

יואב זלכה: קנבס שיתופי



הגנת סייבר

## קנבס שיתופי

מוסד לימודי: ישיבת אמית עמיחי

מגיש: יואב אליהו זלכה

ת"ז: 329461644

מנחה: שלמה וקנין

שם החלופה: הגנת סייבר

**יואב זלכה: קנבס שיתופי**

**קיץ תשפ"ד**

## תוכן עניינים

1.	מבוא.....	4
1.1	ייזום.....	4
1.2	פירוט תיאור המערכת.....	5
1.2.1	לוח זמנים:.....	6
2.	תיאור תחום הידע.....	7
2.1	פירוט היכולות.....	7
3.	מבנה / ארכיטקטורה של הפרויקט.....	11
3.1	תיאור הארכיטקטורה של המערכת המוצעת, תיאור הטכנולוגיה הרלוונטית.....	11
3.2	תרשים UML – שרת ולקוח.....	11
3.3	תיאור אלגוריתמים מרכזיים בפרויקט.....	13
3.4	כלי פיתוח, סביבת עבודה וכלים לבדיקות.....	14
3.5	תיאור פרוטוקול התקשורת.....	15
3.6	תיאור מסכי הפרויקט.....	16
3.7	תיאור מבני הנתונים.....	20
3.8	סקירת חולשות ואיומים:.....	23
4.	מימוש הפרויקט.....	25
4.1	סקירת המודלים.....	25
4.1.1	סקירת המודלים המיובאים.....	25
4.1.2	מחלקים/מחלקות שפיתחתי בשביל הפרויקט.....	27
4.1.3	הסבר על הבעיות האלגוריתמיות המרכזיות.....	38
5.	מדריך למשתמש.....	43
5.1	פירוט כלל קבצי המערכת.....	43
5.2	התקנת המערכת.....	43
5.3	משתמשי המערכת.....	44
5.4	פירוט הבדיקות.....	46
6.	סיכום אישי / רפלקציה.....	46
7.	ביבליוגרפיה.....	48
8.	קוד הפרויקט.....	49
8.1	client.py.....	49
8.2	server.py.....	61
8.3	cipher.py.....	69
8.4	drawing.py.....	70
8.5	rect_oval.py.....	72
8.6	socket_helper.py.....	73
8.7	constants.py.....	74

## 1. מבוא

### 1.1 ייזום

בסוף שנת 2022, אפל הוציאה אפליקציה למכשירים ניידים בשם "Freeform" שמטרתה היא לאפשר עבודה קולבורטית של כמה אנשים במקביל על אותו קנבס. המשתמשים יכולים להוסיף צורות, לצייר באופן חופשי, להוסיף טקסט וכו'. מיד כשהאפליקציה יצאה, בדקתי אותה יחד עם אחי. יצרנו פרויקט, והתחלנו לצייר בו ביחד, ומיד היו בעיות של סנכרון בין המכשירים. בשלב מסוים במכשירים שלנו היה מידע אחר לגמרי, בלי שום קשר אחד לשני. הניסוי הזה מיד גרם לי לשאול את עצמי מה יכול להיות כל כך קשה ביצירת קנבס שיתופי שאפילו לחברות גדולות כמו אפל יש תקלות כאלה גדולות. החלטתי לקחת את הרעיון ולהפוך אותו לפרויקט שלי בסייבר, וכך לראות את האתגרים השונים ולקבל מענה לשאלה שלי.

התחלתי לעבוד על התוכנה. המטרה שלה דומה למטרת האפליקציה Freeform, אך היא לי חשוב שהתוכנה תעבוד ותמיד תהיה מסונכרנת בין המשתמשים השונים. לא ידעתי ממש לאילו אתגרים לצפות, אך ידעתי שבעיית הסנכרון תהיה ככל הנראה הבעיה הכי קשה.

התוכנה מיועדת לעסקים שצריכים דרך לעבוד בצורה שיתופית גם מרחוק. היא מאפשרת לאנשים להתחבר ממחשבים שונים, ומיד להכניס את הרעיונות שלהם ולהציג אותם לאחרים עם סנכרון בזמן אמת.

הבעיה שניסיתי לפתור היא שאין תוכנה שמאפשרת לעבוד בצורה שיתופית במקביל לאנשים אחרים על אותו הפרויקט בצורה אמינה. המטרה העיקרית היא אמינות המערכת, ובנוסף כמות פיצ'רים מספקת.

מסקירה של האפשרויות הפופולריות, נוכחתי לראות שאין אלטרנטיבות שהן גם אמינות וגם חניניות. לדוגמה, האפליקציה Freeform שהזכרה לעיל היא כאמור אינה אמינה. בנוסף, היא בלעדית למכשירים של אפל, ולכן היא לא מהווה פיתרון לרוב העסקים.

התוכנית מבוססת על python, tkinter, sql. התקשורת מתבצעת בפרוטוקול TCP בין שקע (socket) השרת לשקעי הלקוחות בין מחשבים שמחוברים לאותה רשת פרטית.

## יואב זלכה: קנבס שיתופי

הפרויקט שלי אינו כולל משתמשים, אלא כל אחד יכול לערוך כל פרויקט שהוא יודע את הסיסמה שלו, ולכן אם קבוצה תרצה לעבוד על פרויקט מסוים הם יצטרכו לשתף את שם הפרויקט ואת סיסמתו ביניהם כדי שיוכלו לעבוד ביחד.

### 1.2 פירוט תיאור המערכת

כשהמשתמש פותח את התוכנה, הוא רואה אפשרות לפתוח פרויקט חדש, ואפשרות לפתוח פרויקט קיים.

במידה והוא בוחר באפשרות לפתוח פרויקט חדש, נפתח חלון חדש ובו המשתמש בוחר שם וסיסמה לפרויקט. אם שם הפרויקט כבר קיים במערכת, הוא יקבל הודעה שהשם תפוס ויצטרך להכניס שם חדש. אחרת, דף הפרויקט ייפתח.

במידה והוא בוחר בפרויקט קיים, ייפתח למשתמש חלון שבו הוא צריך להכניס את סיסמת הפרויקט כדי להיכנס אליו.

בחלון הפרויקט עצמו, יש למשתמש מספר אפשרויות:

- שרבוט חופשי (מברשת)
- מלבן
- אליפסה
- שינוי צבע
- שינוי עובי הקו
- שמירת התמונה כ-PNG

במערכת אין סוגים שונים של משתמשים. כל אחד יכול ליצור פרויקטים חדשים, וכל אחד יכול לגשת לכל פרויקט ולערוך אותו במידה ויש לו את הסיסמה.

#### פירוט הבדיקות:

ישנן מספר בדיקות שצריכות להתבצע כדי לראות שהפרויקט עובד ומטרות הפרויקט הושגו:

- יצירת פרויקט חדש

## יואב זלכה: קנבס שיתופי

- כניסה לפרויקט ממחשב אחר
- כניסה רק עם הסיסמה הנכונה
- ציור משני מחשבים שונים במקביל (על מנת לראות את הסנכרון ביניהם)

### 1.2.1 לוח זמנים:

11/17/2023 10:18:31

הוספתי אפשרות למשתמש לבחור רוחב בתיבת טקסט, הוספתי אפשרות למחוק את הדבר האחרון שהמשתמש צייר.

<https://stackoverflow.com/questions/8959815/restricting-the-value-in-tkinter-entry-widget>

11/17/2023 15:16:17

יצרתי מחלקה לשרטוטים ששומרת את הנקודות שלהם ואת ה-ID שלהם בקנבס, במקום לשמור רשימה של ה-IDs. לאחר מכן הוספתי אפשרות redo שמחזירה צורה שהמשתמש מחק.

11/18/2023 21:47:44

שיניתי את הדרך שבה אני מצייר מחדש ציור מאפס, הוספתי אפשרות ל-CTRL+Z ו-CTRL+Y בשביל undo ו-redo. נעזרתי ב-:

<https://stackoverflow.com/questions/51118695/python-tkinter-how-to-create-line-with-multiple-points-in-a-list-tuple>

11/19/2023 19:07:32

הוספת אפשרות שמירה וטעינה של הקנבס. נעזרתי ב-:

<https://www.programiz.com/python-programming/methods/built-in/eval>

11/20/2023 23:05:35

הוספת אפשרות לשמירה ופתיחה של פרויקטים שונים.

11/20/2023 23:05:35

נעזרתי ב-:

<https://stackoverflow.com/questions/69147762/why-is-the-variable-always-set-to-the-last-item-in-the-list>

11/22/2023 22:18:05

עדכון בזמן אמת של ה-DB, הוספת פונקציונליות בסיסית של שרת ולקוח (בלי יישום של העברת הציורים עדיין) נעזרתי ב-:

<https://quizdeveloper.com/faq/typeerror-send-argument-after-must-be-an-iterable-not-socket-in-python-aid2325>

11/24/2023 16:05:33

הוספת אפשרות לכמה משתמשים לצייר במקביל ולראות את הציורים שאחרים מוסיפים. נעזרתי ב-:

<https://stackoverflow.com/questions/17667903/python-socket-receive-large-amount-of-data>

## יואב זלכה: קנבס שיתופי

סיום המעבר לגישה ל-database רק דרך השרת (יצירת db חדש, טעינת db קיים, גישה לנתונים נוספים ששמורים ב-db דרך השרת.  
נעזרתי ב-:

<https://youtu.be/SDAkQq17S2Q?si=cyvBzklfhk6r1ToE>

## 2. תיאור תחום הידע

### 2.1 פירוט היכולות

שם היכולת: חיבור משתמשים חדשים

- מהות: חיבור משתמשים חדשים לשרת, על מנת שיוכלו לתקשר איתו, וכך להתעדכן על מצב הקנבס, ולשנות את מצב הקנבס בעצמם.
- אוסף יכולות נדרשות:
  - יצירת סוקט
  - המתנה לחיבור משתמשים חדשים
  - פתיחת ת'רד עבור כל משתמש על מנת לתקשר עימו בהמשך
  - הוספת המשתמש לרשימה
  - יצירת מפתח משותף עבור ההצפנה
  - חיבור לשרת המשני
- אובייקטים נדרשים: Client, Tkinter, Socket, Thread, Cipher, DiffieHellman

שם היכולת: כניסה לפרויקט

- מהות: כניסת המשתמש לפרויקט בצורה בטוחה, כדי שרק משתמשים היודעים את סיסמת הפרויקט יוכלו לראות ולערוך אותו.
- אוסף יכולות נדרשות:
  - ממשק משתמש: בחירת פרויקט מרשימת הפרויקטים
  - ממשק משתמש: הזנה של סיסמת הפרויקט
  - שליחה מוצפנת של הסיסמה
  - בצד השרת:
    - אימות הסיסמה
    - שיוך הפרויקט למשתמש
    - שליחת נתוני הפרויקט למשתמש (יפורט בהמשך)

## יואב זלכה: קנבס שיתופי

- אובייקטים נחוצים: Tkinter, Thread, hashlib, Client, SelectProjectGUI, sqlite3, EnterPasswordGUI

שם היכולת: יצירת פרויקט חדש

- מהות: יצירת פרויקט חדש במידה והמשתמש רוצה לעבוד על משהו חדש ואחר ממה שכבר פתח
- אוסף יכולות נדרשות:
  - ממשק משתמש: כפתור לפתיחת פרויקט חדש
  - ממשק משתמש: חלון להזנת שם הפרויקט וסיסמה
  - הצפנת הסיסמה
  - שליחת שם המשתמש והסיסמה המוצפנת לשרת

בצד השרת:

- קבלת שם הפרויקט והסיסמה
- יצירת קובץ חדש במאגר המידע עבור הפרויקט, יצירת טבלה לכל סוג ציור, וטבלה לקבועים (ID וסיסמה).
- אובייקטים נחוצים: sqlite3, Client, NewProjectGUI

שם היכולת: ציור חופשי

- מהות: ציור שאינו מוגבל לצורות מסוימות ומאפשר למשתמש לצייר כל דבר שירצה
- אוסף יכולות נדרשות:
  - ממשק משתמש של קנבס
  - קליטת תנועת העבר ולחיצה עליו
  - יצירת משתנה – cur\_drawing, המכיל את המידע של הציור (הנקודות שבהן העכבר עבר, המתעדכן כאשר העכבר עובר בנקודה חדשה
  - ציור בזמן אמת של תנועת העכבר על המסך.
  - קבלת ID ייחודי בתוך הפרויקט לציור (שונה מה-ID של tkinter, שתלוי בקנבס)



## יואב זלכה: קנבס שיתופי

- שליחת הציור לשרת, ושמירה שלו במאגר המידע
- עדכון של משתמשים אחרים בשינוי
- אובייקטים נחוצים: tkinter, Drawing, socket

שם היכולת: יצירת צורה (מלבן/אליפסה)

- מהות: לאפשר למשתמש להכניס צורות פשוטות בצורה מדויקת.
- אוסף יכולות נדרשות:
  - כפתור לכל סוג ציור/צורה
  - קליטת תחילת הלחיצה על העכבר
  - עדכון בזמן אמת של הצורה באמצעות מחיקת הצורה לפני שהעכבר זז
  - ויצירת צורה חדשה לפי המיקום החדש של העכבר
  - עדכון של השרת, וממנו עדכון של מאגר המידע (חלק ייחודי לכל סוג צורה), ועדכון של משתמשים אחרים המחוברים לפרויקט.

