תיק פרויקט

שם בית ספר: מו"ר מכבים ורעות

שם העבודה: SoEasy

שם התלמיד: עדו שמרי

ת"ז התלמיד: 216007187

שם המנחה: אסתר הרשקוביץ

תאריך ההגשה: 10.5.23

תוכן עניינים

הצעת פרויקט

תיאור ראשוני של הפרויקט:

תקציר הפרויקט

העברת קבצים היא עניין מרכזי בעולם המחשבים והסייבר. בגלל חשיבות זו, פותח פרוטוקול המיועד להעברת קבצים בין מחשבים שונים באינטרנט או מחשבים ברשת הפנימית – פרוטוקול FTP.

העברת קבצים בין מכשירי סלולר למחשב מתבצעת כיום בעיקר דרך חיבור לאפליקציות כגון מייל וווטסאפ, שלהם יש להירשם מראש וכן וישנן מגבלות העברה שונות אחרות.
הפרויקט שלי יתמקד בפיתוח שיטה פשוטה ונוחה להעברת קבצים מכשיר Android למחשב, ללא צורך בתיווך של אפליקציות מהסוג המתואר למעלה.

מה המוצר המוגמר יהיה אמור לבצע

המוצר המוגמר הינו פתרון ייעודי ונוח להעברת קבצים, שיכלול תוכנת PC ותוכנת אנדרואיד, שיאפשרו העברה פשוטה של הקבצים בעזרת התחברות דרך WIFI. ההתחברות תקרא כך:
המחשב ייצור קוד QR ובו מכילים הפרטים שהטלפון יצטרכו בשביל ליצור התחברות, והטלפון פותח סורק קודי QR, סורק את הקוד ומתחבר דרך WIFI.
לאחר ההתחברות כל אחד מהמכשירים בוחר איזה קבצים הוא רוצה להעביר, ובוחר מיקומים לקבצים שהמכשיר השני רוצה להעביר אליו.

למה בחרתי בפרויקט הזה ומה האתגרים שאני צופה שיהיו לי

בחרתי בפרויקט הזה כי הנושא של העברת קבצים הוא מעניין והכרחי, אבל כיום, העברת קבצים בין הטלפון למחשב היא מסורבלת ואני רוצה למצא פתרון נוח ופשוט.

האתגרים שאני צופה שיהיו לי:

- למידת פרוטוקול FTP והבנה של איך להעביר קבצים בין טלפון למחשב.
- למידה של QR קוד והבנה של איך להעביר דרכו מידע לטלפון.
- למידה של יצירת חיבור דרך WIFI

הגדרת לקוח

הלקוחות של פרויקט זה יכולים להיות כל אחד שרוצה להעביר קבצים מהטלפון למחשב – גם משתמש פרטי וגם משתמש עסקי.

פירוט המטרות המרכזיות של המערכת

המטרות המרכזיות של פרויקט זה הם:

- יצירת חיבור בין המחשב לטלפון בדרך שבחר הלקוח.
- בחירת הקבצים והמיקומים אליהם הם יגיעו במכשיר השני
- העברת הקובץ/קבצים למיקום המתאים במחשב

הבעיה

הבעיה היא שכרגע אין דרך טובה להעביר קבצים בין טלפון ומחשב ודרכים שכן יש מסורבלות ולא יעילות. העברת קבצים בין מכשירי סלולר למחשב מתבצעת כיום בעיקר דרך חיבור לאפליקציות כגון מייל וווטסאפ, שלהם יש להירשם מראש וכן וישנן מגבלות העברה שונות אחרות.

תועלת מהמערכת

התועלת שהמערכת תביא זה ממשק נוח להעברת קבצים בין הטלפון והמחשב, דבר שיחסוך זמן, ימנע את הצורך להירשם מראש לאפליקציות, וימנע טעויות (שכן הפתרון שלי גם מגדיר את מיקום העברת הקבצים)

שירותים שהמערכת תיתן

השירותים שהמערכת תיתן הם:

- תוכל להעביר את הקבצים בצורה דו כיוונית, כלומר שהטלפון יכול להעביר למחשב קבצים אבל גם המחשב יכול להעביר לטלפון.
- תוכל להעביר מספר קבצים בו זמנית
- אצל המכשיר שמקבל את הקבצים תהיה אפשרות בחירה של המיקום שאליו הקבצים יגיעו.
- במכשיר ששולח את הקבצים תהיה אפשרות לבחור מספר קבצים שונים להעביר למחשב השני.

השוואת העבודה עם פתרונות קיימים

הפתרונות שקיימים היום להעברת קבצים בין טלפון למחשב הם מאוד מסורבלים ולא נוחים למשתמש. פתרון אחד לדוגמא הוא להעביר ווטסאפ או במייל אבל שניהם פתרונות לא נוחים כי צריך להתחבר למייל/ ווטסאפ להוריד את הקובץ, לשמור אותו וזה פשוט לא תהליך נוח ויעיל בעיקר כשאתה רוצה להעביר מספר קבצים.

סקירת טכנולוגיית הפרויקט

הטכנולוגיה שאני יעבוד איתה תהיה בעיקר עבודה עם קוד QR, עם WIFI, ועבודה עם Python כדי ליצור את תוכנת האנדרואיד ואת תוכנת המחשב.

היו מעט קשיים בהגדרת המערכת בגלל חוסר הכרה של הנושא של QR קוד ושל הצפנה דרכו.

הגבלות בפרויקט

לא נדרש ציוד מיוחד לפרויקט זה.

יכולות לצוץ הגבלות בהעברת הקבצים מטלפון למחשב בגלל השוני של מחשבים ומכשירים סלולריים בנושא רשתות.

תיחום הפרויקט

פרויקט זה הוא בתחום העברת הקבצים, ובפרט, העברת קבצים בין מכשיר אנדרואיד למחשב.

פירוט תיאור המערכת

תיאור מפורט של המערכת:

המערכת מורכבת משני מכשירים, מכשיר האנדרואיד והמחשב.

אפליקציית המחשב יוצרת QR ואפליקציית האנדרואיד פותחת סורק QR ואז נוצר חיבור דרך WIFI.

אחרי שהחיבור נוצר יהיה אפשרות של בחירה של קבצים להעביר בשני האפליקציות. ברגע שמישהו בחר קבצים להעברה על האפליקציה השנייה יופיעה מסך שמבקש מהמשתמש להכניס את המיקום שהוא רוצה שהקבצים יגיעו.

לאחר מכן תהיה שליחה של הקבצים ואז תוכל לבחור להמשיך להעביר קבצים או לסגור את הקישור.

פירוט היכולות שהמערכת תיתן למשתמש להשתמש בו:

מספר	יכולת	תיאור
1	יכולת התחברות דרך WIFI	ביכולת זו המחשב יצור קוד QR המכיל את כל הפריטים הדרושים להתחברות והטלפון יפתח סורק QR ויסרוק את הקוד. לאחר מכן תקרה התחברות דרך WIFI
2	בחירת קבצים מתוך רשימת קבצים	למכשירים תהיה אפשרות לבחור את הקבצים להעברה מתוך רשימת קבצים.
3	בחירת מיקום של קובץ/קבצים	למכשירים תהיה אפשרות לבחור מיקום במערכת הקבצים שלהם שאליו הקבצים יגיעו.
4	שליחת הקבצים	שני המכשירים יוכלו להעביר את הקבצים שהם בחרו למיקום שנבחרו.
5	שליחת רשימת הקבצים להעביר	לשני המכשירים תהיה את האפשרות לשלוח את רשימת הקבצים שהם רוצים להעביר למכשיר השני כדי שיוכל לבחור להם מיקומים

פירוט הבדיקות שיתבצעו על ידי המערכת:

מספר	שם בדיקה	מה אמורה לבדוק	איך הבדיקה תעבוד
1	קובץ קיים	בדיקה זו אמורה לוודא שהקובץ שאני רוצה להעביר קיים	אני אנסה לגשת לקובץ ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש
2	חיבור קיים	בדיקה זו אמורה לוודא שיש חיבור בין המכשירים	אני אנסה לשלוח משהו למכשיר השני ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש
3	מיקום קיים	בדיקה שהמיקום שאני רוצה שהקבצים יגיעו אליו קיים	אני אנסה לשמור את הקובץ במיקום הזה ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש
4	הקבצים הועברו בהצלחה	בדיקה האם כל הקבצים הועברו בהצלחה למכשיר השני	אשווה את הקבצים שקיבלתי עם רשימת הקבצים שקיבלתי מהמכשיר השני

5	המכשיר השני מוכן	בדיקה זו תראה האם המכשיר השני מוכן להעברת הקבצים	אשלח הודעה ממכשיר אחד שהוא מוכן לקבל את הקבצים ואחכה להודעה מהמכשיר השני
6	התחברות נכונה	בדיקה זו תוודא שכל סוקט שנפתח מתחבר לסוקט שנפתח במכשיר השני	כל פעם שאני פותח סוקט האזנה אני אשלח הודעה ואחכה שאני מקבל הודעה מהמכשיר השני שהוא פתח סוקט התחברות.

תכנון לוח זמנים לפרויקט

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
מסמך ייזום	25.9.22	6.10.22	25.9.22	10.10.22	
פירוט יכולות	29.11.22	5.12.22	30.11.22	4.12.22	
מסמך אפיון	15.12.22	28.12.22	16.12.22	11.1.23	
מסמך עיצוב	16.1.23	10.3.23	27.1.23	22.3.23	
התחברות דרך WIFI	13.10.22	16.10.22	15.10.23	17.10.23	
הצפנה דרך קוד QR	13.10.22	16.10.22	13.10.22	13.10.22	
פתיחת מצלמה ופיענוח QR	13.10.22	16.10.22	15.10.22	18.10.22	
בחירת קובץ	23.12.22	29.12.22	20.12.22	28.12.22	
העברת רשימת קבצים	4.1.23	10.1.23	5.1.23	9.1.23	
בחירת מיקום	12.2.23	18.2.23	16.2.23	22.2.23	
העברת קבצים	10.3.23	27.3.23	15.3.23	10.4.23	
הצפנה של התקשורת	23.3.23	24.3.23	3.4.23	3.4.23	

ניהול סיכונים בפרויקט:

הסיכון	פירוט הסיכון	רמת הסיכון	תיאור דרכים להתמודדות עם הסיכון ולהקטין	מה בוצע בפועל	תאריך
אי הצלחה של שילוב GUI	שאני לא אצליח ללמוד ולשלב את הGUI בפרויקט שלי בזמן	בינונית	ללמוד באינטרנט איך הספריית GUI שאני עובד איתה עובדת וללמוד אותה (pyqt5)	הצלחתי ללמוד את הספרייה ולשלב אותה בקוד	15.12.22
לא להצליח לשלב את פייטון באנדרואיד	לא להצליח להריץ את הקוד המתאים או לייבא את הספריות לטלפון	בינונית	ללמוד דרכים להרצות קוד באינטרנט ולעבוד עם ספריות שמותאמות גם לאנדרואיד	מצאתי אפליקציה לאנדרואיד בשם pydroid שמאפשרת להריץ פייתון על מכשיר אנדוראיד	12.10.22
לא להצליח להצפין את המידע	שאני לא אצליח להצפין את המידע המועבר ולפענח אותו כראוי	בינונית	ללמוד על הצפנות ועל פיענוחים	אני משתמש בהצפנת fernet בפרויקט שלי להצפין את התקשורת	20.2.22
לא להצליח להעביר את הקבצים	שאני לא אצליח להעביר את הקבצים בין הטלפון והמחשב	בינונית	לפני שאני רושם את כל הפרויקט לבדוק האם בכלל ניתן להעביר קבצים בין טלפון והמחשב	ניסיתי להעביר קובץ בין הטלפון והמחשב והצלחתי	14.10.22

פירוט יכולות

שם יכולת	יכולת התחברות דרך WIFI
מהות	ביכולת זו המחשב יצור קוד QR המכיל את כל הפריטים הדרושים להתחברות והטלפון יפתח סורק QR ויסרוק את הקוד. לאחר מכן תקרה התחברות דרך WIFI.
אוסף פעולות למימוש היכולת	• כתיבת אפליקציית מחשב ואפליקציית טלפון עם תמיכה בבחירה של יכולת זו • קוד באפליקציית המחשב שמאפשר לי ליצור קוד QR ולהצפין בו את כל המידע לצורך חיבור זה • קוד באפליקציית האנדרואיד שיכול לקרוא את הקוד QR • קוד שיודע ליצור חיבור WIFI • קוד שיודע לקבל חיבור WIFI מהמכשיר השני ולאשר אותו
אובייקטים נחוצים	• קישור לרשת WIFI הפנימית • כרטיס רשת • מכשיר ששולח קישור • מכשיר שמקבל קישור • אלגוריתם חיבור • מצלמה לסריקת הקוד QR

שם יכולת	בחירת קבצים מתוך רשימת קבצים
מהות	למכשיר ששולח את הקבצים תהיה אפשרות לבחור את הקבצים להעברה מתוך רשימת קבצים.
אוסף פעולות למימוש היכולת	• שליפת כל הקבצים במכשיר השולח • הצגת הקבצים למשתמש בדרך נוחה • קוד שמאפשר בחירת קובץ (או קבצים) ספציפי
אובייקטים נחוצים	• גישה לקבצים במחשב • משתמש שיבחר את הקבצים

שם יכולת	בחירת מיקום של קובץ/קבצים
מהות	למכשיר שמקבל את הקבצים תהיה אפשרות לבחור את המיקום במערכת הקבצים שהוא רוצה שהקבצים יגיעו אליו.
אוסף פעולות למימוש היכולת	• הצגת כל הקבצים בצורה נוחה • אפשרות לבחירת תיקייה שתהיה המיקום אליו אני מכניס את הקבצים

	• קוד ששומר את התיקייה שאני רוצה לשמור בה את הקובץ/קבצים ושומר אותו בקוד האפליקציה
אובייקטים נחוצים	• גישה לקבצים במחשב • משתמש שיבחר את המיקום של הקבצים

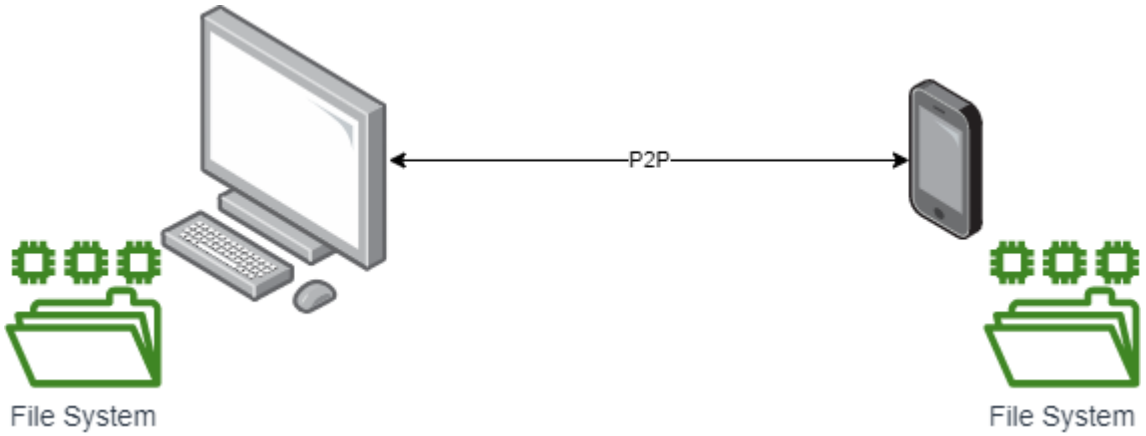
שם יכולת	שליחת הקבצים
מהות	תהיה אפשרות להעביר את הקבצים בין מכשיר אחד לשני, בין אם זה האנדרואיד למחשב או הפוך.
אוסף פעולות למימוש היכולת	• שליפת כל הקבצים במכשיר השולח • פתיחת סוקטים עבור כל קובץ • העברת חלקים של הקובץ עד שכולו מועבר • למקם כל קובץ במיקום שהוגדר לו מראש
אובייקטים נחוצים	• גישה לקבצים במחשב • חיבור בין המכשירים • כרטיס רשת • מכשיר ששולח את הקבצים • מכשיר שמקבל את הקבצים

שם יכולת	שליחת רשימת הקבצים להעביר
מהות	תהיה לכל מכשיר את האפשרות להעביר למכשיר השני את רשימת הקבצים שהוא רוצה להעביר אליו
אוסף פעולות למימוש היכולת	• שליפת כל הקבצים במכשיר השולח • ארגון כל הקבצי ברשימה • העברת הרשימה למכשיר השני • קבלת הרשימה והכנסת כל קובץ למסך הבא בו יבחרו להם מיקומים
אובייקטים נחוצים	• גישה לקבצים במחשב • גישה לאיזה קבצים המשתמש בחר • חיבור בין המכשירים • כרטיס רשת • מכשיר ששולח את הרשימה • מכשיר שמקבל את הרשימה

מסמך עיצוב

תיאור הארכיטקטורה של המערכת המוצעת:

המערכת מורכבת משני מכשירים – מחשב וטלפון שמתקשרים ביניהם כאשר התקשורת ביניהם תתבצע P2P – כאשר שני המכשירים מתנהגים גם כClients וגם כServers.

