בית הספר הרב תחומי
רשת תיכונים ומכללות מקבוצת עמל
פתח תקוה ב'
ע"ש ליידי דשיס

Signatural

שם : מיכל אופנהיימר.

ת.ז : 326195542.

שם המנחה : גולן מור.

חלופה : הגנת סייבר ומערכות הפעלה.

יוני 2022

תוכן עניינים

תקציר ורציונל הפרויקט 3
מבוא ורקע כללי 4
מטרת הפרויקט 5
שפות התכנות וסביבת העבודה 7
ניסוח וניתוח הבעיה האלגוריתמית 8
תיאור אלגוריתמים קיימים 8
הפתרון הנבחר 10
פיתוח הפתרון בשכלול עם שפת התכנות 14
תיאור המודלים של מערכת התוכנה 17
תיעוד הקוד 18
השוואת העבודה עם פתרונות ויישומים קיימים 26
הערכת הפתרון לעומת התכנון והמלצות לשיפור 26
תיאור ממשק המשתמש - הוראות הפעלה 27
מבט אישי על העבודה ותהליך הפיתוח 28
ביבליוגרפיה 29
קוד התוכנית 30

תקציר ורציונל הפרויקט

בעולם בו קיים מאגר חתימות, וכל אזרח במדינה מזוהה עם חתימה השמורה בו, ישנו צורך בחתימה על מסמכים מטעם משרדים ממשלתיים, בסגירת עסקאות כלכליות ומשפטיות ועוד. עולם עבירות הצווארון הלבן הוא די רחב, בו רבים מזייפים מסמכים וחתימות ובכך עוברים על החוק. גם עצם ההגעה הפיזית למקום העסקה רק על מנת לחתום על מסמך מאלצת אנשים לפנות זמן רב לכך וכרוכה במאמץ פיזי, במיוחד אם מדובר בעסקאות בינלאומיות. כדי לצמצם את בעיה זו ולהקשות על הזיוף והפשע, נדרשת מערכת לשליחת מסמכים וחתימה עליהם, תוך זיהוי זיופים של חתימות, ובדיקת אותנטיות של מסמכים.

הפרויקט שלי נועד לעשות בדיוק את זה. "Signatural" תשמש לחתימה בכתב יד על מסמכים תוך בדיקה האם החתימה אותנטית, כלומר, שייכת לאותו בן אדם החותם או שהינה מזויפת. התוכנה תאפשר העלאת מסמכים ושליחתם לנמען מסוים, שיחתום עליהם. התוכנה תאשר את אותנטיות החתימה באמצעות השוואת תמונות של חתימות ע"י מודל מתמטי. במידה והחתימה נמצאה מזויפת, המערכת תתריע על כך ולא תאפשר את המשך העסקה. במידה ולא, המסמך החתום ישלח אל המוען. התוכנה תיצור פרופילי משתמשים עם מידע עליהם והחתימות שלהם, ותיצור סטטיסטיקות מתאימות.

מדוע בחרתי לעשות את הפרויקט בנושא הזה?

בחרתי לעשות את פרויקט הגמר שלי במגמת הנדסת תוכנה בנושא זה מכמה סיבות שונות. ראשית כל, רציתי לאתגר את עצמי ולעשות על משהו שיעניין אותי. ידעתי שאם הנושא לא יעניין אותי לא אהנה מהתהליך ואהיה חסרת מוטיבציה. לכן בחרתי לעסוק בדברים שפחות עסקנו בהם כחלק מתוכנית הלימודים, כך שהפרויקט יהיה כרוך בלמידה עצמית וחקירה של הנושא. פחות התעמקנו בתחום הבינה המלאכותית במהלך שלוש השנים במגמה ולכן, כתלמידה סקרנית שאוהבת תכנות, רציתי לעשות פרויקט שעוסק בתחום.

בנוסף, כשהתבקשנו לבחור נושא לפרויקט הייתי בזמן צפייה בסדרת טלוויזיה בשם "צווארון לבן", שעסקה, בין היתר, בפענוח תעלומות מהסוג הזה: זיוף של חתימות ויצירות אומנות. שני הדברים התקשרו לי, וחשבתי שיהיה מעניין לכתוב תוכנה שתעשה בדיוק את זה – תזהה זיופים של חתימות. באותו זמן הרעיון היה נראה רחוק ונשגב, אך לאחר מעט מחקר הבנתי שהדבר אפשרי. רציתי לשלב בין שני עולמות שאהבתי, ולכן הגיתי את הנושא הנ"ל.

מבוא ורקע כללי

השימוש בחתימות מופיע בחיינו מדי יום ויום. בין אם במשרדים ממשלתיים, בעבודה, בבית הספר ואפילו בסופרמרקט. חתימתו של אדם היא תנאי הכרחי לסגירת עסקאות שונות וכניסתן לתוקף. החתימה מהווה התחייבות לכך שאותו אדם הסכים לתנאי העסקה ואינו הולך לחזור בו, או להכחיש את מעורבתו. עם זאת, בכל הדוגמאות המדוברות, נדרש האדם להגיע פיזית למקום שנקבע רק כדי להעניק את חתימתו. לעיתים נדרש מהאדם להשקיע מאמצים כבירים רק כדי להגיע לפגישה, שמסתיימת לאחר פחות מחמש דקות. כמו כן, בתקופת הקורונה נוכחנו לגלות כמה חשובה היכולת לבצע דברים מהבית, מבלי לבוא במגע פיזי. בנוסף, לא פעם ופעם הצליחו אנשים לזייף את חתימותיהם של אחרים ובכך לגרום להם, ולכל המעורבים בעסקה, לצרות. בעולם בו עולה הדרישה להגנה על הזהות הפרטית של אדם, אנו שואפים ליצור תוכנה שתשרת אותנו בתחום זה כראוי.

כיום קיימות בשוק מערכות שונות לזיהוי של זיוף חתימות, שלרוב מעורבות בפעולות בנקאיות שונות, סריקת דרכונים, בהצבעה בבחירות וכדומה, אך יש להן חיסרון אחד – החתימה צריכה להיעשות במקום הפיזי, בזמן אמת. כמו כן, ישנם תחומי חיים נוספים שיכולים ואף צריכים גישה לאותן תוכנות. כאמור, החתימות מלוות אותנו בחיי היומיום הבסיסיים. לכן יש למצוא דרך לאפשר את ביצוע העסקאות המדוברות גם למוסדות פרטיים ולא רק למוסדות ממשלתיים או מדיניים, כמו גם לאפשר את ביצוע העסקה מרחוק תוך זיהוי זיופים מהיר.

טכנולוגיית עיבוד תמונות וניתוח מידע מבני היא אוסף שיטות פעולה על תמונות במטרה לקבל תוצר משופר שלהן או להוציא מהן מידע שימושי. בתחום זה מעורבים לעיתים תחומים נוספים, כגון בינה מלאכותית ולימוד מכונה. הדבר נעשה ע״י לימוד המחשב באמצעות איסוף נתונים, ניתוח והסקת מסקנות באמצעות אלגוריתמים מתמטיים וסטטיסטיים.

בפרויקט זה נעשה שימוש בטכנולוגיה זו מתוך שאיפה להקל על המשתמשים במערכת ולתת מענה לבעיות שהוצגו לעיל. החתימות איתן התוכנה עובדת הינן דיגיטליות (כלומר מצוירות על המחשב) בפורמט של תמונה, כדי לאפשר את השימוש בתוכנה מרחוק וזיהוי זיופים.

מטרת הפרויקט

המוצר נועד לוודא אותנטיות של חתימות וקבצים ולאפשר סגירת עסקאות בטוחות מרחוק. המערכת שפיתחתי תאפשר שליחת קבצים ממשתמש אחד לשני לפי בקשתם ותמתין לקבלת אישור, או סירוב. לאחר קבלת אישור היא תשלח את הקובץ, תאפשר חתימה עליו ותבחן את החתימה. במידה והחתימה תמצא מזויפת המערכת תתריע על כך ולא תאפשר להמשיך בתהליך. במידה ולא, הקובץ החתום נשלח למוען ובכך העסקה מושלמת.

התוכנה בונה פרופיל משתמש לכל לקוח, שומרת את החתימה שלו בהרשמה הראשונית, מנטרת את הפעילות שלו ומציגה סטטיסטיקה מתאימה.

קהל היעד של התוכנה הוא עובדים במשרה ציבורית ומוסדות ממשלתיים, משפטנים ועורכי דין, כלכלנים, בעלי חברות ועסקים פרטיים וכלל האזרחים שעושים שימוש במסמכים וחתימות.

<u>דרישות מרכזיות:</u>

דרישות פונקציונליות:

1. התוכנה תאפשר למשתמש להעלות קבצים ולשלוח אותם למשתמש נוסף כרצונו. התוכנה תקבל אישור מהמשתמש השני לשם כך.
2. בלחיצת כפתור התוכנה תאפשר למשתמש השני לחתום על המסמך.
3. התוכנה תוודא שהחתימה אותנטית. אם החתימה מזויפת היא תתריע לשני המשתמשים ולא תאפשר להמשיך בתהליך. אם לא, המערכת תצרף אותה למסמך ותשלח אותה בחזרה למוען.
4. ההתרעה תוצג על גבי המסך.
5. המערכת תנטר את מספר הזיופים של כל משתמש ותציג לו סטטיסטיקה מתאימה בדף הבית.

דרישות לא פונקציונליות:

1. התוכנה תהיה נגישה מכל מחשב אישי (לא טלפון חכם) בהינתן רשת אלחוטית או חיבור אינטרנט.
2. שרת התוכנה יעבוד עם מספר בלתי מוגבל של משתמשים במקביל.
3. קצב הגלישה יהיה מהיר (תיתכן איטיות קלה במספר משתמשים גבוה) ונטול הפרעות.
4. ממשק התוכנה יהיה ידידותי וקל לשימוש.

אילוצים:

התוכנה עובדת עם Windows.

התוכנה תומכת אך ורק בקבצי PDF.

התוכנה פועלת במחשבים בעלי כתובת IPv4.

<u>תרחישים במערכת:</u>

1. שימוש ראשוני ובניית פרופיל משתמש:

- המשתמש ירשם למערכת ע"י הזנת שם משתמש, סיסמה וכתובת מייל.
- המשתמש יתבקש להפעיל את חשבונו ע"י הזנת קוד שנשלח למייל.
- המשתמש יזין את חתימתו בחלון ציור שיפתח.

2. שליחת קבצים:

- המשתמש יתחבר לפרופיל שלו ע"י הזנת שם משתמש וסיסמה.
- מבין שני כפתורים, המשתמש ילחץ על "שליחת קבצים".
- המשתמש יזין את שם המשתמש אליו הוא רוצה לשלוח קבצים וימתין לאישור ממנו.
- לאחר קבלת אישור יבחר את הקובץ שעל המחשב אותו הוא רוצה לשלוח. הקובץ יישלח למשתמש הנבחר.
- אם החתימה של הנמען נמצאה מזויפת מופיעה התרעה על המסך שאז חוזר לדף הבית. אם לא, מתקבל הקובץ החתום ומסך שמירה.

3. קבלת קבצים וחתימה:

- המשתמש יתחבר לפרופיל שלו ע"י הזנת שם משתמש וסיסמה.
- מבין שני כפתורים, המשתמש ילחץ על "קבלת קבצים" וימתין.
- המשתמש יקבל התרעה עם שם המשתמש שרוצה לשלוח לו קבצים. הוא יוכל לבחור אם לאשר את הבקשה או לא. אם כן, מתקבל הקובץ הרצוי. אם לא, נשלחת הודעה למשתמש השולח.
- במקרה והבקשה אושרה, יפתח מסך עם הקובץ הרצוי. בלחיצה על כפתור "חתימה" חפתח חלון ציור בו יוכל להזין את חתימתו. אם מתגלה שהיא מזויפת מופיעה התרעה על המסך שאז חוזר לדף הבית. אם לא, הקובץ החתום נשלח בחזרה.

4. שינוי הגדרות ופרטי משתמש:

כפתור "שכחתי סיסמה" במסך הפתיחה, בהזנת מייל המשתמש נשלח לו קוד שאותו יצטרך להזין במערכת. לאחר ההזנה יוכל המשתמש לאפס את סיסמתו, ליצור חדשה ולהתחבר מחדש.

שפות התכנות וסביבת העבודה

Python : שפת התכנות העיקרית בה השתמשתי בפרויקט היא python תוך שימוש בסביבת הרצה של Visual Studio Code. פייתון זוהי שפת תכנות עילית, מונחית עצמים, המאפשרת עבודה נוחה, בעיקר בשל תמיכתה הנרחבת במודלים וספריות שונות. בשפה זו עושים שימוש עיקרי בניתוח מידע וחקר נתונים, ולכן השימוש בה התאים לצורכי הפרויקט. כמו כן, היא מאפשרת התממשקות קלה עם קבצים מסוגים שונים ובסיסי נתונים.

C#: שפת התכנות בה כתבתי את ממשק המשתמש שלי בפרויקט היא C# תוך שימוש בסביבת הרצה של Visual Studio בפלטפורמת פיתוח של Net Framework. שפה זו היא שפת תכנות עילית ומונחית עצמים גם כן, בה ניתן לכתוב ממשקי משתמש לתוכנות שרצות במערכת ההפעלה Windows (winforms). בחרתי לכתוב את ממשק המשתמש שלי בשפה זו מפני שסביבת הפיתוח ידידותית למפתח ומקלה על יצירת הממשק. בנוסף, האפשרויות שהיא מציעה מגוונות ומתאימות לנראות הממשק שרציתי בפרויקט, שמסיבות אחרות גם ככה רץ רק על מערכת ההפעלה הזו.

ניסוח וניתוח הבעיה האלגוריתמית

בעיה 1 : הפרויקט מסתמך על בחינת תמונה של חתימה והשוואה בינה לבין נוספות במטרה לזהות האם החתימה מזויפת או אמיתית. כל חתימה היא ציור ששונה ממשנהו, ועולה השאלה כיצד מזהים האם החתימה מזויפת או שהינה אמיתית, כלומר צוירה ע"י אותו אדם? יש צורך במימוש ושימוש במודלים מתמטיים מורכבים וישנה סכנה שינתן להם מענה חלקי, מה שיפגע בפעילות המערכת ויגביר את סיכוייה למצוא תוצאות שגויות (כלומר, חתימה אמיתית תוכרז כמזויפת ולהיפך).

