



אפליקציה חברתית דמוי טוויטר

**העלאת פוסטים שונים ותקשורת עם משתמשים
אחרים בפומביות**



שם התלמיד: שון איציקובסקי

מספר תעודת זהות: 215402322

מורה: יורם אביטוב

תאריך הגשה: 5/6/2023

(הערה: העלתי את הפרויקט לGitHub על שם SeanZKY)

טבלת שינויים לאורך ממימוש הפרויקט

פעילות	גרסה	תכולה / שינוי	תאריך סיום
יזום	0.1	הצעה ראשונית	25.10.22
יזום	0.1	הצעה ראשונית לאחר שינויים בנוסף להסבר על דרכי מימוש הפרויקט (בוורד)	1.11.22
אפיון דרישות, ארכיטקטורת מערכת, כלי פיתוח	0.2	הסבר על סביבת העבודה הכוללת ספריות וכלים ושפת התכנות. אפיון דרישות, ארכיטקטורת מערכת.	27.11.22
קידוד v1.0 שלד עובד	1	הגשת גרסת השלד של הפרויקט גמורה ללא הרבה פונקציונליות	3.1.23
קידוד v2.0	2	שמירה למסד הנתונים, שימוש במסד הנתונים לבדיקה – מערכת העוקבים וההרשמה והכניסה כבר בעלי פונקציונליות	31.1.23
קידוד v3.0	3	מעבר מhttp לhttps קליטת פוסטים ותמונת פרופיל והצגתם במסך הפתיחה ולעוקבים אחרים	21.2.23
קידוד v4.0	4	תיקון בעיות – שיפור יעילות הקוד, הוספת תיעוד וסידור הקבצים הוספת נגיעות אחרונות	20.4.23

אפליקציה חברתית – דמוי טוויטר

תוכן עניינים

2	טבלת שינויים לאורך ממימוש הפרויקט	
4	מבוא – תיאור נושא הפרויקט	1
5	נושא המחקר בפרויקט	1.1
6	סביבת העבודה והספריות העיקריות שבשימוש בפרויקט	2
7	טכנולוגיות בשימוש בפרויקט	2.1
9	מדריך למשתמש	2.2
12	אפיון דרישות וארכיטקטורת מערכת	3
12	דרישות ושימושי מערכת – Use Cases	3.1
13	סביבת הפרויקט – Eco – SYSTEM	3.2
14	ארכיטקטורת המערכת	3.3
23	טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת	3.4
25	ממשק משתמש – GUI	4
34	מדריך למפתח	5
34	דיאגרמת UML של כל מחלקות הפרויקט והתלויות ביניהן	5.1
35	רשימת פונקציות ומחלקות ותפקידיהם	5.2
39	סיכום אישי / רפלקציה	6
40	מקורות מידע / ביבליוגרפיה	7
41	קטעי קוד	8

1 מבוא – תיאור נושא הפרויקט

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

באפליקציה לא ניתן לאנשים לפרסם קבצים שונים מתמונות, סרטונים או טקסט.

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

יש למשתמשים אפשרות להיכנס לצאט פרטי עם משתמש מסוים רק אם שניהם עוקבים אחד אחרי השני.

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

האפליקציה שומרת את כל הפוסטים שהמשתמשים העלו ותגובות, שומרת נתונים על משתמשים – סיסמא, שם משתמש, מייל, שומרת רשימת כל האנשים שהמשתמש עוקב אחריהם ורשימת כל הבקשות לעקוב וכל המשתמשים שעוקבים אחרי המשתמש (בנוסף לתאריך עדכון פוסטים).

בנוסף לכך ששומרת משתמשים בעת ההרשמה לפני תהליך verification ולאחר שהושלם נחקים. האפליקציה שומרת גם צ'אטים פרטים – את כל ההודעות שנשלחו.

האפליקציה משתמשת בקריפטוגרפיה על מנת להבטיח סודיות, אימות ושלמות המידע – האתר ב https. באפליקציה יש הגנות שונות – man in the middle, xss, sql injection ומימושים בפרויקט – הרחבה על איך כל אחד מההגנות בא לידי ביטוי בפרויקט נמצאת בהמשך הספר.

בחרתי בנושא כי רציתי להתעמק בבניית אתרים ובטיפול בבקשות בפרוטוקול https בנוסף לכך שעניין אותי לממש פרויקט עם מאגר נתונים גדול ולראות איך מטפלים בדבר.

1.1 נושא המחקר בפרויקט

נושא המחקר הינו big data ויותר ספציפית 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. המידע צריך להישמר בכמויות גדולות ושיהיה יחסית פשוט ויעיל להכניס לשנות ולהוציא מידע.

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 פופולארי וישנם ספריות מתקדמות רבות בפייטון הממשות נושא זה.

מהמחקר הסקתי שאני אשתמש בSQLite, הוא בסיס נתונים היוכל להחזיק כמות גדולה של מידע (מספיק מידע לפרויקט). בנוסף לכך במדולים של Django על מנת לשמור פוסטים – Django models database

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

אני אוסיף לשימוש בsqlite גם שימוש בהashing לשמירת פרטים מסוימים כמו סמאות לדוגמא של משתמשים על מנת להעניק עוד סודיות לפרטיהם.

2 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 לפונקציות של דברים רבים CSS – משמש לעיצוב האתר.

העבודה תוכנתה ב PyCharm והשתמשה ב SQLite

ספריות:

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

Pathlib – עוזר לשימוש בשינוי כתובות של קבצים וגישה למיקום של הקובץ 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.

os – יכול לשמש לבדיקת קיום של קבצים, של כתובות בנוסף לכך שיכול גם למחוק קבצים וליצור גם תיקיות – משמש בשמירת הקבצים בפרויקט ומחיקת קבצים מסוימים בעיקר.

datetime – לשמירת תאריך העדכון האחרון של הפוסטים והשוואה עם תאריך העדכון האחרון של המשתמש (לדעת אם יש צורך בעדכון המשתמש)

base64 – על מנת לשלוח קבצים כמו תמונות וסרטונים

hashlib – הצפנת hash של סיסמאות לפני שמירת בדאטאבייס – מהווה הגנה נוספת

smtplib – לשלוח אימייל ביצירת משתמש – על מנת לאשר את כך שהמייל שהוכנס ב sign up pages הוא המייל של המשתמש.

2.1 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, אימייל, 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). (שימוש נוסף בפירוט בהמשך)

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 ושימוש בMany 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 בcookies redirects – זאת על מנת שיהיה וידוי שהמשתמש קודם נתן את פרטיו ורק אז יוכל להשתמש באתר ושיהיה ניתן לזהות את המשתמש.

בפרויקט יש גם שימוש בדרכי תקשורת של https כמו למשל שליחת סטאטוס קוד בתגובה – כמו 200 OK או 304 Not Modified וגם במקרים מסוימים 404 Not Found.

יש שימוש גם בפרוטוקולים מומצאים לצורך עדכון הדאטאבייסים או בקשת מידע מסוים – מצד הקליינט הבקשות נשלחות באמצעות javascript בשימוש בajax בעיקר לשליחת ההודעות (דוגמא לפרוטוקולים מומצאים בהמשך הספר)

נעשה שימוש בהצפנת hash בשמירת סיסמאות בדאטאבייס כדי שגם אם פורץ מגלה מה רשום בדאטאבייס הוא לא יקבל גישה לשום מידע חשוב.

ישנו שימוש נוסף בשליחת מייל למשתמש על מנת לאשר שהמייל שאוכנס במסך ההרשמה הוא הכן המייל של המשתמש.

בסופו של דבר הטכנולוגיה חדישה ומשומשת הרבה לתחומים שבה עוסקת.

הפרויקט עוסק בתקשורת בין השרת לקליינטים בעלת הצפנה והגנה מהתקפות שיצוינו בהמשך.

פירוט יכולות ושימושים של היכולות:

בכל אחת מהיכולות יש שימוש בתקשורת מאובטחת ואמינה מעל פרוטוקול SSL (HTTPS) ולכל יכולת יש דף (GUI) משלה. בנוסף לכך שבכל יכולת יש שימוש בforms ובדאטאבייס לשמירת נתונים. בנוסף יש אבטחה בכל אחת מהיכולות מחסל sql injection.

ההרשמה: השימוש ב-databases של UnvalidatedUsers, (sqlite) בנוסף תהליך ההרשמה כולל בתוכו את תהליך ההverification שלו יש דף משלו ומשתמש בcookies (עבור מי שהכניס את הפרטים ועוד לא סיים את תהליך ההverification) ואף משתמש בsmtp על מנת לשלוח מייל לקליינט שמכיל את הפרטים שצריך להכניס על מנת לסיים את תהליך ההverification. לאחר תהליך ההverification מוכנסים הפרטים של המשתמש לממשק נתונים של הusers. תהליך ההרשמה בנוסף כולל בתוכו בדיקת פרטים (מגבלות של שם משתמש וססמא – כגון מספר אותיות מינמלי).

הכניסה: השימוש בממשק הנתונים של הusers על מנת לבדוק את הפרטים של המשתמש, ואף כאשר המשתמש מזין פרטים נכונים – מקבל cookie עם שם המשתמש שלו כך שיוכל להיכנס לאתר.

האתר הראשי: השימוש בממשק הנתונים של הusers וposts בנוסף לcomments, הכל בשביל הפוסטים ובשביל ניהול מערכת העוקבים. השימוש בxmlhttp request עבור עדכונים או בקשות שונות.

חדר צאט: משתמש בדאטאבייס משלו של הודעות –private_messages, גם השימוש כאן ב xmlhttp request לעדכונים בנוסף יש שימוש בשליחת הודעות כמובן.

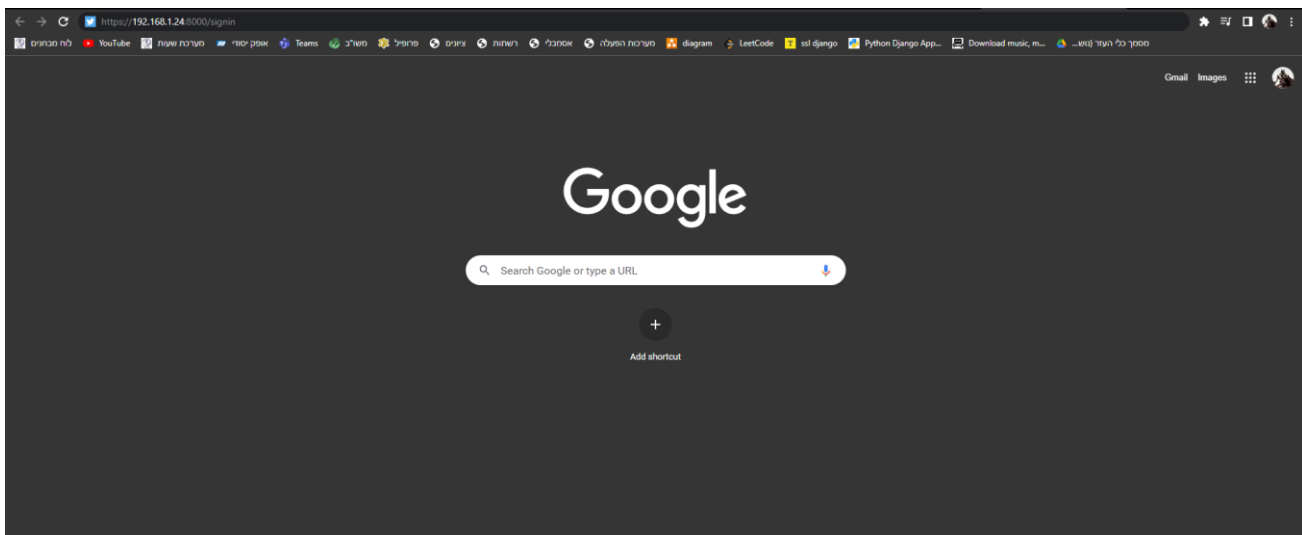
2.2 מדריך למשתמש

צד לקוח:

על מנת לגשת לאפליקציה יהיה צורך רק ללכת ל search engine כמו גוגל למשל ולהכניס את הכתובת שהיא הקן 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:

לאחר התקנת קבצי השרת, בשימוש בPyCharm וpython 3.10 (אם אין יש צורך בהתקנה) יש לפתוח את אחד הקבצים, להוריד Django דרך הטרמינל באמצעות הפקודות (והורדת דברים נוספים):

```
pip install django-extensions, pip install Werkzeug pip install django
```

```
pip install pyOpenSSL, pip install pytz python -m pip install Pillow
```

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 כדי להריץ באמצעות http.

