עבודה בתכנון ותכנות מערכות

התמחות הגנת סייבר

5 יחידות לימוד – מספר שאלון 883589

**Social Painter Dash**

לוח ציור שיתופי

מגיש: איתמר קנה

מורה\מנחה: מוטי מתתיהו

בית ספר: אורט אבין רמת גן

שנה: תש"ף 2019

GitHub: <https://github.com/ItamarK5/rplace.git>

תוכן עניינים

[תוכן עניינים 1](#_Toc35979083)

[מסמך ייזום 3](#_Toc35979084)

[מבוא 4](#_Toc35979085)

[תיאור המוצר 5](#_Toc35979086)

[המטרות המרכזיות של העבודה 5](#_Toc35979087)

[התרחישים שהמערכת עונה עליהם 5](#_Toc35979088)

[אילוצים ודרישות 5](#_Toc35979089)

[אתגרי הפרויקט 5](#_Toc35979090)

[תחום המערכת 7](#_Toc35979091)

[סביבה וכלי פיתוח 7](#_Toc35979092)

[בדיקות 7](#_Toc35979093)

[השלבים הנותרים בפיתוח המוצר 7](#_Toc35979094)

[מסמך אפיון 9](#_Toc35979095)

[פונקציונליות המערכת 10](#_Toc35979096)

[פעולות המשתמש 10](#_Toc35979097)

[אילוצים עיקריים 10](#_Toc35979098)

[דרישות בסיסי הנתונים 11](#_Toc35979099)

[סביבת עבודה 11](#_Toc35979100)

[דרישות המערכת 11](#_Toc35979101)

[מסמך עיצוב 14](#_Toc35979102)

[בסיסי נתונים 15](#_Toc35979103)

[דמות – user: 15](#_Toc35979104)

[פתק – note: 16](#_Toc35979105)

[רישום – record: 16](#_Toc35979106)

[Redis: 17](#_Toc35979107)

[מבנה המערכת: 18](#_Toc35979108)

[Config: 18](#_Toc35979109)

[User.py: 19](#_Toc35979110)

[models\pixel.py: 19](#_Toc35979111)

[תיאור מודלים 21](#_Toc35979112)

[מבנה הפקודות בשרת: 23](#_Toc35979113)

[בקשות HTTP רגילות: 23](#_Toc35979114)

[מסכים: 26](#_Toc35979115)

[מסך הרשמה: 27](#_Toc35979116)

[בית: 27](#_Toc35979117)

[צייר: 27](#_Toc35979118)

[יומן פרויקט: 28](#_Toc35979119)

[ביבליוגרפיה 29](#_Toc35979120)

מסמך ייזום

רקע

מבוא

למה בחרתי בפרויקט זה? קראתי בעבר על r/place, צייר שיתופי ענקי שפעל למשך 3 ימים באחד לאפריל במטרה לחקור את הדרך שבא אנשים מגיבים אחד לשני, קנבס גדול שהיית יכול לצבוע אריח אחד כל חמש דקות, אני חושב לנסות לבנות משהו דומה, בנוסף לזה הפרויקט מתקשר לעולם הציור שאני מאוד מתחבר אליו.

מה מעניין ומאתגר אותי בפרויקט? מאתגר אותי לימוד רשתות ומימוש הידע שלמדתי בכיתה ולבנות מערכת באינטרנט שתאפשר התחברות לכל משתמש מרחוק באמצעות דפדפן.

הפרויקט הוא תוצר של למידה במגמת הנדסת תוכנה וסייבר. הפרויקט מאפשר לי ליישם את תחומי הלימוד: תכנות בשפת Python רשתות, אבטחת מידע, וכן לממש כלים שאותם רכשתי, כמו חקירת נושאים חדשים ולמידה עצמית של: redis, Sqlite, מודלים שונים בפיתון המתקשרים לרשתות. ועוד, המטרה שלי בתהליך זה היא להרחיב את אופקי בנושאי התכנות.

המערכת אמורה בעקרון לתת לאנשים ברחבי העולם קנבס משותף שבו אנשים משאירים את חותמם.

תקציר ורציונל הפרויקט:

המערכת מורכבת משרת יחיד המאפשר . דרך מנועי חיפוש אינטרנטיים שונים (כרום) אדם יוכל להיכנס לאתר האינטרנט ובתגובה ישלח אליו דפי אינטרנט בהתאם לכותבת המבוקשת (ראו תרשים אחד), צד השרת ידאג לעדכן את הלוח לכל עדכון של השחקן, במקרה של התנתקות מהשרת, הלקוח ישלח הודעת תגובה מתאימה באמצעות

הגדרת המערכת ויעדיה

מטרת המערכת היא לאפשר לקהילות שונות לצבוע לוח שיתופי ענקי דרך המחשב הביתי שלהם. בנוסף אפשרות לדווח על משתמשים שאינם פועלים כראוי המנסי

אתגרי הפרויקט

* הפצת מידע למספר רב של משתמשים באופן מסונכרן.
* למידת מודולים חדשים Python, המאפשרים תקשורת של רשתות.
* למידת תכנות באמצעות צד לקוח של דפדפן המתבטאת בשלוש סוגי המסמכים הבאים:
  + שימוש בדפי HTML – שפת תגיות לתצוגה ועיצוב של דפי אינטרנט ותוכן לתצוגה של דפדפן
* עבודה עם בסיסי נתונים מסוגים שונים SQL ו-NoSQL.
* עבודה באמצעות תהליכים חיצוניים – celery worker.
* שימוש בטור הודעות בשביל תהליכים חיצוניים – RabbitMQ.

הגדרות

**עמית לעמית (Peer to Peer)** – רשת תקשורת בה כל אחד מהקצוות מתפקד הן כלקוח והן כשרת, וכל אחד מהקצוות מסוגל ליזום או לסיים התקשורת וכד'...

* **לקוח** – מחשב חיצוני המתקשר עם השרת דרך דפדפן.
* **שרת** – מחשב המשתמש, עליו ישאת ממשק המשתמש וממנו מופעלות הפעולות השונות של המערכת.
* **עובד חיצוני** – עובד חיצוני המשמש לעבודות שאמורות לקחת זמן רב בשביל השרת לבצעם (ספציפית שליחת מייל).

