

# ספר פרויקט - ענן



[https://github.com/ya5huk/safe\\_cloud](https://github.com/ya5huk/safe_cloud)

שם: אילן ישוק

ת"ז: 326102126

תאריך הגשה: 15 ביוני, 2022

מנחה: ברכה עסיס

חלופה: תכנון ותכנות מערכות מידע

שם הפרויקט: Safe Cloud

בי"ס: מקיף אמי"ת באר שבע

## תוכן עניינים

2	תוכן עניינים
4	מבוא
4	תיאור תכולת הפרויקט
4	מוטיבציה לפרויקט
4	הפרויקט
5	אתגרים שפגשתי במהלך הפרויקט
5	קהל יעד
6	פיתוחים עתידיים
7	ארכיטקטורת הפרויקט
9	ארכיטקטורת רשת – שרת לקוח (Flask)
11	פעולת רישום (register)
11	פעולת חיבור משתמש קיים (login)
12	פעולת אימות אימייל (two_step_auth)
14	פעולת הקבצים (files)
16	פעולת הורדת קובץ (download_file)
16	פעולת מחיקת קובץ (delete_file)
17	פעולת הפרופיל (profile)
17	פעולת התנתקות (logout)
17	פעולת לא נמצא (error 404)
18	אבטחה – גיבוב נתונים, אימות דו-שלבי, הצפנת צד לקוח
18	גיבוב נתונים – המס' זיהוי של המשתמש
18	אימות דו-שלבי
20	הצפנת צד לקוח
22	שימוש בטכנולוגיות חיצוניות (API-ו)
23	מדריך למשתמש
23	התקנת התוכנה
23	הרצת התוכנה
32	בסיס הנתונים – sqlite3
32	טבלת משתמשים (דוגמה)
32	טבלת קבצים (דוגמה)
33	מדריך למפתח
33	קבצי git
38	קבצים סטטיים
39	תמונות הסימול של הקבצים
40	דפי אינטרנט (javascript/css/html)

## SafeCloud

58	קבצי קוד שרת
76	קבצים נוספים
77	רפלקציה

## מבוא

### תיאור תכולת הפרויקט

מסמך זה מאפיין, מפרט ומציג את התוצר הסופי אותו אני מגיש במסגרת פרויקט הגמר במגמת הנדסת תוכנה בכיתה י"ב. בנוסף לתוצר וליכולות שלו, הוא מציג אתגרים, דרכי התמודדות עמם, הסברים לדברים עקיפים בנושא ועוד מגוון רב של תכנים. מסמך זה מהווה את כל מה שצריך לדעת על הפרויקט, ויותר מכך.

### מוטיבציה לפרויקט

כולם מכירים את גוגל דרייב. זו אחת מהתוכנות המשותמשות בעולם. כל כך משומשת, עד שהיא שינתה את הדרך שבה אנו מעבירים קבצים ממחשב למחשב. אמנם גוגל דרייב מצוין בכל מידותיו, אבל אני תמיד רציתי משהו פשוט יותר, מהיר יותר, שניתן לבצע פעולות כמו הוספת/מחיקת/הורדת קבצים בלחיצה אחת.

אני לא מתיימר להיות גוגל דרייב או איי-קלאוד הבא ולא מנסה להתחרות בעשרות אלפי אנשים, אבל אני כן מציג דרך נוספת למשתמש, פשוטה ומהירה לשמירת קבצים בענן, בצורה הכי מאובטחת שיש (אם ירצה בזאת).

### הפרויקט

הפרויקט עצמו הוא ענן, ששומר את הקבצים שלך במאגר נתונים הרחק ממך, ובצורה מאובטחת אם תבחר בכך, בשביל שתוכל להשתמש בקבצים האלו במקומות אחרים, במחשבים אחרים ובזמנים אחרים. אני ניסיתי בכוונה לעשות את הממשק הגרפי פשוט ומהיר ככל הניתן, אני לא אוהב להתעסק עם כל הפונקציונליות (שלדעתי מיותרת) בהרבה מהעננים שיש כיום ולכן ניסיתי לצמצם ככל הניתן במישור הזה.

בשביל להשתמש בענן אתה צריך משתמש. לאחר הרשמה שכוללת אימות דו-שלבי עם האימייל, אתה תגיע לאתר שיכיל את הקבצים שלך. אתה תוכל להוסיף קבצים ע"י גרירתם למסך, למחוק ולהוריד בלחיצה. ניתן להוסיף כמות רבה של קבצים בגרירה אחת. בנוסף, אתה יכול לבחור להוסיף ולהוריד קבצים בעזרת מפתח מיוחד. על המפתח אפרט בהמשך, אבל אציין שבעת הכנסת המפתח, המשתמש מוסיף עוד שכבת הגנה – קריטית וטובה מאוד – לקבציו המועברים באינטרנט אל הענן. חוץ מעמוד הקבצים, יש עמוד פרופיל שמציג את פרטיך ונותן לך להתנתק מהמשתמש אם תרצה ועמוד הסבר על הפונקציונליות של התוכנה.

השרת שלי זוכר משתמשים נכנסים ויוצאים, בצורה בטוחה, אצלו בדיסק. תודות לכך, משתמש לא צריך להכנס כל פעם מחדש לאתר, ולמלא את פרטיו. האתר זוכר אותו למשך זמן של יום אחד (שיש שיגידו קצר ויש שיגידו ארוך, אני מאמין שיום אחד זה כמות זמן טובה לנוחות וטובה לאבטחה). כמובן שהשרת לא יודע לא את סיסמת המשתמש, ואם המשתמש יבחר – גם לא את קבצי המשתמש.

## אתגרים שפגשתי במהלך הפרויקט

פגשתי מס' לא מבוטל של אתגרים בזמן עשיית הפרויקט והנה כמה מהם:

- **אין זמן.** כיתה י"ב עמוסה גם ככה, בגרויות מפזיקה, מתמטיקה, תנ"ך ולכן היה צריך לנהל את הזמן בצורה יעילה, יחד עם הצבת יעדים לסיום חלקים בעבודה.

- **ממשק גרפי.** כמעט ואף פעם לא עבדתי עם ממשק גרפי כמו שעבדתי בעבודה זו. הממשק הוא אתר אינטרנט, ולכן יצא לי לעבוד עם javascript הרבה, הרבה יותר ממה שידעתי או הכרתי. למדתי מהבסיס ביותר - כיצד javascript רץ ועד מטרותיו והדברים הקטנים והנוחים שהופכים את השפה הזו לכל כך משומשת בעולם.

- **תכנות אסינכרוני** - הרבה בעיות נוצרו עקב זה שjavascript היא שפה שעובדת רק בצורה אסינכרונית. כלומר, היה צריך לשנות את התפיסה שלי - שכל שורה שאני כותב, אני צריך להוסיף לה תכנותית, "לחכות" שהשורה תסתיים ורק אז לשלוח את זה. זה כולל שימוש בPromises, async וכו'.

- **אבטחה.** ניסיתי לשים דגש על האבטחה משתי סיבות. ראשית, זהו פרויקט בסייבר, כלומר צריך לגעת באבטחת מידע ורשתות. שנית, בן אדם שירצה לשלוח את הקבצים האלו, ירצה ברוב המקרים, שהם יישארו רק שלו ואף אחד, כולל אני (או מנהל מאגר הנתונים) לא ידע מה יש שם. האתגר הוא - כיצד לעשות את זה? מאיפה מתחילים בכלל?

- **הקלאסי.** לשבת על בעיה כמה שעות, שאין לך שמץ של מושג למה משהו לא עובד בכלל. אתן דוגמה על מה אני מדבר. כשאני מוסיף משתמש למאגר נתונים, אני עושה גיבוב של חלק מהנתונים שלו, בצורה כזו שיהיה קשה מאוד (כל כך שאפשר להגיד בלתי אפשרי) לחזור לנתוני המקוריים (מהמחרוזת המגובבת). הבעיה שלי הייתה שמשום מה זה לא היה קבוע. בסופו של דבר, אחרי המון בדיקות הגעתי לפתרון והוא להוסיף סוגריים ל `datetime.timestamp` משום שזו פעולה. נשמע טיפשי נכון? אבל אז קפצה בעיה אחרת. עדיין הגיבוב לא היה קבוע. הפעם גיליתי, שאני שומר תאריך יצירת משתמש מדויק - עד לשניות במאגר הנתונים אבל כשהייתי עושה את הגיבוב (לשם השוואה לאחר הכנסת משתמש נכנס), התאריך היה מדויק לרמת המיקרו-שניות, ולכן בעת הוצאה ממאגר הנתונים וגיבוב המידע, לא הייתי מקבל שווי בין נתונים למרות שהיו שווים. בקיצור... הקלאסי. וזה קרה פעמים רבות, אבל זה התהליך לתוצר הסופי וכמות הלמידה שניתן להוציא מנסיונות כאלו היא ללא הגבלה.

## קהל יעד

קהל היעד של הפרויקט הוא בערך כל מי שמשתמש היום באינטרנט - כלומר כל אחד שרוצה את הקבצים שלו ביותר ממקום אחד. אנשים שעובדים עם חומר וצריכים אותו מסודר - רופאים, מורים, הייטק, פקידים, תלמידים ועוד...

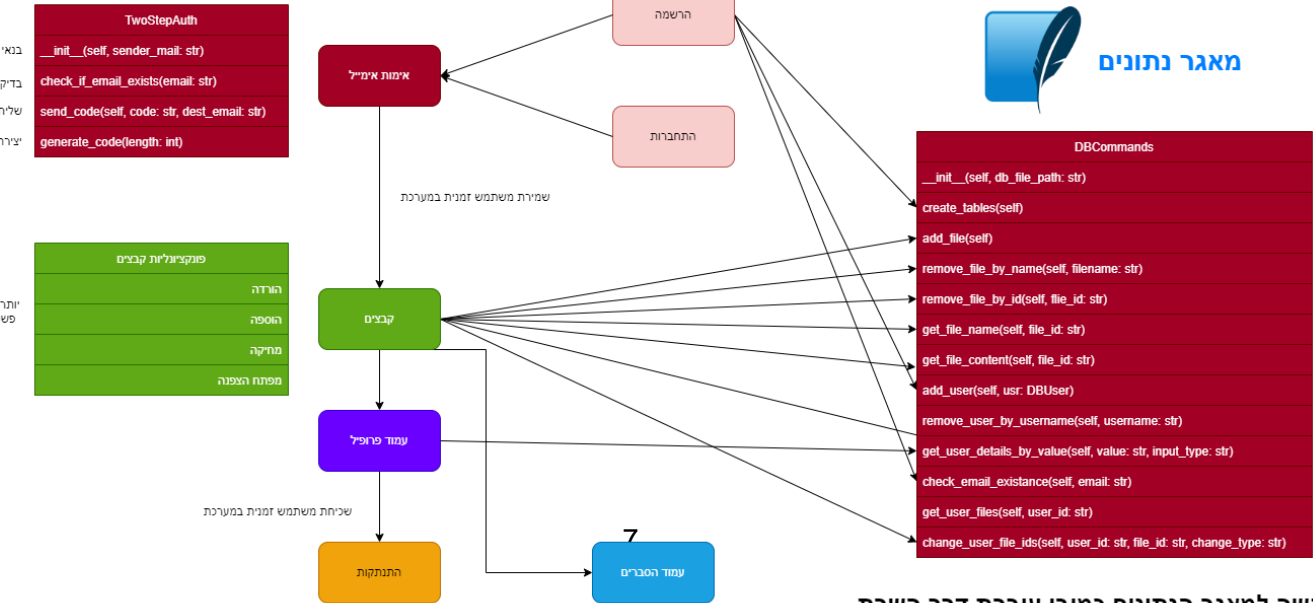
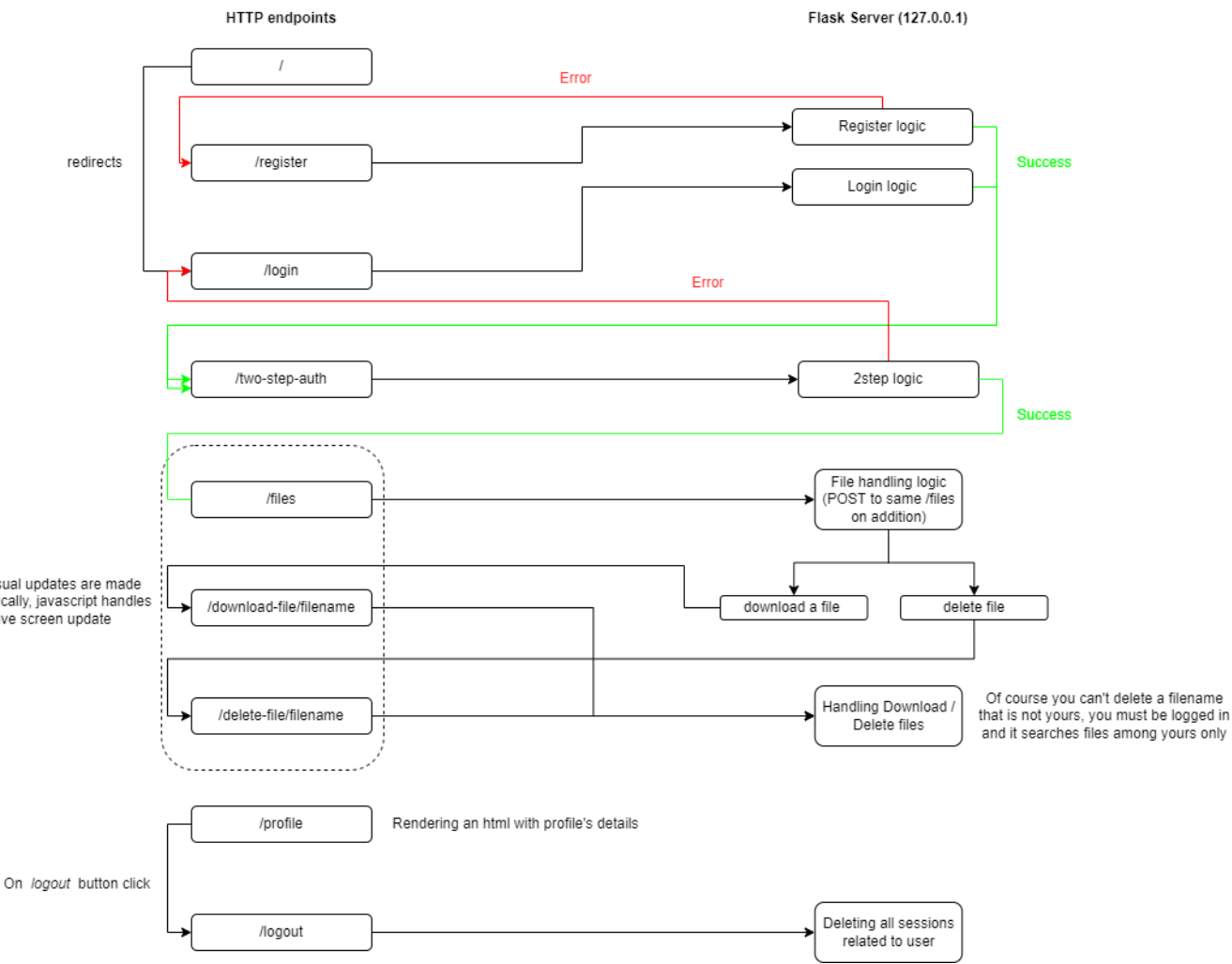
## פיתוחים עתידיים

כמעט כל מה שתכננתי לפרויקט, בוצע. לכן, אין הרבה פערים שנוצרו אבל יש דברים שהיה נחמד להוסיף:

- שיתוף קבצים בין כמה אנשים, כלומר קישור קובץ מסוים ליותר מבן אדם אחד
- יצירת תיקיות שיכילו קבצים (כמובן לא באופן "זריקת תיקייה"... כי בדקתי... לא כל כך אפשרי)

# ארכיטקטורת הפרויקט

ניתן לייצג את ארכיטקטורת הפרויקט באמצעות התרשימים הבאים:



כמו כן, לחלק את הפרויקט לכמה מישורי עבודה שונים:

**- מאגר נתונים:**

- טבלת משתמשים

- טבלת קבצים

**- אבטחה:**

- אימות משתמש דו-שלבי

- הצפנת קבצים במאגר הנתונים

- גיבוב פרטיו של המשתמש ליצירת מחרוזת מזהה ייחודית

**- ממשק גרפי**

- תהליך הרשמות משתמש חדש

- תהליך התחברות של משתמש קיים

- ניהול קבצים (הוספה/מחיקה/הורדה)

הממשק הגרפי הוא אתר, ולכן בקשות HTTP (POST, GET) מהאתר יהיו הקלטים. הבקשות יהיו הקלטים לשרת והכנסת הנתונים לאתר עצמו יהיו הקלטים של המשתמש. קלטים של שאר המישורים, אבטחה ומאגר נתונים, אלו קלטים שמספק אך ורק השרת (כמובן לפי בקשות המשתמש).

**הסדר שמוצג פה הוא גם הסדר לפיו יוצגו החומרים בספר הפרויקט, משום שבדרך זו הכי קל להבין מדוע כל דבר נעשה בצורה שנעשה. הסבר מאגר הנתונים יהיה תחת הכותרת 'בסיס נתונים' לאחר המדריך למשתמש.**



## ארכיטקטורת רשת – שרת לקוח (Flask)

ארכיטקטורת הרשת היא שרת-לקוח, ארכיטקטורה פופולרית מאוד לשם תוכנות כמו שאני יצרתי. כלומר, בקשות חד-פעמיות מהלקוח אל השרת בכל פעם שהוא צריך משהו. לדוגמה, בכניסה לדף הרשמות, הלקוח (או יש שיגידו הדפדפן) ישלח בקשה לכתובת <http://127.0.0.1:5000/login> ויקבל תשובה בהתאם למה שיבקש. אם ישלח בקשת [HTTP GET](#) יקבל את הדף של התחברות, אם ישלח בקשה [HTTP POST](#) עם מידע מסוים, הוא יקבל עליו פידבק מסוים מהשרת, ואם הפידבק חיובי (ההתחברות צלחה) אז הוא יופנה ל <http://127.0.0.1:5000/two-step-auth>.