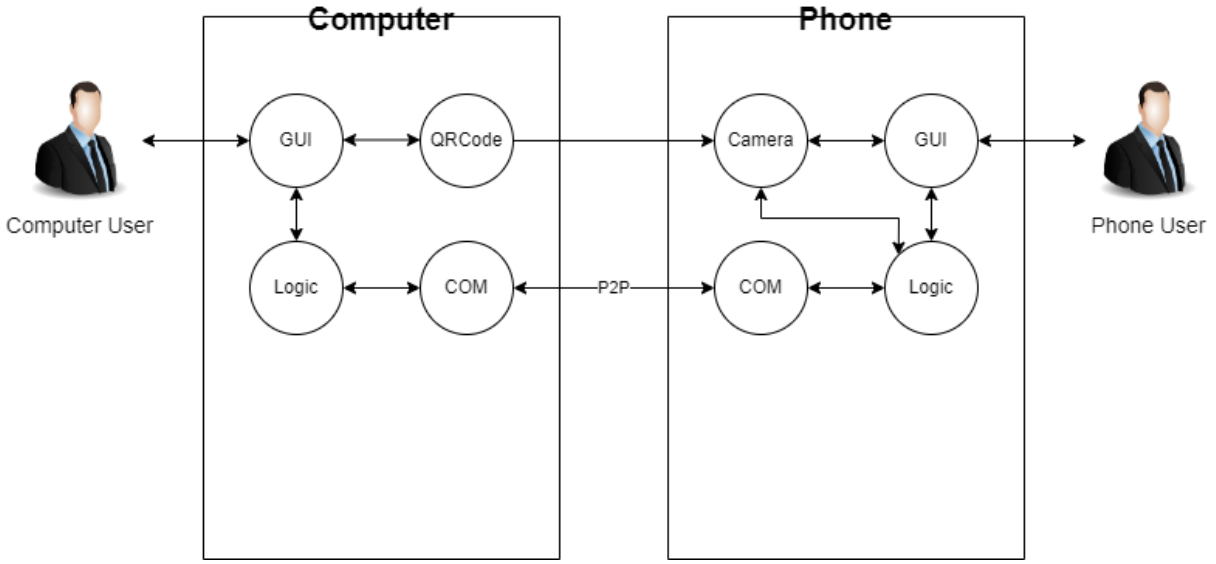


תיאור טכנולוגיה רלוונטית:

1. מערכת הפעלה – Windows 11
2. מערכת הפעלה – Android 8.0
3. שפת התכנות python 3.9
4. IDE – Pydroid 3
5. רשת תקשורת משותפת למחשב ולטלפון
6. ארכיטקטורת P2P

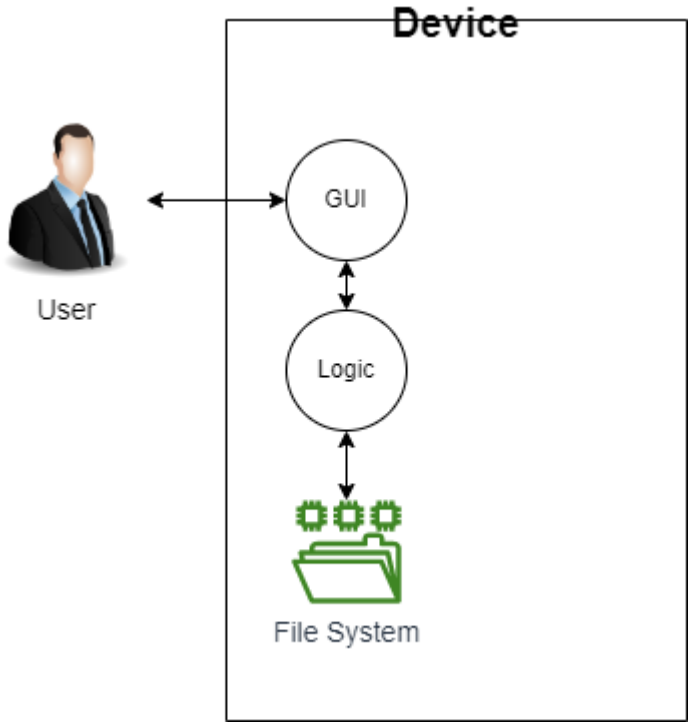
תיאור זרימת המידע במערכת:

התחברות דרך WIFI:



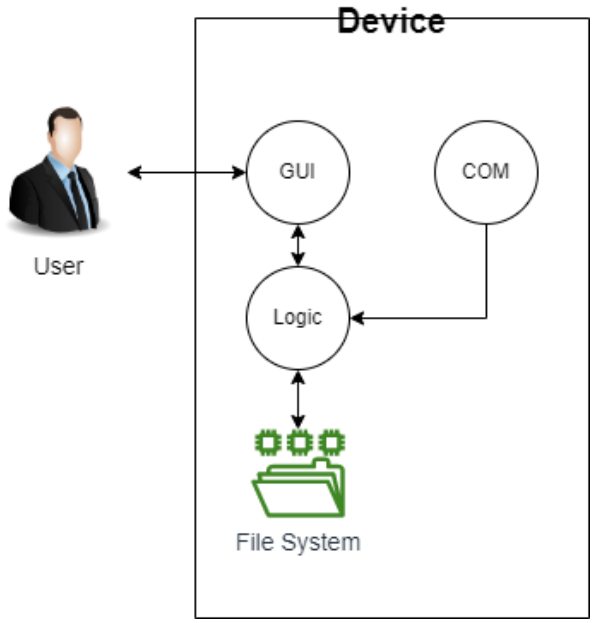
בתרשים זה ניתן לראות שבתהליך ההתחברות דרך WIFI יש למשתמש בטלפון אינטראקציה עם הGUI של הטלפון, שעליו מוצגת המצלמה והיא קוראת את הQR קוד שמפיק המחשב ונוצר תקשורת P2P ביניהם.

בחירת קבצים מתוך רשימת קבצים:



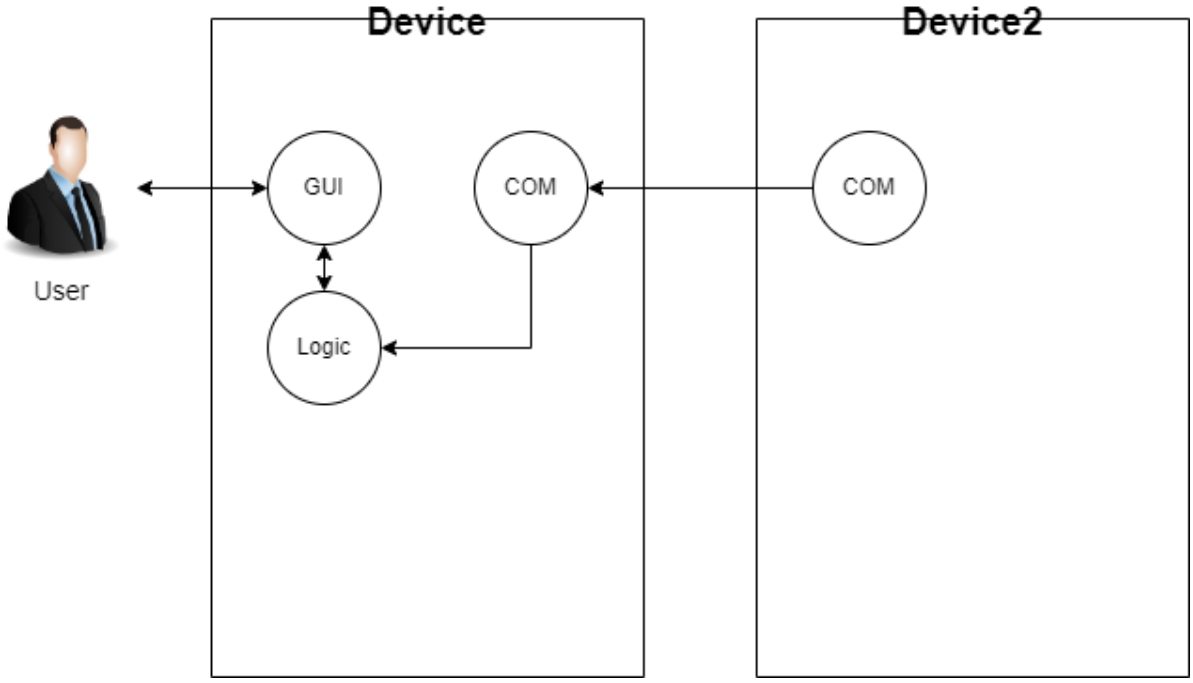
ניתן לראות בתרשים זה למשתמש של מכשיר מסוים יש אינטראקציה עם הGUI של המכשיר הזה והתוכנה, לפי מה שהמשתמש אומר לה דרך הGUI, בוחרת קובץ מתוך מערכת הקבצים של המכשיר.

בחירת מיקום לקבצים של המכשיר השני:



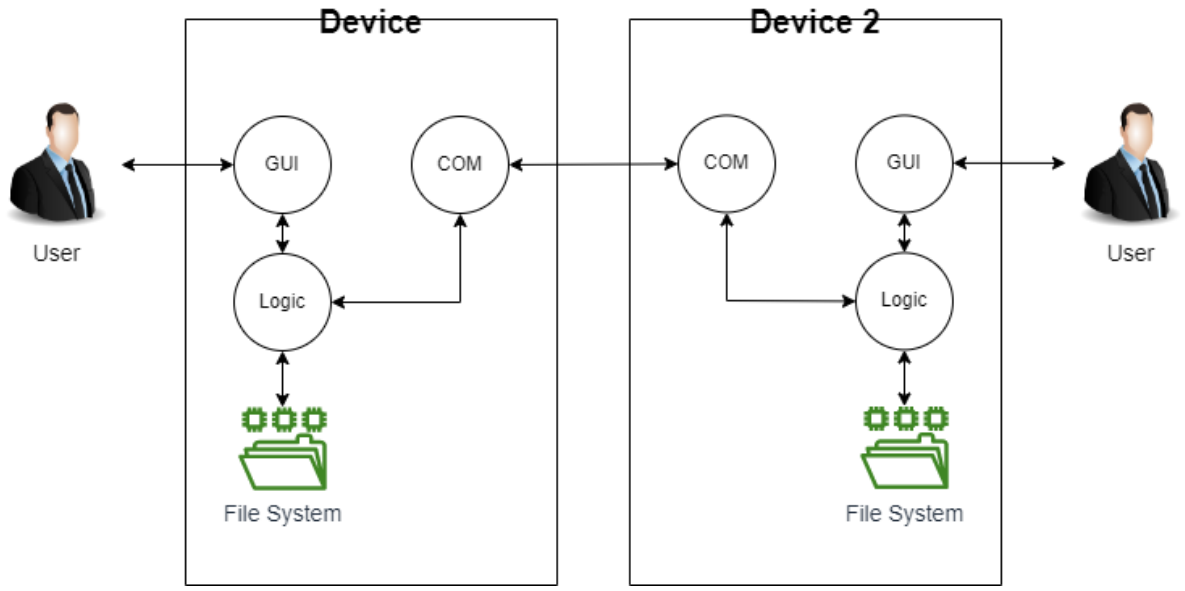
ניתן לראות בתרשים זה שבין המשתמש של אחד המכשירים לGUI של מכשיר זה יש אינטראקציה והוא בוחר לכל קובץ שקיבל מהמכשיר השני, מקום במערכת הקבצים שלו.

העברת רשימת הקבצים להעביר



בתרשים ניתן לראות שהמכשיר השני מעביר למכשיר הראשון את רשימת הקבצים. לאחר מכן, המכשיר הראשון מקבלת את הרשימה ומעדכן את הGUI לפי הקבצים שקיבל מהמכשיר השני

<u>שליחת הקבצים</u>



לפי התרשים, המכשירים מתקשרים אחד עם השני ומעבירים את הקבצים. לאחר מכן הם מועברים למקום המתאים במערכות הקבצים של המכשירים ולבסוף המכשירים מעדכנים את הGUI שההעברה עברה בהצלחה או שהיא נכשלה.

תיאור האלגוריתמים המרכזיים בפרויקט:

התחברות דרך WIFI:

חיבור שני המכשירים (המחשב והטלפון) דרך WIFI (הם צריכים להיות מחוברים לאותה רשת WIFI).

פתרונות אפשריים:

- התחברות עם Qr code: הטלפון סורק QR code שמציג המחשב בו נמצאים הפרטים (IP וPORT של המחשב) בשביל לפתוח סוקט בין הטלפון והמחשב.
- רשימה של כל המכשירים המחוברים לרשת WIFI: המחשב יציג למשתמש רשימה של כל מי שמחובר לרשת הWIFI ואז המשתמש יבחר מהרשימה הזו את הטלפון.

בחרתי את הפתרון של התחברות עם Qr code כי ברשימה של מכשירים יכול להיות שני מכשירים עם אותו שם ואז המחשב יכול בטעות לנסות להתחבר למכשיר לא נכון.

תיאור מילולי של האלגוריתם:

מחשב:

1. הצפנת IP וPORT לתוך QR Code
2. פתיחת סוקט מאזין על הPORT והIP העלה
3. מחכה להתחברות של הטלפון

טלפון:

1. פתיחת מצלמה והרצת עד לגילוי של Code QR
2. פתיחת סוקט והתחברות לIP וPORT שהיו בQR code

בחירת קבצים מתוך רשימת קבצים:

מכשיר מסוים בוחר את הקבצים שהוא רוצה להעביר למשתמש השני.

פתרונות אפשריים:

- עבודה עם QFileDialog של Pyqt5 בשביל לפתוח את מערכת הקבצים בצורה גרפית ואז המשתמש בוחר את הקבצים שהוא רוצה.
- פתיחת הFile Explorer בשביל לפתוח את מערכת הקבצים בצורה גרפית ואז המשתמש בוחר את הקבצים שהוא רוצה.

בחרתי להשתמש בQFileDialog כי הוא של Pyqt5 שזה גם איך שאני ממש GUI והQFileDialog משתלב יותר טוב עם הGUI שלי.

תיאור מילולי של האלגוריתם:

מכשיר:

1. הצגת המסך
2. פתיחה של הQFileDialog
3. לקיחת המיקום של הקובץ הנבחר והוספה שלו לרשימה של מיקומים

4. הוספת הקובץ למסך

<u>בחירת מיקום לקבצים ששלח המכשיר השני:</u>

מכשיר מסוים מקבל רשימת קבצים מהמכשיר השני ובוחר להם את המיקום.

פתרונות אפשריים:

- עבודה עם QFileDialog של Pyqt5 בשביל לפתוח את מערכת הקבצים בצורה גרפית ואז המשתמש בוחר את הקבצים שהוא רוצה.
- פתיחת הFile Explorer בשביל לפתוח את מערכת הקבצים בצורה גרפית ואז המשתמש בוחר את הקבצים שהוא רוצה.
- המשתמש פשוט רושם מיקום מסוים מבלי לפתוח את מערכת הקבצים

בחרתי להשתמש בQFileDialog כי הוא של Pyqt5 שזה גם איך שאני ממש GUI והQFileDialog משתלב יותר טוב עם הGUI שלי, ובחירת מיקום מסוים מבלי לפתוח את מערכת הקבצים יכולה לגרום לזה שהמשתמש יבחר מיקום לא קיים.

תיאור מילולי של האלגוריתם:

מכשיר:

1. קבלת רשימת הקבצים מהמכשיר השני
2. הצגה שלהם על המסך
3. פתיחה של הQFileDialog
4. לקיחת המיקום של התיקייה שנבחרה והוספה שלה למילון של מיקומים וקבצים

<u>שליחת רשימת הקבצים להעביר:</u>

תהיה לכל מכשיר את האפשרות להעביר למכשיר השני את רשימת הקבצים שהוא רוצה להעביר אליו

פתרונות אפשריים:

- להעביר את כל אחד מהשמות של הקבצים בנפרד
- להעביר את השמות של הקבצים כרשימה

בחרתי להעביר את השמות של הקבצים כרשימה פשוט כי זה יותר נוח.

תיאור מילולי של האלגוריתם:

מכשיר 1:

1. איסוף כל השמות של הקבצים מהמסך
2. שליחת השמות כרשימה

מכשיר 2:

1. קבלה של הרשימה
2. עדכון של הGUI בהתאם

<u>שליחת הקבצים</u>

תהיה אפשרות להעביר את הקבצים בין מכשיר אחד לשני, בין אם זה האנדרואיד למחשב או הפוך.

פתרונות אפשריים:

- שליחת כל הקבצים דרך הסוקטים הקיימים
- עבור כל קובץ לפתוח סוקט בשביל העברת הקובץ והעברית כל הקבצים במקביל

בחרתי באפשרות של לפתוח סוקט עבור כל קובץ כי אמנם זה קצת יותר בזבזני, אבל ההעברה תהיה הרבה יותר מהירה.

תיאור מילולי של האלגוריתם

מכשיר 1:

1. עבור כל קובץ שאמור להגיע למכשיר:
   1. פותח סוקט האזנה על thread חדש (הוא מקבל את הקובץ במקביל למה שקורה)
   2. שולח הודעה שהסוקט נפתח
   3. מחכה לקבל אישור שהמכשיר השני פתח סוקט התחברות
2. עבור כל קובץ שצריך להעביר למכשיר השני:
   1. מחכה לקבל אישור שנפתח סוקט האזנה במכשיר השני
   2. פותח סוקט התחברות על thread חדש (הוא שולח את הקובץ במקביל למה שקורה)
   3. שולח הודעה שהסוקט נפתח

מכשיר 2:

1. עבור כל קובץ שצריך להעביר למכשיר השני:
   1. מחכה לקבל אישור שנפתח סוקט האזנה במכשיר השני
   2. פותח סוקט התחברות על thread חדש (הוא שולח את הקובץ במקביל למה שקורה)
   3. שולח הודעה שהסוקט נפתח
2. עבור כל קובץ שאמור להגיע למכשיר:
   1. פותח סוקט האזנה על thread חדש (הוא מקבל את הקובץ במקביל למה שקורה)
   2. שולח הודעה שהסוקט נפתח
   3. מחכה לקבל אישור שהמכשיר השני פתח סוקט התחברות

תיאור סביבת הפיתוח:

הפרויקט נכתב בpython 3.9. במחשב פיתחתי את הפרויקט בPycharm ובטלפון השתמשתי בIDE Pydroid. בטלפון אני משתמש במערכת ההפעלה אנדרואיד שהיא מערכת הפעלה חינמית בקוד פתוח המיועדת בעיקר לטלפון חכמים ומבוססת על לינוקס.

לבדיקה אזדקק למחשב וטלפון בהם אוכל להריץ את הקוד, גישה למערכות הקבצים של הטלפון והמחשב האלו וWireshark בשביל לבדוק את התקשורת בין שני המכשירים.

תיאור פרוטוקול התקשורת:

פירוט תהליך התקשורת

מחשב:

1. פתיחת סוקט האזנה
2. אישור התחברות של הטלפון
3. פתיחת סוקט התחברות לטלפון
4. שליחת רשימת הקבצים להעביר
5. קבלה של הקבצים שהטלפון רוצה להעביר
6. שליחה שהמכשיר מוכן לקבל את הקבצים
7. קבלת הודעה שהטלפון מוכן לקבצים
8. עבור כל קובץ שצריך לשלוח:
    1. מחכה לקבלת הודעה שנפתח סוקט האזנה בטלפון
    2. פתיחת סוקט התחברות
    3. שליחה הודעה שהסוקט התחבר
9. עבור כל קובץ שצריך לקבל:
    1. פותח סוקט האזנה
    2. שולח שנפתח סוקט האזנה
    3. מחכה לקבלת הודעה שהסוקט בטלפון התחבר

טלפון:

1. פתיחת סוקט התחברות למחשב
2. פתיחת סוקט האזנה
3. אישור התחברות של המחשב
4. שליחת רשימת הקבצים להעביר
5. קבלה של הקבצים שהמחשב רוצה להעביר
6. שליחה שהמכשיר מוכן לקבל את הקבצים
7. קבלת הודעה שהמחשב מוכן לקבצים
8. עבור כל קובץ שצריך לקבל:
    1. פותח סוקט האזנה
    2. שולח שנפתח סוקט האזנה
    3. מחכה לקבלת הודעה שהסוקט במחשב התחבר
9. עבור כל קובץ שצריך לשלוח:
    1. מחכה לקבלת הודעה שנפתח סוקט האזנה בטלפון
    2. פתיחת סוקט התחברות
    3. שליחה הודעה שהסוקט התחבר

תיאור מסכי המערכת:

מסך QR code:

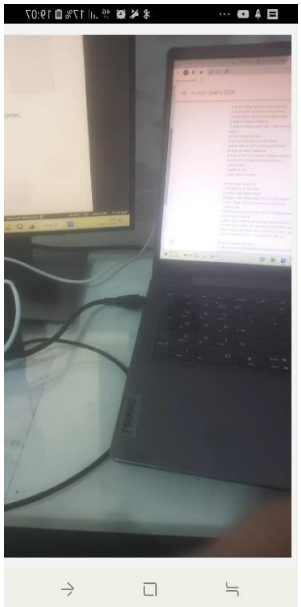
מסך זה נמצא רק במחשב והוא מכיל את הIP והPORT של המחשב מוצפן לתוך QR code.