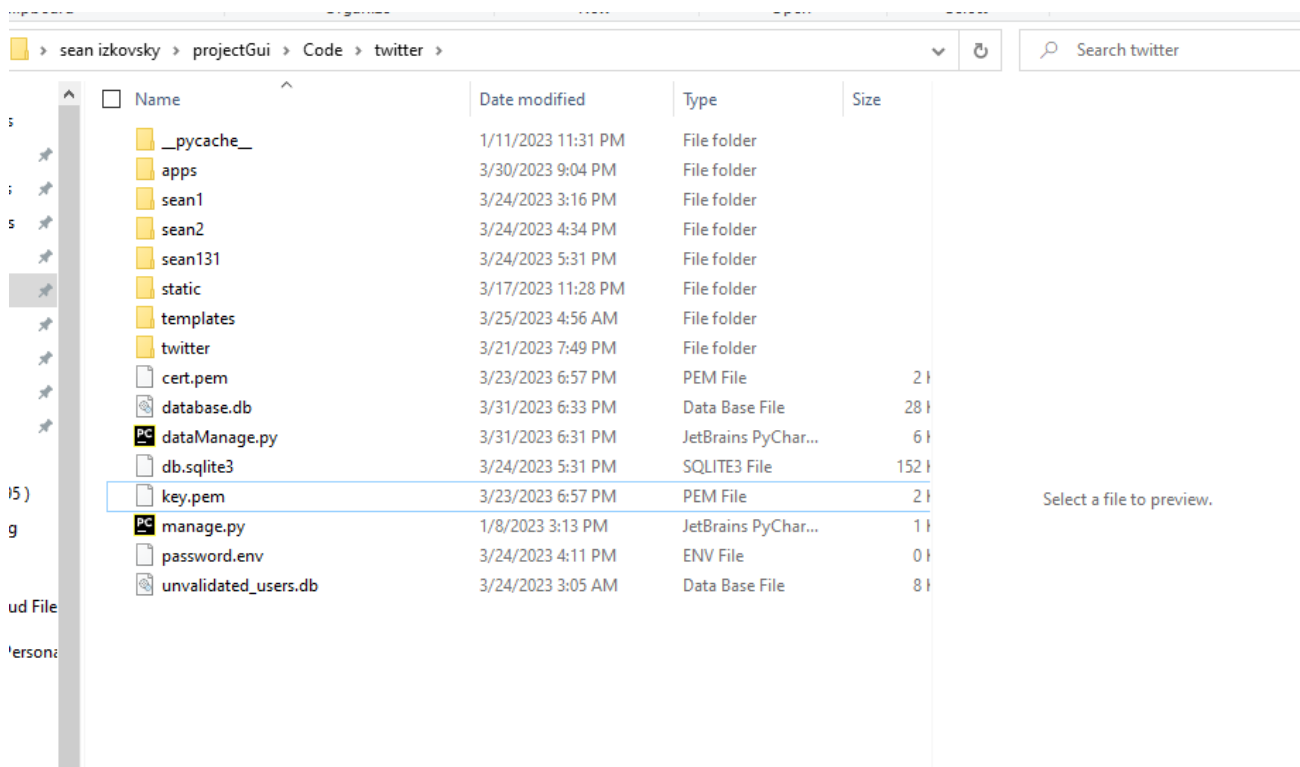
על מנת להריץ באמצעות https יש להריץ את הפקודה: `python manage.py runserver_plus --cert-file cert.pem --key-file key.pem 0.0.0.0:8000` רק כאשר קיימים הקבצים: `cert.pem` ו-`key.pem`, כלומר צריך קודם ליצור או לייבא חתימה דיגיטלית ומפתח להצפנה.

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 הקו צריך להיות הקו של המחשב (מדובר על הרצה ברשת מקומית).

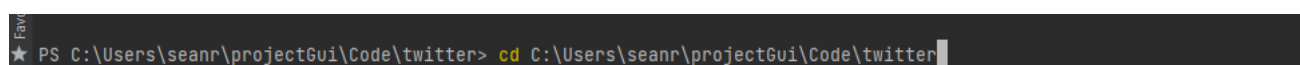
בשביל הדאטאבייס של Django: `python manage.py makemigrations python manage.py migrate`

תמונות להמחשה:

דוגמא למיקום של root folder:



דוגמא לניווט דרך הטרמינל:



ככה השרת אמור להיראות כאשר רץ (בhttps):

```
PS C:\Users\seanr\projectGui\Code\twitter> python manage.py runserver_plus 0.0.0.0:8000 --cert-file cert.pem --key-file key.pem
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on https://127.0.0.1:8000
* Running on https://192.168.1.24:8000
Press CTRL+C to quit
* Restarting with stat
Performing system checks...

System check identified no issues (0 silenced).

Django version 4.1.5, using settings 'twitter.settings'
Development server is running at https://0.0.0.0:8000/
Using the Werkzeug debugger (http://werkzeug.pocoo.org/)
Quit the server with CTRL-BREAK.
* Debugger is active!
* Debugger PIN: 846-676-675
```

דוגמא ליצירת חתימה ומפתח:

```
Command Prompt

Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\seanr>mkcert -cert-file cert.pem -key-file key.pem 192.168.1.24

Created a new certificate valid for the following names
- "192.168.1.24"

The certificate is at "cert.pem" and the key at "key.pem"
It will expire on 4 July 2025

C:\Users\seanr>
```

דוגמא להורדת Django עם pip install
(התגובה תראה כך כאשר Django כבר מורד)

```
Terminal: Local x + v
* Debugger is active!
* Debugger PIN: 846-676-675
PS C:\Users\seanr\projectGui\Code\twitter> pip install django
Requirement already satisfied: django in c:\users\seanr\projectgui\venv\lib\site-packages (4.1.5)
Requirement already satisfied: asgiref<4,>=3.5.2 in c:\users\seanr\projectgui\venv\lib\site-packages (from django) (3.6.0)
Requirement already satisfied: sqlparse>=0.2.2 in c:\users\seanr\projectgui\venv\lib\site-packages (from django) (0.4.3)
Requirement already satisfied: tzdata in c:\users\seanr\projectgui\venv\lib\site-packages (from django) (2022.7)
WARNING: You are using pip version 21.1.2; however, version 23.0.1 is available.
You should consider upgrading via the 'C:\Users\seanr\projectgui\venv\Scripts\python.exe -m pip install --upgrade pip' command.
PS C:\Users\seanr\projectGui\Code\twitter>
```

3 אפיון דרישות וארכיטקטורת מערכת

3.1 דרישות ושימושי מערכת – Use Cases

הדרישה לקבל מידע רב מהמשתמשים 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).

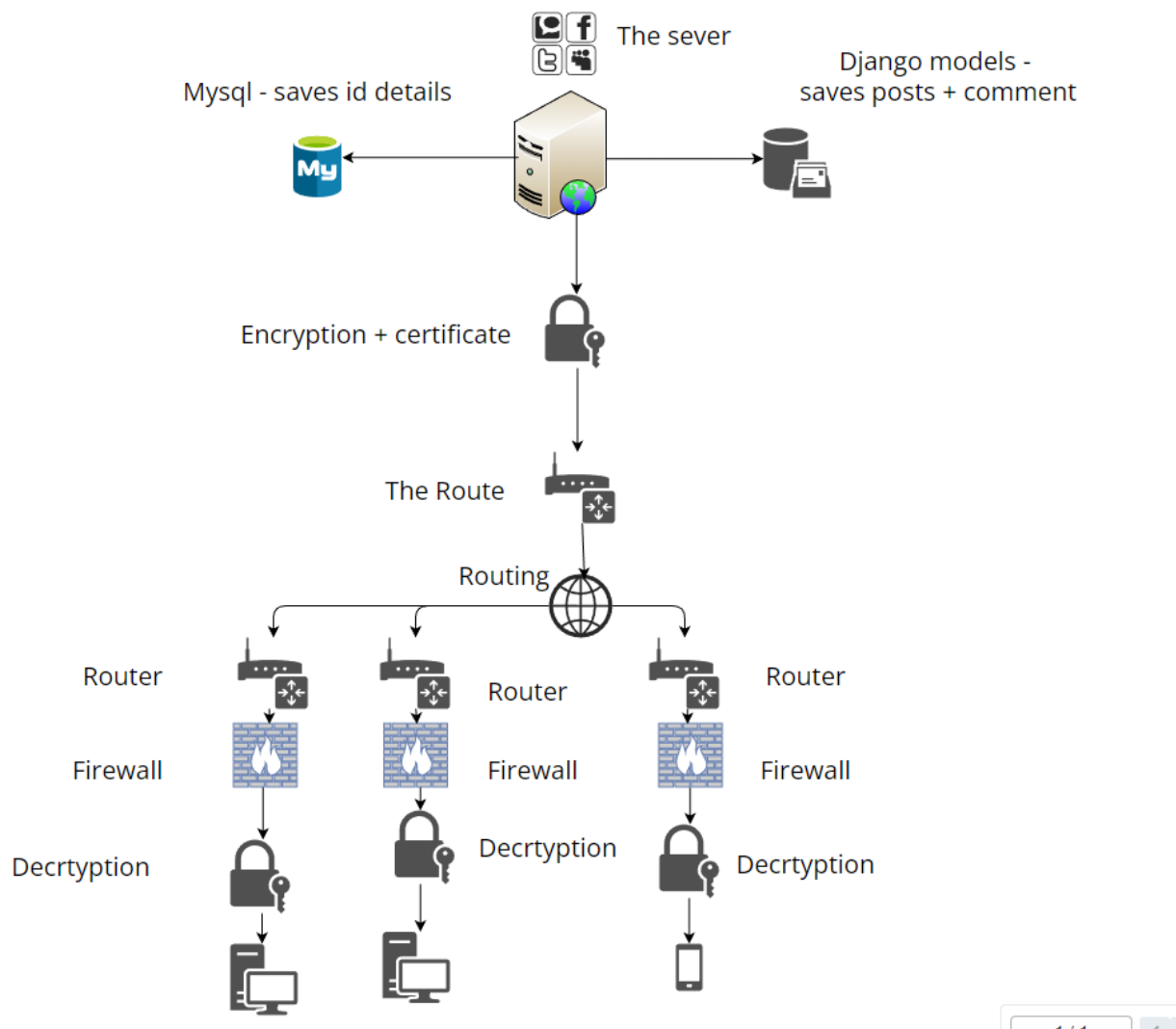
בנוסף, התוכנה צריכה להיות ברורה ונוחה למשתמש – המשתמש יוכל להבין ישר איך לעשות כל דבר מהאופציות של התוכנה בלי לקרוא יותר מדי רק לפי מבנה האתר. האפליקציה צריכה גם להיות מאובטחת ואמינה – המשתמש צריך לדעת שהמידע שלו בטוח ואין אנשים שרואים את המידע שהוא שולח לאתר ללא רשותו.

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

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

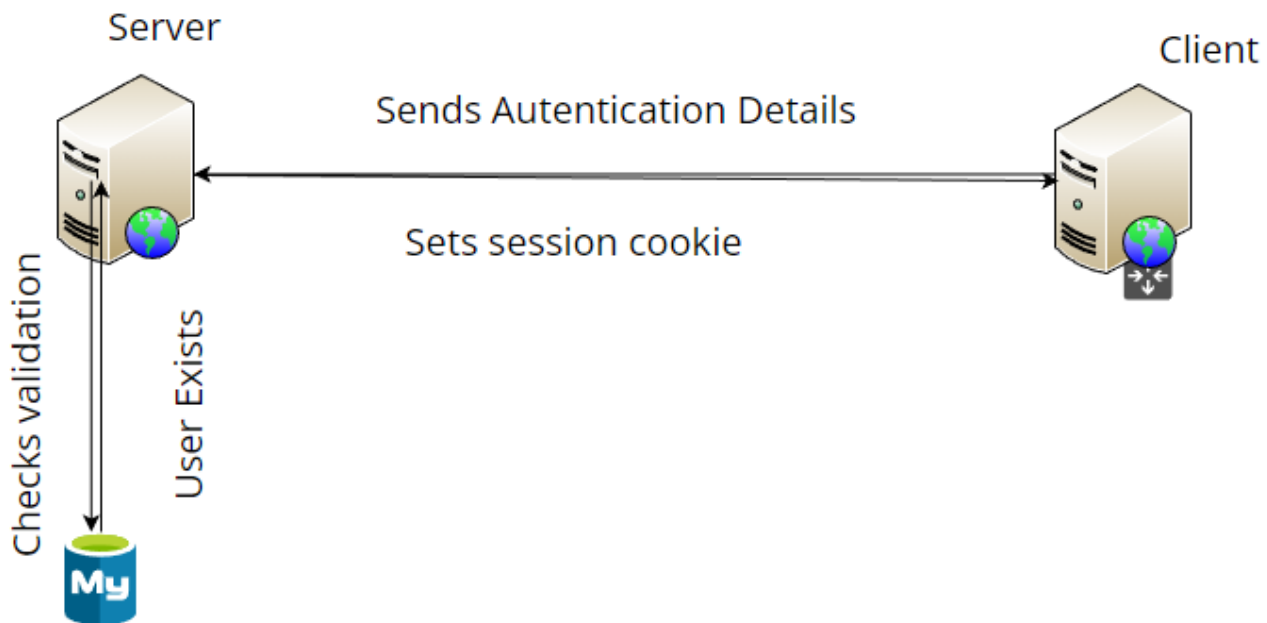
בנוסף, יש אפשרות להגיב לפוסטים גם דבר הצריך לשמור. לסיכום הדרישה העיקרית והכי כבדה היא שמירה וניהול מאגרי מידע גדולים.

3.2 סביבת הפרויקט – Eco – System



בתרשים מתואר גם את הדאטאבייסים (אך ללא התייחסות לצאט הפרטי ולמסך ההתחברות לכן שניים מהדאטאבייס לא נמצאים בתרשים – בנוסף גם יכול להיות firewall בין המחשב של השרת לראוטר שלו).

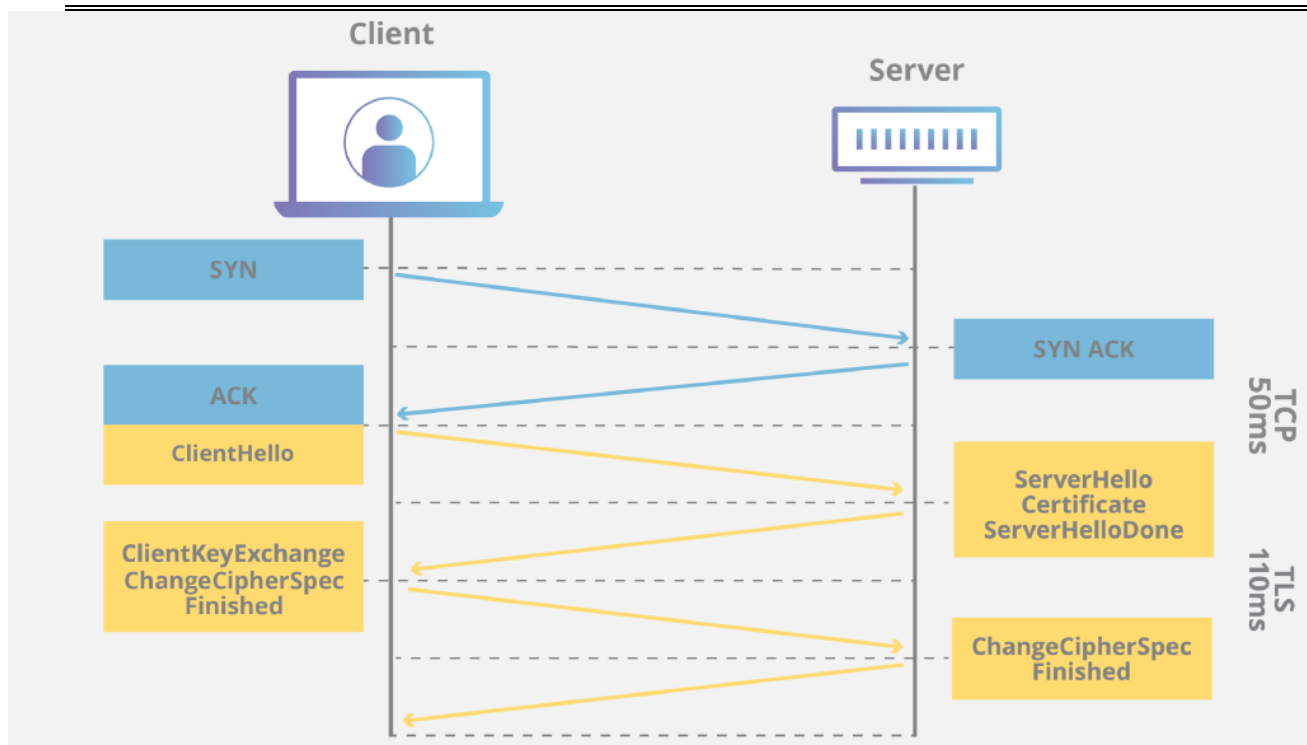
3.3 ארכיטקטורת המערכת



התמונה ממחישה את הכניסה לשרת, הקשר נעשה מעל פרוטוקול הקריפטוגרפי TLS בשימוש ב self signed certificate ו public key ו private key.

