# The function returns if the sign up was successful
def sign_up(self, full_name, email, username, password):
    password = hashlib.md5(password.encode()).hexdigest()
    message = 'SIGN_UP\r\n' + full_name + '\r\n' + email + '\r\n' +
username + '\r\n' + password
    self.client_socket.send(message.encode())
    res = self.client_socket.recv(1024).decode()
    return self.handle_res(res, "SIGN_UP")

# A function that receives a full_name, patient_id, email, gender,
birth_date, case_description, visit_date, x_ray_detection, doctor_id,
organizes it into a add_patient msg and sending it to the server.
# The function returns the details of the requested patient as a list
def add_patient(self, full_name, ID, email, gender, birth_date,
case_description, visit_date, chance, doctor_id):
    message = 'ADD_PATIENT\r\n' + full_name + '\r\n' + ID + '\r\n' +
email + '\r\n' + gender + '\r\n' + birth_date + '\r\n' + case_description
+ '\r\n' + visit_date + '\r\n' + doctor_id
    if chance != '':
        message = message + '\r\n' + chance
    self.client_socket.send(message.encode())
    res = self.client_socket.recv(1024)
    splitted_res = res.decode().splitlines()
    return splitted_res[1]

# A function that receives a patient_id, organizes it into a
get_specific_patient msg and sending it to the server.
# The function returns the details of the requested patient as a list
def get_specific_patient(self, patient_id):
    message = 'GET_SPECIFIC_PATIENT\r\n' + patient_id
    self.client_socket.send(message.encode())
    res = self.client_socket.recv(1024)
    res = res.decode()
    return self.handle_res(res, "GET_SPECIFIC_PATIENT")

```



```

# A function that receives a doctor_id, organizes it into a
get_patients_list msg and sends it to the server.
# The function returns all the patients of the requested doctor as
lists
def get_patients_list(self, doctor_id):
    message = 'GET_PATIENT_LIST\r\n' + doctor_id
    self.client_socket.send(message.encode())
    res = ''
    while True:
        data = self.client_socket.recv(1024)
        data = data.decode()
        res += data
        if len(data) < 1024:
            break
    splitted_res = res.splitlines()
    if len(splitted_res) == 0:
        return None
    if splitted_res[0] == "GET_PATIENT_LIST_ans" and splitted_res[1]
== "True":
        return splitted_res[2:]
    else:
        return None

# A function that receives a msg and command. The function checks if
the message matches the protocol structure
# If it does, and the server returns True, the function returns the
data of the msg
def handle_res(self, res, command):
    splitted_res = res.splitlines()
    if len(splitted_res) == 0:
        return None

    if splitted_res[0] == command + "_ans" and splitted_res[1] ==
"True":
        return splitted_res[2]
    else:
        return None

# A function that receives an email and checks if it is valid
def check_email(self, email):
    regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b'
    # pass the regular expression
    # and the string into the fullmatch() method
    if (re.fullmatch(regex, email)):
        return True
    else:
        return False

if __name__ == '__main__':
    the_client = myClient()
    the_client.run()

```

db.py

```

import sqlite3
from uuid import uuid4

connection = sqlite3.connect("user_details.db", check_same_thread=False)
cur = connection.cursor()

#A function that builds the database if not exists
def build_DB():
    command = """CREATE TABLE IF NOT EXISTS doctors_details_db(full_name
TEXT NOT NULL, email TEXT NOT NULL, username TEXT NOT NULL, password TEXT
NOT NULL, doctor_id TEXT NOT NULL) """
    command_2 = """CREATE TABLE IF NOT EXISTS patients_details_db(full_name
TEXT NOT NULL, ID INTEGER NOT NULL, email TEXT NOT NULL, gender TEXT NOT
NULL, birth_date TEXT NOT NULL, case_description TEXT NOT NULL, visit_date
TEXT NOT NULL, doctor_id TEXT NOT NULL, x_ray_detection TEXT) """
    cur.execute(command)
    connection.commit()
    cur.execute(command_2)
    connection.commit()

#A function that receives a username and password. The function checks if
the details exists in the database.
#The function returns False and None if it doesn't exist \ True and
doctor_id if it does.
def find_username_and_password(username, password):
    cur.execute("SELECT rowid FROM doctors_details_db WHERE username = ? AND
password = ?", (username, password,))
    db_result = cur.fetchall()
    print(type(db_result))
    if len(db_result) == 0:
        return False, None
    else:
        cur.execute("SELECT doctor_id FROM doctors_details_db WHERE username
= ? AND password = ?", (username, password,))
        doctor_id_tuple = cur.fetchall()
        doctor_id_tuple = doctor_id_tuple[0]
        return True, str(doctor_id_tuple[0])

#A function that receives name, email, username, password. The function
adds the user details to the database.
#Returns True if succeeded and False if not.
def add_user(full_name, email, username, password):
    # check if a row is already exist
    cur.execute("SELECT rowid FROM doctors_details_db WHERE username = ?",

```

```

(username,))
    db_result = cur.fetchall()
    if len(db_result) == 0:
        doctor_id = str(uuid4())
        connection.execute(
            "INSERT INTO doctors_details_db (full_name, email, username,
password, doctor_id) VALUES (?, ?, ?, ?, ?)", (full_name, email, username,
password, doctor_id))
        connection.commit()
        return True, doctor_id
    else:
        return False, None

#A function that receives full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id. The function add the details into
a new patient. If the patient is already exists the function deletes the
patient and then add the updated details as a new patient
def add_patient(full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id):
    try:
        # check if the patient is already exists
        cur.execute("SELECT rowid FROM patients_details_db WHERE ID = ?",
(patient_id,))
        db_result = cur.fetchall()
        if len(db_result) != 0:
            connection.execute("DELETE FROM patients_details_db WHERE ID =
'" + patient_id + "'")
            connection.commit()
            connection.execute("INSERT INTO patients_details_db (full_name, ID,
email, gender, birth_date, case_description, visit_date, doctor_id) VALUES
(?, ?, ?, ?, ?, ?, ?, ?)", (full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id))
            connection.commit()
            return True
        except Exception as e:
            print(e)
            return False

#A function that receives full_name, patient_id, email, gender, birth_date,
case_description, visit_date, doctor_id, x_ray_detection. The function add
the details into a new patient. If the patient is already exists the
function deletes the patient and then add the updated details as a new
patient
def add_patient_with_photo(full_name, patient_id, email, gender,
birth_date, case_description, visit_date, doctor_id, x_ray_detection):
    try:
        # check if the patient is already exists
        cur.execute("SELECT rowid FROM patients_details_db WHERE ID = ?",
(patient_id,))
        db_result = cur.fetchall()
        if len(db_result) != 0:
            connection.execute("DELETE FROM patients_details_db WHERE ID =

```

```

" + patient_id + "'")
        connection.commit()
        connection.execute("INSERT INTO patients_details_db (full_name, ID,
email, gender, birth_date, case_description, visit_date, doctor_id,
x_ray_detection) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)", (full_name, patient_id, email,
gender, birth_date, case_description, visit_date, doctor_id,
x_ray_detection))
        connection.commit()
        return True
    except Exception as e:
        print(e)
        return False

#A function that receives patient_id. The function gets specific patient
and returns it as a string (the details are separated with '#').
def get_specific_patient(patient_id):
    cur.execute('SELECT * FROM patients_details_db')
    data = cur.fetchall()
    patient = ''
    for row in data:
        id = str(row[1])
        if id == patient_id:
            if row[-1] == None:
                row = row[:-1]
            patient += '#'.join(str(s) for s in row) + '\r\n'
            break
    if patient == '':
        return False, None
    else:
        return True, patient

#A function that receives doctor_id. The function gets all the patients
that are belong to the doctor. The function returns all the patients as a
list.
def get_patients_list(doctor_id):
    cur.execute('SELECT * FROM patients_details_db')
    data = cur.fetchall()
    patient_list = ''
    for row in data:
        if row[7] == doctor_id:
            patient_list += '#'.join(str(s) for s in row) + '\r\n'
    if patient_list == '':
        return False, None
    else:
        return True, patient_list

```

main_window.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'appDate.ui'
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
import os
import webbrowser
from datetime import date
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QTableWidgetItem, QHeaderView
import statistical_analysis_Dialog
import login_Dialog
import report_a_visit_Dialog
import results_Dialog
import send_curl
import sign_up_Dialog
import progress_bar

class Ui_MainWindow(object):
    doctor_id = None
    selected_row = None
    the_app_config = None
    #A function that receives MainWindow, client, config_parser. The
    function builds the dialog with its details
    def setupUi(self, MainWindow, client, config_parser):
        self.the_app_config = config_parser
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(880, 690)
        MainWindow.setStyleSheet("background-color: rgb(54, 54, 54);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.about_btn = QtWidgets.QPushButton(self.centralwidget)
        self.about_btn.setGeometry(QtCore.QRect(690, 610, 161, 31))
        self.about_btn.setStyleSheet("background-color: rgb(54, 54, 54);\n"
                                     "color: rgb(255,255,255);\n"
                                     "font-size: 14pt\n"
                                     "")
        self.about_btn.setDefault(False)
        self.about_btn.setFlat(False)
        self.about_btn.setObjectName("pushButton_6")
        self.books_img = QtWidgets.QLabel(self.centralwidget)
        self.books_img.setGeometry(QtCore.QRect(690, 60, 141, 91))
        self.books_img.setText("")
```

```

self.books_img.setPixmap(QtGui.QPixmap("books.png"))
self.books_img.setObjectName("label_12")
self.report_a_visit_btn = QtWidgets.QPushButton(self.centralwidget)
self.report_a_visit_btn.hide()
self.report_a_visit_btn.setGeometry(QtCore.QRect(690, 230, 161, 31))
self.report_a_visit_btn.setStyleSheet("background-color: rgb(54, 54,
54);\n"
                                     "color: rgb(255,255,255);\n"
                                     "font-size: 14pt\n"
                                     "")
self.report_a_visit_btn.setDefault(False)
self.report_a_visit_btn.setFlat(False)
self.report_a_visit_btn.setObjectName("pushButton_5")
self.x_ray_detection_btn = QtWidgets.QPushButton(self.centralwidget)
self.x_ray_detection_btn.hide()
self.x_ray_detection_btn.setGeometry(QtCore.QRect(690, 270, 161,
31))
self.x_ray_detection_btn.setStyleSheet("background-color: rgb(54,
54, 54);\n"
                                     "color: rgb(255,255,255);\n"
                                     "font-size: 14pt\n"
                                     "")
self.x_ray_detection_btn.setDefault(False)
self.x_ray_detection_btn.setFlat(False)
self.x_ray_detection_btn.setObjectName("pushButton_7")
self.login_btn = QtWidgets.QPushButton(self.centralwidget)
self.login_btn.setGeometry(QtCore.QRect(690, 140, 161, 31))
self.login_btn.setStyleSheet("background-color: rgb(14, 154,
175);\n"
                              "color: rgb(255,255,255);\n"
                              "font-size: 16pt\n"
                              "")
self.login_btn.setObjectName("pushButton")
self.DeTech_text = QtWidgets.QLabel(self.centralwidget)
self.DeTech_text.setGeometry(QtCore.QRect(230, 10, 211, 111))
self.DeTech_text.setStyleSheet("color: rgb(14, 154, 175);\n"
                              "font-size: 42pt ")
self.DeTech_text.setObjectName("label_2")
self.logo_img = QtWidgets.QLabel(self.centralwidget)
self.logo_img.setGeometry(QtCore.QRect(450, 20, 161, 91))
self.logo_img.setText("")
self.logo_img.setPixmap(QtGui.QPixmap("main_logo.png"))
self.logo_img.setObjectName("label")
self.slogan_text = QtWidgets.QLabel(self.centralwidget)
self.slogan_text.setGeometry(QtCore.QRect(200, 110, 361, 31))
self.slogan_text.setStyleSheet("color: rgb(0, 209, 255);\n"
                              "font-size: 20pt \n"
                              "")
self.slogan_text.setObjectName("label_6")
self.x_ray_img = QtWidgets.QLabel(self.centralwidget)
self.x_ray_img.setGeometry(QtCore.QRect(30, 190, 461, 461))
self.x_ray_img.setText("")