הגדרות

משתמשי המערכת

המערכת מיודעת כצייר שיתופי כללי כך שכל שחקן יכול לצבוע כל דקה שונה, הפרויקט בתיאוריה מיועד לקהל העולם הרחב אך בעקבות מגבלות של מחסור רב באנשים, הפרויקט ייבדק לכמות נמוכה יותר של משתמשים.

תיאור המוצר

המטרות המרכזיות של העבודה

* לוח ציור (קנבס) בגודל 1000x1000 פיקסלים כדי שירגיש מאוד גדול.
* כל הלקוחות צריכים להישמר בסנכרון במבט על מצב הנוכחי של הלוח אחרת לכל שחקן יהיה צורה שונה של הלוח ויהיה קשה לקשר ביניהם.
* לתמוך בלפחות 10 אנשים שונים (למינימום של הפרויקט).
* אנשים יכולים לצבוע אזור אחד כל דקה (לעומת החמש דקות של r/place).

התרחישים שהמערכת עונה עליהם

* לאפשר ללקוח להתחבר עם שם משתמש וסיסמה, להירשם אל האתר ולאתחל את הסיסמה שלו.
* לאפשר לשחקן לצבוע על לוח הציור פיקסל אחד כל דקה.
* לקשר את כל המחשבים יחד.
* לאפשר לאנשי ניהול ומנהל ראשי לעדכן מידע על השחקנים, לחסום משתמשים ולבטל חסימת משתמשים.

אילוצים ודרישות

* המערכת תאפשר חיבור מספר רב של משתמשים (כלפחות 10) שיוכלו לצייר על הלוח במקביל.
* המערכת תשלח באופן יעיל את השינויים בלוח הציור.
  + בפעם הראשונה תשלח את כל המידע שיש על לוח ציור.
  + לאחר מכן, ישלחו רק עדכונים של השינויים בלוח ציור.
  + המערכת תתמוך בפעולת refresh ותעדכן את כל לוח הציור בדפדפן הלקוח.
  + אם השחקן יתנתק המערכת היא תאפשר לשחקן להתחבר מחדש.
* הדפדפן חייב לתמוך בעוגיות.

אתגרי הפרויקט

* הפצת מידע למספר רב של משתמשים באופן מסונכרן.
* למידת מודולים חדשים Python, המאפשרים תקשורת של רשתות.
* למידת תכנות באמצעות צד לקוח של דפדפן המתבטאת בשלוש סוגי המסמכים הבאים:
  + שימוש בדפי HTML – שפת תגיות לתצוגה ועיצוב של דפי אינטרנט ותוכן לתצוגה של דפדפן
* עבודה עם בסיסי נתונים מסוגים שונים SQL ו-NoSQL.
* עבודה באמצעות תהליכים חיצוניים – celery worker.
* שימוש בטור הודעות בשביל תהליכים חיצוניים – RabbitMQ.

תחום המערכת

* המערכת תרוץ על פיתון ותתקשר עם בסיס נתונים מסוג Redis ובסיס
* נתונים מסוג Sqlite.
* המודלים שישמשו אותי יהיו flask, תוספות שונות של flask. לעדכון המשתמש אשתמש בsocketio המאפשר נוחות רבה עם תקשורת וגם מאפשר broadcast לכל השחקנים בקלות. באופן תאורטי המערכת מטפלת בכמות לא מוגבלת של לקוחות בו זמית אך דבר זה יבוא במחיר מהירות המחשב השרת וכמות העומס אותו הוא מסוגל להכיל.

סביבה וכלי פיתוח

* Python 3.7.5 בצד השרת.
* Html, css, js בצד הלקוח.
* Redis – לשמירת ערכים כללים (בעיקר לוח המשחק) ללא טבלה ובנוסף לכך כתור הודעות.
* Sqlite – לשמירת מידע על משתמשי המערכת.
* הפיתוח יתבצע בwindows.
* הלקוח ישתמש בדפדפן כרום (המערכת תומכת בדפדפנים נוספים).
* Visual Studio Code – עורך קוד צד הלקוח (html, css, javascript)
* Pycharm – עורך קוד הפייתון.

השלבים הנותרים בפיתוח המוצר

השלבים הנותרים בפיתוח המוצר הינם:

|  |  |
| --- | --- |
|  | תאריך סיום |
| בניית השרת ומסד הנתונים | 12.12.19 |
| כניסה באמצעות שם משתמש וסיסמה. | 20.12.19 |
| האפשרות לבצע צ'אטים \ שיחות קוליות. | 10.2.20 |
| ציור באמצעות | 29.2.20 |
| מקום רב יותר לצייר | 30.3.20 |

מסמך אפיון

פונקציונליות המערכת

פעולות המשתמש

1. הרשמה ראשונית למערכת.
2. אימות באמצעות אימייל.
3. התחברות משתמש קיים למערכת.
4. אפשרות איפוס סיסמה.
5. צביעת פיקסל על הקנבס:
   * אפשרות לחיצה על הקנבס באמצעות העכבר לצביעת פיקסל.
6. ניווט באמצעות זיהוי משאבים בשורת הurl
   * עדכון המסך.
   * התאמת הפרמטרים בfragment כך שיתאים למידע.
   * כאשר נכנס לקנבס ללא פרמטרים, השחקן יראה את המיקום האהוב עליו.
7. ניווט בקנבס:
   * הזזת הקנבס באמצעות חצים.
   * הזזת הקנבס באמצעות העכבר.
8. אפשרות לבחור צבע לצבוע:
   * באמצעות לחיצה של העכבר.
   * באמצעות מקשים במקלדת.
9. צביעה מסונכרנת.

אילוצים עיקריים

* המערכת מחייבת עבודה עם דפדפן chrome וחיבור לאינטרנט (בעתיד אוסיף תמיכה לדפדפנים אחרים).

**הנחות יסוד**

* המערכת תרוץ בסביבת עבודה MS Windows 7 ומעלה.
* הדפדפן שדרכו השחקן יתחבר יתמוך בcookies.