פרוטוקול משומש: פרוטוקול https. בפרוטוקול יש שימוש ב TLS להעברת המידע והחיבור ונעשה מעל TCP. מתרחשת TCP HANDSHAKE והחלפת המפתח הפומבי באמצעותו מוצפן הקשר (באמצעות הקליינט יוכל לשלוח את private key שהוא אסימטרי ובאמצעותו מוצפן הקשר לפחות בגרסאות הישנות יותר של TLS) וגם בנוסף לכך נשלחת ה certificate של השרת כדי שהקליינט ידע שהוא באמת מדבר עם השרת ולמנוע man in the middle.

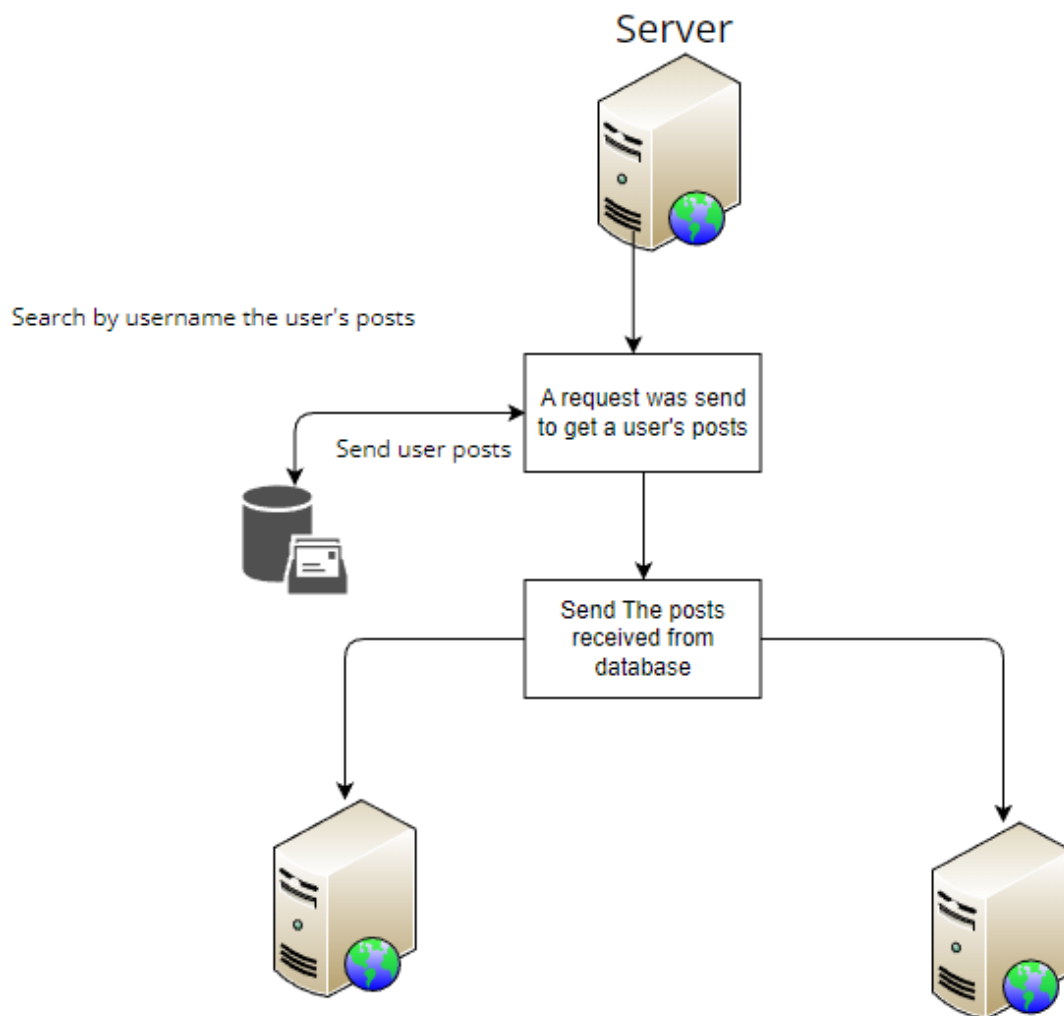
לאחר השמת cookie היא נשלחת עבור כל בקשה של המשתמש ונבדקת כל פעם. רק לאחר זמן מסוים שנקבע בקוד ה cookie מתפוגג.



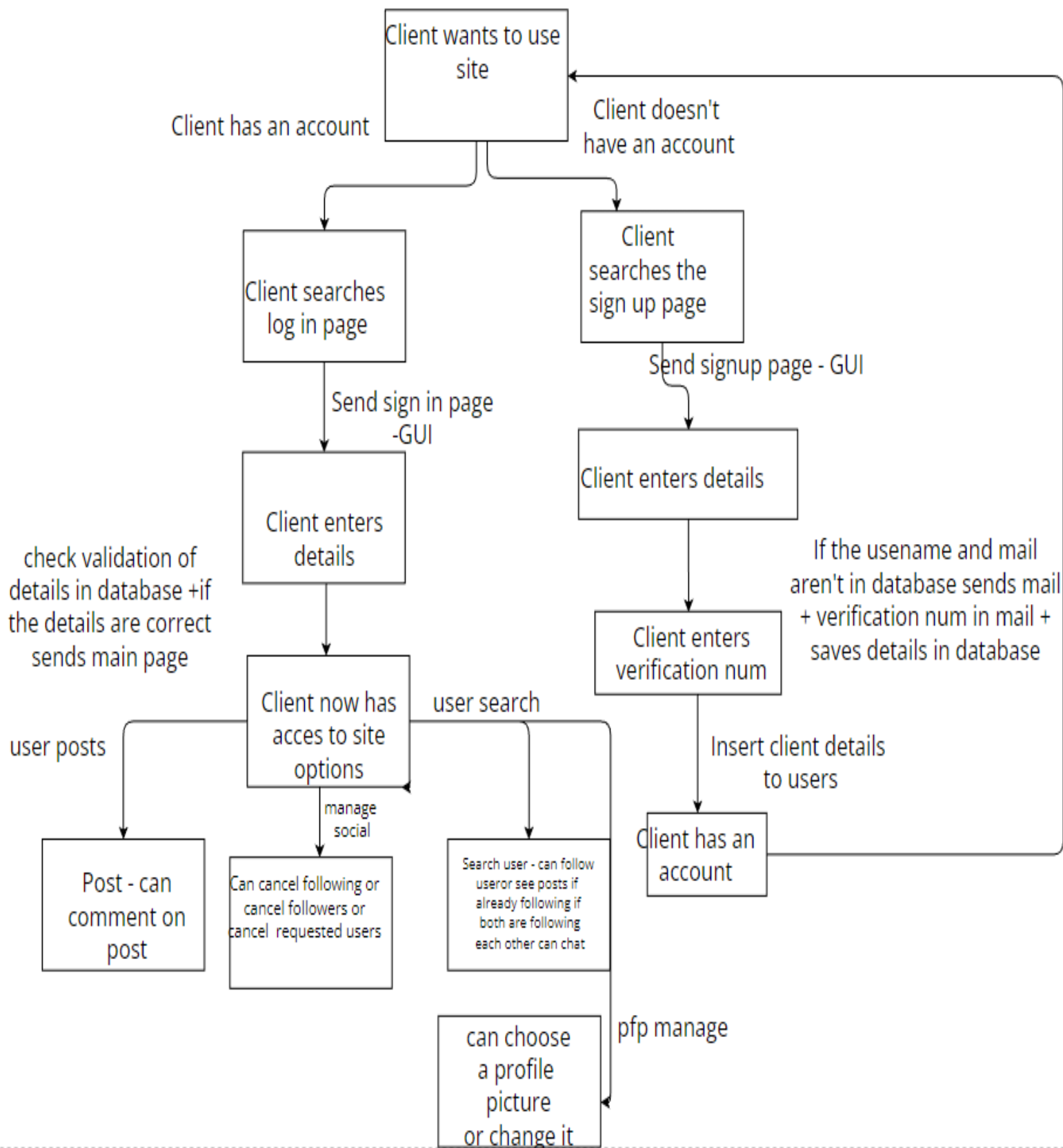
תחילת חיבור - TLS ניתן לראות בחיבור החלפת public key המשתמשים בו לאורך הקשר + ה tcp handshake בתחילת הקשר.



בקשה ותגובה לדוגמא - של מסך ההתחברות.



התמונה ממחישה מה קורה כאשר משתמשים מביקשים 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).



המודלים העיקריים בפרויקט

הסבר על המודלים:

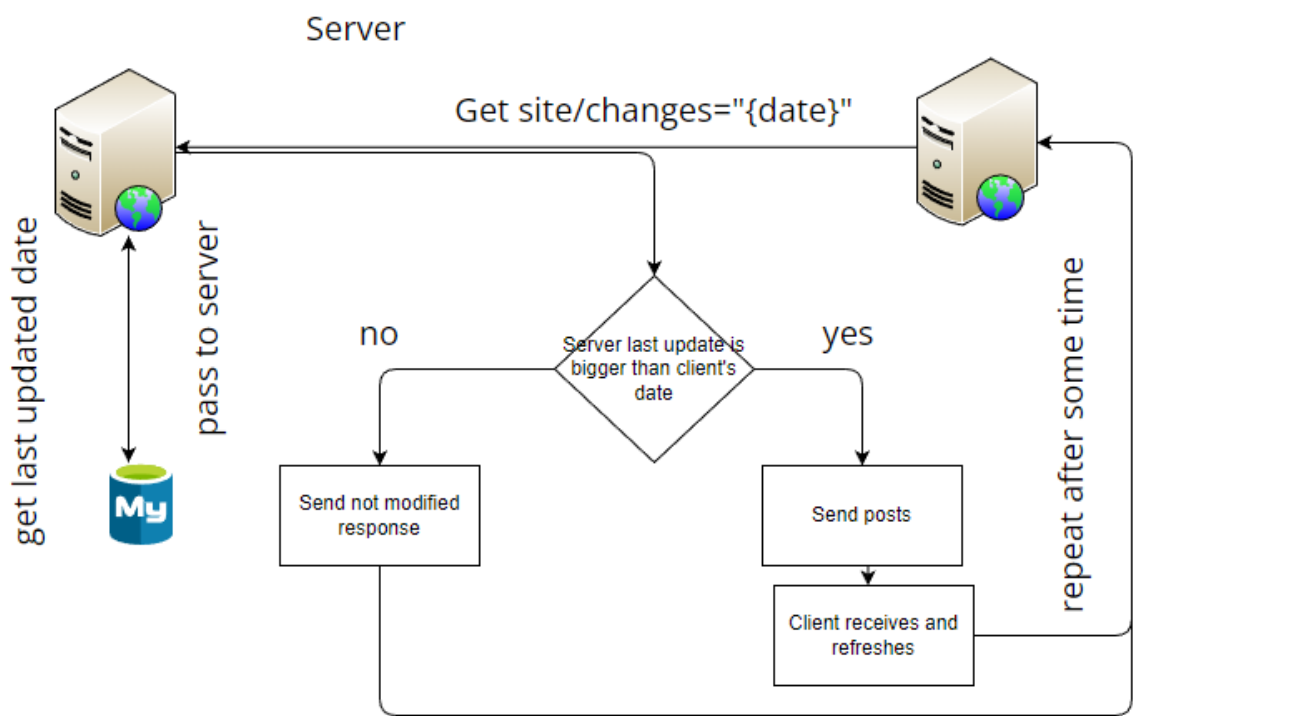
מודל הרשמה לשרת: המודל משמש ליצירת משתמש עבורו יתחברו לשרת בהמשך, המשתמש צריך להיות בעל mail קיים. הפרטים של המשתמש ישמרו בדאטאבייס, אם המשתמש כבר קיים בדאטאבייס או האימייל או השם משתמש שלו קיימים אין לתת ליצור משתמש חדש..

מודל כניסה לשרת: התחברות עבור המשתמש שנוצר – לאחר ההתחברות ניתן לקבוע את השם של אותו משתמש ולשייך אליו את הפוסטים והצ'אטים שמיועדים אליו. ישנה בדיקה האם המשתמש קיים לפי הדאטאבייס.