```

```

self.x_ray_img.setPixmap(QtGui.QPixmap("x_ray.png"))
self.x_ray_img.setObjectName("label_4")
self.ai_img = QtWidgets.QLabel(self.centralwidget)
self.ai_img.setGeometry(QtCore.QRect(400, 280, 421, 321))
self.ai_img.setText("")
self.ai_img.setPixmap(QtGui.QPixmap("ai_human.png"))
self.ai_img.setObjectName("label_5")
self.sign_up_btn = QtWidgets.QPushButton(self.centralwidget)
self.sign_up_btn.setGeometry(QtCore.QRect(690, 180, 161, 31))
self.sign_up_btn.setStyleSheet("\n"
                                "color: rgb(255,255,255);\n"
                                "background-color: rgb(140, 140, 140);\n"
                                "font-size: 16pt\n"
                                "")
self.sign_up_btn.setObjectName("pushButton_4")
self.log_out_btn = QtWidgets.QPushButton(self.centralwidget)
self.log_out_btn.setGeometry(QtCore.QRect(690, 190, 161, 31))
self.log_out_btn.setStyleSheet("background-color: rgb(255, 84, 42);\n"
                                "color: rgb(255,255,255);\n"
                                "font-size: 16pt\n")
self.log_out_btn.setObjectName("pushButton_2")
self.log_out_btn.hide()
self.user_text = QtWidgets.QLabel(self.centralwidget)
self.user_text.setGeometry(QtCore.QRect(690, 150, 201, 31))
self.user_text.setStyleSheet("background-color: rgb(54,54,54);\n"
                              "color: rgb(255, 255, 255);\n"
                              "font: 75 16pt \"MS Shell Dlg 2\";")
self.user_text.setObjectName("user_text")
self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)
self.tableWidget.setGeometry(QtCore.QRect(30, 190, 621, 451))
self.tableWidget.setStyleSheet("background-color: rgb(197,197,197);\n"
                                "color: rgb(42, 42, 42);\n"
                                "font-size: 12pt")

self.tableWidget.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.tableWidget.setObjectName("tableWidget")
self.tableWidget.setColumnCount(4)
self.tableWidget.setRowCount(0)
item = QtWidgets.QTableWidgetItem()
font = QtGui.QFont()
font.setPointSize(12)
item.setFont(font)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
item.setForeground(brush)
self.tableWidget.setHorizontalHeaderItem(0, item)
item = QtWidgets.QTableWidgetItem()
font = QtGui.QFont()

```

```

        font.setPointSize(12)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setPointSize(12)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setPointSize(12)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(3, item)
        self.tableWidget.resizeColumnsToContents()
        self.ai_img.raise_()
        self.books_img.raise_()
        self.DeTech_text.raise_()
        self.logo_img.raise_()
        self.slogan_text.raise_()
        self.x_ray_img.raise_()
        self.report_a_visit_btn.raise_()
        self.x_ray_detection_btn.raise_()
        self.sign_up_btn.raise_()
        self.user_text.raise_()
        self.user_text.hide()
        self.about_btn.raise_()
        self.login_btn.raise_()
        self.tableWidget.raise_()
        self.tableWidget.hide()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 832, 21))
        self.menubar.setObjectName("menubar")
        self.menuHelp = QtWidgets.QMenu(self.menubar)
        self.menuHelp.setStyleSheet("background-color: rgb(255, 255, 255);
color:white")
        self.menuHelp.setTearOffEnabled(False)
        self.menuHelp.setToolTipsVisible(False)
        self.menuHelp.setObjectName("menuHelp")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.menubar.addAction(self.menuHelp.menuAction())
        header = self.tableWidget.horizontalHeader()
        header.setSectionResizeMode(0, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(1, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
        self.detection_stats_btn = QtWidgets.QPushButton(self.centralwidget)
        self.detection_stats_btn.hide()
        self.detection_stats_btn.setGeometry(QtCore.QRect(690, 310, 161,
31))

```



```

        self.detection_stats_btn.setStyleSheet("background-color: rgb(54,
54, 54);\n"
                                                "color: rgb(255,255,255);\n"
                                                "font-size: 14pt\n"
                                                "")
        self.detection_stats_btn.setDefault(False)
        self.detection_stats_btn.setFlat(False)
        self.detection_stats_btn.setObjectName("pushButton_8")
        self.ai_img.raise_()
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        self.login_btn.clicked.connect(lambda:
self.open_login_dialog(client))
        self.sign_up_btn.clicked.connect(lambda:
self.open_sign_up_dialog(client))
        self.x_ray_detection_btn.clicked.connect(lambda:
self.select_photo_and_send_curl(client))
        self.report_a_visit_btn.clicked.connect(lambda:
self.open_report_a_visit_dialog(client))

self.tableWidget.selectionModel().selectionChanged.connect(self.selected_row)

        self.tableWidget.doubleClicked.connect(lambda:
self.watch_patient_details(client))
        self.log_out_btn.clicked.connect(lambda: self.sign_out())
        self.about_btn.clicked.connect(lambda: self.open_project_book())
        self.detection_stats_btn.clicked.connect(lambda:
self.open_statistical_analysis_dialog(client))

#The function opens the project book from a URL.
def open_project_book(self):

webbrowser.open("https://docs.google.com/document/d/1xoxYSfxPT1LbkVRRw4hjJANp
fWASCB1bWV0eGYCthco/edit?usp=sharing")

#A function that always updates the row in the table that is selected
def selected_row(self, selected):
    for ix in selected.indexes():
        self.selected_row = ix.row()

# A function that receives a client. The function opens the
report_a_visit_dialog and changes the display into view mode.
def watch_patient_details(self, client):
    patient_id = self.tableWidget.item(self.selected_row, 1).text()
    self.dialog = QtWidgets.QDialog()
    self.ui = report_a_visit_dialog.Ui_Report_a_visit_dialog()
    self.ui.setupUi(self.dialog, client, self, self.doctor_id)
    self.ui.turn_into_show_patient_mode(client, patient_id)
    self.dialog.show()

#A function that receives a client. The function opens the login dialog
def open_login_dialog(self, client):

```

```

        self.dialog = QtWidgets.QDialog()
        self.ui = login_Dialog.Ui_Login_Dialog()
        self.ui.setupUi(self.dialog, client, self)
        self.dialog.show()

    #A function that receives a client. The function opens the sign up
    dialog
    def open_sign_up_Dialog(self, client):
        self.dialog = QtWidgets.QDialog()
        self.ui = sign_up_Dialog.Ui_Sign_up_Dialog()
        self.ui.setupUi(self.dialog, client, self)
        self.dialog.show()

    #A function that receives the username and its unique id. The function
    changes the view of the main window. It turns it into a screen that shows
    the personal area of the logged in user
    def turn_into_logged_in_mode(self, username, doctor_id):
        _translate = QtCore.QCoreApplication.translate
        self.login_btn.hide()
        self.sign_up_btn.hide()
        self.x_ray_img.hide()
        self.ai_img.hide()
        self.user_text.show()
        self.user_text.setText(_translate("MainWindow", "Hello " + username
+ ", "))
        self.x_ray_detection_btn.show()
        self.detection_stats_btn.show()
        self.report_a_visit_btn.show()
        self.doctor_id = doctor_id
        self.tableWidget.show()
        self.log_out_btn.show()

    #A function that receives a client. The function gets the patient list
    from the server and fills the table of patients
    def fill_table(self, client):
        self.clear_table()
        self.tableWidget.setSortingEnabled(False)
        patient_list = client.get_patients_list(self.doctor_id)
        if patient_list is None:
            return
        table = self.tableWidget
        for row in patient_list:
            try:
                row = row.split('#')
                table.insertRow(table.rowCount())
                rowCount = table.rowCount()
                columnCount = table.columnCount()
                for j in range(columnCount - 1):
                    table.setItem(rowCount - 1, j,
QtWidgets.QTableWidgetItem(row[j]))
                age = self.calculate_age(row[4])
                table.setItem(rowCount - 1, 3,

