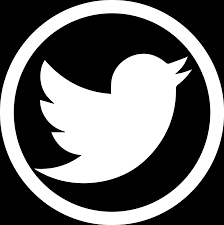
****

**אפליקציה חברתית דמוי טוויטר**

**העלאת פוסטים שונים ותקשורת עם משתמשים**

**אחרים בפומביות**

****

**שם התלמיד: שון איצקובסקי**

**מספר תעודת זהות: 215402322**

**מורה: יורם אביטוב**

**תאריך הגשה:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# טבלת שינויים לאורך ממימוש הפרויקט

יש למלא הטבלה בהתאם להתקדמות הפרויקט

|  |  |  |  |
| --- | --- | --- | --- |
| **תאריך סיום** | **תכולה / שינוי** | **גרסה** | **פעילות** |
| 25.10.21 | הצעה ראשונית | 0.1 | יזום |
| 1.11.21 | הצעה ראשונית לאחר שינויים בנוסף להסבר על דרכי מימוש הפרויקט | 0.1 | יזום |
| 27.11.21 | הסבר על סביבת העבודה הכוללת ספריות  וכלים ושפת התכנות.  אפיון דרישות, ארכיטקטורת מערכת. | 0.2 | אפיון דרישות, ארכיטקטורת מערכת, כלי  פיתוח |
| 3.1.21 | הגשת גרסת השלד של הפרויקט גמורה ללא הרבה פונקצינוליות | 1 | קידוד v1.0 שלד  עובד |
| 31.1.22 | שמירה למסד הנתונים, שימוש במסד הנתונים לבדיקה – מערכת העוקבים וההרשמה והכניסה כבר בעלי פונקיאנוליות | 2 | קידוד v2.0 |
| 21.2.22 | מעבר מhttp לhttps קליטת פוסטים ותמונת פרופיל והצגתם במסך הפתיחה ולעוקבים אחרים | 3 | קידוד v3.0 |
| 14.3.22 | תיקון בעיות – שיפור יעילות הקוד, הוספת תיעוד וסידור הקבצים | 4 | קידוד v4.0 |

**שליטה מרחוק – Remote Desktop**

**תוכן עניינים**

[טבלת שינויים לאורך ממימוש הפרויקט 3](#_Toc116111548)

[1 מבוא– תיאור נושא הפרויקט 4](#_Toc116111549)

[1.1 נושא המחקר בפרויקט 5](#_Toc116111550)

[2 סביבת העבודה והספריות העיקריות שבשימוש בפרויקט](#_Toc116111551)

[2.1 טכנולוגיות בשימוש בפרויקט](#_Toc116111552)

[2.2 מדריך למשתמש](#_Toc116111553)

[3 אפיון דרישות וארכיטקטורת מערכת](#_Toc116111554)

[3.1 דרישות ושימושי מערכת – Use Cases](#_Toc116111555)

[3.2 סביבת הפרויקט – Eco – System](#_Toc116111556)

[3.3 ארכיטקטורת המערכת](#_Toc116111557)

[3.4 טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת](#_Toc116111558)

[4 ממשק משתמש - GUI](#_Toc116111559)

[5 מדריך למפתח](#_Toc116111560)

[5.1 דיאגרמת UML של כל מחלקות הפרויקט והתלויות ביניהן](#_Toc116111561)

[5.2 רשימת פונקציות ומחלקות ותפקידיהם](#_Toc116111562)

[6 סיכום אישי / רפלקציה](#_Toc116111563)

[7 מקורות מידע / ביבליוגרפיה](#_Toc116111564)

# מבוא– תיאור נושא הפרויקט

הנושא שנבחר הינו אתר, אפליקצית מדיה הדומה לטוויטר. האפליקציה נועדה על מנת שאנשים יוכלו לחלוק דעות בחופשיות, יוכלו לפרסם תוכן לפי רצונם, כגון תמונות או סרטונים האפליקציה תמומש בממשק web.

באפליקציה לא ינתן לאנשים לפרסם תוכן כמו וירוסים או קבצים שונים מתמונות, סרטונים או להעלאת טקסט.

באפליקציה המשתמשים יוכלו דברים שכתבו כך שכל העוקבים שלהם יוכלו לראות. יש למשתמשים את היכולת לבדוק מי עוקב אחריהם ולאשר או לדחות או לבטל עוקב מאנשים אחרים.

על כל דבר שיפרסמו משתמשים יוכלו להגיב תחת אותו פוסט. המשתמשים יוכלו גם למחוק את הפוסטים שלהם – שימחוק גם את התגובות.

האפליקציה תשמור את כל הפוסטים שהמשתמשים העלו, תשמור נתונים על משתמשים – סיסמא, שם משתמש, מייל, תשמור רשימת כל האנשים שהמשתמש עוקב אחריהם ורשימת כל הבקשות לעקוב.

האפליקציה תשתמש בקריפטוגרפיה על מנת להבטיח סודיות , אימות ושלמות המידע – האתר יהיה https.

## נושא המחקר בפרויקט

נושא המחקר הינו big data ויותר ספיציפית כיצד לטפל בbig data בצורה יעילה ואיכותית במקרה של הפרויקט שלי.

בפרויקט שבחרתי, נוצר הרבה מידע בשרת מהצורך לפרסם את ההודעות הפומביות של המשתמשים למשתמשים אחרים, המידע מגוון כגון טקסטואלי, תמונות או סרטונים. מכך שיש צורך מרכזי בניהול המידע בשרת בהכנת הפרויקט.