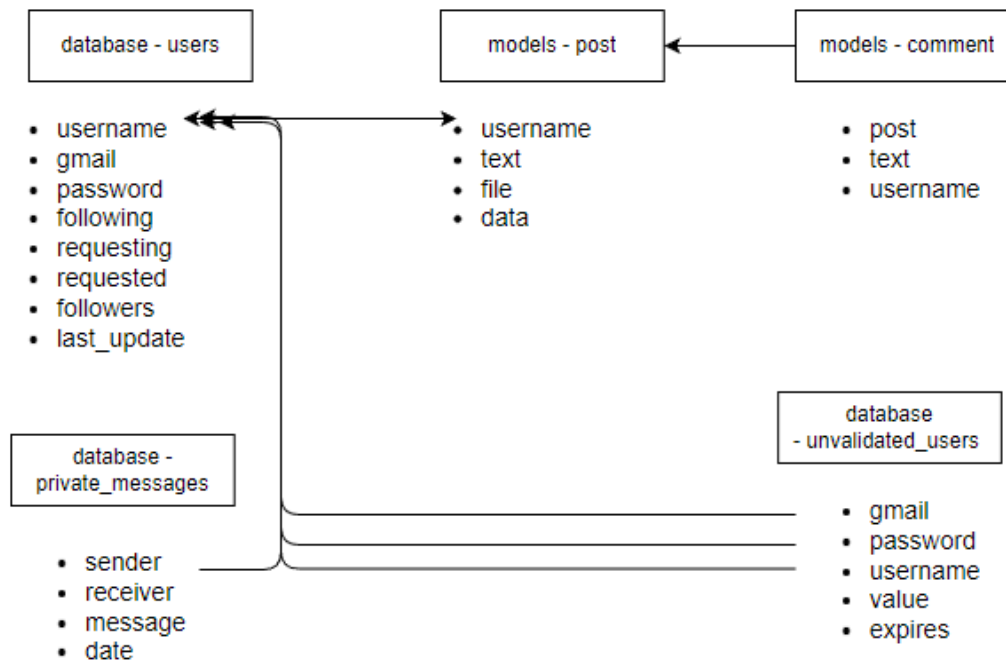
מודל שליחת הפוסטים: שליחה של כל הפוסטים של משתמש מסוים שמסודרים לפי התאריך (החדשים יותר למעלה).

מודל ניהול בקשות: שליחת בקשות לעקוב ושמירתם במאגר מידע, אותו דבר על דחיית אותן בקשות ושינויים כאלו.

מודל ניהול פוסטים: הוספת תגובות לפוסטים, מחיקת פוסטים, שמירת פוסטים צריך להעלות גם את העדכונים לכל המשתנים ולשנות בדאטאבייסים המתאימים את המידע הנחוץ.



התמונה ממחישה את מודל העדכון של הפוסטים מהצד של הקליינט.



במשתמשים נשמרים כמה דברים: username – שם המשתמש, gmail – מייל המשתמש וpassword שכולם משתמשים על מנת זיהוי המשתמש – יש גם שימוש בהצפנת hash של md5 בשמירה לדאטאבייס. 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 אחר 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.

בפוסטים נשמרים – שם המשתמש אליו משייך הפוסט, הטקסט הכתוב בפוסט, הקובץ שנבחר הurl שלו ודברים נוספים אליו date תאריך שמתעדכן אוטומטית.

ובתגובות נשמר id של הפוסט, הטקסט שרשום בתגובה ושם המשתמש שרשם את התגובה. כל אחד מהמאגרי מידע שצוינו מנוהל באמצעות קשר של רבים אל אחד – 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 המשיכות 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 id של הפוסט (בDjango זה נקרא foreign key).

במשתמשים הלא רשומים נשמר המייל, סיסמא, ושם המשתמש – לאחר שהתהליך הוריקציה הושלם אותם פרטים נכנסים לדאטאבייס של המשתמשים בנוסף גם נשמר ערך מספרי מסוים שהוא המספר

שהמשתמש צריך להכניס כדי לאשר שזה המייל שלו ותאריך ונשמר גם תאריך ההתפוגגות של המשתמש (כי המשתמשים האלו זמניים עד שהמשתמש יאשר שזה המשתמש שלו)

דאטאבייס ההודעות הפרטיות מכיל בתוכו שולח, ואדם שנשלח אליו – שניהם חייבים להיות משתמשים רשומים במערכת, הודעה ותאריך.

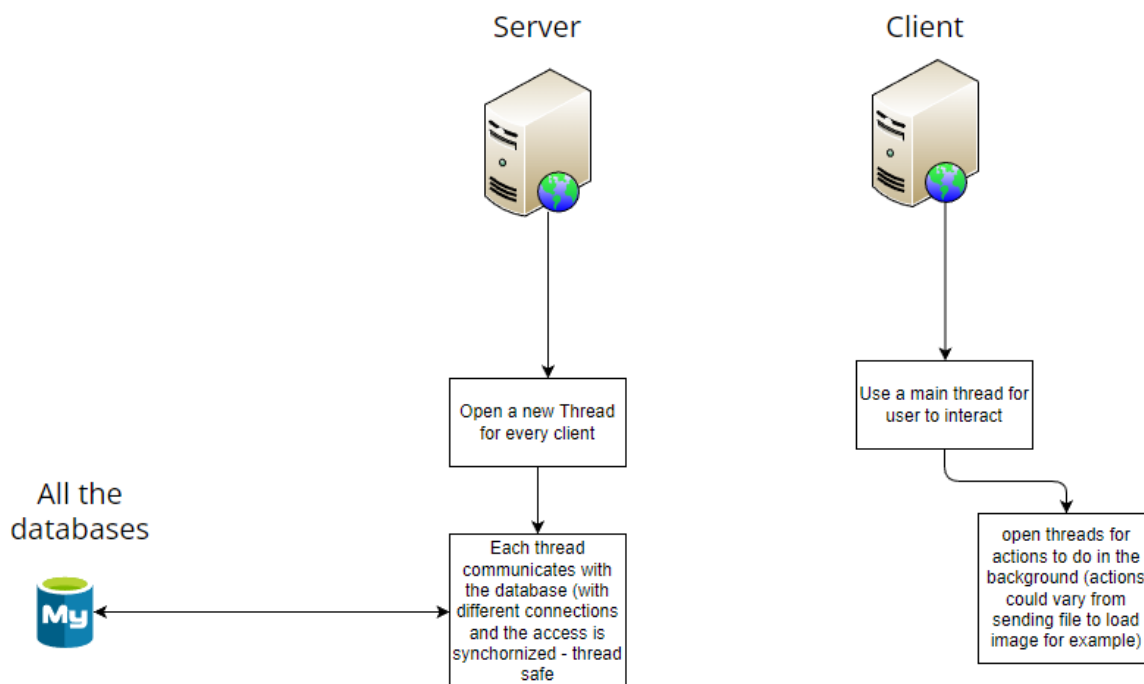
העניין של אבטחה כאן הוא נגד sql injection והאבטחה היא שימוש בניהול מאגרי המידע באמצעות דרך מובנה בsqlite שמונעת sql injection. בשימוש בסימני שאלה איפה שהפרמטרים ובכך אין אפשרות לפריצה לsql injection.

עניין נוסף של אבטחה הוא בדיקת סוג הקובץ שנשמר בפוסטים בתמונת פרופיל והגודל שלו כך שלא יעמיסו על האתר.

בנוסף ישנה הגנה מxss – בשימוש בdjango ובautoescape שמשנה תווים שיכולים להיות מסוכנים בhtml אוטומטית כך שלא יוכלו להכניס וירוס בצורה זו

והגנה מman in the middle שנעשת באמצעות הצפנה וcertificate (פרוטוקול https – שימוש בhttps מעל TLS)

המבנה שמתקיים בפרויקט:



בפרויקט זה אין צורך בprocesses אלא רק בthreads הטרידים מתקשרים אחד עם השני רק בעזרת הדאטאבייסים בשרת – שהם כמובן thread safe, כאשר אחד הטרידים מעדכן משהו בדאטאבייס טרידים שישלחו את מה שעודכן ישלחו את הגרסה המעודכנת כמובן.

בצד של הקליינט לעומת זאת, מכיוון שהקליינט הוא בעצם browser כמות הטרידים תלויה browser – אך כמו שהוסבר בציור יש טריד אחד עיקרי וטרידים נוספים נפתחים לביצוע פעולות ברקע.

הסבר כולל על הפרוטוקולים:

כמו שצוין קודם ישנו שימוש בפרוטוקול https שמשתמש ב TLS ו tcp. האבטחה, העברת דפי html דרך החיבור ואימות שהמידע שהתקבל הוא הזה נשלח מתאימים לפרויקט וממומשים באמצעות הפרוטוקולים שציינתי בנוסף לשימושים בפרוטוקולים מומצאים.

יש שימוש בפרוטוקולים מומצאים שנמצאים בכמה מקומות לאורך הפרויקט. כמה דוגמאות לכך:

```
if request.method == "POST":

    if follow_name == user_name or len(runner.find_from_name(follow_name)) == ZERO_MATCHES: # user searched himself
        return enter_site(SITE, request)
    elif "comment" in list(request.POST.keys()):
        try:
            comment_post = Post.objects.get(id=int(request.POST["post"]))
            comment = Comment(post=comment_post, text=request.COOKIE.get('username') + ":" +
                               request.POST["comment"], username=request.COOKIE.get('username'))
            comment.save()
            return enter_site(SITE, request)
        except ValueError:
            return enter_site(SITE, request)
```

פרוטוקול של שמירה של תגובה. מהצד של השרת צריך להיות בקשת POST ובתוכה פרמטר של comment, הבקשה צריכה להישלח ל site/

כלומר מהצד של הלקוח:

```
http.open("POST", "/site/", true);
```

הסבר: POST – סוג תגובה, /site/ חלק מהכתובת שנשלח אליה, true – הבקשה לא מסונכרת – הקוד ממשיך גם אם לא קיבל תשובה (בשביל לא לגרום לבעיות).

```
var params = 'comment=' + post_data + '&post=' + data;
```

בפרמטר של הcomment נמצאת התגובה שנשלחה וגם יש פרמטר של post שבו נמצא id של הפוסט אליו משויכת בתגובה.

ישנם דוגמאות נוספות לפרוטוקולים מומצאים רבים שהשתמשתי בהם בפרויקט (בשימוש ב https) אז לא אראה את כולם אך אראה כמה נוספים:

```
if request.method == POST_REQUEST:
    if "accept_user" in list(request.POST.keys()):
        runner.remove_data("requested", str(request.POST["accept_user"]), request.COOKIE.get('username'))
        runner.remove_data("requesting", request.COOKIE.get('username'), str(request.POST["accept_user"]))
        runner.add_data("following", request.COOKIE.get('username'), str(request.POST["accept_user"]))
        runner.add_data("followers", str(request.POST["accept_user"]), request.COOKIE.get('username'))
    elif "decline_user" in list(request.POST.keys()):
        runner.remove_data("requested", str(request.POST["decline_user"]), request.COOKIE.get('username'))
        runner.remove_data("requesting", request.COOKIE.get('username'), str(request.POST["decline_user"]))
    elif "cancel_follower" in list(request.POST.keys()):
        runner.remove_data("following", request.COOKIE.get('username'), str(request.POST["cancel_follower"]))
        runner.remove_data("followers", str(request.POST["cancel_follower"]), request.COOKIE.get('username'))
```

גם כאן יש בקשת POST לשרת ועבור פרמטר `accept_user` יש הוספת משתמש שביקש רשות לעקוב אחרי משתמש מסוים – באמצעות שינוי הדאטאבייס, `decline_user` כדי לסרב למשתמש ששלח בקשה ו- `cancel_follower` - כדי לבטל עקיבה אחר משתמש מסוים.

,

3.4 טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת

מיושם	דרישה
שימוש בדאטאבייסים שיכולים להחזיק כמות גדולה של מידע (כמות מספיקה) – הקבצים עצמם נשמרים כל אחד בתיקיה של המשתמש – יכול לאחסן קבצים רבים אך הכמות תלויה בגודל איחסון המחשב	התמודדות עם כמות רבה של מידע לאחסן – ריבוי משתמשים ופוסטים
קישור בין הדאטאבייסים – שמירת שם המשתמש בכל פוסט ושיוך בכך של כמה פוסטים למשתמש אחד, ישנם פונקציות בנויות שאחראיות על הוצאת מידע ובדקות הרשאות.	ניהול המידע – התאמת המידע למשתמש ונתינת מידע לפי ההרשאות של המשתמש (אם משתמש מבקש מידע על משתמש אחר הוא צריך לעקוב אחריו)
השימוש בhttps כרוך בתוכו אבטחת מידע – האות s היא קיצור של secure – ובתוכה יש שימוש בפרוטוקול TLS – פרוטוקול אבטחה שמשתמש גם בהצפנה סימטרית ואסימטרית להעברת המידע.	הקשר צריך להיות מאובטח ומימושים של אבטחה נוספים
ישנה הגנה מס XSS – בשימוש בdjango ובescape אוטומטית שמשנה תווים שיכולים להיות מסוכנים בhtml אוטומטית כך שלא יוכלו להכניס וירוס בצורה זו	האתר צריך להיות בנוי כך שהמשתמש יוכל להבין את אפשרויות התוכנה מבלי להסתבך יותר מדי
הגנה נוספת מס sql injection בשימוש ב? בהכנסת ערכים – כך שמה שהוכנס לא יתרגם לכלום ויכנס פשוט כסרטינג	קבלת מידע מהמשתמש
השימוש בhtml, bootstrap בתוספת לשמות משמעותיים עבור האפשרויות והסברים שרשומים באתר הפכים את השימוש בו לפשוט מאד	
נעשה באמצעות Django forms, ajax, html froms כל אחד מהדברים האלה משמש לקבל מידע של דברים אחרים וביחד מקבלים את כל המידע שצריך מהמשתמש כגון – קבצים, טקסט או בקשת עקיבה ועוד	

בדיקות שנעשו בקוד:

בדיקת עדכונים -

בדיקה שההודעות ובקשות עקיבה מתעדכנות כמו שצריך אוטומטית בלי צורך לעשות refresh לעמוד. הבדיקה נעשתה באמצעות שליחת בקשת עקיבה למשתמש אחר והמשתמש האחר קיבל אותה לאחר כמה שניות ללא צורך בrefresh בנוסף גם הצאט וגם הפוסטים נראו שמתעדכנים אוטומטית לפי הבדיקה. הבדיקה הסתיימה בהצלחה.