```

```

QtWidgets.QTableWidgetItem(str(age)))
        except:
            pass
        self.tableWidget.setSortingEnabled(True)

    #A function that clears the table widget. The row number becomes 0 and
    the contents are empty.
    def clear_table(self):
        self.tableWidget.clearContents()
        self.tableWidget.setRowCount(0)

    #A function that receives birth date and calculates the age of the
    person. Returns the age.
    def calculate_age(self, birth_date):
        today = date.today()
        spllited_birth_date = birth_date.split('-')
        age = today.year - int(spllited_birth_date[0]) - ((today.month,
        today.day) < (int(spllited_birth_date[1]), int(spllited_birth_date[2])))
        return age

    # A function that asks the user to choose an x-ray photo from the
    computer(using another function) and then the function sends it to the web
    server. In the end (if everything went well) the function opens the results
    Dialog with the detection of the x-ray (using another function).
    def select_photo_and_send_curl(self, client):
        photo_path = self.get_file_name()
        if photo_path == '':
            return
        inParamsTuple = (photo_path,True)
        self.workDlg = QtWidgets.QDialog()
        self.ui = progress_bar.Worker_Dialog(self.workDlg)
        self.ui.setupUi(self.workDlg, client, send_curl.send_curl,
        inParamsTuple, self.after_send_curl)
        self.workDlg.show()

    # A function that asks the user to choose an x-ray photo from the
    computer. The function allows only 'jpeg' files. The function returns the
    file location.
    def get_file_name(self):
        file_filter = 'Data File (*.jpeg)'
        response = QFileDialog.getOpenFileName(
            caption='Select a data file',
            directory=os.getcwd(),
            filter=file_filter,
            initialFilter='jpeg File (*.jpeg)'
        )
        print(response)
        return response[0]

    # A function that receives tuple_response. The function returns if the
    detection is pneumonia and returns the exact chance.
    def get_pneumonia_probability(self, tuple_response):

```

```

        res = ''
        if tuple_response == None:
            return None, res
        if len(tuple_response) == 2 and tuple_response[0] == True:
            res = "Pneumonia in " + str(tuple_response[1])
            return True, res
        elif len(tuple_response) == 2 and tuple_response[0] == False:
            res = "Normal in " + str(tuple_response[1])
            return False, res

    # A function that receives res_tuple_pneumonia, client. The function is
    # called after the send curl function is finished. The function opens the
    # results dialog.
    def after_send_curl(self, res_tuple_pneumonia, client):
        is_pneumonia, x_ray_detection =
self.get_pneumonia_probability(res_tuple_pneumonia)
        self.open_results_Dialog(is_pneumonia)

    # A function that receives a client and detection of an x-ray. The
    # function opens the results dialog in accordance with the detection.
    def open_results_Dialog(self, is_pneumonia):
        self.dialog = QtWidgets.QDialog()
        self.ui = results_Dialog.Ui_Results_Dialog()
        self.ui.setupUi(self.dialog)
        self.ui.turn(is_pneumonia, "x_ray")
        self.dialog.show()

    #A function that receives a client. The function opens the
    #report_a_visit dialog
    def open_report_a_visit_Dialog(self, client):
        self.dialog = QtWidgets.QDialog()
        self.ui = report_a_visit_Dialog.Ui_Report_a_visit_Dialog()
        self.ui.setupUi(self.dialog, client, self, self.doctor_id)
        self.dialog.show()

    # A function that changes the display to the design of the home page.
    # The user exits the personal area
    def sign_out(self):
        self.login_btn.show()
        self.sign_up_btn.show()
        self.x_ray_img.show()
        self.ai_img.show()
        self.user_text.hide()
        self.x_ray_detection_btn.hide()
        self.detection_stats_btn.hide()
        self.report_a_visit_btn.hide()
        self.doctor_id = None
        self.clear_table()
        self.tableWidget.hide()
        self.log_out_btn.hide()

    # A function that receives a client. The function opens the

```

```

statistical_analysis Dialog.
    def open_statistical_analysis_dialog(self, client):
        self.dialog = QtWidgets.QDialog()
        self.ui =
statistical_analysis_dialog.Ui_Statistical_analysis_dialog()
        self.ui.setupUi(self.dialog)
        self.dialog.show()
        self.ui.fill_stats_table("x_ray_stats", client, self.dialog,
self.doctor_id)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("DeTech", "DeTech"))
        self.about_btn.setText(_translate("MainWindow", "about DeTech"))
        self.report_a_visit_btn.setText(_translate("MainWindow", "report a
visit"))
        self.x_ray_detection_btn.setText(_translate("MainWindow", "X-ray
detection"))
        self.login_btn.setText(_translate("MainWindow", "Login"))
        self.DeTech_text.setText(_translate("MainWindow", "DeTech"))
        self.slogan_text.setText(_translate("MainWindow", "Pneumonia
detection using AI"))
        self.sign_up_btn.setText(_translate("MainWindow", "Sign up"))
        self.user_text.setText(_translate("MainWindow", "Hello "))
        self.tableWidget.setSortingEnabled(True)
        item = self.tableWidget.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "patient name"))
        item = self.tableWidget.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "ID"))
        item = self.tableWidget.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "email"))
        item = self.tableWidget.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "age"))
        self.menuHelp.setTitle(_translate("MainWindow", "Help"))
        self.log_out_btn.setText(_translate("MainWindow", "Log out"))
        self.detection_stats_btn.setText(_translate("MainWindow",
"Statistical analysis"))

```

login_dialog.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'login.ui'
#
# Created by: PyQt5 UI code generator 5.15.7

```

```

#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Login_Dialog(object):
    # A function that receives a Dialog, client, main_win. The function
    # builds the dialog with its details
    def setupUi(self, Dialog, client, main_win):
        Dialog.setObjectName("Dialog")
        Dialog.resize(353, 313)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
        Dialog.setModal(True)
        self.sign_in_btn = QtWidgets.QPushButton(Dialog)
        self.sign_in_btn.setGeometry(QtCore.QRect(110, 260, 130, 37))
        self.sign_in_btn.setStyleSheet("background-color: rgb(14, 154,
175);\n"
"color: rgb(255,255,255);\n"
"font-size: 18pt\n"
"")
        self.sign_in_btn.setObjectName("pushButton_3")
        self.username_text = QtWidgets.QLabel(Dialog)
        self.username_text.setGeometry(QtCore.QRect(30, 90, 111, 21))
        self.username_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 16pt")
        self.username_text.setObjectName("label_4")
        self.lineEdit_username = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_username.setGeometry(QtCore.QRect(30, 120, 291, 31))
        self.lineEdit_username.setAutoFillBackground(False)
        self.lineEdit_username.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt\n"
"")
        self.lineEdit_username.setText("")
        self.lineEdit_username.setPlaceholderText("")
        self.lineEdit_username.setObjectName("lineEdit_5")
        self.heading_text = QtWidgets.QLabel(Dialog)
        self.heading_text.setGeometry(QtCore.QRect(30, 10, 221, 71))
        self.heading_text.setStyleSheet("background-color: rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 28pt \"MS Shell Dlg 2\";")
        self.heading_text.setObjectName("label_3")
        self.lineEdit_password = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_password.setGeometry(QtCore.QRect(30, 190, 291, 31))
        self.lineEdit_password.setAutoFillBackground(False)
        self.lineEdit_password.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt")
        self.lineEdit_password.setInputMethodHints(QtCore.Qt.ImhHiddenText |
QtCore.Qt.ImhNoAutoUppercase | QtCore.Qt.ImhNoPredictiveText |

```

```

QtCore.Qt.ImhSensitiveData)
    self.lineEdit_password.setInputMask("")
    self.lineEdit_password.setText("")
    self.lineEdit_password.setEchoMode(QtWidgets.QLineEdit.Password)
    self.lineEdit_password.setPlaceholderText("")
    self.lineEdit_password.setObjectName("lineEdit_6")
    self.password_text = QtWidgets.QLabel(Dialog)
    self.password_text.setGeometry(QtCore.QRect(30, 160, 111, 21))
    self.password_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 16pt")
    self.password_text.setObjectName("label_5")
    self.invaile_details_text = QtWidgets.QLabel(Dialog)
    self.invaile_details_text.setGeometry(QtCore.QRect(30, 230, 221,
21))
    self.invaile_details_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
    self.invaile_details_text.setObjectName("label_6")
    self.fill_fields_text = QtWidgets.QLabel(Dialog)
    self.fill_fields_text.setGeometry(QtCore.QRect(30, 230, 221, 21))
    self.fill_fields_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
    self.fill_fields_text.setObjectName("label_11")

    self.invaile_details_text.hide()
    self.fill_fields_text.hide()
    self.retranslateUi(Dialog)
    QtCore.QMetaObject.connectSlotsByName(Dialog)

    self.sign_in_btn.clicked.connect(lambda: self.try_login(Dialog,
client, main_win))

    # A function that receives Dialog, client, main_win. The function gets
the fields from the lines, does a validation, and send the details of the
user to the server to check if the details are correct. The function would
raise error labels when necessary
    def try_login(self, Dialog, client, main_win):
        username = self.lineEdit_username.text()
        password = self.lineEdit_password.text()

        if main_win.the_app_config.is_using_debug_values():
            username = main_win.the_app_config.get_user_name()
            password = main_win.the_app_config.get_password()

        if username == '' or password == '':
            self.fill_fields_text.show()
            self.fill_fields_text.raise_()
            return
        res = client.login(username, password)

```

```

        if res != None:
            main_win.turn_into_logged_in_mode(username, res)
            main_win.fill_table(client)
            Dialog.close()
        else:
            self.invaill_details_text.show()
            self.invaill_details_text.raise_()

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.sign_in_btn.setText(_translate("Dialog", "Sign in"))
        self.username_text.setText(_translate("Dialog", "Username:"))
        self.heading_text.setText(_translate("Dialog", "Login"))
        self.password_text.setText(_translate("Dialog", "Password:"))
        self.invaill_details_text.setText(_translate("Dialog", "invalid
username or password !"))
        self.fill_fields_text.setText(_translate("Dialog", "please fill all
the fields!"))

```

sign_up_Dialog.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'sign_up.ui'
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Sign_up_Dialog(object):
    # A function that receives a Dialog, client, main_win. The function
    # builds the dialog with its details
    def setupUi(self, Dialog, client, main_win):
        Dialog.setObjectName("Dialog")
        Dialog.resize(376, 489)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
        self.sign_up_btn = QtWidgets.QPushButton(Dialog)
        self.sign_up_btn.setGeometry(QtCore.QRect(120, 440, 101, 41))
        self.sign_up_btn.setStyleSheet("background-color: rgb(14, 154,
175);\n"
"color: rgb(255,255,255);\n"
"font-size: 18pt\n")