שם היכולת: מחיקת ציור

- מהות: מחיקה של הציור האחרון שהמשתמש צייר
- אוסף יכולות נדרשות:
  - שמירת הסדר של הציורים, כדי שהמשתמש יוכל למחוק את האחרון, ולאחר מכן את זה שבא לפניו.
  - שמירת ה-ID של הציורים בקנבס
  - שליחת השינוי לשרת
  - עדכון במאגר המידע
  - עדכון למשתמשים אחרים
- אובייקטים נחוצים: Drawing, tkinter.canvas, socket

שם היכולת: בחירת צבע

## יואב זלכה: קנבס שיתופי

- מהות: בחירה חופשית של צבע עבור הציור/צורה שהמשתמש רוצה ליצור
- אוסף יכולות נדרשות:
  - כפתור לשינוי הצבע, המראה את הצבע הנוכחי
  - חלון לשינוי חופשי של הצבע (מובנה במערכת ההפעלה)
  - עדכון של משתנה הצבע
- אובייקטים נחוצים: `tkinter.button`, `tkinter.Color_Picker`

שם היכולת: בחירת עובי קו

- מהות: בחירה חופשית של המשתמש בעובי הקו שהוא מעוניין בו עבור ציור חופשי או עבור מתאר צורה
- אוסף יכולות נדרשות:
  - תיבת טקסט לשינוי העובי, המוגבלת למספרים
  - משתנה השומר את עובי הקו
- אובייקטים נחוצים: `tkinter.Entry`

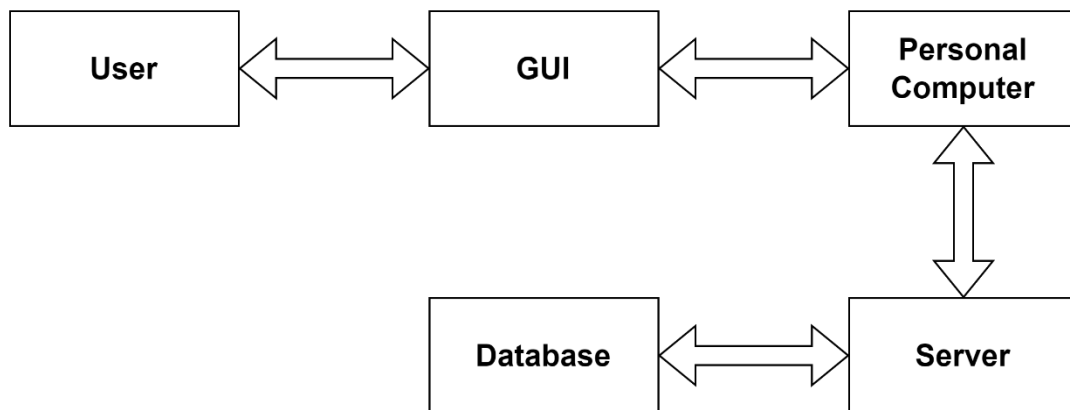
### 3. מבנה / ארכיטקטורה של הפרויקט

#### 3.1 תיאור הארכיטקטורה של המערכת המוצעת, תיאור הטכנולוגיה הרלוונטית

תיאור החומרה:

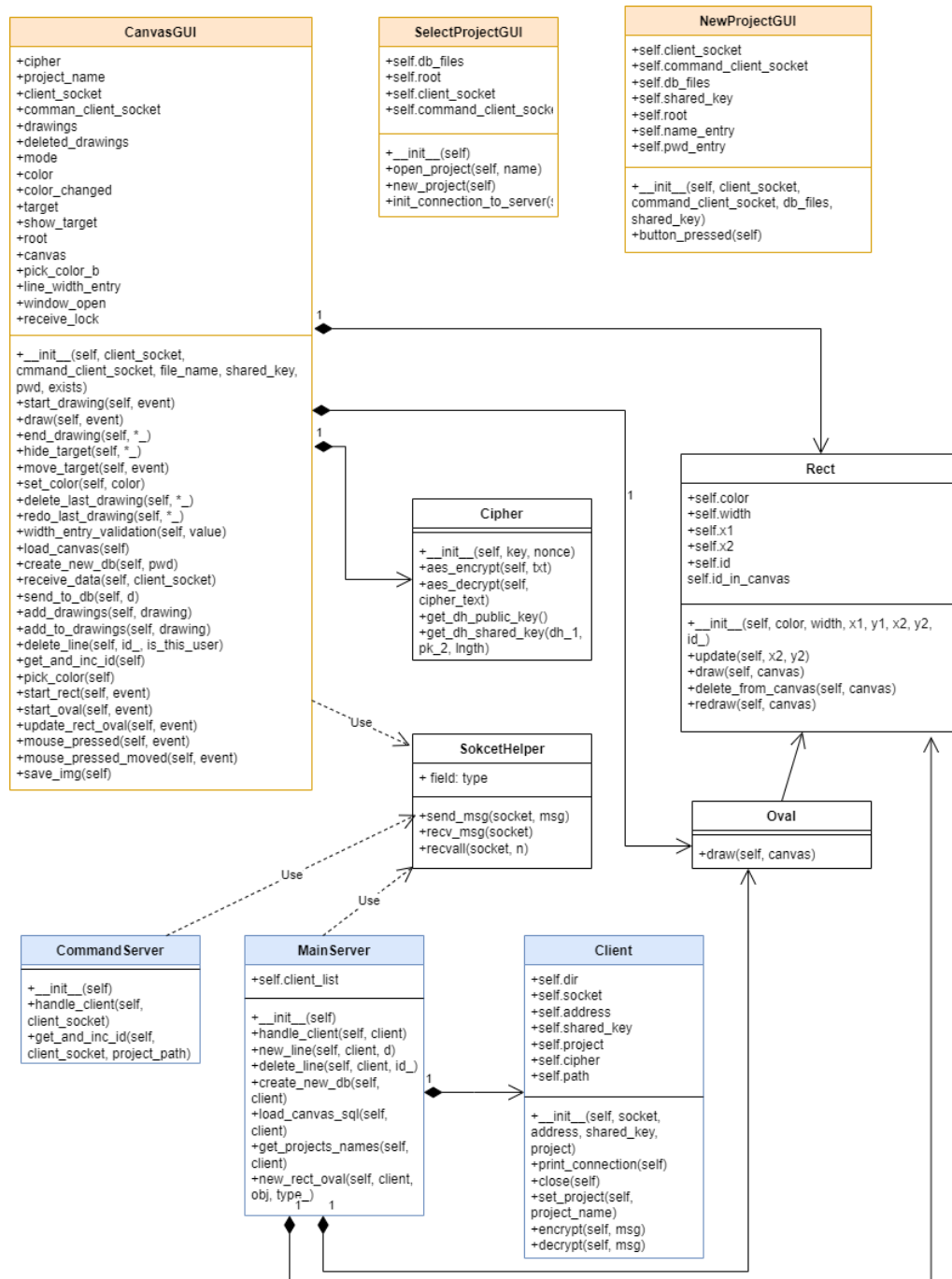
שרת – מחשב המריץ קוד פייתון ויש לו גישה לרשת.

לקוח – מחשב המריץ קוד פייתון ויש לו גישה לרשת, עכבר.



#### 3.2 תרשים UML – שרת ולקוח

## יואב זלכה: קנבס שיתופי



### 3.3 תיאור אלגוריתמים מרכזיים בפרויקט

#### ציור חופשי:

אחת מהפונקציות המרכזיות של הפרויקט היא האפשרות לצייר בצורה חופשית על הקנבס (מברשת). לפני שהתחלתי לעבוד על הפונקציה הזאת חשבתי שיהיה מאוד פשוט ליישם אותה, אך מהר גיליתי שאין כך הדבר.

הבעיה: כיצד יוצרים קו בעובי נבחר לפי מסלול העכבר.

#### אלגוריתמים קיימים:

האלגוריתם הראשון שניסיתי הוא ציור עיגול (לפי העובי שהשתמש בחר) בכל נקודה שבה העבר עובר, אך הפתרון הזה כשל. Tkinter לא מעדכן את מיקום העכבר בקצב מספק כדי שיהיה קו רציף בין הנקודות, ובמקום קו של מברשת יש הרבה נקודות בודדות. האלגוריתם השני שניסיתי הוא יצירת קו בעובי שהשתמש בחר בין כל שתי נקודות שעובר בהן העכבר. הבעיה: כשהעובי גדול, כל תנועה קטנה של העכבר יוצרת קו גדול, אנכי לתנועת העכבר.

#### הפתרון:

את הפתרון שחשבתי עליו לא ראיתי באף מקור, מכיוון שברוב הסביבות מיקום העכבר מתעדכן מספיק מהר כדי שאפשר יהיה להשתמש בפתרון של העיגולים ללא בעיה.

האלגוריתם: ציור צורת גלולה (pill shape), הכוללת שני עיגולים בנקודת ההתחלה ובנקודת הסוף, וקו באמצע. הפתרון הזה עובד בכל עובי ולכל תנועה.

### בקשת מידע ספציפי משרת, תוך כדי המתנה לקבלת ציורים מהשרת בכל רגע

#### נתון:

#### הבעיה:

הלקוח שולח בקשה לשרת, והשרת עונה לבקשה, אך הלקוח עלול לקבל את המידע כציור, מכיוון שהוא מחכה לקבל מידע מהשרת בשני מקומות שונים.

#### פתרונות קיימים:

1. API – להחליף את כל המערכת של הפרויקט ב-API, כך השרת ידע להפריד בין בקשות שונות והלקוח ידע לזהות תגובות לבקשות שונות.

## יואב זלכה: קנבס שיתופי

2. סיווג המידע בתחילת הפקטה – כך הלקוח ידע לזהות אם מה שנשלח אליו זה ציור או המידע הספציפי שהוא ביקש.

3. יצירת שרת נוסף, "שרת פקודות"/"command server", שנועד להתנהג בצורה דומה ל-API – לנהל בקשות. כשהלקוח ירצה בקשה מהשרת, הוא ישלח לשרת הפקודות, והוא ידע למה ההודעה שהוא מקבל מהשרת מתייחסת, כי הוא מקבל אותה משרת נפרד.

### פתרון נבחר:

בחרתי בפתרון השלישי שציינתי, משום שהפרויקט כבר היה בשלב מתקדם ולא רציתי לשנות את כל המערכת שלו, ומשום שהוא יחסית קל ליישום לפרויקטים קטנים.

## 3.4 כלי פיתוח, סביבת עבודה וכלים לבדיקות

### פיתוח:

שפת תכנות: Python 3.11, 3.12.

עורך טקסט: ערכתי את הפרויקט בעזרת VSCode

Github: השתמשתי ב-Github כדי לנהל את השינוי בגרסאות הפרויקט

### ספריות:

tkinter: ספריית ממשק המשתמש הגרפי עבור Python.

pickle: ספרייה לטעינה ושמירה של אובייקטים Python.

threading: ספרייה לניהול משימות בו-זמניות.

### סביבת עבודה לבדיקות:

מערכת הפעלה: Windows, macOS, Linux (כל מערכת תואמת ל-Python 3.x).

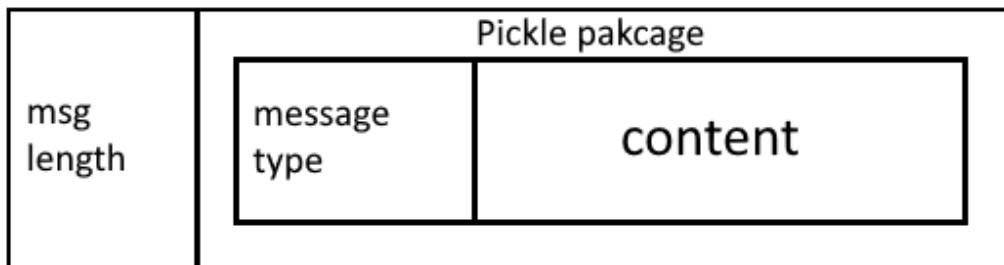
חיבור אינטרנט יציב (חשוב לפיתוח ותקשורת מרובת משתמשים).

כל הספריות מותקנות.

### 3.5 תיאור פרוטוקול התקשורת

התקשורת בפרויקט מתקיימת בפרוטוקול TCP. בכל שליחת הודעה מהלקוח לשרת, יש הוראה בתחילת ההודעה, ובהמשך יש את תוכן ההודעה (כל זה עטוף בעזרת Pickle). כך השרת יודע איך להגיב לכל הודעה שנשלחת אליו מהלקוחות. בנוסף, לפני כל המידע בהודעה כתוב אורך ההודעה (ב-4 הבתים הראשונים), כך הצד המקבל יודע מה אורך ההודעה ויודע כמה מידע לקבל. הצד המקבל מקבל מידע מהצד השולח עד שהוא מגיע לאורך שצוין ב-4 הבתים הראשונים.

לסיכום, פקטה נראית כך:



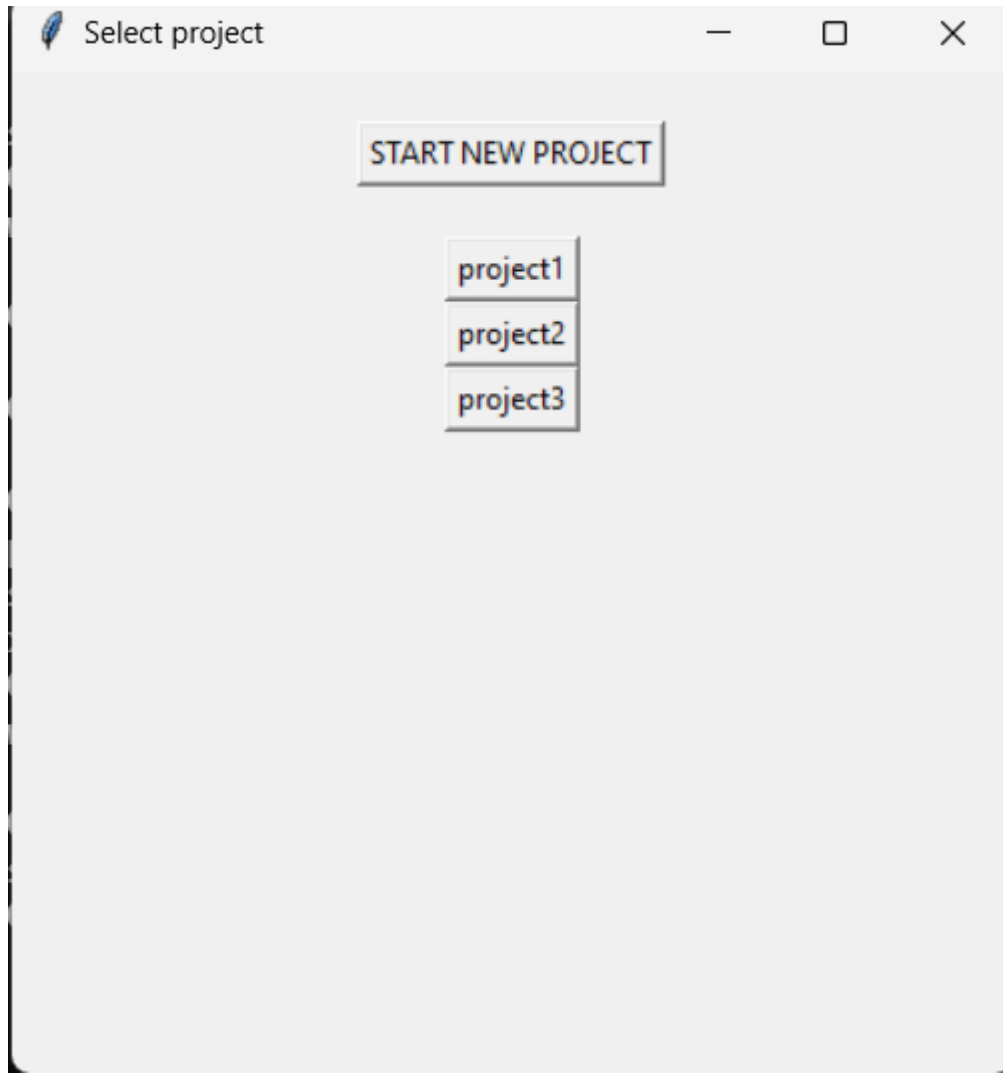
באופן דומה, כשהשרת מקבל ציור בודד ושולח אותו לשאר המשתמשים המחוברים לפרויקט, הפקטה זהה במבנה.

מקרים חריגים - כשהשרת מקבל בקשה ממשתמש שהתחבר לקבל את כל המידע של הפרויקט, הוא שולח ללקוח את המידע ללא סיווג המידע (מכיוון שהלקוח יודע שזה מה שהשרת שולח), אותו הדבר מתקיים כאשר שרת הפקודות שולח מידע ללקוח.