דרישות בסיסי הנתונים

* בבסיס נתונים מסוג SQLITE ישמרו נתוני המשתמש ופתקים שהמנהלים רשמו על משתמשים.
* בסיס הנתונים מסוג Redis ישמור מידע ללא קשר של משתמשים.

סביבת עבודה

* השרת נכתב בPython 3.7
* ה-GUI יהיה ברשת האינטרנט.
* בסיס הנתונים מסוג SQL ייבנה בעזרת ב-SQLite.
* בסיס הנתונים מסוג NoSQL ייבנה בעזרת Redis.
* הודעות של socket.io ישלחו באמצעות message\_queue שמקורו redis.
* לשליחת הודעות מיילים ישמש celery worker.

דרישות המערכת

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | דרישה | קלט | תהליך | פלט | טיפול בשגיאות |
| 1 | תהליך רישום משתמש חדש | נתוני משתמש:  שם משתמש  סיסמה  אימות סיסמה | העברת הנתונים לשרת ושמירתם בבסיס הנתונים | הלקוח נרשם בהצלחה – עליו להיכנס דרך קישור האימות. | אם אחד מהבאים קורה:  סיסמה קצרה  אימייל תפוס או לא תקין.  שם משתמש תפוס.  כינוי תפוס, תוחזר הודעת שגיאה.  אם המשתמש חסום תשלח הודעת שגיאה מתאימה |
| 2 | התחברות של משתמש קיים. | שם משתמש וסיסמה. | העברות הנתונים לשרת ובדיקתם בבסיס הנתונים. | הלקוח התחבר בהצלחה למעכרת – מעבר ללובי. | אם אחד הנתונים לא מתאימים\ שהמשתמש כבר מחובר למערכת\המשתמש חסום תחוזר הודעת שגיאה. |
| 3 | קישור url לאימות | שליחת מייל לאימות השם משתמש והסיסמה. | שליחת הודעת מייל למשתמש עם url לקישור שכניסה דרכו מאפשרת את המשתמש | הלקוח התחבר | N/A |
| 4 | איפוס סיסמה. | שם משתמש ואימייל | בדיקת השם משתמש והאימייל, אם הם תואמים ישלח לאמייל קוד לשחזור סיסמה, לאחר מכן המשתמש יתבקש להכניס קוד זה, אם הקוד נכון תינתן לו אפשרות לשנות את סיסמתו | מתן אפשרות לאיפוס סיסמה. | אם שם משתמש והאמייל לא תואמים (לא שייכים לאף משתמש) או שכתובת האי-מייל לא תקינה\המייל\משתמש חסום תוחזר הודעה מתאימה. |
| 5 | צביעת פיקסל על הקנבס | מיקום עכבר | האדם לוחץ על פיקסל, נשלחת הודעה לשרת עם הפיקסל והשחקן. השרת בודק אם יתכן, | לשרת נשלחת הודעה לצבוע פיקסל | מיקום הפיקסל לא תקין, לא צובע.  השחקן לא יכול לשים צבע – לרשום לו. |
| 6א | הזזת קנבס באמצעות חצים | כפתור | האדם לוחץ על כפתור ומזיז את הקנבס. | הקנבס זז. | הקנבס יוצא מחוץ לטווח. |
| 6ב | הזזת קנבס באמצעות העכבר | מיקום עכבר | האדם לוחץ על כפתור וגורר את הקנבס. | הקנבס זז | הקנבס יוצא מחוץ לטווח. |
| 6ג | הזזת הקנבס באמצעות fragments | Path fragments | המשתמש משנה את כתובת הpath כך שהפרמטרים יתאימו למיקום\גודל שהוא רוצה לראות בקנבס | הצג משתנה כך שהקנבס יראה בהגדלה המתאימה ובמיקום כך שהאמצע יהיה המיקום הנכנס | פרמטר לא תקין - לא ישתנה  פרמטר לא מופיע – יכניס את הרגיל  אם מתרחש תקלה הצג ישנה את הפרמטרים. |
| ג | אפשרות לבחור צבע באמצעות העכבר | לחיצה של עכבר | האדם לוחץ על כפתור המייצג צבע וקובע את ייצוגו | כפתור הצבע מסומן, אם היה צבע לפני הוא מפסיק להיות מסומן. | N/A |
| 8א | אפשרות לבחור צבע באמצעות עכבר. | לחיצה על עכבר. | המשתמש לוחץ על כפתור, "מזיז" את סימון הצבע לצבע מימין או שמאל (מעגלי) | כפתור הצבע מסומן, אם היה צבע לפני הוא מפסיק להיות מסומן. | N/A |
| 8ב | אפשרות לשנות צבע באמצעות המקלדת | לחיצה על מקש | המשתמש לוחץ על כפתור להרמת חלון הצבעים.  כפתור אחר לעבוד לצבע הימיני.  כפתור אחר לעבור לצבע השמאלי. | הצבע שנבחר מקודם מפסיק להיות מסומן והצבע הבא מסומן במקומו\חלון הצבעים מופיע\נסגר. | מנסים לעבור לכפתור הימני מהכפתור הימני ביותר (ומקרה דומה לשמאלי) , פתרון: יעבור לכפתור בקצה השני. |
| 9 | צביעה מסונכרת | הודעה דרך socketio | השרת שולח למשתמש הודעה כי נצבע פיקסל | צביעת הפיקסל המתאים במסכי כל המשתמשים. | N/A |

מסמך עיצוב

בסיסי נתונים

חשוב מאוד לציין, כל העבודה מול בסיס הנתונים מסוג SQLite נעשה דרך המודל sql-alchemy המאפשרת לעבוד עם בסיסי נתונים מסוג SQL בשיטה פייתונית,

משתמש – user

|  |  |  |
| --- | --- | --- |
| שם | SQlite | טיפוס נתונים - פייתון |
| id | **SMALLINT** | **int** |
| name | **varchar(15)** | **Int** |
| password | **varchar(128)** | **bytes** |
| mail | **varchar(254)** | **String** |
| next\_time | **varchar** | **datetime** |
| role | **Boolean** | **Role(Enum)** |
| x | **SMALLINT** | **int** |
| y | **SMALLINT** | **int** |
| scale | **SMALLINT** | **int** |
| color | **SMALLINT** | **int** |
| url | **varchar** | **string** |