לפני שאציג את הפעולות השונות שמתארות את **זרימת התוכנה**, אני אראה את המשתנים שיהיו לנו רלוונטיים בהמשך, להבנה טובה יותר של הפעולות.

אתחיל מהספריות שאני מייבא לקובץ

```
from base64 import b64encode
from datetime import datetime, timedelta
from flask import Flask, jsonify, session, redirect, url_for,
render_template, request as rq
from flask_session import Session # Server-side session
from CloudServer import CloudServer
from TwoStepAuth import TwoStepAuth
import CloudEncrypt
import re, os
```

הספריות כוללות:

- קידוד ב base64
- זמנים בפיתון
- Flask, שזה הנס שבזכותו הכל מקושר בין אתר אינטרנט לבין הקוד. זוהי מעטפת, שבזכותה ניתן ליצור שרת שיטפל בבקשות HTTP לכתובת מסוימת שאנחנו נחליט.
- Sessions, הדרך לשמור נתונים על המשתמש, לאורך כל ריצת הקוד, והדפים השונים. **יש לשים לב** שאני מייבא sessions פעמיים משום שאני רוצה ששמירת הנתונים תהיה בתוך מערכת הקבצים של השרת. אני רוצה את זה מטעמי בטיחות, אסביר בהמשך.
- CloudServer, TwoStepAuth, CloudEncrypt, ספריות שאני כתבתי. הראשונה מתעסקת בגיבוב נתונים, השנייה באימות דו-שלבי והשלישית מטפלת בבקשות שונות של המשתמש.
- Re, דרך לבדוק שהמשתמש מכניס מחרוזות בתבנית מסוימת שאנחנו קובעים
- Os, ספרייה שמתארת את מערכת ההפעלה. מיועדת לפעולות שרק מערכת ההפעלה יכולה לבצע.

## SafeCloud

משתנים שאני שומר לפני הרצת הקוד:

```
DB_FILENAME = os.path.abspath('./database.db')
EXTENSION_ICONS_PATH = os.path.abspath('./images/extension_icons') +
 '/'
tsa = TwoStepAuth('ilan147963@gmail.com')
auth_codes = [] # List of {email: code, email: code} used for 2step
auth
# just to prevent it from be shared in session

app = Flask(__name__)

app.config['SESSION_TYPE'] = 'filesystem' # For server-side sessions
Session(app)

# Secret key, session encrypting
app.secret_key = 'dev' # Const in development
app.config.from_pyfile('config.py', silent=True) # Overrides if
config.py exists

app.permanent_session_lifetime = timedelta(days=1)

cs = CloudServer(DB_FILENAME, EXTENSION_ICONS_PATH)
```

- DB\_FILENAME, כתובת הקובץ של מאגר הנתונים
- PATH\_ICONS\_EXTENSION, כתובת התיקיה של האייקונים
- Tsa, מחלקת אימות דו-שלבי
- Auth\_codes, משתנה שמקשר בין כתובות אימייל לקודים. ככה אנו זוכרים איזה קוד אנו יוצרים לאיזה אימייל באימות הדו-שלבי
- sessions מגדירים את המפתח הסודי שמשמש להצפנת sessions
- מגדירים שהsession ישכח לאחר יום אחד
- יוצרים מחלקה של השרת (CloudServer), שאחראית לפעולות שיבקש השרת

כעת אנחנו מוכנים לצאת לדרך! אנסה להציג בעיקר את הדברים החשובים, משום שאם אציג הכל אורך הספר יהיה ארוך בצורה יוצאת דופן. להלן הפעולות מצד השרת שמופעלות לפי בקשות HTTP:

הפעולה הכי בסיסית, מפנה אדם שפונה לכתובת יחסית "/" אל דף התחברות:

```
@app.route('/')
def home():
    if 'user_id' in session:
        return redirect(url_for('files'))
    return redirect(url_for('login'))
```

## פעולת רישום (register)

משתמש חדש, כאשר אדם מופנה ליצירת משתמש:

```
@app.route('/register', methods=['POST', 'GET'])
def register():
    # The weird microsec cutting causes problems in login so I cut it
    אם המשתמש יטען את האתר הוא יקבל את עמוד ההרשמות, אם יש שגיאה היא תוצג.
```

```
err_msg = session.pop('err_msg', None)
if err_msg:
    return render_template('register.html', err_msg=err_msg)
return render_template('register.html')
```

אם המשתמש ישלח נתונים להרשמות משתמש. השרת יבדוק האם אפשר לרשום את המשתמש (יבדוק שאימייל או משתמש לא תפוסים). אם הכל יתבצע כראוי, השרת יסמן שהאדם הספציפי הזה עתיד להירשם (לשימושים עתידיים) וישלח אותו לעמוד אימות האימייל.

```
if rq.method == 'POST':
    email = rq.form['email']
    username = rq.form['username']
    password = rq.form['password']

    ans = cs.register_possibility(email, username)
    if ans['code'] == 'success':
        # 2step auth is unique for both login, signup
        # But session vars preparation is needed
        session['password'] = password
        session['username'] = username
        session['email'] = email
        session['action'] = 'register'
        return redirect(url_for('two_step_auth'))
    else:
        return render_template('register.html', err_msg=ans['msg'])
```

## פעולת חיבור משתמש קיים (login)

```
@app.route('/login', methods=['POST', 'GET'])
def login():
```

למקרה של בקשת טעינה:

```
err_msg = session.pop('err_msg', None)
if err_msg:
    return render_template('login.html', err_msg=err_msg)
return render_template('login.html')
```

למקרה של שליחת נתונים:

```

if rq.method == 'POST':
    usr_input = rq.form['user_input']
    usr_pass = rq.form['password']
    ans = cs.try_login(usr_input, usr_pass)
    if ans['code'] == 'error':
        return render_template('login.html', err_msg=ans['msg'])
    else:
        user = cs.get_user_details(ans['msg'], 'user_id')
        if not user:
            return render_template('login.html', err_msg='No such
user..')
        session['email'] = user.email # Where to send
        return redirect(url_for('two_step_auth'))

```

הפעולה בודקת האם אפשר להכניס את המשתמש לפי הנתונים שהכניס. מאחר ואפשר להכניס אימייל ושם משתמש בעמודה העליונה בהתחברות, הפעולה try\_login אינה פשוטה, אתם מוזמנים להסתכל עליה במדריך למפתח. לאחר הבדיקה, אם יש בעיה בהתחברות אנו מציגים אותה, ואם אין – מפנים לחלון לעמוד אימות האימייל.

## פעולת אימות אימייל (two\_step\_auth)

הפעולה הזו יותר מורכבת מכל מה שנכתב עליו עד כה, משום שהיא פעולה שפועלת גם על משתמש מתחבר וגם על משתמש חדש שנרשם, אז יש צורך לתמרן בין השניים.

לפני הפעולה, נגדיר משתנים שיעזרו לנו בהמשך:

```

tsa = TwoStepAuth('ilan147963@gmail.com')
auth_codes = [] # List of {email: code, email: code} used for 2step
auth
# just to prevent it from be shared in session

```

הכותרת:

```

@app.route('/two-step-auth', methods=['POST', 'GET'])
def two_step_auth():
    redirect_to = 'files' # Where after 2step auth succeeds

```

העמוד הבא אליו נפנה תלוי במשתנה redirect\_to, כרגע זהו עמוד הקבצים, שאנו מפנים אליו אחרי התחברות תקינה.

למקרה של טעינת העמוד:

```

if 'email' not in session:
    return redirect(url_for('login'))

email = session['email']

if not tsa.check_if_email_exists(email):
    session['err_msg'] = 'Email does not exist!'
    session.pop('email', None) # Clear session

```

```

return redirect(rq.referrer)

code = tsa.generate_code(6)
auth_codes.append({'email': email, 'code': code})

tsa.send_code(code, email)

return render_template('code_enter.html')

```

אנו בודקים האם המשתמש הכניס אימייל. אם לא – מפנים חזרה לחיבור ואם כן, בודקים שהאימייל לא רק בעל תבנית נכונה אלא רשום בתיבות השונות. אם האימייל לא מאושר, אנו מחזירים את המשתמש לפעולתו הקודמת. אם כן, אנו יוצרים קוד, מוסיפים אותו לטבלה גלובלית (auth\_codes) שמקשרת בין אימייל לקודים (בשביל לעקוב אחרי הקודים שאנחנו שולחים, שנוכל להשוות). לבסוף, אנו שולחים את הקוד לתיבת הדואר של המשתמש.

למקרה של הכנסת נתונים. אנו קוראים את מה שהוכנס, ומשיגים את האימייל והפעולה שהמשתמש עושה. הפעולה היא רישום/התחברות.

```

if rq.method == 'POST':
    # If we received a POST req, then we already
    # did a GET here, and values are popped
    # so this if must be first, to not get redirected
    received_code = rq.form['code_input']
    email = session.pop('email', None)
    user_action = session.pop('action', None)

```

עכשיו, נבדוק האם יש התאמה בין הקוד שנשלח ושומר בauth\_codes לבין הקוד שהוכנס. אם יש התאמה, נגדיר את זכירת המשתמש לקבועה (למשך יום אחד)

```

for ac in auth_codes:
    if ac['email'] == email and ac['code'] == received_code:
        # code that is typed is connected to the email
        # success
        auth_codes.remove(ac)
        session.permanent = True

        # Only in register -> Create an account
        if user_action == 'register':
            print('registering')
            password = session.pop('password')
            username = session.pop('username')

            # Conversion is to delete micro-secs from now(),
            # makes problems...
            creation_time = datetime.now().strftime("%d/%m/%Y,
            %H:%M:%S")
            creation_time = datetime.strptime(creation_time,
            "%d/%m/%Y, %H:%M:%S")

```

```

        user_id = CloudEncrypt.hashify_user(email,
password, creation_time)
        cs.register(user_id, username, email,
creation_time)

        session['user_id'] = user_id # as a setup to /files

```

אם המשתמש מנסה להרשם, אנו יוצרים לו משתמש, מגבבים לו את הנתונים בצורה שתוארה בקובץ תחת הכותרת 'אבטחה' ולבסוף, קוראים לפעולה `register` שרושמת את המשתמש במאגר הנתונים. אנו גם מוסיפים ל-`session` שהוא אחראי על שמירת משתנים לטווח הקצר את מס' הזיהוי של המשתמש בשביל שנוכל לתקשר אותו בצורה בטוחה אחר כך.

אם המשתמש מנסה להתחבר ולא להרשם, נדלג על כל השלב של הרישום שמתרחש למעלה, ופשוט נעביר אותו לעמוד הקבצים:

```

else:
    session['user_id'] = cs.get_user_details(email,
'email').user_id # as a setup to /files

    session['filenames'] = []
    return redirect(url_for(redirect_to))

```

בכל הפעולות הבאות אנחנו בודקים תחילה אם המשתמש מחובר, משום שאם לא – אסור שיהיה לא את היכולות להתחבר בכלל. אנו בודקים זאת באמצעות חיפוש זהותו ב-`session`, אם יש לו מס' זיהוי רשום, אז הוא יכול להתחבר. אם אין לו, אז לא.

## פעולת הקבצים (files)

אם יש משתמש מחובר, אנחנו משיגים את קבצי באמצעות `get_user_filenames` ו-`get_file_icon_content` (מידע לתמונת סיומת הקובץ ומידע על שמות הקבצים). אנו שומרים את שמות הקבצים ב-`session` בשביל שנזכור אותם למשך חיבור המשתמש ולא נצטרך לטעון אותם כל פעם מחדש. אחרי השגת כל הקבצים, אנחנו טוענים אותם למסך בעזרת תבנית שקיימת מראש ע"י פעולת `render_template`.

```

@app.route('/files', methods=['POST', 'GET'])
def files():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    if rq.method == 'GET':
        # Fetch user's files
        user_id = session['user_id']
        filenames = cs.get_user_filenames(user_id)
        session['filenames'] = filenames

```

```

files_data = []
for fn in filenames:

    icon_content = b'data:image/png;base64,' +
cs.get_file_icon_content(fn)
    files_data.append(
        {'name': fn,
         'icon_data': icon_content.decode()})
return render_template('files.html', files_data=files_data)

```

אם משתמש רוצה להוסיף קובץ, הוא ישלח בקשת POST לאותה כתובת. במקרה זה השרת יאסוף את המידע הרלוונטי על הקבצים שהוכנסו (או קובץ יחיד, עובד לכל המקרים). לאחר מכן, לכל קובץ, הוא ישנה את השם בהתאם להימצאותו במאגר הנתונים (בשביל לא ליצור בעיות שכתוב קבצים).

```

if rq.method == 'POST':
    icons_data = []
    files_num = int(rq.form['length'])
    for i in range(files_num):

        file = rq.files[f'files-{i}']
        name = file.filename
        content = file.stream.read()

        # Manage duplicates, edits session['filenames']
        saved_filename = configure_filename(name,
session['filenames'])

```

לבסוף הוא יוסיף את הקובץ וישייך אותו למשתמש על ידי `add_file_to_user`. בנוסף, הוא יוסיף את תמונת הסיומת של האייקון ושמו למשתנה מסוג מילון שיישלח לאחר מכן למשתמש.

```

file_id, file_icon_data = cs.add_file(False,
saved_filename, content)

# Update user's files
cs.add_file_to_user(session['user_id'], file_id)
icons_data.append({'data': file_icon_data.decode(),
'filename': saved_filename})

return jsonify(icons_data)

```

המשתמש כעת יוכל ליהנות מהתעדכנות אוטומטית של הקבצים אצלו על המסך.

## פעולת הורדת קובץ (download\_file)

תחילה הפעולה בודקת אם הקובץ שנתבקש להורדה, הוא של המשתמש. אם השם של הקובץ אכן יהיה של המשתמש, אז תוכן הקובץ של המשתמש יוחזר חזרה למבקש הפעולה (לקורא לכתובת './files/download/some\_filename.txt').

```
@app.route('/files/download/<filename>')
def download_file(filename: str):
    if 'user_id' not in session:
        return redirect(url_for('login'))
    # Search for filename among only the logged account
    files_id = cs.get_user_file_ids(session['user_id'])
    for fid in files_id:

        if cs.get_filename(fid) == filename:
            # Found relevant file, so I can download it
            content = cs.return_file_content_by_id(fid)

            return b64encode(content)
    return redirect(url_for('login')) # If file wasn't found
```

## פעולת מחיקת קובץ (delete\_file)

דומה לפעולת ההורדה, רק שבמקרה זה אנו מוחקים את הקובץ ומורידים את הבעלות של המשתמש לקובץ. כמו כן, אנו מורידים את שם הקובץ מהsession לטעינה עתידית. כעקרון אנו לא צריכים להחזיר כלום, אבל מפני שזו חובה, אנחנו מחזירים שמחקנו את הקובץ.

```
@app.route('/files/delete/<filename>')
def delete_file(filename: str):
    if 'user_id' not in session:
        return redirect(url_for('login'))
    # Search for filename among only the logged account
    files_id = cs.get_user_file_ids(session['user_id'])
    for fid in files_id:

        if cs.get_filename(fid) == filename:

            cs.delete_file_by_value(fid, 'file_id') # files
            cs.remove_file_from_user(session['user_id'], fid) # user
            session['filenames'].remove(filename) # session

            # Doesn't really matter if we didn't delete something that
            # didn't exist
            return jsonify({'message': 'Delete occurred'})
```



```
return redirect(url_for('login')) # No file found
```

## פעולת הפרופיל (profile)

פעולה פשוטה מאוד שבעיקרה מציגה את הפרופיל ונותנת אפשרות להתנתק מהאתר למשתמש. משיגה את כל המידע הרלוונטי בעזרת `get_user_details` ואם חוזר משתמש קיים, היא טוענת את המסך עם נתוניו.

```
@app.route('/profile')
def profile():
    if 'user_id' not in session:
        return redirect(url_for('login'))
    user = cs.get_user_details(session['user_id'], 'user_id')
    if user:
        # Send in it parts because I don't want all the details
        # out there, sounds risky
        return render_template('profile.html',
                               username=user.username,
                               email=user.email,
                               creation_date = user.creation_date.strftime("%d %B, %Y"),
                               )
    return 'error'
```

## פעולת התנתקות (logout)

בפנייה לכתובת זו, האתר ישכח את המשתמש. לאחר הניתוק הוא יכוון אותנו חזרה לעמוד ההתחברות.

```
@app.route('/logout')
def logout():
    if 'user_id' in session:
        session.pop('user_id', None)
        session.pop('filenames', None)
    return redirect(url_for('login'))
```

## פעולת לא נמצא (error 404)

למקרה והמשתמש הגיע לדף שלא קיים (לא אמור לקרות, אלא אם כן המשתמש מכניס כתובת ידנית), אני מציג דף בהתאם:

```
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html', err_msg=e)
```

## אבטחה – גיבוב נתונים, אימות דו-שלבי, הצפנת צד לקוח

לכל הקבצים הרלוונטיים, ראה כותרת "המדריך למפתח". בחלק זה אציג את הדברים השונים שאני עושה למען ביטוח והצפנה של פרטי ומידע המשתמש. הכותרות מדורגות לפי רמת האבטחה, כלומר גיבוב נתונים מהווה אבטחה ברמה גבוהה, אך אין פה קונספט יוצא מן הכלל. לעומת זאת, הצפנת צד-לקוח היא מההצפנות החזקות שקיימות, לא מעצם אלגוריתם מופלא או טריק מתמטי (למרות שגם מזה אפשר להתפאר) אלא מהקונספט אותו אסביר תחת התת-כותרת המתאימה.