כאן בא הנושא big data, ניהול המידע שיהיה קל לשלוף, מסודר, חסכוני יחסית ולעשות זאת בחסיכת זמן.

הנושא ישומש בשמירה על הנתונים הפרטיים של כל משתמש – הדברים שכבר צוינו קודם לכן.

מכיוון שכבר צוינו דברים רבים שיש צורך בשמירה, ניתן להבין שיש כמות ענקית של מידע שצריך לסדר. המידע יצטרך להישמר במקום מאובטח כאן טיפול בbig data נכנס.

לאחר בדיקת מצב השוק של נושא זה, ניתן להבין שנושא זה מאד פופאלארי וישנם ספריות מתקדמות רבות בפייטון הממשות נושא זה.

מהמחקר הסקתי שאני אשתמש ב SQLite, הוא בסיס נתונים היוכל להחזיק כמות עצומה של מידע. בנוסף לכך במדולים של Django על מנת לשמור פוסטים –  Django ORM

באמצעות SQLite יהיה ניתן לסדר את המידע לפי צורך הפרויקט וגם לגשת אליו בקלות רבה ורק לשרת יהיה את הגישה אליו.

# סביבת העבודה והספריות העיקריות שבשימוש בפרויקט

שפות תכנות:

Python - פייטון נבחרה בעקבות כך שיש בה מבחר רב של ספריות להעברת נתונים, ומאגרי מידע, דברים אלו והספרייה Django מקלים על ניהול שרת HTTPS ושמירה ושימוש בנתונים מהמשתמש.

HTML, JavaScript, CSS – שפות פשוטות על מנת ליצור דף לפרוטוקול https – html משמש למבנה של האתר ולכתוב בו, JavaScript משמש לביצוע שינויים בנוסף לפונקציליות של דברים רבים CSS – משמש לעיצוב האתר.

העבודה תוכנתה בPyCharm והשתמשה בSQLite

ספריות:

sqlite3 – שימוש database, פשוט לשימוש, המידע שנשמר בתוכו טקסטואלי והשמירה נעשת כשמירה בטבלה.

Pathlib – עוזר לשימוש בשינוי כתובות של קבוצים וגישה למיקום של הקובוץ בהינתן יש צורך לבנות את הכתובת.

Django – עוזר לניהול התקשורת מול קליינטים רבים, משמש גם בתור ממסד נתונים, ניתן להכין בתוכו forms ולקבל מהם מידע (והמידע נשמר בממסד נתונים), מכיל בתוכו ספריות רבות, בנוסף מקבל על שליחת פרמטרים מסוימים והכנסתם אל הGUI.

os – יכול לשמש לבדיקת קיום של קבצים, של כתובות בנוסף לכך שיכול גם למחוק קבצים וליצור גם תיקיות.

## טכנולוגיות בשימוש בפרויקט

שרת מרובה משתמשים, mysql לאיחסון מידע רב מהמשתמשים, אבטחת מידע, בנוסף השתמשות בהצפנות, נתינת סודיות ואימות מידע. יש שימוש בdigital certificate של השרת לאבטחת אמינות בקשר, שימוש בממשק וובי – https.

השימוש בmysql נעשה כדי לשמור פרטים על המשתמש כגון שם משתמש , אמייל, סיסמה ואיזה משתמש עוקב אחרי משתמש כלשהו (כולל בקשה לעקוב). בנוסף לתאריך העידכון האחרון שנשמר לצורך ידיעה של משתמשים אחרים (יש לעדכן משתמשים אחרים באתר אם המשתמש בדיוק מחק פוסט לדוגמא ובתאריך רשום גם מה עודכן).

שמירת הפוסטים והתגובות נעשתה באמצעות Django models. קבלת המידע באמצעות Django forms.

השימוש בDjango models נעשה בגלל שעובד ממש טוב עם Django forms הדבר הופך את השמירה של המידע שניתן מהמשתמש ליותר פשוטה ובשימוש Many to one relationships השיוך של התגובות לפוסטים פשוט מאד.

נעשה גם שימוש בספריית os על מנת בדיקת קיום של קבצים ומחיקה של קבצים (כאשר יש מחיקה של פוסטים או כאשר יש החלפה של תמונת פרופיל).

בנוסף בפרויקט יש העברת קבצים בקשר (קבצי תמונות וקבצי סרטונים), העברת מידע טקסטואלי כמובן דרך הקשר.

נעשה שימוש גם בredirects וcookies – זאת על מנת שיהיה וידוי שהמשתמש קודם נתן את פרטיו ורק אז יוכל להשתמש באתר ושיהיה ניתן לזהות את המשתמש.

## מדריך למשתמש

צד לקוח:

על מנת לגשת לאפליקציה יהיה צורך רק ללכת לsearch engine כמו גוגל למשל ולהכניס את הכתובת שהיא הIp של השרת, להתחבר בפורט 8000 אל site או login או signup.

דוגמא לכניסה (אך בחיבור LAN)

https://192.168.1.24:8000/signin

לשימוש במערכת כמובן צריך חיבור לאינטרנט, הקליינט יצטרך ליצור משתמש במערכת לפני שיכול להתחיל להשתמש בה או שיתחבר למשתמש שקיים שלו.

לביצוע פעולות מסוימות המורשות על ידי המערכת, יהיה ברור למשתמש לפי השמות שלהם בגרפיקה והסברים לפעולות.