### 3.6 תיאור מסכי הפרויקט

מסך ראשון – בחירת פרויקט:

במסך הזה, שנפתח כאשר לקוח פותח את הפרויקט, יש אפשרות לפתוח פרויקט חדש, ויש אפשרות להיכנס לפרויקט קיים (יש כפתור עבור כל פרויקט, וכפתור עבור יצירת פרויקט חדש).

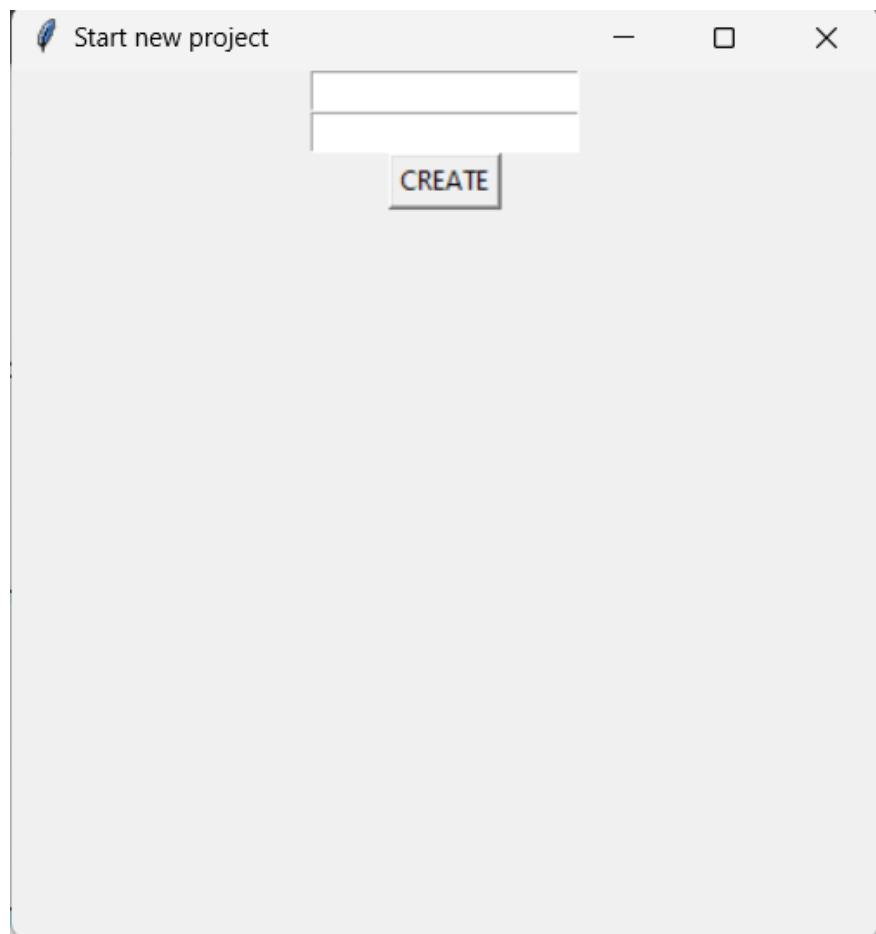




## יואב זלכה: קנבס שיתופי

### מסך יצירת פרויקט:

אם המשתמש בוחר לפתוח פרויקט חדש, המסך הזה נפתח לו. במסך הזה ישנם שני שדות טקסט – שדה לשם הפרויקט ושדה לסיסמת הפרויקט. בנוסף, יש במסך כפתור, אשר הלקוח לוחץ עליו לאחר שהכניס את המידע הנדרש בשדות.



### מסך הכנסת סיסמת פרויקט:

אם המשתמש בחר לפתוח פרויקט קיים, זה המסך שנפתח. במסך הזה המשתמש נדרש להכניס את סיסמת הפרויקט על מנת להיכנס לפרויקט.

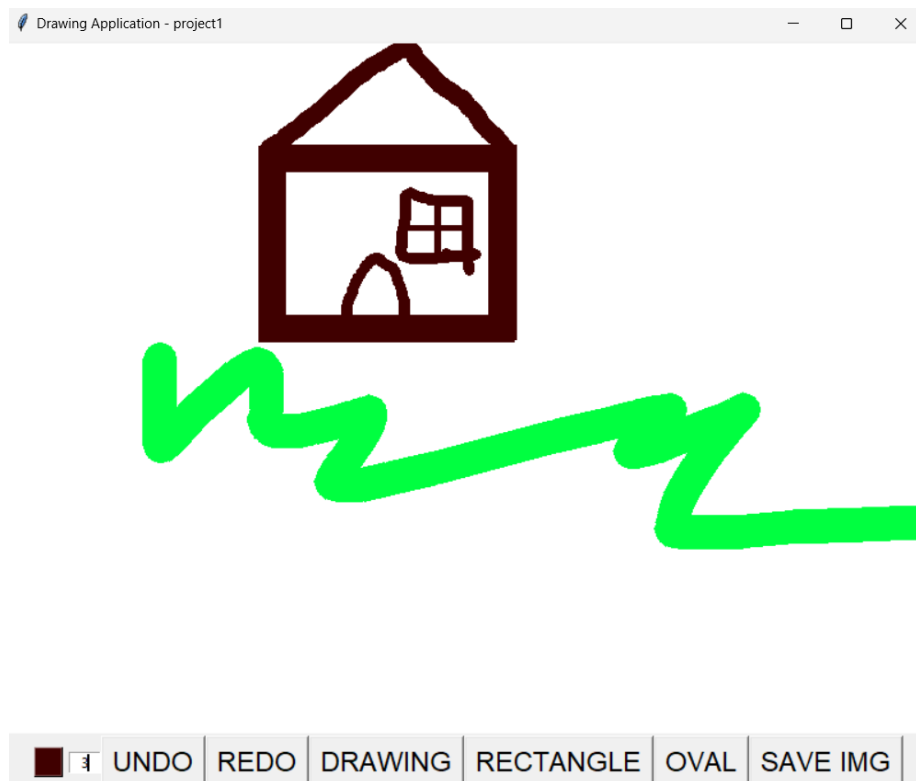
### מסך עיקרי – עריכת הפרויקט:

לאחר שהמשתמש נכנס לפרויקט (בין אם פרויקט חדש ובין אם פרויקט קיים), זה המסך

## יואב זלכה: קנבס שיתופי

שנפתח. במסך הזה המשתמש רואה את הקנבס כמו שהוא, ומתעדכן בכל שינוי שנעשה בו במחשבים אחרים. בנוסף, למשתמש מוצבות מספר אפשרויות לעריכת הקנבס: ציור חופשי (מברשת), יצירת צורה (מלבן/אליפסה), קביעת עובי המברשת / מתאר הצורה, קביעת הצבע שהמשתמש מעוניין לצייר איתו (במסך בחירת צבע).

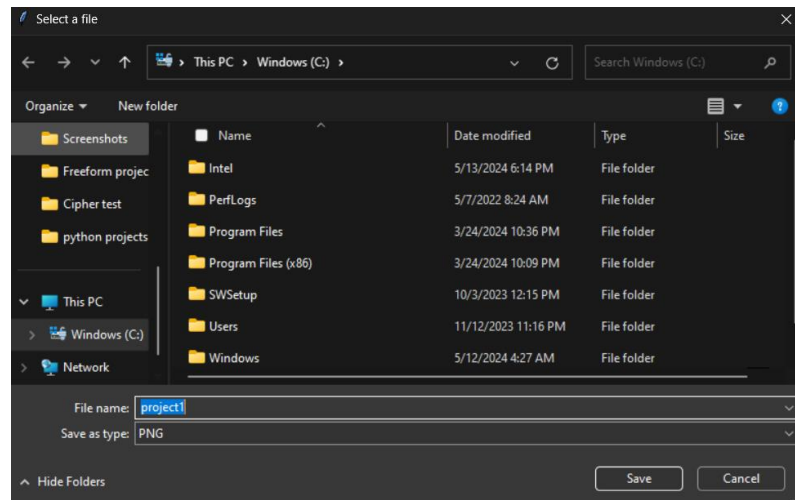
בנוסף: ישנו כפתור המאפשר למשתמש לשמור את הקנבס כ-PNG במחשב שהוא משתמש בו (במסך בחירת תיקייה), וישנם כפתורים של חזרה אחורה וקדימה Undo/Redo. לחלופין, המשתמש יכול להשתמש בקיצורים Ctrl+Z ו-Ctrl+Y בשביל לחזור אחורה וקדימה.



### מסך משני – בחירת תיקייה

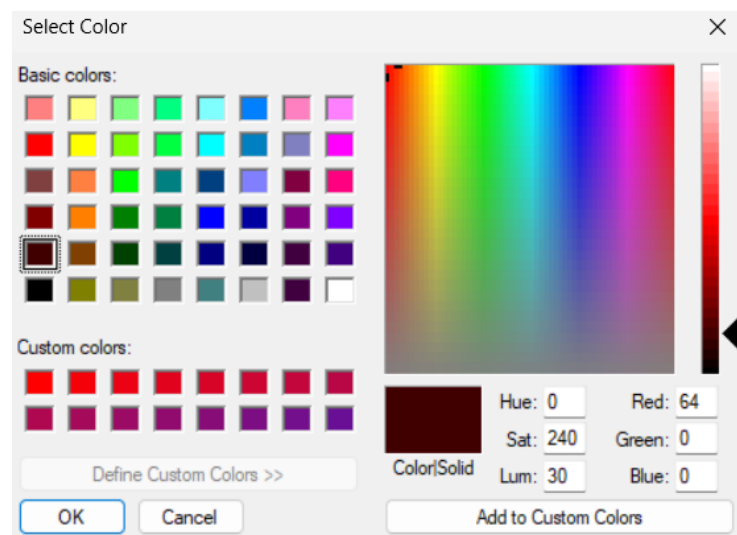
אם המשתמש שומר את הקנבס כתמונה, המסך הזה נפתח ומאפשר למשתמש לנווט בתיקיות שבמחשב שלו ולבחור מיקום לשמירת התמונה. המסך הזה הוא מסך מובנה במערכת ההפעלה.

## יואב זלכה: קנבס שיתופי

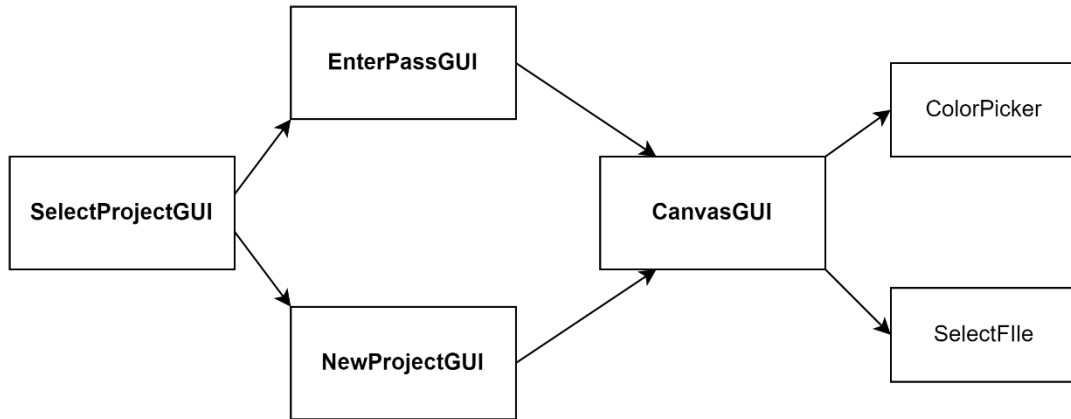


### מסך משני – בחירת צבע חופשית:

אם המשתמש בוחר לשנות את הצבע שהוא מצייר איתו, המסך הזה נפתח. המסך כולל בחירת צבע חופשית ממניפת צבעים, הכנסת ערכי RGB, ושמירת צבעים שהמשתמש מתכוון לעשות בהם שימוש נוסף בהמשך. המסך מובנה במערכת ההפעלה.



תרשים מסכים:



### 3.7 תיאור מבני הנתונים

המידע של כל הפרויקטים שמור בתיקייה ייעודית במחשב השרת "projects". לכל פרויקט יש קובץ ייעודי הנושא את השם שלו (לדוגמה, לפרויקט "project1" יהיה קובץ תואם "project1.db". בכל קובץ כזה יש 5 טבלאות – טבלה של כל ציורי המברשת, טבלה של כל המלבנים, טבלה של כל האליפסות, טבלה של משתנים מסוג מספרים שלמים (id הכי גבוה של ציור), טבלה של משתנים מסוג מערך בתים (סיסמה מוצפנת)

א. טבלאות:

1. drawings (ציורים):

- שדה: id (מזהה)
  - טיפוס: INTEGER
  - מפתח ראשי
  - תיאור: מזהה ייחודי לכל ציור.
- שדה: color (צבע)

## יואב זלכה: קנבס שיתופי

- טיפוס: TEXT
- תיאור: צבע הציור (לדוגמה, "אדום", "FF0000").
- שדה: width (עובי קו)
  - טיפוס: INTEGER
  - תיאור: עובי הקו של הציור.
- שדה: pt\_list (רשימת נקודות)
  - טיפוס: TEXT
  - תיאור: מאחסן רשימה של נקודות המגדירות את צורת הציור (פורמט JSON אפשרי).

### 2. rects (מלבנים):

- שדה: id (מזהה)
  - טיפוס: INTEGER
  - מפתח ראשי: PRIMARY KEY
  - תיאור: מזהה ייחודי לכל מלבן.
- שדה: color (צבע)
  - טיפוס: TEXT
  - תיאור: צבע המתאר של המלבן (לדוגמה, "אדום", "FF0000").
- שדה: width (עובי קו)
  - טיפוס: INTEGER
  - תיאור: עובי הקו של המלבן.
- שדה: x1 (קואורדינטת X של הפינה הראשונה של המלבן)
  - טיפוס: INTEGER
  - תיאור: קואורדינטת X של הפינה הראשונה של המלבן (לפי סדר הציור של המשתמש)
- שדה: y1 (קואורדינטת Y של הפינה הראשונה של המלבן)
  - טיפוס: INTEGER
  - תיאור: קואורדינטת Y של הפינה הראשונה של המלבן (לפי סדר הציור של המשתמש)
- שדה: x2 (קואורדינטת X של הפינה השנייה של המלבן)
  - טיפוס: INTEGER

## יואב זלכה: קנבס שיתופי

- תיאור: קואורדינטת X של הפינה השנייה של המלבן (לפי סדר הציור של המשתמש)

- שדה: y2 (קואורדינטת Y של הפינה השנייה של המלבן)

- טיפוס: INTEGER

- תיאור: קואורדינטת Y של הפינה השנייה של המלבן (לפי סדר הציור של המשתמש)

3. ovals (אליפסות):

- שדה: id (מזהה)

- טיפוס: INTEGER

- מפתח ראשי: PRIMARY KEY

- תיאור: מזהה ייחודי לכל אליפסה.

- שדה: color (צבע)

- טיפוס: TEXT

- תיאור: צבע המתאר של האליפסה (לדוגמה, "אדום", "#FF0000").