תמונת מסך:

מסך סורק QR code:

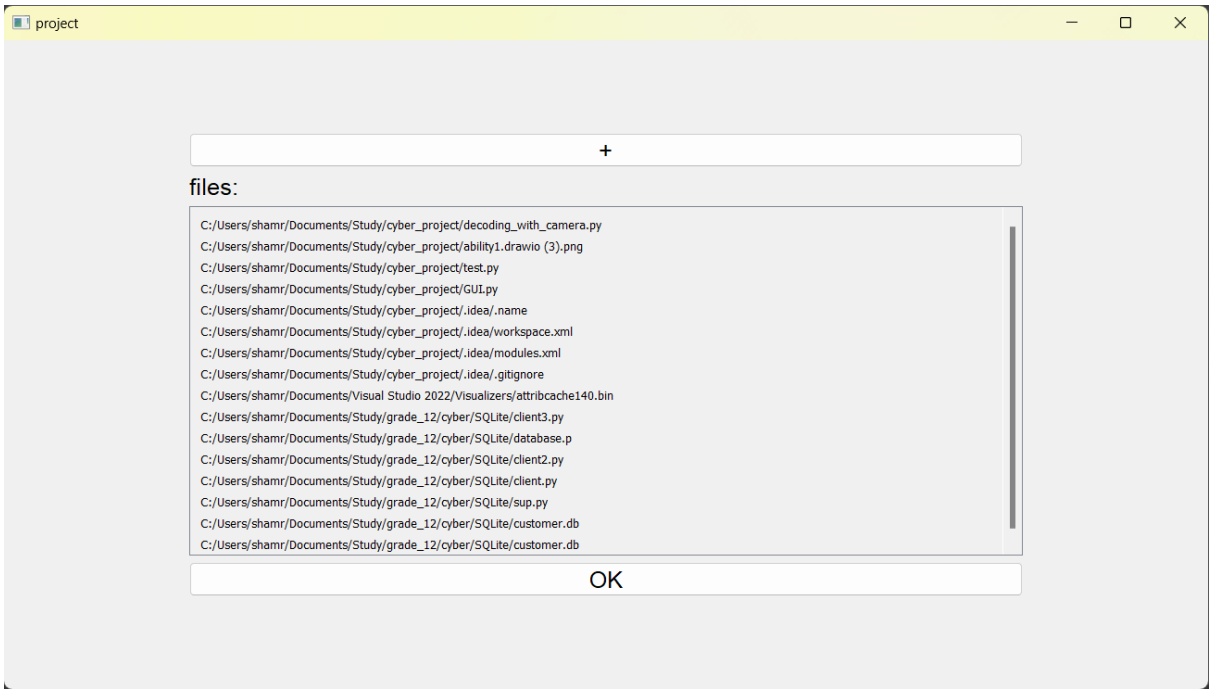
מסך זה נמצא רק בטלפון והוא בעצם סורק QR code שמציג המחשב.

תמונת מסך:

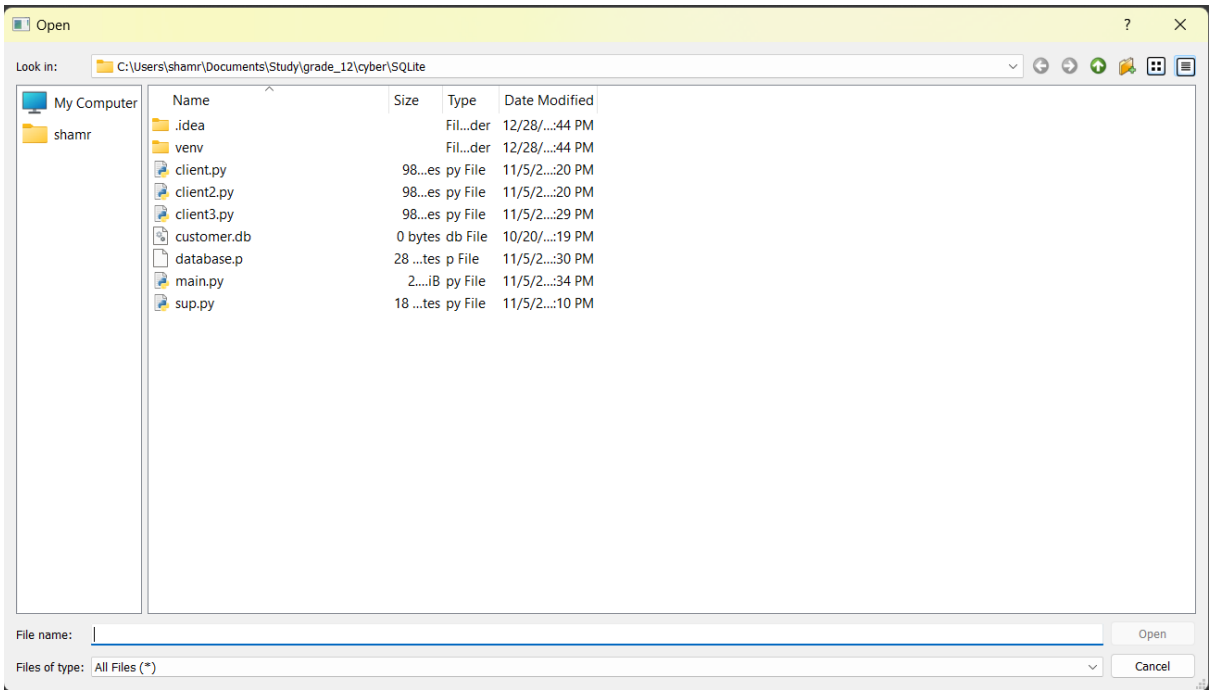


מסך בחירת הקבצים:

מסך זה נמצא במחשב ובטלפון והוא מאפשר לך להוסיף קבצים לרשימת קבצים שאתה רוצה להעביר למכשיר השני

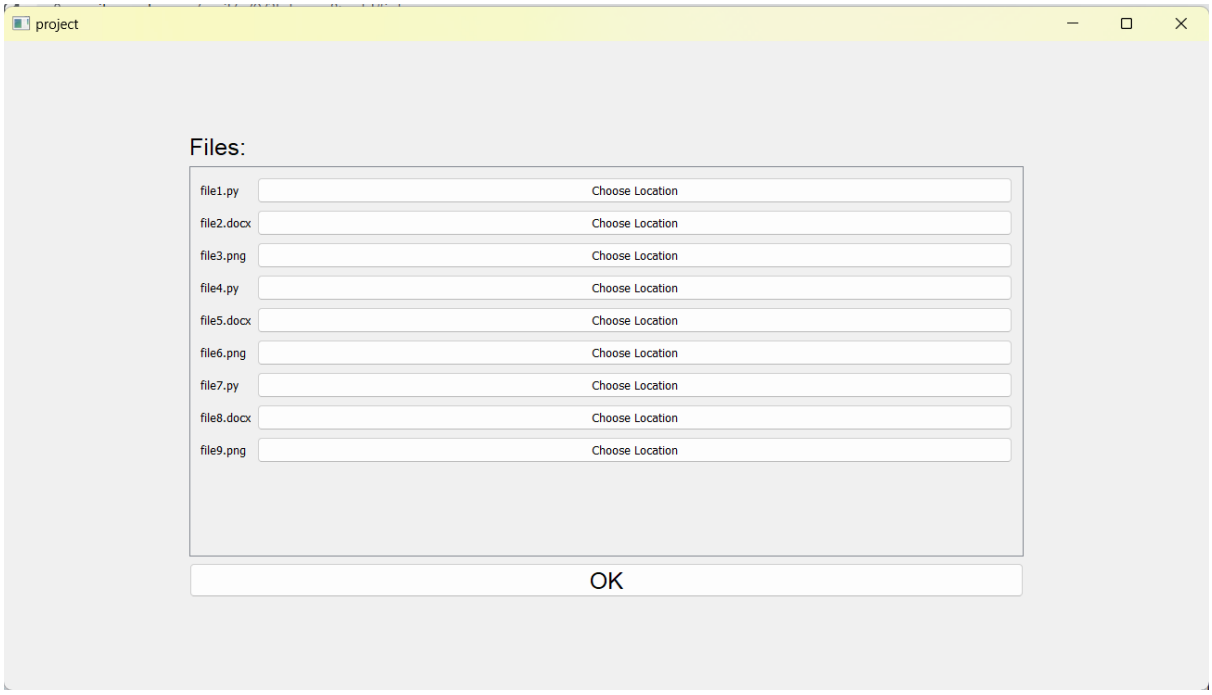


וזה המסך שאתה רואה כשאתה לוחץ על ה"+" (פתיחת מערכת הקבצים):

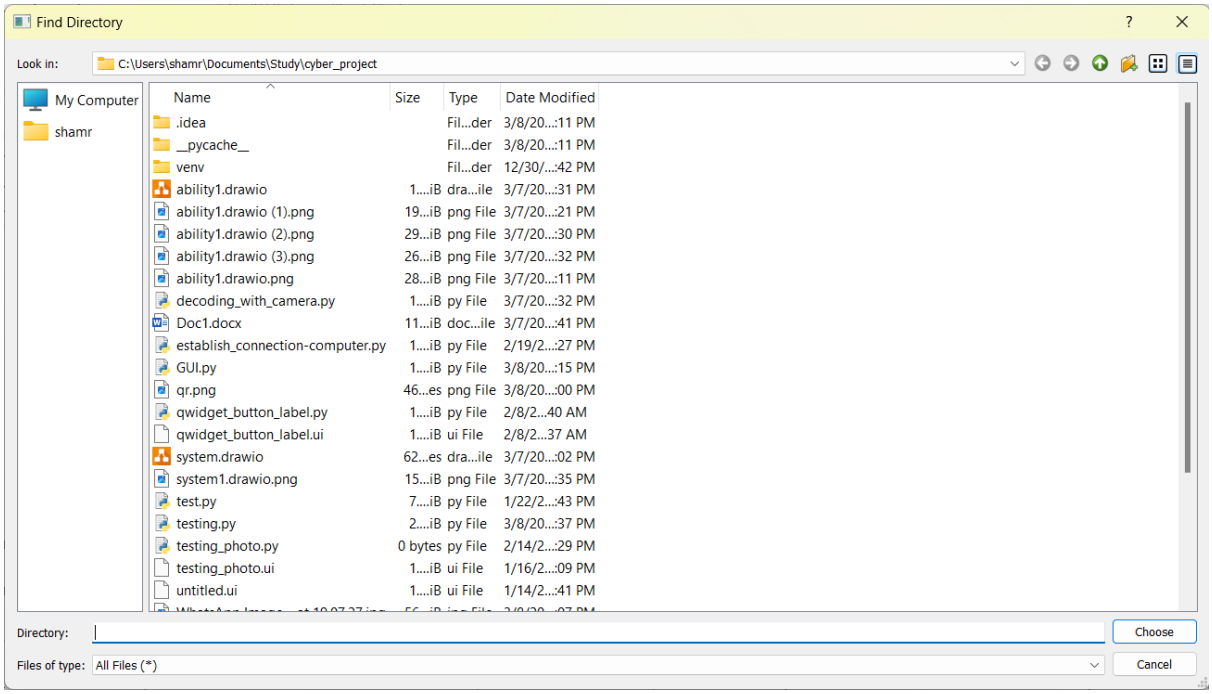


מסך בחירת מיקומים:

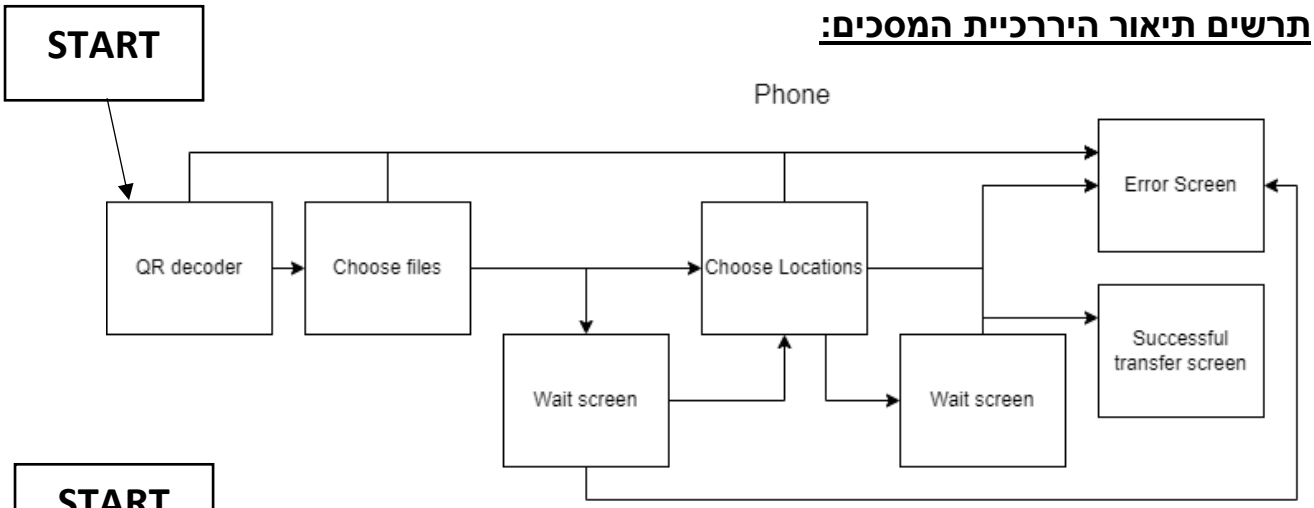
מסך זה נמצא במחשב ובטלפון והוא מאפשר לך לבחור מיקום לכל הקבצים שהמכשיר השני רוצה לשלוח

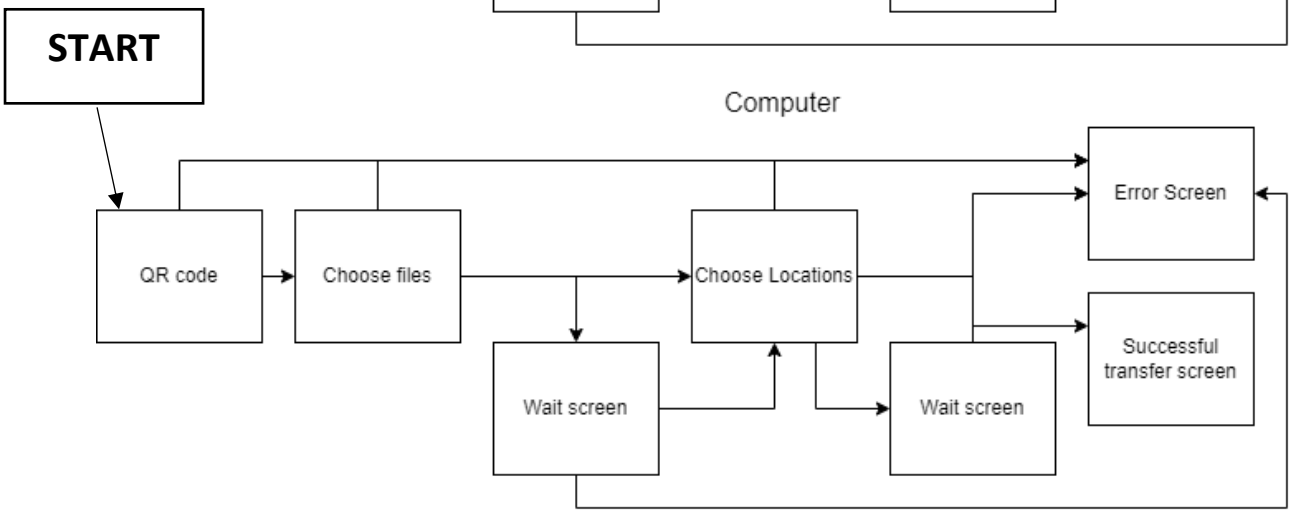


וזה המסך שאתה רואה כשאתה לוחץ על ה"Choose Location" (פתיחת מערכת הקבצים):



תרשים תיאור היררכיית המסכים:





תיאור מבני הנתונים:

מערכת הקבצים של המחשב ושל הטלפון

רשימות של קבצים להעביר

מילון שבו עבור כל קובץ שמכשיר העביר יש ערך של המיקום שנבחר עבור קובץ זה (אם לא נבחר עבורו מקום יופיעה None)

מפתח: שם של קובץ (string)	האם נבחר בשבילו מקום
File1.py	C:\
File2.docx	None

סקירת חולשות ואיומים:

בפרויקט אני מעביר קבצים ואיום שיכול להיות זה שמישהו יצליח להאזין לתקשורת של המכשירים ולקחת את הקובץ ולכן אני מצפין את הקובץ בעזרת אלגוריתם ההצפנה fernet. הצפנה מהסוג הזה מבטיחה שלא יהיה אפשר לפענח הודעה שהוצפנה בשיטה זו ללא המפתח והיא משתמשת באלגוריתם AES סימטרי.

בנוסף אני יוצר התחברות עם TCP sockets ואני עושה לחיצת יד משולשת.

מימוש הפרויקט

תיאור המודולים בהם נעשה שימוש:

MessageWindow	
מגדיר את המסך שמציג הודעות למשתמש (הודעות שגיאת, הצלחה וכו')	
מודולים מיובאים	
זה המודול שאני משתמש בו בשביל הGUI שלי	Pyqt5
פרמטרים	
ההודעה שאני רוצה להציג במסך	message
התווית שבה יש את הטקסט שאני צריך להציג על המסך	message_label
פעולות שלי	
יוצרת אובייקט של MessageWindow	__init__(self, message="")
פעולה זו "מכינה" את הUI של המסך	setup_ui(self)
זה פעולה שמחליפה את ההודעה שיש על המסך	change_message(self, message)

Window1	
זה מגדיר את המסך הראשון של המחשב בו מוצג הקוד QR	
מודולים מיובאים	
זה המודול שאני משתמש בו בשביל הGUI שלי	Pyqt5
פרמטרים	
זה פרמטר שמייצג את המיקום של התמונה שמכילה את הקוד QR	path
זה התווית שמכילה את התמונה בשביל שתוצג במסך	photo
פעולות שלי	
פעולה שמאתחלת אובייקט של המסך הראשון	__init__(self, path)
פעולה ש"מכינה" את הUI של המסך	Window1_ui(self)

Window2	
זה מגדיר את המסך השני של המחשב בו בוחרים את הקבצים להעברה	
מודולים מיובאים	
זה המודול שאני משתמש בו בשביל הGUI שלי	Pyqt5
פרמטרים	
זה פרמטר שמאותת שסיימנו לבחור קבצים והוא מופעל על ידי כפתור הOK	finished_choosing_files
זה רשימה שמכילה את כל הקבצים שנבחרו על ידי המשתמש	Files
פרמטר זה מכיל את כל הרכיבים שנמצאים ב vertical layout	vertical_layout_widget
זה פרמטר שמכיל את הvertical layout עצמו	vertical_layout

push_button	זה פרמטר שמכיל את הכפתור שלוחצים עליו בשביל לפתוח את מערכת הקבצים ולבחור קובץ
title_label	זה פרמטר שמכיל את הכותרת של המסך
scroll_area	זה פרמטר שמכיל את האזור בו אפשר לגלול למעלה ולמטה
scroll_area_widget_contents	זה פרמטר שמכיל את כל הרכיבים שבתוך אזור הגלילה
ok_button	זה הכפתור שכלוחצים עליו כשמסיימים לבחור את הקבצים
פעולות שלי	
__init__(self)	פעולה שמאתחלת אובייקט של המסך השני
setup_ui(self)	פעולה ש"מכינה" את הUן של המסך
add_file(self, file_path)	פעולה שמוסיפה שם של קובץ למסך
get_file_path(self)	פעולה שפותחת את מערכת הקבצים ומעבירה לפעולה add_file את המיקום של הקובץ שנבחר
ok_button_clicked(self)	פעולה ששולחת אות שבה יש את רשימת הקבצים שנבחרה על ידי המשתמש