צד שרת:

ספריות נוספות:

פירוט (חסר)

לאחר התקנת קבצי השרת, בשימוש בPyCharm ו python 3.10(אם אין יש צורך בהתקנה) יש לפתוח את אחד הקבצים, להוריד Django דרך הטרמינל באמצעות הפקודות: pip install django וpip install django-extensions.

כאשר כל הספריות מורדות יש לנווט דרך הטרמינל אל הroot folder של הפרויקט – שהיא היכן שהקובץ: "manage.py" נמצא, ולהריץ את הפקודה: python manage.py runserver 0.0.0.0:8000 זאת כדיי להריץ באמצעות http.

על מנת להריץ באמצעות https יש להריץ את הפקודה: python manage.py runserver\_plus 0.0.0.0:8000 --cert-file cert.pem --key-file key.pem רק כאשר קיימים הקבצים: cert.pem וkey.pem, כלומר צריך קודם ליצור או לייבא חתימה דיגטילית ומפתח להצפנה.

אם אין ניתן להכין באמצעות השלבים הבאים: להוריד את הספרייה mkcert (אפשר דרך הcmd) ואז להריץ את הפקודה: mkcert -cert-file cert.pem -key-file key.pem 192.168.1.24 כאשר הIp צריך להיות הIp של המחשב.

# 

# אפיון דרישות וארכיטקטורת מערכת

## דרישות ושימושי מערכת –Use Cases

הדרישה לקבל מידע רב מהמשתמים ולאחסן אותו במאגר מידע בטוח, אמין וקל לגישה בשביל השרת. דרישה של השרת לטפל בכמה קליינטים בו זמנית. הדרישה להפיץ את המידע המאוחסן בעת הצורך, כאשר משתמש העלה פוסט חדש לתת אפשרות לעוקבים של אותו משתמש לראות את הפוסט ללא דיליי רב כלל לאחר העלאת הפוסט (או התגובות או משהו נוסף).

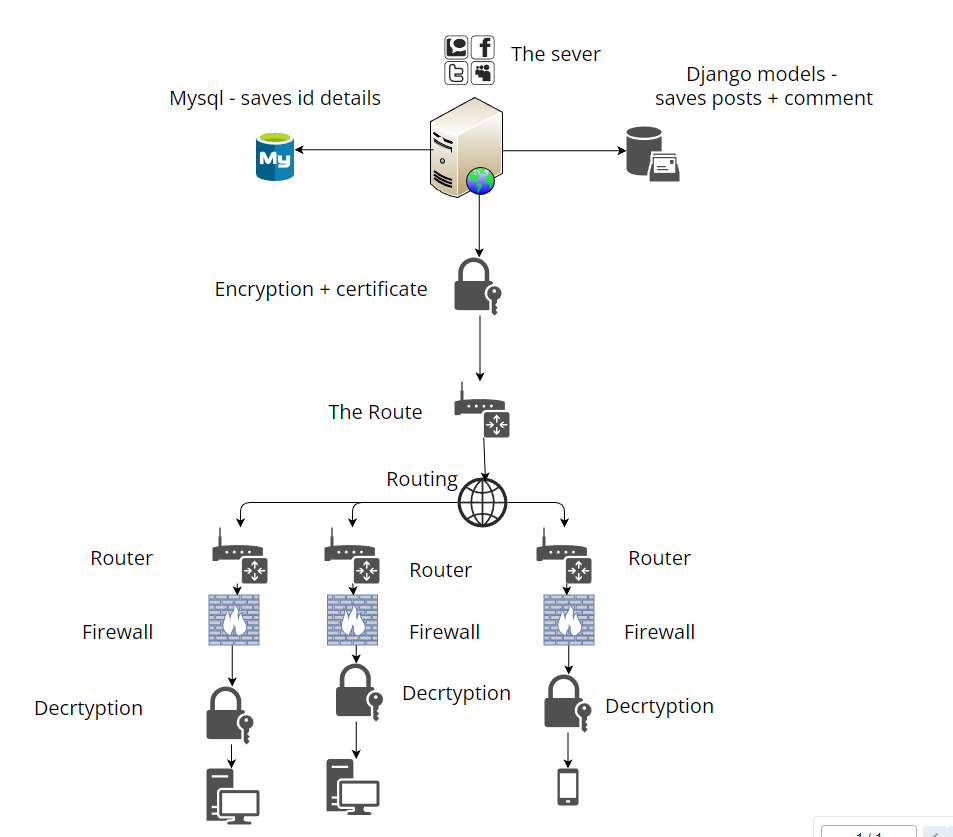
בנוסף, התוכנה צריכה להיות ברורה ונוחה למשתמש – המשתמש יוכל להבין ישר איך לעשות כל דבר מהאופציות של התוכנה בלי לקרוא יותר מדי רק לפי מבנה האתר. האפליקציה צריכה גם להיות מאובטחת ואמינה – המשתמש צריך לדעת שהמידע שלו בטוח ואין אנשים שרואים את המידע שהוא שולח לאתר ללא רשותו.

האפשרויות שהתכונה צריכה לתת להן מענה הן: ניתן יהיה להשתמש במערכת להעלת פוסטים הכוללים תמונות או סרטונים או סתם הודעות. המשתמשים יוכלו לעקוב אחרי משתמשים אחרים ובכך יוכלו לראות את הפוסטים שלהם. דבר זה יבוצע על ידי שמירת פוסטים עבור כל משתמש שהעלה אותם ושיוך הפוסט אליו. בנוסף שמירת האנשים אחריהם המשתמש עוקב.