### גיבוב נתונים – המס' זיהוי של המשתמש

אסור בשום פנים ואופן שהשרת, או האחראי על מבנה הנתונים, יוכל לגשת למשתמשים השמורים במאגר. דבר כזה יהיה אסון אם יתגלה על ידי המשתמשים משום שיהווה פגיעה איומה בפרטיות שלהם. לצורך מניעת בעיה כזו, אני מקבל את פרטי המשתמש, מפעיל עליהם פונקציית גיבוב, שכידוע לא ניתן לשחזרה (מכיוון שמדובר [בפונקציות מתמטיות חד-כיווניות](#)). הנה הקוד:

```
def hashify_user(email: str, password: str, creation_date: datetime):
    # Some dynamic salting for md5 hash

    creation_time_mixed = str(creation_date.timestamp())[:-2]
    salt = hashlib.md5(email.encode()).digest()[::2] + \
        hashlib.md5(creation_time_mixed.encode()).digest()[::3]
    encstr = hashlib.md5(password.encode() + salt).hexdigest()
    return encstr
```

- קבלת אימייל, סיסמה, תאריך יצירה כקלט
- הפיכת זמן יצירה לכמות שניות מאז 1970 ובלבול המחרוזת
- יצירת [מלח](#), שהוא גיבוב מחרוזת אימייל מבולבלת בתוספת גיבוב המחרוזת המבולבלת של זמן היצירה
- גיבוב המלח בתוספת הסיסמה לקבלת התוצאה הסופית
- התוצאה היא גיבוב הנתונים, תוצאה יחידה במינה לכל משתמש, מס' זיהוי המשתמש

### אימות דו-שלבי

אימות דו-שלבי קיים בשביל לוודא שגם אם פרטיו של המשתמש נחשפו, עדיין לא יהיה אפשר להכנס לאתר משום שיהיה חייב גם להשיג את פרטי המשתמש לאימייל. כלומר, זה מקשה על הפורץ משום שעליו גם לפרוץ לתיבת הדואר של אותו משתמש, דבר שלא יקרה אם המשתמש לא ייפול למרמות מסוימות.

## SafeCloud

הוא עובד בצורה הבאה, ישנו אובייקט TwoStepAuth שנוצר עם הרצת קובץ המקור של האתר. האובייקט נוצר פעם אחת, מתחבר לתיבת הדואר של גוגל (משום שהאימייל השולח מאוחסן בשרתים אלו).

```
def __init__(self, sender_email: str):
    # Personal details
    self.sender_email = sender_email
    self.smtp_server = 'smtp.gmail.com'

    # Create .env for these
    self.username = config('EMAIL_USERNAME')
    self.password = config('EMAIL_PASSWORD')
    print(f'Trying to log in with {self.username},
{self.password}... ', end='')

    self.conn = SMTP_SSL(self.smtp_server, 465)
    self.conn.set_debuglevel(False)
    self.conn.login(self.username, self.password)
    print('Logged in!')
```

הוא כולל את הפעולות הבאות:

1. בדיקה האם אימייל באמת קיים במערכות השונות (לא רק אם הוא כתוב בצורה מותרת מבחינת תווים, אורך וכו')

```
@staticmethod
def check_if_email_exists(email_address: str):
    url = 'https://isitarealemail.com/api/email/validate' # A
    blessing for developers
    res = requests.get(url, params={'email': email_address})
    status = res.json()['status']

    if status == 'valid':
        return True
    return False
```

2. יצירת קוד רנדומלי

```
@staticmethod
def generate_code(len: int):
    letters = 'abcdefghijklmnopqrstuvwxyz'
    letters_big = letters.upper()
    digits = '0123456789'
    final_str = ''
    for i in range(len):
        element = random.choice([letters, letters_big, digits])
        final_str += random.choice(element)
    return final_str
```

```
def send_code(self, code: str, dest_email: str):
    # Beautify
    text_subtype = 'html'
    content = f"""
    <p>Thanks for logging in.</p>
    <p>Code: <b>{code}</b></p>
    """
    subject = 'SafeCloud code confiramtion'
    try:
        msg = MIMEText(content, text_subtype)
        msg['Subject'] = subject
        msg['From'] = 'SafeCloud'
        self.conn.sendmail(self.sender_email, dest_email,
msg.as_string())

    except Exception as e:
        print(f'Message failed: {e}')
```

תחילה יש הגדרה ראשונית של האובייקט, שכוללת התחברות לאימייל עם שם משתמש  
 וסיסמה שנמצאים במשתנים סביבתיים (EMAIL\_USERNAME, EMAIL\_PASSWORD).

כאשר משתמש מתחבר או נרשם למערכת, תחילה יש בדיקת תקינות אימייל ולאחר מכן  
 נוצר קוד רנדומלי ונשלח לאימייל שהתחבר.

למען ביטחון החיבור, יש סגירה נכונה של החיבור בין תיבת הדואר לאימייל השולח. הסגירה  
 מתבצעת אם הקוד מפסיק לעבוד, או קופצת איזושהי תקלה:

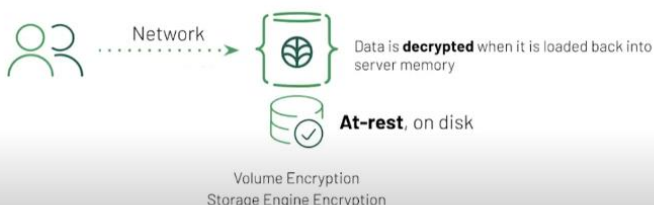
```
def __del__(self):
    print('Closing smtp connection')
    self.conn.close()
```

## הצפנת צד לקוח

ההצפנה הכי חזקה לדעתי, היא הצפנה שאף אחד מלבדך לא יכול לפצח. הפורץ בעצם יכול  
 לראות את הקובץ שהלקוח שולח, רק ברגע אחד, והוא בין השליחה לבין "המנוחה" במאגר  
 הנתונים. כלומר, אם אני שולח לשרת קובץ, הוא יהיה חשוף לגמרי בינתיים שהשרת עובד

עליו, ובזמן זה משהו יכול בצורה כלשהי  
 להאזין לשרת ולגלות את הקובץ (כמובן  
 שאנו מנסים להימנע מכך אך באבטחה  
 צריך תמיד להתכונן למקרה הנורא מכל).  
 בשביל **לטפל** בזה, אנו פשוט מסתירים  
 את הקובץ מהרגע שהוא עוזב את  
 המחשב. ככה, אף אחד, לא השרת ולא

Moving and **storing** data: most databases  
 have it covered



מבנה הנתונים ולא הפורץ, אף אחד, לא יוכל לדעת מה יש בקובץ. אנו עושים זאת מאוד פשוט (ולכן אמרתי שהרעיון מלהיב יותר מהביצוע). ההצפנה היא AES שממומשת בספריה CryptoJS בעת העלאת הקובץ (אתמקד בחלקים החשובים), אנו יוצרים פקודה לקריאתו, בצד הלקוח. לאחר הקריאה, אנו מתכוננים לשליחת המידע לשרת אבל לפני זה, בודקים אם הוכנס קוד סודי (ניתן לראות במדריך למשתמש בדף הקבצים). אם כן:

```
if (pk != '') {  
    // AES client-side encryption is needed  
    let wordArray = CryptoJS.lib.WordArray.create(file.content);  
    let encrypted = CryptoJS.AES.encrypt(wordArray,  
pk).toString();  
    formData.append(`files-${counter}`, new Blob([encrypted]),  
file.filename);  
}
```

אנו מצפינים את המידע בקובץ (file.content) ורק אז מוסיפים את המידע שלו. בעת ההורדה אנו עושים אותו דבר, מפענחים לפי מפתח. byteCharacters מייצג ב-base64 את הקובץ:

```
// Convert base64 to blob  
if (pk != '') {  
    let decrypted = CryptoJS.AES.decrypt(byteCharacters, pk)  
    let typedArray = convertWordArrayToUint8Array(decrypted);  
    saveData(filename, `data:${extension}`, new  
Blob([typedArray]));  
    return false  
}
```

saveData היא פעולה שדואגת לשמור את הקובץ במחשב המשתמש, ניתן לראות אותה במדריך למפתח.

## שימוש בטכנולוגיות חיצוניות (ו-API)

להקמת פרויקטים כאלו, לא ניתן לממש הכל מהתחלה מסיבה ראשית ועיקרית, אין למה להמציא את הגלגל מחדש, אם נתמקד כל פעם מחדש בהמצאת הגלגל לעולם לא נגיע למכונית. לכן, השתמשי בטכנולוגיות שונות שאציג כאן:

פירוט	טכנולוגיה
למען הפיכת השרת לשרת HTTP שיכול לטפל בכל בקשות HTTP ולהחזיר מידע בקלות (יחסית לsocket רגיל)	<a href="#">Flask</a> ונגזרותיו
בשביל לשמור את המידע הזמני על המשתמשים בקבצי השרת. מטרת הספרייה היא להזיז את מיקום המידע הנשמר מקבצי המשתמש לקבצי השרת, וזאת מטעמי אבטחה – אם אני מעביר פרטים כמו סיסמאות, אני מעדיף שישמרו בשרת.	<a href="#">Flask session</a>
לגיבוב הנתונים של המשתמשים	<a href="#">Hashlib</a>
ליצור מס' זיהוי ייחודי לקבצים שנוספים למערכת	<a href="#">Uuid</a>
קידוד קבצים בפורמט base64, שהוא הופך מידע לבינארי לטקסטואלי	<a href="#">Base64</a>
דרך בפייתון לעבוד עם תהליכים שונים בו זמנית (לדוגמה לתקשר עם שני משתמשים בו זמנית)	<a href="#">threading</a>
תיקייה שאחרית לחיבור לשרת SSL. השימוש בו הוא לאימות דו-שלבי עם google gmail. בשביל להתחבר לאימייל זה, יש ליצור חיבור מאובטח באמצעות SSL.	<a href="#">Smtplib</a>
API שבודק האם אימייל מסויים הוא אימייל שבאמת רשום במערכת, ולא סתם אימייל שמוקלד	<a href="#">Is it a real email?</a>
לשמירת וניהול גרסאות של הקוד. עוזר לסידור, ועבודה בכמה מקומות במקביל.	<a href="#">github</a>
להצפנת קבצים מצד לקוח בהצפנת AES. משומש בjavascript בקוד לקוח	<a href="#">CryptoJS</a>
ספריות קלות (מבחינת זכרון) המקלות על שליחת בקשות וטעינת גופים בצירה דינמית לתוך אתר. הספריות משומשות בjavascript וכל כך מפורסמות עד שנהיו לסטנדרט	<a href="#">jquery / Ajax</a>
ספריית CSS שעוזרת לעצב דפי אינטרנט בקצב הרבה יותר מהיר מהרגיל. השתמשתי מפני שלא רציתי לבזבז יותר מדי זמן על העיצוב (למרות שבפועל זה גם לא היה פשוט) מאחר וזה לא מטרת הפרויקט	<a href="#">Bootstrap</a>

## מדריך למשתמש

### התקנת התוכנה

התוכנה נמצאת ב [virtual enviornment](#), כלומר ההתקנה יחסית פשוטה.

- עושים git clone מאתר הפרויקט - [https://github.com/ya5huk/safe\\_cloud](https://github.com/ya5huk/safe_cloud)
- מתקינים את כל התיקיות (שלא צריך לחפש מאחר וזה virtual enviornment)
- בקובץ app.py אפשר לשנות את DB\_FILENAME בשביל שיהיה קובץ מאגר נתונים אחר.
- יש להגדיר את המשתנים הסביבתיים בעת הרצה, EMAIL\_USERNAME, EMAIL\_PASSWORD שייצגו את נתוני האימייל השולח קודים לאימות דו-שלבי
- כעת יש להריץ את הקובץ app.py

### הרצת התוכנה

בעת פתיחת האתר שירוצ ב 127.0.0.1:5000 (אלא אם הוגדר אחרת) יפתח חלון התחברות.

The image shows a dark-themed authentication form. At the top, it says "Authenticate, darling." in a large, white, serif font. Below this, there are two input fields. The first is labeled "Username or Email address" and contains the placeholder text "Either ur name or @ smthin'". The second is labeled "Password" and contains the placeholder text "I'm blind". Below the password field, there is a link that says "New to the game? Register". To the right of the register link is a red-outlined button with the word "Submit" inside.

למקרה ואין משתמש, יש ללחוץ על Register בשביל לעשות משתמש חדש. בעת לחיצה יתקבל חלון רישום משתמש חדש:

# Authenticate, darling.

Email address

Username

Password

Have an account? [Login](#)

החלונות האלו נותנים פידבק למקרה ויש הכנסת נתונים תפוסה או שגויה:

## Authenticate, darling.

Email address

Username

Password

ilan147963@gmail.com already exists!

Have an account? [Login](#)

## Authenticate, darling.

Username or Email address

Password

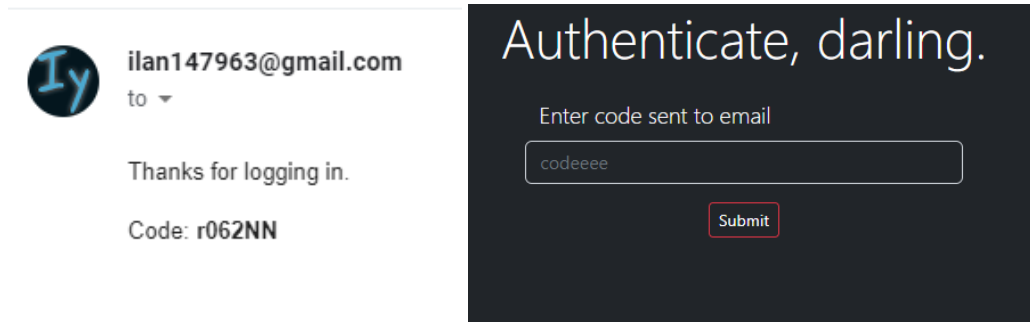
One of the credentials is incorrect.

New to the game? [Register](#)

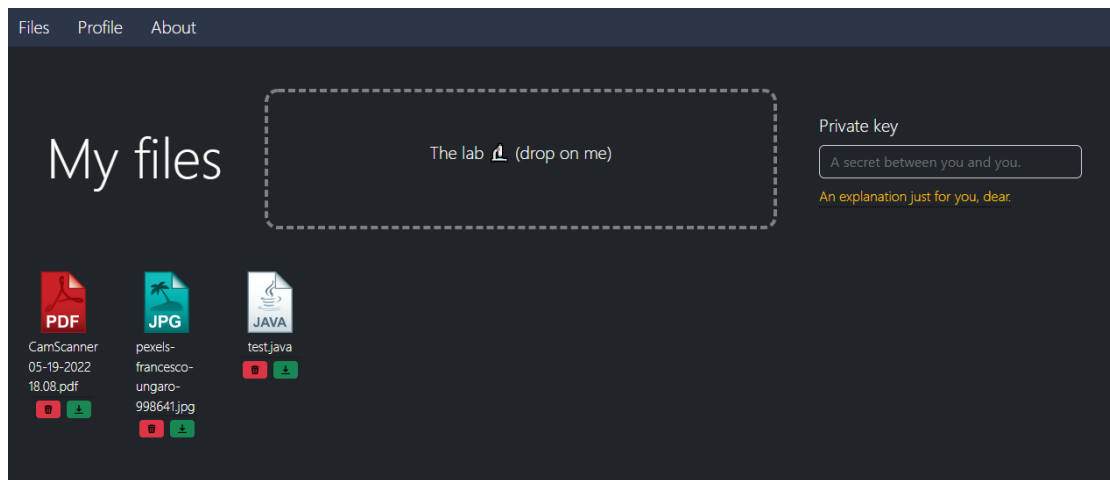
לא משנה מה תהיה הפעולה – רישום משתמש חדש או התחברות, החלון הבא שיקפוץ, לאחר לחיצה על submit הוא **חלון אימות האימייל**. החלון נועד לכך שאנשים לא סתם יכניסו אימייל שהוא לא שלהם, ושלמקרה שגילו את פרטי המשתמש שלהם, עדיין לא יוכלו להכנס למשתמש כי הפורצים לא יחזיקו בתיבת הדואר (שמאובטחת על ידי google, apple, yahoo וכו').