Window3	
זה מגדיר את המסך השלישי של המחשב בו בוחרים מיקומים לקבצים שהועברו על ידי המכשיר השני	
מודולים מיובאים	
Pyqt5	זה המודול שאני משתמש בו בשביל הGUI שלי
פרמטרים	
all_files_have_location	זה פרמטר שמאותת שלכל הקבצים יש מיקום והוא מופעל על ידי כפתור הOK
file_location_dict	זה מילון בו המפתח הוא קובץ והערך הוא המיקום שלו
vertical_layout_widget	פרמטר זה מכיל את כל הרכיבים שנמצאים ב vertical layout
vertical_layout	זה פרמטר שמכיל את הvertical layout עצמו
title_label	זה פרמטר שמכיל את הכותרת של המסך
scroll_area	זה פרמטר שמכיל את האזור בו אפשר לגלול למעלה ולמטה
scroll_area_widget_contents	זה פרמטר שמכיל את כל הרכיבים שבתוך אזור הגלילה
ok_button	זה הכפתור שכלוחצים עליו כשמסיימים לבחור את הקבצים
פעולות שלי	
__init__(self)	פעולה שמאתחלת אובייקט של המסך השלישי
setup_ui(self)	פעולה ש"מכינה" את הUI של המסך
select_directory(self, file_name)	פעולה שפותחת את מערכת הקבצים בשביל לבחור תיקייה לשמור את הקובץ מסוים
check_all_files_have_location(self)	פעולה שבודת האם לכל הקבצים יש מיקום במערכת הקבצים

add_files(self, files)	פעולה שמאפשר להוסיף קבצים למסך (קבצים שצריך לבחור להם מיקום)
create_select_directory_function(self, label_text)	פעולת עזר בשביל הוספת הקבצים

MainWindow	
מגדיר את המסך המרכזי שאחראי הפיקוח ההחלפה בין המסכים השונים	
מודולים מיובאים	
window1	זה מודול שמכיל את המחלקה של Window1
window2	זה מודול שמכיל את המחלקה של Window2
window3	זה מודול שמכיל את המחלקה של Window3
message_win	זה מודול שמכיל את המחלקה של MessageWindow
פרמטרים	
path	זה פרמטר שמייצג את המיקום של התמונה שמכילה את הקוד QR
current_win	פרמטר שמכיל מספר המייצג איזה מסך מוצג עכשיו למשתמש
window1	פרמטר שמכיל את המסך הראשון
window2	פרמטר שמכיל את המסך השני
window3	פרמטר שמכיל את המסך השלישי
message_win	פרמטר שמכיל את המסך שמציג את ההודעות
stack	פרמטר שמכיל את כל המסכים ואיתו אפשר להעביר ביניהם
hbox	פרמטר שמכיל את הhorizontal layout
הפעולות שלי	
__init__(self)	פעולה שמאתחלת אובייקט של המסך הראשי
setup_ui(self)	פעולה ש"מכינה" את הUI של המסך
change_win(self)	פעולה שמעביר למסך הבא
change_to_message_win(self, message)	פעולה שמעבירה למסך שמציג את ההודעות

connection.py	
זה קובץ שמכיל את כל המחלקות שמשומשות לתקשורת	
מודולים מיובאים	
Pyqt5	זה המודול שאני משתמש בו בשביל הGUI שלי
socket	מודול זה יוצר עצמים מסוג Socket שאיתם ניתן לקיים תקשורת בין מכשירים
pickle	מודול בשביל להעביר אובייקטים מאובייקט לבינארי
os	מודול זה מאפשר עבודה מול מערכת ההפעלה
MainSendingSocket	
מגדיר את הסוקט המרכזי שאחראי על שליחה במערכת הP2P	
פרמטרים	
got_file_list	פרמטר זה הוא בשביל לאותת שהSocket קיבל את רשימת הקבצים

ready_to_send	פרמטר זה הוא בשביל לאותת שהמכשיר מוכן לשליחת הקבצים
send_massage	פרמטר זה הוא בשביל לאותת שצריך לשלוח הודעה
done_signal	פרמטר זה הוא בשביל לאותת שסיימנו עם העברת הקבצים
exception_rose	פרמטר זה הוא בשביל לאותת כשצצה שגיאה
ip	מכיל את הip שצריך להתחבר אליו
port	מכיל את הport שצריך להתחבר אליו
sending_socket	מכיל את הסוקט שבו שולחים את המידע
mutex	שני הפרמטרים האלו הם בשביל לעצור את הקוד עד שהוא מקבל איתות שאפשר להמשיך
condition	
done_condition	האם ההעברה של הקבצים נגמרה
files	רשימת הקבצים להעביר
message	הודעה להעביר
encrypting_object	אובייקט ההצפנה
BUFFER_SIZE	כמה בתים אפשר להעביר כל פעם דרך הסוקט
הפעולות שלי	
__init__(self)	פעולה שמאתחלת אובייקט של סוקט השליחה הראשי
run(self)	הפעולה שרצה על הthread הזה
connect_to_phone(self)	פעולה שמתחברת לטלפון
got_files(self, files)	הthread הזה מחכה עד שהמשתמש בוחר את הקבצים. פעולה זו ממשיכה את הthread ברגע שהוא בוחר אותם.
ready_to_send_files(self)	הthread הזה מחכה עד שהוא מקבל הודעה מהמכשיר השני שהוא מוכן לקבל את הקבצים
send(self, message)	פעולה זו היא בשביל לשלוח הודעה
done(self)	פעולה זו הופכת את הdone_condition לאמת
FileSendingSocket(MainSendingSocket)	
מגדיר את הסוקט עבור שליחת קובץ	
פרמטרים (בלי אלה של מחלקת האב)	
file_path	המיקום של הקובץ שצריך לשלוח
BUFFER_SIZE	כמה ביטים שלוחים בכל פעם
הפעולות שלי	
__init__(self, ip, port, file_path, key)	פעולה שמאתחלת אובייקט של סוקט שליחה של קובץ
run(self)	הפעולה שרצה על הthread הזה
MainReceivingSocket	
מגדיר את הסוקט המרכזי שאחראי על קבלה במערכת הP2P	
פרמטרים	
connection_made	פרמטר שמאותת האם חיבור נעשה
got_file_list_from_phone	פרמטר שמאותת האם קיבלנו את הקבצים מהטלפון
ready_for_files	פרמטר שמאותת האם אנחנו מוכנים לקבל את הקבצים

receive	פרמטר זה הוא בשביל לאותת שצריך לקבל הודעה
done_signal	פרמטר זה הוא בשביל לאותת שסיימנו עם העברת הקבצים
exception_rose	פרמטר זה הוא בשביל לאותת כשצצה שגיאה
ip	מייצג את הIP שצריך להקשיב עליו
port	מייצג את הPORT שצריך להקשיב עליו
done_condition	האם ההעברה של הקבצים נגמרה
receiving_socket	אובייקט הסוקט שמקבל את ההודעות
address	הכתובת של הסוקט שמתחבר
encrypting_object	אובייקט ההצפנה
BUFFER_SIZE	כמה בתים אפשר להעביר כל פעם דרך הסוקט
הפעולות שלי	
__init__(self)	פעולה שמאתחלת אובייקט של סוקט הקבלה הראשי
run(self)	הפעולה שרצה על הthread הזה
handle_connection(self)	פועלה שמטפלת בחיבור
handle_address(self)	פעולה שמטפלת בכתובת של הסוקט שמתחבר
done(self)	פעולה זו הופכת את הdone_condition לאמת
add_encrypting_object(self, key)	פעולה שמוסיפה את אובייקט ההצפנה לסוקט הקבלה הראשי (מופיעה רק בטלפון כי הוא מתחיל את סוקט הקבלה לפני שהוא מקבל את מפתח ההצפנה).
FileReceivingSocket(MainReceivingSocket)	
מגדיר את הסוקט עבור קבלת קובץ	
פרמטרים (בלי אלה של מחלקת האב)	
file	
files_and_paths	זה מילון בו יש את כל הקבצים שצריך לקבל ואת המיקומים שהם אמורים להגיע אליהם.
finished	זה משתנה שאומר האם הסוקט סיים את פעולתו.
הפעולות שלי	
__init__(self, ip, port, files_and_paths, key)	פעולה שמאתחלת אובייקט של סוקט קבלה של קובץ
run(self)	הפעולה שרצה על הthread הזה

Project	
זה המחלקה שמנהלת גם את החלק הגרפי וגם את התקשורת ביחד.	
מודולים מיובאים	
time	מודול שיש בו פעולות הקשורות לזמן כמו לחכות מספר שניות
cryptography	מודול בשביל להצפין ולפענח
gui	זה המודול של המסך הראשי שאחראי לפקח על כל הGUI
connection	זה המודול שיש בו את כל המחלקות שקשורות לתקשורת
qrcode	מודול בשביל ליצור קודי QR

threading	מודול בשביל ליצור עוד threads בשביל להריץ מספר דברים במקביל
sys	במודול זה יש מספר פונקציות ספציפיות בשביל לתקשר עם מערכת ההפעלה
פרמטרים	
key	מפתח ההצפנה
ip	מייצג את הIP שלי
port	מייצג את הPORT אני מאזין אליו
phone_port	מייצג את הPORT של המכשיר השני שאליו אני מתחבר
port_for_files	מייצג את הפורט בו אני מעביר את הקבצים (המספר הזה ישתנה כי אני פותח לכל קובץ סוקט)
path	המיקום של תמונת קוד הQR
qr_image	התמונה עצמה
main_window	אובייקט של המסך הראשי שאחראי על הGUI
main_receiving_socket	אובייקט של סוקט הקבלה הראשי
got_files	משתנה שמכיל האם קיבלתי את רשימת הקבצים מהמכשיר השני
finished_window2	משתנה שמכיל האם סיימתי לבחור את כל הקבצים
ready_for_the_files	משתנה שמכיל האם אני מוכן לקבל את הקבצים
is_phone_ready_for_the_files	משתנה שמכיל האם הטלפון מוכן לקבל את הקבצים (בטלפון זה האם המחשב מוכן לקבל את הקבצים)
mutex	שני הפרמטרים האלו הם בשביל לעצור את הקוד עד שהוא מקבל איתות שאפשר להמשיך
condition	
phone_ip	מייצג את הIP של המכשיר השני שאליו אני מתחבר
main_sending_socket	אובייקט של סוקט הקבלה הראשי
files_from_phone	רשימה של הקבצים שהטלפון רוצה להעביר (בטלפון זה רשימה של הקבצים של שהמחשב רוצה להעביר)
file_sending_sockets	רשימה של הסוקטים ששולחים קבצים
file_receiving_sockets	רשימה של הסוקטים שמקבלים קבצים
files	רשימה של הקבצים שהמכשיר הזה רוצה להעביר
files_and_paths	מילון של הקבצים ששהמכשיר השני רוצה להעביר והמיקום שלהם במכשיר הזה.
פעולות שלי	
__init__(self)	מאתחל אובייקט של המחלקה שאחראית על כל הפרויקט
exception_rose(self, error_message)	פעולה שנקראת ברגע שיש שגיאה ואז היא מציגה את הודעת השגיאה למשתמש כדי שלא כל התוכנה תקרוס

handle_connection(self, address)	פעולה שמקבלת את הIP והPORT של המכשיר השני, פותח את סוקט השליחה הראשי ומתחבר אליו
handle_files(self, list_of_files)	פעולה שמקבלת את רשימת הקבצים ומחליטה האם להעביר למסך השלישי או לחכות
phone_ready_for_files(self)	פעולה שנקראת כאשר המכשיר השני מוכן לקבל את הקבצים ומחליטה האם מתחילים את שליחת הקבצים או לא.
send_files(self)	הפעולה שאחראית על השליחה והקבלה של הקבצים
finished_window3(self, files)	פעולה שנקראת כשהמסך השלישי מסתיים
received_message(self, message)	פעולה שנקראת כאשר מתקבל הודעה מסוקט הקבלה הראשי (זה בשלב שליחת הקבצים בו מחכים להודעה מהמכשיר השני שנפתח סוקט וכו')
finished_window2(self, files)	פעולה שנקראת כשהמסך השני מסתיים

קטעי הקוד

התחברות דרך WIFI:

ביכולת זו הטלפון פותח מצלמה וסורק קוד QR בו נמצאים הפרטים להתחברות (WIFI וPORT). לאחר מכן הוא מעביר את המידע לאובייקט הפרויקט והוא פותח סוקט שמתחבר לסוקט שמאזין אצל המחשב.

פתיחת המצלמה ופיענוח הקוד QR:

```
class CameraThread(QThread):
    ImageUpdate = pyqtSignal(QImage)
    got_data = pyqtSignal(str)
    def run(self):
        # this is the object to capture an image
        capture = cv2.VideoCapture(0)
        # this is the object to decode the qr code
        detector = cv2.QRCodeDetector()
        while True:
            # it reads from the camera
            ret, frame = capture.read()
            if ret:
                # it's converting it to an image
                Image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                ConvertToQtFormat = QImage(Image.data, Image.shape[1],
Image.shape[0], QImage.Format_RGB888)
                Pic = ConvertToQtFormat.scaled(1440, 2960,
Qt.KeepAspectRatio)
                # updating the GUI with the new image
                self.ImageUpdate.emit(Pic)
                # here it tries to find and decode a qr code
                data, bbox, _ = detector.detectAndDecode(Image)
                if data:
                    # if it decoded the qr code it emits the data to the
project object
                    self.got_data.emit(str(data))
                    break
```

קבלת המידע ויצירת סוקט התחברות (יש פה גם קוד שקשור בהצפנה של המידע):

```
def handle_data(self, data):
    data = data.split()

    # extracting the data from what we got
    self.computer_ip = data[0]
    self.computer_port = int(data[1])
    self.key = data[2].encode()

    # adding the encryption key to the main receiving socket
    self.main_receiving_socket.add_encrypting_object(self.key)

    # creating and starting the main sending socket
    self.main_sending_socket = MainSendingSocket(self.computer_ip,
self.computer_port, self.key)
    self.main_sending_socket.exception_rose.connect(self.exception_rose)
    self.main_sending_socket.start()
```

<u>בחירת קבצים:</u>

ביכולת זו ניתן לבחור קבצים מתוך מערכת הקבצים של המשתמש.

הגדרת הUI:

```
def setup_ui(self):
    self.files = []
    self.setObjectName("Form")

    # creating the vertical layout and the actual widget
    self.vertical_layout_widget = QWidget(self)
    self.vertical_layout_widget.setGeometry(QRect(200, 100, 900, 500))
    self.vertical_layout_widget.setObjectName("verticalLayoutWidget")

    self.vertical_layout = QVBoxLayout(self.vertical_layout_widget)
    self.vertical_layout.setContentsMargins(0, 0, 0, 0)
    self.vertical_layout.setObjectName("verticalLayout")

    # creating the button that you press to add a file and adding it to the
vertical layout
    self.push_button = QPushButton(self.vertical_layout_widget)
    self.push_button.setObjectName("pushButton")
    self.push_button.setText("+")
    self.push_button.setFont(QFont('Arial', 15))
    self.push_button.clicked.connect(self.get_file_path)
    self.vertical_layout.addWidget(self.push_button)

    # creating a label and adding it to the vertical layout
    self.title_label = QLabel(self.vertical_layout_widget)
    self.title_label.setText("files:")
    self.title_label.setFont(QFont('Arial', 15))
    self.vertical_layout.addWidget(self.title_label)

    # creating the area that you add the files to in the GUI
    self.scroll_area = QScrollArea(self.vertical_layout_widget)
    self.scroll_area.setWidgetResizable(True)
    self.scroll_area.setObjectName("scrollArea")

    self.scroll_area_widget_contents = QWidget()
    self.scroll_area_widget_contents.setGeometry(QRect(0, 0, 287, 153))

self.scroll_area_widget_contents.setObjectName("scrollAreaWidgetContents")

    layout = QVBoxLayout(self.scroll_area_widget_contents)
    layout.addStretch()

    self.scroll_area_widget_contents.setLayout(layout)
    self.scroll_area.setWidget(self.scroll_area_widget_contents)
    self.vertical_layout.addWidget(self.scroll_area)

    # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
    self.ok_button = QPushButton(self.vertical_layout_widget)
    self.ok_button.setObjectName("okButton")
    self.ok_button.setText("OK")
    self.ok_button.setFont(QFont('Arial', 15))
    self.ok_button.clicked.connect(self.ok_button_clicked)
    self.vertical_layout.addWidget(self.ok_button)
```

פתיחת מערכת הקבצים ובחירת קובץ:

```
def get_file_path(self):
    # opening the file system and getting the location of the file i choose
    file_app = QFileDialog(self)
    file_app.fileSelected.connect(self.add_file)
    file_app.setFixedSize(1300, 700)
    file_app.show()
```

עדכון הGUI עם הקובץ שבחרתי:

```
def add_file(self, file_path):
    # adding the file to the list of files updating the GUI with the file
    self.files.append(file_path)
    self.label = QLabel(file_path, self.scroll_area_widget_contents)
    self.scroll_area_widget_contents.layout().addWidget(self.label)
```

בחירת מיקום לקבצים במערכת הקבצים:

ביכולת זו לאחר שהמכשיר השני העביר את רשימת הקבצים שהוא רוצה להעביר, אתה בוחר מיקום לכל אחד מהקבצים האלו במערכת הקבצים של המכשיר הזה.

הגדרת הUI:

```
def setup_ui(self):
    self.setObjectName("Form")

    # creating the vertical layout and the actual widget
    self.vertical_layout_widget = QWidget(self)
    self.vertical_layout_widget.setGeometry(QRect(200, 100, 900, 500))
    self.vertical_layout_widget.setObjectName("vertical_layout_widget")

    self.vertical_layout = QVBoxLayout(self.vertical_layout_widget)
    self.vertical_layout.setContentsMargins(0, 0, 0, 0)
    self.vertical_layout.setObjectName("vertical_layout")

    # creating a label and adding it to the vertical layout
    self.title_label = QLabel(self.vertical_layout_widget)
    self.title_label.setObjectName("label")
    self.title_label.setText("Files:")
    self.title_label.setFont(QFont('Arial', 15))
    self.vertical_layout.addWidget(self.title_label)

    # creating the area where the files that the other device chose will be
    self.scroll_area = QScrollArea(self.vertical_layout_widget)
    self.scroll_area.setWidgetResizable(True)
    self.scroll_area.setObjectName("scroll_area")

    self.scroll_area_widget_contents = QWidget()
    self.scroll_area_widget_contents.setGeometry(QRect(0, 0, 417, 255))

self.scroll_area_widget_contents.setObjectName("scroll_area_widget_contents
")

    layout = QFormLayout(self.scroll_area_widget_contents)

    self.scroll_area_widget_contents.setLayout(layout)
    self.scroll_area.setWidget(self.scroll_area_widget_contents)
    self.vertical_layout.addWidget(self.scroll_area)

    # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
    self.ok_button = QPushButton(self.vertical_layout_widget)
    self.ok_button.setObjectName("ok_button")
    self.ok_button.setText("OK")
    self.ok_button.setFont(QFont('Arial', 15))
    self.ok_button.clicked.connect(self.check_all_files_have_location)
    self.vertical_layout.addWidget(self.ok_button)
```

פותח את מערכת הקבצים ומקבל את התיקייה שבחרת:

```
def select_directory(self, file_name):
    # creating the object of the file system
    options = QFileDialog.Options()
    options |= QFileDialog.ReadOnly
    file_dialog = QFileDialog(self, options=options)
```

```
    # setting the object to a mode in which you choose a directory instead
of a file
    file_dialog.setFileMode(QFileDialog.Directory)
    file_dialog.setOption(QFileDialog.ShowDirsOnly, True)
    file_dialog.setFixedSize(1300, 700)
    file_dialog.show()
    try:
        # it tries to see if you chose a directory or just closed the
window
        # if you chose a directory it would work and if not the try except
will catch it
        if file_dialog.exec_():
            directory = file_dialog.selectedFiles()[0]
        self.file_location_dict[file_name] = directory
    except:
        pass
```

בודק עם לכל הקבצים יש מיקום:

```
def check_all_files_have_location(self):
    files_have_location = True
    # it goes over each file and checking if it has a location or is it a
None type
    for location in self.file_location_dict.values():
        if location == None:
            files_have_location = False
    if files_have_location:
        # if all files have a location then it emits the files and their
location
        self.all_files_have_location.emit(self.file_location_dict)
```

<u>שליחת רשימת הקבצים להעביר:</u>

ביכולת זו המכשיר מעביר את רשימת הקבצים שהוא בחר למכשיר השני.