- שדה: width (עובי קו)

- טיפוס: INTEGER

- תיאור: עובי הקו של האליפסה.

- שדה: x1 (קואורדינטת X של הפינה הראשונה של האליפסה)

- טיפוס: INTEGER

- תיאור: קואורדינטת X של הפינה הראשונה של האליפסה (לפי סדר הציור של המשתמש)

- שדה: y1 (קואורדינטת Y של הפינה הראשונה של האליפסה)

- טיפוס: INTEGER

- תיאור: קואורדינטת Y של הפינה הראשונה של האליפסה (לפי סדר הציור של המשתמש)

- שדה: x2 (קואורדינטת X של הפינה השנייה של האליפסה)

- טיפוס: INTEGER

- תיאור: קואורדינטת X של הפינה השנייה של האליפסה (לפי סדר הציור של המשתמש)

- שדה: y2 (קואורדינטת Y של הפינה השנייה של האליפסה)

- טיפוס: INTEGER

## יואב זלכה: קנבס שיתופי

- תיאור: קואורדינטת Y של הפינה השנייה של האליפסה (לפי סדר הציור של המשתמש)

4. int\_variables (משתנים):

- שדה: name (שם)
  - טיפוס: INTEGER
  - תיאור: שם המשתנה
- שדה: value (ערך)
  - טיפוס: INTEGER
  - תיאור: ערך המשתנה

5. bytes\_variables (משתנים):

- שדה: name (שם)
  - טיפוס: INTEGER
  - תיאור: שם המשתנה
- שדה: value (ערך)
  - טיפוס: BYTES
  - אורך: 32 בתיים
  - תיאור: ערך המשתנה

## 3.8 סקירת חולשות ואיומים:

- SQL injection – הקוד חסין להזרקת SQL, מכיוון שבשום שלב אין למשתמש שליטה ישירה על מה מבוצע בקוד SQL, אלא רק בעזרת הפרמטרים של SQLite, שעמידים בהזרקת SQL (כתיבת סימני שאלה במקומות שאני רוצה שיהיו תלויים במשתמש).
- תהליך ה-login בטוח, הסיסמה מקודדת (hashed) ואפילו אם פורץ יקבל גישה לסיסמה כפי שהיא שמורה במסד הנתונים, הוא לא יוכל לדעת מה הסיסמה. (עם זאת, הקוד חשוף לשימוש ב-rainbow tables כדי לפרוץ)

## יואב זלכה: קנבס שיתופי

- MITM – המערכת אינה חשופה לתקיפת MIDM, משום שהמידע שעובר (שחשוב שיוצפן) מוצפן מצקה לקצה בפרוטוקול AES.
- הקוד חשוף לפגיעות DOS/DDOS, אין שימוש ב-Firewalls.



## 4. מימוש הפרויקט

### 4.1 סקירת המודלים

#### 4.1.1 סקירת המודלים המיובאים

:Tkinter

מטרה: ספריה זו משמשת לבניית ממשקי משתמש גרפיים (GUI) ב-Python. היא מספקת פונקציות ורכיבים רבים ליצירת חלונות, תפריטים, כפתורים, תיבות טקסט ועוד. כל ממשקי המשתמש בפרויקט מבוססים עליה.

:Pickle

מטרה: ספריה זו משמשת לשמירה ושחזור אובייקטים של Python. היא מאפשרת להמיר אובייקטים למבנה נתונים בינארי, המאפשר לאחסן אותם בקבצים או להעביר אותם ברשת. בפרויקט, השתמשתי ב-Pickle כדי להעביר מידע בין הלקוח לשרת ולהפך.

:hashlib

מטרה: ספריה זו מספקת פונקציות לחישוב פונקציות גיבוב (hash functions). בפרויקט השתמשתי בפונקציה sha256 הנמצאת במחלקה כדי לגבב את הסיסמאות של הפרויקטים.

:socket

מטרה: ספריה זו משמשת לתקשורת רשת ב-Python. בפרויקט השתמשתי בה כדי ליצור חיבור ברשת בין השרת ללקוחות, בפרוטוקול TCP.

:pathlib

מטרה: ספריה זו משמשת לעבודה עם נתיבי קבצים ב-Python. היא מספקת פונקציות ליצירת, ניתוח ושליטה בנתיבי קבצים, המאפשרת לעבוד עם קבצים ותיקיות בקלות. בפרויקט השתמשתי בה בשרת כדי לנהל את קבצי מסד הנתונים.

## יואב זלכה: קנבס שיתופי

:threading

מטרה: ספריה זו משמשת ליצירת וניהול Threads, המאפשרים להריץ מספר משימות בו זמנית. בפרויקט נעשה שימוש נרחב ב-threading – בלקוח יש thread המנהל את פעולות המשתמש, ובשרת יש thread המנהל הודעות מכל לקוח.

:sqlite3

מטרה: ספריה זו משמשת לעבודה עם מסדי נתונים ב-Python. בפרויקט השתמשתי בה כדי ליצור ולעשות שינויים בקבצי מסד הנתונים.

:PIL

מטרה: ספריה זו משמשת לעבודה עם תמונות ב-Python. בפרויקט השתמשתי בה כדי לאפשר למשתמש לשמור את הציור שעל הקנבס כקובץ PNG (כשהמשתמש לוחץ על כפתור השמירה, המערכת יוצרת תמונת PIL ומוסיפה לה את כל האובייקטים שבקנבס, ושומרת).

:tkinter.colorchooser

מטרה: מודול זה משמש לבחירת צבעים ב-Tkinter. בפרויקט הוא מאפשר למשתמש לבחור צבעים בצורה חופשית ולשמור צבעים שהוא רוצה להשתמש בהם בהמשך.

## יואב זלכה: קנבס שיתופי

### 4.1.2 מודולים/מחלקות שפיתחתי בשביל הפרויקט

#### CanvasGUI

- **תפקיד:** מממשק גרפי לניהול וביצוע פעולות על קנבס ציור.
- **משתנים:**
  - `cipher`: אובייקט הצפנה לטיפול בהצפנה ובפענוח של נתונים.
  - `project_name`: שם הפרויקט.
  - `client_socket`: סוקט הלקוח שמשמש לתקשורת עם השרת.
  - `command_client_socket`: סוקט הלקוח שמשמש לתקשורת עם שרת הפקודות.
  - `drawings`: רשימת הציורים הקיימים בקנבס.
  - `deleted_drawings`: רשימת הציורים שנמחקו.
  - `mode`: המצב הנוכחי של הממשק (ציור, מלבן וכו').
  - `color`: הצבע הנוכחי לציור.
  - `color_changed`: דגל המציין אם הצבע השתנה.
  - `target`: סמן המראה איפה המשתמש עומד לצייר.
  - `show_target`: דגל המציין אם הסמן מוצג.
  - `root`: שורש ה-UI של הממשק.
  - `canvas`: אובייקט הקנבס לציור.
  - `pick_color_b`: כפתור לבחירת צבע.
  - `line_width_entry`: שדה להזנת רוחב הקו.
  - `window_open`: דגל המציין אם חלון הממשק פתוח (נועד כדי לסגור כראוי את התוכנה בסדר הנכון).
  - `receive_lock`: מנעול לסינכרון קבלת נתונים.

## יואב זלכה: קנבס שיתופי

### • פעולות:

- `init__(self, client_socket, command_client_socket, file_name, __`  
`shared_key, pwd, exists)`
- טענת כניסה: פרטי הלקוח (`client_socket`), סוקט פקודות  
(`command_client_socket`), שם קובץ, מפתח משותף  
(`shared_key`), סיסמה (`pwd`), דגל המציין אם הקובץ קיים.
- טענת יציאה: אתחול אובייקט CanvasGUI עם הפרטים שנמסרו.
- `start_drawing(self, event)`
- טענת כניסה: מיקום העכבר.
- טענת יציאה: התחלת ציור חדש בקנבס.
- `draw(self, event)`
- טענת כניסה: מיקום העכבר.
- טענת יציאה: המשך הציור בהתאם למיקום העכבר.
- `end_drawing(self, event)`
- טענת כניסה: מיקום העכבר.
- טענת יציאה: סיום ציור בקנבס.
- `hide_target(self, __)`
- טענת כניסה: אין.
- טענת יציאה: הסתרת סמן המברשת (`target`).
- `move_target(self, event)`
- טענת כניסה: מיקום העכבר.
- טענת יציאה: הזזת הסמן בהתאם למיקום העכבר.
- `set_color(self, color)`
- טענת כניסה: צבע.
- טענת יציאה: הגדרת הצבע לציור.

## יואב זלכה: קנבס שיתופי

- `delete_last_drawing(self, _*)`
- טענת כניסה: אין.
- טענת יציאה: מחיקת הציור האחרון.
- `redo_last_drawing(self)`
- טענת כניסה: אין.
- טענת יציאה: ביצוע מחדש של הציור האחרון שנמחק.
- `width_entry_validation(self, value)`
- טענת כניסה: ערך רוחב.
- טענת יציאה: אימות הערך שנמסר, לבדיקה שהוא מספרי. (משמש לשדה להזנת רוחב הקו)
- `load_canvas(self)`
- טענת כניסה: אין.
- טענת יציאה: טעינת הקנבס מהקובץ, דרך השרת.
- `create_new_db(self, pwd)`
- טענת כניסה: סיסמה.
- טענת יציאה: יצירת מסד נתונים חדש, בשרת.
- `receive_data(self, client_socket)`
- טענת כניסה: סוקט לקוח.
- טענת יציאה: קבלת נתונים מהסוקט וביצוע פעולות בהתאם.
- `send_to_db(self, drawing)`
- טענת כניסה: אובייקט ציור.
- טענת יציאה: שליחת הציור לשרת.
- `add_to_drawings(self, drawing)`
- טענת כניסה: אובייקט ציור.

## יואב זלכה: קנבס שיתופי

- טענת יציאה: הוספת הציור לרשימת הציורים.
- `save_img(self)`
- טענת כניסה: אין.
- טענת יציאה: שמירת התמונה במחשב.

## SelectProjectGUI

- **תפקיד:** ממשיק לבחירת פרויקט קיים.
- **משתנים:**
  - `db_files`: רשימת קבצי מסד הנתונים הזמינים, לפי מה שהשרת שלח.
  - `root`: שורש ה UI של הממשק.
  - `client_socket`: סוקט הלקוח שמשמש לתקשורת עם השרת.
  - `command_client_socket`: סוקט הלקוח שמשמש לתקשורת עם שרת הפקודות.
- **פעולות:**
  - `init__(self)___`
  - טענת כניסה: אין.
  - טענת יציאה: אתחול אובייקט `SelectProjectGUI`.
  - `open_project(self)`
  - טענת כניסה: אין.
  - טענת יציאה: פתיחת פרויקט קיים.
  - `new_project(self)`
  - טענת כניסה: אין.
  - טענת יציאה: פתיחת מסך ליצירת פרויקט חדש.
  - `init_connection_to_server(self)`

## יואב זלכה: קנבס שיתופי

- טענת כניסה: אין.
- טענת יציאה: יצירת חיבור לשרת ולשרת הפקודות.

### NewProjectGUI

- **תפקיד:** ממשיק ליצירת פרויקט חדש.
- **משתנים:**
  - `client_socket`: סוקט הלקוח שמשמש לתקשורת עם השרת.
  - `command_client_socket`: סוקט פקודות נוסף לתקשורת עם השרת.
  - `db_files`: רשימת קבצי מסד הנתונים הזמינים, לפי מה שהשרת שלח.
  - `shared_key`: מפתח משותף להצפנה.
  - `root`: שורש ה UI של הממשק.
  - `name_entry`: שדה להזנת שם הפרויקט.
  - `pwd_entry`: שדה להזנת הסיסמה.
- **פעולות:**
  - `init__(self, client_socket, command_client_socket, db_files, __, shared_key)`
    - טענת כניסה: סוקט הלקוח (`client_socket`), סוקט פקודות (`command_client_socket`), קבצי מסד נתונים (`db_files`), מפתח משותף להצפנה (`shared_key`).
  - טענת יציאה: אתחול אובייקט NewProjectGUI עם הפרטים שנמסרו (על ידי השרת) ופתיחה של הפרויקט.
  - `button_pressed(self)`
    - טענת כניסה: אין.
    - טענת יציאה: טיפול בלחיצה על כפתור יצירת פרויקט חדש.

## יואב זלכה: קנבס שיתופי

### Cipher

- **תפקיד:** הצפנה ופענוח של נתונים.
- **משתנים:**
  - **key:** מפתח הצפנה.
  - **nonce:** מספר אקראי שמשמש להצפנה.
- **פעולות:**
  - `init__(self, key, nonce)___`
    - טענת כניסה: מפתח (key) ו- nonce.
    - טענת יציאה: אתחול אובייקט Cipher עם המפתח וה- nonce שנמסרו.
  - `aes_encrypt(self, txt)`
    - טענת כניסה: טקסט להצפנה.
    - טענת יציאה: טקסט מוצפן.
  - `aes_decrypt(self, cipher_text)`
    - טענת כניסה: טקסט מוצפן.
    - טענת יציאה: טקסט מפוענח.
  - `get_dh_public_key()`
    - טענת כניסה: אין.
    - טענת יציאה: מפתח ציבורי של Diffie-Hellman.
  - `get_dh_shared_key(dh_1, pk_2, length)`
    - טענת כניסה: מפתחות Diffie-Hellman ומפתח ציבורי שני ( $pk_2$ ), אורך המפתח (length).
    - טענת יציאה: מפתח משותף של Diffie-Hellman.



## יואב זלכה: קנבס שיתופי

Rect

- **תפקיד:** ייצוג של מלבן בקנבס.
- **משתנים:**
  - `color`: צבע המלבן.
  - `width`: רוחב הקו של המלבן.
  - `x1`: הקואורדינטה האופקית של נקודת ההתחלה.
  - `x2`: הקואורדינטה האופקית של נקודת הסיום.
  - `y1`: הקואורדינטה האנכית של נקודת ההתחלה.
  - `y2`: הקואורדינטה האנכית של נקודת הסיום.
  - `id_in_canvas`: מזהה המלבן בקנבס ה-Tkinter.
- **פעולות:**
  - `init__(self, color, width, x1, y1, x2, y2, id_)`
  - טענת כניסה: צבע, רוחב, קואורדינטות, מזהה.
  - טענת יציאה: אתחול אובייקט Rect עם הפרטים שנמסרו.
  - `update(self, x2, y2)`
  - טענת כניסה: קואורדינטות חדשות.
  - טענת יציאה: עדכון קואורדינטות המלבן.
  - `draw(self, canvas)`
  - טענת כניסה: קנבס.
  - טענת יציאה: ציור המלבן בקנבס.
  - `delete_from_canvas(self, canvas)`
  - טענת כניסה: קנבס.
  - טענת יציאה: מחיקת המלבן מהקנבס.
  - `redraw(self, canvas)`

## יואב זלכה: קנבס שיתופי

- טענת כניסה: קנבס.
- טענת יציאה: ציור מחדש של המלבן בקנבס.