```



```

    "")
    self.sign_up_btn.setObjectName("pushButton_2")
    self.sign_up_btn.raise_()
    self.fullname_text = QtWidgets.QLabel(Dialog)
    self.fullname_text.setGeometry(QtCore.QRect(40, 80, 191, 21))
    font = QtGui.QFont()
    font.setPointSize(12)
    self.fullname_text.setFont(font)
    self.fullname_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 12pt")
    self.fullname_text.setObjectName("label_4")
    self.lineEdit_full_name = QtWidgets.QLineEdit(Dialog)
    self.lineEdit_full_name.setGeometry(QtCore.QRect(40, 100, 281, 31))
    self.lineEdit_full_name.setAutoFillBackground(False)
    self.lineEdit_full_name.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt\n"
"")
    self.lineEdit_full_name.setText("")
    self.lineEdit_full_name.setPlaceholderText("")
    self.lineEdit_full_name.setObjectName("lineEdit_4")
    self.heading_text = QtWidgets.QLabel(Dialog)
    self.heading_text.setGeometry(QtCore.QRect(30, 10, 261, 71))
    self.heading_text.setStyleSheet("background-color: rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 28pt \"MS Shell Dlg 2\";")
    self.heading_text.setObjectName("label_3")
    self.lineEdit_username = QtWidgets.QLineEdit(Dialog)
    self.lineEdit_username.setGeometry(QtCore.QRect(40, 160, 281, 31))
    self.lineEdit_username.setAutoFillBackground(False)
    self.lineEdit_username.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt\n"
"")
    self.lineEdit_username.setInputMethodHints(QtCore.Qt.ImhSensitiveData)
    self.lineEdit_username.setText("")
    self.lineEdit_username.setPlaceholderText("")
    self.lineEdit_username.setObjectName("lineEdit_3")
    self.username_text = QtWidgets.QLabel(Dialog)
    self.username_text.setGeometry(QtCore.QRect(40, 140, 111, 21))
    self.username_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 12pt")
    self.username_text.setObjectName("label_5")
    self.password_text = QtWidgets.QLabel(Dialog)
    self.password_text.setGeometry(QtCore.QRect(40, 260, 111, 21))
    self.password_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 12pt")
    self.password_text.setObjectName("label_7")
    self.confirm_password_text = QtWidgets.QLabel(Dialog)

```

```

        self.confirm_password_text.setGeometry(QtCore.QRect(40, 320, 181,
21))
        self.confirm_password_text.setStyleSheet("color:
rgb(255,255,255);\n"
"font-size: 12pt")
        self.confirm_password_text.setObjectName("label_8")
        self.lineEdit_email = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_email.setGeometry(QtCore.QRect(40, 220, 281, 31))
        self.lineEdit_email.setAutoFillBackground(False)
        self.lineEdit_email.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt")
        self.lineEdit_email.setInputMethodHints(QtCore.Qt.ImhSensitiveData)
        self.lineEdit_email.setText("")
        self.lineEdit_email.setPlaceholderText("")
        self.lineEdit_email.setObjectName("lineEdit_5")
        self.email_text = QtWidgets.QLabel(Dialog)
        self.email_text.setGeometry(QtCore.QRect(40, 200, 111, 21))
        self.email_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 12pt")
        self.email_text.setObjectName("label_9")
        self.lineEdit_password = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_password.setGeometry(QtCore.QRect(40, 280, 281, 31))
        self.lineEdit_password.setAutoFillBackground(False)
        self.lineEdit_password.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt")
        self.lineEdit_password.setInputMethodHints(QtCore.Qt.ImhHiddenText |
QtCore.Qt.ImhNoAutoUppercase | QtCore.Qt.ImhNoPredictiveText |
QtCore.Qt.ImhSensitiveData)
        self.lineEdit_password.setInputMask("")
        self.lineEdit_password.setText("")
        self.lineEdit_password.setEchoMode(QtWidgets.QLineEdit.Password)
        self.lineEdit_password.setPlaceholderText("")
        self.lineEdit_password.setObjectName("lineEdit_6")
        self.lineEdit_confirm_password = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_confirm_password.setGeometry(QtCore.QRect(40, 340,
281, 31))
        self.lineEdit_confirm_password.setAutoFillBackground(False)
        self.lineEdit_confirm_password.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt")

        self.lineEdit_confirm_password.setInputMethodHints(QtCore.Qt.ImhHiddenText
| QtCore.Qt.ImhNoAutoUppercase | QtCore.Qt.ImhNoPredictiveText |
QtCore.Qt.ImhSensitiveData)
        self.lineEdit_confirm_password.setInputMask("")
        self.lineEdit_confirm_password.setText("")

```

```

self.lineEdit_confirm_password.setEchoMode(QtWidgets.QLineEdit.Password)
self.lineEdit_confirm_password.setPlaceholderText("")
self.lineEdit_confirm_password.setObjectName("lineEdit_7")
self.fill_fields_text = QtWidgets.QLabel(Dialog)
self.fill_fields_text.setGeometry(QtCore.QRect(40, 380, 221, 21))
self.fill_fields_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
self.fill_fields_text.setObjectName("label_11")
self.not_same_passwords_text = QtWidgets.QLabel(Dialog)
self.not_same_passwords_text.setGeometry(QtCore.QRect(40, 380, 251,
21))
self.not_same_passwords_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
self.not_same_passwords_text.setObjectName("label_12")
self.user_exists_text = QtWidgets.QLabel(Dialog)
self.user_exists_text.setGeometry(QtCore.QRect(40, 380, 251, 21))
self.user_exists_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
self.invalid_email_text = QtWidgets.QLabel(Dialog)
self.invalid_email_text.setObjectName("label_14")
self.invalid_email_text.setGeometry(QtCore.QRect(40, 380, 251, 21))
self.invalid_email_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";")
self.invalid_password_text = QtWidgets.QTextEdit(Dialog)
self.invalid_password_text.setEnabled(False)
self.invalid_password_text.setGeometry(QtCore.QRect(30, 380, 351,
61))
self.invalid_password_text.setStyleSheet("color:
rgb(255,255,255);\n"
"font-size: 10pt")
self.invalid_password_text.setObjectName("textEdit")
self.invalid_password_text.setFrameShape(QtWidgets.QFrame.NoFrame)
self.fill_fields_text.hide()
self.not_same_passwords_text.hide()
self.user_exists_text.hide()
self.invalid_email_text.hide()
self.invalid_password_text.hide()
self.retranslateUi(Dialog)
QtCore.QMetaObject.connectSlotsByName(Dialog)
self.sign_up_btn.clicked.connect(lambda: self.try_sign_up(Dialog,
client, main_win))

# A function that receives Dialog, client, main_win. The function gets
the fields from the lines, does a validation, and send the details of the

```

user to the server to check if the username already exists. If not the server would add the user to the database The function would raise error labels when necessary

```
def try_sign_up(self, Dialog, client, main_win):
    self.invalid_password_text.hide()
    full_name = self.lineEdit_full_name.text()
    email = self.lineEdit_email.text()
    username = self.lineEdit_username.text()
    password = self.lineEdit_password.text()
    confirm_pass = self.lineEdit_confirm_password.text()
    if full_name == '' or email == '' or username == '' or password == '' or confirm_pass == '':
        self.fill_fields_text.show()
        self.fill_fields_text.raise_()
        return
    if password != confirm_pass:
        self.not_same_passwords_text.show()
        self.not_same_passwords_text.raise_()
        return
    if not self.strong_password_checker(password):
        self.invalid_password_text.show()
        self.invalid_password_text.raise_()
        return
    if not client.check_email(email):
        self.invalid_email_text.show()
        self.invalid_email_text.raise_()
        return
    res = client.sign_up(full_name, email, username, password)
    if res != None:
        main_win.turn_into_logged_in_mode(username, res)
        main_win.fill_table(client)
        Dialog.close()
    else:
        self.user_exists_text.show()
        self.user_exists_text.raise_()

# The function receives a password and checks if it's strong enough.
Returns True of False
def strong_password_checker(self, password):
    if len(password) < 6:
        return False
    if not any('a' <= c <= 'z' for c in password):
        return False
    if not any('A' <= c <= 'Z' for c in password):
        return False
    if not any(c.isdigit() for c in password):
        return False
    return True

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Sign_up", "Sign_up"))
```

```

        self.sign_up_btn.setText(_translate("Dialog", "Sign up"))
        self.fullname_text.setText(_translate("Dialog", "First and last
name:"))
        self.heading_text.setText(_translate("Dialog", "Personal details"))
        self.username_text.setText(_translate("Dialog", "Username:"))
        self.password_text.setText(_translate("Dialog", "Password:"))
        self.confirm_password_text.setText(_translate("Dialog", "Confirm
password:"))
        self.email_text.setText(_translate("Dialog", "e-mail:"))
        self.fill_fields_text.setText(_translate("Dialog", "please fill all
the fields!"))
        self.not_same_passwords_text.setText(_translate("Dialog", "make sure
you wrote the same password!"))
        self.user_exists_text.setText(_translate("Dialog", "username already
exists!"))
        self.invalid_email_text.setText(_translate("Dialog", "please enter a
valid email!"))
        self.invalid_password_text.setText(_translate("Dialog", "not a valid
password. enter a password that at least 6 characters long with a
combination of uppercase letters, lowercase letters, numbers"))

```

report_a_visit_Dialog.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'report_case.ui'
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
import os
from PyQt5 import QtCore, QtGui, QtWidgets
import results_Dialog
import send_curl
import progress_bar
from PIL import Image

class Ui_Report_a_visit_Dialog(object):
    can_add_photo = True
    view_mode = False

    #A function that receives MainWindow, client. The function builds the
    dialog with its details
    def setupUi(self, Dialog, client, main_win, doctor_id):

```

```

self.patient_dialog = Dialog
self.main_window = main_win
self.doctor_id= doctor_id
self.view_mode = False