המשתמש גם יוכל לבחור תמונת פרופיל בשביל עצמו, ויהיה צורך לשמור אותה כמובן כדי להראות אותה למשתמשים אחרים.

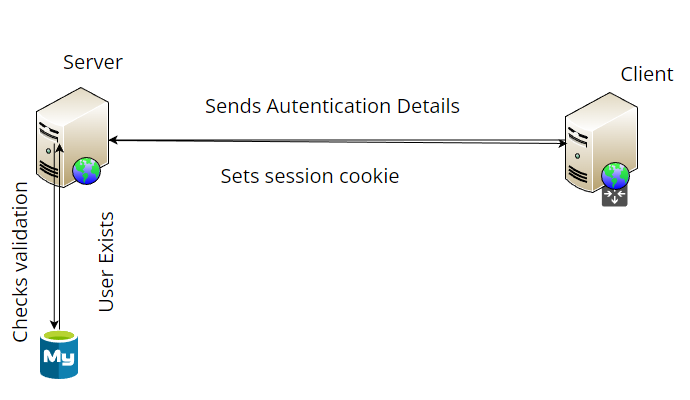
בנוסף, יהיה אפשרות להגיב לפוסטים גם דבר הצריך לשמור. לסיכום הדרישה העיקרית והכי כבדה היא שמירה וניהול מאגרי מידע גדולים.

## סביבת הפרויקט –Eco – System

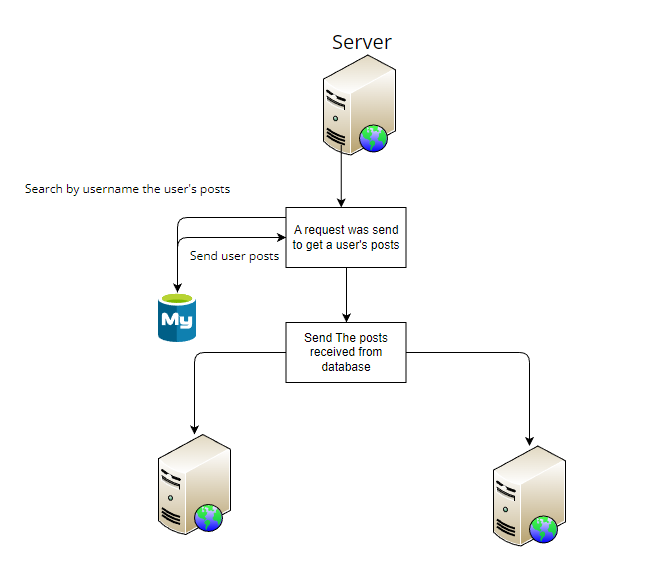


## ארכיטקטורת המערכת

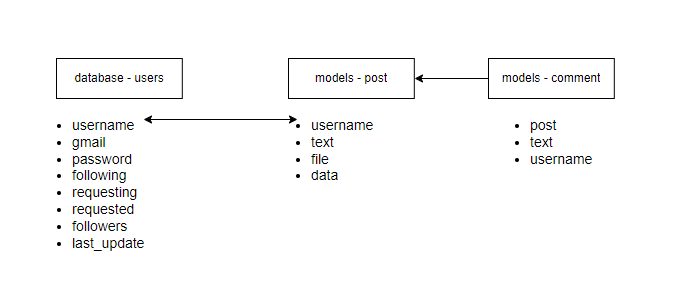
* הגישה כאן Top-Down Level Design
* שרת- לקוח
* יש לצייר דיאגרמת מלבנים כללית של המודולים העיקריים ואופן זרימת המידע ביניהם
* עבור על מודול יש לפרט תפקידו, קלט ופלט למודול (קלט יכול להיות מהמשתמש, ממסד נתונים או כל מודול אחר בפרויקט)
* פרוטוקולי תקשורת – באילו משתמשים ואיפה
* אזורים הדורשים אבטחה : שרת, בקרת גישה, מאגר מידע, ותיאור מהי האבטחה שנבחרה
* תיאור מסד הנתונים שהשתמשתם בהם. (טבלאות ואם יש יותר מאחת הקשרים ביניהם אם קיימים)
* תיאור ארכיטקטורת הרשת



התמונה ממחישה את הכניסה לשרת, הקשר נעשה מעל פרוטוקול הקריפוטגרפי TLS בשימוש בself signed certificate וpublic key וprivate key.



התמונה ממחישה מה קורה כאשר משתמשים מבקשים פוסטים של משתמשים כלשהם (זאת ללא ההנחה שהפוסטים לא שייכים למשתמש שביקש אותם אך ישנה ההנחה שאם הפוסטים לא שייכים לאותו משתמש שביקש אותם, המשתמש שביקש את הפוסטים עוקב אחרי המשתמש המבוקש, הדבר נבדק כמובן בתוכנה).



במשתמשים נשמרים כמה דברים: username – שם המשתמש, gmail – מייל המשתמש וpassword שכולם משתמשים על מנת זיהוי המשתמש וusername משתמש גם להצגת המשתמש בפני משתמשים אחרים. following – משמש לראות אחר מי המשתמש עוקב כדי לדעת אם ניתן להראות את הפוסטים של אותו משתמש, requesting משמש לראות אחרי מי המשתמש מבקש לעקוב, requested מראה מי מבקש לעקוב אחר המשתמש וfollowers מי עוקב אחרי המשתמש. בlast\_update נשמר גם מתי המשתמש עשה שינוי בפעם האחרונה – השינוי כולל בתוכו כמה דברים אך לא הכל, דוגמא לכך היא שתאריך חדש נשמר כאשר המשתמש מחק פוסט. בתאריך רשום גם מה השתנה

בפוסטים נשמרים – שם המשתמש אליו משויך הפוסט, הטקסט הכתוב בפוסט, הקובץ שנבחר הurl שלו ודברנים נוספים אליו וdate תאריך שמתעדכן אוטומטית.