בעיה 2 : בעיה נוספת שעולה מהפרויקט היא יצירת שרת TCP המטפל במספר גדול של משתמשים במקביל, ללא פגיעה במהירות המערכת והממשק.

בעיה 3 : יצירת תקשורת בין שני לקוחות ששולחים זה לזה קבצים.

תיאור אלגוריתמים קיימים

<u>פתרונות קיימים לבעיה 1 :</u>

תחום השוואת החתימות מתחלק לשניים : השוואה תלויית כותב (writer-dependent) ולא תלויית כותב (writer-independent). בהשוואה תלויית כותב מורכב מודל שמאמן את עצמו לפי כל משתמש בנפרד, ויוצר פרופיל מידע לגבי התכונות שבחתימתו של אדם ספציפי. כך ההשוואה מותאמת לכל אדם. בהשוואה לא תלויית כותב המודל הוא זהה לכל המשתמשים. ההשוואה בין החתימות יכולה להיות סטטית - כלומר השוואת הפיקסלים בין שתי התמונות, ודינאמית - בחינת ההבדלים בקימורים ובעיקולים שבין הציורים.

א. שימוש בלימוד מכונה ובינה מלאכותית :

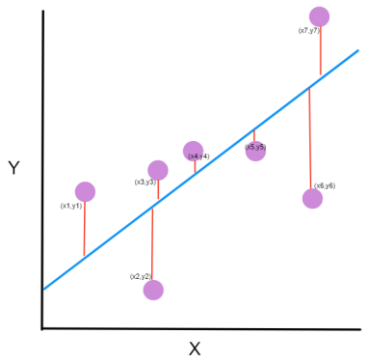
מדובר על יצירת רשת נוירונים באמצעות CNN. רשת נוירונים או רשת עצבית מלאכותית הוא מודל מתמטי חישובי, שפותח בהשראת תהליכים מוחיים ומשמש במסגרת למידת מכונה. רשת מסוג זה מכילה בדרך כלל מספר רב של יחידות מידע (קלט ופלט) המקושרות זו לזו, אלו הן "שכבות". באמצעות שימוש ברשת נוירונים מתחלקת תמונת החתימה למספר שכבות ונשלף מתוכן מידע על הוקטורים שבציור. כל שכבה מבצעת פעולה מתמטית פשוטה, ובהיקשרם יחד הן מסוגלות להתנהגות מורכבת, כגון חישובי סטייה ועיקולים בציור החתימה. לאחר יצירת רשת נוירונים כזו נדרש לימוד מכונה מסוג חיזוק כדי לאמן אותה לקבל החלטות ספציפיות - האם החתימה מזויפת או לא. המכונה נחשפת לסביבה בה היא לומדת את עצמה ללא הרף באמצעות ניסוי וטעייה. באלגוריתם זה, המכונה לומדת מניסיון העבר ומנסה לתפוס את הידע הטוב ביותר לצורך קבלת החלטות עסקיות מדויקות. בצורה זו מכניסים מספר ניסיונות של חתימות ומאמנים את המערכת לנתח אותן.

ב. ניתוח התמונות ועיבוד המידע מתוכן ע״י עבודה עם פיקסלים:

מדובר במציאת דמיון מבני בין התמונות, באמצעות מודלים מתמטיים גם כן. בהשוואה סטטית משווים את הפיקסלים שבשתי תמונות ובוחנים הבדלים ושינויים ביניהם. מתוך התוצאה מחשבים אחוזי התאמה. בתחום זה ישנם מודלים מתמטיים הבוחנים ירידה באיכות של תמונה ע״י בדיקת השינויים בין תמונת מקור ותמונה מעובדת:

Mean Squared Error - ממוצע שגיאות בריבוע. ערך MSE מציין את ההבדל הממוצע של הפיקסלים על פני כל התמונה. ערך גבוה יותר של MSE מציין הבדל גדול יותר בין התמונה המקורית והתמונה המעובדת. הנוסחה לחישוב ערך זה היא להלן:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$



בגרף משמאל, הנקודות הסגולות מייצגות את ציור החתימה המתקבל. הישר הכחול מייצג את הציור המקורי, אליו אנחנו משווים. הקווים האדומים הם מרחק הנקודות המרכיבות את הציור מהמקום בו הן צריכות להיות (y - y′). סכום ריבועי המרחקים הללו מייצג את ערך ה- MSE. כלומר, בנוסחה i=1 הוא הנקודה הראשונה, ו-n שווה למספר הנקודות שבוחנים.

Structure Similarity- מדד הדמיון המבני (SSIM) הוא מדד תפיסתי שמכמת ירידה באיכות התמונה הנגרמת על ידי עיבוד כגון דחיסת נתונים או על ידי אובדן בהעברת נתונים. ערכי SSIM נעים בין 1- ל- 1, ככל שהערך גבוה יותר התמונות דומות יותר. הנוסחה המתמטית שלו מסובכת יותר, ולמזלנו היא ממומשת כספריית פייתון שניתן להשתמש בה:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_2)}$$

<u>פתרונות קיימים לבעיה 2:</u>

ישנן כמה דרכים ליצור שרת אסינכרוני שמטפל בכמה לקוחות במקביל:

א. שימוש בספריית Asyncio. Asyncio היא ספריה לכתיבת קוד בו-זמני באמצעות הפקודות async/await. הספרייה משמשת כבסיס למסגרות אסינכרוניות מרובות של Python המספקות רשתות ושרתי אינטרנט עם ביצועים גבוהים, ספריות ומסדי נתונים וכו'.

ב. שימוש ב-threading. פתיחת thread משמעה פתיחת תהליכון בתוך התהליך שבו עובדים - הקוד. התהליכון עובד במקביל לתוכנית המרכזית עם פונקציית מטרה שניתנת לו. כך ניתן לפתוח לכל לקוח תהליכון, ולא להפריע לטיפול בלקוחות נוספים בשרת.

<u>פתרונות קיימים לבעיה 3:</u>

התקשורת בין הלקוחות יכולה להיעשות בכמה דרכים:

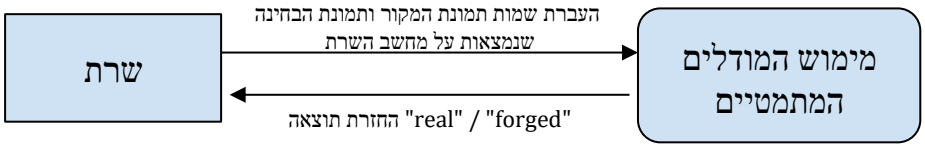
א. יצירת סוקט האזנה בפורט נפרד ללקוח, שימתין ללקוח אחר שיתחבר אליו. פתרון זו יוצר בעיה נוספת והיא כיצד מחליטים אילו לקוחות יצרו סוקט נוסף ויחכו לחיבור, ואילו לקוחות יתחברו? בנוסף, כיצד הלקוחות ידעו לאיזו כתובת להתחבר?

ב. שימוש בשרת כצד שלישי שמעביר בין הלקוחות את המידע. במקרה של משתמשים מרובים נוצרת בעיה של ארגון ומציאת המשתמש המתבקש לתקשורת עם אחד נוסף.

הפתרון הנבחר

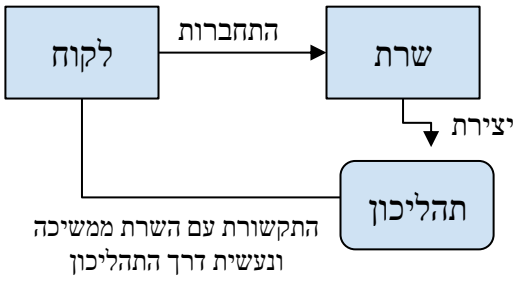
<u>פתרון בעיה 1:</u>

הפתרון שבחרתי להשתמש בו הוא הפתרון השני - עיבוד התמונות וניתוח המידע המבני בהן ע"י עבודה עם הפיקסלים. לאחר מחקר מעמיק אודות עולם רשת הנוירונים, CNN, בניית הבינה המלאכותית, ועיבוד התמונות והשוואתן הגעתי למסקנה שמטעמי זמן עדיף לי להשתמש בפתרון עיבוד התמונות ע"י מימוש המודלים המתמטיים. אמנם התוצאה שתתקבל לא תהיה מדויקת לכל חתימה ואדם בצורה מושלמת, אך היא תהיה מתאימה מספיק לדרישות הפרויקט ותשרת אותי בצורה טובה. בנוסף, פתרון זה יגרום למערכת לא להיות תלויה באימון ויהיה ניתן לעשות בה שימוש מיידי לאחר התקנתה.



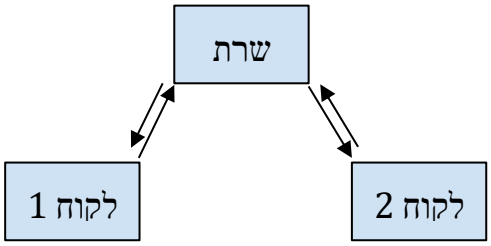
<u>פתרון בעיה 2:</u>

בחרתי להשתמש בפתרון השני - פתיחת תהליכון לכל לקוח. בחרתי בפתרון זה מפני שהשימוש בthreading מוכר לי ולא מסובך לעומת ספריית asyncio, כך שאשאיר לי זמן להתעסק עם החלקים המסובכים יותר בפרויקט. אמנם פתיחת תהליכון במחשב לוקחת זמן מסוים, אך מצאתי כי הדבר לא משפיע כל כך על מהירות הממשק והמערכת.

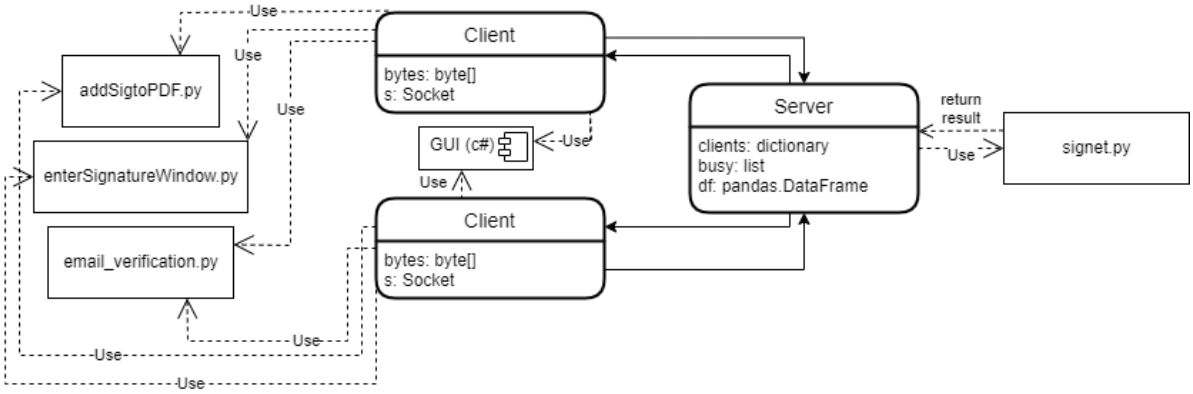


<u>פתרון בעיה 3 :</u>

כדי לפתור את בעיה מספר 3 בחרתי להשתמש בפתרון השני - שימוש בצד השרת כמתווך בין התקשורת בין שני הלקוחות. בחרתי בפתרון זה עקב הבעיות הנוספות שעולות כתוצאה מהשימוש בפתרון הראשון שציינו לעיל. זאת נראית לי הגישה הפשוטה ביותר, שלא תעורר בעיות ותקלות בחיבור. כדי לשמור על סדר במקרה של משתמשים מרובים, אשתמש באובייקט מסוג מילון כדי לשמור את הסוקט של כל לקוח אל מול שם המשתמש שלו.