Dialog.setObjectName("Dialog")
Dialog.resize(529, 751)
Dialog.setAutoFillBackground(False)
Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
Dialog.setInputMethodHints(QtCore.Qt.ImhNone)
self.visit_date_text = QtWidgets.QLabel(Dialog)
self.visit_date_text.setGeometry(QtCore.QRect(20, 530, 121, 16))
self.visit_date_text.setStyleSheet("color: rgb(255,255,255);\n"
                                   "font-size: 12pt")
self.visit_date_text.setObjectName("label_7")
self.email_text = QtWidgets.QLabel(Dialog)
self.email_text.setGeometry(QtCore.QRect(20, 210, 111, 21))
self.email_text.setStyleSheet("color: rgb(255,255,255);\n"
                              "font-size: 12pt")
self.email_text.setObjectName("label_9")
self.lineEdit_full_name = QtWidgets.QLineEdit(Dialog)
self.lineEdit_full_name.setGeometry(QtCore.QRect(20, 110, 391, 31))
self.lineEdit_full_name.setAutoFillBackground(False)
self.lineEdit_full_name.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                     "color: rgb(42, 42, 42);\n"
                                     "font-size: 12pt\n"
                                     "")
self.lineEdit_full_name.setText("")
self.lineEdit_full_name.setPlaceholderText("")
self.lineEdit_full_name.setObjectName("lineEdit_1")
self.lineEdit_id = QtWidgets.QLineEdit(Dialog)
self.lineEdit_id.setGeometry(QtCore.QRect(20, 170, 391, 31))
self.lineEdit_id.setAutoFillBackground(False)
self.lineEdit_id.setStyleSheet("background-color:
rgb(197,197,197);\n"
                              "color: rgb(42, 42, 42);\n"
                              "font-size: 12pt\n"
                              "")
self.lineEdit_id.setInputMethodHints(QtCore.Qt.ImhSensitiveData)
self.lineEdit_id.setText("")
self.lineEdit_id.setPlaceholderText("")
self.lineEdit_id.setObjectName("lineEdit_2")
self.lineEdit_email = QtWidgets.QLineEdit(Dialog)
self.lineEdit_email.setGeometry(QtCore.QRect(20, 230, 391, 31))
self.lineEdit_email.setAutoFillBackground(False)
self.lineEdit_email.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                  "color: rgb(42, 42, 42);\n"
                                  "font-size: 12pt")
self.lineEdit_email.setInputMethodHints(QtCore.Qt.ImhSensitiveData)
self.lineEdit_email.setText("")

```

```

        self.lineEdit_email.setPlaceholderText("")
        self.lineEdit_email.setObjectName("lineEdit_3")
        self.add_patient_heading_text = QtWidgets.QLabel(Dialog)
        self.add_patient_heading_text.setGeometry(QtCore.QRect(20, 10, 331,
71))
        self.add_patient_heading_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                                    "color: rgb(255, 255,
255);\n"
                                                    "font: 75 28pt \MS
Shell Dlg 2\";")
        self.add_patient_heading_text.setObjectName("label_3")
        self.watch_patient_heading_text = QtWidgets.QLabel(Dialog)
        self.watch_patient_heading_text.setGeometry(QtCore.QRect(20, 10,
331, 71))
        self.watch_patient_heading_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                                    "color: rgb(255, 255,
255);\n"
                                                    "font: 75 28pt \MS
Shell Dlg 2\";")
        self.watch_patient_heading_text.setObjectName("label_3")
        self.add_patient_heading_text.raise_()
        self.id_text = QtWidgets.QLabel(Dialog)
        self.id_text.setGeometry(QtCore.QRect(20, 150, 111, 16))
        self.id_text.setStyleSheet("color: rgb(255,255,255);\n"
                                   "font-size: 12pt")
        self.id_text.setObjectName("label_5")
        self.description_text = QtWidgets.QLabel(Dialog)
        self.description_text.setGeometry(QtCore.QRect(20, 390, 181, 16))
        self.description_text.setStyleSheet("color: rgb(255,255,255);\n"
                                             "font-size: 12pt")
        self.description_text.setObjectName("label_8")

        self.full_name_text = QtWidgets.QLabel(Dialog)
        self.full_name_text.setGeometry(QtCore.QRect(20, 90, 191, 16))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.full_name_text.setFont(font)
        self.full_name_text.setStyleSheet("color: rgb(255,255,255);\n"
                                           "font-size: 12pt")
        self.full_name_text.setObjectName("label_4")
        self.description_textEdit = QtWidgets.QTextEdit(Dialog)
        self.description_textEdit.setGeometry(QtCore.QRect(20, 410, 491,
111))
        self.description_textEdit.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                                "color: rgb(42, 42, 42);\n"
                                                "font-size: 12pt")
        self.description_textEdit.setObjectName("textEdit")
        self.save_btn = QtWidgets.QPushButton(Dialog)
        self.save_btn.setGeometry(QtCore.QRect(20, 700, 131, 31))

```

```

        self.save_btn.setStyleSheet("background-color: rgb(14, 154, 175);\n"
                                     "color: rgb(255,255,255);\n"
                                     "font-size: 18pt\n"
                                     "")
        self.save_btn.setObjectName("pushButton_2")
        self.x_ray_text = QtWidgets.QLabel(Dialog)
        self.x_ray_text.setGeometry(QtCore.QRect(20, 600, 171, 16))
        self.x_ray_text.setStyleSheet("color: rgb(255,255,255);\n"
                                       "font-size: 12pt")
        self.x_ray_text.setObjectName("label_10")
        self.birth_dateEdit = QtWidgets.QDateEdit(Dialog,
calendarPopup=True)
        self.birth_dateEdit.setGeometry(QtCore.QRect(20, 350, 171, 31))
        self.birth_dateEdit.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                         "color: rgb(42, 42, 42);\n"
                                         "font-size: 12pt")
        self.birth_dateEdit.setObjectName("dateEdit")
        self.visit_dateEdit = QtWidgets.QDateEdit(Dialog,
calendarPopup=True)
        self.visit_dateEdit.setGeometry(QtCore.QRect(20, 550, 171, 31))
        self.visit_dateEdit.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                         "color: rgb(42, 42, 42);\n"
                                         "font-size: 12pt")
        self.visit_dateEdit.setObjectName("dateEdit_2")
        self.lineEdit_photo_path = QtWidgets.QLineEdit(Dialog)
        self.lineEdit_photo_path.setGeometry(QtCore.QRect(20, 630, 491, 31))
        self.lineEdit_photo_path.setAutoFillBackground(False)
        self.lineEdit_photo_path.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                              "color: rgb(42, 42, 42);\n"
                                              "font-size: 12pt")
        self.lineEdit_photo_path.setInputMethodHints(QtCore.Qt.ImhSensitiveData)
        self.lineEdit_photo_path.setText("")
        self.lineEdit_photo_path.setPlaceholderText("")
        self.lineEdit_photo_path.setObjectName("lineEdit_4")
        self.fill_fields_text = QtWidgets.QLabel(Dialog)
        self.fill_fields_text.setGeometry(QtCore.QRect(20, 670, 221, 21))
        self.fill_fields_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                           "color: rgb(255, 255, 255);\n"
                                           "font: 75 12pt \"MS Shell Dlg
2\";")
        self.fill_fields_text.setObjectName("label_11")
        self.birth_date_text = QtWidgets.QLabel(Dialog)
        self.birth_date_text.setGeometry(QtCore.QRect(20, 330, 121, 16))
        self.birth_date_text.setStyleSheet("color: rgb(255,255,255);\n"
                                           "font-size: 12pt")
        self.birth_date_text.setObjectName("label_12")
        self.gender_text = QtWidgets.QLabel(Dialog)

```



```

self.gender_text.setGeometry(QtCore.QRect(20, 270, 111, 21))
self.gender_text.setStyleSheet("color: rgb(255,255,255);\n"
                                "font-size: 12pt")
self.gender_text.setObjectName("label_13")

self.comboBox = QtWidgets.QComboBox(Dialog)
self.comboBox.setGeometry(QtCore.QRect(20, 290, 101, 31))
self.comboBox.setStyleSheet("background-color: rgb(197,197,197);\n"
                              "color: rgb(42, 42, 42);\n"
                              "font-size: 12pt")
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.add_photo_btn = QtWidgets.QPushButton(Dialog)
self.add_photo_btn.setGeometry(QtCore.QRect(190, 600, 61, 21))
self.add_photo_btn.setStyleSheet("background-color: rgb(14, 154,
175);\n"
                                "color: rgb(255,255,255);\n"
                                "font-size: 12pt\n"
                                "")
self.add_photo_btn.setObjectName("pushButton_3")
self.invalid_id_text = QtWidgets.QLabel(Dialog)
self.invalid_id_text.setGeometry(QtCore.QRect(20, 670, 221, 21))
self.invalid_id_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                "color: rgb(255, 255, 255);\n"
                                "font: 75 12pt \"MS Shell Dlg
2\";")
self.invalid_id_text.setObjectName("label_1")
self.invalid_id_text.hide()
self.invalid_email_text = QtWidgets.QLabel(Dialog)
self.invalid_email_text.setGeometry(QtCore.QRect(20, 670, 221, 21))
self.invalid_email_text.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                "color: rgb(255, 255, 255);\n"
                                "font: 75 12pt \"MS Shell Dlg
2\";")
self.invalid_email_text.setObjectName("label_2")
self.invalid_email_text.hide()
self.birth_dateEdit.setDisplayFormat("dd-MM-yyyy")
self.birth_dateEdit.setDateTime(QtCore.QDateTime.currentDateTime())
self.visit_dateEdit.setDisplayFormat("dd-MM-yyyy")
self.visit_dateEdit.setDateTime(QtCore.QDateTime.currentDateTime())
self.fill_fields_text.hide()

self.retranslateUi(Dialog)
QtCore.QMetaObject.connectSlotsByName(Dialog)

self.add_photo_btn.clicked.connect(lambda:
self.select_X_ray(main_win))
self.save_btn.clicked.connect(lambda:
self.handle_save_btn_click(client))

```

```

    #A function that receives client. The function gets the fields from the
    lines and does a validation. The function checks if x_ray_detection is
    necessary. If yes, the function calls the worker and the progress bar. If
    not, the function saves the patient's details.
    def handle_save_btn_click(self, client):
        full_name = self.lineEdit_full_name.text()
        ID = self.lineEdit_id.text()
        email = self.lineEdit_email.text()
        case_description = self.description_textEdit.toPlainText()

        # checks if all the fields were filled
        if full_name == '' or ID == '' or email == '' or case_description ==
        '':
            self.fill_fields_text.show()
            self.fill_fields_text.raise_()
            return

        # checks: if the length of the id is valid, if the string contains
        only number and if the id is someone's else
        if len(ID) != 9 or not ID.isnumeric():
            self.invalid_id_text.show()
            self.invalid_id_text.raise_()
            return

        if not self.view_mode:
            if not self.check_ID(client, ID, self.doctor_id):
                self.invalid_id_text.show()
                self.invalid_id_text.raise_()
                return

        if not client.check_email(email):
            self.invalid_email_text.show()
            self.invalid_email_text.raise_()
            return

        photo_path = self.lineEdit_photo_path.text()
        parameters_tuple = (photo_path, True)

        if self.main_window.the_app_config.is_using_debug_values():
            parameters_tuple =
            (self.main_window.the_app_config.get_string_value('debug', 'xray_path'),
            True)

        if parameters_tuple[0] == '' or not self.can_add_photo:
            self.save_patient(None, client)
        else:
            print(parameters_tuple)
            self.worker_dialog = QtWidgets.QDialog()
            self.ui = progress_bar.Worker_Dialog(self.worker_dialog)
            self.ui.setupUi(self.worker_dialog, client, send_curl.send_curl,
            parameters_tuple, self.save_patient)
            self.worker_dialog.show()

    # A function that receives res_tuple_pneumonia, client. The function

```

gets the fields from the lines and sends the details of the patient to the server. If an x-ray was uploaded the function would send it to the web server. The function will close the Dialog at the end of the function.