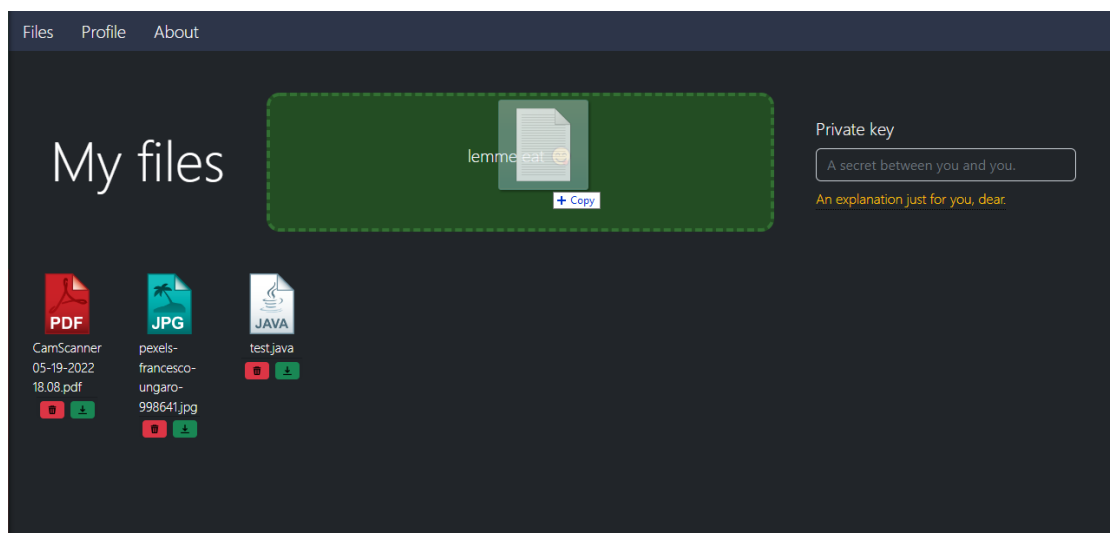
## SafeCloud



לאחר התחברות תקינה (אם תכניס קוד שגוי, תקבל פידבק על זה), מגיעים לדף העיקרי של הפרויקט והוא דף הקבצים:

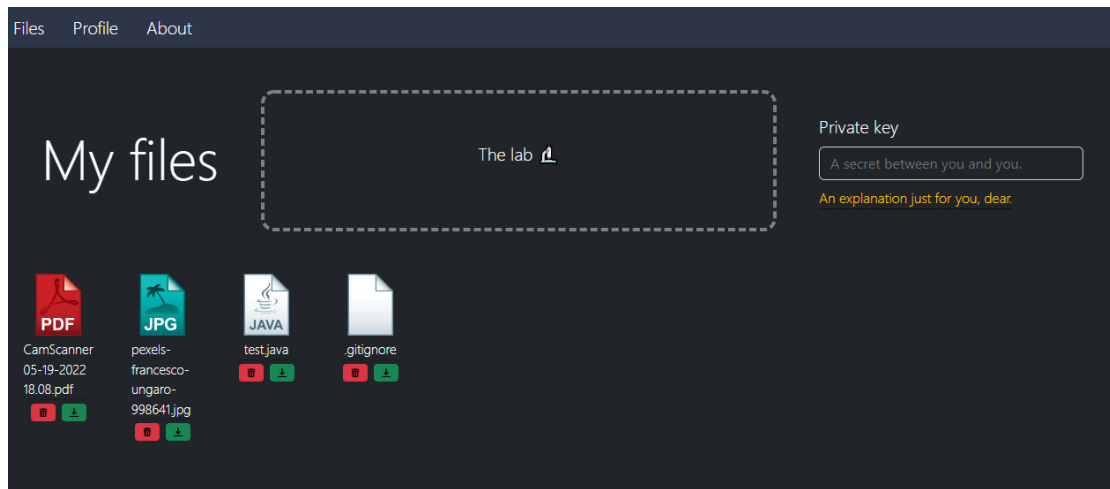


נכנסתי למשתמש שכבר יש בו קבצים, אז זה טען את הקבצים הקודמים. אם נרצה להוסיף קובץ, נגרור את הקובץ לתוך המסגרת האמצעית:

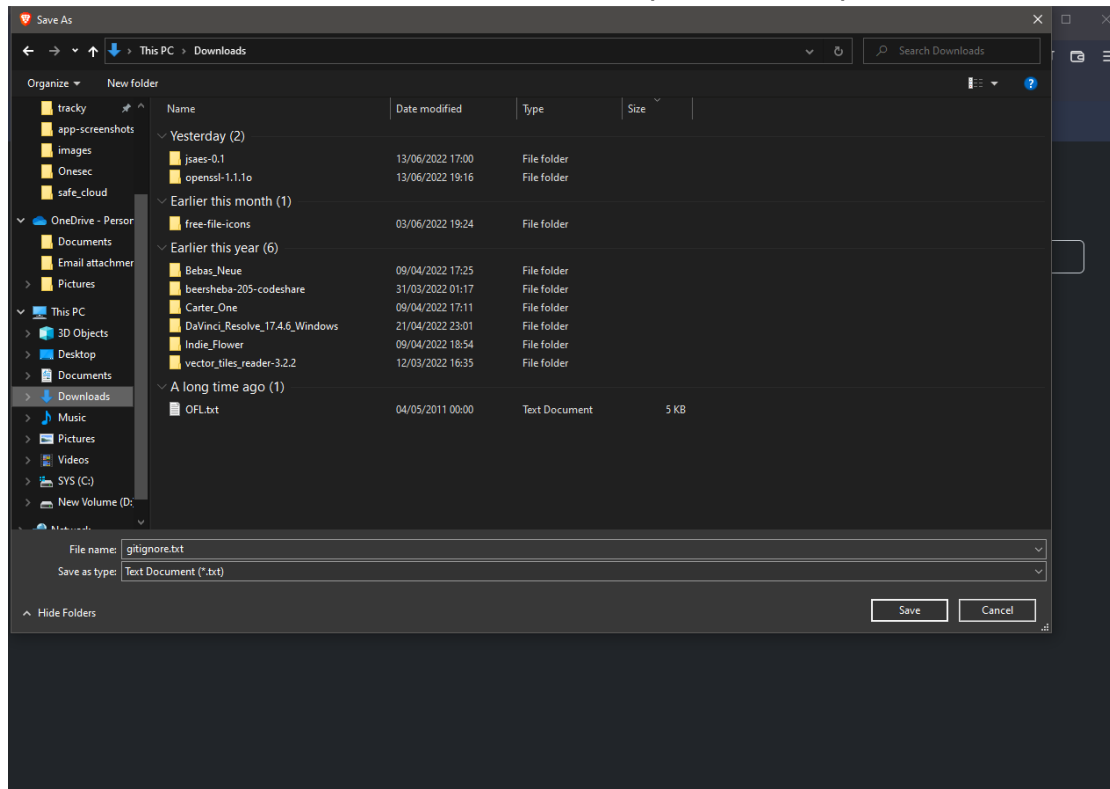


נראה שהקובץ אכן נוסף:

## SafeCloud

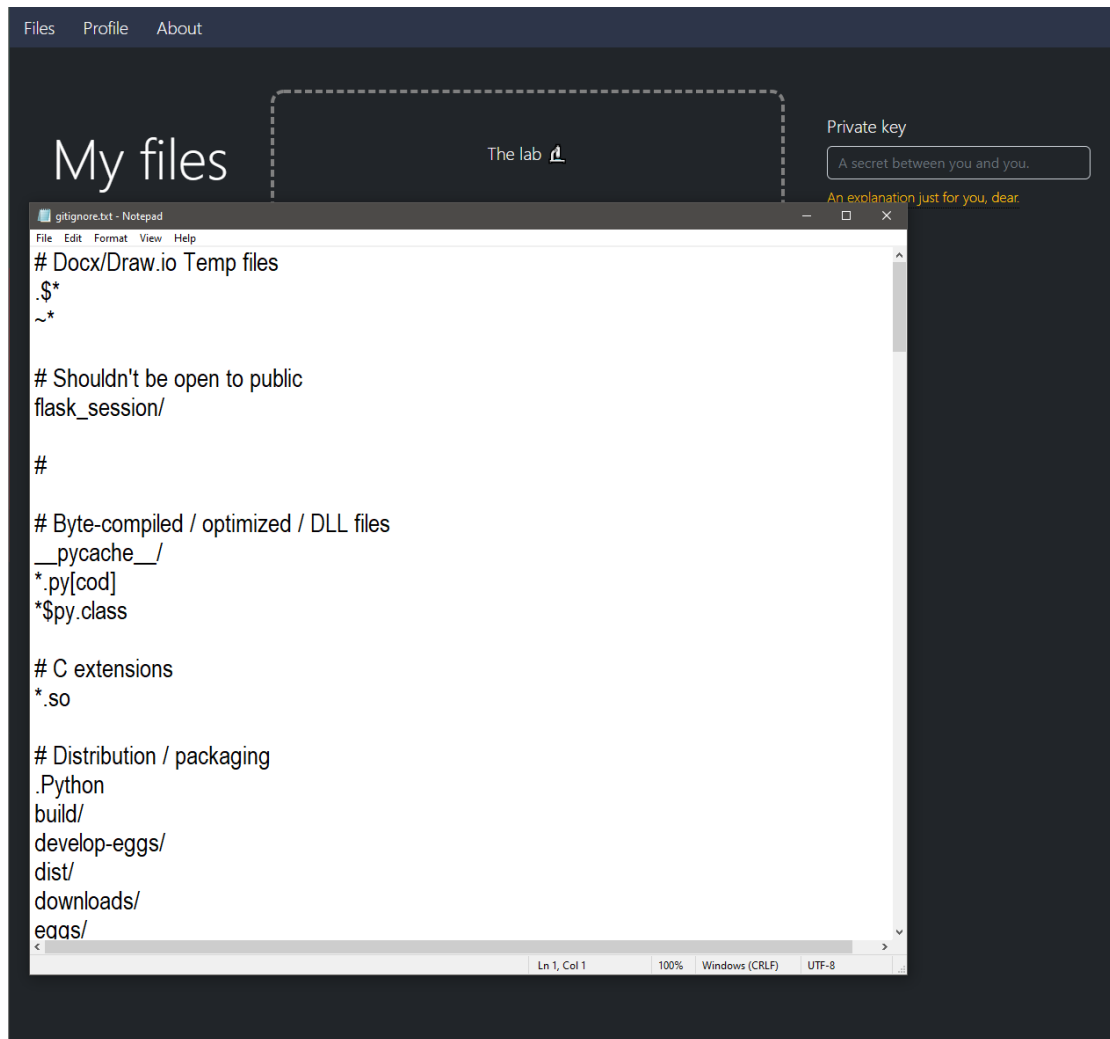


בנוסף, אם נרצה להוריד אותו, נלחץ על הלחצן הירוק ולמחיקה נלחץ על הלחצן האדום.  
בעת הורדה האתר יבקש ממנו את מיקום ההורדה:

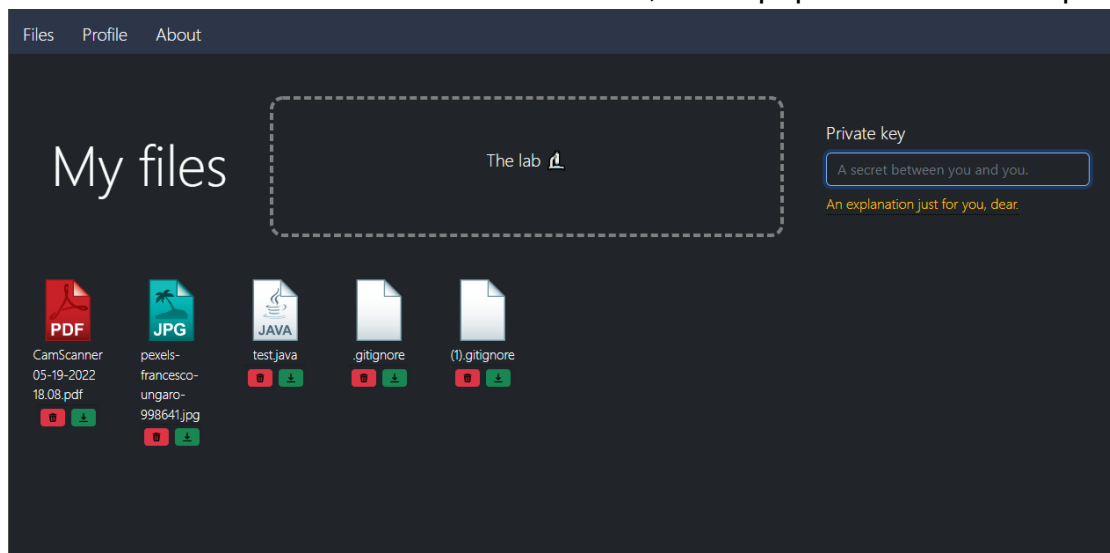


ונראה שהקובץ אכן הורד כמו שצריך.

## SafeCloud

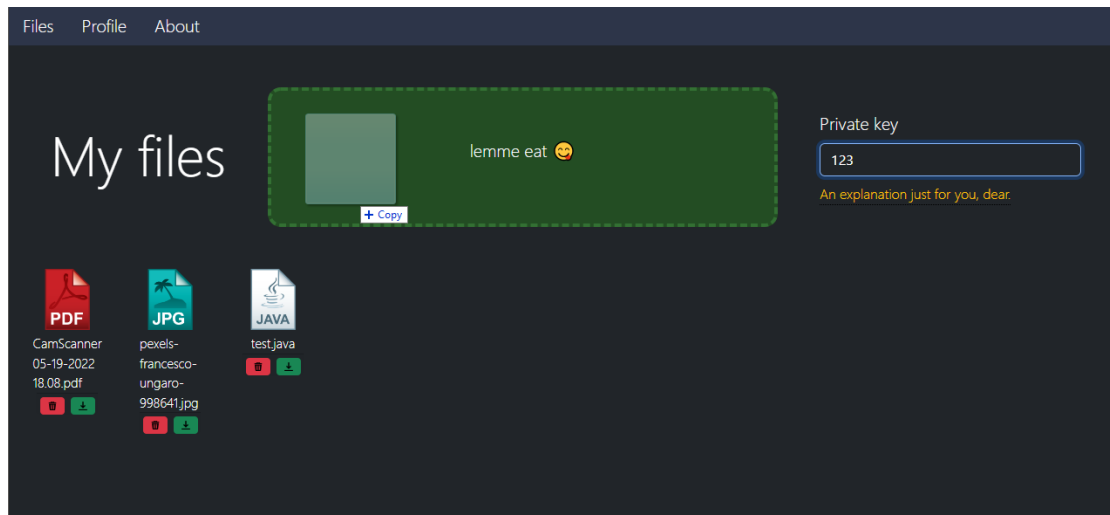


במקרה שנכניס את אותו קובץ פעמיים, נראה שהשם משתנה בהתאם:

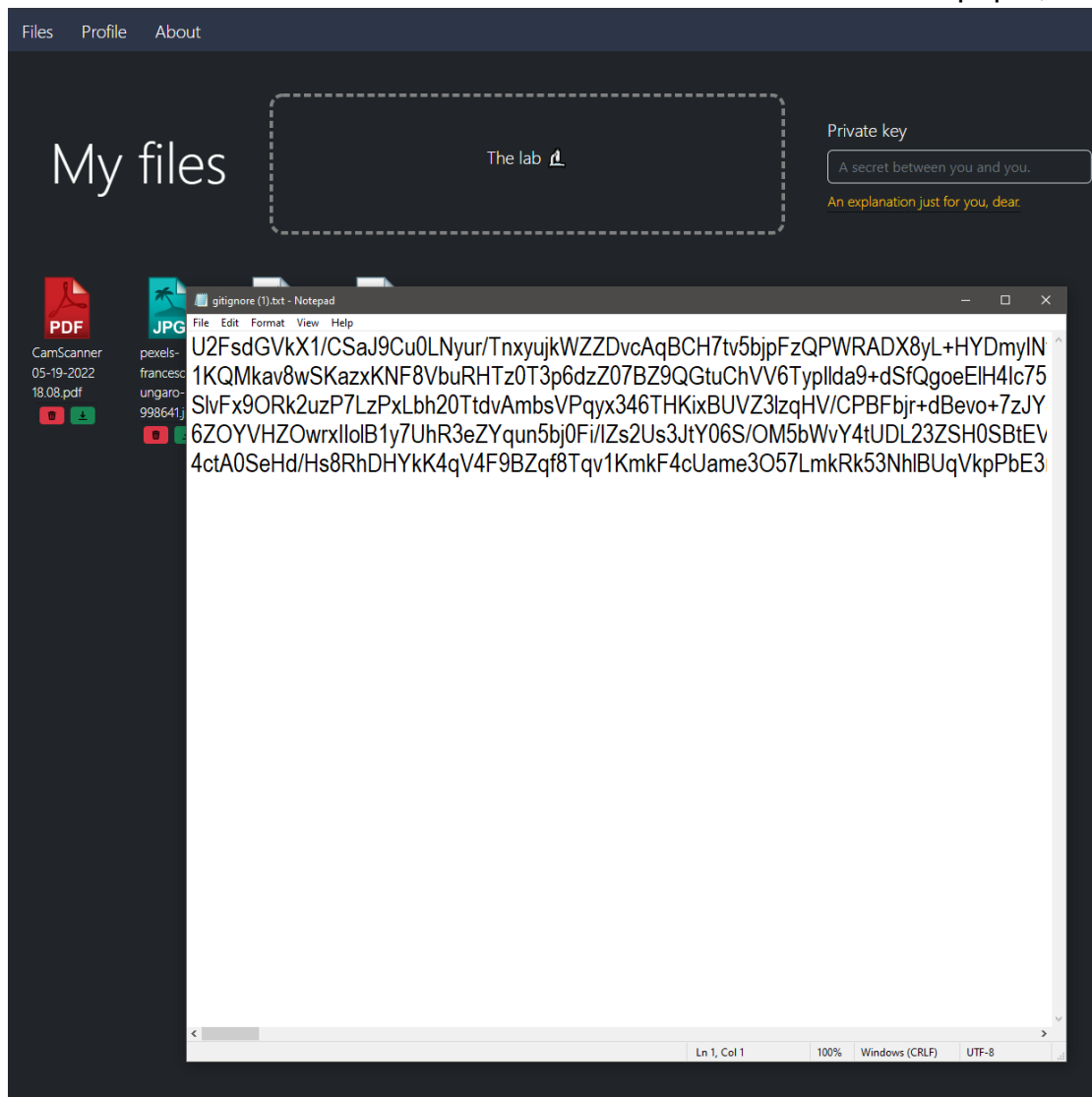


כמו כן, לדעתי הפיצ'ר הכי חזק בתוכנה הזאת, שאנחנו יכולים להצפין קובץ מצד הלקוח. כלומר, להצפין את הקובץ עוד לפני שיוצא מהמחשב שלנו – וכך אף אחד מלבדנו לא יוכל לדעת מה יש בקובץ, אלא אם כן הוא מחזיק בקוד שלנו. יש לזכור קוד הצפנה ולהוסיף קובץ כשהוא מוקלד:

## SafeCloud

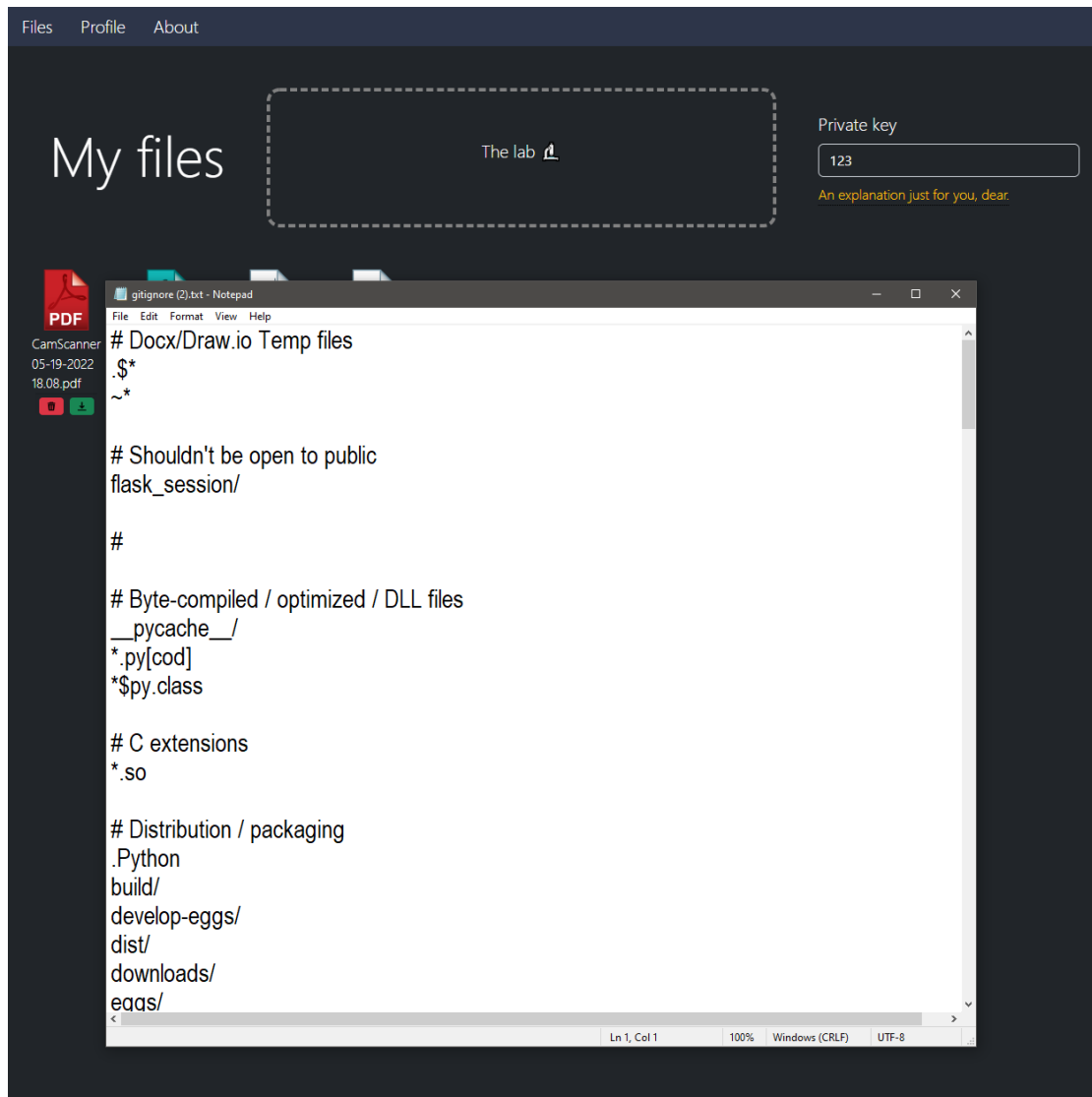


כעת, הקובץ שלנו נשלח בהצפנת AES עם המפתח "123". אם ננסה להוריד בלי המפתח:



לא נצליח לקרוא את הקובץ. אך אם נוריד את הקובץ עם המפתח:

## SafeCloud



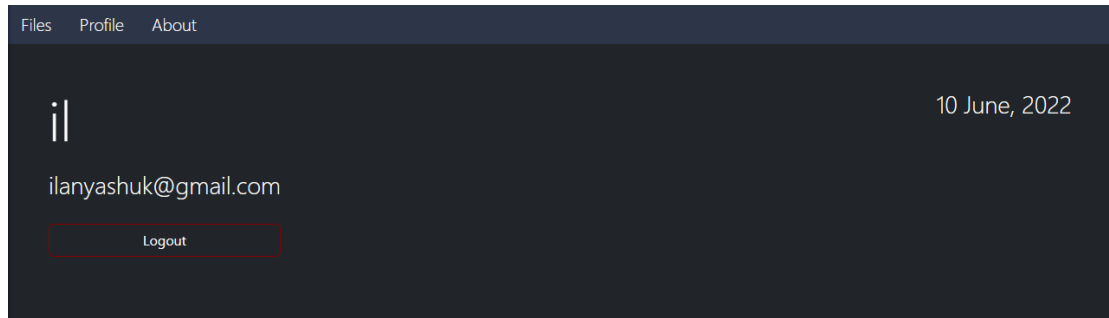
בום!

חוץ מזה, אציין שעוד פיצ'ר נחמד הוא שאפשר לגרור כמה קבצים ביחד (זזה עובד גם עם הצפנה). אם המשתמש יצטרך הסבר יותר מפורט, הוא יוכל להצביע על הטקסט הכתום:

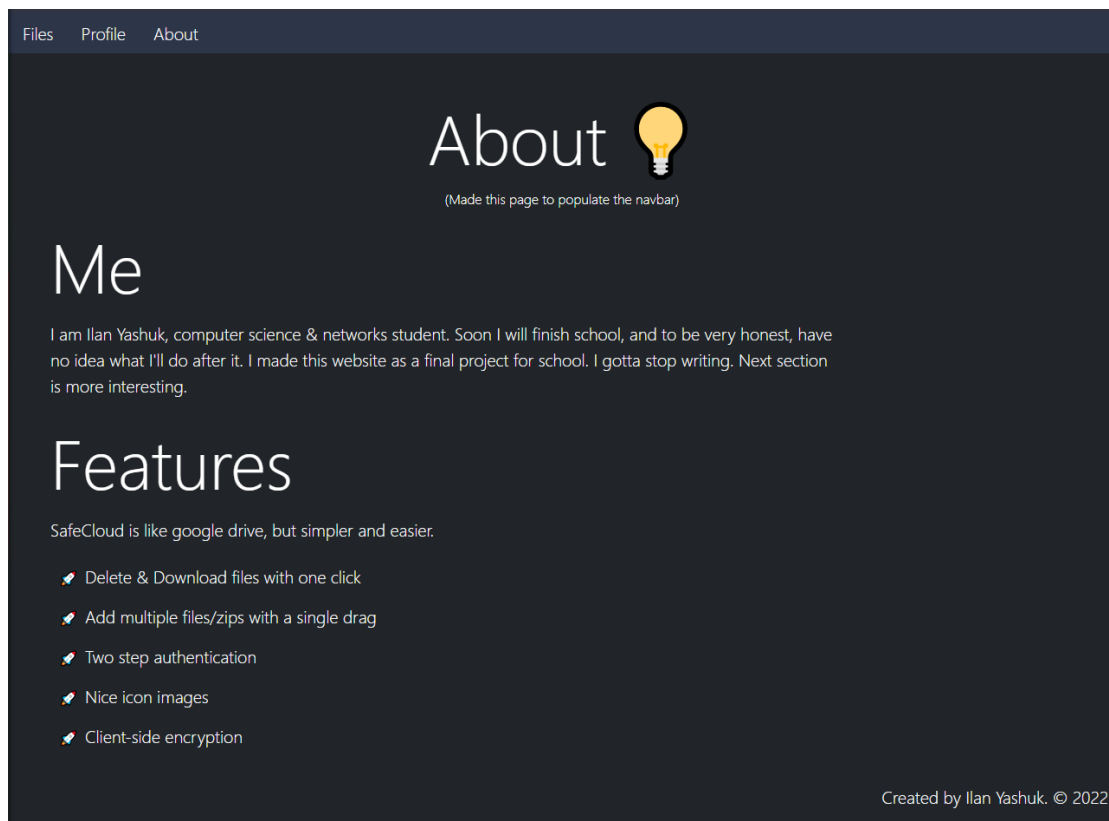


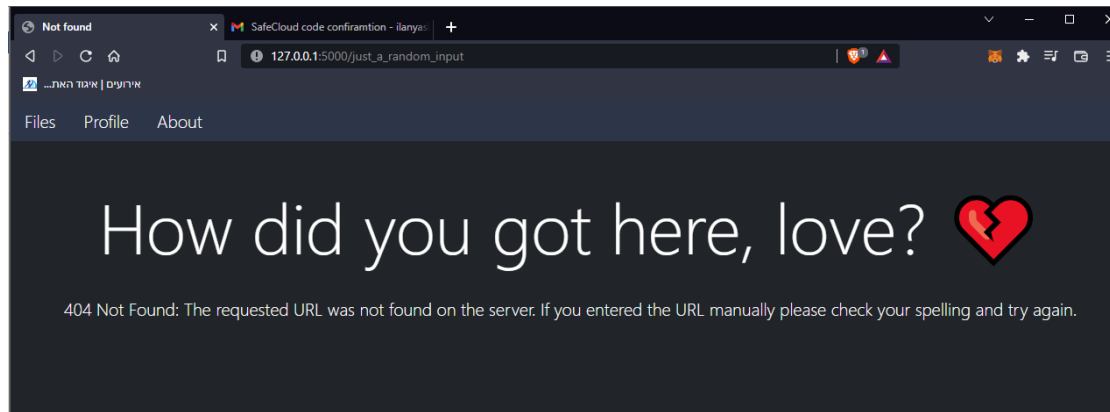
## SafeCloud

עוד עמוד מעניין הוא **עמוד הפרופיל**, שלא מכיל הרבה אבל מכיל את האפשרות להתנתק מהמשתמש. בלחיצה על הכפתור, השרת ישכח את המשתמש ואת ה"session" עמו, עד להתחברות חדשה.



שני הדפים האחרונים הם דף שמסביר על פונקציונליות האתר:





## בסיס הנתונים – sqlite3

בסך הכל יש רק שתי טבלאות במאגר הנתונים. טבלת משתמשים וטבלת קבצים:

### טבלת משתמשים (דוגמה)

Reset Filters Records: 2					
	user_id	username	email	creation_date	files
1	508179a3e96c081fcc...	ilan	ilanyashuk@gmail.com	10/06/2022, 00:13:48	d9efd378-e839-11ec...
2	18b02c6aa045a73e4...	ilan	ilan147963@gmail.co...	10/06/2022, 00:16:24	6e74353a-e839-11ec...

- מס' מזהה למשתמש, מיוצר ע"י גיבוב סיסמה, תאריך יצירה, ואימייל, מפתח ראשי
- שם משתמש, לא מוצפן
- אימייל, גם לא יכול להיות מוצפן בשביל אימות דו-שלבי
- תאריך יצירה
- קבצים, מחרוזת המכילה את כל המס' הייחודיים של הקבצים של אותו משתמש

### טבלת קבצים (דוגמה)

Reset Filters Records: 27					
	file_id	in_dir	name	content	added_date
1	e5ab8748-e76c-11ec...		bin.png	10.56 KB	08/06/2022, 23:52:23
2	ee61ef3b-e76c-11ec...		mech.doc	137.5 KB	08/06/2022, 23:52:38
3	f094b5c8-e76c-11ec...		download.jpg	2.31 KB	08/06/2022, 23:52:42
4	4983daa6-e76d-11ec...		download.jpg	2.31 KB	08/06/2022, 23:55:11
5	4c35bfb5-e76d-11ec...		Document (1).docx	13.81 KB	08/06/2022, 23:55:15
6	4db8ff1-e76d-11ec...		Document (2).docx	13.81 KB	08/06/2022, 23:55:18
7	4f8e2d2d-e76d-11ec...		Document (1) (9).docx	13.81 KB	08/06/2022, 23:55:21
8	5a8ee64a-e76d-11ec...		download (11).jpg	2.31 KB	08/06/2022, 23:55:39
9	7158a37c-e76d-11ec...		download (13).jpg	2.31 KB	08/06/2022, 23:56:18
10	a97b4918-e76d-11ec...		-.pdf	212.59 KB	08/06/2022, 23:57:52
11	aa9f95ec4-e76d-11ec...		-(17).pdf	212.59 KB	08/06/2022, 23:57:54
12	b85f9949-e76d-11ec...		-(18).pdf	212.59 KB	08/06/2022, 23:58:17
13	d98d60e3-e76d-11ec...		-(3).pdf	212.59 KB	08/06/2022, 23:59:12
14	ae821225-e776-11ec...		test.java	2.35 KB	09/06/2022, 01:02:26
15	5d1e0958-e777-11ec...		WhatsApp Image 20...	307.42 KB	09/06/2022, 01:07:19
16	2d749781-e778-11ec...		2.pdf	670.8 KB	09/06/2022, 01:13:08
17	c40166a9-e779-11ec...		database.db	3.75 MB	09/06/2022, 01:24:30
18	20f71ac3-e82f-11ec...		_blank.png	558 Bytes	09/06/2022, 23:02:45
19	22f46527-e82f-11ec...		file_icon.png	4.9 KB	09/06/2022, 23:02:48
20	2579cf2c-e82f-11ec...		Untitled document.d...	6.66 KB	09/06/2022, 23:02:53
21	15002038-e839-11ec...		Untitled document.d...	6.66 KB	10/06/2022, 00:14:00
22	16d541d5-e839-11ec...		white-stars-transpare...	43.46 KB	10/06/2022, 00:14:03

- מס' ייחודי, מבוסס זמן ולכן ייחודי לכל קובץ, מפתח ראשי
- האם בתוך תיקייה, למקרה של מימוש תיקיות בעתיד
- שם הקובץ
- תוכן הקובץ (כאן נכנסת ההצפנה אם המשתמש יבחר)
- תאריך הוספה



## מדריך למפתח

תחת כותרת זו אסביר ואראה את המוח שמאחורי הפרויקט (כמובן שאני מדבר על המחשב). אני אראה קטעי קוד, קבצים שונים ואסביר עליהם את הרקע – שאדם שירצה אולי להמשיך ולפתח את התוכנה, יוכל לעשות זאת עם ההסברים פה.

### קבצי git

גיט היא מערכת לניהול גרסאות של פרויקטים. גיט הוא מנוע ששינה את השוק לתמיד ופתח את עולם הפיתוח לכולם, משתי בחינות. ראשית, כל אחד יכול להכנס ולראות/לשנות/להוריד קוד של אדם אחר בכמה שניות. שנית, אנשים יכולים לעבוד ביחד על הפרויקטים, בלי כל הבעיות של כתיבה ביחד, פגיעה בקוד אחד של השני.

יש הרבה מה ללמוד על גיט, כיצד לעבוד נכון, כיצד לעבוד בקבוצות קטנות, כיצד לעבוד בחברות ענק וכו'. הנה הקבצים:

### .gitignore

קובץ שמורה לגיט – למה לא להתייחס (על מה "להעלים עין")

```
# Docx/Draw.io Temp files
.$*
~*

# Shouldn't be open to public
flask_session/

#

# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
```

```

parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos
# into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo
*.pot

# Django stuff:
*.log
local_settings.py
db.sqlite3
db.sqlite3-journal

# Flask stuff:
instance/

```

```

.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these files
#   since the code is
#   intended to run in multiple environments; otherwise, check them in:
#   .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include
#   Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific
#   dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies
#   that don't work, or not
#   install all needed dependencies.
#Pipfile.lock

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include
#   poetry.lock in version control.
#   This is especially recommended for binary packages to ensure
#   reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-
#   file-to-version-control
#poetry.lock

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include
#   pdm.lock in version control.

```

```
#pdm.lock
#  pdm stores project-wide configurations in .pdm.toml, but it is
#  recommended to not include it
#  in version control.
#  https://pdm.fming.dev/#use-with-ide
.pdm.toml

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and
# github.com/pdm-project/pdm
__pypackages__/_

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/
```

## SafeCloud

```
# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate
# JetBrains.gitignore that can
# be found at
https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this
# file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore
# the entire idea folder.
# .idea/
```

## קבצים סטטיים

תיקייה static מכילה תמונות וקבצים סטטיים שלא משתנים מטעינה לטעינה. התיקייה שלי לא מכילה הרבה. הנה הקבצים:

**\_blank.png**



**Bin.png**



**Download\_icon.png**



## תמונות הסימול של הקבצים

התיקיה images מכילה את התמונות הללו. אני לא אציג תמונה-תמונה כי גם ככה הספר פרויקט ארוך. לכן אציג תמונה שמרכזת את כולם:



## דפי אינטרנט (html/css/javascript)

קבצי html הם קבצי העיצוב של client והם אחראים על שני דברים. ראשית, שהאתר יראה טוב ונעים לעין המשתמש. שנית, הם אחראים על העברת המידע של המשתמש לשרת. אני קובע איך יועבר המידע ואיך לקבל אותו.

בנוסף, הרבה פעמים אתם תראו שאני משתמש במחלקות (class) שלא פירטתי עליהם. זה לא כי אין לי כוח, אלא כי אני לא עשיתי אותם. זה **Bootstrap**, שמוסיף דרך קלה ועדיין יפה לעצב את האתר שלך – לאנשים שיותר מתרכזים על הפונקציונליות ולא על העיצוב. בשביל לדעת עוד, אתם יכולים להכנס לאתר שלהם שהלינק אליו נמצא תחת הכותרת 'שימוש בטכנולוגיות...!'

יתרה מזאת, בפרויקט אני משתמש במשהו שנקרא **template inheritance**. משמעות הדבר היא כזאת: במקום ליצור את אותו קובץ html כמה פעמים, אני יוצר אחד יותר כללי, ופשוט שאר הקבצים "יירשו" ממנו (עובד דומה מאוד לתכנות מונחה עצמים). לדוגמה, אני רוצה שבכל הקבצים יהיה רקע כחול. אין לי שום סיבה לכתוב בכל קובץ שיהיה בו רקע כחול, זה יהיה מעייף, סזיפי, ובמקרים יותר מציאותיים (כשלא רק הרקע קבוע) זה פשוט לא פרקטי. אז אני יוצר לדוגמה קובץ base.html שכתוב בו שהרקע יהיה כחול. וכל שאר הקבצים ירשו ממנו. אנו נוכל לראות בקבצים פקודות שכמו extends, block שנמצאות בתוך סוגריים מסולסלות כפולות ({{}}). במקרים כאלו דעו כי זהו לא html רגיל אלא תוספת אישית של html. בכל מקום שיש את הסוגריים המסולסלות הללו, אנו מכניסים נתונים חיצוניים (מהקוד, שרת) אל תוך הhtml – דברים דינמיים שמשתנים לכל משתמש.