עדכון הקבצים באובייקט של סוקט השליחה הראשי:

```
def got_files(self, files):
    # update the files
    self.files = files

    # this part stops the wait in the run function of the main sending
socket
    self.mutex.lock()
    self.condition.wakeAll()
    self.mutex.unlock()
```

שליחת הקבצים:

```
# encoding the file list into bytes
serialized_file_list = pickle.dumps(self.files)
try:
    # sending the file list

self.sending_socket.send(self.encrypting_object.encrypt(serialized_file_lis
t))
```

שליחה וקבלה של הקבצים:

ביכולת זו המכשירים מעבירים אחד לשני את הקבצים שהם רוצים להעביר ושמים אותם במיקומים שנבחרו להם.

שליחה של קובץ:

```
with open(self.file_path, "rb") as f:
    while True:
        bytes_read = f.read(self.BUFFER_SIZE)
        if not bytes_read:
            # file transmitting is done
            break
        # we use sendall to assure transmission in
        # busy networks

self.sending_socket.sendall(self.encrypting_object.encrypt(bytes_read))
```

קבלה של קובץ:

```
with open(os.path.join(location, file_name), "wb") as file:
    while True:
        # read 1024 bytes from the socket (receive)
        bytes_read =
self.encrypting_object.decrypt(self.receiving_socket.recv(self.BUFFER_SIZE)
)
        if not bytes_read:
            # nothing is received
            # file transmitting is done
            break
        # write to the file the bytes we just received
        file.write(bytes_read)
```

הקוד שאחראי לפקח על כל השליחה והקבלה של הקבצים (במכשיר השני הסדר של הלולאות הוא הפוך):

```
self.file_sending_sockets = []
self.file_receiving_sockets = []
i=0

# sending the files
self.mutex.lock()
for file in self.files:
    # waiting for the other device to tell us he opened a socket and is
listening on it
    self.condition.wait(self.mutex)

    # connection to that socket
    sock = FileSendingSocket(self.phone_ip, self.g_port, file, self.key)
    self.g_port +=1
    sock.start()
    self.file_sending_sockets.append(sock)

    # telling the other device we connected to the socket and he can open
another one
    self.main_sending_socket.send_massage.emit("connected")
    time.sleep(1)
```

```
# receiving the files
for file in self.files_and_paths:
    # opening a listening socket
    sock = FileReceivingSocket(self.ip, self.g_port, self.files_and_paths,
self.key)
    self.g_port +=1
    sock.start()
    self.file_sending_sockets.append(sock)

    # telling the other device we opened a socket and he can connect to it
    self.main_sending_socket.send_massage.emit("socket opened")
    time.sleep(1)

    # waiting for the other device to tell us he opened connected to our
socket
    self.condition.wait(self.mutex)
    i+=1
```

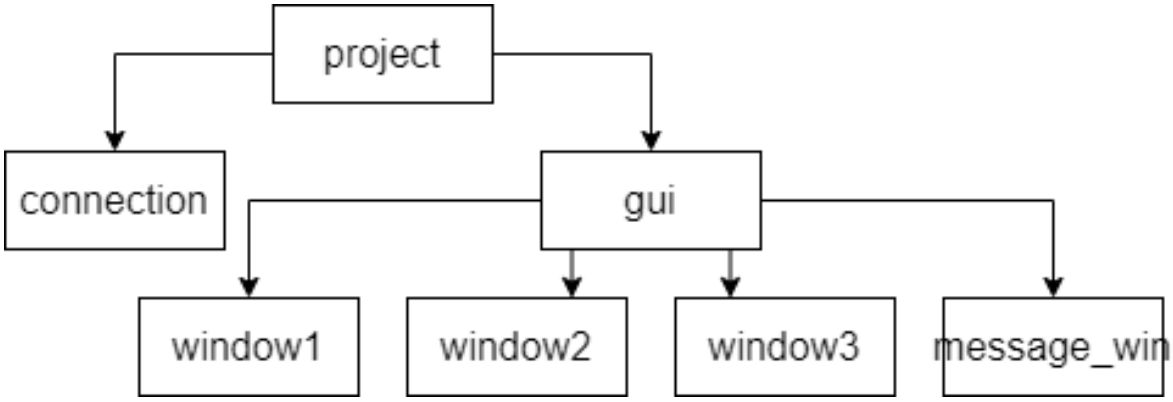
מסמך הבדיקות המלא

שם בדיקה	מטרת הבדיקה	מה שתכננתי לבצע	מה בוצע בפועל	בעיות שנתגלו והפתרונות שלהם
קובץ קיים	בדיקה זו אמורה לוודא שהקובץ שאני רוצה להעביר קיים	אני אנסה לגשת לקובץ ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש	ניסיתי לגשת לקובץ ואם נוצר שגיאה, תפסתי והצגתי למשתמש	וידאתי בהתחלה שהקובץ קיים בהתחלה אבל אז אחרי זה יכלו למחוק אותו והתוכנה לא תשים לב. מה שעשיתי זה שמתי את כל החלק של פתיחת הקובץ בשביל השליחה תחת מנגנון של תפיסת שגיאות.
חיבור קיים	בדיקה זו אמורה לוודא שיש חיבור בין המכשירים	אני אנסה לשלוח משהו למכשיר השני ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש	ניסיתי לשלוח משהו בין המכשירים ואם לה היה חיבור תפסתי את זה והצגתי למשתמש שהחיבור נותק	בהתחלה ניסיתי פשוט לשלוח בהתחלה משהו בין שני המכשירים לראות שהחיבור קיים אבל אז אם החיבור נותק באמצע התוכנה הייתה קורסת. מה שעשיתי זה שמתי כל שליחה תחת מנגנון תפיסת שגיאות
מיקום קיים	בדיקה שהמיקום שאני רוצה שהקבצים יגיעו אליו קיים	אני אנסה לשמור את הקובץ במיקום הזה ואם תיווצר שגיאה, אתפוס אותה ואציג למשתמש	ניסיתי לשמור את הקובץ בתיקייה הזו ואם זה לא עבד אז הצגתי למשתמש	בהתחלה הייתי בטוח שאם אני מנסה לפתוח מיקום לא קיים זה ייצור כזה אבל זה לא. מה שעשיתי זה הוספתי את הבדיקה הזו
הקבצים הועברו בהצלחה	בדיקה האם כל הקבצים הועברו בהצלחה למכשיר השני	אשווה את הקבצים שקיבלתי עם רשימת הקבצים שקיבלתי מהמכשיר השני	לכל סוקט שמקבל קובץ יש משתנה סיימתי של אמת או שקר ורק כאשר כולם היו באמת המשכתי הלאה	כשקובץ עוד היה בתהליך של העברה וניסיתי לגשת אליו ולבדוק אם הוא הועבר זה יצר בעיה. מה שעשיתי זה החלפתי את שיטת הבדיקה.
המכשיר השני מוכן	בדיקה זו תראה האם המכשיר השני מוכן להעברת הקבצים	אשלח הודעה ממכשיר אחד שהוא מוכן לקבל את הקבצים ואחכה להודעה מהמכשיר השני	שלחתי הודעה ממכשיר אחד שהוא מוכן לקבל את הקבצים וחיכתי להודעה מהמכשיר השני	כשהמכשיר האחר היה עוד במסך השלישי הוא לא יכל לקבל את ההודעה שאני מוכן, התוכנה לא זזה. מה שעשיתי זה פתחתי thread חדש שיאזין להודעות שמגיעות.
התחברות נכונה	בדיקה זו תוודא שכל סוקט שנפתח מתחבר לסוקט	כל פעם שאני פותח סוקט האזנה אני אשלח הודעה ואחכה שאני מקבל	כל פעם שפתחתי סוקט שלחתי שסוקט נפתח.	היה לי קשה לתזמן את כל ההודעות אחת עם השנייה. מה שעשיתי זה הוספתי עצירות באורך של שנייה

	שנפתח במכשיר השני	הודעה מהמכשיר השני שהוא פתח סוקט התחברות.	אחרי שסוקט מהמכשיר השני התחבר שלחתי מהמכשיר השני שסוקט התחבר.	בשביל שהתזמון של ההודעות יהיה טוב.

מדריך למשתמש

עץ הקבצים:



התקנות המערכת:

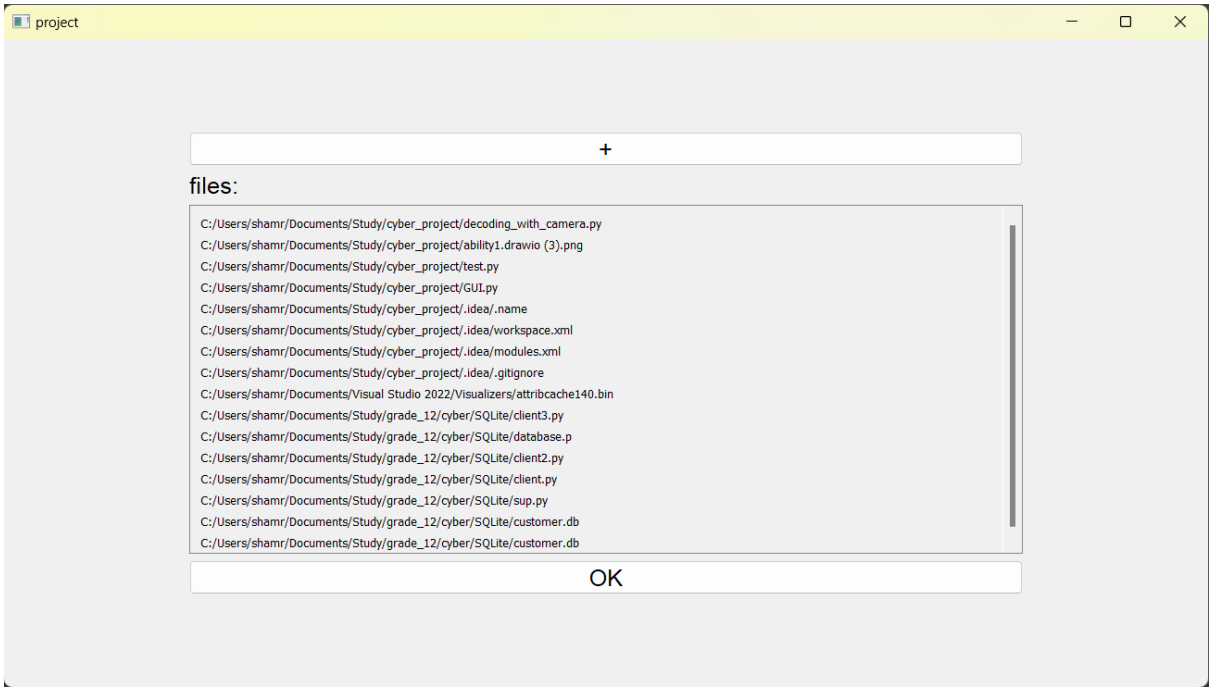
אני צריך python 3.9 על מחשב pydroid 3 עם כל הספריות ועם הרשאות בטלפון אנדרואיד.
בטלפון צריך מצלמה עובדת, וצריך רשת כלשהי שהמכשירים יתחברו אליה. בנוסף צריך שיתאפשר ארכיטקטורת P2P בין המכשירים.

מדריך עבור משתמש המחשב:

המסך הראשון שהוא יראה זה קוד הQR

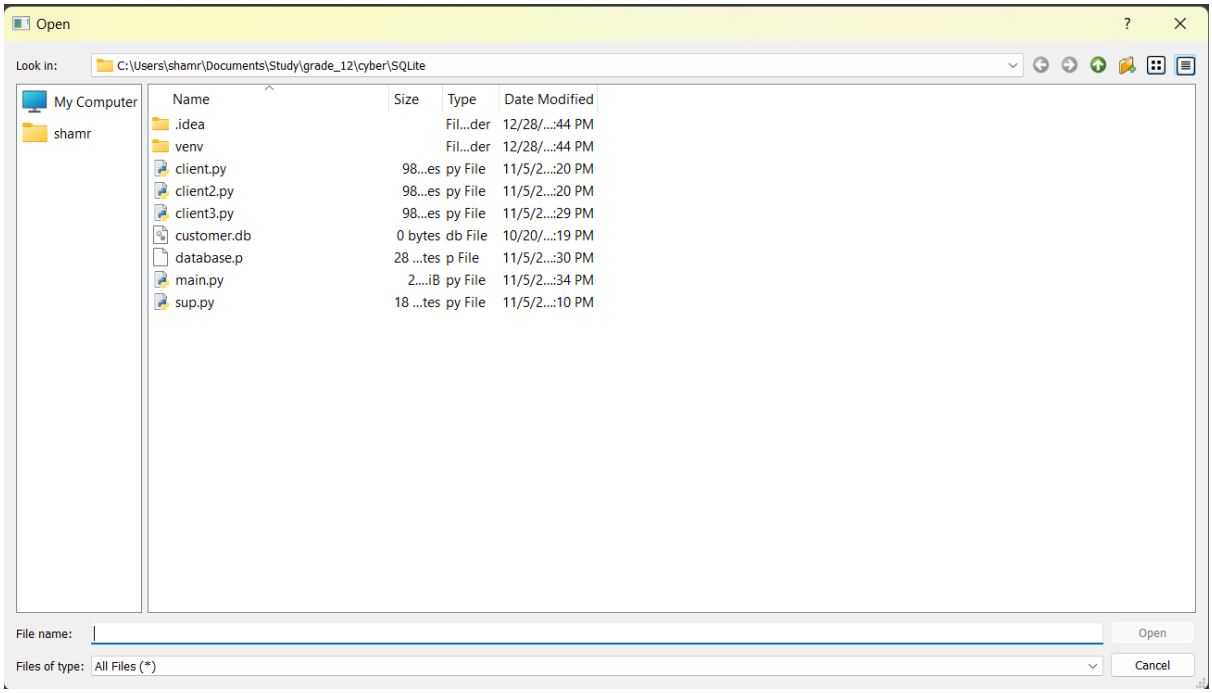
פה מוצפנים הIP והPORT של המחשב ומפתח ההצפנה שהוא משתמש בו. בשביל לעבור הלאה למסך הבא הטלפון צריך לסרוק את הקוד QR הזה.

המסך הבא הוא מסך בחירת הקבצים:

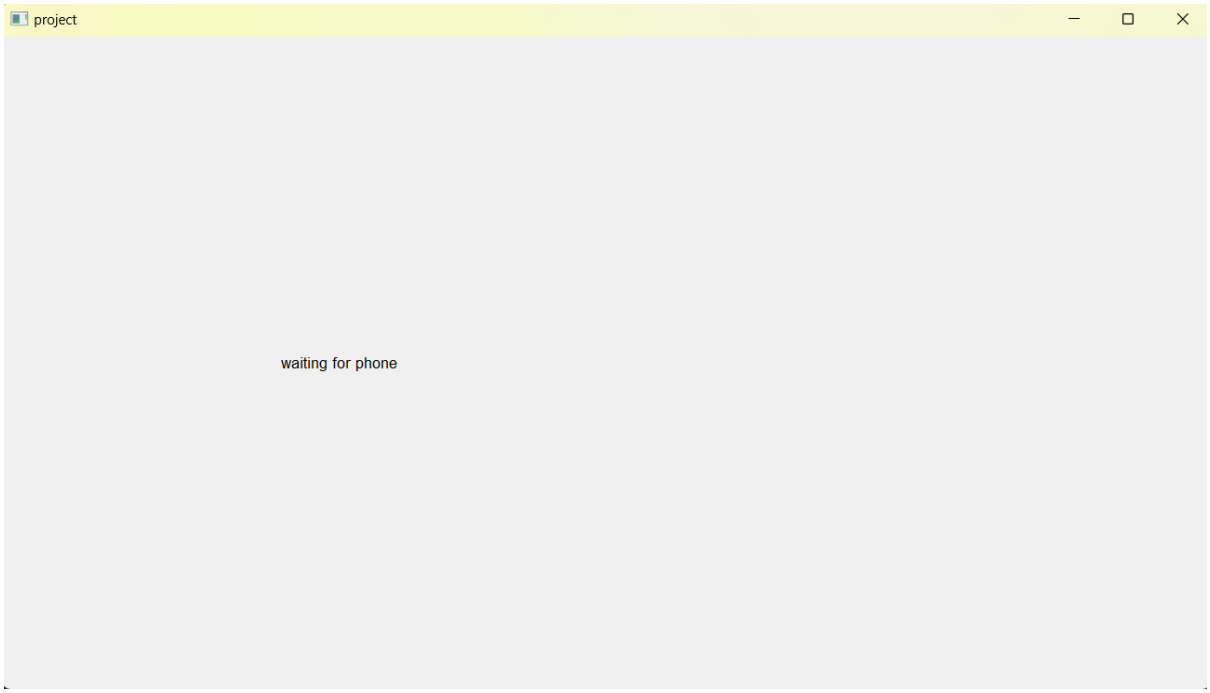


ניתן לראות שיש בלמעלה של המסך כפתור +. עם הכפתור הזה אפשר להוסיף קבצים. כשאתה לוחץ עליו מערכת הקבצים נפתחת ואתה יכול לבחור קובץ. כשאתה בוחר קובץ הוא מוצג לך כמו שניתן לראות מתחת לכפתור זה. כשסיימת לבחור את הקבצים אתה עובר למסך הבא.

זה המסך של מערכת ההפעלה:

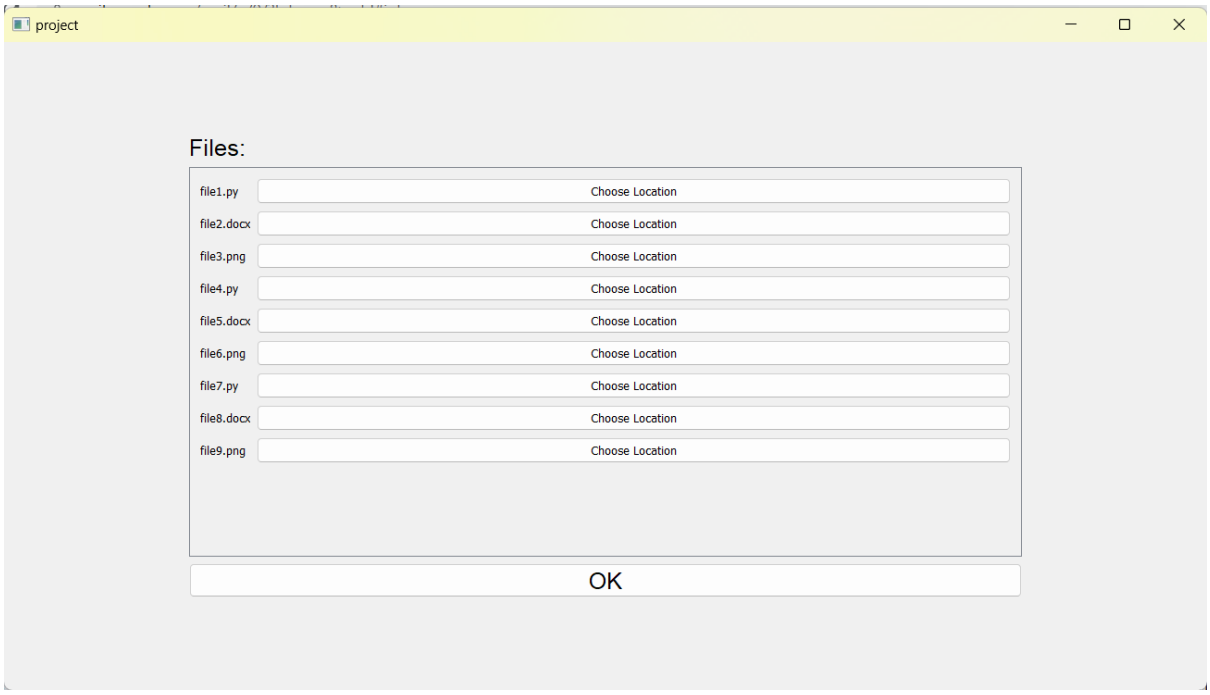


לאחר מכן או שאתה עובר למסך שבוא מחכים לטלפון:



במסך זה מחכים לטלפון עד שהוא יבחר את הקבצים שהוא רוצה להעביר.