```
def save_patient(self, res_tuple_pneumonia, client):
    full_name = self.lineEdit_full_name.text()
    ID = self.lineEdit_id.text()
    email = self.lineEdit_email.text()
    birth_date = self.birth_dateEdit.date()
    birth_date = str(str(birth_date.toPyDate()))
    gender = self.comboBox.currentText()
    visit_date = self.visit_dateEdit.date()
    visit_date = str(str(visit_date.toPyDate()))
    case_description = self.description_textEdit.toPlainText()
    photo_path = self.lineEdit_photo_path.text()

    is_pneumonia, x_ray_detection =
self.main_window.get_pneumonia_probability(res_tuple_pneumonia)
    show_results = True

    if photo_path != '' and not self.can_add_photo:
        x_ray_detection = photo_path
        show_results = False

    if photo_path != '' and self.can_add_photo:
        self.can_add_photo = False
        self.resize_and_save_photo(photo_path, ID)

    client.add_patient(full_name, ID, email, gender, visit_date,
case_description, birth_date, str(x_ray_detection), self.doctor_id)
    if show_results and not self.can_add_photo:
        self.open_results_Dialog()
        if is_pneumonia is not None:
            if self.view_mode:
                self.ui.turn(is_pneumonia, "edit")
            else:
                self.ui.turn(is_pneumonia, "add")
        self.main_window.clear_table()
        self.main_window.fill_table(client)

    self.patient_dialog.close()

# A function that receives a photo path and patient_id. The function
resizes the image and saves it in the "x_ray_stats" folder.
def resize_and_save_photo(self, photo_path, ID):
    img = Image.open(photo_path)
    img.thumbnail((150, 100))
    os.makedirs('x_ray_stats', exist_ok=True)
    image_path = "x_ray_stats"
    img.save(f"{image_path}/{ID} + ".jpeg")

# A function that receives a photo path, and sends an HTTP request to
the web server. The function returns the detection of the photo by the AI
```

```

model.
    def send_curl(self, photo_path):
        return send_curl.send_curl(photo_path)

    #A function that receives client, ID, doctor_id. The
    def check_ID(self, client, ID, doctor_id):
        patient_list = client.get_patients_list(doctor_id)
        if patient_list is None:
            return True
        for row in patient_list:
            row = row.split('#')
            if row[1] == ID:
                return False
        return True

    #A function that opens the results dialog
    def open_results_Dialog(self):
        self.dialog = QtWidgets.QDialog()
        self.ui = results_Dialog.Ui_Results_Dialog()
        self.ui.setupUi(self.dialog)
        self.dialog.show()

    # A function that receives the main_window. The function call to
    another function that ask the user to choose x-ray photo from the
    computer. The function updates the lineEdit of the photo path with the path
    the user selected.
    def select_X_ray(self, main_win):
        _translate = QtCore.QCoreApplication.translate
        photo_path = main_win.get_file_name()
        self.lineEdit_photo_path.setText(_translate("Dialog", photo_path))

    #A function that receives client, patient_id. The function changes the
    display into view mode. It puts the details of the selected patient in the
    appropriate fields.
    def turn_into_show_patient_mode(self, client, patient_id):
        self.view_mode = True
        self.fill_fields_text.raise_()
        self.watch_patient_heading_text.raise_()
        self.lineEdit_full_name.setEnabled(False)
        self.lineEdit_id.setEnabled(False)
        self.comboBox.setEnabled(False)
        self.birth_dateEdit.setEnabled(False)
        patient = client.get_specific_patient(patient_id)
        if patient is None:
            return
        spllited_patient = patient.split('#')
        _translate = QtCore.QCoreApplication.translate
        self.lineEdit_full_name.setText(_translate("Dialog",
spllited_patient[0]))
        self.lineEdit_id.setText(_translate("Dialog", spllited_patient[1]))
        self.lineEdit_email.setText(_translate("Dialog",
spllited_patient[2]))

```

```

        if spllited_patient[3] == "Female":
            self.comboBox.setCurrentIndex(1)
        birth_date =
self.from_string_to_date(spllited_patient[4].split('-'))
        self.birth_dateEdit.setDate(birth_date)
        self.description_textEdit.setText(_translate("Dialog",
spllited_patient[5]))
        visit_date =
self.from_string_to_date(spllited_patient[6].split('-'))
        self.visit_dateEdit.setDate(visit_date)
        if len(spllited_patient) > 8:
            self.can_add_photo = False
            self.birth_date_text.raise_()
            self.add_photo_btn.setEnabled(False)
            self.lineEdit_photo_path.setText(_translate("Dialog",
spllited_patient[8]))

#A function that receives a date(string) spllited in a list. The
function returns the date as a QDate type.
def from_string_to_date(self, spllited_date):
    str_date = spllited_date[2] + spllited_date[1] + spllited_date[0]
    date = QtCore.QDate.fromString(str_date, 'ddMMyyyy')
    return date

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.visit_date_text.setText(_translate("Dialog", "Visit date:"))
    self.email_text.setText(_translate("Dialog", "E-mail:"))
    self.add_patient_heading_text.setText(_translate("Dialog", "Report a
visit"))
    self.watch_patient_heading_text.setText(_translate("Dialog",
"Patient's details"))
    self.id_text.setText(_translate("Dialog", "ID:"))
    self.description_text.setText(_translate("Dialog", "Description of
the case:"))
    self.full_name_text.setText(_translate("Dialog", "Full name:"))
    self.save_btn.setText(_translate("Dialog", "Save"))
    self.x_ray_text.setText(_translate("Dialog", "X-ray photo
(Optional):"))
    self.fill_fields_text.setText(_translate("Dialog", "please fill all
the fields!"))
    self.birth_date_text.setText(_translate("Dialog", "Birth date:"))
    self.gender_text.setText(_translate("Dialog", "Gender:"))
    self.comboBox.setItemText(0, _translate("Dialog", "Male"))
    self.comboBox.setItemText(1, _translate("Dialog", "Female"))
    self.add_photo_btn.setText(_translate("Dialog", "add"))
    self.invalid_id_text.setText(_translate("Dialog", "please enter a
valid ID!"))
    self.invalid_email_text.setText(_translate("Dialog", "please enter a
valid email!"))

```

date_picker.py

```
# importing libraries
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys

class Window(QMainWindow):
    def __init__(self):
        super().__init__()
        # setting title
        self.setWindowTitle("Python ")
        # setting geometry
        self.setGeometry(100, 100, 650, 400)
        # calling method
        self.UiComponents()
        # showing all the widgets
        self.show()

    # method for components
    def UiComponents(self):
        # creating a QCalendarWidget object
        self.calendar = QCalendarWidget(self)
        # setting geometry to the calendar
        self.calendar.setGeometry(50, 10, 400, 250)
        # setting cursor
        self.calendar.setCursor(Qt.PointingHandCursor)
        # removing special cursor
        self.calendar.unsetCursor()
        # setting accept delay value
        self.calendar.setDateEditAcceptDelay(2000)

# create pyqt5 app
App = QApplication(sys.argv)
# create the instance of our Window
window = Window()
# start the app
sys.exit(App.exec())
```

send_curl.py

```
import json
import requests

#A function that receives a photo path. The function sends an HTTP request
to the web server. The function returns the detection of the photo by the
AI model.
def send_curl(photo_path):
    headers = {
        'accept': 'application/json',
    }

    files = {
        'image_file': open(photo_path, 'rb'),
    }
    res = None
    try:
        response = requests.post('http://localhost:5000/pneumonia/predict',
headers=headers, files=files)
        res = response.text
        response.raise_for_status()
    except requests.exceptions.ConnectionError as errc:
        print("Error Connecting:", errc)
        res = "{\"predicted_class\": \"pneumonia\",
\"pneumonia_probability\": \"[0.97447765]\"}"
        print(res)
    jRes = json.loads(res)
    print(jRes)

    is_pneumonia = str(jRes['predicted_class']) == 'pneumonia'

    pneumonia_chance = str(jRes['pneumonia_probability'])
    pneumonia_chance = pneumonia_chance[1:-1]
    pneumonia_chance = float(pneumonia_chance) * 100
    pneumonia_chance = str(pneumonia_chance)
    pneumonia_chance = pneumonia_chance[:4]
    pneumonia_chance = float(pneumonia_chance)
    if not is_pneumonia and pneumonia_chance < 0.5:
        pneumonia_chance= str(1-pneumonia_chance) + "%"
    else:
        pneumonia_chance= str(pneumonia_chance) + "%"
    print(pneumonia_chance)