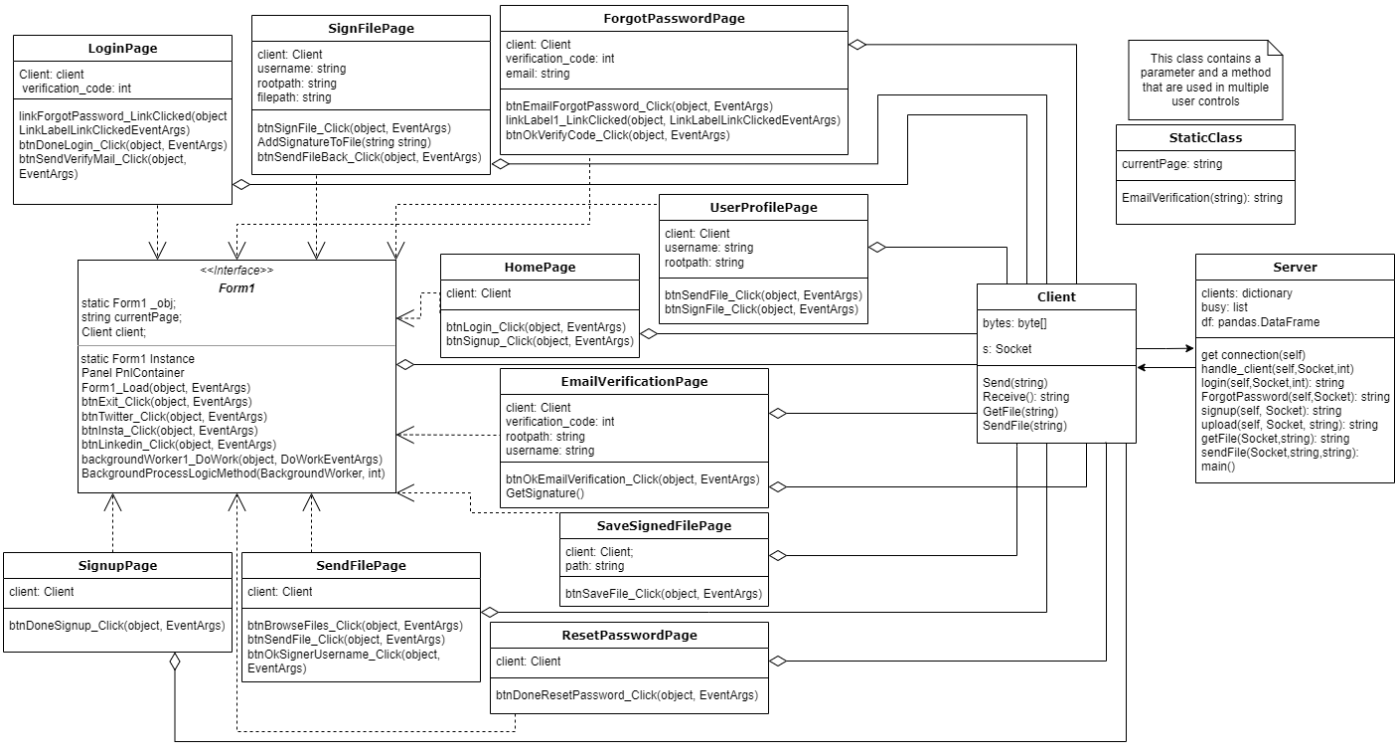


<u>מבנה המערכת :</u>



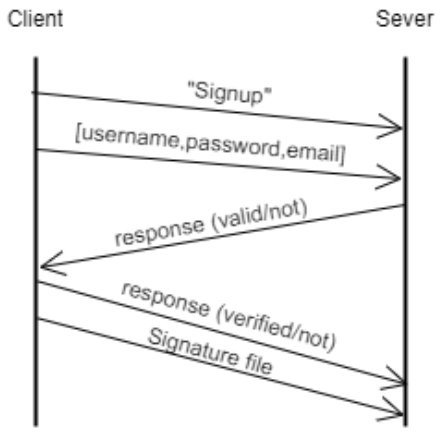
מבנה **Classes** :

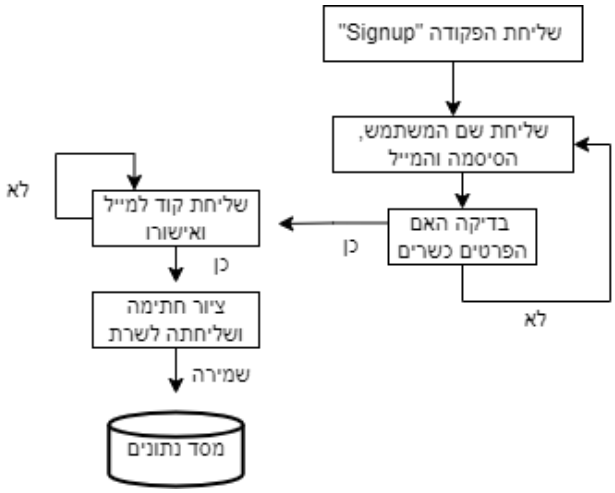
המערכת בנויה מממשק משתמש הבנוי מחלון המכיל user controls. כל אחד מהם מכיל אובייקט מסוג הקלאס Client, שמאותחל בתוך החלון ופועל במקביל לממשק (באמצעות שימוש במרכיב backgoundworker).



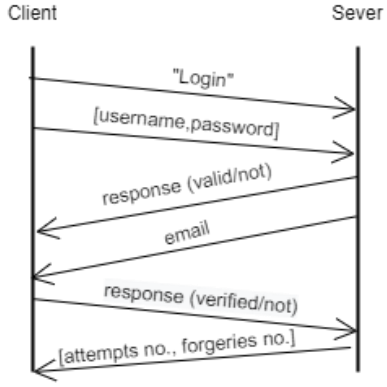
פרוטוקול תקשורת :

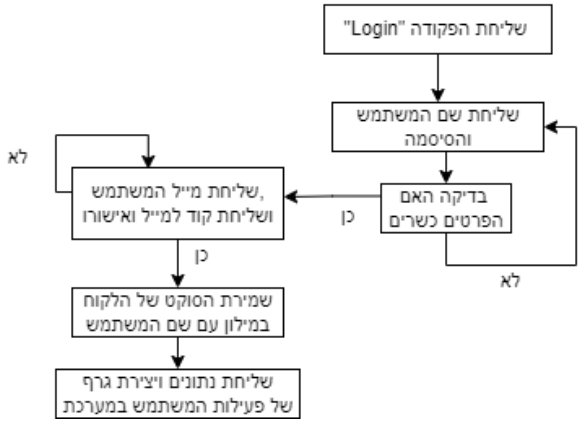
א. שימוש ראשוני ובניית פרופיל משתמש :



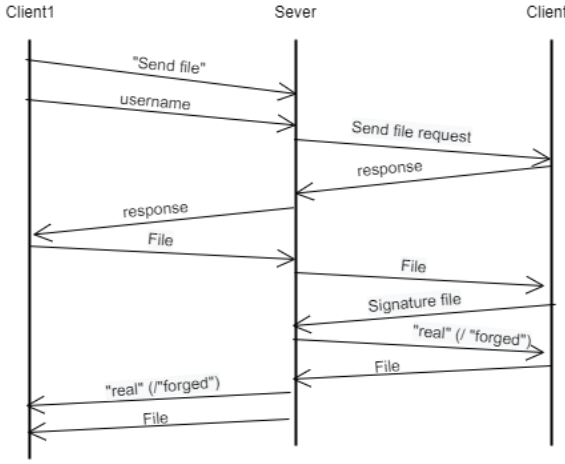


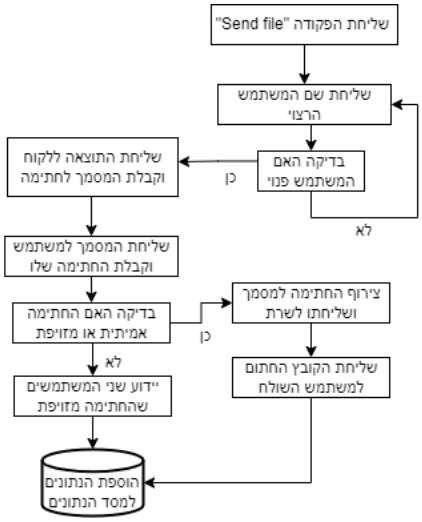
ב. התחברות למשתמש :



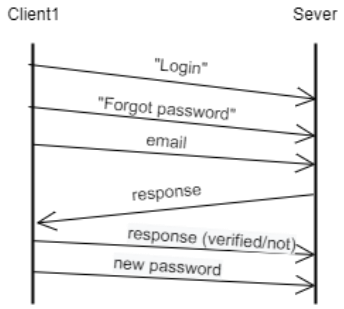


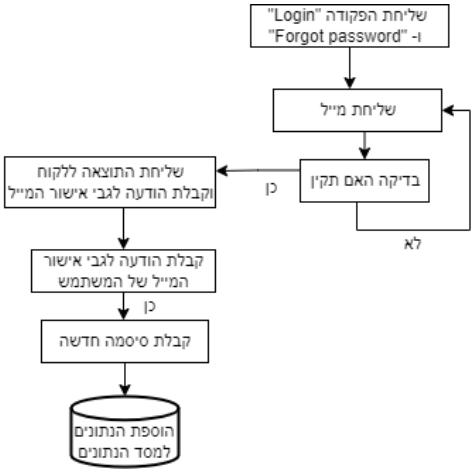
ג. שליחת קבצים וקבלתם :





ד. שינוי הגדרות ופרטי משתמש :





פיתוח הפתרון בשכלול עם שפת התכנות

מימוש המודלים המתמטיים:

ע"י המרת התמונות למערך נקודות מסוג numpy array וע"י שימוש במודלים המתמטיים שהוצעו בפתרון בעיה 1 אקבע האם החתימה מזויפת או אמיתית:

```
1. from skimage.metrics import structural_similarity
2. import numpy as np
3. from PIL import Image
4.
5. def mse(A, B):
6.     """
7.         Computes Mean Squared Error between two images.
8.         Arguments:
9.             A: numpy array of image 1.
10.            B: numpy array of image 2.
11.        Returns:
12.            err: float.
13.    """
14.
15.    # sigma(1, n-1)(a-b)^2)
16.    err = np.sum((A - B) ** 2)
17.
18.    # mean of the sum (r,c) => total elements: r*c
19.    err /= float(A.shape[0] * B.shape[1])
20.    return err
21.
22. def ssim(A, B):
23.    """
24.        Computes Structure Similarity between two images.
25.        Arguments:
26.            A: numpy array of image 1.
27.            B: numpy array of image 2.
28.        Returns:
29.            score: float.
30.    """
31.    return structural_similarity(A.flatten(), B.flatten())
32.
33. def main(signature1,signature2):
34.    # Open images as numpy arrays.
35.    real_img = Image.open(signature1)
36.    realnp = np.asarray(real_img)
37.    test_img = Image.open(signature2)
38.    testnp = np.asarray(test_img)
39.
40.    # Make sure the images have the same size.
41.    testnp.resize(realnp.shape)
42.    mse = mse(realnp,testnp)
43.    ssim = ssim(realnp,testnp)
44.    if(mse < 0.3 and ssim > 0.7)
45.        return "real"
46.    else
47.        return "forged"
```

טיפול בכמה לקוחות במקביל בצד השרת:

ע"י פתיחת תהליכון לכל לקוח ושמירת הסוקט של כל אחד במילון לצד השם שלו, אסדר את ניהול המספר הרב של הלקוחות במקביל. אעביר לה פונקציית מטרה שתמלא את כל התפקוד בצד השרת:

```
1. def get_connection(self):
2.     # Accepts connection of a client
3.     client, addr = self.socket.accept()
4.     print("got connection from: " + str(addr))
5.     client_id = len(self.clients)
6.
7.     # Creates a thread for each client and adds
8.     # them to the clients dictionary.
9.     t = threading.Thread(target=Server.handle_client
   ,args=(self,client,client_id,)).start()
10.    self.clients.update({client_id:client})
```

תקשורת בין שני לקוחות:

ע"י שימוש בשרת כצד שלישי ומתווך המעביר בין לקוחות מידע, נוכל להעביר בין לקוחות ספציפיים מידע ובכך ליצור ביניהם תקשורת. בקטע הקוד הבא מוצגת פעולת שליחת הקובץ מלקוח אחד לשני במערכת, ניתן לראות שהשרת מקבל את הקבצים ומעביר אותם הלאה, הוא עומד בין הלקוחות:

```
1. def upload(self, client, username):
2.     requested_signer = client.recv(1024).decode()
3.     # If requested user is connected to the system
4.     if(requested_signer in self.clients):
5.         if(requested_signer in self.busy):
6.             client.send("User Unavailable.".encode())
7.             return -1
8.         else:
9.             message = username + " wants you to sign a PDF file. Do you
   accept?"
10.            self.clients[requested_signer].send(message.encode())
11.            response = self.clients[requested_signer].recv(1024).decode()
12.
13.            # If signer accepts, the server gets the PDF file from the
   sender
14.            if(response == "Yes"):
15.                # Adds the signer to the busy users list.
16.                if(requested_signer not in self.busy):
17.                    self.busy.append(requested_signer)
18.
19.                client.send("Request accepted".encode())
20.                path = r"Demo\\users\\" + username + "\\"
21.
22.                # The server gets the wanted file from the sender
23.                filename = Server.getFile(client,path)
24.                print("Got file: " + filename + "from: " + username + " !")
25.
26.                # The server sends the file to the signer.
27.
   Server.sendFile(self.clients[requested_signer],filename,path)
28.                print("Sent file: " + filename + " to: " + requested_signer
   +"!")
```

```
29.
30.                 # Gets the signature attempt of the signer
31.                 signatureName =
   Server.getFile(self.clients[requested_signer],path)
32.
33.                 # Checks if the signature is real of forged
34.                 authenticity = signet.main()
35.
36.                 if(authenticity=="real"):
37.                     # Gets signed file from signer
38.                     self.clients[requested_signer].send("real".encode())
39.                     filename =
   Server.getFile(self.clients[requested_signer],path)
40.
41.                     # Sends it to the sender
42.                     client.send("real".encode())
43.                     Server.sendFile(client,filename,path)
44.                 else:
45.                     # Notifies both users that the signature is forged
46.                     # and that the deal is off.
47.                     client.send("forged".encode())
48.                     self.clients[requested_signer].send("forged".encode())
49.
50.                 return "Success!"
51.
52.             else:
53.                 client.send("Request denied".encode())
54.                 return -1
55.
56.     else:
57.         client.send("User not connected.".encode())
58.         return -1
```

תיאור המודלים של מערכת התוכנה

המערכת משתמשת במודולים שונים על מנת להריץ את האלמנטים השונים בתוכנה. כעת אפרט על כמה ממודולי המערכת:

Server.py

מחלקת השרת המתקשרת עם הלקוחות.

addSigtoPDF.py

מודל המצרף תמונה (חתימה) לתוך קובץ PDF במקום מסוים.

enterSignatureWindow.py

מודל הפותח חלון ציור ומאפשר שרטוט חתימה עליו ושמירתה כקובץ מסוג png.

signet.py

מימוש המודלים המתמטיים שהוסברו בפרק תיאור הבעיה האלגוריתמית ופתרונותיה. המודל מקבל שתי תמונות (מקורית ותמונה לבחינה) ומחזיר תשובה חד משמעית: תמונת הבחינה אמיתית או מזויפת.

email_verification.py

מודל שמטרתו ביצוע אישור מייל משתמש. המודל מגריל מספר אותו הוא מצרף למייל ומבצע שליחה שלו לכתובת שהוא מקבל כקלט. זהו הקוד אותו יצטרך המשתמש להזין בממשק המשתמש.

Client.cs

מחלקה המייצגת את צד הלקוח במערכת.

GUI.sln

אוסף של 12 מחלקות המרכיבות את ממשק המשתמש. הממשק מורכב מחלון המכיל user controls.