### Oval

- **תפקיד:** ייצוג של אליפסה בקנבס.
- **משתנים:**
  - צבע האליפסה.
  - width: רוחב הקו של האליפסה.
  - x1: הקואורדינטה האופקית של נקודת ההתחלה.
  - x2: הקואורדינטה האופקית של נקודת הסיום.
  - y1: הקואורדינטה האנכית של נקודת ההתחלה.
  - y2: הקואורדינטה האנכית של נקודת הסיום.
  - id\_in\_canvas: מזהה האליפסה בקנבס ה-Tkinter.
- **פעולות:**
  - כל הפעולות שב-Rect (ירושה)
  - draw(self, canvas)
  - טענת כניסה: קנבס.
  - טענת יציאה: ציור האליפסה בקנבס

### SocketHelper

- **תפקיד:** ניהול חיבורי TCP, שליחה וקבלה של נתונים מוצפנים בין לקוחות לשרת.
- **פעולות:**
  - send\_msg(socket, msg)
  - טענת כניסה: הסוקט אליו רוצים לשלוח, ההודעה שרוצים לשלוח.

## יואב זלכה: קנבס שיתופי

- טענת יציאה: שליחת ההודעה לסוקט, בצירוף אורך ההודעה בתחילת הפקטה.  
`recv_msg(socket)`
- טענת כניסה: הסוקט ממנו רוצים לקבל מידע.
- טענת יציאה: תוכן ההודעה שהתקבלה מהסוקט.
- `recvall(socket, n)` (פעולה כדי לאפשר את `recv_msg`)
- טענת כניסה: הסוקט ממנו מקבלים מידע, אורך ההודעה.
- טענת יציאה: המידע המתקבל מהסוקט.

### CommandServer

- **תפקיד:** ניהול וביצוע פקודות על השרת.
- **פעולות:**
  - `init__(self)___`
  - טענת כניסה: אין.
  - טענת יציאה: אתחול אובייקט `CommandServer`.
  - `handle_client(self, client_socket)`
  - טענת כניסה: סוקט הלקוח.
  - טענת יציאה: ניהול התקשורת עם הלקוח.
  - `get_and_inc_id(self, client_socket, project_path)`
  - טענת כניסה: סוקט הלקוח, נתיב הפרויקט.
  - טענת יציאה: קבלת המזהה ממאגר המידע והגדלתו, ושליחתו ללקוח.

### MainServer

## יואב זלכה: קנבס שיתופי

- **תפקיד:** ניהול השרת והתקשורת עם הלקוחות.
- **משתנים:**
  - `client_list`: רשימת הלקוחות המחוברים לשרת.
- **פעולות:**
  - `init__(self)___`
    - טענת כניסה: אין.
    - טענת יציאה: אתחול אובייקט `MainServer`.
  - `handle_client(self, client)`
    - טענת כניסה: אובייקט לקוח.
    - טענת יציאה: ניהול התקשורת עם הלקוח.
  - `new_line(self, client, d)`
    - טענת כניסה: אובייקט לקוח, ציור חדש.
    - טענת יציאה: הוספת שורה חדשה למסד הנתונים המכילה את המידע של הציור החדש.
  - `delete_line(self, client, id_)`
    - טענת כניסה: אובייקט לקוח, מזהה הציור.
    - טענת יציאה: מחיקת השורה עם המזהה הנתון ממסד הנתונים ושליחת עדכון ללקוחות המחוברים לפרויקט.
  - `create_new_db(self, client)`
    - טענת כניסה: אובייקט לקוח.
    - טענת יציאה: יצירת מסד נתונים חדש.
  - `load_canvas_sql(self, client)`
    - טענת כניסה: אובייקט לקוח.
    - טענת יציאה: טעינת הקנבס ממסד הנתונים ושליחת הנתונים ללקוח.

## יואב זלכה: קנבס שיתופי

- `get_project_names(self, client)`
- טענת כניסה: אובייקט לקוח.
- טענת יציאה: קבלת שמות הפרויקטים עבור הלקוח.
- `new_rect_oval(self, client, obj: Rect|Oval, type_)`
- טענת כניסה: אובייקט לקוח, אובייקט מלבן/אליפסה, סוג האובייקט.
- טענת יציאה: עדכון הנתונים במסד ועדכון שאר הלקוחות המחוברים לפרויקט.

## Client

- **תפקיד:** ייצוג לקוח בתקשורת עם השרת.
- **משתנים:**
  - `dir`: תיקיית עבודה של הלקוח.
  - `socket`: סוקט התקשורת של הלקוח.
  - `address`: כתובת הלקוח.
  - `shared_key`: מפתח משותף להצפנה.
  - `project`: שם הפרויקט הנוכחי שאליו מחובר הלקוח.
  - `cipher`: אובייקט הצפנה לטיפול בהצפנה ובפענוח של נתונים.
  - `path`: נתיב הפרויקט שאליו מחובר הלקוח.
- **פעולות:**
  - `init__(self, socket, address, shared_key, project)___`
  - טענת כניסה: סוקט, כתובת, מפתח משותף, שם הפרויקט.
  - טענת יציאה: אתחול אובייקט Client עם הפרטים שנמסרו.
  - `print_connection(self)`
  - טענת כניסה: אין.

## יואב זלכה: קנבס שיתופי

- טענת יציאה: הדפסת פרטי החיבור.
- `close(self)`
- טענת כניסה: אין.
- טענת יציאה: סגירת החיבור.
- `set_project(self, project_name)`
- טענת כניסה: שם הפרויקט.
- טענת יציאה: הגדרת שם הפרויקט הנוכחי.
- `encrypt(self, msg)`
- טענת כניסה: הודעה להצפנה.
- טענת יציאה: הודעה מוצפנת.
- `decrypt(self, msg)`
- טענת כניסה: הודעה מוצפנת.
- טענת יציאה: הודעה מפוענחת.

### 4.1.3 הסבר על הבעיות האלגוריתמיות המרכזיות

#### ציור חופשי

הסבר: ציור צורת גלולה (pill shape), הכוללת שני עיגולים בנקודת ההתחלה ובנקודת הסוף, וקו באמצע.

כשהמשתמש מתחיל לצייר במברשת, מצויר עיגול, וכשהוא מזיז את העכבר נוסף לציור pill shape בין המיקום הנוכחי של העכבר למיקומו בעדכון הקודם.

כשהציור מצויר מ-0 (כשהוא מעודכן אצל לקוחות אחרים המחוברים לפרויקט או כשטוענים את הפרויקט), נוצרים כל הקווים בין הנקודות והעיגולים בכל הנקודות.

#### קוד

`CanvasGUI.start_drawing`

```
def start_drawing(self, event):
```

## יואב זלכה: קנבס שיתופי

```
'''
    Initializes a new drawing and creates an oval shape at the
    starting position of the mouse cursor.
'''

width = int(self.line_width_entry.get())

self.deleted_drawings = [] # Clears the redo option

self.prev_x, self.prev_y = event.x, event.y
self.cur_drawing = Drawing(self.color, width)

circle_off = width/2 - 1
x1, y1 = (event.x - circle_off), (event.y - circle_off)
x2, y2 = (event.x + circle_off), (event.y + circle_off)
id_ = self.canvas.create_oval(
    x1, y1, x2, y2, fill=self.color, outline='')

self.cur_drawing.add_point((event.x, event.y))
self.cur_drawing.add_id(id_)
```

### CanvasGUI.draw

```
def draw(self, event):
    '''
        Creates a line between the previous and current position of the
        mouse cursor.
    '''

    # Deals with a bug where the target would update if the mouse is
    # pressed
    self.move_target(event)

    width = int(self.line_width_entry.get())

    x, y = event.x, event.y
    id_ = self.canvas.create_line(
        self.prev_x, self.prev_y, x, y, fill=self.color,
        width=width)
    self.cur_drawing.add_id(id_)
    self.prev_x, self.prev_y = x, y

    if (width > 3): # otherwise it looks weird
        # offset, got to be slightly lower than width/2 so it won't
        # buldge out
        circle_off = width/2 - 1
        x1, y1 = (event.x - circle_off), (event.y - circle_off)
        x2, y2 = (event.x + circle_off), (event.y + circle_off)
```

## יואב זלכה: קנבס שיתופי

```
id_ = self.canvas.create_oval(
    x1, y1, x2, y2, fill=self.color, outline="")
self.cur_drawing.add_id(id_)

self.cur_drawing.add_point((x, y))
```

### Drawing.draw

```
def draw(self, canvas):
    """
    draws the entire drawing
    """

    if len(self.pt_list) == 1: # Deals with the case of only one
point in the drawing
        self.draw_oval(canvas, self.pt_list[0])
        return

    # converts the list of tuples to a list of the format
[x,y,x,y,x,y,...]
    flattened = [a for x in self.pt_list for a in x]
    id_ = canvas.create_line(*flattened, width=self.width,
fill=self.color)

    self.id_list.append(id_)

    if (self.width > 3): # otherwise it looks weird
        self.draw_oval(canvas, self.pt_list[0])
        self.draw_oval(canvas, self.pt_list[-1])
```

## בקשת מידע ספציפי משרת, תוך כדי המתנה לקבלת ציורים מהשרת בכל רגע נתון

### הסבר:

יצירת שרת נוסף, "שרת פקודות"/"command server", שנועד להתנהג בצורה דומה ל-API – לנהל בקשות. כשהלקוח ירצה בקשה מהשרת, הוא ישלח לשרת הפקודות, והוא ידע למה ההודעה שהוא מקבל מהשרת מתייחסת, כי הוא מקבל אותה משרת נפרד.

### הקוד:

### CommandServer

```
class CommandServer():
```



```

def __init__(self):
    """Initializes the Server object and starts listening for
client connections."""
    # self.client_list = []

    server_socket = socket.socket()

    server_address = ("0.0.0.0", 1730)
    server_socket.bind(server_address)

    server_socket.listen(100)

    print(
        f'Command server started. Listening on
{server_address[0]}:{server_address[1]}')

    # Handle new connections
    while True:
        # Accept a client connection
        client_socket, client_address = server_socket.accept()

        print(
            f'Client connected to command server:
{client_address[0]}:{client_address[1]}')

        client_thread = threading.Thread(
            target=self.handle_client, args=(client_socket,))
        client_thread.start()

    def handle_client(self, client_socket):
        '''Handles a single client connection.'''
        data = SocketHelper.recv_msg(client_socket)
        if data is None:
            client_socket.close()
            return

        dir = Path(r'C:\Users\hp\Desktop\Freeform project\projects
(db)')
        project_name = data.decode()
        project_path = dir.joinpath(project_name).with_suffix('.db')

        while True:
            data = SocketHelper.recv_msg(client_socket)
            if data is None:
                break

            action, content = pickle.loads(data)

```

## יואב זלכה: קנבס שיתופי

```
if action == "get_and_inc_id":
    self.get_and_inc_id(client_socket, project_path)

client_socket.close()
```

### CanvasGUI.get\_and\_inc\_id

```
def get_and_inc_id(self):
    '''Gets the current ID from the server and increments it.'''
    SocketHelper.send_msg(self.command_client_socket,
                          pickle.dumps(("get_and_inc_id", None)))

    id_ =
int(SocketHelper.recv_msg(self.command_client_socket).decode())

    print("id from server:", id_)

    return id_
```

## 5. מדריך למשתמש

### 5.1 פירוט כלל קבצי המערכת

cipher.py – קובץ המכיל את המחלקה Cipher

client.py – קובץ המכיל את המחלקות SelectProjectGUI, NewProjectGUI, CanvasGUI ומריץ את SelectProjectGUI בעת הרצה.

constants.py – קובץ המכיל את הקבוע NONCE

drawing.py – קובץ המכיל את המחלקה Drawing

rect\_oval.py – קובץ המכיל את המחלקות Rect, Oval

server.py – קובץ המכיל את המחלקות Client, MainServer, CommandServer ומריץ את שתי האחרונות בעת הרצה.

socket\_helper.py – קובץ המכיל את המודול SocketHelper

```
└─ Freeform Project/
   ├── cipher.py
   ├── client.py
   ├── constants.py
   ├── drawing.py
   ├── rect_oval.py
   ├── server.py
   └── socket_helper.py
```

בנוסף, ישנה תיקייה בשרת למסד הנתונים (קבצי db). שניתן לקבוע על ידי שינוי ערך אחד בקוד לכל מקום במחשב השרת.

```
└─ project_database/
   ├── project1.db
   ├── project2.db
   └── project3.db
```

### 5.2 התקנת המערכת

הפרויקט מבוסס על פייתון 3.11+, משתמש במודולים socket, threadlib, threading, pickle, sqlite3, tkinter, DiffieHellman

ישנם שני נתונים התחלתיים: ה-NONCE, הנמצא בקובץ הפייתון constants.py, ומיקום קבצי מסד הנתונים, הניתן לשינוי בקוד השרת.

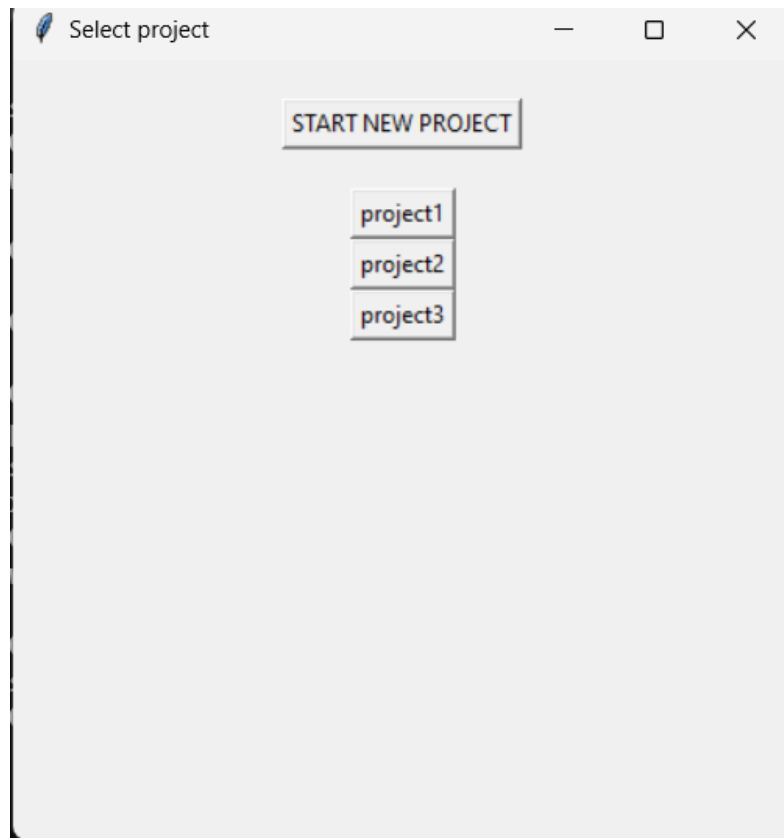
לצורך הפעלת הפרויקט חייבים חיבור ברשת משותפת בין המחשב לשרת.

### 5.3 משתמשי המערכת

מנהל המערכת – הדבר היחיד שעל מנהל המערכת לעשות הוא להפעיל את השירות. מנהל המערכת יוכל לראות (ללא GUI) עדכונים כגון התחברות מחשבים חדשים אל השרת.