    res_tuple = (is_pneumonia, pneumonia_chance)
    return res_tuple

if __name__ == '__main__':
```

```
send_curl(r"D:\Projects\pneumonia-detection\fixtures\pneumonia_1.jpeg")
```

results_Dialog.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'results_dialog.ui'
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Results_Dialog(object):
    # A function that receives a Dialog. The function builds the dialog with
    # its details
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(601, 321)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
        Dialog.setModal(True)
        self.add_patient_heading = QtWidgets.QLabel(Dialog)
        self.add_patient_heading.setGeometry(QtCore.QRect(20, 20, 441, 41))
        self.edit_patient_heading = QtWidgets.QLabel(Dialog)
        self.edit_patient_heading.setGeometry(QtCore.QRect(20, 20, 441, 41))
        self.x_ray_uploaded_heading = QtWidgets.QLabel(Dialog)
        self.x_ray_uploaded_heading.setGeometry(QtCore.QRect(20, 20, 441,
41))
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(20)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(9)
        self.add_patient_heading.setFont(font)
        self.add_patient_heading.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255, 255);\n"
"font: 75 20pt \"MS Shell Dlg 2\";")
        self.add_patient_heading.setObjectName("label_3")
        self.edit_patient_heading.setFont(font)
        self.edit_patient_heading.setStyleSheet("background-color:
rgb(54,54,54);\n"
"color: rgb(255, 255,
```



```

255);\n"
                                "font: 75 20pt \"MS Shell Dlg
2\";")
    self.edit_patient_heading.setObjectName("label_3")
    self.x_ray_uploaded_heading.setFont(font)
    self.x_ray_uploaded_heading.setStyleSheet("background-color:
rgb(54,54,54);\n"
                                "color: rgb(255, 255,
255);\n"
                                "font: 75 20pt \"MS Shell Dlg
2\";")
    self.x_ray_uploaded_heading.setObjectName("label_3")
    self.pneumonia_text = QtWidgets.QTextEdit(Dialog)
    self.pneumonia_text.setEnabled(False)
    self.pneumonia_text.setGeometry(QtCore.QRect(10, 70, 491, 241))
    self.pneumonia_text.setStyleSheet("color: rgb(255,255,255);\n"
"font-size: 16pt")
    self.pneumonia_text.setObjectName("textEdit_2")
    self.pneumonia_text.setFrameShape(QtWidgets.QFrame.NoFrame)

    self.label = QtWidgets.QLabel(Dialog)
    self.label.setGeometry(QtCore.QRect(490, 0, 91, 91))
    self.label.setText("")
    self.label.setPixmap(QtGui.QPixmap("results_logo.png"))
    self.label.setObjectName("label")
    self.normal_text = QtWidgets.QTextEdit(Dialog)
    self.normal_text.setEnabled(False)
    self.normal_text.setGeometry(QtCore.QRect(10, 70, 491, 241))
    self.normal_text.setStyleSheet("color: rgb(255,255,255);\n"
"background-color: rgb(54, 54, 54);\n"
"font-size: 16pt")
    self.normal_text.setFrameShadow(QtWidgets.QFrame.Sunken)
    self.normal_text.setObjectName("textEdit_3")
    self.normal_text.setFrameShape(QtWidgets.QFrame.NoFrame)

    self.ok_btn = QtWidgets.QPushButton(Dialog)
    self.ok_btn.setGeometry(QtCore.QRect(510, 270, 81, 41))
    self.ok_btn.setStyleSheet("background-color: rgb(14, 154, 175);\n"
"color: rgb(255,255,255);\n"
"font-size: 18pt\n"
"")
    self.ok_btn.setObjectName("pushButton_4")
    self.label.raise_()
    self.ok_btn.raise_()

    self.normal_text.hide()
    self.pneumonia_text.hide()
    self.ok_btn.clicked.connect(lambda: self.close_dialog(Dialog))
    self.retranslateUi(Dialog)
    QtCore.QMetaObject.connectSlotsByName(Dialog)
# A function that receives a detection of an x-ray and a reason to call

```

```

the results dialog. The function changes the display of the dialog
according to these parameters.
def turn(self, is_pneumonia, call_reason):
    if is_pneumonia:
        self.pneumonia_text.show()
        self.pneumonia_text.raise_()
    else:
        self.normal_text.show()
        self.normal_text.raise_()
    if call_reason == "add":
        self.add_patient_heading.raise_()
    elif call_reason == "edit":
        self.edit_patient_heading.raise_()
    elif call_reason == "x-ray":
        self.x_ray_uploaded_heading.raise_()

#A function that receives a Dialog and close it.
def close_dialog(self, Dialog):
    Dialog.close()

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.add_patient_heading.setText(_translate("Dialog", "Patient was
added susccesfully"))
    self.edit_patient_heading.setText(_translate("Dialog", "Patient was
edited susccesfully"))
    self.x_ray_uploaded_heading.setText(_translate("Dialog", "X-ray was
uploaded susccesfully"))
    self.pneumonia_text.setHtml(_translate("Dialog", "<!DOCTYPE HTML
PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style
type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:'MS Shell Dlg 2';
font-size:16pt; font-weight:400; font-style:normal;\n">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\n">Our detection is
that your patient <span style=\" font-weight:600;\n">is suffering</span>
from pneumania.</p>\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0;
text-indent:0px;\n"><br /></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\n"><span style=\"
font-size:14pt;\n">In that case we reccomend you to make sure your diagnose
is accurate and that you are giving the right treatment.</span></p>\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;
font-size:14pt;\n"><br /></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;

```

```

margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\"
font-size:12pt;\">We want to remind you that the results</span><span
style=\" font-size:12pt; font-weight:600;\"> are not</span><span style=\"
font-size:12pt;\"> accurate in 100%.</span></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\"
font-size:12pt;\">The tool is an </span><span style=\" font-size:12pt;
font-weight:600;\">advisory tool only</span><span style=\"
font-size:12pt;\"> and is not intended to replace the doctor\'s
judgment.</span></p></body></html>"))
        self.normal_text.setHtml(_translate("Dialog", "<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style
type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\';
font-size:16pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\">Our detection is
that your patient <span style=\" font-weight:600;\">isn\'t suffering</span>
from pneumonia.</p>\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;
font-size:14pt;\"><br /></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\"
font-size:14pt;\">In that case we reccomend you to make sure your diagnose
is accurate. We reccomnd you to check if the patient is suffering from
another disease</span><span style=\" font-size:12pt;\">.</span></p>\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;
font-size:12pt;\"><br /></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\"
font-size:12pt;\">We want to remind you that the results</span><span
style=\" font-size:12pt; font-weight:600;\"> are not</span><span style=\"
font-size:12pt;\"> accurate in 100%.</span></p>\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\"
font-size:12pt;\">The tool is an </span><span style=\" font-size:12pt;
font-weight:600;\">advisory tool only</span><span style=\"
font-size:12pt;\"> and is not intended to replace the doctor\'s
judgment.</span></p></body></html>"))
        self.ok_btn.setText(_translate("Dialog", "Ok"))

```

statistical_analysis.py

```
# -*- coding: utf-8 -*-
```

```

# Form implementation generated from reading ui file 'try.ui'
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.
import os
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Statistical_analysis_Dialog(object):
    # A function that receives a Dialog. The function builds the dialog with
    # its details
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(640, 705)
        Dialog.setModal(True)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
        self.tableWidget = QtWidgets.QTableWidget(Dialog)
        self.tableWidget.setGeometry(QtCore.QRect(10, 210, 531, 461))
        self.tableWidget.setStyleSheet("background-color:
rgb(197,197,197);\n"
"color: rgb(42, 42, 42);\n"
"font-size: 12pt")

        self.tableWidget.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers
)

        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setColumnCount(3)
        self.tableWidget.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setPointSize(12)
        item.setFont(font)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        item.setForeground(brush)
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setPointSize(12)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setPointSize(12)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(2, item)
        self.tableWidget.verticalHeader().setDefaultSectionSize(100)
        self.tableWidget.horizontalHeader().setDefaultSectionSize(150)
        self.heading_text = QtWidgets.QLabel(Dialog)
        self.heading_text.setGeometry(QtCore.QRect(20, 20, 331, 71))
        self.heading_text.setStyleSheet("background-color: rgb(54,54,54);\n"

```

```

"color: rgb(255, 255, 255);\n"
"font: 75 28pt \"MS Shell Dlg 2\";")
    self.heading_text.setObjectName("label_4")
    self.textEdit = QtWidgets.QTextEdit(Dialog)
    self.textEdit.setEnabled(False)
    self.textEdit.setGeometry(QtCore.QRect(20, 80, 571, 131))
    self.textEdit.setStyleSheet("color: rgb(255,255,255);\n"
                                "font-size: 15pt")
    self.textEdit.setObjectName("textEdit")
    self.textEdit setFrameShape(QtWidgets.QFrame.NoFrame)
    self.textEdit.raise_()
    self.retranslateUi(Dialog)
    QtCore.QMetaObject.connectSlotsByName(Dialog)
# A function that receives directory_path, client, dialog, doctor_id.
The function fills a table with x-ray images and it's results.
def fill_stats_table(self, directory_path, client, dialog, doctor_id):
    count = 0
    patient_list = client.get_patients_list(doctor_id)
    if patient_list is None:
        return
    for filename in os.listdir(directory_path):
        image_label = QtWidgets.QLabel(dialog)
        image_label.setText("")
        image_label.setScaledContents(True)
        pixmap = QtGui.QPixmap()
        # Convert digital data to binary format
        photo_path = os.path.join(directory_path, filename)
        with open(photo_path, 'rb') as file:
            binaryData = file.read()
            pixmap.loadFromData(binaryData, 'jpeg')
            image_label.setPixmap(pixmap)
            item = image_label
            filename = filename.split('.')
            for row in patient_list:
                row = row.split('#')
                if row[1] == filename[0]:
                    results = row[8]
                    if results == 'None':
                        continue
                    results = results.split(' ')
                    self.tableWidget.insertRow(self.tableWidget.rowCount())
                    self.tableWidget.setCellWidget(count, 0, item)
                    self.tableWidget.setItem(count, 1,
QtWidgets.QTableWidgetItem(results[0]))
                    self.tableWidget.setItem(count, 2,
QtWidgets.QTableWidgetItem(results[2]))
                    count += 1
                break
    _translate = QtCore.QCoreApplication.translate
    self.textEdit.setText(_translate("Dialog", "Below are presented some
statistical data regarding the x-rays you uploaded to the system.\r\nX-rays
uploaded: " + str(self.tableWidget.rowCount()) + "\r\nX-rays that were

```

```

detected ad pneumonia: " + self.count_pneu_among_table() + "\r\nOverall
accuracy: " + self.check_accuracy())

# A function that counts the number of x-rays that were detected as
pneumonia
def count_pneu_among_table(self):
    count = 0
    for row in range(self.tableWidget.rowCount()):
        if self.tableWidget.item(row, 1).text() == 'pneumonia':
            count += 1
    return str(count)

# A function that calculates the average accuracy of detection.
def check_accuracy(self):
    sum = 0
    for row in range(self.tableWidget.rowCount()):
        result = self.tableWidget.item(row, 2).text()
        result = result[:-1]
        sum += float(result)
    sret = str(sum/self.tableWidget.rowCount())
    return sret[:4] + "%"

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.tableWidget.setSortingEnabled(True)
    item = self.tableWidget.horizontalHeaderItem(0)
    item.setText(_translate("Dialog", "X-ray"))
    item = self.tableWidget.horizontalHeaderItem(1)
    item.setText(_translate("Dialog", "Detection"))
    item = self.tableWidget.horizontalHeaderItem(2)
    item.setText(_translate("Dialog", "Accuracy"))
    self.heading_text.setText(_translate("Dialog", "Statistical
analysis:"))