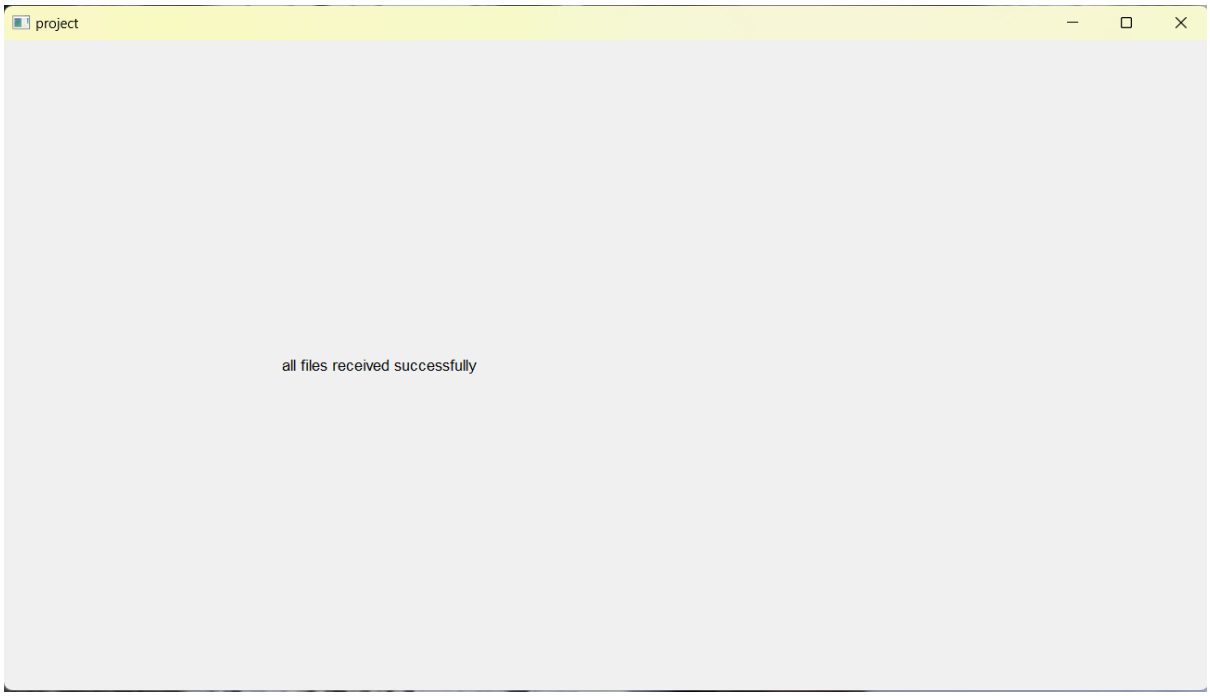
למסך זה מגיעים או אחרי המסך הזה או ישר אחרי שבוחרים קבצים:



ניתן לראות שעבור כל קובץ אפשר לבחור מיקום. כשלוחצים על אחד הכפתורים האלה מערכת הקבצים נפתחת ואפשר לבחור מיקום לקובץ שבאותו שורה. כשמסיימים אז לוחצים על כפתור הOK

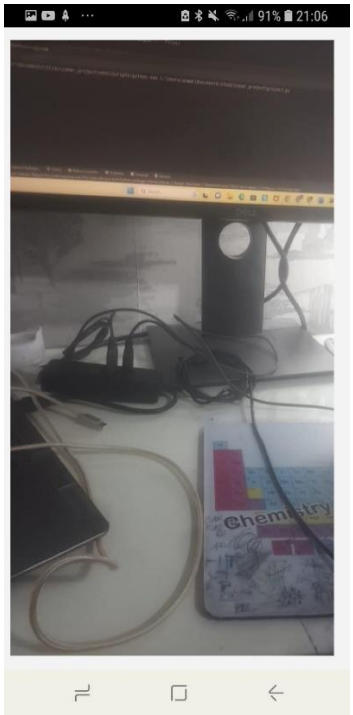
אחרי זה או שמגיעים למסך שבוא מחכים לטלפון או שהעברת הקבצים מתחילה.

אם העברת הקבצים עברה בהצלחה אז מגיעים למסך הזה:



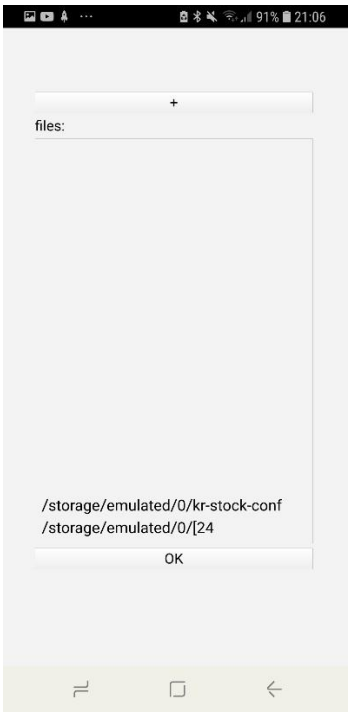
מדריך עבור משתמש הטלפון:

המסך הראשון הוא המצלמה:



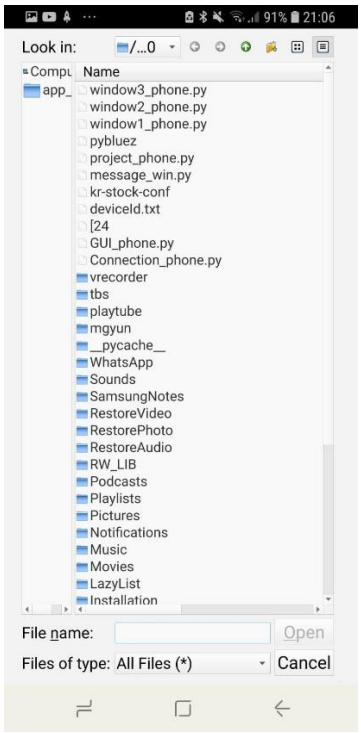
במסך זה צריך לסרוק את קוד הQR. לאחר מכן הטלפון מקבל את כל המידע שהוא צריך בשביל להעביר את הקבצים.

אחרי זה נפתח מסך בחירת הקבצים:

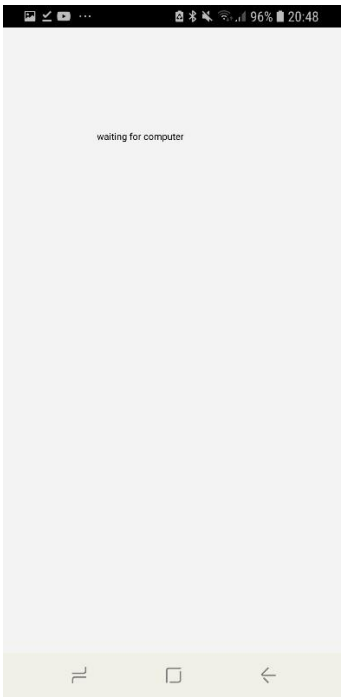


ניתן לראות שיש בלמעלה של המסך כפתור +. עם הכפתור הזה אפשר להוסיף קבצים. כשאתה לוחץ עליו מערכת הקבצים נפתחת ואתה יכול לבחור קובץ. כשאתה בוחר קובץ הוא מוצג לך כמו שניתן לראות מתחת לכפתור זה. כשסיימת לבחור את הקבצים אתה עובר למסך הבא.

זה המסך של מערכת הקבצים:

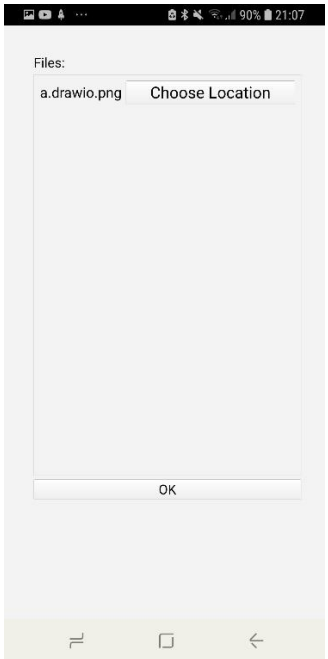


לאחר מכן או שאתה עובר למסך שבוא מחכים לטלפון:



במסך זה מחכים לטלפון עד שהוא יבחר את הקבצים שהוא רוצה להעביר.

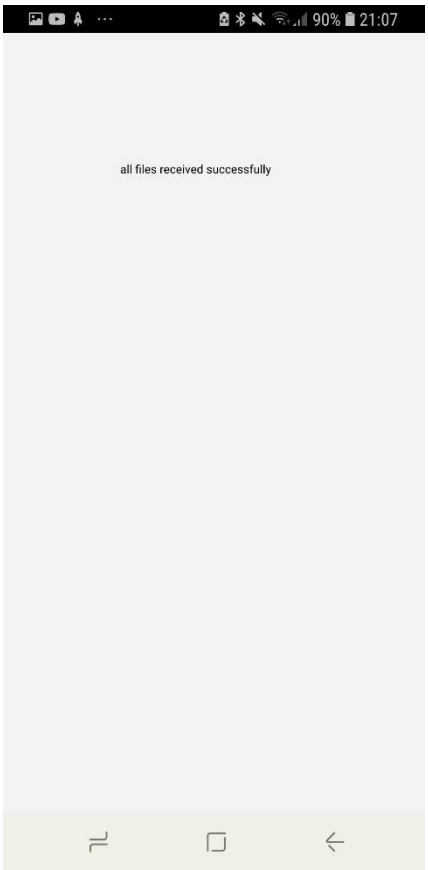
למסך זה מגיעים או אחרי המסך הזה או ישר אחרי שבוחרים קבצים:



ניתן לראות שעבור כל קובץ אפשר לבחור מיקום. כשלוחצים על אחד הכפתורים האלה מערכת הקבצים נפתחת ואפשר לבחור מיקום לקובץ שבאותו שורה. כשמסיימים אז לוחצים על כפתור הOK

אחרי זה או שמגיעים למסך שבוא מחכים לטלפון או שהעברת הקבצים מתחילה.

אם העברת הקבצים עברה בהצלחה אז מגיעים למסך הזה:



סיכום אישי

תיאור תהליך העבודה:

פרויקט זה היה פרויקט קשה, מאתגר ומהנה. למדתי ממהלך העבודה ולמידה הרבה על איך בונים פרויקט בתוכנה, איך מתמודדים עם אי הצלחה ותסכול, איך לחקור וללמוד דברים שאני לא מכיר בנושאים שאני לא מכיר, איך לפתור בעיות וכו'.

הצלחות בפרויקט:

- העברת הקבצים בין טלפון ומחשב – בתחילת הפרויקט משהו שפחדתי ממנו זה כל העניין של העברת הקבצים בין הטלפון למחשב. חששתי שזה לא יעבוד טוב ויהיו בעיות ואחד ההצלחות שלי זה להעביר את הקבצים מהטלפון למחשב בהצלחה.
- הצפנה – לפני הפרויקט לא כל כך ידעתי איך משתמשים בהצפנות ואיך משלבים אותם בתוך הפרויקט אבל לאחר למידה הצלחתי לשלב את ההצפנות של המידע העובר בתקשורת בקלות.
- ממשק המשתמש – בתחילת הפרויקט לא הכרתי שום ספרייה של ממשק משתמש ובחיים לא עבדתי עם אחת. בנוסף לכך הייתי צריך למצוא ספרייה שתעבוד גם עם הטלפון ולהתאים את כל הממשק לטלפון. אני מאוד מרוצה מאיך שממשק המשתמש נראה.

אתגרים בפרויקט:

- אחד מהאתגרים הכי גדולים בפרויקט היה איך אני משלב את כל הקבצים השונים יחד. לא ידעתי כל כך איך אני מחליף בין המסכים השונים, איך אני עובד גם עם הGUI וגם עם התקשורת במקביל.
- אתגר אחר שהיה לי זה לעבוד ולכתוב קוד בטלפון. בכללי מאוד לא היה לי נוח לכתוב קוד בטלפון ובגלל זה רוב הזמן רשמתי את הקוד במחשב. עבור שינויים קטנים לא הרגשתי שזה בזבוז זמן לכתוב במחשב ולהעביר לטלפון עבור כל שינוי קטן אז שינויים קטנים בקוד רשמתי בטלפון. זה גרם לכך שכאשר עשיתי שיניים גדולים בקבצים של הטלפון (שינויים אלו נעשים במחשב), כל השינויים הקטנים שעשיתי לא היו. בנוסף היה מאוד קשה למצוא בעיות ובאגים שהיו בטלפון כי לא הייתה אפשרות לדבג ולא יכולתי גם להדפיס דברים וגם לפתוח מסכי gui.

דברים חדשים שלמדתי באופן עצמאי:

- Pyqt5 – זו ספריית הGUI שהשתמשתי בה ולפני הפרויקט לא הכרתי אותה בכלל.
- עבודה עם מערכת הקבצים
- עבודה עם מצלמה
- יצירת קודי QR ופענוח שלהם.

כלים שאני אקח להמשך:

- יכולת ניהול זמנים – היה המון דברים לעשות בפרויקט זה בין אם זה כתיבת הקוד עצמו או עבודה על התיק פרויקט והיה צריך יכולת ניהול זמנים על מנת לתכנון הכל בצורה טובה.
- יכולת פתרון בעיות – בפרויקט זה היו מספר בעיות וקשיים שהיה צריך לפתור בדרכים יצירתיות.

במבט לאחור יכול להיות שהייתי עושה את הדברים הבאים אחרת:

- יכול להיות שהייתי מעביר את המידע שהטלפון צריך בשביל להתחבר בדרך אחרת כי קצת הסתבכתי עם המצלמה וקוד הQR בתחילת הפרויקט.

- הייתי מוסיף, במסך של בחירת הקבצים, את היכולת לבטל אחד מהקבצים שבחרתי, כי למרות שיש אפשרות פשוט לסגור את מערכת הקבצים וככה לא צריך לבחור קובץ אם לחצת על כפתור ה+ בטעות, זה היה עוזר למקרה שמישהו בחר את הקובץ הלא נכון בטעות.

ביבליוגרפיה

1. ChatGPT – השתמשתי לפתור בעיות שהיו לי:

https://www.geeksforgeeks.org/

2. GeeksForGeeks – השתמשתי בשביל הקוד QR והעברת הקבצים:

https://www.geeksforgeeks.org/

3. Python documentation – השתמשתי בזה בשביל ללמוד על פונקציות של כל מיני ספריות:

https://docs.python.org/

4. Qt documentation – השתמשתי בזה בשביל ללמוד על הספרייה Pyqt5:

https://doc.qt.io/qtforpython-5

5. RealPython – השתמשתי בזה בשביל ללמוד קונבנציות כתיבה נכונות:

https://realpython.com/

6. StackOverFlow – השתמשתי בזה למצוא פתרונות לבעיות שהיו לי:

https://stackoverflow.com/

7. Tech With Tim – השתמשתי בערוץ יוטיוב שלו ללמוד Pyqt5:

https://www.youtube.com/@TechWithTim

8. The python code – השתמשתי בזה בשביל שליחת הקבצים:

https://www.thepythoncode.com/

9. tutorials point – השתמשתי בזה בשביל להחליף בין המסכים:

https://www.tutorialspoint.com/pyqt

נספחים

הקוד של המחשב:

המסך הראשון:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

class Window1(QWidget):
    # this window will present the QR code
    def __init__(self, path):
        super(Window1, self).__init__()
        self.path = path
        self.window1_ui()

    def window1_ui(self):
        # creating the label that will hold the QR code
        self.photo = QLabel(self)
        self.photo.setPixmap(QPixmap(self.path))
        self.photo.setScaledContents(True)
        self.photo.setGeometry(QRect(300, 0, 700, 700))
```

המסך השני:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *


class Window2(QWidget):
    # this is the window where you choose files
    finished_choosing_files = pyqtSignal(list)

    def __init__(self):
        super(Window2, self).__init__()
        self.setup_ui()

    def setup_ui(self):
        self.files = []
        self.setObjectName("Form")

        # creating the vertical layout and the actual widget
        self.vertical_layout_widget = QWidget(self)
        self.vertical_layout_widget.setGeometry(QRect(200, 100, 900, 500))
        self.vertical_layout_widget.setObjectName("verticalLayoutWidget")

        self.vertical_layout = QVBoxLayout(self.vertical_layout_widget)
        self.vertical_layout.setContentsMargins(0, 0, 0, 0)
        self.vertical_layout.setObjectName("verticalLayout")

        # creating the button that you press to add a file and adding it to
the vertical layout
        self.push_button = QPushButton(self.vertical_layout_widget)
        self.push_button.setObjectName("pushButton")
        self.push_button.setText("+")
```

```
        self.push_button.setFont(QFont('Arial', 15))
        self.push_button.clicked.connect(self.get_file_path)
        self.vertical_layout.addWidget(self.push_button)

        # creating a label and adding it to the vertical layout
        self.title_label = QLabel(self.vertical_layout_widget)
        self.title_label.setText("files:")
        self.title_label.setFont(QFont('Arial', 15))
        self.vertical_layout.addWidget(self.title_label)

        # creating the area that you add the files to in the GUI
        self.scroll_area = QScrollArea(self.vertical_layout_widget)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setObjectName("scrollArea")

        self.scroll_area_widget_contents = QWidget()
        self.scroll_area_widget_contents.setGeometry(QRect(0, 0, 287, 153))

self.scroll_area_widget_contents.setObjectName("scrollAreaWidgetContents")

        layout = QVBoxLayout(self.scroll_area_widget_contents)
        layout.addStretch()

        self.scroll_area_widget_contents.setLayout(layout)
        self.scroll_area.setWidget(self.scroll_area_widget_contents)
        self.vertical_layout.addWidget(self.scroll_area)

        # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
        self.ok_button = QPushButton(self.vertical_layout_widget)
        self.ok_button.setObjectName("okButton")
        self.ok_button.setText("OK")
        self.ok_button.setFont(QFont('Arial', 15))
        self.ok_button.clicked.connect(self.ok_button_clicked)
        self.vertical_layout.addWidget(self.ok_button)

    def add_file(self, file_path):
        # adding the file to the list of files updating the GUI with the
file
        self.files.append(file_path)
        self.label = QLabel(file_path, self.scroll_area_widget_contents)
        self.scroll_area_widget_contents.layout().addWidget(self.label)

    def get_file_path(self):
        # opening the file system and getting the location of the file i
choose
        file_app = QFileDialog(self)
        file_app.fileSelected.connect(self.add_file)
        file_app.setFixedSize(1300, 700)
        file_app.show()

    def ok_button_clicked(self):
        self.finished_choosing_files.emit(self.files)
```

המסך השלישי:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
```

```
class Window3(QWidget):
    # this is the window where you pick locations for the files the other
device chose
    all_files_have_location = pyqtSignal(dict)
    def __init__(self):
        super(Window3, self).__init__()
        self.file_location_dict = {}
        self.setup_ui()

    def setup_ui(self):
        self.setObjectName("Form")

        # creating the vertical layout and the actual widget
        self.vertical_layout_widget = QWidget(self)
        self.vertical_layout_widget.setGeometry(QRect(200, 100, 900, 500))
        self.vertical_layout_widget.setObjectName("vertical_layout_widget")

        self.vertical_layout = QVBoxLayout(self.vertical_layout_widget)
        self.vertical_layout.setContentsMargins(0, 0, 0, 0)
        self.vertical_layout.setObjectName("vertical_layout")