תיעוד הקוד

פירוט מבנה הנתונים:

שם המבנה	תפקיד המבנה
users_data	המבנה מכיל נתונים על המשתמשים הרשומים במערכת : שם משתמש, סיסמה, מייל, מספר ניסיונות חתימה, מספר זיופים שאותרו.

פירוט הפעולות:

Server.py:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
def __init__(self,ip,port)	ip, port	-	אתחול אובייקט הסרבר
get_connection(self)	-	-	קבלת חיבורים ואישורם, יצירת תהליכון לכל לקוח והכנסתו למילון לקוחות
handle_client(self,client,client_id)	הסוקט של הלקוח ומספר id	-	פונקציית המטרה של כל תהליכון, מנווטת את הלקוח לפונקציות הנכונות בשרת
login(self, client,client_id)	הסוקט של הלקוח ומספר הid שלו	שם המשתמש של הלקוח שהתחבר	מנהלת את תהליך ההתחברות למערכת
ForgotPassword(self, client)	הסוקט של הלקוח	קוד הצלחה / כישלון	מנהלת את תהליך אחזור ושינוי הסיסמה
signup(self, client)	הסוקט של הלקוח	קוד הצלחה / כישלון	מנהלת את תהליך ההרשמה למערכת
upload(self, client, username)	הסוקט של הלקוח ושם המשתמש שלו	קוד הצלחה / כישלון	מנהלת את תהליך שליחת הקבצים בין הלקוחות
getFile(sender,path)	הסוקט של הלקוח השולח קבצים והכתובת של תקיית הroot.	שם הקובץ שהתקבל	מבצעת קבלה של קובץ מלקוח אחר
sendFile(getter,filename,path)	הסוקט של הלקוח המקבל קבצים, שם הקובץ הנשלח והכתובת של תקיית הroot.	-	מבצעת שליחה של קובץ ללקוח אחר
main()	-	-	מפעילה את התוכנית

Client.cs:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
public Client(int port)	port	-	אתחול אובייקט הלקוח וחיבורו לשרת
public void Send(string msg)	ההודעה שיש לשלוח	-	מנהלת את שליחת ההודעות מהלקוח לשרת
public string Receive()	-	המידע שהתקבל	מנהלת את קבלת ההודעות מהשרת
public void GetFile(string filename)	שם הקובץ שהולכים לקבל	-	מנהלת את תהליך קבלת הקבצים מהשרת
public void SendFile(string filepath)	כתובת הקובץ שיש לשלוח	-	מנהלת את תהליך שליחת הקבצים מהלקוח לשרת
public void Close()	-	-	סוגרת את הסוקט

StaticClass.cs:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
public static string EmailVerification(string email)	האימייל לו יש לשלוח מייל	הקוד שהוגרל ונשלח במייל למשתמש	אישור מייל ושליחת קוד להזנה בממשק המשתמש

Signet.py:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
def mse(A, B)	שתי תמונות שהומרו ל numpy array	ערך ריבועי השגיאה הממוצע	מימוש המודל המתמטי
def mse(A, B)	שתי תמונות שהומרו ל numpy array	ערך הדמיון המבני בין שתי התמונות	מימוש המודל המתמטי
def main()	-	ההחלטה שהתקבלה - מזויף או אמיתי	ממירה את תוצאות המודל המתמטי לתוצאה ברורה.

addSigntoPDF.py:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
def start(file,signature):	שם הקובץ ושם הקובץ של תמונת החתימה	-	צירוף תמונת החתימה לקובץ

enterSignatureWindow.py:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
def create_window():	-	אובייקט החלון	אתחול אובייקט החלון
def mmove(event):	אירוע תזוזה על החלון	-	איתור תזוזות העכבר של החלון ושמירת הקורדינטות
def draw(event):	אירוע תזוזה על המסך תוך לחיצה על העכבר	המידע שהתקבל	ציור על המסך בהתאם לתזוזות העכבר והלחיצות עליו
def clearScreen():	-	-	ניקוי מסך הציור חזרה ללבן
def saveSignature(username):	שם המשתמש שעל שמו תשמר תמונת הסיסמה	-	שמירת ציור החתימה כתמונה בשם שהתקבל כקלטץ.
def create_canvas(root):		מחזיר את אובייקט קנבאס הציור שנמצא בתוך החלון	יצירת אובייקט ציור שנמצא בתוך החלון
def create_buttons(username):		-	יצירת הכפתורים לשמירה ולניקוי המסך
def main(username):	שם המשתמש שעל שמו תשמר תמונת הסיסמה	-	הפעלת התוכנית

email_verification.py:

כותרת הפונקציה	טענת כניסה	טענת יציאה	תפקיד
def send(email)	כתובת המייל שאליה יש לשלוח מייל	-	שליחת מייל עם קוד להזנה בממשק המשתמש

Form1.cs:

public static Form1 Instance	-	-	יצירת אינסטנס של החלון
public Panel PnlContainer	-	-	יצירת פאנל שיכיל את כל מרכיבי הממשק
public Form1()	-	-	אתחול אובייקט החלון
private void Form1_Load(object sender, EventArgs e)	אירוע הרצת התוכנית	-	מגיבה לאירוע הרצת התוכנית, מטעינה את החלון ומציגה אותו
private void btnExit_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור יציאה	-	סוגרת את החלון ואת התוכנית
private void btnTwitter_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור טוויטר	-	פותחת את אתר טוויטר בדפדפן.
private void btnInsta_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור אינסטגרם	-	פותחת את אתר אינסטגרם בדפדפן
private void btnLinkedin_Click(object sender, EventArgs e)	אירוע לחיצה של כפתור linkedin	-	פותחת את אתר linkedin בדפדפן.
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)	אירוע הפעלת מרכיב הbackgroundworker	-	מפעילה פונקציה שפועלת ברקע של הממשק, במקביל אליו
private int BackgroundProcessLogicMethod(BackgroundWorker bw, int a)	אובייקט backgroundworker ומספר	-	הפעולה שפועלת ברקע והיא מאתחלת את אובייקט הלקוח.

HomePage.cs:

public HomePage(Client theClient)	הסוקט של הלקוח	-	אתחול אובייקט ה-user control
private void btnLogin_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור login	-	מנהלת את תהליך ההתחברות למשתמש בשרת
private void	אירוע לחיצה על כפתור	-	מנהלת את תהליך

btnSignup_Click(object sender, EventArgs e)	signup		ההרשמה ויצירת המשתמש בשרת

LoginPage.cs:

public LoginPage(Client theClient)	הסוקט של הלקוח	-	אתחול אובייקט ה-user control
private void linkForgotPassword_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)	אירוע לחיצה על לינק forgot password	-	מנהלת את תהליך החלפת הסיסמה מול השרת
private void btnDoneLogin_Click (object sender, EventArgs e)	אירוע לחיצה על כפתור login	-	מנהלת את ההתחברות של המשתמש מול השרת
private void btnSendVerifyMail_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור ok	-	שולחת מייל עם קוד להזנה בממשק המשתמש

SignupPage.cs:

public SignupPage(Client theClient)	הסוקט של הלקוח	-	אתחול אובייקט ה-user control
private void btnDoneSignup_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור signup	-	מנהלת את תהליך הרשמה למערכת מול השרת

EmailVerificationPage.cs:

public EmailVerificationPage(Client theClient, string verification_code,string username)	הסוקט של הלקוח, הקוד שהוגרל במודל email_verification..py, שם המשתמש של הלקוח	-	אתחול אובייקט ה-user control
private void btnOkEmailVerification_Click(object	אירוע לחיצה על כפתור ok	-	מנהלת את תהליך שליחת המייל למייל של הלקוח

sender, EventArgs e)			
private void GetSignature()	-	-	מנהלת את פתיחת חלון החתימה וההזנה שלה

EmailVerificationPage.cs:

public UserProfilePage(Client theClient,string username,string filename)	הסוקט של הלקוח, שם המשתמש של הלקוח ושם קובץ	-	אתחול אובייקט ה-user control
private void btnSendFile_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור send file	-	מנהלת את תהליך שליחת הקובץ למשתמש רצוי
private void btnSignFile_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור sign file	-	מנהלת את תהליך החתימה על קובץ בצד הלקוח

SendFilePage.cs:

public SendFilePage(Client theClient)	הסוקט של הלקוח, שם המשתמש של הלקוח ושם קובץ	-	אתחול אובייקט ה-user control
private void btnBrowseFiles_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור browse	-	פותחת חלון בחירת קובץ מתוך כלל הקבצים במחשב
private void btnSendFile_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור send	-	מנהלת את תהליך שליחת הקובץ הנבחר לשרת
private void btnOkSignerUsername_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור ok	-	מוודאת מול השרת את זמינות המשתמש שנבחר לקבל קבצים

SaveSignedFilePage.cs:

public SaveSignedFilePage(Client theClient,string filename)	הסוקט של הלקוח ושם הקובץ	-	אתחול אובייקט ה-user control
private void btnSaveFile_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור save	-	פותחת חלון בחירת מיקום בו ישמר הקובץ מתוך כלל התיקיות במחשב

SignFilePage.cs:

public SignFilePage(Client theClient,string filepath,string username)	הסוקט של הלקוח, כתובת הקובץ ושם המשתמש של הלקוח	-	אתחול אובייקט ה-user control
private void btnSignFile_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור sign	-	פותחת חלון בו ניתן לצייר חתימה ולשמור אותה
private void AddSignatureToFile(string filepath, string signature)	כתובת הקובץ ושם קובץ תמונת החתימה	-	מנהלת את תהליך שילוב חתימת הלקוח בקובץ הרצוי
private void btnSendFileBack_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור send	-	מנהלת את תהליך שליחת הקובץ החתום לשרת

ForgotPasswordPage.cs:

public ForgotPasswordPage(Client theClient)	הסוקט של הלקוח	-	אתחול אובייקט ה-user control
private void btnEmailForgotPassword_Click(object sender, EventArgs e)	אירוע לחיצה על כפתור send	-	מנהלת את שליחת מייל האישור עם הקוד להזנה בממשק המשתמש
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)	אירוע לחיצה על לינק send mail again	-	מנהלת את שליחת המייל בשנית במקרה והוא לא נשלח כשנלחץ הכפתור
private void	אירוע לחיצה על כפתור	-	בודקת את התאמת

הקוד שהוכנס לקוד שהוגרל במייל שנשלח.		ok	btnOkVerifyCode_Click(object sender, EventArgs e)

ResetPasswordPage.cs:

אתחול אובייקט ה-user control	-	הסוקט של הלקוח	public ResetPasswordPage(Client theClient)
מנהלת את שינוי הסיסמה לאחת חדשה ושליחתה לשרת.	-	אירוע לחיצה על כפתור reset password	private void btnDoneResetPassword_Click(object sender, EventArgs e)

השוואת העבודה עם פתרונות ויישומים קיימים

כאמור, קיימים פתרונות בתחום זיהוי בחתימות ושמירה על הזהות הפרטית של אדם כיום בשוק. השימוש המערכות כאלה נעשה לרוב בממסדים ממשלתיים או בעסקים גדולים. המערכות הללו בנויות על גבי בינה מלאכותית שכל הדבר משפרת את ביצועיה, ובוחנת את החתימות בצורה קפדנית יותר לכל אדם ואדם. במערכת שלי אמנם אין את הדבר הזה, אך היא נותנת מענה לדבר אחר: מהירות ויעילות פיזית. מפני שכל חתימה נבחנת אל מול חתימה מקורית, כלומר נעשית השוואה של שתי תמונות, אין צורך בלימוד המכונה פר אדם. ניתן לעשות בה שימוש מיד עם התקנתה, והיא אינה מכבידה על המחשב. בנוסף, אחת הבעיות שציינתי בפרק המבוא הינה הצורך בהגעה פיזית למקום המוסד כדי להעניק חתימה, מדובר בדבר שנמשך שניות ספורות. בבעיה זו לוקים המוצרים הקיימים בשוק בעידן בו לעיתים נבצר מאנשים להגיע פיזית למקום מסוים, ואלמנטים יום יומיים רבים עוברים להתבצע מהבית, יש צורך בביצוע הדבר מרחוק, מה שהמערכת שלי מאפשרת. כמו כן, התוכנה שלי תהיה נגישה גם לאדם הפרטי ולכן תוכל להשתלב בחיי היומיום האזרחיים ולא רק במוסדות רשמיים.

אמנם ביצועי המערכות הקיימות בשוק טובים יותר מביצועי המערכת שלי, עקב העובדה שהן נשענות על לימוד מכונה ובינה מלאכותית ושלי לא. למרות זאת, רמת דיוק המוצר שלי מספקת למטרת הפרויקט ולזמן שהוקצב לביצועו.

הערכת הפתרון לעומת התכנון והמלצות לשיפור

את החלקים העיקריים של התכנון המקורי הצלחתי להוציא לפועל. התכנון המקורי כלל יצירת חתימה, השוואת חתימות וזיהוי האם מדובר בזיוף או בחתימה אמיתית, והוספת החתימה לתוך קובץ מתבקש. חלקים אלו הם הלב של המערכת שלי ולכן אני מרוצה מהתוצאה. עם זאת, המערכת תוכננה לאפשר בחירה של מיקום החתימה הרצוי על הקובץ, דבר שלא הספקתי לעשות טרם זמן ההגשה. בנוסף, אלמנט השוואה נוסף שרציתי לממש היה מדידת הזמן שלוקח לאדם לחתום. את הפרמטר הנוסף תכננתי לבחון בעת קבלת ההחלטה על זיוף החתימה או אמיתותה.

אלו דברים שהייתי רוצה לשפר במערכת שלי, מעבר לרמת דיוקה ע״י בניית בינה מלאכותית. בנוסף הייתי רוצה ליצור לה גם אפליקציה וכך להפוך אותה לנגישה עוד יותר. כמו כן, הופתעתי לגלות כי המודלים המתמטיים נותנים תוצאות מדויקות יותר בהשוואת תמונות חתימה ידניות ולא דיגיטליות, מה שהייתי רוצה לתקן ולשפר גם כן.

תיאור ממשק המשתמש - הוראות הפעלה

* הוראות ההפעלה נכתבו בלשון זכר בשל טעמי נוחות בלבד.