בדיקת התחברות –

בדיקה שההתחברות והverification עובדים כמו שצריך ושאינן שגיאות גם כאשר מוכנסים פרטים לא נכונים ויש תגובה נכונה כאשר מוכנסים פרטים שגויים. הוכנסו פרטים לא תקינים ופרטי משתמשים לא לפי ההגבלות והפרטים נדחו כמו שצריך ונשלחו ההודעות שהיו אמורות להשילח לקליינט.

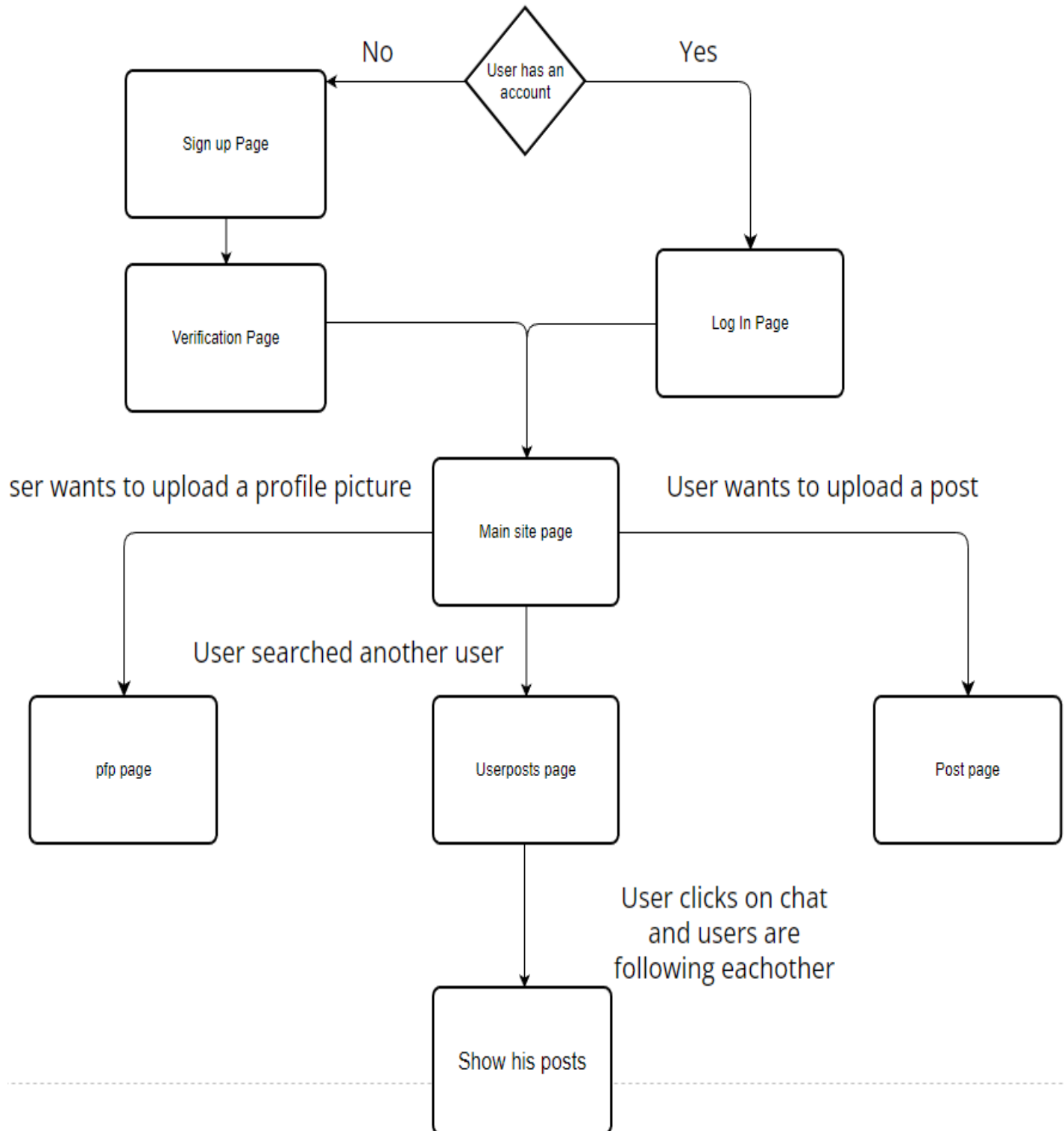
הבדיקה הסתיימה בהצלחה.

בדיקה ממכשיר אחר-

בדיקה שניתן להשתמש באפליקציה ממחשב אחר הבדיקה נעשתה באותו החל – נעשתה באמצעות התחברות דרך הטלפון לאתר. הבדיקה נכשלה – נלמד מהבדיקה כי יש קודם לגרום למכשיר לזהות את החתימה שהשתמשי בה מהצד של השרת. לאחר תיקונים זה עובד.

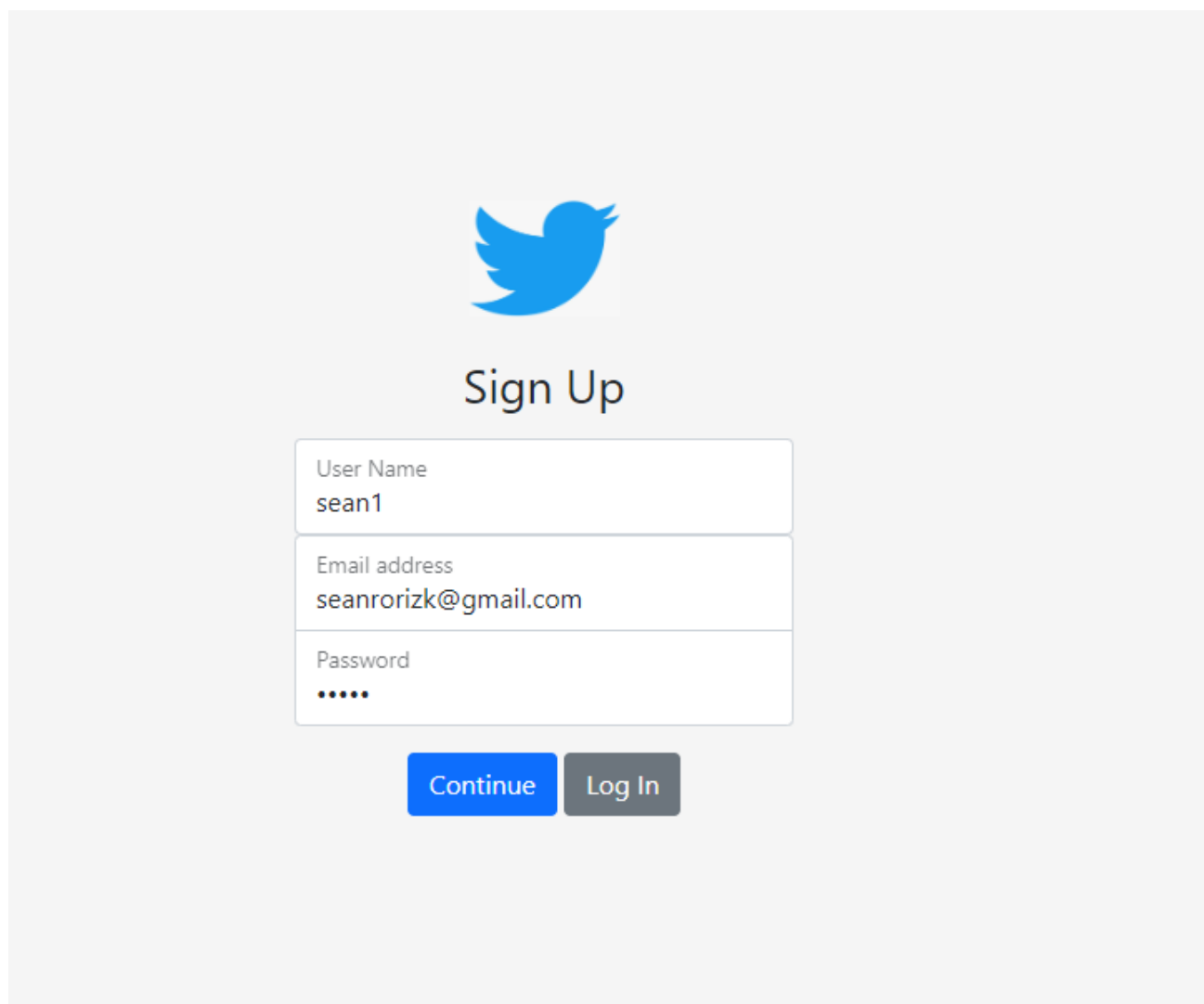
4 ממשק משתמש - GUI

הסבר כללי על המסכים: (הדפים השונים)



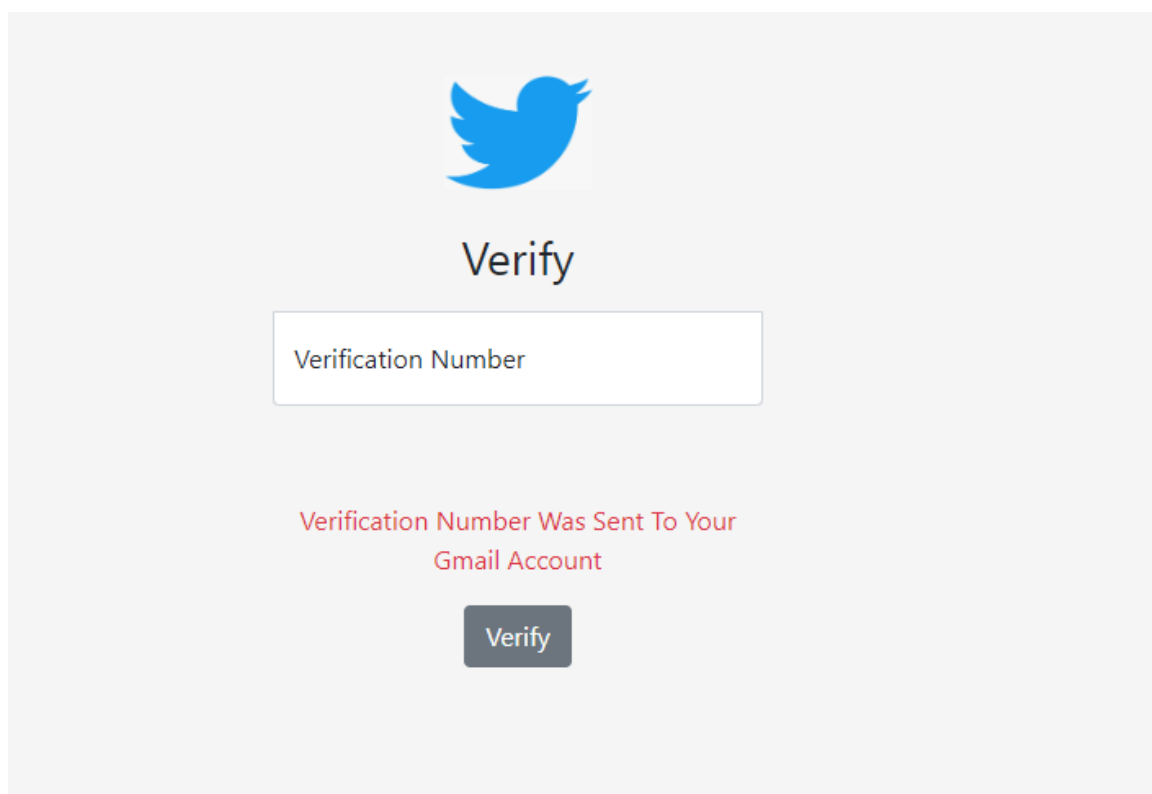
דוגמא להרשמה וכניסה לאתר ואז העלאת פוסט + בחירת תמונת פרופיל.

מסך הרשמה:

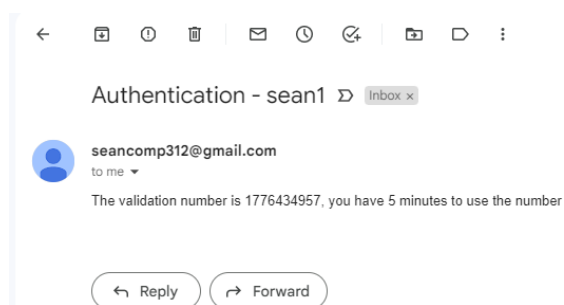


The image shows the Twitter sign-up interface. At the top center is the blue Twitter bird logo. Below it, the text "Sign Up" is displayed in a large, bold, sans-serif font. Underneath the text are three stacked input fields. The first field is labeled "User Name" and contains the text "sean1". The second field is labeled "Email address" and contains the text "seanrorizk@gmail.com". The third field is labeled "Password" and contains five dots. Below the input fields are two buttons: a blue button labeled "Continue" and a grey button labeled "Log In".

מסך הזדהות (verification)



המייל שנשלח



הפרטים נקלטו



Account Successfully
Created!