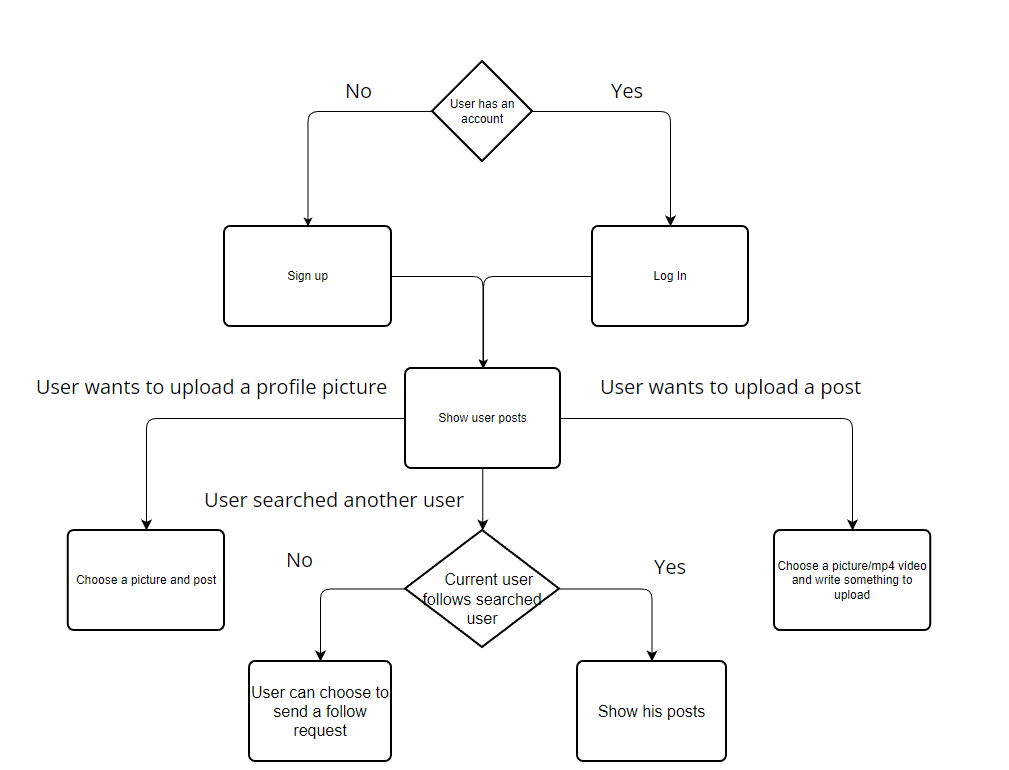
ובתגובות נשמר הid של הפוסט, הטקסט שרשום בתגובה ושם המשתמש שרשם את התגובה.

כל אחד מהמאגרי מידע מנוהל באמצעות קשר של רבים אל אחיד – Many to one relationship כלומר עבור כל משתמש שנשמר יכול להיות לו פוסטים רבים ועבור כל פוסט יכולות להיות תגובות רבות. פוסטים גם יכולים להימחק ולכן כאשר הפוסט נמחק כל התגובות המשויכות לפוסט צריכות גם הן להימחק. יש צורך לשמירת זיהוי של כל אחד מהמאגרים עם המאגר שקשור אליו כלומר הפוסטים שומר את השם משתמש והתגובות עובדות לפי הid של הפוסט )בDjango זה נקרא foreign key).

## טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת

|  |  |
| --- | --- |
| דרישה | מימוש |
| התמודדות עם כמות רבה של מידע לאחסן – ריבוי משתמשים ופוסטים | שימוש בדאטאבייסים שיכולים להחזיק כמות גדולה של מידע – הקבצים עצמם נשמרים כל אחד בתיקייה של המשתמש – יכול לאחסן קבצים רבים אך הכמות תלויה בגודל איחסון המחשב |
| ניהול המידע – התאמת המידע למשתמש ונתינת מידע לפי ההרשאות של המשתמש (אם משתמש מבקש מידע על משתמש אחר הוא צריך לעקוב אחריו) | קישור בין הדאטאבייסים – שמירת שם המשתמש בכל פוסט ושיוך בכך של כמה פוסטים למשתמש אחד, ישנם פונקציות בנויות שאחראיות על הוצאת מידע ובודקות הרשאות. |
| הקשר צריך להיות מאובטח | השימוש בhttps כרוך בתוכו אבטחת מידע – האות s היא קיצור של secure – ובתוכה יש שימוש בפרוטוקול TLS – פרוטוקל אבטחה שמשתמש גם בהצפנה סימטרית ואסימטרית להעברת המידע. |
| האתר צריך להיות בנוי כך שהמשתמש יוכל להבין את אפשרויות התוכנה במלי להסתבך יותר מדי | השימוש בhtml, bootstrap בתוספת לשמות משמעותיים עבור האפשרויות והסברים שרשומים באתר הופכים את השימוש בו לשפוט מאד |
| קבלת מידע מהמשתמש | נעשה באמצעות html froms, Django forms, ajax כל אחד מהדברים האלה משמש לקבל מידע של דברים אחרים וביחד מקבלים את כל המידע שצריך מהמשתמש כגון – קבצים, טקסט או בקשת עקיבה ועוד |

# ממשק משתמש - GUI



דוגמא להרשמה וכניסה לאתר ואז העלאת פוסט + בחירת תמונת פרופיל.