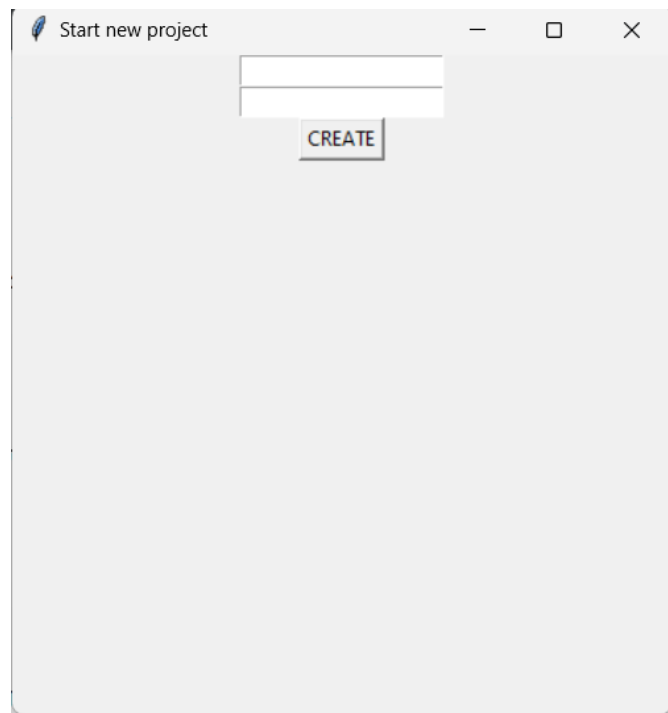
משתמש קצה – משתמש הקצה מפעיל את התוכנה, ולאחר מכן עובר בין מסכים כפי שפורט בתיאור מסכי הפרויקט (3.6).

#### בחירת פרויקט

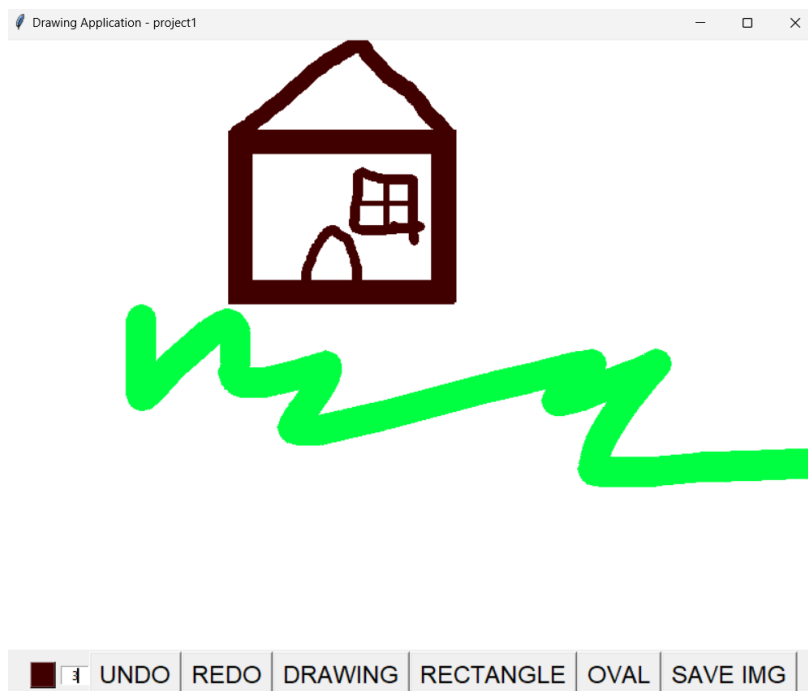


## יואב זלכה: קנבס שיתופי

### יצירת פרויקט



### קנבס ציור



## 5.4 פירוט הבדיקות

- יצירת פרויקט חדש
  - מטרת הבדיקה: בדיקת היכולת ליצור פרויקטים חדשים.
  - מה בוצע בפועל: הפעלת המערכת ולחיצה על כפתור יצירת הפרויקט החדש. בדיקה אם הקנבס נפתח לאחר מכן ואם אפשר לצאת ולפתוח את הפרויקט מחדש.
  - תוצאות הבדיקה: הצלחה.
- כניסה לפרויקט ממחשב אחר
  - מטרת הבדיקה: לבדוק אם אכן אפשר לפתוח את אותו פרויקט ממספר מחשבים שונים.
  - מה בוצע בפועל: לאחר יצירת הפרויקט, הפעלת המערכת ממחשב נוסף ובדיקה אם הוא עובד.
  - תוצאות הבדיקה: הצלחה.
- כניסה רק עם הסיסמה הנכונה
  - מטרת הבדיקה: לבדוק אם הפרויקט מאובטח מבחינת סיסמאות.
  - מה בוצע בפועל: לאחר יצירת הפרויקט, ניסיון להיכנס אליו עם סיסמה לא נכונה, ובדיקה שזה לא אפשרי.
  - תוצאות: הצלחה.
- ציור משני מחשבים שונים במקביל (על מנת לראות את הסנכרון ביניהם)
  - מטרת הבדיקה: לאחר שפתחנו פרויקט משני מחשבים שונים, אנחנו רוצים לראות שהם באמת מתעדכנים בשינויים אחד של השני.
  - מה בוצע בפועל: ציור מברשת/מלבן/אליפסה, בעוביים שונים ובצבעים שונים, ובדיקה שהמחשב השני תמיד מעודכן. בדיקה דו כיוונית.
  - תוצאות: הצלחה.

## 6. סיכום אישי / רפלקציה

בפרויקט הנוכחי עבדתי על מערכת מורכבת לניהול קנבס ציור מרוחק, שכללה מגוון רכיבים כמו ממשקים גרפיים, ניהול תקשורת מוצפנת בין לקוחות לשרת, ואינטגרציה עם מסדי נתונים. בעבודה על הפרויקט העמקתי את הידע והיכולות שלי בתחומים שונים של פיתוח תוכנה, כולל תכנות ממשקים גרפיים, ניהול תקשורת רשת, הצפנה ופענוח נתונים, וניהול מסדי נתונים.

## יואב זלכה: קנבס שיתופי

### אתגרים

אחד האתגרים המרכזיים היה לתכנן וליישם את התקשורת בין הלקוחות לשרת בצורה יעילה ומאובטחת. הבנת העקרונות של הצפנה ופענוח נתונים, ובמיוחד עבודה עם מפתחות הצפנה ושיטות כמו AES ו-Diffie-Hellman, היוו חלק חשוב ומאתגר בתהליך. בנוסף, ההחלטה על שיטת התקשורת (UDP/TCP) שיותר מתאימה לפרויקט דרשה מחקר.

הופתעתי מכך שאתגר הסנכרון בין המשתמשים, שהוא האתגר שחשבתי שיהיה העיקרי בעבודה על הפרויקט, לא היווה אתגר משמעותי.

### תהליך הלמידה

העבודה על הפרויקט לימדה אותי שחשוב לחקור נושא לפני שמתחילים לעסוק בו. כשעבדתי בלי תכנון מספק, נתקלתי בבעיות בהמשך העבודה, אך כשחקרתי לפני הבנתי איך להשיג את המטרות שלי בצורה היעילה ביותר.

### תובנות

זו הפעם הראשונה שעבדתי על פרויקט בסדר גודל כזה, ולכן באו עם העבודה מסקנות רבות. ראשית, אחד הכלים שעזרו לי משמעותית הוא git; האפשרות לראות את כל השינויים שעשיתי ולשחזר גרסאות ישנות עזרה מאוד באיתור ותיקון תקלות בקוד. שנית, כשעובדים על פרויקט לאורך זמן רואים את החשיבות בתייעוד הפרויקט. בזכות תיעוד מקיף של הפרויקט יכולתי לחזור אליו גם אחרי הפסקה של מספר שבועות ולהבין אותו כראוי.

### ראייה לאחור

אני יחסית מרוצה מאיך שיישמתי את הפרויקט, אך ישנם מספר דברים שהייתי משנה. לדוגמה: ככל שהוספתי יותר יכולות לתקשורת בין הלקוח לשרת, נדרשתי להעביר יותר פרמטרים בין ה-GUIs השונים. במקום לעשות זאת כך, היה חכם יותר ליצור מחלקה של כל המידע שמועבר בין ה-GUIs, ולהעביר את המידע בפרמטר אחד.

## 7. ביבליוגרפיה

- Stack Overflow. (n.d.). Restricting the value in Tkinter Entry widget. Stack Overflow. Retrieved June 5, 2024, from <https://stackoverflow.com/questions/8959815/restricting-the-value-in-tkinter-entry-widget>
- Stack Overflow. (n.d.). Python tkinter how to create line with multiple points in a list/tuple. Stack Overflow. Retrieved June 5, 2024, from <https://stackoverflow.com/questions/51118695/python-tkinter-how-to-create-line-with-multiple-points-in-a-list-tuple>
- Programiz. (n.d.). Python eval(). Programiz. Retrieved June 5, 2024, from <https://www.programiz.com/python-programming/methods/built-in/eval>
- Stack Overflow. (n.d.). Why is the variable always set to the last item in the list. Stack Overflow. Retrieved June 5, 2024, from <https://stackoverflow.com/questions/69147762/why-is-the-variable-always-set-to-the-last-item-in-the-list>
- QuizDeveloper. (n.d.). TypeError: send() argument after must be an iterable, not socket in Python. QuizDeveloper. Retrieved June 5, 2024, from <https://quizdeveloper.com/faq/typeerror-send-argument-after-must-be-an-iterable-not-socket-in-python-aid2325>
- Stack Overflow. (n.d.). Python socket receive large amount of data. Stack Overflow. Retrieved June 5, 2024, from <https://stackoverflow.com/questions/17667903/python-socket-receive-large-amount-of-data>
- Tech with Tim. (2019, March 18). Python Multithreading Tutorial #3 - Synchronizing & Locking Threads| Network programming in Python | Python Socket Programming YouTube. <https://youtu.be/SDAkQq17S2Q?>



## 8. קוד הפרויקט

### 8.1 client.py

```
from drawing import Drawing
from rect_oval import Rect, Oval
from socket_helper import SocketHelper
import tkinter as tk
from tkinter import colorchooser
# import sqlite3
# import os
import socket
import threading
import pickle
from PIL import Image, ImageDraw
# from pathlib import Path
# import queue
# import time
# import struct
from tkinter import filedialog
from cipher import Cipher
from constants import NONCE
from hashlib import sha256

class CanvasGUI:
    def __init__(self, client_socket, command_client_socket, file_name,
shared_key, pwd=None, exists=False):
        # self.dir = r'C:\Users\hp\Desktop\Freeform project\projects
(db)\\'
        # self.file_name = file_name
        # self.full_path = self.dir + self.file_name

        self.cipher = Cipher(shared_key, NONCE)

        self.project_name = file_name

        self.client_socket = client_socket
        self.command_client_socket = command_client_socket

        self.drawings = []
        self.deleted_drawings = []

        self.mode = "drawing"
        self.color = "#000000"
        self.color_changed = False
        self.target = -1 # setup for target
        self.show_target = True
```

## יואב זלכה: קנבס שיתופי

```
self.root = tk.Tk()
self.root.title("Drawing Application - " + file_name)

self.canvas = tk.Canvas(self.root, width=800, height=600,
bg="white")
self.canvas.pack()

self.canvas.bind("<ButtonPress-1>", self.mouse_pressed)
self.canvas.bind("<B1-Motion>", self.mouse_pressed_moved)
self.canvas.bind(
    "<ButtonRelease-1>", lambda _:
threading.Thread(target=self.end_drawing).start())

self.canvas.bind("<Motion>", self.move_target)
self.canvas.bind("<Leave>", self.hide_target)

self.root.bind("<Control-z>", self.delete_last_drawing)
self.root.bind("<Control-y>", self.redo_last_deleted_drawing)

self.colors_frame = tk.Frame(self.root)
self.colors_frame.pack()

font = ("Arial", 18)

self.pick_color_b = tk.Button(
    self.colors_frame, text="", command=self.pick_color,
background=self.color, padx=8)
self.pick_color_b.grid(row=0, column=0, padx=5)

def set_mode(mode):
    self.mode = mode

    if mode == "rect" or mode == "oval":
        self.show_target = False
        self.canvas.config(cursor="tcross")
    elif mode == "drawing":
        self.show_target = True
        self.canvas.config(cursor="none")

tk.Button(self.colors_frame, text="DRAWING", font=font,
command=lambda: set_mode("drawing")).grid(row=0,
column=4)
tk.Button(self.colors_frame, text="RECTANGLE", font=font,
command=lambda: set_mode("rect")).grid(row=0,
column=5)
tk.Button(self.colors_frame, text="OVAL", font=font,
command=lambda: set_mode("oval")).grid(row=0,
column=6)
```

```

        vcmd = (self.root.register(self.width_entry_validation),
                "%P") # used to deal with validation in Tcl
        self.line_width_entry = tk.Entry(self.colors_frame,
        validate="all", validatecommand=vcmd,
                                width=4, justify="center") #
    %P -> new value of entry box
        self.line_width_entry.grid(row=0, column=1)
        self.line_width_entry.insert(0, "10") # Sets starting width

        tk.Button(self.colors_frame, text="UNDO", font=font,
        command=self.delete_last_drawing).grid(row=0,
column=2) # ~
        tk.Button(self.colors_frame, text="REDO", font=font,
        command=self.redo_last_deleted_drawing).grid(row=0,
column=3) # ~

        tk.Button(self.colors_frame, text="SAVE IMG", font=font,
        command=self.save_img).grid(row=0, column=8)

        self.canvas.config(cursor="none")

        self.window_open = True # Used to cut the connection when the
user closes the window

        self.receive_lock = threading.Lock()

        if (exists):
            self.load_canvas()
        else:
            self.create_new_db(pwd)

        receive_thread = threading.Thread(
            target=self.receive_data, args=(self.client_socket,))
        receive_thread.start()

        self.root.mainloop()

        self.window_open = False # Used to cut the connection when the
user closes the window
        print("windows closed")

        receive_thread.join()

    def start_drawing(self, event):
        '''
            Initializes a new drawing and creates an oval shape at the
starting position of the mouse cursor.