        # creating a label and adding it to the vertical layout
        self.title_label = QLabel(self.vertical_layout_widget)
        self.title_label.setObjectName("label")
        self.title_label.setText("Files:")
        self.title_label.setFont(QFont('Arial', 15))
        self.vertical_layout.addWidget(self.title_label)

        # creating the area where the files that the other device chose
will be
        self.scroll_area = QScrollArea(self.vertical_layout_widget)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setObjectName("scroll_area")

        self.scroll_area_widget_contents = QWidget()
        self.scroll_area_widget_contents.setGeometry(QRect(0, 0, 417, 255))

self.scroll_area_widget_contents.setObjectName("scroll_area_widget_contents
")

        layout = QFormLayout(self.scroll_area_widget_contents)

        self.scroll_area_widget_contents.setLayout(layout)
        self.scroll_area.setWidget(self.scroll_area_widget_contents)
        self.vertical_layout.addWidget(self.scroll_area)

        # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
        self.ok_button = QPushButton(self.vertical_layout_widget)
        self.ok_button.setObjectName("ok_button")
        self.ok_button.setText("OK")
        self.ok_button.setFont(QFont('Arial', 15))
        self.ok_button.clicked.connect(self.check_all_files_have_location)
        self.vertical_layout.addWidget(self.ok_button)

    def select_directory(self, file_name):
        # creating the object of the file system
        options = QFileDialog.Options()
        options |= QFileDialog.ReadOnly
        file_dialog = QFileDialog(self, options=options)
```

```
        # setting the object to a mode in which you choose a directory
instead of a file
        file_dialog.setFileMode(QFileDialog.Directory)
        file_dialog.setOption(QFileDialog.ShowDirsOnly, True)
        file_dialog.setFixedSize(1300, 700)
        file_dialog.show()
        try:
            # it tries to see if you chose a directory or just closed the
window
            # if you chose a directory it would work and if not the try
except will catch it
            if file_dialog.exec_():
                directory = file_dialog.selectedFiles()[0]
            self.file_location_dict[file_name] = directory
        except:
            pass

    def check_all_files_have_location(self):
        files_have_location = True
        # it goes over each file and checking if it has a location or is it
a None type
        for location in self.file_location_dict.values():
            if location == None:
                files_have_location = False
        if files_have_location:
            # if all files have a location then it emits a dictionary with
the files and their location
            self.all_files_have_location.emit(self.file_location_dict)

    def add_files(self, files):
        # here you add files to this window
        for file in files:
            # for each file it extracts the file name insert it to the dict
and adds a label for it in the GUI
            file = file.split("/")[-1]
            self.file_location_dict[file] = None
            label = QLabel(file)
            button = QPushButton()

button.clicked.connect(self.create_select_directory_function(label.text()))
            button.setText("Choose Location")
            self.scroll_area_widget_contents.layout().addRow(label, button)

    def create_select_directory_function(self, label_text):
        return lambda: self.select_directory(label_text)
```

מסך ההודעות:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys

class MessageWindow(QWidget):
    # this is the message window
    def __init__(self, message):
        super(MessageWindow, self).__init__()
        self.message = message
        self.setupUI()
```

```
    def setupUI(self):
        # we create the label that holds the message
        self.message_label = QLabel(self)
        self.message_label.setText(self.message)
        self.message_label.setFont(QFont('Arial', 10))
        self.message_label.setGeometry(QRect(300, 300, 700, 100))

    def change_message(self, message):
        # with this function we can change that message
        self.message_label.setText(message)
```

המסך הראשי:

```
from window1 import *
from window2 import *
from window3 import *
from message_win import *


class MainWindow(QMainWindow):
    # this is the window that supervises the GUI
    def __init__(self, path):
        super(MainWindow, self).__init__()
        self.path = path
        self.setGeometry(100, 100, 1300, 700)
        self.setWindowTitle("project")
        self.setup_ui()

    def setup_ui(self):
        self.current_win = 0

        # here i create the windows
        self.window1 = Window1(self.path)
        self.window2 = Window2()
        self.window3 = Window3()
        self.message_win = MessageWindow("")

        # and add them to the stack
        self.stack = QStackedWidget(self)
        self.stack.setGeometry(0, 0, 1300, 700)
        self.stack.addWidget(self.window1)
        self.stack.addWidget(self.window2)
        self.stack.addWidget(self.window3)
        self.stack.addWidget(self.message_win)

        # with the stack i can replace which window is being presented to
the user

        self.hbox = QHBoxLayout(self)
        self.hbox.addWidget(self.stack)

        self.stack.setCurrentIndex(self.current_win)

    def change_win(self):
        # this function changes the window to the next one
        self.current_win += 1
        self.stack.setCurrentIndex(self.current_win)

    def change_to_message_win(self, message):
```

```
        # this function changes the window to the message window
        self.message_win.change_message(message)
        self.stack.setCurrentIndex(3)
```

קובץ שבוא המחלקות האחראיות על התקשורת:

```
import time

from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import socket
import pickle
import os
from cryptography.fernet import Fernet



class MainSendingSocket(QThread):
    got_file_list = pyqtSignal(list)
    ready_to_send = pyqtSignal()
    send_massage = pyqtSignal(str)
    done_signal = pyqtSignal()
    exception_rose = pyqtSignal(str)

    def __init__(self, ip, port, key):
        super(MainSendingSocket, self).__init__()
        # here i set up all the important information for the connection
        self.ip = ip
        self.port = port
        self.sending_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.encrypting_object = Fernet(key)

        self.mutex = QMutex()
        self.condition = QWaitCondition()

        # here i connect the signals of this class to a slot
        self.got_file_list.connect(self.got_files)
        self.ready_to_send.connect(self.ready_to_send_files)
        self.send_massage.connect(self.send)
        self.done_signal.connect(self.done)
        self.done_condition = False

        self.files = []

    def run(self):
        self.mutex.lock()
        # here it connects to the phone and then waits until it has the
list of files to send
        self.connect_to_phone()
        self.condition.wait(self.mutex)

        # it converts the file list to bytes and sends it
        serialized_file_list = pickle.dumps(self.files)

        try:

self.sending_socket.send(self.encrypting_object.encrypt(serialized_file_lis
t))
```

```
            # here it wait again until the computer is ready to start
sending the files
            self.condition.wait(self.mutex)
            self.sending_socket.send(self.encrypting_object.encrypt("ready
for files".encode()))

            while True:
                # here it waits until the project tells it to send a
message
                # the message is that either a socket opened or that a
socket connected
                self.condition.wait(self.mutex)

self.sending_socket.send(self.encrypting_object.encrypt(self.message.encode
()))
                if self.done_condition:
                    break

        except Exception as exception:
            self.exception_rose.emit(str(exception))
        self.mutex.unlock()

    def connect_to_phone(self):
        # this function connects to the socket on the phone
        try:
            self.sending_socket.connect((self.ip, self.port))

        except Exception as exception:
            self.exception_rose.emit(str(exception))


    def got_files(self, files):
        # when the program will have the files a signal will be emitted to
this slot and this will run
        # this function sets the file list and tells the program to stop
waiting and continue
        self.files = files

        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

    def ready_to_send_files(self):
        # when the program will be ready to send the files a signal will be
emitted to this slot and this will run
        # this function tells the program to stop waiting and continue
        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

    def send(self, message):
        # when the program needs to send a message (during the file
sending) a signal will be emitted to this slot and this will run
        # this function sets the message list and tells the program to stop
waiting and continue
        self.mutex.lock()
        self.message = message
        self.condition.wakeAll()
        self.mutex.unlock()
```

```
    def done(self):
        # when the program is done a signal will be emitted to this slot
and this will run
        # this function sets the done condition to true list and tells the
program to stop waiting and continue
        self.done_condition = True
        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()


class FileSendingSocket(MainSendingSocket):
    def __init__(self, ip, port, file_path, key):
        super(FileSendingSocket, self).__init__(ip, port, key)
        self.file_path = file_path
        self.BUFFER_SIZE = 1024

    def run(self):
        self.connect_to_phone()

        # we send the socket the file name
        file_name = self.file_path.split("/")[-1]
        try:

self.sending_socket.send(self.encrypting_object.encrypt(str(file_name).enco
de()))

            with open(self.file_path, "rb") as f:
                while True:
                    # we read 1024 bytes from the file
                    bytes_read = f.read(self.BUFFER_SIZE)
                    if not bytes_read:
                        # file transmitting is done
                        break

                    # we encrypt the data
                    encrypted_bytes =
self.encrypting_object.encrypt(bytes_read)
                    size = len(encrypted_bytes)

                    # we send the size and then the data
                    self.sending_socket.send(str(size).encode())
                    message = self.sending_socket.recv(1024)
                    self.sending_socket.send(encrypted_bytes)


        except Exception as exception:
            self.exception_rose.emit(str(exception))

        self.sending_socket.close()


class MainReceivingSocket(QThread):
    connection_made = pyqtSignal(tuple)
    got_file_list_from_phone = pyqtSignal(list)
    ready_for_files = pyqtSignal()
    receive = pyqtSignal(str)
    done_signal = pyqtSignal()
    exception_rose = pyqtSignal(str)

    def __init__(self, ip, port, key):
```

```
        super(MainReceivingSocket, self).__init__()
        # here i set up all the important information for the connection
        self.ip = ip
        self.port = port
        self.done_signal.connect(self.done)
        self.done_condition = False
        self.encrypting_object = Fernet(key)



    def run(self):
        self.handle_connection()
        try:
            # we receive the file list and emit to the project object
            serialized_file_list =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024))
            list_of_files = pickle.loads(serialized_file_list)

            self.got_file_list_from_phone.emit(list_of_files)

            # we wait to receive a message that the phone is ready for
sending
            message =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024)).decode()
            if message == "ready for files":
                self.ready_for_files.emit()

            while True:
                # here it waits to receive a message
                # the message is that either a socket opened or that a
socket connected
                message =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024)).decode()
                self.receive.emit(message)
                if self.done_condition:
                    break

        except Exception as exception:
            self.exception_rose.emit(str(exception))
        self.receiving_socket.close()

    def handle_connection(self):
        # here we open a socket and listen for connections on the address
specified
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.bind((self.ip, self.port))
        sock.listen()
        try:
            self.receiving_socket, self.address = sock.accept()
        except Exception as exception:
            self.exception_rose.emit(str(exception))
        sock.close()
        self.handle_address()

    def handle_address(self):
        self.connection_made.emit(self.address)


    def done(self):
        self.done_condition = True
```

```
class FileReceivingSocket(MainReceivingSocket):
    def __init__(self, ip, port, files_and_paths, key):
        super(FileReceivingSocket, self).__init__(ip, port, key)
        self.file = None
        self.BUFFER_SIZE = 1024
        self.files_and_paths = files_and_paths
        self.finished = False

    def run(self):
        self.handle_connection()
        try:
            # here we receive the file name form the socket
            file_name =
self.encrypting_object.decrypt(self.receiving_socket.recv(self.BUFFER_SIZE)
).decode()
            location = self.files_and_paths[file_name]

            time.sleep(1)

            with open(f"{location}/{file_name}", "wb") as file:
                while True:
                    # we read the size of the part from the socket
                    size = self.receiving_socket.recv(1024)
                    self.receiving_socket.send(size)
                    if size == "":
                        # their is no next piece of data so it doesn't have
a size
                        break
                    size = int(size.decode())

                    # we receive the data and decrypt it
                    encrypted_bytes = self.receiving_socket.recv(size)
                    bytes_read =
self.encrypting_object.decrypt(encrypted_bytes)

                    if not bytes_read:
                        # nothing is received
                        # file transmitting is done
                        break
                    # write to the file the bytes we just received
                    file.write(bytes_read)
        except Exception as exception:
            self.exception_rose.emit(str(exception))
        self.finished = True
        self.receiving_socket.close()

    def handle_address(self):
        pass
```

אובייקט הפרויקט האחראי על הכל:

```
import time
from cryptography.fernet import Fernet
from gui import *
from connection import *
import qrcode
from threading import Thread
import sys

class Project:
```

```
    def __init__(self):
        # we set all the important stuff here
        self.key = Fernet.generate_key()
        self.ip = str(socket.gethostbyname(socket.gethostname()))
        self.port = 6744
        self.phone_port = 6745
        self.sending_port = 6746
        self.path = "qr.png"
        self.qr_image = qrcode.make(f"{self.ip} {str(self.port)}
{self.key.decode()}")
        self.qr_image.save(self.path)
        self.main_window = MainWindow(self.path)
        self.main_receiving_socket = MainReceivingSocket(self.ip,
self.port, self.key)
        self.got_files = False
        self.finished_window2 = False
        self.ready_for_the_files = False
        self.is_phone_ready_for_the_files = False
        self.mutex = QMutex()
        self.condition = QWaitCondition()
        self.files_received = []

        # here we connect all the signals from the sockets and windows

self.main_receiving_socket.connection_made.connect(self.handle_connection)

self.main_receiving_socket.got_file_list_from_phone.connect(self.handle_fil
es)

self.main_receiving_socket.ready_for_files.connect(self.phone_ready_for_fil
es)

self.main_window.window2.finished_choosing_files.connect(self.finished_wind
ow2)

self.main_window.window3.all_files_have_location.connect(self.finished_wind
ow3)
        self.main_receiving_socket.receive.connect(self.received_message)

self.main_receiving_socket.exception_rose.connect(self.exception_rose)

        self.main_receiving_socket.start()

    def exception_rose(self, error_message):
        # this function gets called in case of an exception
        self.main_window.change_to_message_win(error_message)

    def handle_connection(self, address):
        # this gets called when the phone connected and now we need to
connect to the phone
        self.phone_ip = address[0]
        self.main_sending_socket = MainSendingSocket(self.phone_ip,
self.phone_port, self.key)

self.main_sending_socket.exception_rose.connect(self.exception_rose)
        self.main_sending_socket.start()
        self.main_window.change_win()

    def handle_files(self, list_of_files):
        # this handles the file list from the phone and adds it to the GUI
        self.files_from_phone = list_of_files
```

```
        self.main_window.window3.add_files(self.files_from_phone)
        self.got_files = True
        if (self.finished_window2):
            self.main_window.change_win()

    def phone_ready_for_files(self):
        # this gets called when the phone sends a message that it is read
to send the files
        self.is_phone_ready_for_the_files = True
        if self.ready_for_the_files:
            # if we are also ready to send the files the file sending
starts
            self.main_window.change_to_message_win("transferring files")
            thread = Thread(target=self.send_files)
            thread.start()
        else:
            pass

    def send_files(self):
        self.file_sending_sockets = []
        self.file_recving_sockets = []

        # sending the files
        self.mutex.lock()
        for file in self.files:
            # waiting for the other device to tell us he opened a socket
and is listening on it
            self.condition.wait(self.mutex)
            # connecting to that socket
            sock = FileSendingSocket(self.phone_ip, self.sending_port,
file, self.key)
            self.sending_port +=1
            sock.start()
            self.file_sending_sockets.append(sock)
            # telling the other device we connected to the socket and he
can open another one
            self.main_sending_socket.send_massage.emit("connected")
            time.sleep(1)
        i=0
        self.sockets = ["" for f in self.files_and_paths.keys()]

        # receiving the files
        for file in self.files_and_paths:
            # opening a listening socket
            self.sockets[i] = FileReceivingSocket(self.ip,
self.sending_port, self.files_and_paths, self.key)
            self.sending_port +=1
            self.sockets[i].start()
            self.file_recving_sockets.append(self.sockets[i])
            # telling the other device we opened a socket and he can
connect to it
            self.main_sending_socket.send_massage.emit("socket opened")
            time.sleep(1)
            # waiting for the other device to tell us he opened connected
to our socket
            self.condition.wait(self.mutex)
            i+=1

        # emitting to the main sockets that we finished transferring so
they can close
        self.main_sending_socket.done_signal.emit()
```

```
        self.main_receiving_socket.done_signal.emit()

        # making sure all receiving socket have finished
        all_socket_finished = True
        while True:
            for socket in self.sockets:
                # going through each socket and making sure it's finished
                if not socket.finished:
                    all_socket_finished = False
            if all_socket_finished:
                break
            all_socket_finished = True

        # updating the GUI that file transfer went well
        self.main_window.change_to_message_win("all files received
successfully")

        self.mutex.unlock()



    def finished_window2(self, files):
        # this happens when the second window finishes. it updates the
files
        self.files = files
        self.finished_window2 = True
        # if the phone already sent the file than we can move to the next
window and if not we wait for him
        if (self.got_files):
            self.main_window.change_win()
        else:
            self.main_window.change_to_message_win("waiting for phone")
        # sending to the phone the files we chose
        self.main_sending_socket.got_files(self.files)

    def finished_window3(self, files_and_paths):
        # this happens when the third window is finished
        # we update the locations of the files the phone sent
        self.files_and_paths = files_and_paths
        # sending that we are ready to send the files
        self.main_sending_socket.ready_to_send_files()
        self.ready_for_the_files = True
        # if the phone already sent that he is ready to send files we start
sending files
        # if not we wait for it
        if self.is_phone_ready_for_the_files:
            self.main_window.change_to_message_win("transferring files")
            thread = Thread(target=self.send_files)
            thread.start()
        else:
            self.main_window.change_to_message_win("waiting for phone")

    def received_message(self, message):
        # this get called when we receive a message during the file sending
        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

def main():
    app = QApplication(sys.argv)
    project = Project()
```

```
    project.main_window.show()

    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

הקוד של הטלפון:

המסך הראשון:

```
import socket
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import cv2

class Window1(QWidget):
    # this is window where we display the camera
    def __init__(self):
        super(Window1, self).__init__()
        self.initUI()

    def initUI(self):
        self.VBL = QVBoxLayout()

        # this is the label that will display the camera
        self.FeedLabel = QLabel()
        self.VBL.addWidget(self.FeedLabel)

        # starting the camera
        self.CameraThread = CameraThread()

        self.CameraThread.start()
        self.CameraThread.ImageUpdate.connect(self.ImageUpdateSlot)
        self.setLayout(self.VBL)


    def ImageUpdateSlot(self, Image):
        # this will change the image displayed
        self.FeedLabel.setPixmap(QPixmap.fromImage(Image))

class CameraThread(QThread):
    ImageUpdate = pyqtSignal(QImage)
    got_data = pyqtSignal(str)
    def run(self):
        # this is the object to capture an image
        Capture = cv2.VideoCapture(0)
        # this is the object to decode the qr code
        detector = cv2.QRCodeDetector()
        while True:
            # it reads from the camera
            ret, frame = Capture.read()
            if ret:
                # it's converting it to an image
                Image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                ConvertToQtFormat = QImage(Image.data, Image.shape[1],
Image.shape[0], QImage.Format_RGB888)
                Pic = ConvertToQtFormat.scaled(1440, 2960,
Qt.KeepAspectRatio)
                # updating the GUI with the new image
                self.ImageUpdate.emit(Pic)
                data, bbox, _ = detector.detectAndDecode(Image)
                if data:
                    # if it decoded the qr code it emits the data to the
project object
```