```

progress_bar.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file
'app_code\progress_bar.ui'
#
# Created by: PyQt5 UI code generator 5.15.9
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.

```

```

# -- sample for worker
# https://realpython.com/python-pyqt-qthread/

# -- sample for signal & slots
#
https://stackoverflow.com/questions/64343633/how-to-return-a-variable-from-
another-window-in-pyqt
from PyQt5.QtWidgets import (QProgressBar)
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (
    QApplication,
    QPushButton,
    QVBoxLayout,
)
from PyQt5.QtCore import QObject, QThread, pyqtSignal
from PyQt5 import QtCore, QtGui, QtWidgets

class Worker(QObject):

    def __init__(self, caller):
        super().__init__()
        self.the_caller = caller

        finished = pyqtSignal()
        progress = pyqtSignal(int)
        response_tuple = None

        #A function that runs the caller function. It informs when the function
        is finished.
        def run(self):
            """Long-running task."""
            self.response_tuple =
self.the_caller.funcForBackgroundWorker(self.the_caller.inParams[0])
            # for i in range(5):
            #     sleep(1)
            #     self.progress.emit(i + 1)

            self.finished.emit()

class Worker_Dialog(object):

    def __init__(self, Dialog, parent=None):
        self.clicksCount = 0
        self.dialog = Dialog

        # A function that receives Dialog, client, func_for_background_worker,
        parameters_tuple, func_for_callback. The function builds the dialog with
        its details
        def setUpUi(self, Dialog, client, func_for_background_worker,
parameters_tuple, func_for_callback):
            self.funcForBackgroundWorker = func_for_background_worker

```

```

        self.funcForCallback = func_for_callback
        self.inParams = parameters_tuple
        self.theClient = client
        Dialog.setWindowTitle("Diagnose pneumonia")
        Dialog.resize(300, 100)
        Dialog.setModal(True)
        # Create and connect widgets
        self.diagnose_x_ray_btn = QPushButton("Click to detect X-ray!",
Dialog)
        self.diagnose_x_ray_btn.setStyleSheet("background-color:
rgb(197,197,197);\n"
                                             "color: rgb(255,255,255);\n"
                                             "font-size: 18pt\n"
                                             "")
        self.diagnose_x_ray_btn.clicked.connect(self.run_long_task)
        self.progress = QProgressBar(Dialog)
        self.progress.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
        self.progress.setMaximum(0)
        self.progress.hide()

        # Set the layout
        layout = QVBoxLayout()
        layout.addWidget(self.diagnose_x_ray_btn)
        layout.addWidget(self.progress)
        layout.addStretch()
        Dialog.setLayout(layout)

    #A function that starts a new thread, runs the worker, shows the
progress and the calls after_worker_finished when the finction is finished
    def run_long_task(self):
        self.progress.show()
        # Step 2: Create a QThread object
        self.thread = QThread()
        # Step 3: Create a worker object
        self.worker = Worker(self)
        # Step 4: Move worker to the thread
        self.worker.moveToThread(self.thread)
        # Step 5: Connect signals and slots
        self.thread.started.connect(self.worker.run)
        self.worker.finished.connect(self.thread.quit)
        self.worker.finished.connect(self.worker.deleteLater)
        self.thread.finished.connect(self.thread.deleteLater)
        # Step 6: Start the thread
        self.thread.start()

        # Final resets
        self.diagnose_x_ray_btn.setEnabled(False)

        self.thread.finished.connect(
            lambda: self.after_worker_finished()
        )

```



```

    #A function that is called after the run_long_task is finished. The
    function is calling to the callback function.
    def after_worker_finished(self):
        self.funcForCallback(self.worker.response_tuple, self.theClient)
        self.diagnose_x_ray_btn.setEnabled(True)
        self.progress.hide()
        self.dialog.close()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    workDlg = QtWidgets.QDialog()
    ui = Worker_Dialog(workDlg)
    workDlg.show()
    sys.exit(app.exec())

```

config.py

```

from configparser import ConfigParser

class AppConfig(object):
    the_config_parser = None

    def __init__(self, config_file_path):
        self.the_config_parser = ConfigParser()
        self.read_config(config_file_path)

    #A function that receives config_file_path. The function prints details
    about the config files.
    def read_config(self, config_file_path):
        print(self.the_config_parser.read(config_file_path))

        print("config sections are: ", self.the_config_parser.sections())
        print("username:", self.the_config_parser.get('debug', 'user_name'))
        print("port: ", self.the_config_parser.getint('server', 'port'))

    # A function that receives category, name. The function returns the
    value from the config file in accordance to the parameters. (Strings only)
    def get_string_value(self, category, name):
        return self.the_config_parser.get(category, name)

    # A function that receives category, name. The function returns the
    value from the config file in accordance to the parameters. (ints only)
    def get_int_value(self, category, name):
        return self.the_config_parser.getint(category, name)

    # A function that receives category, name. The function returns the
    value from the config file in accordance to the parameters. (booleans only)
    def get_bool_value(self, category, name):
        return self.the_config_parser.getboolean(category, name)

```

```

# A function that returns if the current run is debug values in
accordance to the config file.
def is_using_debug_values(self):
    return self.the_config_parser.getboolean('debug',
'use_debug_values')

# A function that returns the username value from the config file.
def get_user_name(self):
    return self.the_config_parser.get('debug', 'user_name')

# A function that returns the password value from the config file.
def get_password(self):
    return self.the_config_parser.get('debug', 'user_password')

```

app.py

```

from fastapi import FastAPI
from classifier.routers import pneumonia_router

app = FastAPI()
app.include_router(pneumonia_router.router, prefix='/pneumonia') # noqa

@app.get('/healthcheck', status_code=200)
async def healthcheck():
    return 'Good to go'

```

AI_model.py

```

import os
import zipfile
import logging

from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

class AI_model:
    def __init__(self, force_download=False):

```

```

self.force_download = force_download
self.data_path = os.path.join(os.getcwd(), 'classifier\\data\\')
self.model_path = os.path.join(os.getcwd(), 'classifier\\models\\')
self.dataset_name = 'paultimothymooney/chest-xray-pneumonia'
self.model: Sequential
self.val: ImageDataGenerator
self.train: ImageDataGenerator
self.test: ImageDataGenerator
self.epochs = 5

#A function that loads the x-ray files.
def load_data(self):
    #self.download_data()
    train_generator = ImageDataGenerator(rescale=1/255.0)
    test_generator = ImageDataGenerator(rescale=1/255.0)
    val_generator = ImageDataGenerator(rescale=1/255.0)
    self.train =
train_generator.flow_from_directory(os.path.join(self.data_path,
'chest_xray\\chest_xray\\train'),
                                target_size=(64,
64),
                                batch_size=32,
                                color_mode='grayscale',
                                class_mode='binary')
    self.test =
test_generator.flow_from_directory(os.path.join(self.data_path,
'chest_xray\\chest_xray\\test'),
                                target_size=(64,
64),
                                batch_size=32,
                                color_mode='grayscale',
                                class_mode='binary')

    self.val =
val_generator.flow_from_directory(os.path.join(self.data_path,
'chest_xray\\chest_xray\\test'),
                                target_size=(64, 64),
                                batch_size=32,
                                color_mode='grayscale',
                                class_mode='binary')

    logger.info('{} images in training set; {} are
PNEUMONIA'.format(self.train.samples, sum(self.train.labels)))

def download_data(self):
    if self.force_download or (not os.path.isdir(self.data_path) and not
os.path.isdir(self.model_path)):
        import kaggle
        kaggle.api.authenticate()

```

```

        kaggle.api.dataset_download_files(self.dataset_name,
path=self.data_path)
        with zipfile.ZipFile(os.path.join(self.data_path,
'chest-xray-pneumonia.zip')) as z:
            z.extractall(self.data_path)

#A function that defines the AI model. The function returns the model.
def define_model(self):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64,
1)))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    return model

#A function that trains the model.
def train_model(self):
    self.load_data()
    steps_per_epoch = self.train.samples // self.train.batch_size
    validation_steps = self.val.samples // self.val.batch_size
    self.model = self.define_model()
    self.model.fit_generator(self.train,
                            epochs=self.epochs,
                            validation_data=self.val,
                            steps_per_epoch=steps_per_epoch,
                            validation_steps=validation_steps)

#A function that calls the train_model function and save the new trained
model to the directory.
def deploy_model(self):
    self.train_model()
    loss, acc = self.model.evaluate_generator(self.test,
                                              steps=self.test.samples //
self.test.batch_size)
    logger.info('Model has been trained with loss, accuracy of {},
{}'.format(loss, acc))
    self.model.save_weights(os.path.join(self.model_path, 'weights.h5'))
    logger.info('Model weights have been saved to
{}'.format(self.model_path))

if __name__ == '__main__':
    #train_model = Train(force_download=True)
    train_model = Train()
    train_model.deploy_model()

```

pneumonia_router.py

```

from io import io
import tensorflow as tf

from fastapi import APIRouter, File
from PIL import Image
#from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import img_to_array

from classifier.train import Train

router = APIRouter()

def predict_prob(number):
    return [number[0], 1-number[0]]

@router.post('/predict')
def pneumonia_router(image_file: bytes = File(...)):
    #model = Train().define_model()
    #model.load_weights('classifier/models/weights.h5')

    image = Image.open(io.BytesIO(image_file))

    if image.mode != 'L':
        image = image.convert('L')

    image = image.resize((64, 64))
    image = img_to_array(image)/255.0
    image = image.reshape(1, 64, 64, 1)

    #graph = tf.get_default_graph()
    graph = tf.compat.v1.get_default_graph()

    with graph.as_default():
        model = Train().define_model()
        model.load_weights('classifier/models/weights.h5')
        prediction = model.predict(image)
        #prediction = model.predict_proba(image)

    predicted_class = 'pneumonia' if prediction[0] > 0.5 else 'normal'

    return {'predicted_class': predicted_class,
            'pneumonia_probability': str(prediction[0])}

```