```

```

'''

width = int(self.line_width_entry.get())

self.deleted_drawings = [] # Clears the redo option

self.prev_x, self.prev_y = event.x, event.y
self.cur_drawing = Drawing(self.color, width)

circle_off = width/2 - 1
x1, y1 = (event.x - circle_off), (event.y - circle_off)
x2, y2 = (event.x + circle_off), (event.y + circle_off)
id_ = self.canvas.create_oval(
    x1, y1, x2, y2, fill=self.color, outline="")

self.cur_drawing.add_point((event.x, event.y))
self.cur_drawing.add_id(id_)

def draw(self, event):
    '''
    Creates a line between the previous and current position of the
mouse cursor.
    '''

    # Deals with a bug where the target would update if the mouse i
pressed
    self.move_target(event)

    width = int(self.line_width_entry.get())

    x, y = event.x, event.y
    id_ = self.canvas.create_line(
        self.prev_x, self.prev_y, x, y, fill=self.color,
width=width)
    self.cur_drawing.add_id(id_)
    self.prev_x, self.prev_y = x, y

    if (width > 3): # otherwise it looks weird
        # offset, got to be slightly lower than width/2 so it wont
buldge out
        circle_off = width/2 - 1
        x1, y1 = (event.x - circle_off), (event.y - circle_off)
        x2, y2 = (event.x + circle_off), (event.y + circle_off)
        id_ = self.canvas.create_oval(
            x1, y1, x2, y2, fill=self.color, outline="")
        self.cur_drawing.add_id(id_)

    self.cur_drawing.add_point((x, y))

```

```

def end_drawing(self, *_): # *_ to deal with event
    """
        Assigns an ID to the current drawing, adds it to the list
of drawings and sends it to the server.
    """
    cur_id = self.get_and_inc_id()
    self.cur_drawing.id = cur_id

    self.drawings.append(self.cur_drawing)
    self.send_to_db(self.cur_drawing)

def hide_target(self, *_): # *_ to deal with event
    """
        Hides the target shape indicating the area to be painted.
(used when the mouse leaves the canvas)
    """
    self.canvas.delete(self.target)
    self.target = -1

def move_target(self, event):
    """
        Updates the position of the target shape to follow the
mouse cursor.
    """
    if (not self.show_target):
        return

    if (self.target != -1):
        self.canvas.delete(self.target)
        circle_off = max(5/2, int(self.line_width_entry.get())/2)
        x1, y1 = (event.x - circle_off), (event.y - circle_off)
        x2, y2 = (event.x + circle_off), (event.y + circle_off)
        # self.target = self.canvas.create_oval(x1, y1, x2, y2,
fill="", outline=self.color, dash=(2,1)) # shows the outline of the
area you'd paint
        # shows the outline of the area you'd paint
        self.target = self.canvas.create_oval(
            x1, y1, x2, y2, fill="", outline=self.color)

def set_color(self, color):
    """
        Sets the color variable to the selected color.
    """
    self.color = color

def delete_last_drawing(self, *_): # *_ to deal with event if
given

```

## יואב זלכה: קנבס שיתופי

```
'''
    Removes the last drawing from the list of drawings, deletes
it from the canvas, and sends a delete msg to the server.
'''
if self.drawings:
    d = self.drawings.pop()
    d.delete_from_canvas(self.canvas)
    self.deleted_drawings.append(d)

    # get id to delete
    id_ = d.id

    print("id", id_)

    SocketHelper.send_msg(self.client_socket,
                          pickle.dumps(("delete", id_)))

def redo_last_deleted_drawing(self, *_): # *_ to deal with event
if given
'''
    Restores the last deleted drawing from the list of deleted
drawings and redraws it on the canvas.
'''
if self.deleted_drawings:
    d = self.deleted_drawings.pop()
    self.drawings.append(d)
    d.draw(self.canvas)

    self.send_to_db(d)

    # self.save_row(d)

def width_entry_validation(self, value):
    return value == "" or (value.isnumeric() and int(value) <= 45)

def load_canvas(self):
'''
    Loads drawings from the database and adds them to the
canvas.
'''
    SocketHelper.send_msg(self.client_socket,
                          pickle.dumps(("load_canvas", None)))

with self.receive_lock:
    data = self.cipher.aes_decrypt(
        SocketHelper.recv_msg(self.client_socket))

    drawings = pickle.loads(data)
```

```

    for d in drawings:
        d.draw(self.canvas)
        self.drawings.append(d)

def create_new_db(self, pwd):
    """
    Creates a new database for the canvas.
    """
    SocketHelper.send_msg(self.client_socket, pickle.dumps(
        ("create_new_db", sha256(pwd.encode()).hexdigest())))

def receive_data(self, client_socket):
    '''Receives and processes data from the server.'''
    while self.window_open:
        try:
            # Set a timeout for the socket operation so it will
            know if the window is closed
            client_socket.settimeout(0.5) # in seconds

            with self.receive_lock:
                data = SocketHelper.recv_msg(client_socket)

            action, content = pickle.loads(data)
            print(f'action: {action}, content: {content}')

            if action == "new_line":
                self.add_to_drawings(content)
            elif action == "new_rect" or action == "new_oval":
                self.add_to_drawings(content)
            elif action == "delete":
                self.delete_line(content)

        except socket.timeout:
            # Timeout occurred, check the window state
            if not self.window_open:
                break
            continue

def send_to_db(self, d): # d - Drawing/Rect/Oval object
    '''Sends a new drawing to the server.'''
    # Define the data to be sent
    mode = ""
    if isinstance(d, Drawing):
        mode = "new_line"
    elif isinstance(d, Oval):
        mode = "new_oval"
    elif isinstance(d, Rect):
        mode = "new_rect"

```

```

        # print(data)

        data = pickle.dumps((mode,
self.cipher.aes_encrypt(pickle.dumps(d))))
        SocketHelper.send_msg(self.client_socket, data)

    def add_to_drawings(self, drawing):
        '''Adds a new drawing to the list of drawings and draws it on
the canvas.'''
        self.drawings.append(drawing)
        self.root.after(0, lambda: drawing.draw(self.canvas))

    def delete_line(self, id_, is_this_user=False):
        '''Removes a drawing from the list of drawings and deletes
it.'''
        for d in self.drawings:
            if d.id == id_:
                self.drawings.remove(d)
                self.root.after(0, d.delete_from_canvas(self.canvas))

                if is_this_user: # Only append drawing if it was
deleted by this user
                    self.deleted_drawings.append(d)

                break

    def get_and_inc_id(self):
        '''Gets the current ID from the server and increments it.'''
        SocketHelper.send_msg(self.command_client_socket,
                                pickle.dumps(("get_and_inc_id", None)))

        id_ =
int(SocketHelper.recv_msg(self.command_client_socket).decode())

        print("id from server:", id_)

        return id_

    def pick_color(self):
        # The first time the color picker is opened, the color picker
will start with #50dca4, otherwise it will default to the last picked
color
        if not self.color_changed:
            color = colorchooser.askcolor(
                title="Select Color", initialcolor="#50dca4")
            self.color_changed = True
        else:
            color = colorchooser.askcolor(
                title="Select Color", initialcolor=self.color)

```



```

        if color[1]: # Check if a color was chosen
            chosen_color = color[1]
            self.color = chosen_color
            self.pick_color_b.config(background=chosen_color)

    def start_rect(self, event):
        width = int(self.line_width_entry.get())
        self.cur_drawing = Rect(self.color, width, event.x, event.y)
        self.cur_drawing.draw(self.canvas)

    def start_oval(self, event):
        width = int(self.line_width_entry.get())
        self.cur_drawing = Oval(self.color, width, event.x, event.y)
        self.cur_drawing.draw(self.canvas)

    def update_rect_oval(self, event):
        self.cur_drawing.update(event.x, event.y)
        self.cur_drawing.redraw(self.canvas)

    def mouse_pressed(self, event):
        if self.mode == "drawing":
            self.start_drawing(event)
        elif self.mode == "rect":
            self.start_rect(event)
        elif self.mode == "oval":
            self.start_oval(event)

    def mouse_pressed_moved(self, event):
        if self.mode == "drawing":
            self.draw(event)
        elif self.mode == "rect" or self.mode == "oval":
            self.update_rect_oval(event)

    def save_img(self):
        # Select file location and name
        file_path = filedialog.asksaveasfilename(initialdir="/",
        title="Select a file", filetypes=(
            ("PNG", "*.png"),), defaultextension=".png",
        initialfile=self.project_name+".png")
        if file_path == "":
            return

        # Create a new image
        image = Image.new("RGB", (800, 600), (255, 255, 255))

        # Create an ImageDraw object
        draw = ImageDraw.Draw(image)

```

```

for drawing in self.drawings:
    if isinstance(drawing, Oval):
        x1 = min(drawing.x1, drawing.x2)
        x2 = max(drawing.x1, drawing.x2)
        y1 = min(drawing.y1, drawing.y2)
        y2 = max(drawing.y1, drawing.y2)
        draw.ellipse((x1, y1, x2, y2), width=drawing.width,
                      outline=drawing.color)
    elif isinstance(drawing, Rect):
        x1 = min(drawing.x1, drawing.x2)
        x2 = max(drawing.x1, drawing.x2)
        y1 = min(drawing.y1, drawing.y2)
        y2 = max(drawing.y1, drawing.y2)
        draw.rectangle((x1, y1, x2, y2),
                       width=drawing.width,
outline=drawing.color)
    elif isinstance(drawing, Drawing):
        # draw.line(drawing.pt_list, fill=drawing.color,
width=drawing.width)
        drawing.draw_PIL(draw)

# Save the image to the computer
image.save(file_path)

class SelectProjectGUI:
    def __init__(self):
        self.init_connection_to_server()

        # self.dir = r'C:\Users\hp\Desktop\Freeform project\projects
(db)\\'
        SocketHelper.send_msg(self.client_socket, pickle.dumps(
            ("get_projects_names", None)))
        # files = pickle.loads(self.client_socket.recv(2048))
        self.db_files =
pickle.loads(SocketHelper.recv_msg(self.client_socket))

        self.root = tk.Tk()
        self.root.geometry("400x400")
        self.root.title("Select project")

        tk.Button(text="START NEW PROJECT",
                  command=self.new_project).pack(pady=20)

        for file in self.db_files:
            tk.Button(
                text=file, command=lambda name=file:
self.open_project(name)).pack()

```

```

        self.root.mainloop()

    def open_project(self, name):
        self.root.destroy()

        # file = self.dir + file_name
        SocketHelper.send_msg(self.client_socket,
                               pickle.dumps(("set_project_name", name)))
        SocketHelper.send_msg(self.command_client_socket,
                               name.encode())
        print(name)

        CanvasGUI(self.client_socket, self.command_client_socket,
                   name, self.shared_key, exists=True)

    def new_project(self):
        self.root.destroy()

        NewProjectGUI(self.client_socket, self.command_client_socket,
                      self.db_files, self.shared_key)

    def init_connection_to_server(self):
        hostname = 'localhost'

        self.client_socket = socket.socket()
        server_address = (hostname, 1729)
        self.client_socket.connect(server_address)

        self.command_client_socket = socket.socket()
        command_server_address = (hostname, 1730)
        self.command_client_socket.connect(command_server_address)

        # generate shared key
        dh, pk = Cipher.get_dh_public_key()
        self.client_socket.send(pk)
        reply = self.client_socket.recv(1024)
        self.shared_key = Cipher.get_dh_shared_key(dh, reply)

        print("shared key:", self.shared_key)

        print(f'Connected to server {server_address[0]}:{server_address[1]}')

class NewProjectGUI:
    def __init__(self, client_socket, command_client_socket, db_files,
                 shared_key):
        self.client_socket = client_socket
        self.command_client_socket = command_client_socket

```

```

self.db_files = db_files
self.shared_key = shared_key

# self.dir = r'C:\Users\hp\Desktop\Freeform project\projects
(db)\\'
self.root = tk.Tk()
self.root.geometry("400x400")
self.root.title("Start new project")
tk.Label(self.root, text="PICK A NAME FOR YOUR PROJECT")
self.name_entry = tk.Entry(self.root)
self.name_entry.pack()
self.pwd_entry = tk.Entry(self.root)
self.pwd_entry.pack()
tk.Button(text="CREATE", command=self.button_pressed).pack()

def button_pressed(self):
    '''Opens an existing project by setting the project name and
creating a new Canvas_GUI.'''
    name = self.name_entry.get()
    pwd = self.pwd_entry.get()
    if name not in self.db_files:
        self.root.destroy()
        SocketHelper.send_msg(self.client_socket,
                                pickle.dumps(("set_project_name",
name)))
        SocketHelper.send_msg(self.command_client_socket,
name.encode())
        CanvasGUI(self.client_socket, self.command_client_socket,
                    name, self.shared_key, pwd, False)
    else:
        tk.Label(text="NAME ALREADY TAKEN, PLEASE TRY
AGAIN").pack()

# class Client:
#     def __init__(self):
#         self.socket = socket.socket()

#         server_address = ('localhost', 1730)
#         self.socket.connect(server_address)

#         print(f'Connected to server
{server_address[0]}:{server_address[1]}')

#         receive_thread = threading.Thread(target=self.receive_data,
args=(client_socket,))
#         receive_thread.start()

#     def recieve_data(self):

```

```
#         while True:
#             # Receive data from the server
#             data = self.socket.recv(1024).decode()
#             if data == "exit":
#                 break

#             # Print the received data
#             print('Received from server:', data)

#         # Close the client socket
#         self.socket.close()

if __name__ == "__main__":
    SelectProjectGUI()

# gui = Canvas_GUI("drawings.db", True)
```

## 8.2 server.py

```
# Notes on some problems:
# client checks if new project name is valid based on the list it got
before
# client receiving from server with threading lock can be problematic

from drawing import Drawing
from rect_oval import Rect, Oval
from socket_helper import SocketHelper
import socket
from pathlib import Path
import threading
import pickle
import sqlite3
# import os
# import time
from cipher import Cipher
from constants import NONCE

dir = Path(r'C:\Users\hp\Desktop\Freeform project\projects (db)')
class Client:
    def __init__(self, socket, address, shared_key, project=None):
        self.dir = dir

        self.socket = socket
        self.address = address
        self.shared_key = shared_key
        self.project = project
```

```

        self.cipher = Cipher(self.shared_key, NONCE)

    def print_connection(self):
        """Prints the client connection details."""
        print(
            f'Client connected to main server: {self.address[0]}:{self.address[1]}'
        )

    def close(self):
        '''Closes the client connection.'''
        socket.close()

    def set_project(self, project_name):
        '''Assigns a project to the client'''
        self.project = project_name
        self.path = self.dir.joinpath(self.project).with_suffix('.db')

    def encrypt(self, msg):
        return self.cipher.aes_encrypt(msg)

    def decrypt(self, msg):
        return self.cipher.aes_decrypt(msg)

# Example usage:
# client = Client(socket, address)
# project_name = "Project1"
# set_project(client, project_name)

class MainServer:
    def __init__(self):
        """Initializes the Server object and starts listening for
        client connections."""
        self.client_list = []

        server_socket = socket.socket()

        server_address = ("0.0.0.0", 1729)
        server_socket.bind(server_address)

        server_socket.listen(100)

        print(
            f'Main server started. Listening on {server_address[0]}:{server_address[1]}'
        )

        # Handle new connections