שלום חבר יקר,

באמצעות תוכנה זו נוכל להקל עליך בתהליך סגירת עסקאות וחתימה על מסמכים.

כל שעליך לעשות, הוא ליצור משתמש במערכת באמצעות הכנסת שמך, כתובת מייל

והגדרת סיסמא. לאחר מכן תתבקש להזין קוד שישלח אליך למייל, ואז יפתח חלון בו תצייר את חתימתך.

לאחר הצטרפותך, תוכל להתחבר לחשבונך באמצעות הזנת הפרטים וקוד נוסף מהמייל ולהתחיל לשלוח קבצים למי שתרצה.

כעת, בלחיצת כפתור Send File יפתח חלון בו תוכל לציין את שמו של המשתמש לו תרצה לשלוח קבצים. במידה והמשתמש יאשר את בקשתך, תוכל לבחור את הקובץ שתרצה לשלוח לו, וללחוץ "Send". במידה ולא, תוחזר לדף הראשי. לאחר שליחת הקובץ חברך יצרף את חתימתו, שתעבור בדיקת אמינות. במידה והחתימה תמצא מזויפת תקבל על כך התרעה על גבי המסך, ותוחזר לדף הראשי. במידה והחתימה תמצא אמינה התהליך ימשיך כרגיל. אתה תועבר לדף בו יוצג הקובץ החתום ותוכל לשמור אותו על מחשבך. לאחר סיום העסקה תוכל לחזור לדף הראשי.

במידה והינך מעוניין לקבל קבצים, כל שעליך לעשות הוא ללחוץ על הכפתור "Sign File" ולהמתין לקבלת בקשת שליחה ממשתמש נוסף, אותה תוכל לאשר או לסרב לה. במידה לא יקרה דבר ותישאר בדף הראשי. במידה ותאשר את הבקשה, תועבר לדף בו תוכל לצרף חתימה לקובץ שיוצג מולך. החתימה תעבור בדיקה, ובמידה ותמצא מזויפת תקבל על כך התרעה ותוחזר לדף הראשי מבלי להשלים את העסקה. אם החתימה תמצא אמיתית תוכל להמשיך בתהליך וללחוץ "Send". אז תוחזר לדף הראשי גם כן.

במידה ושכחת את סיסמתך תוכל ללחוץ על "I forgot my password" שבמסך ההתחברות. בהכנסת מייל איתו נרשמת למערכת ישלח לך קוד אותו תצטרך להזין. לאחר הזנת הקוד תוכל להזין סיסמה חדשה ולאשר אותה. אתה תוחזר לדף הראשי ותוכל להמשיך ולבצע התחברות מחדש.

מאחלים לך גלישה נעימה ובטוחה :)

מבט אישי על העבודה ותהליך הפיתוח

תחילה, העבודה הפרויקט הייתה מאתגרת עבורי. הרגשתי מעט מאוימת מסדר גודל הפרויקט כפי שנשמע בהתחלה. ככל שהתקדמתי בביצוע הפרויקט, בהדרגה נפתחו בפניי תחומים חדשים ומעניינים בעולם התוכנה, והסקרנות שלי גברה. ככל שחקרתי יותר, הלכה ודעכה הרתיעה הזו. הרגשתי שאני מרחיבה את אופקיי ומחזקת את הידע שלי בשפות התכנות בהן השתמשתי לבניית המערכת.

נהניתי מאוד מתהליך העבודה והפיתוח וכתיבת הפרויקט. את התהליך התחלתי בכתיבת סקריפט קצר בפייתון אשר מבצע השוואה בין שתי תמונות, אחת עם חתימה מקורית והשנייה חתימה לבחינה, ומזהה האם חתמת הבחינה מזויפת או אמיתית. כדי להגיע לאבן דרך זו בדקתי את הדרכים הקיימות לביצוע הדבר ומצאתי ספריית פייתון שעושה בדיוק את זה, אך לאחר ניסיונות רבים היא לא פעלה בדרך שסיפקה אותי. בהמשך נוכחתי לגלות שהדבר מסובך יותר משחשבתי, ושיש צורך בבניית רשת נוירונים. לאחר מחקר מעמיק מצאתי דרך מתמטית לבצע את הדבר, ומימשתי אותה. בעקבות המחקר ומימושו השכלתי רבות בתחום הבינה המלאכותית ובייחוד בתחום החתימות, ושמחתי שהחזון שהיה לי כאשר הגיתי את רעיון הפרויקט התממש.

בהמשך בניתי את צד השרת וצד הלקוח, יחד עם ממשק המשתמש. את ממשק המשתמש והלקוח כתבתי בשפה שלא השתמשנו בה למימוש הדברים הללו במסגרת הלימודים בבית הספר, והיה מעניין לגלות כיצד מבצעים זאת לבדי. לבסוף הייתי מרוצה מנראות ממשק הלקוח, שהתאים למה שרציתי מההתחלה.

כדי להתגבר על קשיים ובעיות שצוינו בפרק תיאור הבעיה האלגוריתמית, וכן בבעיות שצצו בזמן העבודה על הפרויקט, חקרתי ועשיתי נסיונות, הגיתי והכרתי דרכי פתרון רבות לעומק. אשמח מאוד להשתמש בידע זה בעתיד.

מרבית החלקים מהתכנון הראשוני של הפרויקט הוצאו לפועל, ועל כך אני מרוצה ומלאת מוטיבציה להמשיך, לשפר ולשדרג את הפרויקט בדרכים נוספות. התכנון הראשוני היה תכנון פשוט וצנוע שדגל בעקרונות התכנות מונחה העצמים. ככל שהתקדמתי בעבודה נוספו לי רעיונות לתכונות נוספות במערכת, והסקרנות והציפייה שלי גדלו. נהניתי לחשוב כיצד לפתח את המערכת יותר יותר בצורה טובה. בתור אדם פרפקציוניסטי, העובדה שמרבית ציפיותיי מהתוצר המוגמר נענו משמחת ומספקת.

לסיכום, העבודה על הפרויקט העניקה שלי תחושת התפתחות, למידה, העשרה, סקרנות והנאה. למדתי רבות במהלך העבודה העצמאית והרחבתי את הידע שלי בעולם התכנות. הפרויקט משרת את מטרתו - למידה - בצורה טובה.

ביבליוגרפיה

"SigNet (Detecting Signature Similarity Using Machine Learning/Deep Learning): Is This the End of Human Forensic Analysis?"
Kapoor Aadit, August 17 2020, published in TheStartup website.
https://medium.com/swlh/signet-detecting-signature-similarity-using-machine-learning-deep-learning-is-this-the-end-of-1a6bdc76b04b

"How to calculate the Structural Similarity Index (SSIM) between two images with Python", Delgado Carlos, July 17 2019, published in Our Code World website.
https://ourcodeworld.com/articles/read/991/how-to-calculate-the-structural-similarity-index-ssim-between-two-images-with-python

"Socket Programming In C#", Dottys, November 8 2021, published on C-Sharpcorner website.
https://www.c-sharpcorner.com/article/socket-programming-in-C-Sharp/

"BackgroundWorker In C#", Chand Mahesh, July 29 2019, published on C-Sharpcorner website.
https://www.c-sharpcorner.com/uploadfile/mahesh/backgroundworker-in-C-Sharp/

"Working with PDFs in Python: Adding Images and Watermarks", Hofmann Frank, June 5th 2019, published on Stack Abuse website.
https://stackabuse.com/working-with-pdfs-in-python-adding-images-and-watermarks/

קוד התוכנית

server:

```
1. import socket, threading, re, os
2. import pandas as pd
3. from matplotlib import pyplot as plt
4.
5. class Server:
6.     def __init__(self,ip,port):
7.         self.socket = socket.socket()
8.         self.socket.bind((ip,port))
9.         self.socket.listen(5)
10.        # dictionary of connected users and their client sockets
11.        self.clients = {}
12.        # list of busy/unavailable currently connected users to the system
13.        self.busy = []
14.        self.df =
   pd.read_csv(r"C:\Users\idd\Desktop\Michals\cyber\Signatural\Demo\users_data.
   csv")
15.        print("Server initiallized.")
16.
17.    def get_connection(self):
18.        # Accepts connection of a client
19.        client, addr = self.socket.accept()
20.        print("got connection from: " + str(addr))
21.        client_id = len(self.clients)
22.
23.        # Creates a thread for each client and adds
24.        # them to the clients dictionary.
25.        t = threading.Thread(target=Server.handle_client ,
26.                             args=(self,client,client_id,)).start()
27.        self.clients.update({client_id:client})
28.
29.    def handle_client(self,client,client_id):
30.        # Handles the clients actions in the server
31.        try:
32.            action = client.recv(1024).decode()
33.            action = action[1:-1]
34.            while(action!="Exit"):
35.                if(action=="Login"):
36.                    username = Server.login(self, client,client_id)
37.                    while(username == -1):
38.                        username = Server.login(self, client,client_id)
39.                if(action=="Signup"):
40.                    code = Server.signup(self,client)
41.                    while(code=="Failed"):
42.                        code = Server.signup(self,client)
43.                if(action=="Send file"):
44.                    code = Server.upload(self,client,username)
45.                    while(code==-1):
46.                        code = Server.upload(self,client,username)
47.
48.                action = client.recv(1024).decode()
49.                action = action[1:-1]
50.        except:
51.            # Connection was cut off, closes the socket
```

```
52.             # and deletes it from the clients dictionary
53.             try:
54.                 client.close()
55.                 del self.clients[username]
56.             except:
57.                 # client hasn't logged in yet and still has
58.                 # an id instead of a username as the dictionary key.
59.                 del self.clients[client_id]
60.
61.
62.     def login(self, client,client_id):
63.         # Get username and password from client, or forgot password
64.         # statement from client
65.         details = client.recv(1024).decode()
66.         if(details=="Forgot Password"):
67.             while(code==-1):
68.                 code = Server.ForgotPassword(self,client)
69.             return "Out"
70.         details = details[1:-1].split(',')
71.         username = details[0]
72.         password = details[1]
73.
74.         # if username doesn't exists
75.         usernames_list = self.df['username'].tolist()
76.         if(username not in usernames_list):
77.             client.send("Invalid username.".encode())
78.             return -1
79.         else:
80.             # Get password of user from the dataframe
81.             user_row_index = self.df.index[self.df["username"]==username] #
    Find row of user
82.             tmp = self.df.iloc[user_row_index] # Create data frame with this
    row only
83.             user_password = tmp["password"][tmp.index[0]] # Get the value of
    the cell in the password columns
84.             user_email = tmp["email"][tmp.index[0]]
85.         # If given password is not the one in the df
86.         if(password != str(user_password)):
87.             client.send("Invalid password.".encode())
88.             return -1
89.
90.         else:
91.             client.send("Logged in".encode())
92.             # Sends the user's email in order to perform email verification
93.             client.send(user_email.encode())
94.             response = "Valid"#client.recv(1024).decode()
95.             if(response=="Valid"):
96.                 # changes client_id key to the client's username in clients
    dictionary
97.                 self.clients[username] = self.clients[client_id]
98.                 del self.clients[client_id]
99.                 #sends number of signingn attempts and number of forgeries
100.                                                 attepmts =
    tmp["attempts"][tmp.index[0]]
101.                                                 forgeries =
    tmp["forgeries"][tmp.index[0]]
```

```
102.                                                    string =
    str([attempts,forgeries])
103.
    client.send(string.encode())
104.
105.
    plt.plot(range(attempts),range(forgeries), color = 'Orange')
106.                                                    plt.xlabel("Num of
    attempts")
107.                                                    plt.ylabel("Num of
    forgeries")
108.                                                    plt.title("Number of
    forged signatures out of total attempts")
109.                                                    path =
    "C:\\Users\\idd\\Desktop\\Michals\\cyber\\Signatural\\Demo\\" + username
110.                                                    plt.savefig(path +
    "\\username"+"_graph.png")
111.
    Server.sendFile(client,"username"+"_graph.png",path)
112.
113.                                                    print(str(username) + "
    has logged in.")
114.                                                    return username
115.
116.                                        def ForgotPassword(self,client):
117.                                            email =
    client.recv(1024).decode()
118.                                            if(email not in
    list(self.df['email'])):
119.                                                client.send("Email doesn't
    exists.".encode())
120.                                                return -1
121.                                            else:
122.
    client.send("Exists".encode())
123.                                                response =
    client.recv(1024).deocde()
124.                                                if(response =="valid"):
125.                                                    newPass =
    client.recv(1024).deocde() #####
126.                                                    user_row_index =
    self.df.index[self.df["email"]==email]
127.
    self.df.at[user_row_index,'password'] = newPass
128.
    self.df.to_csv(r"C:\Users\idd\Desktop\Michals\cyber\Signatural\Demo\users_da
    ta.csv", index=False)
129.                                                    return 0
130.
131.
132.                                        def signup(self, client):
133.                                            reg = "[a-z0-9]+@[a-z]+\.[a-
    z]{2,3}" # Email format
134.                                            # Gets user details from client
135.                                            details =
    client.recv(1024).decode()
```

```
136.                                            details = details[1:-1].split(',')
137.                                            username = details[0]
138.                                            password = details[1]
139.                                            email = details[2]
140.
141.                                            # If the username already exists
142.                                            if(username in list(self.df['username'])):
143.                                                client.send("Username already exists.".encode())
144.                                                return "Failed"
145.                                            elif(email in list(self.df['email'])):
146.                                                client.send("Email is already taken.".encode())
147.                                                return "Failed"
148.
149.                                            # If the email is not in email format
150.                                            elif(re.search(reg, email) == None):
151.                                                client.send("Got not email format.".encode())
152.                                                return "Failed"
153.                                            else:
154.                                                client.send("Valid".encode())
155.                                                response = client.recv(1024).deocde()
156.                                                if(response=="Valid"):
157.                                                    # Creates a folder for the new user with their signature and files.
158.                                                    path = "C:\\Users\\idd\\Desktop\\Michals\\cyber\\Signatural\\Demo\\users\\" + username
159.                                                    os.mkdir(path)
160.
161.                                                    # Gets and Saves the client's signature, names it after username
162.                                                    path = path + "\\"
163.                                                    signatureName = Server.getFile(client,path)
164.
165.                                                    # Enters details to database
166.                                                    self.df.loc[len(self.df)] = [username,password,email,0,0]
167.                                                    self.df.index += 1
168.                                                    self.df.to_csv(r"C:\Users\idd\Desktop\Michals\cyber\Signatural\Demo\users_data.csv", index=False)
169.                                                    return "Success"
170.
171.
172.                                        def upload(self, client, username):
```

```
173.                                            # Adds the user to the busy users
    list.
174.                                            if(username not in self.busy):
175.                                                self.busy.append(username)
176.
177.                                            requested_signer =
    client.recv(1024).decode()
178.                                            # If requested user is connected
    to the system
179.                                            if(requested_signer in
    self.clients):
180.                                                if(requested_signer in
    self.busy):
181.                                                    client.send("User
    Unavailable.".encode())
182.                                                    return -1
183.                                                else:
184.                                                    message = username + "
    wants you to sign a PDF file. Do you accept?"
185.
    self.clients[requested_signer].send(message.encode())
186.                                                    response =
    self.clients[requested_signer].recv(1024).decode()
187.
188.                                                    # If signer accepts, the
    server gets the PDF file from the sender
189.                                                    if(response == "Yes"):
190.                                                        # Adds the signer to
    the busy users list.
191.                                                        if(requested_signer
    not in self.busy):
192.
    self.busy.append(requested_signer)
193.
194.                                                        client.send("Request
    accepted".encode())
195.                                                        path =
    r"Demo\\users\\" + username + "\\"
196.
197.                                                        # The server gets the
    wanted file from the sender
198.                                                        filename =
    Server.getFile(client,path)
199.                                                        print("Got file: " +
    filename + "from: " + username + " !")
200.
201.                                                        # The server sends
    the file to the signer.
202.
    Server.sendFile(self.clients[requested_signer],filename,path)
203.                                                        print("Sent file: " +
    filename + " to: " + requested_signer +"!")
204.
205.                                                        # Gets the signature
    attempt of the signer
```

```
                                                        signatureName = Server.getFile(self.clients[requested_signer],path)

                                                        # Checks if the signature is real of forged
                                                        authenticity = signet.main()


                                                        if(authenticity=="real"):
                                                            # Gets signed file from signer
                                                            self.clients[requested_signer].send("real".encode())
                                                            filename = Server.getFile(self.clients[requested_signer],path)

                                                            # Sends it to the sender
                                                            client.send("real".encode())
                                                            Server.sendFile(client,filename,path)
                                                        else:
                                                            # Notifies both users that the signature is forged
                                                            # and that the deal is off.
                                                            client.send("forged".encode())
                                                            self.clients[requested_signer].send("forged".encode())

                                                        return "Success!"

                                                    else:
                                                        client.send("Request denied".encode())
                                                        return -1

                                            else:
                                                client.send("User not connected.".encode())
                                                return -1

                                            self.busy.pop(username)
                                            self.busy.pop(requested_signer)


                                        def getFile(sender,path):
                                            filename = sender.recv(1024).decode()
                                            path = path + filename
                                            with open(path,'wb') as f:
                                                while True:
```

```
                                                data = client.recv(1024)
                                                if not data:
                                                    break
                                                f.write(data)
                                            return filename

                                        def sendFile(getter,filename,path):
                                            getter.send(filename.encode())
                                            path = path + filename
                                            with open (path, 'rb') as f:
                                                while True:
                                                    data=f.read(1024)
                                                    if not data:
                                                        break
getter.send(data.encode())

                                    def main():
                                        server = Server("",55876)
                                        while True:
                                            server.get_connection()
                                        server.close()

                                    main()
```