* id – האינדקס של הדמות (יותר מהיר לחפש באמצעות int מאשר string)
* name – השם משתמש של השחקן.
* password – הסיסמה המוצפנת של השחקן, בעזרתה המשתמש מתחבר לשרת.
* email – המייל של המשתמש בשביל אימותים עם השרת.
* next\_time – הזמן הבא שבוא השחקן יוכל לצבוע פיקסל, הזמן נשמר ככמות השניות מאז הראשון הראשון 1970 בחצות.
* role–לבדוק את רמת הניהול של המשתמש, מיוצג כEnum.
* creation – התאריך שבו השחקן נוצר.
* y – המיקום האהוב על השחקן בציר ה-x (500 התחלתי).
* x – המיקום האהוב על השחקן בציר ה-y (500 התחלתי).
* scale – רמת הגדלה של המשחק הראשונית כאשר המשחק עולה (4 התחלתי).
* color – האינדקס של הצבע שנפתח כאשר המפה נפתחת בידי השחקן.
* url – קישור לאתר צ'אט לבחירת המשתמש.

פתק – note

פתק המייצג מידע שאחד המנהלים של המשחק רשם על המשתמש.

|  |  |  |
| --- | --- | --- |
| שם | SQlite | טיפוס נתונים - פייתון |
| Id | **Integer** | **int** |
| user\_subject\_id | **Integer** | **int** |
| description | **varchar** | **string** |
| post\_date | **varchar** | **datetime** |
| user\_writer\_id | **Integer** | **int** |
| Is\_record | **Boolean** | **boolean** |

* Id – המזהה של האובייקט
* user\_subject\_id – מזהה השחקן אשר עליו רשום הפתק.
* description – המידע של הפתק. התוכן שלו.
* post\_date – התאריך שבוא נכתב הפתק.
* user\_writer\_id – מזהה השחקן אשר רשם את הפתק.
* is\_record – ערך בוליאני המזהה אם הפתק קשור לרישום על השחקן לעומת, קח המערכת מסוגלת לקשר אם השחקן הינו חלק מהמערכת או לא.

רישום – record

המחלקה המייצגת את טיפוס הנתונים רישום, יורשת מפתק, שהרי פתק הוא מידע המתייחס אל השחקן בעוד רישום הוא רישום המתקשר לפעילות השחק ברשת. לאחר שאובייקט note נטען מתבצעת בדיקה אם שדה is\_record של השורה דלוק, אם כן השרת מחפש שורת record בעל id השווה למספר ה-Id של ההודעה. זה לא הדבר הכי גנרי אך מספק את הצרכים.

|  |  |  |
| --- | --- | --- |
| שם | SQlite | טיפוס נתונים - פייתון |
| Id | Integer | int |
| reason | varchar | string |
| is\_active | Boolean | bool |
| affect\_from | Nullable(varchar) | datetime |

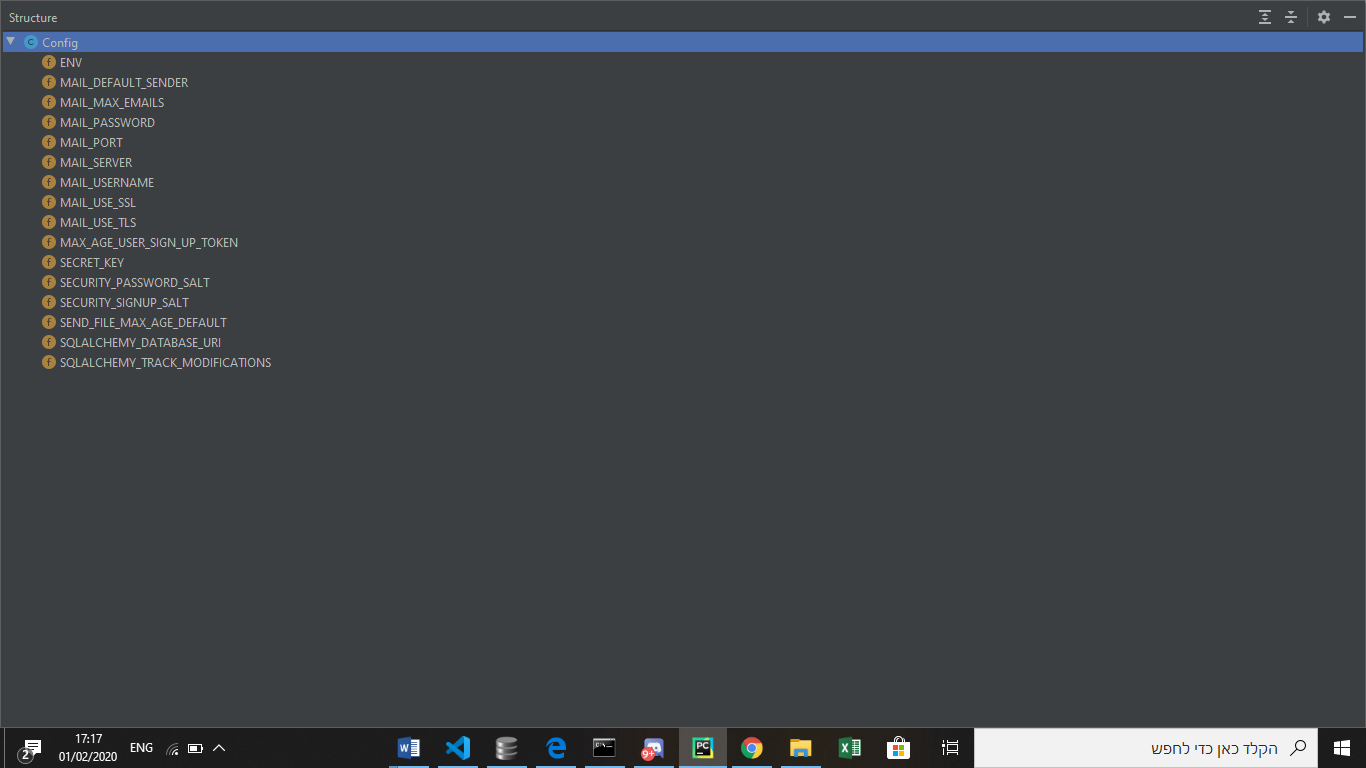
* Id – המזהה של הרישום (והמזהה של האב שלו – הודעה)
* reason – הסיבה למה המשתמש נמחק ומוצגת לפניו כאשר מנסה להתחבר.
* affect\_from – ממתי הרישום לוקח השפעה (ייצוג טקסט), אם null הכוונה שהרישום פועל החל מהזמן שהרישום נרשם.