מסך הרשמה:

Graphical user interface, application

Description automatically generated

מסך כניסה:

Graphical user interface, application

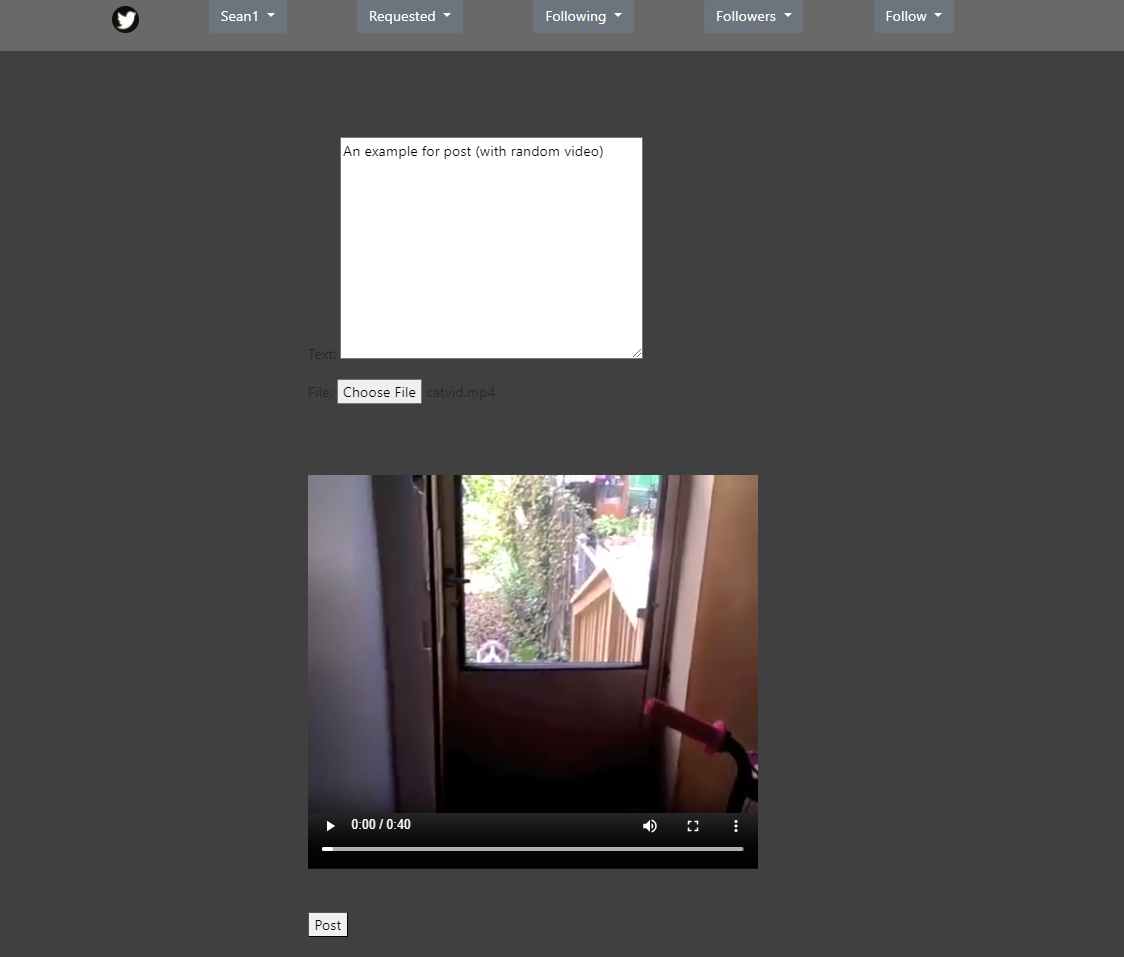
Description automatically generated

מסך האפליקציה הראשי (ריק כי אין למשתמש פוסטים)

Graphical user interface, text, application

Description automatically generated

העלאת פוסט:



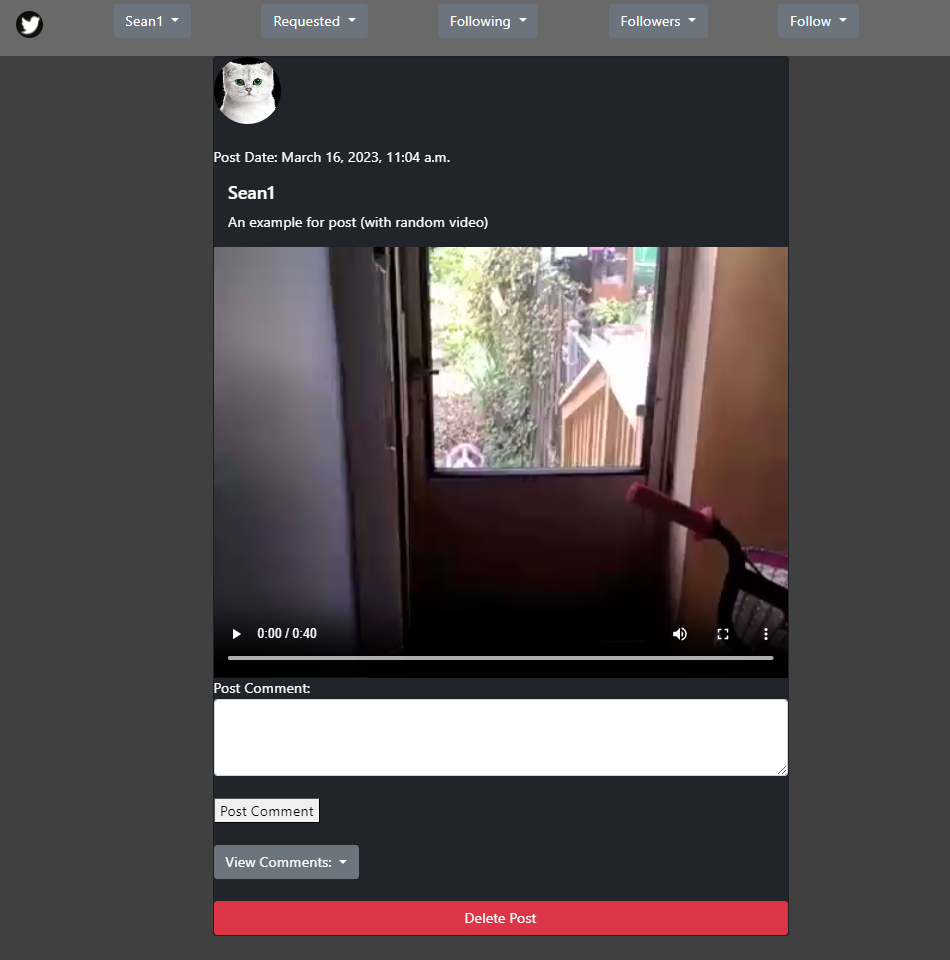
העלאת תמונת פרופיל:

Graphical user interface

Description automatically generated

מסך הפתיחה כעת:

כפי שניתן לראות בתמונה יש אפשרות להעלאת תגובה, לראות את התגובות וגם למחוק את הפוסט. בלמעלה של המסך יש את כל האפשרויות לעוקבים, לבקשות של עוקבים ולבקש לעקוב אחר משתמש מסוים.



# מדריך למפתח

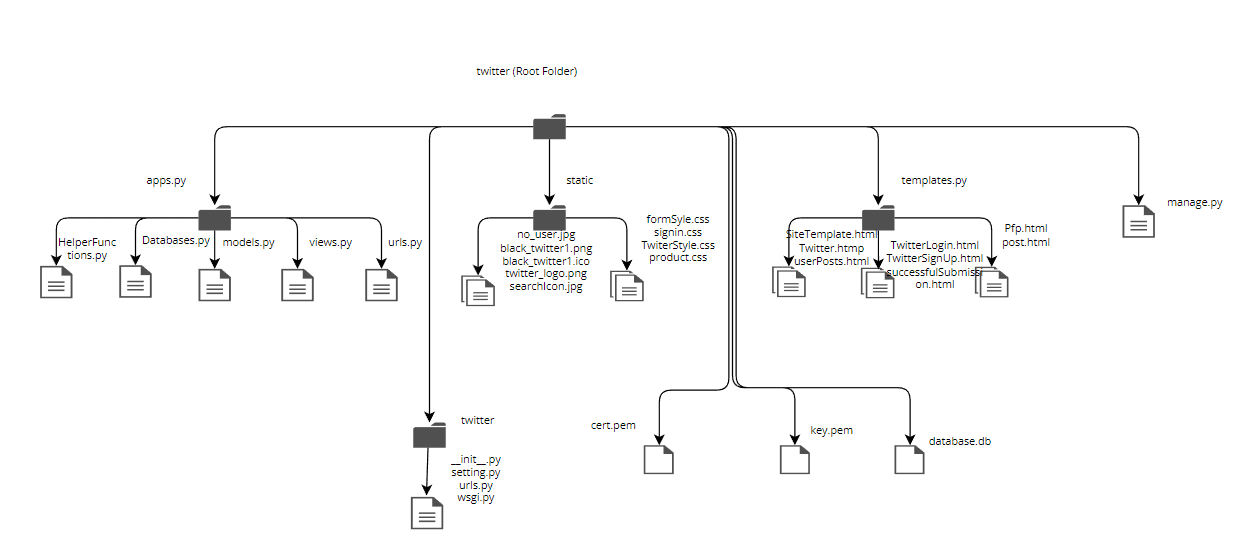
## 

## דיאגרמת UML של כל מחלקות הפרויקט והתלויות ביניהן

## רשימת פונקציות ומחלקות ותפקידיהם

עבור כל קובץ בפרויקט יש לרשום: (אפשר גם בטבלה)