Client:

```
using System;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace GUI
{
    public class Client
    {
        private byte[] bytes;
        private Socket s;
        public Client(int port)
        {
            bytes = new byte[1024];

            // Get IPs of the server host.
            IPAddress[] IPs = Dns.GetHostAddresses("127.0.0.1");

            // Initialize a TCP socket of IPv4 type.
            s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            Console.WriteLine("Initialized client.");

            // Connect to the server in the first IP and given port.
            s.Connect(IPs[0], port);
            Console.WriteLine("Connection established.");
```

```
29.         }
30.
31.         /// <summary>
32.         /// Converts string to bytes and sends it to the server.
33.         /// </summary>
34.         /// <param name="msg"></param>
35.         public void Send(string msg)
36.         {
37.             s.Send(Encoding.ASCII.GetBytes(msg));
38.             Console.WriteLine("Sent message.");
39.         }
40.
41.         /// <summary>
42.         /// Saves received bytes from server in bytesArray parameter of the
    class and converts it to string.
43.         /// </summary>
44.         /// <returns></returns>
45.         public string Receive()
46.         {
47.             // Saves the bytes count of the message in length variable.
48.             int length = s.Receive(bytes);
49.             Console.WriteLine("Got message.");
50.
51.             // Decodes the byte array back into a string and returns it.
52.             // Encoding.ASCII.GetString({bytes array}, {index of first byte
    to decode}, {count of bytes to decode})
53.             string data = Encoding.ASCII.GetString(bytes, 0, length);
54.             return data;
55.         }
56.
57.         /// <summary>
58.         /// Gets file from server and saves it in filesReceived folder.
59.         /// </summary>
60.         /// <param name="filename">></param>
61.         public void GetFile(string filename)
62.         {
63.             string path =
    "C:\\Users\\idd\\Desktop\\Michals\\cyber\\Signatural\\filesReceived\\" +
    filename;
64.
65.             // Opens a new file in write mode.
66.             using (FileStream file = File.Create(path))
67.             {
68.                 // Receives packets with 1024 bytes and adds them to the
    file each time,
69.                 // until the whole file was received.
70.                 int length = s.Receive(bytes);
71.                 while (length > 0)
72.                 {
73.                     file.Write(bytes, 0, length); // file.write({byte
    array}, {index of start}, {bytes count})
74.                     length = s.Receive(bytes);
75.                 }
76.                 Console.WriteLine("Received file.");
77.             }
78.         }
```

```
79.
80.         /// <summary>
81.         /// Sends file to server.
82.         /// </summary>
83.         /// <param name="filepath"></param>
84.         public void SendFile(string filepath)
85.         {
86.             // sends the filename
87.             string[] tmp = filepath.Split('\\');
88.             Console.WriteLine(tmp[tmp.Length - 1]); //
89.             s.Send(Encoding.ASCII.GetBytes(tmp[tmp.Length-1]));
90.
91.             // Opens the specific file in read mode.
92.             using (FileStream file = new FileStream(filepath, FileMode.Open,
    FileAccess.Read))
93.             {
94.                 // Reads 1024 bytes, stores them in b and sends it each
    time,
95.                 // untill there are no more bytes to read.
96.                 byte[] b = new byte[1024];
97.                 while (file.Read(b, 0, b.Length) > 0)
98.                 {
99.                     s.Send(b);
100.                                                 }
101.                                                 Console.WriteLine("Sent
    file.");
102.                                             }
103.                                         }
104.
105.                                         public void Close()
106.                                         {
107.                                             s.Close();
108.                                         }
109.                                     }
110.                                 }
```

enterSignatureWindow.py:

```
1.  import tkinter as tk
2.  from PIL import ImageGrab
3.  import win32gui, sys
4.  from functools import partial
5.
6.  def create_window():
7.      '''Creates window.'''
8.      root = tk.Tk()
9.      root.title("Enter Signature Here:")
10.     root.geometry('400x250+400+200') # set dimensions & place of window
11.     return root
12.
13.
14. coordinates=["x","y"] # coordinates (x,y) of mouse
15. def mmove(event):
16.     '''Saves the mouse's cordinates and prints them.'''
17.     coordinates[0]= event.x
18.     coordinates[1]= event.y
```

```
19.
20.
21. def draw(event):
22.     '''Draws signature on canvas according to mouse movements.'''
23.     newCoordsx = event.x
24.     newCoordsy = event.y
25.
    canvas.create_line(coordinates[0],coordinates[1],newCoordsx,newCoordsy,fill=
    "black")
26.
27. def clearScreen():
28.     '''Clears canvas.'''
29.     canvas.delete("all")
30.
31. def saveSignature(username):
32.     '''Saves the signature as a png file.'''
33.     HWND = canvas.winfo_id()  # get the handle of the canvas
34.     coords = win32gui.GetWindowRect(HWND)  # get the coordinate of the
    canvas
35.     im = ImageGrab.grab(coords).save(username+'.png')
36.     print(username+'.png')
37.     #print("Image coordinates: " + str(coords)) # tuple of
    (left,upper,right,lower)
38.     root.destroy()
39.
40.
41. def create_canvas(root):
42.     '''Createa a canvas inside the window.'''
43.     canvas = tk.Canvas(root,width=400, height=200, bg = "white",
    cursor="cross")
44.     canvas.pack()
45.     tk.Label(root, text="Draw your signature, enter 'Done!' when
    finished.").pack()
46.     return canvas
47.
48. def create_buttons(username):
49.     '''Creates clear & done buttons.'''
50.     button_clear = tk.Button(text = "Clear", command = clearScreen)
51.     button_clear.pack(side="left", fill="both",expand=True)
52.
53.     button_done = tk.Button(text = "Done!", command =
    partial(saveSignature,username))
54.     button_done.pack(side="right", fill="both",expand=True)
55.
56.
57. def main(username):
58.     global root
59.     root = create_window()
60.     root.bind('<Motion>', mmove)# binds event=motion to the window and mmove
61.     global canvas
62.     canvas = create_canvas(root)
63.     create_buttons(username)
64.     # coordinates of mouse-hovering movements on canvas are sent to mmove
    function
65.     canvas.bind("<Motion>", mmove)
```

```
    # coordinates of next mouse-clicked movements are sent to draw function
    canvas.bind("<B1-Motion>", draw)
    root.mainloop() # open final window on screen

main(sys.argv[1])
```

Signet.py:

```
from skimage.metrics import structural_similarity
import numpy as np
from PIL import Image

def mse(A, B):
    """
        Computes Mean Squared Error between two images.
        Arguments:
            A: numpy array of image 1.
            B: numpy array of image 2.
        Returns:
            err: float.
    """

    # sigma(1, n-1)(a-b)^2)
    err = np.sum((A - B) ** 2)

    # mean of the sum (r,c) => total elements: r*c
    err /= float(A.shape[0] * B.shape[1])
    return err

def ssim(A, B):
    """
        Computes Structure Similarity between two images.
        Arguments:
            A: numpy array of image 1.
            B: numpy array of image 2.
        Returns:
            score: float.
    """
    return structural_similarity(A.flatten(), B.flatten())

def main(signature1,signature2):
    # Open images as numpy arrays.
    real_img = Image.open(signature1)
    realnp = np.asarray(real_img)
    test_img = Image.open(signature2)
    testnp = np.asarray(test_img)

    # Make sure the images have the same size.
    testnp.resize(realnp.shape)
    mse = mse(realnp,testnp)
    ssim = ssim(realnp,testnp)
    if(mse < 0.3 and ssim > 0.7)
        return "real"
    else
        return "forged"
```

addSingtoPDF:

```
import fitz, sys

def start(file,signature):
    doc = fitz.open(file)
    rect = fitz.Rect(200, 20, 400, 204)   # put thumbnail in upper left corner
    pix = fitz.Pixmap(signature)# an image file
    for page in doc:
        page.insert_image(rect, pixmap = pix)
        break
    doc.saveIncr()
    doc.close()

start(sys.argv[1],sys.argv[2])
```

email_verification.py:

```
import os, sys, random
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

def send(email):
    mail_content = '''Hello,
Thank you for joining us! We are more than happy to welcome you to our family :)
Just one more thing! Please enter the following code to the text box in our app,
so that we can verify that it is you and activate your account.
If your do not know what we are talking about, please ignore this email. Thank you!"
Code: '''
    code = str(random.randint(0, 3000))
    mail_content += code
    #The mail addresses and password
    sender_address = 'signaturalteam@gmail.com'
    sender_pass = '326195542'
    receiver_address = email
    #Setup the MIME
    message = MIMEMultipart()
    message['From'] = sender_address
    message['To'] = receiver_address
    message['Subject'] = 'Verify your email address and account!'   #The subject line
    #The body and the attachments for the mail
    message.attach(MIMEText(mail_content, 'plain'))
    #Create SMTP session for sending the mail
    session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port
    session.starttls() #enable security
    session.login(sender_address, sender_pass) #login with mail_id and password
    text = message.as_string()
    session.sendmail(sender_address, receiver_address, text)
    session.quit()
    print(code)

```

```
send(sys.argv[1])
```

StaticClass.cs:

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

namespace GUI
{
    static class StaticClass
    {
        public static string currentPage;

        public static string EmailVerification(string email)
        {
            string script = @"C:\Users\idd\Desktop\Michals\cyber\Signatural\email_verification.py";
            var psi = new ProcessStartInfo();
            psi.FileName = @"C:\Users\idd\AppData\Local\Programs\Python\Python38-32\python.exe"; //my/full/path/to/python.exe
            psi.Arguments = string.Format(script + " " + email); //cmd, args
            psi.UseShellExecute = false;
            psi.CreateNoWindow = true;
            psi.RedirectStandardOutput = true;
            psi.RedirectStandardError = true;
            var errors = "";
            var result = "";
            using (var process = Process.Start(psi))
            {
                errors = process.StandardError.ReadToEnd();
                result = process.StandardOutput.ReadToEnd();
                Console.WriteLine(result);
            }
            return result;
        }
    }
}
```

Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
```

```
13.     public partial class Form1 : System.Windows.Forms.Form
14.     {
15.         static Form1 _obj;
16.         public string currentPage;
17.         private Client client;
18.         public static Form1 Instance
19.         {
20.             get
21.             {
22.                 if (_obj == null)
23.                     _obj = new Form1();
24.                 return _obj;
25.             }
26.         }
27.
28.         public Panel PnlContainer
29.         {
30.             get { return panelContainer; }
31.             set { panelContainer = value; }
32.         }
33.
34.         public Button BackButton
35.         {
36.             get { return btnBack; }
37.             set { btnBack = value; }
38.         }
39.
40.         public Form1()
41.         {
42.             InitializeComponent();
43.             backgroundWorker1.RunWorkerAsync(2000);
44.             currentPage = "HomePage";
45.
46.
47.         }
48.
49.         private void Form1_Load(object sender, EventArgs e)
50.         {
51.             btnBack.Visible = false;
52.             _obj = this;
53.
54.             HomePage homePage = new HomePage(client);
55.             homePage.Dock = DockStyle.Fill;
56.             panelContainer.Controls.Add(homePage);
57.             StaticClass.currentPage = "HomePage";
58.         }
59.
60.         private void btnExit_Click(object sender, EventArgs e)
61.         {
62.             client.Close();
63.             this.Close();
64.         }
65.
66.         private void btnTwitter_Click(object sender, EventArgs e)
67.         {
```

```
            System.Diagnostics.Process.Start("https://twitter.com");
        }

        private void btnInsta_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("https://instagram.com");
        }

        private void btnLinkedin_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("https://linkedin.com");
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            BackgroundWorker helperBW = sender as BackgroundWorker;
            int arg = (int)e.Argument;
            e.Result = BackgroundProcessLogicMethod(helperBW, arg);
            if (helperBW.CancellationPending)
            {
                e.Cancel = true;
            }
        }
        private int BackgroundProcessLogicMethod(BackgroundWorker bw, int a)
        {
            int result = 0;
            client = new Client(55876);
            return result;
        }
    }
}
```

HomePage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    public partial class HomePage : UserControl
    {
        private Client client;
        public HomePage(Client theClient)
        {
            InitializeComponent();
            Form1.Instance.BackButton.Visible = false;
            client = theClient;
        }

```

```
23.         private void btnLogin_Click(object sender, EventArgs e)
24.         {
25.             client.Send("[Login]");
26.
    if(!Form1.Instance.PnlContainer.Controls.ContainsKey("LoginPage"))
27.             {
28.                 LoginPage loginPage = new LoginPage(client);
29.                 Form1.Instance.PnlContainer.Controls.Add(loginPage);
30.             }
31.
    Form1.Instance.PnlContainer.Controls["LoginPage"].BringToFront();
32.             Form1.Instance.BackButton.Visible = true;
33.             StaticClass.currentPage = "LoginPage";
34.         }
35.
36.         private void btnSignup_Click(object sender, EventArgs e)
37.         {
38.             client.Send("[Signup]");
39.             if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("SignupPage"))
40.             {
41.                 SignupPage signupPage = new SignupPage(client);
42.                 Form1.Instance.PnlContainer.Controls.Add(signupPage);
43.             }
44.
    Form1.Instance.PnlContainer.Controls["SignupPage"].BringToFront();
45.             Form1.Instance.BackButton.Visible = true;
46.             StaticClass.currentPage = "SignupPage";
47.         }
48.     }
49. }
```

LoginPage.cs:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Linq;
7.  using System.Text;
8.  using System.Threading.Tasks;
9.  using System.Windows.Forms;
10. using System.Security.Cryptography;
11.
12.
13. namespace GUI
14. {
15.     public partial class LoginPage : UserControl
16.     {
17.         private Client client;
18.         private int verification_code;
19.         public LoginPage(Client theClient)
20.         {
21.             InitializeComponent();
22.             Form1.Instance.BackButton.Visible = true;
23.             client = theClient;
24.             boxVerifyCode.Visible = false;
25.             lblVerifyCode.Visible = false;
```

```
26.         }
27.
28.         private void linkForgotPassword_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
29.         {
30.             if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("ForgotPasswordPage"))
31.             {
32.                 ForgotPasswordPage forgotPasswordPage = new
    ForgotPasswordPage(client);
33.
    Form1.Instance.PnlContainer.Controls.Add(forgotPasswordPage);
34.             }
35.
    Form1.Instance.PnlContainer.Controls["ForgotPasswordPage"].BringToFront();
36.             StaticClass.currentPage = "ForgotPasswordPage";
37.             client.Send("Forgot Password");
38.
39.         }
40.         private void btnDoneLogin_Click(object sender, EventArgs e)
41.         {
42.             int input_code = Int32.Parse(boxVerifyCode.Text);
43.             if (input_code == verification_code)
44.             {
45.                 client.Send("Valid");
46.                 string response = client.Receive();
47.                 int attempts = Int32.Parse((response.Substring(1,
    response.Length - 2)).Split(',')[0]);
48.                 int forgeries = Int32.Parse((response.Substring(1,
    response.Length - 2)).Split(',')[1]);
49.                 string filename = client.Receive();
50.                 client.GetFile(filename);
51.                 if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("UserProfilePage"))
52.                 {
53.                     UserProfilePage userProfilePage = new
    UserProfilePage(client, boxUsernameLogin.Text,filename);
54.
    Form1.Instance.PnlContainer.Controls.Add(userProfilePage);
55.                 }
56.
    Form1.Instance.PnlContainer.Controls["UserProfilePage"].BringToFront();
57.                 StaticClass.currentPage = "UserProfilePage";
58.                 Form1.Instance.BackButton.Visible = false;
59.             }
60.             else
61.             {
62.                 MessageBoxIcon error = MessageBoxIcon.Error;
63.                 MessageBox.Show("Incorrect code.", "Oops...",
    MessageBoxButtons.OK, error);
64.             }
65.         }
66.
67.         private void btnSendVerifyMail_Click(object sender, EventArgs e)
68.         {
69.             string username = boxUsernameLogin.Text;
70.             string password = boxPasswordLogin.Text;
```

```
71.
72.             // Convert password to MD5
73.             var md5Hash = MD5.Create(); //Creates an instance of the MD5
   hash algorithm.
74.             var sourceBytes = Encoding.ASCII.GetBytes(password); // Byte
   array representation of the password
75.             var hashBytes = md5Hash.ComputeHash(sourceBytes); // Generate
   hash value (in Byte Array) of the input
76.             var hash = BitConverter.ToString(hashBytes).Replace("-",
   string.Empty); // Convert hash byte array to string
77.
78.             client.Send($"[{username},{hash}]");
79.             string response = client.Receive();
80.             if (response != "Logged in")
81.             {
82.                 MessageBoxIcon error = MessageBoxIcon.Error;
83.                 MessageBox.Show(response, "Oops...", MessageBoxButtons.OK,
   error);
84.             }
85.             else
86.             {
87.                 string email = client.Receive();
88.                 verification_code =
   Int32.Parse(StaticClass.EmailVerification(email));
89.                 lblVerifyCode.Visible = true;
90.                 boxVerifyCode.Visible = true;
91.                 btnSendVerifyMail.Visible = false;
92.             }
93.         }
94.     }
95. }
```

Signup.cs:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Diagnostics;
6.  using System.Drawing;
7.  using System.IO;
8.  using System.Linq;
9.  using System.Security.Cryptography;
10. using System.Text;
11. using System.Threading.Tasks;
12. using System.Windows.Forms;
13.
14. namespace GUI
15. {
16.     public partial class SignupPage : UserControl
17.     {
18.         private Client client;
19.         public SignupPage(Client theClient)
20.         {
21.             InitializeComponent();
22.             Form1.Instance.BackButton.Visible = true;
23.             client = theClient;
24.         }
25.
26.         private void btnDoneSignup_Click(object sender, EventArgs e)
27.         {
28.             string username = boxUsernameSignup.Text;
29.             string password = boxPasswordSignup.Text;
30.             string email = boxEmailSignup.Text;
31.             if(password.Length<6)
32.             {
33.                 MessageBoxIcon error = MessageBoxIcon.Error;
34.                 MessageBox.Show("Password must be at least 6 characters.",
    "Oops...", MessageBoxButtons.OK, error);
35.                 return;
36.             }
37.             var md5Hash = MD5.Create(); //Creates an instance of the MD5 hash
    algorithm.
38.             var sourceBytes = Encoding.ASCII.GetBytes(password); // Byte array
    representation of the password
39.             var hashBytes = md5Hash.ComputeHash(sourceBytes); // Generate hash
    value (in Byte Array) of the input
40.             var hash = BitConverter.ToString(hashBytes).Replace("-",
    string.Empty); // Convert hash byte array to string
41.
42.             //checks details...
43.             client.Send($"[{username},{hash},{email}]");
44.             string response = client.Receive();
45.             if (response == "Valid")
46.             {
47.                 //email verification here!!!!
48.                 string verificationCode = StaticClass.EmailVerification(email);
49.
50.                 if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("EmailVerificationPage"))
```

```
51.                 {
52.                     EmailVerificationPage emailVerificationPage = new
    EmailVerificationPage(client,verificationCode, boxUsernameSignup.Text);
53.
    Form1.Instance.PnlContainer.Controls.Add(emailVerificationPage);
54.                 }
55.
    Form1.Instance.PnlContainer.Controls["emailVerificationPage"].BringToFront();
56.             }
57.             else
58.             {
59.                 MessageBoxIcon error = MessageBoxIcon.Error;
60.                 MessageBox.Show(response, "Oops...", MessageBoxButtons.OK,
    error);
61.             }
62.         }
63.
64.     }
65. }
```

EmailVerificationPage.cs:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Diagnostics;
6.  using System.Drawing;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10. using System.Windows.Forms;
11.
12. namespace GUI
13. {
14.     public partial class EmailVerificationPage : UserControl
15.     {
16.         private Client client;
17.         private int verification_code;
18.         private string rootpath =
    @"C:\Users\idd\Desktop\Michals\cyber\Signatural\";
19.         private string username;
20.
21.         public EmailVerificationPage(Client theClient, string
    verification_code,string username)
22.         {
23.             InitializeComponent();
24.             Form1.Instance.BackButton.Visible = false;
25.             client = theClient;
26.             this.verification_code = Int32.Parse(verification_code);
27.             this.username = username;
28.
29.         }
30.
31.         private void btnOkEmailVerification_Click(object sender, EventArgs
    e)
32.         {
33.             //check verification code
34.             int input_code = Int32.Parse(boxActivateAccCode.Text);
35.             if (verification_code != input_code)
36.             {
37.                 MessageBoxIcon icon = MessageBoxIcon.Error;
38.                 MessageBox.Show("Wrong code.", "Oops...!",
    MessageBoxButtons.OK, icon);
39.             }
40.             else
41.             {
42.                 client.Send("Valid");
43.                 // Get the new user's signature
44.                 GetSignature();
45.                 client.SendFile(this.rootpath + username);
46.                 MessageBoxIcon icon = MessageBoxIcon.Information;
47.                 MessageBox.Show("You've successfully signed up!\r\nLog in
    with your new Account.", "Success!", MessageBoxButtons.OK, icon);
48.
    Form1.Instance.PnlContainer.Controls["HomePage"].BringToFront();
49.                 StaticClass.currentPage = "HomePage";
50.             }
51.         }
```

```
52.
53.         private void GetSignature()
54.         {
55.             string script = this.rootpath + "enterSignatureWindow.py";
56.             var psi = new ProcessStartInfo();
57.             psi.FileName =
    @"C:\Users\idd\AppData\Local\Programs\Python\Python38-32\python.exe";
58.             psi.Arguments = string.Format(script + " " + this.username);
59.             psi.UseShellExecute = false;
60.             psi.CreateNoWindow = true;
61.             psi.RedirectStandardOutput = true;
62.             psi.RedirectStandardError = true;
63.             var errors = "";
64.             var result = "";
65.             using (var process = Process.Start(psi))
66.             {
67.                 errors = process.StandardError.ReadToEnd();
68.                 result = process.StandardOutput.ReadToEnd();
69.                 process.WaitForExit();
70.             }
71.         }
72.     }
73. }
```

ForgotPasswordPage.cs:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Linq;
7.  using System.Text;
8.  using System.Threading.Tasks;
9.  using System.Windows.Forms;
10.
11. namespace GUI
12. {
13.     public partial class ForgotPasswordPage : UserControl
14.     {
15.         private Client client;
16.         private int verification_code;
17.         private string email;
18.         public ForgotPasswordPage(Client theClient)
19.         {
20.             InitializeComponent();
21.             lblVerifyCode.Visible = false;
22.             boxVerifyCode.Visible = false;
23.             btnOkVerifyCode.Visible = false;
24.             linkSendEmailAgain.Visible = false;
25.             Form1.Instance.BackButton.Visible = true;
26.             client = theClient;
27.         }
28.
29.         private void btnEmailForgotPassword_Click(object sender, EventArgs
    e)
30.         {
31.             //check email
32.             client.Send($"{boxEmailForgotPassword.Text}");
```

```
33.             string response = client.Receive();
34.             if (response != "Exists")
35.             {
36.                 MessageBoxIcon error = MessageBoxIcon.Error;
37.                 MessageBox.Show(response, "Oops...", MessageBoxButtons.OK,
    error);
38.             }
39.             else
40.             {
41.                 this.email = boxEmailForgotPassword.Text;
42.                 verification_code =
    Int32.Parse(StaticClass.EmailVerification(email));
43.                 lblVerifyCode.Visible = true;
44.                 boxVerifyCode.Visible = true;
45.                 btnOkVerifyCode.Visible = true;
46.                 linkSendEmailAgain.Visible = true;
47.             }
48.         }
49.
50.         private void linkLabel1_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
51.         {
52.             //send email again, check if email textbox is not nul!!.
53.             if(boxEmailForgotPassword.Text == "")
54.             {
55.                 MessageBoxIcon icon = MessageBoxIcon.Error;
56.                 MessageBox.Show("No email stated.", "Oops...",
    MessageBoxButtons.OK, icon);
57.             }
58.             else
59.             {
60.                 verification_code =
    Int32.Parse(StaticClass.EmailVerification(this.email));
61.             }
62.
63.         }
64.
65.         private void btnOkVerifyCode_Click(object sender, EventArgs e)
66.         {
67.             //check if code is good
68.             int input_code = Int32.Parse(boxVerifyCode.Text);
69.             if (input_code == this.verification_code)
70.             {
71.                 if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("ResetPasswordPage"))
72.                 {
73.                     ResetPasswordPage resetPasswordPage = new
    ResetPasswordPage(client);
74.
    Form1.Instance.PnlContainer.Controls.Add(resetPasswordPage);
75.                 }
76.
    Form1.Instance.PnlContainer.Controls["ResetPasswordPage"].BringToFront();
77.                 StaticClass.currentPage = "ResetPasswordPage";
78.             }
79.             else
80.             {
```