### Authbase.html

קובץ html שנוצר בשביל שאחרים יירשו ממנו. צובע את הרקע בצבע קרוב לכחול-אפור ומייבא את הספרייה של ajax.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFB�6D0itfPri4tjfhXaWutUpFmBp4vmVor"
      crossorigin="anonymous"
    />
    <title>Authentication</title>
  </head>
  <body class="bg-dark text-white">
    <div class="d-flex justify-content-center my-3">
```



```

    <h1 class="display-5">Authenticate, darling.</h1>
  </div>
  {% block content %} {% endblock %}
  <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"><
/script>
  </body>
</html>

```

## Login.html

קוד html שמכיל טופס למילוי נתונים להתחברות לאתר. הטופס, ברגע שלוחצים על הכפתור submit, שולח את הפרטים לbackend שמטפל במידע, ושולח את המשתמש לדף הבא (בהתאם לinput שלו)

```

{% extends "authbase.html" %} {% block content %}
<style>
  .form-control:focus {
    border-color: #891a1a;
    box-shadow: 0 0 0 0.2rem rgba(137, 72, 25, 0.25);
  }
</style>
<div class="d-flex justify-content-center">
  <form
    class="d-flex flex-column align-items-center lead gap-3"
    style="width: 30%"
    action="#"
    method="post"
  >
    <div class="row w-100 form-group gap-2">
      <label for="inputUser">Username or Email address</label>
      <input
        minlength="4"
        type="text"
        name="user_input"
        class="form-control bg-transparent text-white"
        id="inputUser"
        placeholder="Either ur name or @ smthin'"
      />
    </div>
    <div class="row w-100 form-group gap-2">
      <label for="exampleInputPassword1">Password</label>
      <input
        minlength="3"
        name="password"
        type="password"
        class="form-control bg-transparent text-white"

```

```

        id="exampleInputPassword1"
        placeholder="I'm blind"
    />
</div>
<div
    class="w-100 d-flex justify-content-between align-content-center
form-group"
>
    <div class="d-flex flex-column w-100">
        {% if err_msg %}
        <p class="form-text text-danger fs-6">{{err_msg}}</p>
        {% endif %}
        <div class="d-flex justify-content-between align-content-
center">
            <p class="text-white lead fs-6">
                New to the game?
                <a class="text-white" href="/register">Register</a>
            </p>
            <button
                type="submit"
                value="submit"
                class="lead btn-custom text-white border rounded fs-6 bg-
transparent border-danger"
            >
                Submit
            </button>
        </div>
    </div>
</div>
</form>
</div>
{% endblock %}

```

## Register.html

קובץ שמכיל טופס הרשמה לאתר. דומה מאוד לקובץ login. גם פה, בעת לחיצה על submit המשתמש יישלח לדף בהתאם להכנסותיו.

```

{% extends "authbase.html" %} {% block content %}
<style>
    .form-control:focus {
        border-color: #891a1a;
        box-shadow: 0 0 0 0.2rem rgba(137, 72, 25, 0.25);
    }
</style>
<div class="d-flex justify-content-center">
    <form

```

```

class="d-flex flex-column align-items-center lead gap-3"
style="width: 30%"
action="#"
method="post"
>
<div class="w-100 form-group gap-2">
  <label for="inputEmail">Email address</label>
  <input
    type="email"
    name="email"
    class="form-control bg-transparent text-white"
    id="inputEmail"
    placeholder="That thing with @"
  />
</div>
<div class="w-100 form-group gap-2">
  <label for="InputUsername">Username</label>
  <input
    minlength="4"
    name="username"
    type="text"
    maxlength="15"
    class="form-control bg-transparent text-white"
    id="InputUsername"
    placeholder="How u call urself?"
  />
</div>
<div class="w-100 form-group gap-2">
  <label for="InputPass">Password</label>
  <input
    minlength="4"
    name="password"
    type="password"
    class="form-control bg-transparent text-white"
    id="InputPass"
    placeholder="Won't see it anyway"
  />
</div>
<div
  class="w-100 d-flex justify-content-between align-content-center
form-group"
>
  <div class="d-flex flex-column w-100">
    {% if err_msg %}
    <p class="form-text text-danger fs-6">{{err_msg}}</p>
    {% endif %}
    <div class="d-flex justify-content-between align-content-
center">

```

```

    <p class="text-white lead fs-6">
      Have an account?
      <a class="text-white" href="/login">Login</a>
    </p>
    <button
      type="submit"
      value="submit"
      class="lead btn-custom text-white border rounded fs-6 bg-
transparent border-danger"
    >
      Submit
    </button>
  </div>
</div>
</div>
</form>
</div>
{% endblock %}

```

### Code\_enter.html

קובץ שסה"כ מכיל את המקום להכניס קוד שנכנס לאימייל (לאחר שליחתו על ידי השרת בתהליך אימות דו-שלבי). בעת הכנסת הקוד, השרת יבדוק האם הקוד נכון, הוא יכוון אותו חזרה לחלון ההתחברות במקרה של שגיאה ולחלון הראשי – הקבצים במקרה של הצלחה.

```

{% extends "authbase.html" %} {% block content %}
<div class="d-flex justify-content-center">
  <form
    class="d-flex flex-column align-items-center lead gap-3"
    style="width: 30%"
    action="#"
    method="post"
  >
    <div class="row w-100 form-group gap-2">
      <label for="inputCode">Enter code sent to email</label>
      <input
        type="text"
        name="code_input"
        class="form-control bg-transparent text-white"
        id="inputCode"
        placeholder="codeeeee"
      />
    </div>
    <button
      type="submit"
      value="submit"
    >

```

```

        class="btn btn-sm text-white border-danger"
    >
        Submit
    </button>
</form>
</div>
{% endblock %}

```

## General.html

קובץ שיש לו שימוש יחיד והוא שיירשו ממנו. הוא משנה את צבע הרקע ומוסיף למעלה תפריט (navbar) שאחראי לקפוץ בין עמודים שונים, לשביעות רצון המשתמש. בנוסף, הוא מייבא את הספרייה של jquery.

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6D0itfPri4tjfhXaWutUpFmBp4vmVor"
      crossorigin="anonymous"
    />
    <title>{% block title %} {% endblock %}</title>
    <style>
      .navcolors {
        background-color: rgb(45, 53, 73);
      }
    </style>
  </head>
  <body class="bg-dark">
    <ul class="nav text-white navcolors lead">
      <li class="nav-item">
        <a class="nav-link text-white" href="/files">Files</a>
      </li>
      <li class="nav-item">
        <a class="nav-link text-white" href="/profile">Profile</a>
      </li>
      <li class="nav-item">
        <a class="nav-link text-white" href="/about">About</a>
      </li>
    </ul>
  </body>
</html>

```

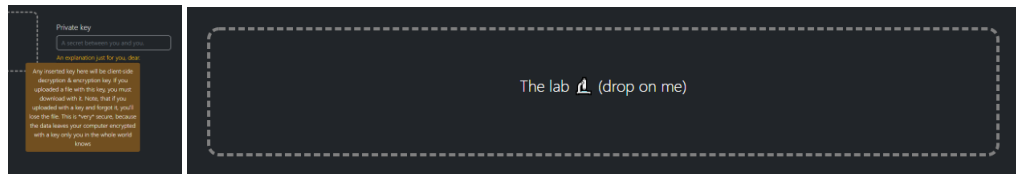
```

</ul>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"><
/script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-
js/4.1.1/crypto-js.min.js"></script>
{% block content %} {% endblock %}
</body>
</html>

```

## Files.html

הקובץ הראשי של הפרויקט (מבחינת ראיית המשתמש). אורך הקובץ כזה ארוך משום שהוא מטפל בהרבה דברים. ראשית, הוא דורש הרבה עיצוב CSS מפני שהוא יוצר תיבה מרחפת ותיבה שאפשר לזרוק עליה קבצים:



שנית, הוא דורש יחסית הרבה קוד HTML משום שיש הרבה אלמנטים על המסך. במצב התחלתי, יש את התיבה, כותרת ועוד תיבה להכנסת קוד. בנוסף, יש "קובץ מוסתר" שהוא לא קובץ אמיתי אלא קיים בשביל לשמש לנו כמעין תבנית לקבצים אחרים. כלומר בתוספת קובץ, אני מעתיק אותו ומשנה רק את התמונה, שמות זיהוי, שם קובץ וכד'.

שלישית, הוא מכיל המון קוד פונקציונליות של JavaScript. וזאת משום שהוא צריך פונקציונליות לכפתור הורדה ומחיקה. הוא צריך להצפין ולפענח מצד הלקוח את הקבצים (אם הלקוח ירצה בכך) והוא צריך לטפל בתיבת "זריקת הקבצים".

ניתן לפצל את הקבצים הללו לHTML, CSS, JavaScript. אבל לדעתי הוא לא ארוך ברמה מזוועה והוא יותר מסודר ככה. אני סבור שעדיף ככה מאשר לקפוץ בין 3 קבצים ולנסות להבין מה כל דבר עושה.

להלן הקובץ:

```

{% extends "general.html" %} {% block title %} My files {% endblock %}
{% block
content %}

<style>
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

```

```

main {
  display: flex;
  flex-direction: column;
}

header {
  margin-top: 1rem;
}

/* For file dropping */
.droparea {
  width: 384px;
  max-width: 100%;
  height: 160px;
  border: 4px dashed grey;
  border-radius: 15px;
}

.selectarea {
  background-color: #234d23;
  border-color: #337033
}


/* Only for encryption explanation */
/* https://sebastian.com/html-hover-text/ */
.hovertext {
  position: relative;
  border-bottom: 1px dotted black;
}

.hovertext:before {
  content: attr(data-hover);
  visibility: hidden;
  opacity: 0;
  width: 300px;
  background-color: #714f20;
  color: #fff;
  text-align: center;
  border-radius: 5px;
  padding: 10px 5px;
  transition: opacity 1s ease-in-out;

  position: absolute;
  z-index: 1;
  right: 0;
  top: 110%;
}

```

```

    .hovertext:hover:before {
      opacity: 1;
      visibility: visible;
    }
  </style>
  <div class="d-flex flex-wrap px-5 align-items-center justify-content-between my-5">
    <h1 class="display-3 text-white">My files</h1>
    <section class="d-flex flex-grow-1 mx-5 align-items-center justify-content-center droparea flex-column">
      <p class="text-white lead user-select-none" id="section-text">
        The lab  (drop on me)
      </p>
    </section>
    <div class="d-flex flex-column text-white lead">

      <div class="d-flex flex-column align-items-start form-group gap-2">
        <label for="inputUser">Private key</label>

        <input type="text" name="key_input" class="form-control bg-transparent lead text-white" style="width: 300px"
          id="inputSecretKey" placeholder="A secret between you and you."
        />

        <small class="text-warning fs-6 hovertext"
          data-hover="Any inserted key here will be client-side
          decryption & encryption key. If you uploaded a file with this key, you
          must download with it. Note, that if you uploaded with a key and forgot
          it, you'll lose the file. This is *very* secure, because the data
          leaves your computer encrypted with a key only you in the whole world
          knows">
          An explanation just for you, dear.</small>
        </div>

      </div>
    </div>
    <ul class="d-flex flex-wrap gap-5" id="files-list">
      <!-- Example li -->
      <li style="width: 70px" id="filetemplate" hidden>
        <div class="d-flex align-items-center card bg-transparent border-0">
          
          <div class="card-body p-0 pt-1">
            <p class="card-text text-white lead overflow-auto" style="font-size: 15px" id="filetemplateText">
              filename.txt
            </p>
          </div>
        </div>
      </li>
    </ul>
  </div>

```



```

        </div>
        <div class="card-footer d-flex justify-content-around text-white
pt-1 p-0 m-0">
            <button class="d-flex align-items-center justify-content-center
btn btn-danger lead btn-sm" style="width: 40%"
                id="delete-button-template" onclick="deleteFile(this.id)">
                
            </button>
            <button class="d-flex align-items-center justify-content-center
btn btn-success lead btn-sm" style="width: 40%"
                id="download-button-template"
onclick="downloadFile(this.id)">
                
            </button>
        </div>
    </div>
</li>
{% for fd in files_data %}
<!-- ID'S FOR OBJECTS ARE VERY IMPORTANT -->
<li style="width: 70px" id="{{fd["name"]}}-list-item">
    <div class="d-flex align-items-center card bg-transparent border-
0">
        <img id="{{fd["name"]}}-file-icon" class="card-img-top"
src={{fd["icon_data"]}} />
        <div class="card-body p-0 pt-1">
            <p class="card-text text-white lead overflow-auto" style="font-
size: 15px" id="{{fd["name"]}}-file-body">
                {{fd["name"]}}
            </p>
        </div>
        <div class="card-footer d-flex justify-content-around text-white
pt-1 p-0 m-0">
            <button class="d-flex align-items-center justify-content-center
btn btn-danger lead btn-sm" style="width: 40%"
                id="{{fd["name"]}}-delete-button"
onclick="deleteFile(this.id)">
                
            </button>
            <button class="d-flex align-items-center justify-content-center
btn btn-success lead btn-sm" style="width: 40%"
                id="{{fd["name"]}}-download-button"
onclick="downloadFile(this.id)">
                
            </button>
        </div>
    </div>
</li>
{% endfor %}

```

```

</ul>
<script>

let PrivateKeyInput = document.getElementById('inputSecretKey')

// thanks to https://www.youtube.com/watch?v=9Xh_ZpFkROI
const initApp = () => {
  const droparea = document.querySelector(".droparea");

  // Color border
  const active = () => {
    droparea.classList.add("selectarea");
    document.getElementById("section-text").innerHTML = "lemme eat
😊";
  };

  const inactive = () => {
    droparea.classList.remove("selectarea");
    document.getElementById("section-text").innerHTML =
      "The lab ↓";
  };

  // Prevent default reload for all file-related events
  const prevents = (e) => e.preventDefault();
  ["dragenter", "dragover", "dragleave", "drop"].forEach((evtName) =>
  {
    droparea.addEventListener(evtName, prevents);
  });

  ["dragenter", "dragover"].forEach((evtName) => {
    droparea.addEventListener(evtName, active);
  });

  ["dragleave", "drop"].forEach((evtName) => {
    droparea.addEventListener(evtName, inactive);
  });

  droparea.addEventListener("drop", handleDrop);
};

document.addEventListener("DOMContentLoaded", initApp);

const readFile = (file) => {
  return new Promise((resolve, reject) => {
    let fr = new FileReader()
    fr.onload = () => {
      resolve({filename: file.name, content: fr.result})
    }
  })
}

```

```

    fr.onerror = reject;
    fr.readAsArrayBuffer(file)
  })
}

const addFileIconItem = (iconData) => {

  const filesList = document.getElementById("files-list");
  // Copy hidden list item into li (prepared for file visuals)
  const li = document
    .getElementById("filetemplate")
    .cloneNode(true);
  li.hidden = false;
  li.id = `${iconData.filename}-list-item`;

  // Customizing objects -> filenames, fileicons, ext...
  const liParagraph = li.querySelectorAll(
    '[id="filetemplateText"]'
  )[0];
  liParagraph.innerHTML = iconData.filename;
  liParagraph.id = `${iconData.filename}-file-body`;

  const liImage = li.querySelectorAll('[id="filetemplate-img"]')[0];
  liImage.src = `data:image/png;base64, ${iconData.data}`;
  liImage.id = `${iconData.filename}-file-icon`;

  // To enable frontend know what file he needs for this operations
  const liDownloadButton = li.querySelectorAll(
    '[id="download-button-template"]'
  )[0];
  liDownloadButton.id = `${iconData.filename}-download-button`;
  const liDeleteButton = li.querySelectorAll(
    '[id="delete-button-template"]'
  )[0];
  liDeleteButton.id = `${iconData.filename}-delete-button`;

  // li.textContent = returnedFilename;
  filesList.appendChild(li);
}

const handleDrop = (e) => {
  // To send multiple files we first drop the files,
  // Then read async, wait until all are read and then
  // send it to server, server parses it and sends back
  // icon images

```

```

const pk = PrivateKeyInput.value;
const dt = e.dataTransfer;
const files = [...dt.files];
let read_files = []

let desiredLength = files.length;

for (const file of files) {
  read_files.push(readFile(file))
}

Promise.all(read_files).then(files => {
  let formData = new FormData();
  let counter = 0
  for (const file of files) {
    if (pk !== '') {
      // AES client-side encryption is needed
      let wordArray = CryptoJS.lib.WordArray.create(file.content);
      let encrypted = CryptoJS.AES.encrypt(wordArray,
pk).toString();
      formData.append(`files-${counter}`, new Blob([encrypted]),
file.filename);
    }
    else {

      formData.append(`files-${counter}`, new Blob([file.content]),
file.filename);
    }

    counter++
  }
  formData.append('length', files.length)
  return formData
}).then((fd) => {

  // formData is ready
  // Sending arr of files

$(document).ready(() => {
  $.ajax({
    url: "/files",
    type: "POST",
    data: fd,
    processData: false,
    contentType: false,
    success: (icons_data) => {
      icons_data.forEach(icd => {