* שם הקובץ / המחלקה ותפקידו, עבור מחלקות תיאור הפעולות העיקריות בהן.
* מיקום הקבצים בפרויקט
* מבנה הנתונים שהשתמשת בהם סוג מסד הנתונים וקבצים רלבנטיים





|  |  |  |
| --- | --- | --- |
| קובץ | תפקיד | מחלקות |
| manage.py | הרצה של הקבצים הנצרכים ובמקרה של https הרצה מעל TLS בשימוש בkey.pem וcert.pem |  |
| key.pem | המפתח המשומש להצפנה |  |
| cert.pem | הcertificate המשומש בתקשורת |  |
| settings.py | משמש את הmanage.py מגדיר שימוש בספריות וקבצים מסוימים, דוגמאות להגדרות נוספות הן איזור זמן ושפה |  |
| asgi.py | ממשק אסינכרוני לאתר |  |
| wasgi.py | נותן את הכתובת לאפליקציה |  |
| database.db | המאגר נתונים של הsqlite – שומר את פרטי המשתמשים שכבר צוינו |  |
| db.sqlite3 | המאגר הנתונים של Django בשימוש בmodels שומר פוסטים ותגובות |  |
| pfp.html | הממשק הגרפי בשימוש לבחירת תמונת פרופיל |  |
| post.html | הממשק הגרפי בשימוש להעלאת פוסט |  |
| siteTemplate.html | הממשק הגרפי המשמש להרבה מהממשקים הגרפים האחרים שבאתר |  |
| successfulSubmission.html | הדף המודיע על קליטת נתונים שהסתיימה בהצלחה |  |
| Twitter.html | הדף הראשי באתר – לאחר שנכסים שמראה את הפוסטים של המשתמש |  |
| TwitterLogin.html | דף ההתחברות לאתר |  |
| TwitterSignUp.html | דף ההרשמה לאתר |  |
| userPosts.html | דף המראה את הפוסטים של משתמש לאחר שחיפשו אותו בsearch bar |  |
| black\_twitter1.ico | הicon |  |
| no\_user.jpg | תמונה של משתמש ללא תמונת פרופיל שתיוצג בפוסטים |  |
| black\_twitter1.jpg | תמונה שמומשת כדי להציג את סמל האתר |  |
| twitter\_logo.jpg | תמונה נוספת אחרת גם מציגה את האתר |  |
| searchIcon.jpg | תמונה שמראה icon של חיפוש |  |
| signin.css, TwitterStyle.css, product.css, formStyle.css, twitterStyle.css | קבצים לעיצוב הדפים |  |
| urls.py – apps/urls.py | קובץ קצר תוכן אך חשוב מאד – הקובץ מפנה את הurl לקובץ views וספיציפית לפונקציה המתאימה לו ושולח למשתמש את מה שהפונקציה מחזירה |  |
| views.py | הפונקציות שמקבלות את הבקשות של המשתמש ומשתמשות בהן לדברים שונים בנוסף למחזירות למשתמש דף מסוים או משהו אחר בהתאם לצורך הבקשה |  |
| models.py | המודלים והכלאסים שמשמשים עבור הדאטאבייס של Django וforms מסוימים נמצאים שם | ProfilePicture- המחלקה משמשת לבניית התכונות שנשמרות בדאטאבייס לתמונת פרופיל של המשתמש  PictureForm – המחלקה משתמש בתכונות המחלקה הקודמת ומכינה form שנשלח למשתמש שממלא את התכונות הללו לפי הצורך  Post – המחלקה משמשת לבניית התכונות שנשמרות בדאטאבייס לפוסטים  PostForm – המחלקה משתמש במחלקה הקודמה ליצור form בעת הצורך למילוי התכונות שנצרכות  comment – מחלקה של תגובות לפוסטים – לפוסט אחד יכולות להיות תגובות רבות והתגובות נמחקות כאשר הפוסט נמחק |
| Databases.py | הקובץ שאחראי על ניהול המאגר sql של פרטי המשתמשים | ManageDatabase – אחראי על ניהול מאגר המידע של פרטי המשתמשים – פעולות עיקריות בו הן להחזיר עמודה במיקום ספיציפי לפי שם המשתמש, הוספת מידע לעמודה ספיציפית ומחיקה. |
| HelperFunctions.py | הקובץ בעל פעולות עזר לקבצים שונים המטרה העיקרית שלו היא להפוך את הפרויקט ליותר פשוט להבנה ולקריאה ולכך שפונקציות בviews יהיו רק פונקציות שימשו עבור כתובות ספיציפיות (פונקציות עזר לא ימצאו ב views) |  |
| apps.py | עוזר לDjango לכלול את הספריות המורדות ביותר קלות |  |

# סיכום אישי / רפלקציה

* פרק חשוב ורצוי להרחיב ככל שניתן ולא להסתפק במילים כלליות מידי...
* איך היה עבורך העבודה בפרויקט
* מה למדת תוך כדי העבודה על הפרויקט.
* אילו קשיים נתקלת במהלך הפיתוח , ומהן המסקנות מקשיים אילו
* מה הייתה עושה אחרת לו הייתה מתחיל היום את הפרויקט, בהינתן הידע שצברת תוך כדי המימוש

# מקורות מידע / ביבליוגרפיה

<https://stackoverflow.com/>

<https://docs.djangoproject.com/en/4.1/topics/forms/>

<https://docs.djangoproject.com/en/4.1/topics/db/models/>

<https://www.w3schools.com/js/default.asp>

<https://www.w3schools.com/css/default.asp>

<https://www.w3schools.com/html/html_attributes.asp>

<https://api.jquery.com/jQuery.ajax/>

<https://docs.djangoproject.com/en/4.1/ref/templates/language/>

<https://timonweb.com/django/https-django-development-server-ssl-certificate/>

<https://www.geeksforgeeks.org/getting-started-with-django/>

<https://getbootstrap.com/docs/5.1/getting-started/introduction/>

<https://docs.djangoproject.com/en/4.1/topics/http/shortcuts/>