You can Log In now

מסך כניסה:



Sign In

Email address
exampleMail@gmail.com

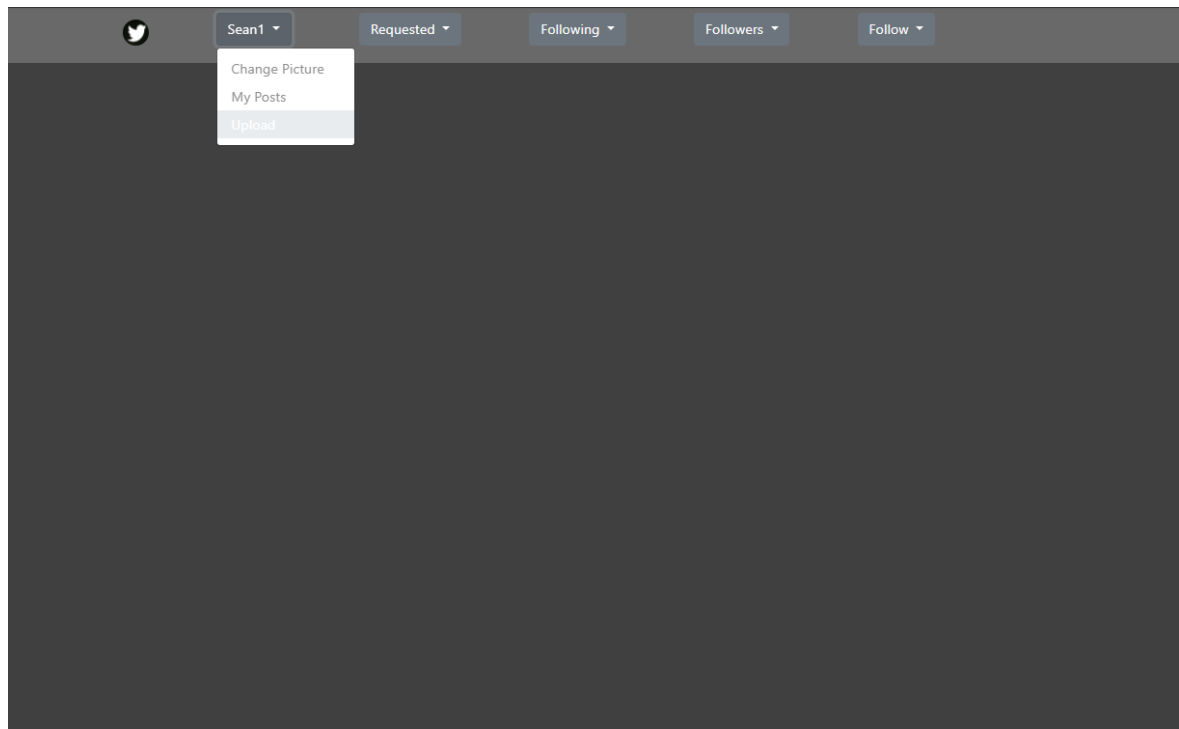
Password
.....

☐ Remember me

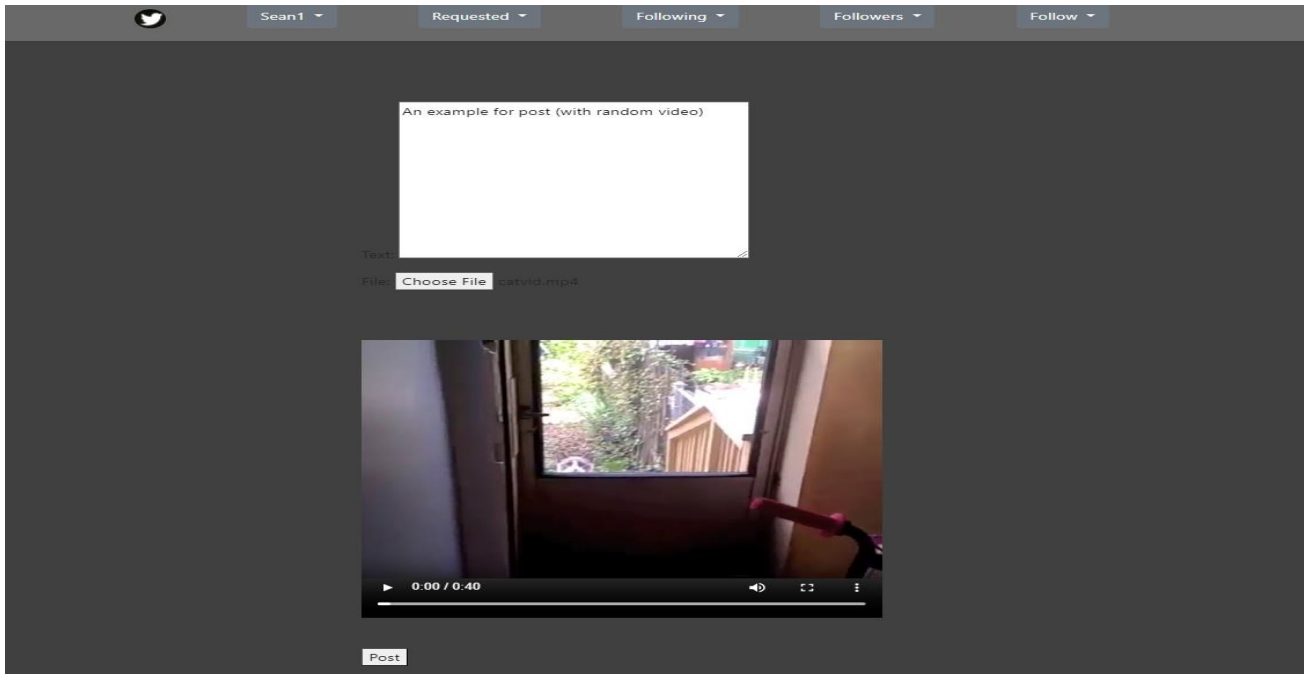
Continue

Sign Up

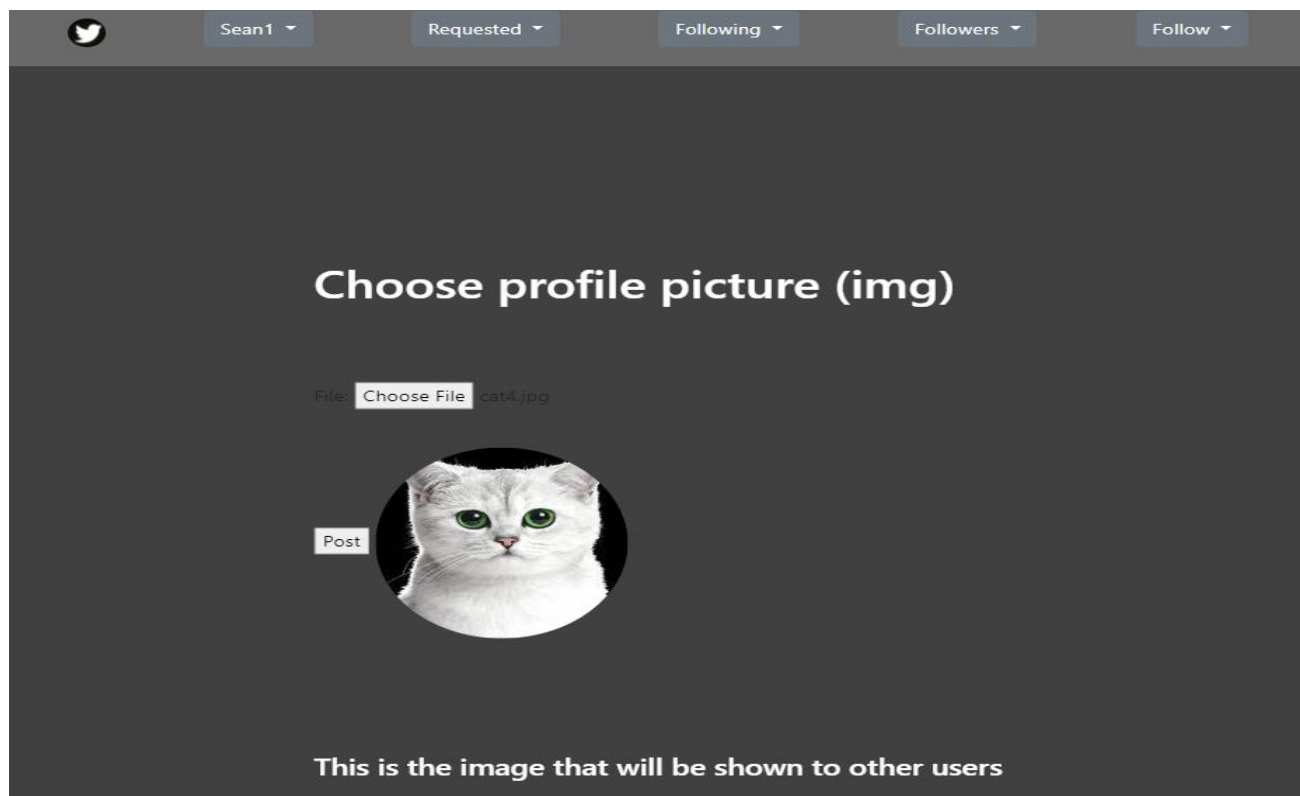
מסך האפליקציה הראשי (ריק כי אין למשתמש פוסטים)



העלאת פוסט:

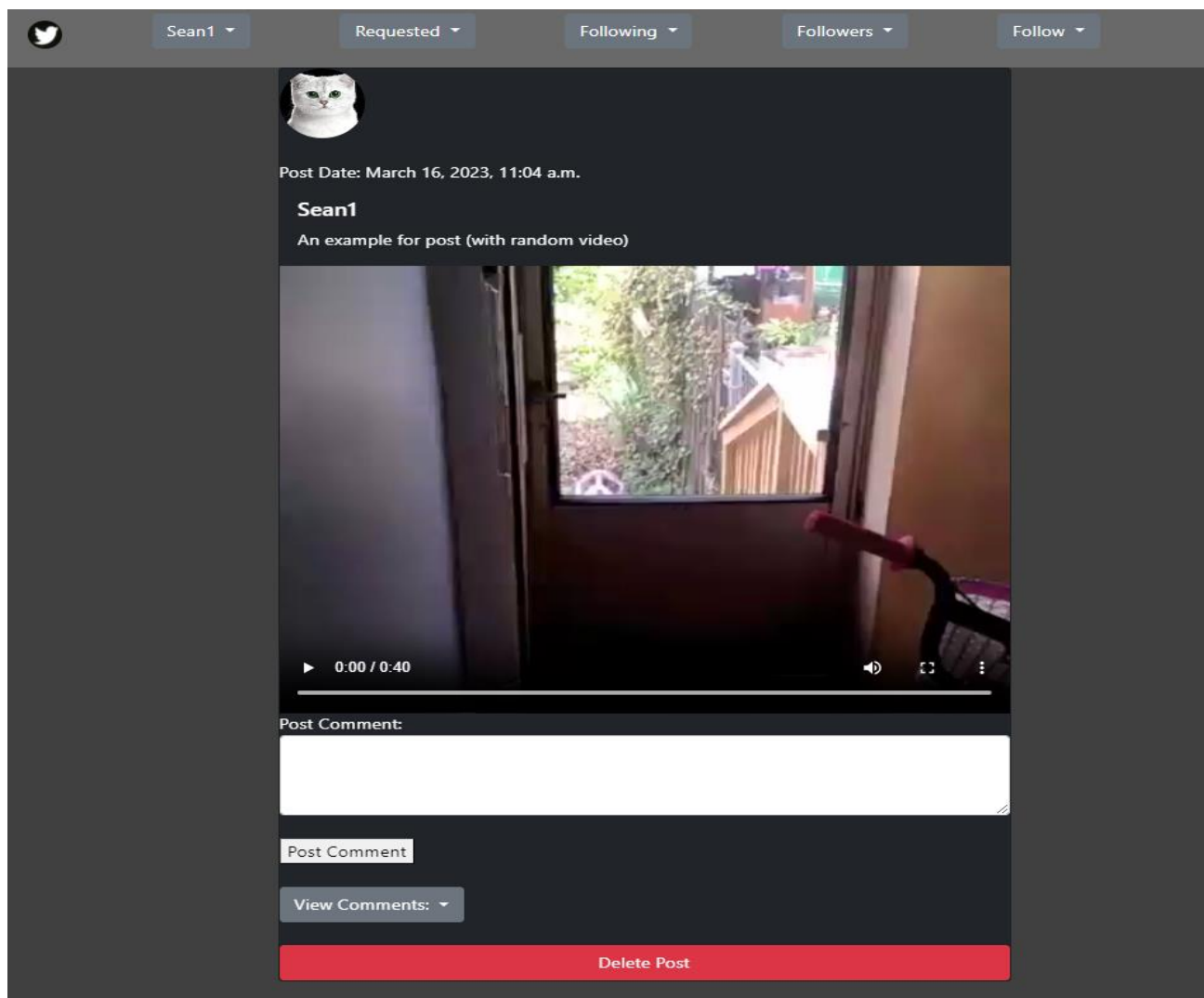


העלאת תמונת פרופיל:



מסך הפתיחה כעת:

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



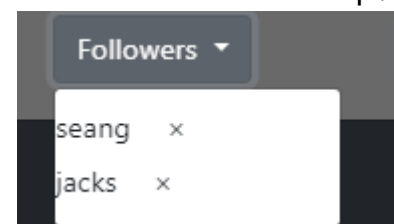
מערכת התגובות:

viktor:First comment
seang:second comment
seang:comment
seang:comment
seang:comment
viktor:comment
jacks:comment

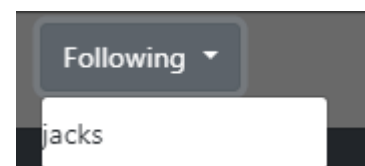
View Comments: ▾

Delete Post

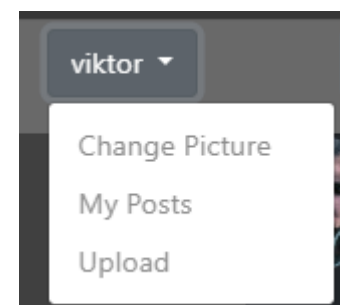
עוקבים:



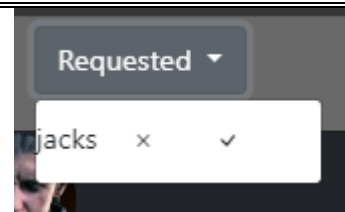
אנשים שהמשתמש עוקב אחרי:



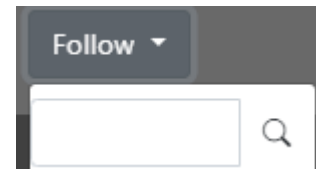
אפשרויות:



בקשה לעקוב:



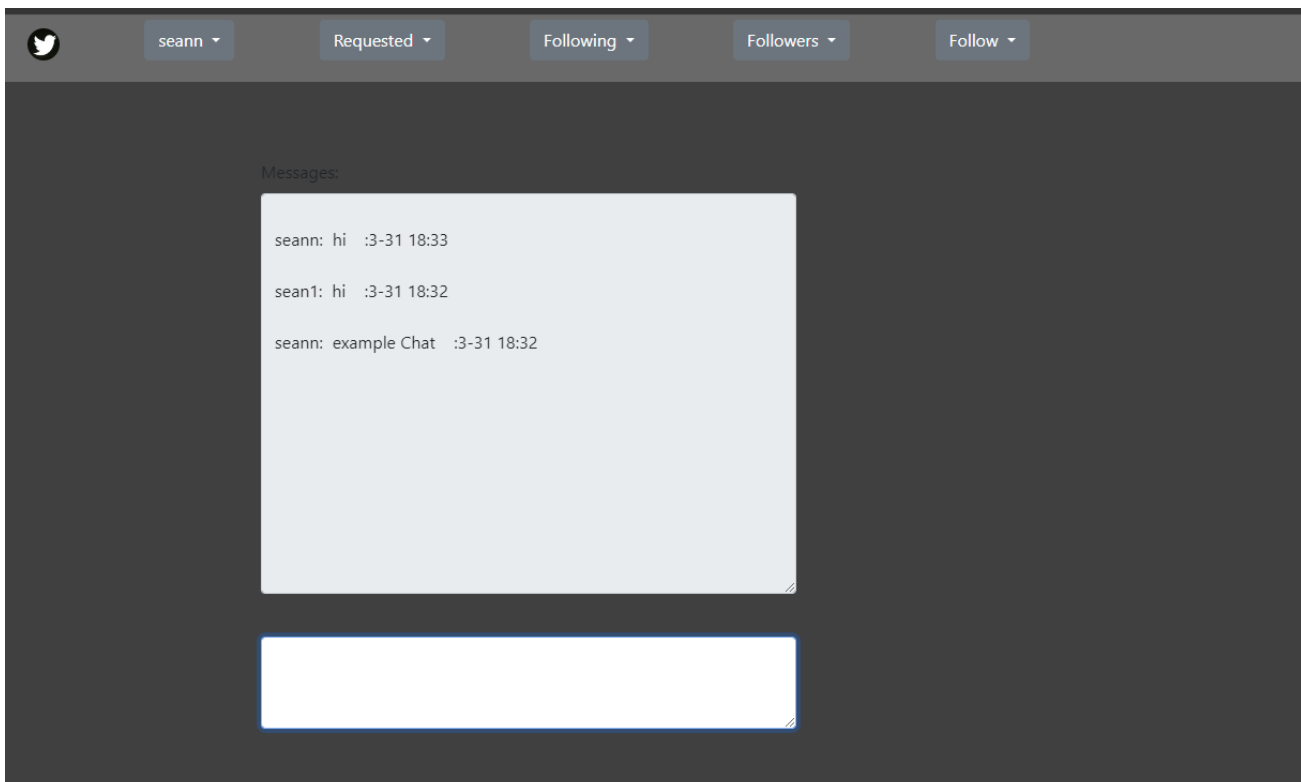
חיפוש משתמש:



אפשרות לchat פרטי:



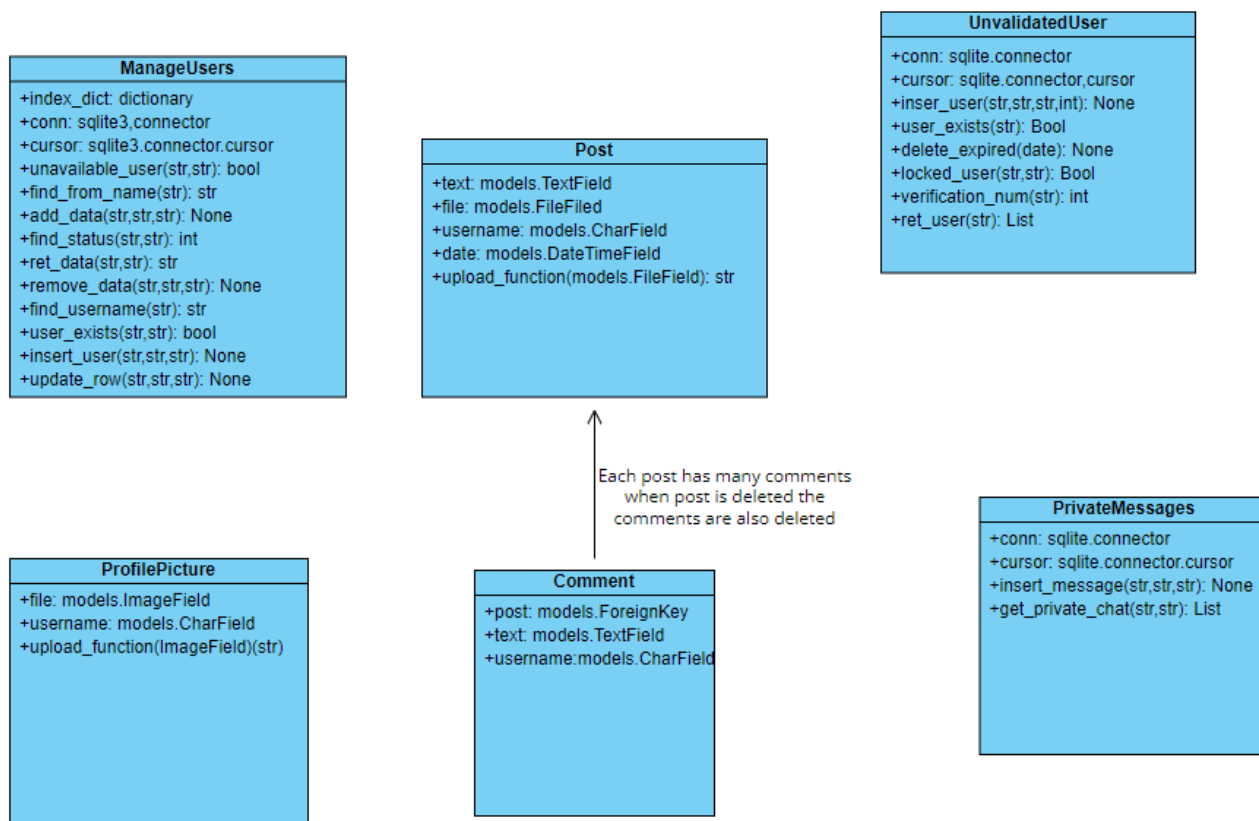
(נותן אפשרות רק כאשר שני המשתמשים עוקבים אחד אחרי השני)



chatn

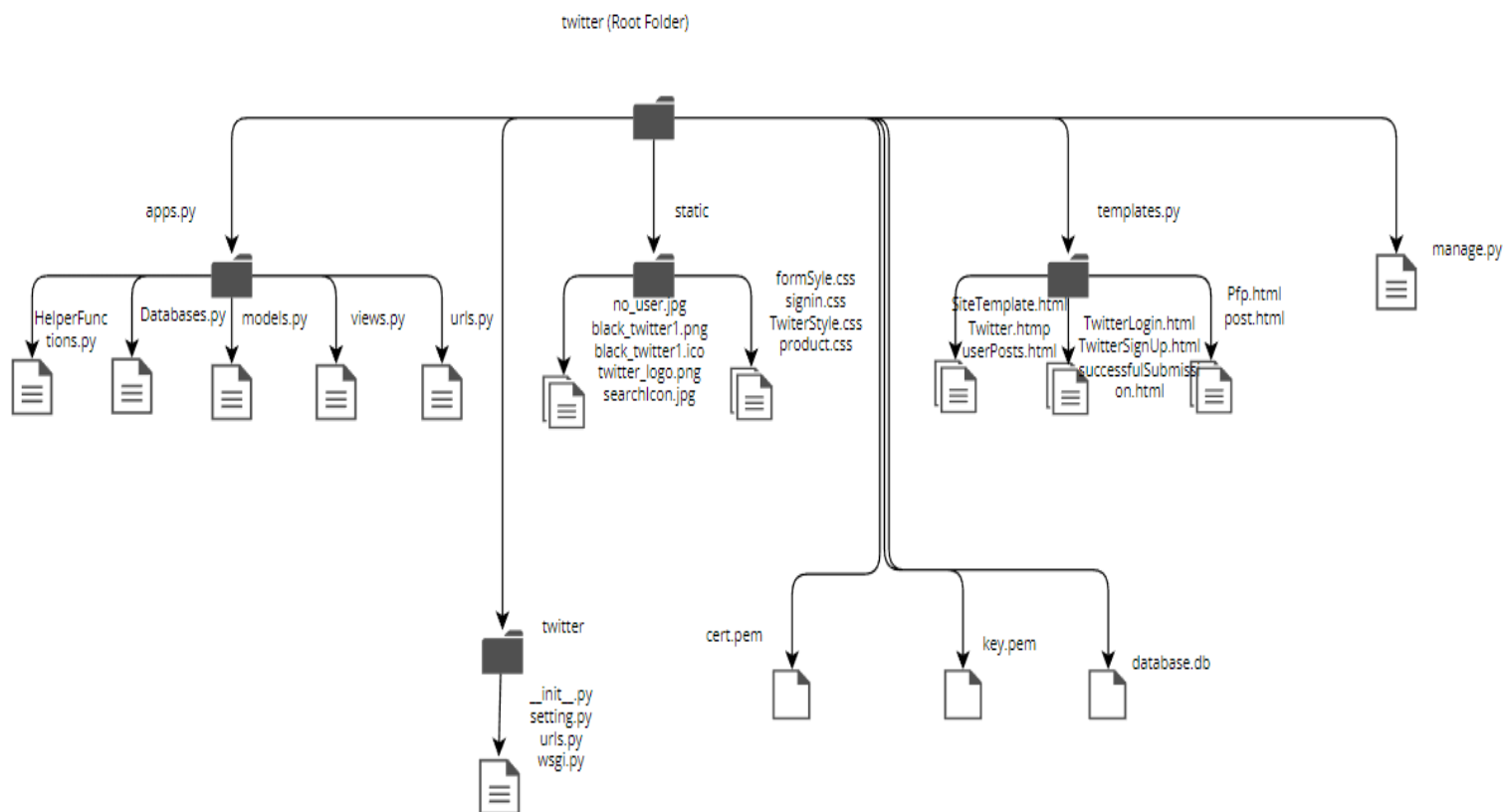
5 מדרוך למפתח

5.1 דיאגרמת UML של כל מחלקות הפרויקט והתלויות ביניהן



ראוי לציין כי התכונה: שם משתמש שמופיעה בפוסטים, בתמונות פרופיל ובדאטבייס של המשתמשים (ManageDatabase) זהה – התמונות פרופיל בעלי שם משתמש מסוים שייכות לפוסטים בעלי שם משתמש זהה שהם הפוסטים של אותו משתמש שהשם משתמש שלו נמצא בדאטאבייס של המשתמשים.

5.2 רשימת פונקציות ומחלקות ותפקידיהם



קובץ	תפקיד	מחלקות
manage.py	הרצה של הקבצים הנצרכים ובמקרה של https הרצה מעל TLS בשימוש ב cert.pem key.pem	
key.pem	המפתח המשמש להצפנה	
cert.pem	המפתח המשמש בתקשורת	
settings.py	משמש את manage.py מגדיר שימוש בספריות וקבצים מסוימים, דוגמאות להגדרות נוספות הן איזור זמן ושפה	
asgi.py	ממשק אסינכרוני לאתר (built-in Django)	
wasgi.py	נותן את הכתובת לאפליקציה (built Django in)	
database.db	המאגר נתונים של sqlite – שומר את פרטי המשתמשים ועוד דברים 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 הגרפיים האחרים שבאתר	
successfulSubmission.html	הדף המודיע על קליטת נתונים שהסתיימה בהצלחה	
Twitter.html	הדף הראשי באתר – לאחר שנכנסים 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 search bar	
black_twitter1.ico	icon	
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	
apps/urls.py – urls.py	קובץ קצר תוכן אך חשוב מאד – הקובץ מפנה את url 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 והכלאים שמשמשים עבור הדאטאבייס של Django formsi מסוימים נמצאים שם	<p>ProfilePicture – המחלקה משמשת לבניית התכונות שנשמרות בדאטאבייס לתמונת פרופיל של המשתמש</p> <p>PictureForm – המחלקה משתמשת בתכונות המחלקה הקודמת ומכינה form שנשלח למשתמש שממלא את התכונות הללו לפי הצורך</p> <p>Post – המחלקה משמשת לבניית התכונות שנשמרות בדאטאבייס לפוסטים</p> <p>PostForm – המחלקה משתמשת במחלקה הקודמת ליצור form בעת הצורך למילוי התכונות שנצרכות</p> <p>comment – מחלקה של תגובות לפוסטים – לפוסט אחד יכולות להיות תגובות רבות והתגובות נמחקות כאשר הפוסט נמחק</p>
Databases.py	הקובץ שאחראי על ניהול המאגר sql של פרטי המשתמשים	<p>ManageUsers – אחראי על ניהול מאגר המידע של פרטי המשתמשים – פעולות עיקריות בו הן להחזיר עמודה במיקום ספיציפי לפי שם המשתמש, הוספת מידע לעמודה ספיציפית ומחיקה.</p> <p>UnvalidatedUsers – אחראי על ניהול המשתמשים הזמניים עד אישור המייל הניהול נעשה באמצעות דאטאבייס הכנסה של משתמשים ומחיקתם לאחר זמן מסוים במחלקה הזאת</p>

PrivateMessages – אחראי על ניהול מאגר המידע של ההודעות הפרטיות, מוסיף מידע ומחזיר מידע מהמאגר נתונים		
	הקובץ בעל פעולות עזר לקבצים שונים המטרה העיקרית שלו היא להפוך את הפרויקט ליותר פשוט להבנה ולקריאה ולכך שפונקציות בviews יהיו רק פונקציות שישמשו עבור כתובות ספיציפיות (פונקציות עזר לא ימצאו בviews)	HelperFunctions.py
	עוזר לDjango לכלול את הספריות המורדות ביותר קלות – (built in Django)	apps.py

6 סיכום אישי / רפלקציה

בשבילי לעבוד על הפרויקט היה מעניין ביותר. יצא לי להתנסות עם ספריות שונות ולעבוד על הפרויקט הכי רציני שעבדתי עליו עד עכשיו ואני מרגיש שעכשיו יש לי קצת ניסיון. הפרויקט גם היה מאד מענה בשבילי והרבה פעמים היה מאתגר.