```

```

while True:
    # Accept a client connection
    client_socket, client_address = server_socket.accept()

    # generate shared key
    dh, pk = Cipher.get_dh_public_key()
    client_pk = client_socket.recv(1024)
    shared_key = Cipher.get_dh_shared_key(dh, client_pk)
    print("shared key:", shared_key)
    client_socket.send(pk)

    new_client = Client(client_socket, client_address,
shared_key)
    self.client_list.append(new_client)
    new_client.print_connection()

    # Create a new thread to handle the client
    client_thread = threading.Thread(
        target=self.handle_client, args=(new_client,))
    client_thread.start()

def handle_client(self, client: Client):
    '''Handles a single client connection.'''
    while True:
        # try:
        # data = b'' + client.socket.recv(1024)

        data = SocketHelper.recv_msg(client.socket)

        if data is None:
            break

        action, content = pickle.loads(data)

        print(
            f'Client {client.address[0]}:{client.address[1]}:
action: {action}, content: {content}')

        if action == "set_project_name":
            client.set_project(content)
        elif action == "get_projects_names":
            self.get_projects_names(client)
        elif action == "new_line":
            self.new_line(client,
pickle.loads(client.decrypt(content)))
        elif action == "new_rect":
            self.new_rect_oval(client, pickle.loads(
                client.decrypt(content)), 'rect')

```

```

elif action == "new_oval":
    self.new_rect_oval(client, pickle.loads(
        client.decrypt(content)), 'oval')
elif action == "delete":
    self.delete_line(client, content) # client, id
# elif action == "get_and_inc_id":
#     self.get_and_inc_id(client)
elif action == "create_new_db":
    self.create_new_db(client)
elif action == "load_canvas":
    self.load_canvas_sql(client)

    # elif action == "get_id":
    #     self.send_id(client)
    # elif action == "inc_id":
    #     self.inc_id(client)

# except ConnectionResetError:
#     break

# except Exception as e:
#     # ... PRINT THE ERROR MESSAGE ... #
#     print(e)
#     # print("couldnt get data from client")
#     break

# Close the client socket
print(f"Client disconnected:
{client.address[0]}:{client.address[1]}")
client.socket.close()
self.client_list.remove(client)

# print("clients:")
# for c in self.client_list:
#     print(c.address)

def new_line(self, client, d): # d - Drawing object
    """Updates the database with a new drawing and sends it to
other clients."""
    # Update DB
    conn = sqlite3.connect(client.path)
    c = conn.cursor()
    c.execute('INSERT INTO drawings VALUES (?, ?, ?, ?)',
              (d.id, d.color, d.width, str(d.pt_list)))

    conn.commit()
    conn.close()

```



```

        # Send to other clients
        for c in self.client_list:
            print(c.project)
            if (c != client and c.project == client.project):
                SocketHelper.send_msg(c.socket,
pickle.dumps(("new_line", d)))

    def delete_line(self, client, id_):
        """Deletes a drawing from the database and sends a delete
message to other clients."""
        # Delete from db
        conn = sqlite3.connect(client.path)
        c = conn.cursor()

        # Delete from either table
        c.execute('DELETE FROM drawings WHERE id = ?', (id_,))
        c.execute('DELETE FROM rects WHERE id = ?', (id_,))
        c.execute('DELETE FROM ovals WHERE id = ?', (id_,))

        conn.commit()
        conn.close()

        for c in self.client_list:
            if (c != client and c.project == client.project):
                SocketHelper.send_msg(c.socket, pickle.dumps(("delete",
id_)))

    def create_new_db(self, client):
        '''Initializes a new database'''
        conn = sqlite3.connect(client.path)
        c = conn.cursor()

        c.execute(
            'CREATE TABLE drawings (id INTEGER PRIMARY KEY, color
TEXT, width INTEGER, pt_list TEXT)')
        c.execute('CREATE TABLE rects (id INTEGER PRIMARY KEY, color
TEXT, width INTEGER, x1 INTEGER, y1 INTEGER, x2 INTEGER, y2 INTEGER)')
        c.execute('CREATE TABLE ovals (id INTEGER PRIMARY KEY, color
TEXT, width INTEGER, x1 INTEGER, y1 INTEGER, x2 INTEGER, y2 INTEGER)')
        c.execute('CREATE TABLE int_variables (name TEXT, value
INTEGER)')
        c.execute('INSERT INTO int_variables VALUES (?, ?)', ("id", 0))

        conn.commit()
        conn.close()

    def load_canvas_sql(self, client: Client):

```

```

    """Gets the current ID from the database, sends it to the
    client, and increments the ID."""
    print(f"path:{client.path}, project:{client.project}")
    conn = sqlite3.connect(client.path)
    c = conn.cursor()

    c.execute('SELECT * FROM drawings')

    drawings = []

    for row in c.fetchall():
        id = row[0]
        color = row[1]
        width = row[2]
        pt_list = row[3]

        d = Drawing(color, width, eval(pt_list), id)
        drawings.append(d)

    c.execute('SELECT * FROM rects')

    for row in c.fetchall():
        id = row[0]
        color = row[1]
        width = row[2]
        x1 = row[3]
        y1 = row[4]
        x2 = row[5]
        y2 = row[6]

        r = Rect(color, width, x1, y1, x2, y2, id)
        drawings.append(r)

    c.execute('SELECT * FROM ovals')
    for row in c.fetchall():
        id = row[0]
        color = row[1]
        width = row[2]
        x1 = row[3]
        y1 = row[4]
        x2 = row[5]
        y2 = row[6]

        r = Oval(color, width, x1, y1, x2, y2, id)
        drawings.append(r)

    drawings.sort(key=lambda d: d.id)
    SocketHelper.send_msg(

```

```

        client.socket, client.encrypt(pickle.dumps(drawings)))

    conn.close()

    # def is_db_file(self, file):
    #     return file.suffix == ".db" #checks the last 3 characters in
the string

    def get_projects_names(self, client):
        """Gets a list of project names from the directory and sends it
to the client."""

        files = Path.iterdir(client.dir)
        # db_files = list(filter(self.is_db_file, files))
        # Filter the db files
        db_files = [file.stem for file in files if file.suffix ==
".db"]
        print(db_files)

        SocketHelper.send_msg(client.socket, pickle.dumps(db_files))

    def new_rect_oval(self, client, obj: Rect | Oval, type_):
        """Updates the database with a new drawing and sends it to
other clients."""
        # Update DB
        conn = sqlite3.connect(client.path)
        c = conn.cursor()

        c.execute(f'INSERT INTO {type_}s VALUES (?, ?, ?, ?, ?, ?, ?)',
            (obj.id, obj.color, obj.width, obj.x1, obj.y1,
obj.x2, obj.y2))

        conn.commit()
        conn.close()

        # Send to other clients
        for c in self.client_list:
            if (c != client and c.project == client.project):
                SocketHelper.send_msg(
                    c.socket, pickle.dumps((f"new_{type_}", obj)))

class CommandServer():
    def __init__(self):
        """Initializes the Server object and starts listening for
client connections."""
        # self.client_list = []

        server_socket = socket.socket()

```

```

server_address = ("0.0.0.0", 1730)
server_socket.bind(server_address)

server_socket.listen(100)

print(
    f'Command server started. Listening on {server_address[0]}:{server_address[1]}'
)

# Handle new connections
while True:
    # Accept a client connection
    client_socket, client_address = server_socket.accept()

    print(
        f'Client connected to command server: {client_address[0]}:{client_address[1]}'
    )

    client_thread = threading.Thread(
        target=self.handle_client, args=(client_socket,)
    )
    client_thread.start()

def handle_client(self, client_socket):
    '''Handles a single client connection.'''
    data = SocketHelper.recv_msg(client_socket)
    if data is None:
        client_socket.close()
        return

    project_name = data.decode()
    project_path = dir.joinpath(project_name).with_suffix('.db')

    while True:
        data = SocketHelper.recv_msg(client_socket)
        if data is None:
            break

        action, content = pickle.loads(data)

        if action == "get_and_inc_id":
            self.get_and_inc_id(client_socket, project_path)

    client_socket.close()
    # self.client_list.remove(client_socket)

def get_and_inc_id(self, client_socket, project_path):

```

```
        """Gets the current ID from the database, sends it to the
client, and increments the ID."""
        print(project_path)
        conn = sqlite3.connect(project_path)
        c = conn.cursor()

        c.execute('SELECT value FROM int_variables WHERE name = "id"')

        id_ = c.fetchone()[0]

        # print(id_)

        SocketHelper.send_msg(client_socket, str(id_).encode())

        inc_cmd = '''
            UPDATE int_variables
            SET value = ?
            WHERE name = "id"'''

        c.execute(inc_cmd, (id_+1,))
        conn.commit()
        conn.close()

if __name__ == "__main__":
    # Start the server
    t1 = threading.Thread(target=lambda: MainServer())
    t1.start()
    threading.Thread(target=lambda: CommandServer()).start()
    t1.join()
```

## 8.3 cipher.py

```
import random
# pip install pycryptodome
from Crypto.Cipher import AES
# pip install py-diffie-hellman
from diffiehellman import DiffieHellman

class Cipher:
    def __init__(self, key, nonce):
        self.key = key
        self.nonce = nonce

    def aes_encrypt(self, txt):
        cipher = AES.new(self.key, AES.MODE_EAX, nonce=self.nonce)
```

```

        ciphertext, tag = cipher.encrypt_and_digest(txt)
        return ciphertext

    def aes_decrypt(self, cipher_text):
        cipher = AES.new(self.key, AES.MODE_EAX, nonce=self.nonce)
        msg = cipher.decrypt(cipher_text)
        return msg

    @staticmethod
    def get_dh_public_key():
        dh = DiffieHellman(group=14, key_bits=540)
        pk = dh.get_public_key()
        return dh, pk

    @staticmethod
    def get_dh_shared_key(dh_1, pk_2, lngth=32):
        dh_shared = dh_1.generate_shared_key(pk_2)
        return dh_shared[:lngth]

if __name__ == "__main__":
    import pickle
    import ast

    text = pickle.dumps("hello world 1234567")
    PUBLIC_KEY = b"it is my secret password"
    NONCE = b"better to try than not try"
    print("start text:", text)

    c1 = Cipher(PUBLIC_KEY, NONCE)
    encrypted_text = c1.aes_encrypt(text)
    c2 = Cipher(PUBLIC_KEY, NONCE)
    message = c2.aes_decrypt(encrypted_text)
    print("after text: ", pickle.loads(message))

    dh1, dh1_public = Cipher.get_dh_public_key()
    dh2, dh2_public = Cipher.get_dh_public_key()

    sk1 = Cipher.get_dh_shared_key(dh1, dh2_public)
    sk2 = Cipher.get_dh_shared_key(dh2, dh1_public)
    print("shared key 1: ", sk1)
    print("shared key 2: ", sk2)

```

## 8.4 drawing.py

```
from PIL import Image, ImageDraw
```

## יואב זלכה: קנבס שיתופי

```
class Drawing:
    def __init__(self, color, width, pt_list=[], id_=-1):
        self.pt_list = pt_list.copy()
        self.id_list = []

        self.id = id_ # id of the whole drawing in the sql db, not an
id of a stroke on the canvas

        self.color = color
        self.width = width

    def add_point(self, point):
        '''
        Recives a point as a tuple adds it to its list
        '''
        self.pt_list.append(point)

    def add_id(self, id):
        '''Receives an id and adds it to the list of ids'''
        self.id_list.append(id)

    def draw_oval(self, canvas, point):
        """Draws an oval in a given point"""
        x, y = point
        # offset, got to be slightly lower than width/2 so it wont
buldge out
        circle_off = self.width/2 - 1
        c_x1, c_y1 = (x - circle_off), (y - circle_off)
        c_x2, c_y2 = (x + circle_off), (y + circle_off)
        id = canvas.create_oval(c_x1, c_y1, c_x2, c_y2,
                                fill=self.color, outline="")
        self.id_list.append(id)

    def delete_from_canvas(self, canvas):
        '''Deletes the entire drawing from the canvas.'''
        for id_ in self.id_list:
            canvas.delete(id_)

        self.id_list = []

    def draw(self, canvas):
        '''
        draws the entire drawing
        '''

        if len(self.pt_list) == 1: # Deals with the case of only one
point in the drawing
            self.draw_oval(canvas, self.pt_list[0])
```

```

        return

        # converts the list of tuples to a list of the format
        [x,y,x,y,x,y,...]
        flattened = [a for x in self.pt_list for a in x]
        id_ = canvas.create_line(*flattened, width=self.width,
        fill=self.color)

        self.id_list.append(id_)

        if (self.width > 3): # otherwise it looks weird
            self.draw_oval(canvas, self.pt_list[0])
            self.draw_oval(canvas, self.pt_list[-1])

    def draw_PIL(self, draw: ImageDraw.Draw):
        draw.line(self.pt_list, fill=self.color, width=self.width)
        for pt in self.pt_list:
            x, y = pt
            # offset, got to be slightly lower than width/2 so it wont
            buldge out
            circle_off = self.width/2 - 1
            c_x1, c_y1 = (x - circle_off), (y - circle_off)
            c_x2, c_y2 = (x + circle_off), (y + circle_off)
            id = draw.ellipse((c_x1, c_y1, c_x2, c_y2),
                             fill=self.color, outline=None)
            self.id_list.append(id)

```

## 8.5 rect\_oval.py

```

import tkinter as tk

class Rect():
    def __init__(self, color, width, x1, y1, x2=None, y2=None, id_=-1):
        self.color = color
        self.width = width
        self.x1, self.y1 = x1, y1
        # if x2 is None, then x2 = x1, if y2 is None then y2 = y1
        self.x2, self.y2 = x2 or x1, y2 or y1
        self.id = id_ # id of the whole drawing in the sql db, not an
        id of a stroke on the canvas

    def update(self, x2, y2):
        self.x2, self.y2 = x2, y2

    def draw(self, canvas: tk.Canvas):
        self.id_in_canvas = canvas.create_rectangle(

```



```

        self.x1, self.y1, self.x2, self.y2, width=self.width,
outline=self.color)

    def delete_from_canvas(self, canvas: tk.Canvas):
        canvas.delete(self.id_in_canvas)

    def redraw(self, canvas: tk.Canvas):
        self.delete_from_canvas(canvas)
        self.draw(canvas)

class Oval(Rect):
    # def __init__(self, color, width, x1, y1, x2=None, y2=None, id_=-
1):
    #     super().__init__(color, width, x1, y1, x2, y2, id_)

    def draw(self, canvas: tk.Canvas):
        self.id_in_canvas = canvas.create_oval(
            self.x1, self.y1, self.x2, self.y2, width=self.width,
outline=self.color)

```

## 8.6 socket\_helper.py

```

import struct
import pickle

class SocketHelper:
    def send_msg(socket, msg):
        # Prefix each message with a 4-byte length (network byte order)
        msg = struct.pack('>I', len(msg)) + msg
        socket.sendall(msg)

    def recv_msg(socket):
        # Read message length and unpack it into an integer
        raw_msglen = SocketHelper.recvall(socket, 4)
        if not raw_msglen:
            return None
        msglen = struct.unpack('>I', raw_msglen)[0]
        # Read the message data
        return SocketHelper.recvall(socket, msglen)

    def recvall(socket, n):
        # Helper function to recv n bytes or return None if EOF is hit
        try:
            data = bytearray()
            while len(data) < n:

```

## יואב זלכה: קנבס שיתופי

```
        packet = socket.recv(n - len(data))
        # print(pickle.loads(packet))
        if not packet:
            return None
        data.extend(packet)
    return data
except ConnectionResetError:
    return None
```

### 8.7 constants.py

```
NONCE = b"Encryption is the process of transforming information into a  
form that is unreadable"
```