```

```

        addFileIconItem(icd)
    })
}
})
})
}

// A code to create temp a link to download file
// https://stackoverflow.com/a/36899900/12432147
function saveData(name, type, data) {
    // for new ie
    if (data !== null && navigator.msSaveBlob)
        return navigator.msSaveBlob(new Blob([data], { type: type }),
name);

    var a = $("<a style='display: none;' />");
    var url = window.URL.createObjectURL(new Blob([data], { type: type
})));
    a.attr("href", url);
    a.attr("download", name);
    $("body").append(a);
    a[0].click();
    window.URL.revokeObjectURL(url);
    a.remove();
}

// Big thanks to https://stackoverflow.com/a/60550134/12432147
// for explaining the AES dec & enc
// word array is to use 32/64 bit instead of 8-bit arrays
// this is for cryptography optimization
const convertWordArrayToUint8Array = (wordArray) => {
    var arrayOfWords = wordArray.hasOwnProperty("words") ?
wordArray.words : [];
    var length = wordArray.hasOwnProperty("sigBytes") ?
wordArray.sigBytes : arrayOfWords.length * 4;
    var uInt8Array = new Uint8Array(length), index=0, word, i;
    for (i=0; i<length; i++) {
        word = arrayOfWords[i];
        uInt8Array[index++] = word >> 24;
        uInt8Array[index++] = (word >> 16) & 0xff;
        uInt8Array[index++] = (word >> 8) & 0xff;
        uInt8Array[index++] = word & 0xff;
    }
    return uInt8Array;
}

const downloadFile = (id) => {

```

```

    // Finding filename & extension from set id
    const filename = id.slice(0, id.length - "-download-
button".length);
    const sliced = filename.split(".");
    const extension = sliced[sliced.length - 1];
    const pk = PrivateKeyInput.value;
    $.ajax({
      url: `/files/download/${filename}`,
      type: "GET",
      success: (res) => {
        const byteCharacters = atob(res);
        // Convert base64 to blob
        if (pk !== '') {
          let decrypted = CryptoJS.AES.decrypt(byteCharacters, pk);
          let typedArray = convertWordArrayToUint8Array(decrypted);
          saveData(filename, `data:${extension}`, new
Blob([typedArray]));
          return false
        }

        const byteNumbers = new Array(byteCharacters.length);
        for (let i = 0; i < byteCharacters.length; i++) {
          byteNumbers[i] = byteCharacters.charCodeAt(i);
        }
        const byteArray = new Uint8Array(byteNumbers);
        const blob = new Blob([byteArray], { type: extension });
        saveData(filename, `data:${extension}`, blob);
      },
    });
  };

  const deleteFile = (id) => {
    filename = id.slice(0, id.length - "-delete-button".length);
    $.ajax({
      url: `/files/delete/${filename}`,
      type: "GET",
      success: (res) => {
        // Deleting visually (After serverside deletion already done)
        document.getElementById(`${filename}-list-item`).remove();
      },
      error: (err) => {
        console.log(`Error deleting: ${err}`);
      },
    });
  };
}
</script>
{% endblock %}

```

## Profile.html

הקובץ של הפרופיל של המשתמש. לא עושה הרבה חוץ מלהציג את נתוני המשתמש וכפתור התנתקות:

```
{% extends "general.html" %} {% block title %} Profile {% endblock %}
{% block
content %}
<style>
  .btn-hover-special {
    border-color: darkred;
  }
  .btn-hover-special:hover {
    background-color: darkred;
    border-color: transparent;
  }
</style>
<div class="d-flex justify-content-between pt-5 ps-5">
  <div class="d-flex flex-column gap-2">
    <h1 class="display-3 text-white">{{username}}</h1>
    <p class="lead fs-3 text-white">{{email}}</p>
    <a class="btn btn-hover-special text-white lead"
href="/logout">Logout</a>
  </div>
  <p class="lead fs-3 text-white pe-5">{{creation_date}}</p>
</div>

{% endblock %}
```

## About.html

קובץ שמראה את הפונקציונליות של האתר. לא מכיל שום פונקציונליות והוא בעיקרו קובץ html בלבד:

```
{% extends "general.html" %} {% block title %} About {% endblock %} {%
block
content %}
<div
  class="d-flex flex-column align-items-center text-center pt-5 gap-1
text-white"
>
  <h1 class="display-1">About !</h1>
  <p class="lead fs-6">(Made this page to populate the navbar)</p>
</div>
<div
  class="d-flex flex-column w-75 align-items-start text-center ps-5
gap-2 text-white"
>
```

```

>
<h1 class="display-1">Me</h1>
<p class="text-start lead">
  I am Ilan Yashuk, computer science & networks student. Soon I will
finish
  school, and to be very honest, have no idea what I'll do after it.
I made
  this website as a final project for school. I gotta stop writing.
Next
  section is more interesting.
</p>
<h1 class="display-1">Features</h1>
<p class="text-start lead">
  SafeCloud is like google drive, but simpler and easier.<br />
</p>
<ul class="text-start" style="list-style-type: '🔪'">
  <li>
    <p class="text-white ps-2 fs-5 lead">
      Delete & Download files with one click
    </p>
  </li>
  <li>
    <p class="text-white ps-2 fs-5 lead">
      Add multiple files/zips with a single drag
    </p>
  </li>
  <li>
    <p class="text-white ps-2 fs-5 lead">Two step authentication</p>
  </li>
  <li>
    <p class="text-white ps-2 fs-5 lead">Nice icon images</p>
  </li>
  <li>
    <p class="text-white ps-2 fs-5 lead">Client-side encryption</p>
  </li>
</ul>
</div>

{% endblock %}

```

## 404.html

קובץ שמטרתו להראות שגיאת 404 בכל פעם שיש שגיאה כזו (שגיאה שהמשתמש הגיע לכתובת לא ידועה)

```

{% extends "general.html" %} {% block title %} Not found {% endblock %}
{% block

```



```
content %}  
<div  
  class="d-flex flex-column align-items-center text-center pt-5 gap-4  
text-white"  
>  
  <h1 class="display-1">How did you got here, love? ❤️</h1>  
  <p class="lead">{{err_msg}}</p>  
</div>  
{% endblock %}
```

## קבצי קוד שרת

## DBCommands.py

מהשם ניתן להבין שהוא אחראי על כל האינטגרציה עם מבנה הנתונים. הקוד שמטפל במאגר הנתונים משתמש במנעול אחד לכל המאגר, שלא ייגשו אליו שני משתמשים בו זמנית, ויפגעו במידע. לא אפרט בקובץ זה אך [מנעול](#) משומש בשביל לשמור על משאב משותף לכמה תהליכים שרצים בו זמנית. לדוגמה, אם משתמש א' ומשתמש ב' יוספו למערכת בקירוב באותו הזמן, המערכת לא תדע את מי מהם לשמור. לכן, אחד מהמשתמשים ינעל מנעול על מאגר הנתונים, ולאחר שיסיים להשתמש בו – ישחרר את המנעול וייתן למשתמש אחר לגשת.

```
from datetime import datetime
import sqlite3
import threading
from DBObjects import DBFile, DBUser

class DBCommands:
    def __init__(self, db_file_path: str):
        self.db_file_path = db_file_path
        self.db_con = sqlite3.connect(db_file_path,
check_same_thread=False)
        self.cur = self.db_con.cursor() # Will be used to operate in db
        self.lock = threading.Lock()

    def create_tables(self):
        self.cur.execute('''CREATE TABLE IF NOT EXISTS files
        (file_id text primary key, in_dir integer, name text, content
blob, added_date text)''')

        self.cur.execute('''CREATE TABLE IF NOT EXISTS users
        (user_id text primary key, username text, email text,
creation_date text, files text, encryption_key text)''')

        self.cur.execute('''CREATE TABLE IF NOT EXISTS dirs
        (dir_id text primary key, name text, files text)''')

        self.db_con.commit()

    # Files

    def add_file(self, file: DBFile):
        try:
            self.lock.acquire(True)
            self.cur.execute(f'''INSERT INTO files(file_id, in_dir,
name, content, added_date)
            values (?, ?, ?, ?, ?)''',
```

```

        (file.file_id, '1' if file.in_dir else '0', file.name,
file.content, str(file.added_date.strftime("%d/%m/%Y, %H:%M:%S"))))
        self.db_con.commit()
    finally:
        self.lock.release()

    def remove_file_by_name(self, filename: str):
        self.cur.execute(f'''DELETE FROM files WHERE name=:filename''',
{'filename': filename})
        self.db_con.commit()

    def remove_file_by_id(self, file_id: str):
        self.cur.execute(f'''DELETE FROM files WHERE
file_id=:file_id''', {'file_id': file_id})
        self.db_con.commit()

    def get_file_name(self, file_id: str):
        self.cur.execute(f'''SELECT name FROM files WHERE
file_id=:file_id''', {'file_id': file_id})
        ans = self.cur.fetchall()
        if len(ans) == 0:
            return None

        name = ans[0][0] # If more than a file popps up, that is a
problem
        return name # Bytes of the file

    def get_file_content(self, file_id: str):
        self.cur.execute(f'''SELECT content FROM files WHERE
file_id=:file_id''', {'file_id': file_id})
        ans = self.cur.fetchall()
        if len(ans) == 0:
            return None

        content = ans[0][0] # If more than a file pops up, that is a
problem
        return content # Bytes of the file

# Users

    def add_user(self, usr: DBUser):
        self.cur.execute(f'''INSERT INTO users(user_id, username,
email, creation_date, files)
        values (?, ?, ?, ?, ?, ?)''',
        (usr.user_id,usr.username, usr.email,
str(usr.creation_date.strftime("%d/%m/%Y, %H:%M:%S")),
', '.join(usr.files)), '1' if usr.encryption_key else '0')
        self.db_con.commit()

```

```

def remove_user_by_username(self, username: str):
    self.cur.execute(f'''DELETE FROM users WHERE
username=:username''', {'username': username})
    self.db_con.commit()

def remove_user_by_id(self, id: str):
    self.cur.execute(f'''DELETE FROM users WHERE user_id=:id''',
{'id': id})
    self.db_con.commit()

def get_user_details_by_value(self, value: str, input_type: str):
    if input_type not in ['user_id', 'username', 'email']:
        # You can find user only with those value types
        return None

    # command based on given type
    fetch_command = f'''SELECT * FROM users WHERE
{input_type}=:{input_type}'''
    self.cur.execute(fetch_command, {input_type: value})
    ans = self.cur.fetchall()
    if len(ans) == 0:
        return None

    user = ans[0] # Should be only one user
    userid, username, email, creation_date, files = user #
Splitting the tuple
    return DBUser(userid, username, email,
datetime.strptime(creation_date, "%d/%m/%Y, %H:%M:%S"),
files.split(','))

def check_email_existance(self, email: str):
    self.cur.execute(f'''SELECT * FROM users WHERE email=:email''',
{'email': email})
    ans = self.cur.fetchall()
    if len(ans) == 0: # empty list = no accounts
        return False
    return True

def check_username_existance(self, username: str):
    self.cur.execute(f'''SELECT * FROM users WHERE
username=:username''', {'username': username})
    ans = self.cur.fetchall()
    if len(ans) == 0: # empty list = no accounts
        return False
    return True

def get_user_files(self, user_id: str): # file_ids

```

```

    try:
        self.lock.acquire(True)
        self.cur.execute(f'''SELECT files FROM users WHERE
user_id=:user_id''', {'user_id': user_id})
        ans = self.cur.fetchall()
        if len(ans) == 0: # empty list = no accounts
            return []

        if ans[0][0] == '':
            return []
        else:
            return ans[0][0].split(',')
    finally:
        self.lock.release()

    def change_user_file_ids(self, user_id: str, file_id: str,
change_type: str):
        if change_type not in ['remove', 'add']:
            return None

        try:
            self.lock.acquire(True)
            self.cur.execute(f'''SELECT files FROM users WHERE
user_id=:user_id''', {'user_id': user_id})
            ans = self.cur.fetchall()

            if ans[0][0] == '':
                self.cur.execute(f'''UPDATE users SET
files=:file_id''', {'file_id': file_id})

            else:
                files = ans[0][0].split(',')
                if change_type == 'add':
                    files.append(file_id)
                else:
                    files.remove(file_id)
                new_files = ','.join(files)
                self.cur.execute(f'''UPDATE users SET files=:new_files
WHERE user_id=:user_id''',
                    {'new_files': new_files, 'user_id': user_id})

            self.db_con.commit()
        finally:
            self.lock.release()

    # Runs on deconstruction of object
    def __del__(self):

```

```

        print('Closing db connection')
        self.cur.close()

    def __repr__(self) -> str:
        return f'db api for -> {self.db_file_path}'

if __name__ == "__main__":
    db = DBCommands('./database.db')
    db.create_tables()
    print(db.check_username_existance('r'))

```

### DBObjects.py

בד"כ כשיוצרים מאגר נתונים, יוצרים אובייקטים שמטרתם לייצג את הטבלאות במאגר. כלומר אני יצרתי אובייקט למשתמש ואובייקט לקובץ, בשביל שאוכל לעבוד עם הקבצים שאני מכניס למאגר, בצורה תכנותית.

```

import datetime

class DBFile:
    def __init__(self, file_id: str, in_dir: bool, name: str, content: bytes, added_date: datetime.datetime):
        self.file_id = file_id
        self.in_dir = in_dir
        self.name = name
        self.content = content
        self.added_date = added_date

    def __repr__(self):
        return f'{self.file_id} -> {self.name}, {self.added_date}'

class DBUser:
    def __init__(self, user_id: str, username: str, email: str, creation_date: datetime.datetime, files: list[str], encryption_key=''):
        self.user_id = user_id
        self.username = username
        self.email = email
        self.creation_date = creation_date
        self.files = files
        self.encryption_key = encryption_key

    def __repr__(self):
        return f'{self.user_id} -> {self.username} ({self.email})'

```

## IconGrabber.py

קובץ שנועד בשביל למצוא תמונה מתאימה לסוג הקובץ (כלומר, תמונה של מסמך אם זה קובץ וורד, אייקון אדום אם זה קובץ pdf, וכו')

```
import os

class IconGrabber:
    def __init__(self, extension_icons_repo: str):
        self.extension_icons_repo = extension_icons_repo

    def grab_filepath(self, filename: str):
        # Filename is a file brought from outside source, Example:
        # examp.xlsx
        if '.' in filename:
            ext = filename.split('.')[-1]
            file_projected_path = self.extension_icons_repo + ext +
            '.png'

            if os.path.isfile(file_projected_path):
                return file_projected_path

        # No file extension found
        return self.extension_icons_repo + '_blank.png'
```

## CloudEncrypt.py

קובץ האחראי על גיבוב המידע. פירטתי על גיבוב המידע תחת הכותרת 'אבטחה'. לכן, אציג פה את הקובץ אך להסבר המלא, יש לעלות למעלה ולקרוא על כך.

```
import hashlib
from datetime import datetime

def hashify_user(email: str, password: str, creation_date: datetime):
    # Some dynamic salting for md5 hash

    creation_time_mixed = str(creation_date.timestamp())[:-2]
    salt = hashlib.md5(email.encode()).digest()[::2] + \
        hashlib.md5(creation_time_mixed.encode()).digest()[::3]
    encstr = hashlib.md5(password.encode() + salt).hexdigest()
    return encstr
```

## TwoStepAuth.py

הקובץ שאחראי לאימות דו-שלבי של המשתמש. האימות הדו-שלבי מתרחש באמצעות האימייל, גם עליו פירטתי למעלה תחת הכותרת 'אבטחה'. בקצרה, המחלקה הזאת אחראית

## SafeCloud

על התחברות לאימייל השולח, ושליחת קודים לתיבות הדואר של משתמשים. יש ליצור אותה פעם אחת בלבד. (לא ניתן להתחבר לאותו משתמש כמה פעמים)

```
import random
import requests
from smtplib import SMTP_SSL
from email.mime.text import MIMEText
from decouple import config # For enviornment variables

class TwoStepAuth:
    def __init__(self, sender_email: str):
        # Personal details
        self.sender_email = sender_email
        self.smtp_server = 'smtp.gmail.com'

        # Create .env for these
        self.username = config('EMAIL_USERNAME')
        self.password = config('EMAIL_PASSWORD')
        print(f'Trying to log in with {self.username},
{self.password}... ', end='')

        self.conn = SMTP_SSL(self.smtp_server, 465)
        self.conn.set_debuglevel(False)
        self.conn.login(self.username, self.password)
        print('Logged in!')

    @staticmethod
    def check_if_email_exists(email_address: str):
        url = 'https://isitarealemail.com/api/email/validate' # A
        blessing for developers
        res = requests.get(url, params={'email': email_address})
        status = res.json()['status']

        if status == 'valid':
            return True
        return False

    def send_code(self, code: str, dest_email: str):
        # Beautify
        text_subtype = 'html'
        content = f"""
        <p>Thanks for logging in.</p>
        <p>Code: <b>{code}</b></p>
        """
        subject = 'SafeCloud code confiramtion'
        try:
            msg = MIMEText(content, text_subtype)
```



## SafeCloud

```
msg['Subject'] = subject
msg['From'] = 'SafeCloud'
self.conn.sendmail(self.sender_email, dest_email,
msg.as_string())

except Exception as e:
    print(f'Message failed: {e}')

def __del__(self):
    print('Closing smtp connection')
    self.conn.close()

@staticmethod
def generate_code(Len: int):
    letters = 'abcdefghijklmnopqrstuvwxyz'
    letters_big = letters.upper()
    digits = '0123456789'
    final_str = ''
    for i in range(Len):
        element = random.choice([letters, letters_big, digits])
        final_str += random.choice(element)
    return final_str
```