```
                    self.got_data.emit(str(data))
                    break
```

המסך השני

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys


class Window2(QWidget):
    # this is the window where you choose files
    finished = pyqtSignal(list)
    def __init__(self):
        super(Window2, self).__init__()
        self.i = 0
        self.setupUi()
    def setupUi(self):
        self.files = []
        self.setObjectName("Form")
        self.setGeometry(100, 100, 1300, 700)

        # creating the vertical layout and the actual widget
        self.verticalLayoutWidget = QWidget(self)
        self.verticalLayoutWidget.setGeometry(QRect(100, 200, 870, 1500))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")

        self.verticalLayout = QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")

        # creating the button that you press to add a file and adding it to
the vertical layout
        self.pushButton = QPushButton(self.verticalLayoutWidget)
        self.pushButton.setObjectName("pushButton")
        self.pushButton.setText("+")
        self.pushButton.setFont(QFont('Arial', 15))
        self.pushButton.clicked.connect(self.get_file_path)
        self.verticalLayout.addWidget(self.pushButton)

        # creating a label and adding it to the vertical layout
        self.title_label = QLabel(self.verticalLayoutWidget)
        self.title_label.setText("files:")
        self.title_label.setFont(QFont('Arial', 15))
        self.verticalLayout.addWidget(self.title_label)

        # creating the area that you add the files to in the GUI
        self.scrollArea = QScrollArea(self.verticalLayoutWidget)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")

        self.scrollAreaWidgetContents = QWidget()
        self.scrollAreaWidgetContents.setGeometry(QRect(0, 0, 287, 153))

self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")

        layout = QVBoxLayout(self)
        layout.addStretch()
```

```
        self.scrollAreaWidgetContents.setLayout(layout)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)

        self.verticalLayout.addWidget(self.scrollArea)

        # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
        self.okButton = QPushButton(self.verticalLayoutWidget)
        self.okButton.setObjectName("okButton")
        self.okButton.setText("OK")
        self.okButton.setFont(QFont('Arial', 15))
        self.okButton.clicked.connect(self.ok_button_clicked)
        self.verticalLayout.addWidget(self.okButton)

    def add_file(self, file_path):
        # adding the file to the list of files updating the GUI with the
file
        self.files.append(file_path)
        self.label = QLabel(file_path)
        self.scrollAreaWidgetContents.layout().addWidget(self.label)

    def get_file_path(self):
        # opening the file system and getting the location of the file i
choose
        file_app = QFileDialog(self)
        file_app.fileSelected.connect(self.add_file)
        file_app.setFixedSize(1000, 2000)
        file_app.show()

    def ok_button_clicked(self):
        self.finished.emit(self.files)
```

המסך השלישי:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys

class Window3(QWidget):
    # this is the window where you pick locations for the files the other
device chose
    all_files_have_location = pyqtSignal(dict)

    def __init__(self):
        super(Window3, self).__init__()
        self.file_location_dict = {}
        self.setupUi()

    def setupUi(self):
        self.setObjectName("Form")

        # creating the vertical layout and the actual widget
        self.verticalLayoutWidget = QWidget(self)
        self.verticalLayoutWidget.setGeometry(QRect(100, 100, 900, 1500))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")

        self.verticalLayout = QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
```

```
        # creating a label and adding it to the vertical layout
        self.label = QLabel(self.verticalLayoutWidget)
        self.label.setObjectName("label")
        self.label.setText("Files:")
        self.label.setFont(QFont('Arial', 15))
        self.verticalLayout.addWidget(self.label)

        # creating the area where the files that the other device chose
will be
        self.scrollArea = QScrollArea(self.verticalLayoutWidget)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")

        self.scrollAreaWidgetContents = QWidget()
        self.scrollAreaWidgetContents.setGeometry(QRect(0, 0, 417, 255))

self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")

        layout = QFormLayout(self.scrollAreaWidgetContents)

        self.scrollAreaWidgetContents.setLayout(layout)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.verticalLayout.addWidget(self.scrollArea)

        # creating the button that you press when you're finished choosing
files and adding it to the vertical layout
        self.ok_button = QPushButton(self.verticalLayoutWidget)
        self.ok_button.setObjectName("ok_button")
        self.ok_button.setText("OK")
        self.ok_button.setFont(QFont('Arial', 15))
        self.ok_button.clicked.connect(self.check_all_files_have_location)
        self.verticalLayout.addWidget(self.ok_button)

    def select_directory(self, file_name):
        # creating the object of the file system
        options = QFileDialog.Options()
        options |= QFileDialog.ReadOnly
        file_dialog = QFileDialog(self, options=options)
        # setting the object to a mode in which you choose a directory
instead of a file
        file_dialog.setFileMode(QFileDialog.Directory)
        file_dialog.setOption(QFileDialog.ShowDirsOnly, True)
        file_dialog.setFixedSize(1000, 2000)
        file_dialog.show()
        try:
            # it tries to see if you chose a directory or just closed the
window
            # if you chose a directory it would work and if not the try
except will catch it
            if file_dialog.exec_():
                directory = file_dialog.selectedFiles()[0]
            self.file_location_dict[file_name] = directory
        except:
            pass

    def check_all_files_have_location(self):
        files_have_location = True
        # it goes over each file and checking if it has a location or is it
a None type
        for location in self.file_location_dict.values():
```

```
            if location == None:
                files_have_location = False
        if files_have_location:
            # if all files have a location then it emits a dictionary with
the files and their location
            self.all_files_have_location.emit(self.file_location_dict)

    def add_files(self, files):
        # here you add files to this window
        for file in files:
            # for each file it extracts the file name insert it to the dict
and adds a label for it in the GUI
            file = file.split("/")[-1]
            self.file_location_dict[file] = None
            label = QLabel(file)
            button = QPushButton()

button.clicked.connect(self.create_select_directory_function(label.text()))
            button.setText("Choose Location")
            self.scrollAreaWidgetContents.layout().addRow(label, button)

    def create_select_directory_function(self, label_text):
        return lambda: self.select_directory(label_text)
```

מסך ההודעות:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys

class MessageWindow(QWidget):
    # this is the message window
    def __init__(self, message):
        super(MessageWindow, self).__init__()
        self.message = message
        self.setupUI()

    def setupUI(self):
        # we create the label that holds the message
        self.message_label = QLabel(self)
        self.message_label.setText(self.message)
        self.message_label.setFont(QFont('Arial', 10))
        self.message_label.setGeometry(QRect(300, 300, 700, 100))

    def change_message(self, message):
        # with this function we can change that message
        self.message_label.setText(message)
```

המסך הראשי:

```
#
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from message_win import *
from window1_phone import *
from window2_phone import *
from window3_phone import *
```

```
import time


class MainWindow(QMainWindow):
    # this is the window that supervises the GUI
    def __init__(self):
        super(MainWindow, self).__init__()
        self.path = "qr.png"
        self.setGeometry(0, 0, 1050, 2000)
        self.setWindowTitle("project")
        self.setupUI()

    def setupUI(self):
        self.current_index = 0

        # here i create the windows
        self.window1 = Window1()
        self.window2 = Window2()
        self.window3 = Window3()
        self.message_win = MessageWindow("")

        # and add them to the stack
        self.Stack = QStackedWidget(self)
        self.Stack.setGeometry(0, 0, 1050, 2000)
        self.Stack.addWidget(self.window1)
        self.Stack.addWidget(self.window2)
        self.Stack.addWidget(self.window3)
        self.Stack.addWidget(self.message_win)

        # with the stack i can replace which window is being presented to
the user

        self.hbox = QHBoxLayout(self)
        self.hbox.addWidget(self.Stack)

        self.Stack.setCurrentIndex(self.current_index)

    def change_win(self):
        # this function changes the window to the next one
        self.current_index += 1
        self.Stack.setCurrentIndex(self.current_index)

    def change_to_message_win(self, message):
        # this function changes the window to the message window
        self.message_win.change_message(message)
        self.Stack.setCurrentIndex(3)
```

<u>קובץ שבוא המחלקות האחראיות על התקשורת:</u>

```
#
import time
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import socket
import pickle
import os
import time
from cryptography.fernet import Fernet
```

```
class MainSendingSocket(QThread):
    got_file_list = pyqtSignal(list)
    ready_to_send = pyqtSignal()
    send_massage = pyqtSignal(str)
    done_signal = pyqtSignal()
    exception_rose = pyqtSignal(str)


    def __init__(self, ip, port, key):
        super(MainSendingSocket, self).__init__()
        # here i set up all the important information for the connection
        self.ip = ip
        self.port = port
        self.sending_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.encrypting_object = Fernet(key)

        self.mutex = QMutex()
        self.condition = QWaitCondition()

        # here i connect the signals of this class to a slot
        self.got_file_list.connect(self.got_files)
        self.ready_to_send.connect(self.ready_to_send_files)
        self.send_massage.connect(self.send)
        self.done_signal.connect(self.done)
        self.done_condition = False
        self.files = []


    def run(self):
        self.mutex.lock()
        # here it connects to the phone and then waits until it has the
list of files to send
        self.connect_to_phone()
        self.condition.wait(self.mutex)

        # it converts the file list to bytes and sends it
        serialized_file_list = pickle.dumps(self.files)
        try:

self.sending_socket.send(self.encrypting_object.encrypt(serialized_file_lis
t))

            # here it wait again until the computer is ready to start
sending the files
            self.condition.wait(self.mutex)
            self.sending_socket.send(self.encrypting_object.encrypt("ready
for files".encode()))

            while True:
                # here it waits until the project tells it to send a
message
                # the message is that either a socket opened or that a
socket connected
                self.condition.wait(self.mutex)

self.sending_socket.send(self.encrypting_object.encrypt(self.message.encode
()))
                if self.done_condition:
```

```
                    break

        except Exception as exception:
            self.exception_rose.emit(str(exception))

        self.mutex.unlock()

        self.sending_socket.close()

    def connect_to_phone(self):
        # this function connects to the socket on the phone
        try:
            self.sending_socket.connect((self.ip, self.port))
        except Exception as exception:
            self.exception_rose.emit(str(exception))

    def got_files(self, files):
        # when the program will have the files a signal will be emitted to
this slot and this will run
        # this function sets the file list and tells the program to stop
waiting and continue
        self.files = files

        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

    def ready_to_send_files(self):
        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

    def send(self, message):
        self.mutex.lock()
        self.message = message
        self.condition.wakeAll()
        self.mutex.unlock()

    def done(self):
        self.done_condition = True

class FileSendingSocket(MainSendingSocket):
    def __init__(self, ip, port, file_path, key):
        super(FileSendingSocket, self).__init__(ip, port, key)
        self.file_path = file_path
        self.BUFFER_SIZE = 1024

    def run(self):
        self.connect_to_phone()

        file_name = self.file_path.split("/")[-1]
        try:

self.sending_socket.send(self.encrypting_object.encrypt(str(file_name).enco
de()))

            with open(self.file_path, "rb") as f:
                while True:
                    bytes_read = f.read(1024)
                    if not bytes_read:
                        # file transmitting is done
```

```
                        break
                    # we use sendall to assure transmission in
                    # busy networks
                    encrypted_bytes =
self.encrypting_object.encrypt(bytes_read)
                    size = len(encrypted_bytes)
                    self.sending_socket.send(str(size).encode())
                    message = self.sending_socket.recv(1024)
                    self.sending_socket.send(encrypted_bytes)

        except Exception as exception:
                self.exception_rose.emit(str(exception))

        self.sending_socket.close()


class MainReceivingSocket(QThread):
    connection_made = pyqtSignal()
    got_file_list_from_phone = pyqtSignal(list)
    ready_for_files = pyqtSignal()
    receive = pyqtSignal(str) #wrong name
    done_signal = pyqtSignal()
    exception_rose = pyqtSignal(str)



    def __init__(self, ip, port, key=""):
        super(MainReceivingSocket, self).__init__()
        self.ip = ip
        self.port = port
        self.done_signal.connect(self.done)
        self.done_condition = False
        self.receiving_socket=None
        if key != "":
            self.encrypting_object = Fernet(key)

    def run(self):
        self.handle_connection()
        try:
            serialized_file_list =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024))
            list_of_files = pickle.loads(serialized_file_list)

            self.got_file_list_from_phone.emit(list_of_files)

            message =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024)).decode()
            if message == "ready for files":
                self.ready_for_files.emit()

            while True:
                message =
self.encrypting_object.decrypt(self.receiving_socket.recv(1024)).decode()
                self.receive.emit(message)  # add signal connected
                if self.done_condition:
                    break

        except Exception as exception:
            self.exception_rose.emit(str(exception))

        self.receiving_socket.close()
```

```
    def handle_connection(self):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.bind((self.ip, self.port))
        sock.listen()
        try:
            self.receiving_socket, self.address = sock.accept()
        except Exception as exception:
                self.exception_rose.emit(str(exception))

        sock.close()
        self.handle_address()

    def handle_address(self):
        self.connection_made.emit()

    def done(self):
        self.done_condition = True

    def add_encrypting_object(self, key):
        self.encrypting_object = Fernet(key)

class FileReceivingSocket(MainReceivingSocket):
    def __init__(self, ip, port, files_and_paths, key):
        super(FileReceivingSocket, self).__init__(ip, port, key)
        self.file = None
        self.BUFFER_SIZE = 1464
        self.files_and_paths = files_and_paths
        self.finished = False

    def run(self):
        self.handle_connection()
        error_line = 0
        try:
            file_name =
self.encrypting_object.decrypt(self.receiving_socket.recv(self.BUFFER_SIZE)
).decode()
            location = self.files_and_paths[file_name]
            time.sleep(1)

            with open(f"{location}/{file_name}", "wb") as file:
                while True:
                    error_line = 0
                    # read 1024 bytes from the socket (receive)
                    size = self.receiving_socket.recv(1024)
                    self.receiving_socket.send(size)
                    size = int(size.decode())
                    bytes_encrypted = self.receiving_socket.recv(size)

                    error_line += 1
                    bytes_read =
self.encrypting_object.decrypt(bytes_encrypted)

                    error_line += 1
                    if not bytes_read:
                # nothing is received
                # file transmitting is done
                        break
            # write to the file the bytes we just received
                    error_line += 1
                    file.write(bytes_read)
```

```
        except Exception as exception:
            self.exception_rose.emit(str(exception))
        self.finished = True
        self.receiving_socket.close()

    def handle_address(self):
        pass
```

אובייקט הפרויקט האחראי על הכל:

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from GUI_phone import *
from Connection_phone import *
import netifaces
from threading import Thread
import time
from cryptography.fernet import Fernet


class Project:
    def __init__(self):
# Get a list of network interfaces on the device
        interfaces = netifaces.interfaces()
        # Loop through the interfaces and find the one that is active and
has an IP address
        for iface in interfaces:
            addrs = netifaces.ifaddresses(iface)
            if netifaces.AF_INET in addrs:
                self.ip = addrs[netifaces.AF_INET][0]['addr']
                if self.ip != '127.0.0.1':
                    break
        # we set all the important stuff here
        self.port = 6745
        self.g_port = 6746
        self.main_window = MainWindow()
        self.main_receiving_socket = MainReceivingSocket(self.ip,
self.port)
        self.got_files = False
        self.finished_window2 = False
        self.ready_for_the_files = False
        self.is_phone_ready_for_the_files = False
        self.mutex = QMutex()
        self.condition = QWaitCondition()
        self.files_received = []

        # here we connect all the signals from the sockets and windows
        self.main_receiving_socket.receive.connect(self.received_message)

self.main_receiving_socket.connection_made.connect(self.handle_connection)

self.main_receiving_socket.got_file_list_from_phone.connect(self.handle_fil
es)

self.main_receiving_socket.ready_for_files.connect(self.computer_ready_for_
files)


self.main_window.window1.CameraThread.got_data.connect(self.handle_data)
```

```
        self.main_window.window2.finished.connect(self.finished_window2)

self.main_window.window3.all_files_have_location.connect(self.finished_wind
ow3)


self.main_receiving_socket.exception_rose.connect(self.exception_rose)

        self.main_receiving_socket.start()

    def exception_rose(self, error_message):
        # this function gets called in case of an exception
        self.main_window.change_to_message_win(error_message)

    def handle_data(self, data):
        # scan the qr code and we need to connect the the computer
        # the data we get is from the qr code
        data = data.split()
        self.computer_ip = data[0]
        self.computer_port = int(data[1])
        self.key = data[2].encode()
        self.main_receiving_socket.add_encrypting_object(self.key)
        # we start the sending socket so it will connect to the computer
        self.main_sending_socket = MainSendingSocket(self.computer_ip,
self.computer_port, self.key)

self.main_sending_socket.exception_rose.connect(self.exception_rose)
        self.main_sending_socket.start()

    def handle_connection(self):
        self.main_window.change_win()

    def handle_files(self, list_of_files):
        # this handles the file list from the computer and adds it to the
GUI
        self.files_from_computer = list_of_files
        self.main_window.window3.add_files(self.files_from_computer)
        self.got_files = True
        if (self.finished_window2):
            self.main_window.change_win()

    def computer_ready_for_files(self):
        # this gets called when the computer sends a message that it is
read to send the files
        self.is_phone_ready_for_the_files = True
        if self.ready_for_the_files:
            # if we are also ready to send the files the file sending
starts
            self.main_window.change_to_message_win("transferring files")
            thread = Thread(target=self.send_files)
            thread.start()
        else:
            pass

    def send_files(self):
        self.file_sending_sockets = []
        self.file_recving_sockets = []
        self.mutex.lock()
        i=0
        # receiving the files
        self.sockets = ["" for f in self.files_and_paths.keys()]
```

```
        for file in self.files_and_paths:
            time.sleep(1)
            # opening a listening socket
            self.sockets[i] = FileReceivingSocket(self.ip, self.g_port,
self.files_and_paths, self.key)
            self.g_port+=1
            self.sockets[i].start()
            self.file_recving_sockets.append(self.sockets[i])
            # telling the other device we opened a socket and he can
connect to it
            self.main_sending_socket.send_massage.emit("socket opened")
            # waiting for the other device to tell us he opened connected
to our socket
            self.condition.wait(self.mutex)
            i += 1

        # sending the files
        for file in self.files:
            # waiting for the other device to tell us he opened a socket
and is listening on it
            self.condition.wait(self.mutex)
            # connecting to that socket
            sock = FileSendingSocket(self.computer_ip, self.g_port, file,
self.key)
            self.g_port+=1
            sock.start()
            self.file_sending_sockets.append(sock)
            # telling the other device we connected to the socket and he
can open another one
            self.main_sending_socket.send_massage.emit("connected")
            time.sleep(1)

        # emitting to the main sockets that we finished transferring so
they can close
        self.main_sending_socket.done_signal.emit()
        self.main_receiving_socket.done_signal.emit()


        # making sure all receiving socket have finished
        all_socket_finished = True
        while True:
            for socket in self.sockets:
                # going through each socket and making sure it's finished
                if not socket.finished:
                    all_socket_finished = False
            if all_socket_finished:
                break
            all_socket_finished = True

        # updating the GUI that file transfer went well
        self.main_window.change_to_message_win("all files received
successfully")

        self.mutex.unlock()

    def finished_window2(self, files):
        # this happens when the second window finishes. it updates the
files
        self.files = files
        self.finished_window2 = True
        # if the computer already sent the file than we can move to the
```

```
next window and if not we wait for him
        if (self.got_files):
            self.main_window.change_win()
        else:
            self.main_window.change_to_message_win("waiting for computer")
        # sending to the computer the files we chose
        self.main_sending_socket.got_files(self.files)

    def finished_window3(self, files_and_paths):
        # this happens when the third window is finished
        # we update the locations of the files the computer sent
        self.files_and_paths = files_and_paths
        # sending that we are ready to send the files
        self.main_sending_socket.ready_to_send_files()
        self.ready_for_the_files = True
        # if the computer already sent that he is ready to send files we
start sending files
        # if not we wait for it
        if self.is_phone_ready_for_the_files:
            self.main_window.change_to_message_win("transferring files")
            thread = Thread(target=self.send_files)
            thread.start()
        else:
            self.main_window.change_to_message_win("waiting for computer")

    def received_message(self, message):
        # this get called when we receive a message during the file sending
        self.mutex.lock()
        self.condition.wakeAll()
        self.mutex.unlock()

def main():
    app = QApplication(sys.argv)
    project = Project()
    project.main_window.show()

    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```