Redis:

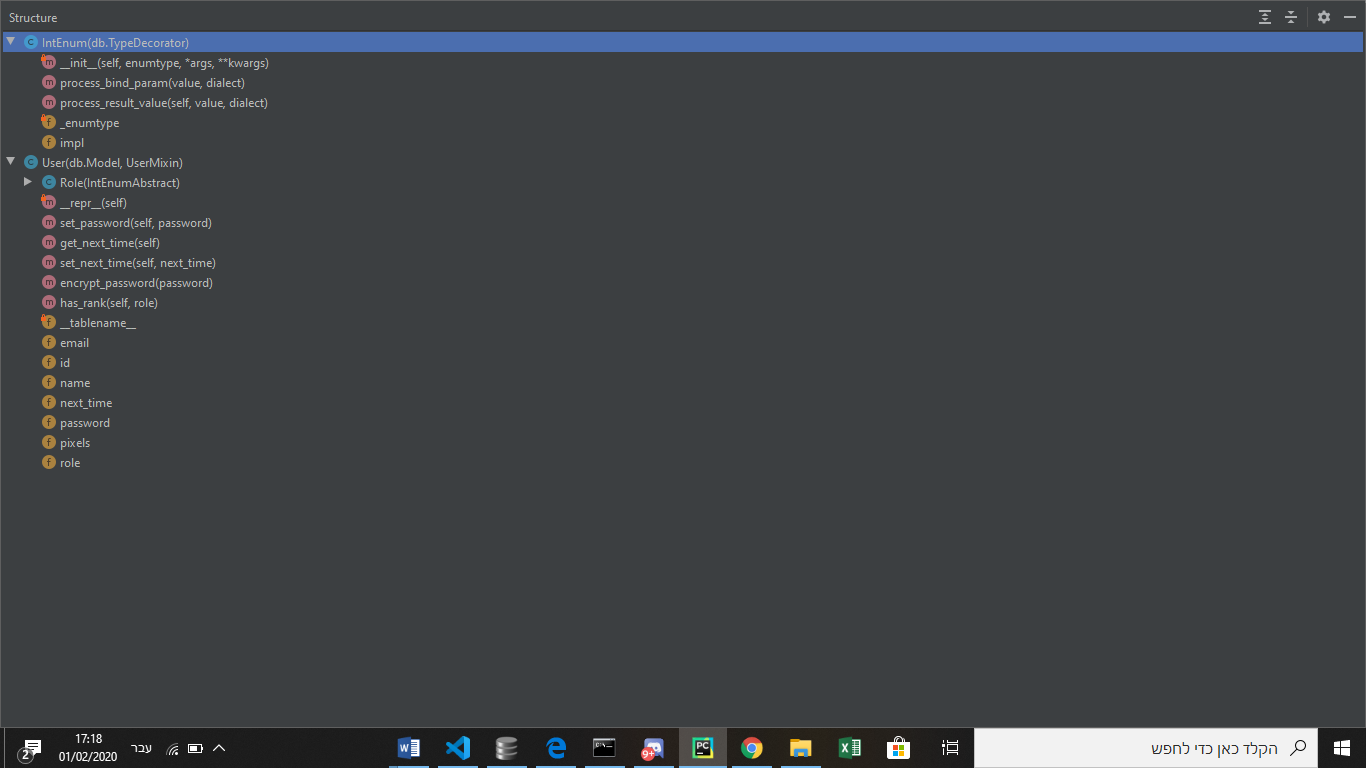
|  |  |  |
| --- | --- | --- |
| מפתח | סוג טיפוס | ערך |
| Board | String | הלוח של המשחק. כדי לחסוך מיקום וזמן הלוח נשמר כמו שהוא נראה, מכיוון שניתן לצבוע את הלוח ב16 צבעים שונים אני שומר בכל תו של הסטרינג שני ערכים שונים. (הסרבר צריך לדאוג שכאשר הוא משתמש בפעולה bitfield על הלוח הוא זוכר שהמערכת שומרת ב-little endian). |
| Lock | Boolean | ערך בוליאני השומר אם ניתן לערוך את הלוח באותה העת.  כאשר דלוק אינו דולק.  כאשר אינו דלוק אינו דולק. |

מבנה המערכת:

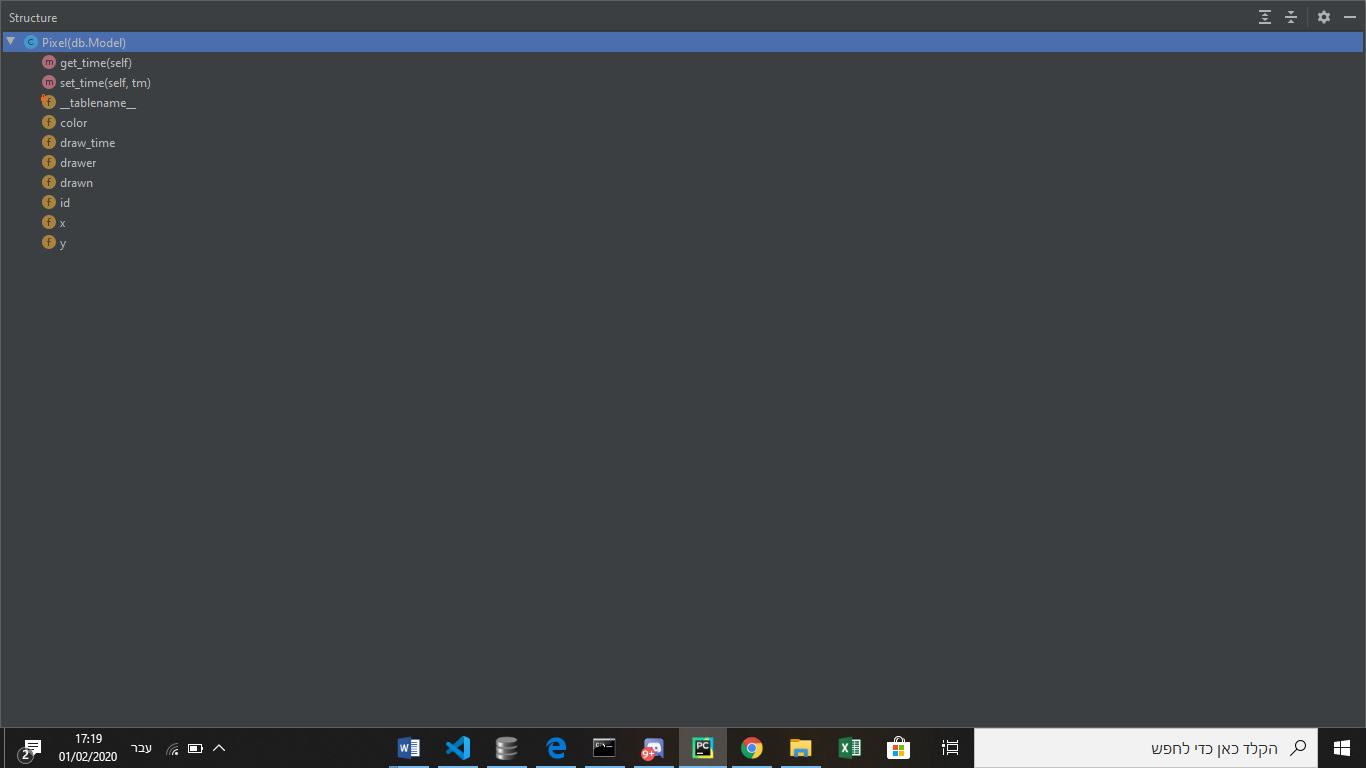
Config:



User.py:



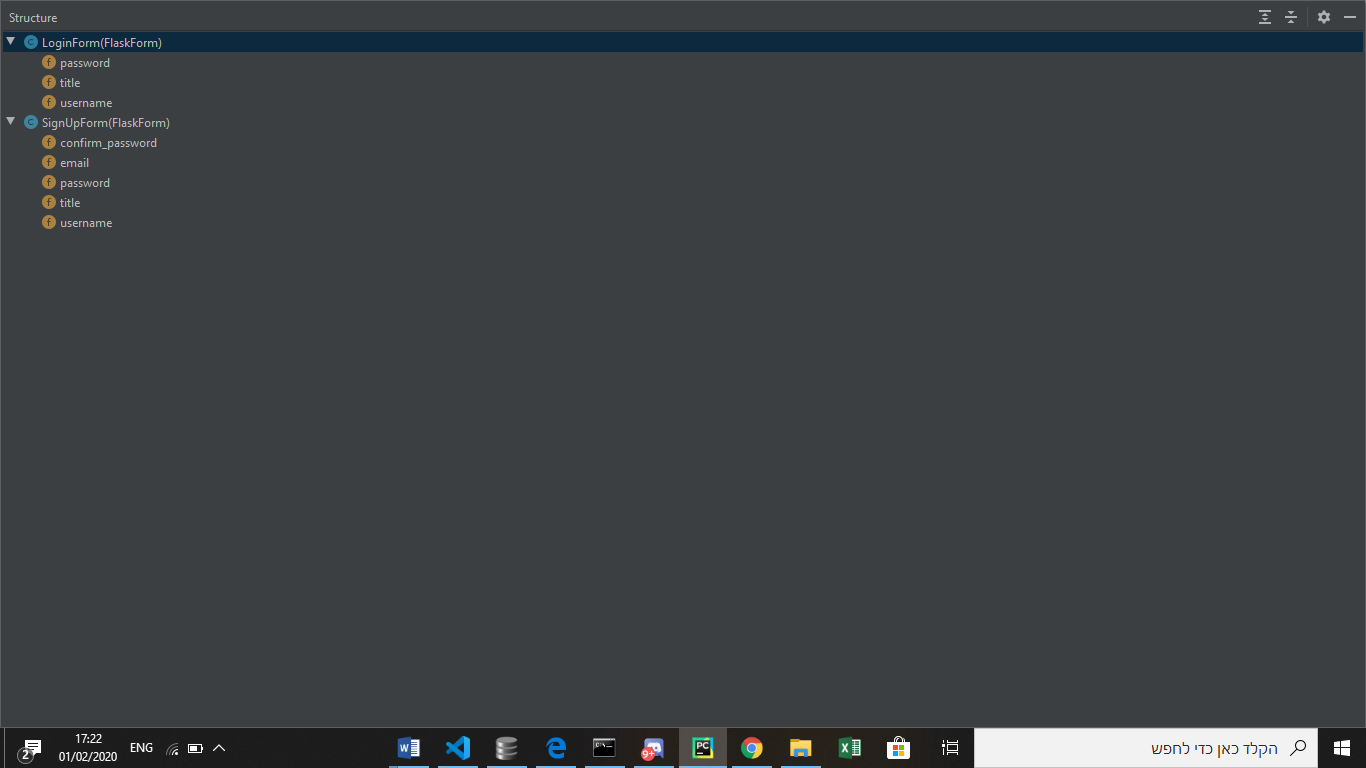
models\pixel.py:



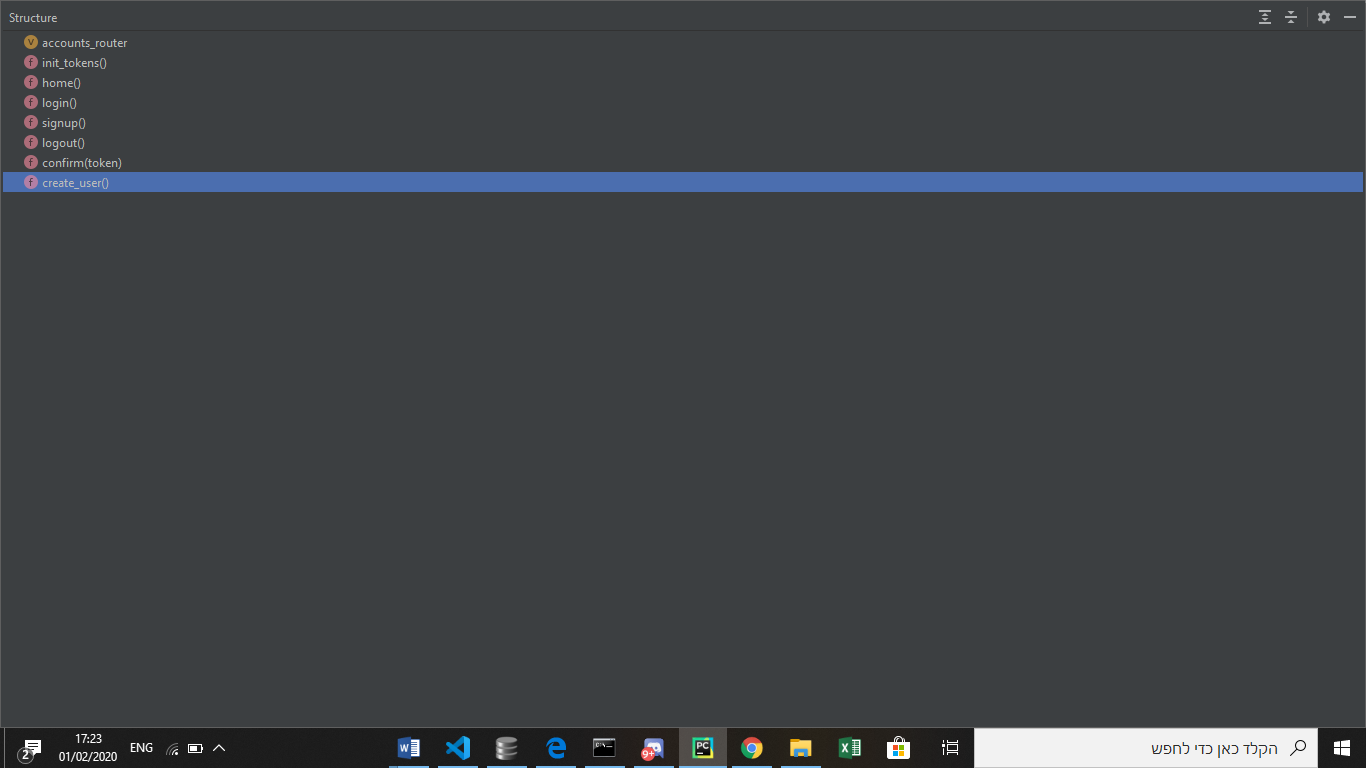
### Apps:

### \Accounts:

#### \forms.py:



### \urls.py:



תיאור מודלים

Apps:

* Accounts – חבילה הקשורה לכל הנושא של תקשורת עם חשבונות.
  + urls.py – הקובץ מחזיק את הבלופרינט המתקשר לכל הקישורים המשתייכים לנושא ההתחברות, ושם כל הפונקציות שמקושרים לurls.
  + Helpers.py – מחזיק את כל הפונקציות והאובייקטים המתקשרים ספציפית אל accounts.py ספציפית.
  + Forms.py – מחזיק את כל אובייקטי הwtf\_forms המקושרים להתחברות, הרשמה וכד...
  + Mail.py – מחזיק פונקציות היוצרות אובייקטי מייל לשלוח באמצעות accounts.py. כגון מייל הרשמה.
  + Place – חבילה הקשורה לקישור האפליקציה.
  + Urls.py – כל הניווט לאפליקציה.
* Other.py – חבילה הקשורה לכל הניווטים שאינם בכיוון ראשי עם האפליקציה כגון הmemes או שליחת קבצים קבועים.
  + urls.py – הקובץ מחזיק את הבלופרינט המתקשר לכל הקישורים המשתייכים לנושא הצייר השיתופי.
  + Helpers.py – מחזיק את כל הפונקציות והאובייקטים המתקשרים לניווט.
* Other – חבילה הקשורה לכל הדברים השונים, כגון אפשרות הmeme ולקישור.
* Models – חבילה המחזיקה את כל האובייקטים הנשמרים בבסיס הנתונים באמצעות sqlalchemy.
  + User.py – המחלקה של משתמש שעבר את האימות נתונים\שנתוניו נשמרו.
  + Note.py – שומרת את מחלקת הנתונים של המשתמש ואת הפונקציה של טעינת משתמש בשביל flask\_login.
* Constants.py – קובץ שבו רשומים קבועים המשמשים את כל התוכנה.
* Extensions.py – קובץ אשר מכיל את התוספים לפלאסק ובונה אותם.
* Functions.py – פונקציות שונות כלליות המשמשות בתוכנה.
* Config.py – קובץ המחזיק מחלקה הנקראת Config, שמגדירה תכונות לאפליקציה
* \_\_init\_\_.py קורא לכל הפעולות המתחילות את השרת, מה שמורץ שם בונה את אפליקצית הפלאסק.
* App.py – קובץ חיצוני המריץ את האפליקציה.

מבנה הפקודות בשרת:

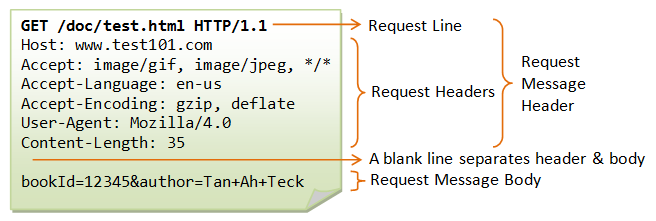
הפקודות נחלקות לשני סוגים של בקשות:

בקשות HTTP רגילות:

פרוטוקול HTTP הוא פרוטוקול תקשורת בשכבת היישום שנועד להעברת דפי HTML ואובייקטים שהם מכילים (כמו תמונות, קובצי קול, סרטוני פלאש וכו'), ברשת האינטרנט וברשתות האינטרנט. התקשורת ב-HTTP מתחילה ביצירת שיחה בין השרת ללקוח באמצעות פרוטוקול TCP בשכבת התעבורה של פרוטוקול TCP/IP, ונמשכת בסדרה של בקשות (requests) ותשובות (responses) שנשלחות על ידי הלקוח והשרת, בהתאמה. ראשית הלקוח יוצר חיבור לכתובת ה-IP של ולפורט שבו השרת נמצא, בדרך כלל פורט 80. לאחר מכן נשלחת הבקשה הכוללת את הכתובת של האובייקט המבוקש (למשל, דף HTML). ופרטים נוספים על הבקשה ועל הלקוח. השרת קורא את הבקשה, מפענח אותה, שולח ללקוח תשובה בהתאם ולרוב מנתקאת החיבור ללקוח כשהשליחה הסתיימה.