## CloudServer.py

הקובץ הזה, ניתן לקרוא לו "ליבת הפרויקט", לא משום שהוא מכיל את כל הקוד אלא משום שהוא קורא לכל הקוד הרלוונטי. הקובץ הזה אחראי להשיג את המידע שיבקש המשתמש, לשנות אותו, להוסיף ולמחוק חלקים. הוא כולל מחלקה שיוצרת אובייקט DBCommands ואובייקט IconGrabber בשביל שיהיה לה את היכולת לשלוח איוונים של קבצים ולשנות את מאגר הנתונים. לכל בקשת משתמש, יש כאן פעולה ולפעמים אפילו שניים:

```
from datetime import datetime
import json
from DBCommands import DBCommands
from DBObjects import DBFile, DBUser
import uuid
import base64
from IconGrabber import IconGrabber
from CloudEncrypt import hashify_user
class Codes:
    ADD_FILE = '101'
    GET_FILE_CONTENT = '102'
    DELETE_FILE = '103'

    REGISTER_USER = '201'

MAX_FILENAME_SIZE = 50
```

```

class CloudServer:
    def __init__(self, db_filename: str, extension_icons_path: str):
        self.db = DBCommands(db_filename)
        self.db.create_tables() # Doesn't do anything if exists
        self.icon_grabber = IconGrabber(extension_icons_path)

        # Fucntion adds a file to database and returns it's supposed icon
        # in base64
    def add_file(self, in_dir: bool, filename: str, content: bytes):
        file_id = str(uuid.uuid1()) # By timestamp
        creation_time = datetime.now()
        file = DBFile(file_id, in_dir, filename, content,
creation_time)

        print('Adding file: ', file)
        self.db.add_file(file)

        # After file is added, send back the icon to the client
        # so website will show file's added icon

        return file_id, self.get_file_icon_content(filename)

    def get_file_icon_content(self, filename: str):
        icon_filepath = self.icon_grabber.grab_filepath(filename)

        with open(icon_filepath, 'rb') as f:
            icon_content = f.read()

        return base64.b64encode(icon_content)

    # Function returns file base64 content based on filename
    def return_file_content_by_id(self, file_id: str):
        content = self.db.get_file_content(file_id)
        if content == None:
            print("Didn't receive anything from file, should get an
error")

        return content

    # Function deletes file with a give filename
    def delete_file_by_value(self, value: str, value_type: str):
        if value_type == 'file_id':
            self.db.remove_file_by_id(value)
        if value_type == 'filename':
            self.db.remove_file_by_name(value)

    # Function registers a user and returns a msg accordingly

```

## SafeCloud

```
def register(self, user_id: str, username: str, email: str,
creation_date: datetime):
    if self.register_possibility(email, username)['code'] ==
'error':
        print(f'User is already taken')
        return

    usr = DBUser(user_id, username, email, creation_date, [])
    self.db.add_user(usr)
    print(f'Registered {user_id} as {username} ({email})')

def register_possibility(self, email: str, username: str):
    # Must check for username & email uniqueness because
    # two users with same pass & username may sign in to different
accounts
    # because enetering a username will fetch their email for
unique code
    # and db won't know what email to fetch

    print(email, self.db.check_email_existance(email) )
    if self.db.check_email_existance(email):
        return {'code': 'error', 'msg': f'{email} already exists!'}

    if self.db.check_username_existance(username):
        print(username, 'already exists!')
        return {'code': 'error', 'msg': f'{username} ia already
taken.'}

    return {'code': 'success', 'msg': ''}

def try_login(self, usr_input: str, usr_pass: str):
    # User input may be email or username so we must check for both
cases
    input_type = 'email'
    if not self.db.check_email_existance(usr_input): # email not
exists
        input_type = 'username'
        db_usr = self.db.get_user_details_by_value(usr_input,
input_type)
        if db_usr == None:
            # Account does not exist
            return {'code': 'error', 'msg': 'User does not exist.'}

        # if input is email we hash with that, otherwise we use db
email
        if db_usr.user_id == \
            hashify_user(db_usr.email, usr_pass, db_usr.creation_date):
```

## SafeCloud

```
        return {'code': 'success', 'msg': db_usr.user_id}

    return {'code': 'error', 'msg': 'One of the credentials is
incorrect.'}

def get_user_file_ids(self, user_id: str):
    return self.db.get_user_files(user_id)

def get_user_filenames(self, user_id: str):
    filenames = []
    files_id = self.db.get_user_files(user_id)
    for fd in files_id:
        filename = self.db.get_file_name(fd)
        if filename != None:
            filenames.append(filename)

    return filenames

def add_file_to_user(self, user_id: str, file_id: str):
    self.db.change_user_file_ids(user_id, file_id, 'add')

def remove_file_from_user(self, user_id: str, file_id: str):
    self.db.change_user_file_ids(user_id, file_id, 'remove')

def get_filename(self, file_id: str):
    return self.db.get_file_name(file_id)

def get_user_details(self, value: str, value_type: str):
    return self.db.get_user_details_by_value(value, value_type)

if __name__ == '__main__':
    cs = CloudServer('127.0.0.1', 8081)
```

## app.py

אחד מהקבצים הארוכים בפרויקט. הסברתי כמעט על כל פעילותיו תחת הכותרת 'ארכיטקטורת רשת'. הקובץ אחראי למענה הקלטים שהדפדפן שולח (בהוראת המשתמש כמובן). הוא אחראי על תמרון הדפדפן, שינוי המידע בו, וכל התגובות שחוזרות אליו.

פעולה אחת שלא הסברתי, שאינה כל כך רלוונטית (או מעניינת), היא `configure_filename`, שבהינתן רשימת שמות ושם של קובץ, תשנה את שם הקובץ בהתאם לכמות הפעמים ששם הקובץ מופיע. כלומר אם יתנו לה רשימה `[safe.txt, ilan.txt]` ואת השם `safe.txt` היא תחזיר `safe (1).txt`. הפעולה מפורטת היטב בכתוביות.

```
from base64 import b64encode
from datetime import datetime, timedelta
from flask import Flask, jsonify, session, redirect, url_for,
render_template, request as rq
```

## SafeCloud

```
from flask_session import Session # Server-side session
from CloudServer import CloudServer
from TwoStepAuth import TwoStepAuth
import CloudEncrypt
import re, os

DB_FILENAME = os.path.abspath('./database.db')
EXTENSION_ICONS_PATH = os.path.abspath('./images/extension_icons') +
 '/'
tsa = TwoStepAuth('ilan147963@gmail.com')
auth_codes = [] # List of {email: code, email: code} used for 2step
auth
# just to prevent it from be shared in session

app = Flask(__name__)

app.config['SESSION_TYPE'] = 'filesystem' # For server-side sessions
Session(app)

# Secret key, session encrypting
app.secret_key = 'dev' # Const in development
app.config.from_pyfile('config.py', silent=True) # Overrides if
config.py exists

app.permanent_session_lifetime = timedelta(days=1)

cs = CloudServer(DB_FILENAME, EXTENSION_ICONS_PATH)

# NOTE Sessions are server-side
# - sensitive info can pass in
# - I tried to minimize it
# -> The only place where I send sensitive info is in register
# -> where it is not very important to hide from the client the info
#     he already entered...

@app.route('/')
def home():
    if 'user_id' in session:
        return redirect(url_for('files'))
    return redirect(url_for('login'))

@app.route('/register', methods=['POST', 'GET'])
def register():
    # The weird microsec cutting causes problems in login so I cut it

    if rq.method == 'POST':
        email = rq.form['email']
```

```

username = rq.form['username']
password = rq.form['password']

ans = cs.register_possibility(email, username)
if ans['code'] == 'success':
    # 2step auth is unique for both login, signup
    # But session vars preperation is needed
    session['password'] = password
    session['username'] = username
    session['email'] = email
    session['action'] = 'register'
    return redirect(url_for('two_step_auth'))
else:
    return render_template('register.html', err_msg=ans['msg'])

err_msg = session.pop('err_msg', None)
if err_msg:
    return render_template('register.html', err_msg=err_msg)
return render_template('register.html')

@app.route('/two-step-auth', methods=['POST', 'GET'])
def two_step_auth():
    redirect_to = 'files' # Where after 2step auth succeeds

    if rq.method == 'POST':
        # If we received a POST req, then we already
        # did a GET here, and values are popped
        # so this if must be first, to not get redirected
        received_code = rq.form['code_input']
        email = session.pop('email', None)
        user_action = session.pop('action', None)
        for ac in auth_codes:
            if ac['email'] == email and ac['code'] == received_code:
                # code that is typed is connected to the email
                # success
                auth_codes.remove(ac)
                session.permanent = True

                # Only in register -> Create an account
                if user_action == 'register':
                    print('registering')
                    password = session.pop('password')
                    username = session.pop('username')

                # Conversion is to delete micro-secs from now(),
                # makes problems...

```

## SafeCloud

```
        creation_time = datetime.now().strftime("%d/%m/%Y,
%H:%M:%S")
        creation_time = datetime.strptime(creation_time,
"%d/%m/%Y, %H:%M:%S")
        user_id = CloudEncrypt.hashify_user(email,
password, creation_time)
        cs.register(user_id, username, email,
creation_time)

        session['user_id'] = user_id # as a setup to /files
    else:
        session['user_id'] = cs.get_user_details(email,
'email').user_id # as a setup to /files

        session['filenames'] = []
        return redirect(url_for(redirect_to))

    session['err_msg'] = "Wrong code, please try again."
    return redirect(url_for('login'))

if 'email' not in session:
    return redirect(url_for('login'))

email = session['email']

if not tsa.check_if_email_exists(email):
    session['err_msg'] = 'Email does not exist!'
    session.pop('email', None) # Clear session
    return redirect(rq.referrer)

code = tsa.generate_code(6)
auth_codes.append({'email': email, 'code': code})

tsa.send_code(code, email)

return render_template('code_enter.html')

@app.route('/login', methods=['POST', 'GET'])
def login():
    if rq.method == 'POST':
        usr_input = rq.form['user_input']
        usr_pass = rq.form['password']
        ans = cs.try_login(usr_input, usr_pass)
        if ans['code'] == 'error':
            return render_template('login.html', err_msg=ans['msg'])
        else:
            user = cs.get_user_details(ans['msg'], 'user_id')
```

```

        if not user:
            return render_template('login.html', err_msg='No such
user..')

        session['email'] = user.email # Where to send
        return redirect(url_for('two_step_auth'))

err_msg = session.pop('err_msg', None)
if err_msg:
    return render_template('login.html', err_msg=err_msg)
return render_template('login.html')

@app.route('/files', methods=['POST', 'GET'])
def files():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    if rq.method == 'GET':
        # Fetch user's files
        user_id = session['user_id']
        filenames = cs.get_user_filenames(user_id)
        session['filenames'] = filenames

        files_data = []
        for fn in filenames:

            icon_content = b'data:image/png;base64,' +
cs.get_file_icon_content(fn)
            files_data.append(
                {'name': fn,
                 'icon_data': icon_content.decode()})
        return render_template('files.html', files_data=files_data)

    if rq.method == 'POST':
        icons_data = []
        files_num = int(rq.form['length'])
        for i in range(files_num):

            file = rq.files[f'files-{i}']
            name = file.filename
            content = file.stream.read()

            # Manage duplicates, edits session['filenames']
            saved_filename = configure_filename(name,
session['filenames'])
            file_id, file_icon_data = cs.add_file(False,
saved_filename, content)

```



```

        # Update user's files
        cs.add_file_to_user(session['user_id'], file_id)
        icons_data.append({'data': file_icon_data.decode(),
'filename': saved_filename})

    return jsonify(icons_data)

@app.route('/files/download/<filename>')
def download_file(filename: str):
    if 'user_id' not in session:
        return redirect(url_for('login'))
    # Search for filename among only the logged account
    files_id = cs.get_user_file_ids(session['user_id'])
    for fid in files_id:

        if cs.get_filename(fid) == filename:
            # Found relevant file, so I can download it
            content = cs.return_file_content_by_id(fid)

            return b64encode(content)
    return redirect(url_for('login')) # If file wasn't found

@app.route('/files/delete/<filename>')
def delete_file(filename: str):
    if 'user_id' not in session:
        return redirect(url_for('login'))
    # Search for filename among only the logged account
    files_id = cs.get_user_file_ids(session['user_id'])
    for fid in files_id:

        if cs.get_filename(fid) == filename:

            cs.delete_file_by_value(fid, 'file_id') # files
            cs.remove_file_from_user(session['user_id'], fid) # user
            session['filenames'].remove(filename) # session

            # Doesn't really matter if we didn't delete something that
            # didn't exist
            return jsonify({'message': 'Delete occurred'})

    return redirect(url_for('login')) # No file found

@app.route('/profile')
def profile():
    if 'user_id' not in session:
        return redirect(url_for('login'))
    user = cs.get_user_details(session['user_id'], 'user_id')
    if user:

```

## SafeCloud

```
# Send in it parts because I don't want all the details
# out there, sounds risky
return render_template('profile.html',
    username=user.username,
    email=user.email,
    creation_date = user.creation_date.strftime("%d %B, %Y"),
)
return 'error'

@app.route('/logout')
def logout():
    if 'user_id' in session:
        session.pop('user_id', None)
        session.pop('filenames', None)
    return redirect(url_for('login'))

def configure_filename(filename: str, curr_filenames: list[str]):
    saved_filename = filename
    actual_name = '.'.join(saved_filename.split('.')[:-1])
    ext = saved_filename.split('.')[-1]

    # Adding (duplicated file number) if needed

    if saved_filename not in curr_filenames:
        curr_filenames.append(filename)

    else:
        # turn name.ext -> name (counter).ext
        occurrences = 0
        for fn in curr_filenames:
            # we want to count 'filename' and 'filename (1)' for
            # example
            # so filenames.count is not enough (we won't count the
            # duplicated ones)
            # regex finds files with struct -> filename (num).ext
            if re.match(f'{actual_name}.* \\(\\d*\\).{ext}', fn) != None
            or fn == filename:
                occurrences += 1

            saved_filename = f'{actual_name} ({occurrences}).{ext}'

            curr_filenames.append(saved_filename)

        return saved_filename

@app.route('/about')
def about():
```

```
    return render_template('about.html')

# error pages -----

@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html', err_msg=e)

app.register_error_handler(404, page_not_found)

if __name__ == '__main__':
    # For SSL encryption a purchasable certificate is needed
    # as well as domain
    app.run(debug=True, )
```

## קבצים נוספים

קבצים שלא הצלחתי לקטלג אותם.

### Database.db

קובץ מאגר הנתונים, יוצר השרת יכול לבחור באיזה קובץ להשתמש, בעזרת שינוי משתנה המכיל את השם של הקובץ בapp.py. הקובץ הוא בינארי אז לא אוכל להראות אותו כאן אבל בשביל לקבל רעיון כיצד הוא נראה – אפשר להסתכל בכותרת 'בסיס נתונים'.

### .env

קובץ שלא תראו כשתעתיקו את הקוד מהגיט. הקובץ הזה מוסתר מהגיט משום שהוא מכיל פרטים שאסור לאחרים לראות. אתן דוגמה איך הוא אמור להראות:

```
EMAIL_USERNAME=username  
EMAIL_PASSWORD=password
```

## רפלקציה

אני אתחיל מזה, שבעת העבודה על הפרויקט – הרגשתי כאילו הוא לא נגמר. ואני לא אומר זאת מסיבה רעה, לא משום שהוא היה נוראי או משעמם, אלא משום שתמיד רציתי להוסיף עוד. כל פעם שחשבתי שאני מסיים, קפץ לי רעיון שהרגשתי – שבלעדיו אין שום סיכוי שאני מסוגל לקנות את המשתמש. רציתי שהנוחות ומהירות השימוש תהיה העדיפות הראשונה. לכן, כל פעם, הייתי מתעקש להוסיף עוד משהו, ועוד משהו, ועוד משהו... עד שהיה יוצא נוח למשתמש.

אני קיבלתי מהפרויקט הזה המון ידע. הטכנולוגיות והשפות שצברתי בהן ידע הן:

- Javascript, ajax, jquery, dynamic frontend
- Javascript events, global variables
- Flask, flask-sessions
- Bootstrap css

גם כן צברתי ידע תיאורטי בתחומים כמו:

- Clientside encryption
- Diffie Helman key exchange
- AES Encryption

ועוד נושאים רבים שכנראה הספיקו להיטמע במוחי.

עמדו בפני הרבה אתגרים בעת עשיית הפרויקט. על רובם פירטתי בתחילת הספר ולכן במקום לפרט אגיד שעיקר האתגרים – היה שהייתי צריך לממש דברים שבחיים לא יצא לי לממש מקודם. הייתי צריך לממש "זריקת" קובץ, טעינה דינמית של קבצים, הצפנות AES עם מפתח, שליחת אימיילים אוטומטיים, סוגי הצפנות שונות וכו'.

מהאתגרים והלקחים שלמדתי, הגעתי למסקנה אחת – לא משנה כמה קשה, ויהיה קשה (אחרת איך תלמד?), צריך להמשיך להתקדם כי זה פשוט חלק מיצירת פרויקט, חלק מהמקצוע הכל כך מדובר הזה, וחלק מההצלחות בחיים.

לו הייתי מתחיל היום, האמת, שלא הייתי משנה כל כך את אופן העבודה. אני חושב, שבניגוד לפרויקטים אחרים, הפעם פחות התרכזתי על סידור הקוד וחילוקו לקבצים (באופן יחסי, פעם הייתי נותן לזה חשיבות עליונה) ולכן, דווקא הפרויקט זז לי יותר מהר וכמעט ולא היה איבוד מוטיבציה. מניגוד לזה, הייתי מפנה יותר זמן לפרויקט – משום שיצא שעבדתי עליו מאוד אינטנסיבי בתקופות מסוימות – בצורה שאולי לא הייתה הכי בריאה לעיתים.

לסיכום, באמת נהנתי מעשיית הפרויקט. אני שמח מאוד שהגעתי לתוצר, כי פעמים רבות לא מצליחים לסיים דברים, במיוחד במקצוע הזה (אם זה בגלל מחסור במוטיבציה ואם זה בגלל מחסור בזמן). אני שמח שהתוצר שלי עובד יחסית טוב, וסיכוי גבוה שאני עצמי הייתי משתמש בו, עם עוד קצת עבודה. אני שמח שלמדתי המון מהפרויקט ושהתמודדתי עם כל כך הרבה אתגרים, כי זה רק הפך את הסיום למתוק יותר.