למדתי דברים רבים על מאגרי מידע, אך למדתי משמעותית ללמוד על ספריות גדולות שלא ידעתי עליהן מקודם ולעבוד איתן כמו Django. למדתי הרבה על יצירת אתרים – ודברים רבים שכרוכים בכך.

כאשר בעבר למדתי על התקשורת בפרוטוקול <https> זאת הפעם הראשונה שיצא לי באמת לממש את הקשר. בנוסף לכך שלמדתי על בניית אתר בשימוש בhtml וגם לעבוד על הפונקציות שלו בJavaScript ועל העיצוב בCSS.

למדתי גם על bootstrap שעזר לי מאד וגם להשתמש בajax לשליחת בקשות מהצד של הקליינט. למדתי גם לעבוד עם forms וגם לשלוח את המידע בלי forms, ואיך לעדכן את האתר כל זמן מסוים על מנת שיהיה דינאמי.

במהלך הפרויקט נתקלתי בקשיים עם הדאטאבייס של Django בעיקר מכיוון שלא כל כך הבנתי איך להשתמש בו בהתחלה. ניסיתי לשמור באמצעות forms של Django והmodels תמונות והדבר לא כל כך עבד לי בהתחלה.

לאחר חיפושים באינטרנט מצאתי את עיקר הבעיה וקראתי שוב את הדוקומנטציות ההכרחיות ולבסוף הבנתי איך להשתמש נכון בכלים הנתונים. בנוסף, בהתחלה גם היו לי קצת בעיות עם שליחת הבקשות מהצד של האתר ללא forms – הצלחתי לסדר זאת אחרי שקראתי יותר מStack overflow ובכללי על שליחת בקשות עם JavaScript.

המסקנה שלי מקשיים אלו היא שעדיף קודם לקרוא היטב על הספריות שאתה משתמש בהן ורק כאשר סיימת לקרוא באמת היטב את ממש רוב המידע הנחוץ עדיף להתחיל לעבוד ולא לקרוא רק חלק מהמידע פעם אחת ואז להתחיל לעבוד.

אילו הייתי מתחיל את הפרויקט היום, הייתי קודם קורא היטב על בניית אתר באמצעות bootstrap html וCSS. ואז הייתי לומד איך לשלב בJavaScript לדברים שאני צריך ואז הייתי מתחיל לעבוד על שלד הפרויקט בלי הפונקציות.

לאחר מכן הייתי קורא בקפידות רבה על Django בעיקר מהדוקומנטציות ורק אז מתחיל לעבוד על הצד של השרת. ואז לאחר מכן הייתי קורא על דברים נוספים כמו עם ajax וממשיך לבנות את מה שנשאר לי.

לבסוף, אני חושב שאני הכן מרוצה מאד מהפרויקט שהכנתי ואני כן גאה בו והצלחתי לענות על כל הדרישות שהיו לי מהפרויקט.

7 מקורות מידע / ביבליוגרפיה

[/https://stackoverflow.com](https://stackoverflow.com)
[/https://docs.djangoproject.com/en/4.1/topics/forms](https://docs.djangoproject.com/en/4.1/topics/forms)
[/https://docs.djangoproject.com/en/4.1/topics/db/models](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://api.jquery.com/jQuery.ajax)
[/https://docs.djangoproject.com/en/4.1/ref/templates/language](https://docs.djangoproject.com/en/4.1/ref/templates/language)
[/https://timonweb.com/django/https-django-development-server-ssl-certificate](https://timonweb.com/django/https-django-development-server-ssl-certificate)
[/https://www.geeksforgeeks.org/getting-started-with-django](https://www.geeksforgeeks.org/getting-started-with-django)
[/https://getbootstrap.com/docs/5.1/getting-started/introduction](https://getbootstrap.com/docs/5.1/getting-started/introduction)
[/https://docs.djangoproject.com/en/4.1/topics/http/shortcuts](https://docs.djangoproject.com/en/4.1/topics/http/shortcuts)
[/https://docs.djangoproject.com/en/3.2/topics/forms/modelforms](https://docs.djangoproject.com/en/3.2/topics/forms/modelforms)
<https://docs.djangoproject.com/en/3.2/ref/forms/fields/#django.forms.SlugField>
[/https://docs.djangoproject.com/en/4.1/ref/models/options](https://docs.djangoproject.com/en/4.1/ref/models/options)

[/https://docs.djangoproject.com/en/4.1/topics/security](https://docs.djangoproject.com/en/4.1/topics/security)

8 קטעי קוד מהפרויקט

קטעי קוד + הסברים:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>
<script type="text/javascript">
const myInterval = setInterval(reload_all_comments, 5000);
var helper = Number("{{comments_num}}")
function reload_all_comments(){
  var http = new XMLHttpRequest();
  http.open("GET", "/site/" + "comment_update=" + helper, true);
  http.onreadystatechange = function() {
    if (http.readyState === 4) {

      if(Number(http.responseText) > helper)
      {
        helper = Number(http.responseText);

        {% for x in id_arr %}
          $(''#' + "{{x}}").load('#' + "{{x}}");
        {% endfor %}

      }
    }
  };
  http.setRequestHeader("X-CSRFToken", "{{csrf_token}}");
  http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  http.send();
}
</script>
```

הקוד הנתון רשום בצד של השרת בjavascript בשימוש בXMLHttpRequest() כלומר השימוש בajax לשליחת בקשה.

הקוד הנתון מבקש עדכון של התגובות במקרה שנוספו. הקוד בעצם מבקש מהשרת לרשום את מספר התגובות של המשתמש וזה מה שהשרת מחזיר. זאת כאשר בפעם הראשונה שהשרת שולח את הדף הראשי של המשתמש הוא שולח את מספר התגובות בפרמטר comments_num אז לאחר הבקשה הזאת למספר התגובות הנוכחי.

הקליינט משווה בין מספר התגובות הנוכחי למספר התגובות שהוא שומר, אם מספר התגובות הנוכחי גדול יותר יש צורך בעידכון אז הוא מעדכן את helper ומעדכן את כל הסגמנטים של התגובות – idn שלהם נשמר בר id_arr שנשלח בהתחלה.

```
class Post(models.Model): # post class for the database
    def upload_function(instance, filename): # gets the filename from the FileField object, returns the folder
        # in which the file would be uploaded to
        return os.path.join(filename.split(SEPARATOR)[FIRST_INDEX], filename)
    text = models.TextField()
    file = models.FileField(upload_to=upload_function, validators=[FileExtensionValidator(
        allowed_extensions=['png', 'jpg', 'mp4']), validate_file_size])
    username = models.CharField(max_length=MAX_STR_LENGTH, blank=True, null=True)
    date = models.DateTimeField(auto_now=True)

class PostForm(forms.ModelForm): # post form
    class Meta:
        model = Post
        fields = ['text', 'file']
```

כאן ניתן לראות את הבנייה של הדאטאבייס של המודלים ואת forms.

המחלקה של הPOST היא הבנייה של המודל של פוסט, תכונה של טקסט, קובץ, שם משתמש ותאריך העלאה. ניתן לראות שלקובץ יש הגבלת גודל והגבלת סוג קובץ. הupload_function אחראית להעלאת קובץ לתיקייה שמתאימה לו. כאשר המשתמש שולח קובץ יש שינוי של שם הקובץ בקוד לשם המשתמש סימן "-" media ואחרי זה מספר כדי להקל על הupload_function.

הupload_function לוקחת מכך את השם של המשתמש ואומרת שלשם הוא יעלה – התיקייה אליה קבצי המשתמשים עולה היא שם המשתמש (ייחודי לכל משתמש).

ניתן לראות בPostForm model = post השורה הזאת אומרת שתכונות הפוסט כתכונות המודל וכך ישמר ולאחר מכן הfields שהדבר אומר מה להציג בפוסט – איזה תכונות יהיה ניתן להציג.

דברים חשובים: auto-now=true כלומר כאשר התאריך לא מוכנס הוא ישמר כתאריך נוכחי, null=true – אומר שיכול להיות לשדה ערך של סטרינג ריק או בכלל ריק. לblank=true אומר שהform יכול להישמר כאשר ערך ריק. null-קשור רק לדאטאטייפ, blank – form.

```
def remove_data(self, table, data, username): # removes data from one of the tables - gets table name data and
# username returns None
try:
    row = self.cursor.execute("""SELECT {} FROM database where username = ?""".format(table),
                              (username,)).fetchall()[FIRST_INDEX]
    new_data = row[FIRST_INDEX].split(SEPARATOR) # change to select specific row
    if data in new_data:
        new_data.remove(data)
    new_data = SEPARATOR.join(new_data)
    self.cursor.execute('UPDATE database SET {} = ? WHERE username = ?'.format(table), (new_data, username))
    self.conn.commit()
except ValueError:
    pass
```

הפונקציה מבקשת את הטבלה, את המידע ואת שם המשתמש ומורידה אותו אם קיים. הכוונה כאן במידע – הוא לשם של משתמש אחר ובטבלה לטבלה של עוקבים, עקיבה או בקשות שנתקבלו. השורה הראשונה של row שומרת מהדאטאבייס בפרמטר את מה שנמצא בחלק של המשתמש – ובrow המסוים. השורה השנייה משנה את המבנה לרשימה. לאחר מכן מוחקים את הדאטא שהתקבלה בפרמטר – ואז מחזירים לסטרינג עם הדאטא החדש ומחזירים אותו לדאטאבייס.

```
if request.method == "POST":
    if "comment" in list(request.POST.keys()):
        try:
            comment_post = Post.objects.get(id=int(request.POST["post"]))
            comment = Comment(post=comment_post, text=request.COOKIE.get('username') + ":" +
                              request.POST["comment"],
                              username=request.COOKIE.get('username'))
            comment.save()
            return enter_site(SITE, request)
        except ValueError:
            pass
```

זאת הפונקציה ששומרת את התגובות – ניתן לראות שקודם נמצא הפוסט אליו משויכת התגובה באמצעות id שנשלח, לאחר מכן נשמרת התגובה לפי כל התכונות שלה ואז מוחזר האתר.

קטעי קוד נוספים:

```
def pfp(request): # let the user choose a profile picture - gets request returns the page + form
    if request.COOKIE.get('username') is None \
        or len(ManageUsers().find_from_name(request.COOKIE.get('username'))) == 0:
        return redirect("/signin/")

    if 'view_user' in list(request.GET.keys()):
        return redirect("/site/?view_user=" + request.GET['view_user'])

    if request.method == 'POST':
        form = PictureForm(request.POST, request.FILES)
        file_type = request.FILES['file'].name.split(".")[-1]
        request.FILES['file'].name = request.COOKIE.get('username') + "-" + \
            "ProfilePicture" + "." + file_type
        if os.path.exists(request.COOKIE.get('username')) and Path(request.COOKIE.get('username') + "/" +
            request.FILES['file'].name + ".jpg").is_file():
            os.remove(Path(request.COOKIE.get('username') + "/" + request.FILES['file'].name + ".jpg"))
        for profile_pic in ProfilePicture.objects.all():
            if request.COOKIE.get('username') + "/" + request.COOKIE.get('username') + "-" + "ProfilePicture." \
                + file_type == profile_pic.file.name:
                profile_pic.delete()

        if os.path.exists(request.COOKIE.get('username')) and Path(request.COOKIE.get('username') + "/" +
            request.FILES['file'].name + ".png").is_file():
            os.remove(Path(request.COOKIE.get('username') + "/" + request.FILES['file'].name + ".png"))
        for profile_pic in ProfilePicture.objects.all():
            if request.COOKIE.get('username') + "/" + request.COOKIE.get('username') + "-" + "ProfilePicture." \
                + file_type == profile_pic.file.name:
```

```
def private_chat(request): # gets a request with a specific user to search - returns the chat page with all the
    # messages sent with that user if both are following each other
    if 'view_user' in list(request.GET.keys()):
        return redirect("/site/?view_user=" + request.GET['view_user'])
    if request.COOKIE.get('username') is None or \
        len(ManageUsers().find_from_name(request.COOKIE.get('username'))) == ZERO_MATCHES:
        return redirect("/signin/")
    if 'user' not in list(request.GET.keys()):
        return enter_site("PrivateChat.html", request)

    helper = ManageUsers()
    if request.GET['user'] in helper.ret_data("following", request.COOKIE.get('username')) and \
        request.COOKIE.get('username') in helper.ret_data("following", request.GET['user']):

        if request.method == POST_REQUEST:
            message = request.POST["message"].replace("\n", "")
            PrivateMessages().insert_message(request.COOKIE.get('username'), request.GET['user'], message)
            data = PrivateMessages().get_private_chat(request.COOKIE.get('username'), request.GET['user'])
```

```
def site_posts(request, user, ret_site, adder=None): # send the posts of the user that the function got - gets request
# , user, site and can get more data to be returned and returns the site + the posts of the user + more data if got
posts = Post.objects.filter(username=user)
img_lst = []
comments = []
id_lst = []
for x in posts:
    img_lst.append(to_base_64(x.file.url[SECOND_INDEX:]))
    id_lst.append("post" + str(x.id))
posts = reversed(posts)
img_lst.reverse()
id_lst.reverse()

context = {"posts_lst": zip(posts, img_lst, id_lst)}

user_pfp = ProfilePicture.objects.filter(username=user)
if len(user_pfp) == ZERO_LENGTH:
    context['Pfp'] = to_base_64(BLANK_PFP_FILE)
else:
    user_pfp = user_pfp[FIRST_INDEX]
    context['Pfp'] = to_base_64(user_pfp.file.url[SECOND_INDEX:])

for x in Comment.objects.filter(post__username=user):
    comments.append(x)
```

```
setInterval(function() {
    var http = new XMLHttpRequest();
    http.open("GET", "/site/?view_user=" + "{{user_search}}" + "&changes={{update_date}}" + "&status={{user_status}}", true);
    http.onreadystatechange = function() {
        if (http.readyState === 4) {
            if(http.responseText == "need update")
            {
                window.location.href = window.location.href;
            }
        }
    };
    http.setRequestHeader("X-CSRFToken", "{{csrf_token}}");
    http.setRequestHeader("Content-type","application/x-www-form-urlencoded");
    var params = 'changes=' + "{{update_date}}";
    http.send(params);
}, 3000);
```