```
                MessageBoxIcon icon = MessageBoxIcon.Error;
                MessageBox.Show("Wrong code.", "Oops...", MessageBoxButtons.OK, icon);
            }
        }
    }
}
```

ResetPasswordPage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    public partial class ResetPasswordPage : UserControl
    {
        private Client client;
        public ResetPasswordPage(Client theClient)
        {
            InitializeComponent();
            Form1.Instance.BackButton.Visible = false;
            client = theClient;
        }

        private void btnDoneResetPassword_Click(object sender, EventArgs e)
        {
            string newPass = boxNewPassword.Text;
            string newPass2 = boxNewPassword2.Text;
            if (newPass.Length < 6)
            {
                MessageBoxIcon icon1 = MessageBoxIcon.Error;
                MessageBox.Show("Password must be a least 6 charachters.", "Oops...", MessageBoxButtons.OK, icon1);
            }
            else if (newPass == newPass2)
            {
                var md5Hash = MD5.Create(); //Creates an instance of the MD5 hash algorithm.
                var sourceBytes = Encoding.ASCII.GetBytes(newPass); // Byte array representation of the password
                var hashBytes = md5Hash.ComputeHash(sourceBytes); // Generate hash value (in Byte Array) of the input
                var hash = BitConverter.ToString(hashBytes).Replace("-", string.Empty);

```

```
40.                 client.Send(hash); // not sending the new password for some
   reason!!
41.                 MessageBoxIcon icon2 = MessageBoxIcon.Information;
42.                 MessageBox.Show("Your password has been reset.\r\nReturning to
   home page.", "Success!", MessageBoxButtons.OK, icon2);
43.                 Form1.Instance.PnlContainer.Controls["HomePage"].BringToFront();
44.                 StaticClass.currentPage = "HomePage";
45.             }
46.             else
47.             {
48.                 MessageBoxIcon icon3 = MessageBoxIcon.Error;
49.                 MessageBox.Show("The passwords you've entered are not the same.",
   "Oops...", MessageBoxButtons.OK, icon3);
50.             }
51.         }
52.     }
53. }
```

UserProfilePage.cs:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Linq;
7.  using System.Text;
8.  using System.Threading.Tasks;
9.  using System.Windows.Forms;
10.
```

```
11. namespace GUI
12. {
13.     public partial class UserProfilePage : UserControl
14.     {
15.         private Client client;
16.         private string username;
17.         private string rootpath =
    @"C:\Users\idd\Desktop\Michals\cyber\Signatural\";
18.         public UserProfilePage(Client theClient,string username,string
    filename)
19.         {
20.             InitializeComponent();
21.             Form1.Instance.BackButton.Visible = false;
22.             client = theClient;
23.             this.username = username;
24.             lblHelloUser.Text += username;
25.             pictureBox1.Image =
    Image.FromFile(System.Environment.GetFolderPath
26.                 (System.Environment.SpecialFolder.Personal) + "_graph.png");
27.             this.Controls.Add(pictureBox1);
28.         }
29.
30.         private void btnSendFile_Click(object sender, EventArgs e)
31.         {
32.             client.Send("Send file");
33.             if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("SendFilePage"))
34.             {
35.                 SendFilePage sendFilePage = new SendFilePage(client);
36.                 Form1.Instance.PnlContainer.Controls.Add(sendFilePage);
37.             }
38.
    Form1.Instance.PnlContainer.Controls["SendFilePage"].BringToFront();
39.
    Form1.Instance.PnlContainer.Controls.Remove(Controls["userProfilePage"]);
40.             Form1.Instance.PnlContainer.Visible = true;
41.             StaticClass.currentPage = "SendFilePage";
42.         }
43.
44.         private void btnSignFile_Click(object sender, EventArgs e)
45.         {
46.             // Gets request from other user thorugh the server
47.             btnSignFile.BackColor = Color.AliceBlue;
48.             string msg = client.Receive();
49.             MessageBoxIcon icon = MessageBoxIcon.Information;
50.             DialogResult result = MessageBox.Show(msg, "Hey there!",
    MessageBoxButtons.YesNo, icon);
51.             if(result == DialogResult.Yes)
52.             {
53.                 client.Send("Yes");
54.                 // Gets needed file
55.                 string filename = client.Receive();
56.                 client.GetFile(filename);
57.
58.                 if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("SignFilePage"))
59.                 {
```

```
                    SignFilePage signFilePage = new SignFilePage(client,rootpath+filename,this.username);
                    Form1.Instance.PnlContainer.Controls.Add(signFilePage);
                }
Form1.Instance.PnlContainer.Controls["SignFilePage"].BringToFront();
Form1.Instance.PnlContainer.Controls.Remove(Controls["userProfilePage"]);
            }
            else
            {
                btnSignFile.BackColor = Color.DodgerBlue;
            }
        }
    }
}
```

SendFilePage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    public partial class SendFilePage : UserControl
    {
        private Client client;
        public SendFilePage(Client theClient)
        {
            InitializeComponent();
            browseFilesPanel.Visible = false;
            btnSendFile.Visible = false;
            lblSignerUserStatus.Text = "";
            Form1.Instance.BackButton.Visible = true ;
            client = theClient;
        }

        private void btnBrowseFiles_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog
            {
                InitialDirectory = @"C:\", //root directory
                Title = "Browse PDF Files", //title of window
                CheckFileExists = true, //alert if user stated a nonexisting file
                CheckPathExists = true, //alert if user stated a nonexisting path
                DefaultExt = "pdf", //filter - show only pdf files
                Filter = "pdf files (*.pdf)|*.pdf",
                FilterIndex = 2,
            };
```

```
38.
39.             if (openFileDialog1.ShowDialog() == DialogResult.OK)
40.             {
41.                 filenamebox.Text = openFileDialog1.FileName;
42.                 btnSendFile.Visible = true;
43.             }
44.         }
45.
46.         private void btnSendFile_Click(object sender, EventArgs e)
47.         {
48.             //send file to server
49.             client.SendFile(filenamebox.Text);
50.             Console.WriteLine($"Sent {filenamebox.Text}!");
51.
52.             // Gets notified if there was a forgery attempt or not
    (signature is real / forged)
53.             string response = client.Receive();
54.             if (response == "real")
55.             {
56.                 string filename = client.Receive();
57.                 client.GetFile(filename);
58.
59.                 Form1.Instance.BackButton.Visible = false;
60.                 if
    (!Form1.Instance.PnlContainer.Controls.ContainsKey("SaveSignedFilePage"))
61.                 {
62.                     SaveSignedFilePage saveSignedFilePage = new
    SaveSignedFilePage(client, filename);
63.
    Form1.Instance.PnlContainer.Controls.Add(saveSignedFilePage);
64.                 }
65.
    Form1.Instance.PnlContainer.Controls["SaveSignedFilePage"].BringToFront();
66.                 StaticClass.currentPage = "SaveSignedFilePage";
67.
68.
    Form1.Instance.PnlContainer.Controls.Remove(Controls["SendFilePage"]);
69.             }
70.             else
71.             {
72.                 MessageBoxIcon icon = MessageBoxIcon.Stop;
73.                 MessageBox.Show("This user tries to forge a signature!",
    "The Deal's Off!", MessageBoxButtons.OK, icon);
74.
    Form1.Instance.PnlContainer.Controls["UserProfilePage"].BringToFront();
75.                 StaticClass.currentPage = "UserProfilePage";
76.
    Form1.Instance.PnlContainer.Controls.Remove(Controls["SendFilePage"]);
77.             }
78.         }
79.
80.         private void btnOkSignerUsername_Click(object sender, EventArgs e)
81.         {
82.             //check signer user status
83.             if (boxUsernameOfSigner.Text == "")
84.             {
```

```
                MessageBoxIcon icon = MessageBoxIcon.Error;
                MessageBox.Show("You must enter a username in this field.", "Oops...", MessageBoxButtons.OK, icon);
            }
            else
            {
                client.Send(boxUsernameOfSigner.Text);
                string response = client.Receive();
                lblSignerUserStatus.Text = response;
                if (response == "Request accepted")
                {
                    lblSignerUserStatus.ForeColor = Color.DarkSeaGreen;
                    browseFilesPanel.Visible = true;
                }
                else
                    lblSignerUserStatus.ForeColor = Color.DeepPink;

                                                    }
                                                }
                                            }
                                        }
```

SaveSignedFilePage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    public partial class SaveSignedFilePage : UserControl
    {
        private Client client;
        private string path = "C:\\Users\\idd\\Desktop\\Michals\\cyber\\Signatural\\filesReceived\\";
        public SaveSignedFilePage(Client theClient,string filename)
        {
            InitializeComponent();
            this.path = this.path + filename;
            webBrowser1.Navigate(this.path);
            webBrowser1.Invalidate();
            Form1.Instance.BackButton.Visible = false;
            client = theClient;
        }

        private void btnSaveFile_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveDialog = new SaveFileDialog
            {
                InitialDirectory = @"C:\", //root directory
                Title = "Save PDF File", //title of window
```

```
                CheckFileExists = false, //alert if user stated a nonexisting file
                CheckPathExists = true, //alert if user stated a nonexisting path
                DefaultExt = "pdf", //filter - show only pdf files
                Filter = "pdf files (*.pdf)|*.pdf",
                FilterIndex = 2,
            };
            saveDialog.FileName = filenamebox.Text;
            if (saveDialog.ShowDialog() == DialogResult.OK)
            {
                string filepath = saveDialog.FileName; // In case the user changed it inside the dialog

                // Copy file to the user's wanted location
                FileInfo fi1 = new FileInfo(this.path);
                FileInfo fi2 = new FileInfo(filepath);
                fi1.CopyTo(filepath);


Form1.Instance.PnlContainer.Controls["UserProfilePage"].BringToFront();
                StaticClass.currentPage = "UserProfilePage";

                // So the page will have to be created with a new filename
                // in each and every deal, and the page won't just be brought to front with
                // the incorrect filename.
Form1.Instance.PnlContainer.Controls.Remove(Controls["SaveSignedFilePage"]);
            }
        }
    }
}
```

SignFilePage.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    public partial class SignFilePage : UserControl
    {
        private Client client;
        private string username;
        private string rootpath = @"C:\Users\idd\Desktop\Michals\cyber\Signatural\";
        private string filepath;
```

```
20.         public SignFilePage(Client theClient,string filepath,string
    username)
21.         {
22.             InitializeComponent();
23.             btnSendFileBack.Visible = false;
24.             webBrowser1.Navigate(filepath);
25.             webBrowser1.Invalidate();
26.             Form1.Instance.BackButton.Visible = false;
27.             client = theClient;
28.             this.username = username;
29.             this.filepath = filepath;
30.         }
31.
32.         private void btnSignFile_Click(object sender, EventArgs e)
33.         {
34.             //open signature drawing window...
35.             string script = this.rootpath + "enterSignatureWindow.py";
36.             var psi = new ProcessStartInfo();
37.             psi.FileName =
    @"C:\Users\idd\AppData\Local\Programs\Python\Python38-32\python.exe";
38.             psi.Arguments = string.Format(script + " " + this.username);
39.             psi.UseShellExecute = false;
40.             psi.CreateNoWindow = true;
41.             psi.RedirectStandardOutput = true;
42.             psi.RedirectStandardError = true;
43.             var errors = "";
44.             var result = "";
45.             using (var process = Process.Start(psi))
46.             {
47.                 errors = process.StandardError.ReadToEnd();
48.                 result = process.StandardOutput.ReadToEnd();
49.                 process.WaitForExit();
50.             }
51.             // send signature, get response
52.             client.SendFile(this.rootpath + result);
53.             string response = client.Receive();
54.             if (response == "real")
55.             {
56.                 AddSignatureToFile(this.filepath, this.rootpath + result);
57.                 btnSendFileBack.Visible = true;
58.             }
59.             else
60.             {
61.                 MessageBoxIcon icon = MessageBoxIcon.Stop;
62.                 MessageBox.Show("This signature is a forgery!", "The Deal's
    Off!", MessageBoxButtons.OK, icon);
63.
    Form1.Instance.PnlContainer.Controls["UserProfilePage"].BringToFront();
64.                 StaticClass.currentPage = "UserProfilePage";
65.
    Form1.Instance.PnlContainer.Controls.Remove(Controls["SignFilePage"]);
66.             }
67.         }
68.
69.         private void AddSignatureToFile(string filepath, string signature)
70.         {
71.             string script = this.rootpath + "addSigtoPDF.py";
72.             var psi = new ProcessStartInfo();
```

```
73.             psi.FileName =
   @"C:\Users\idd\AppData\Local\Programs\Python\Python38-32\python.exe";
74.             psi.Arguments = string.Format(script + " " + filepath + " " +
   signature);
75.             psi.UseShellExecute = false;
76.             psi.CreateNoWindow = true;
77.             psi.RedirectStandardOutput = true;
78.             psi.RedirectStandardError = true;
79.             var errors = "";
80.             var result = "";
81.             using (var process = Process.Start(psi))
82.             {
83.                 errors = process.StandardError.ReadToEnd();
84.                 result = process.StandardOutput.ReadToEnd();
85.                 process.WaitForExit();
86.             }
87.         }
88.
89.         private void btnSendFileBack_Click(object sender, EventArgs e)
90.         {
91.             //send signed file to server...
92.             client.SendFile(this.filepath);
93.
   Form1.Instance.PnlContainer.Controls["UserProfilePage"].BringToFront();
94.             StaticClass.currentPage = "UserProfilePage";
95.
   Form1.Instance.PnlContainer.Controls.Remove(Controls["SignFilePage"]);
96.         }
97.     }
98. }
```