בקשת HTTP מורכבות מהנתונים הבאים:

* שיטת הבקשה (POST, GET, PUT, DELETE, OPTIONS, TRACE)
* כתובת של האובייקט המבוקש.
* גרסת הפרוטוקול שלפיו מורכבת הבקשה.
* שדות כותרת המתייחסים לבקשה, ללקוח או לתוכן הנמצא בגוף הבקשה.
* גוף הבקשה.



socket.io

אך בחלק מהפעולות (של האפליקציה) משתמשים במבנה פקודה של socket.io, שולחים מחרוזת המחזיקה את שם אירוע ואובייקט (פייתון מתייחס אליו כמילון בעוד ג'אווסקריפט מתייחס אליו כאל אובייקט) המייצג פרמטרים לאירוע. דוגמא לכך:

|  |  |
| --- | --- |
| שם האירוע | פרמטרים |
| set-board | x, y, color |

בפרויקט יש שני שימושים לבקשות socket.io

פרט לשימוש כתקשורת מהירה לhttp, הsocket.io משמש לבדיקה זריזה אם השרת מחובר באותה העת, בפעם הראשונה שהsocket.io בצד הלקוח מנסה להתחבר מחדש ואינו מצליח להתחבר ללקוח, סימן שאי אפשר לתקשר עם השרת ועקב כך מציג הודעת אי אפשרות להתחבר.

האירועים של השרת ללקוח:

|  |  |  |  |
| --- | --- | --- | --- |
| הודעה | שם האירוע | הסבר | פרמטרים |
| לנסות לקבוע פיקסל על הלוח. | set-board | הלקוח מנסה לצבוע פיקסל על הקנבס | x, y, color |
| התחברות | connect | הלקוח מתחבר באמצעות socket.io אל השרת. השרת מחזיר אל הלקוח את הקנבס ומתי שבו יוכל לעדכן את הלוח באמצעות start-board | כלום |

### תגובה של השרת ללקוח, שאינו אירוע משלו:

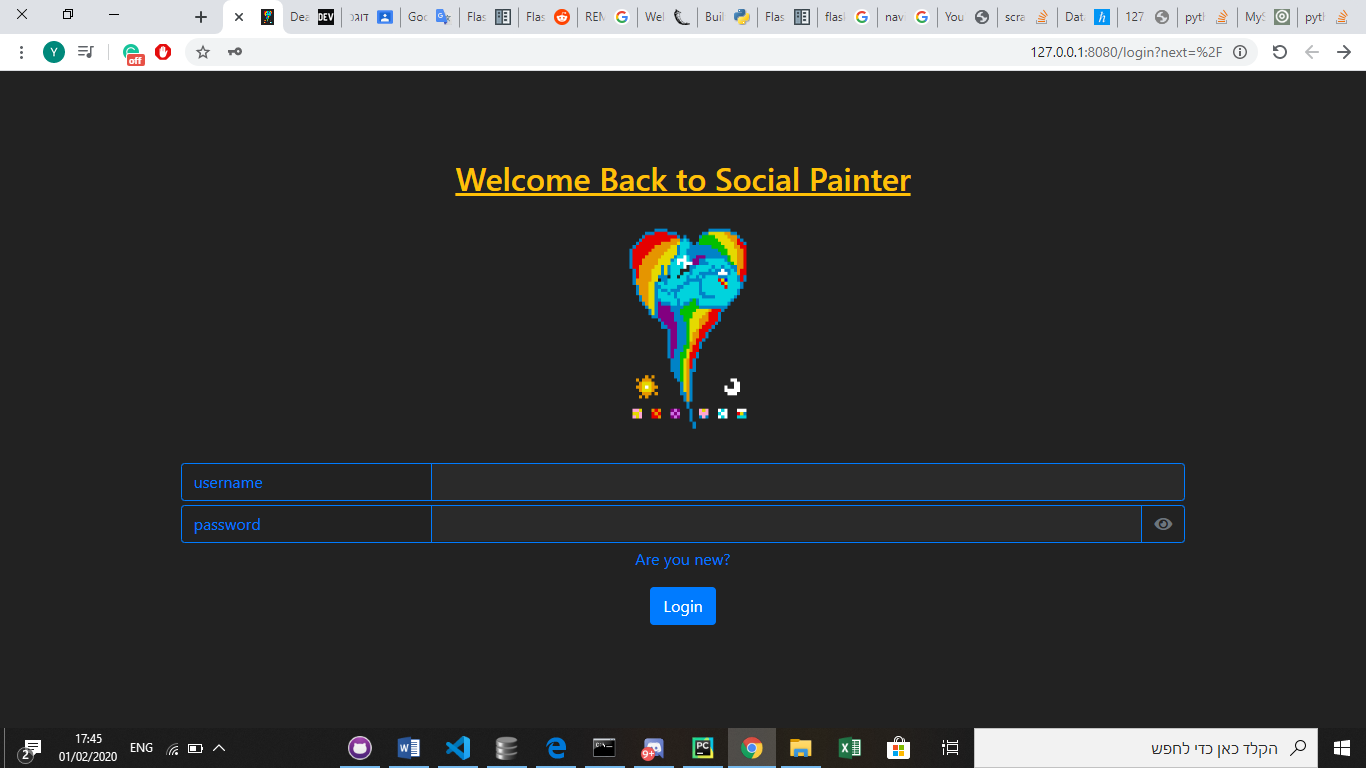
|  |  |  |  |
| --- | --- | --- | --- |
| אירוע | שם האירוע | ביצוע השרת | ערכים |
| לנסות לקבוע פיקסל על הלוח. | set-board | השרת בודק את הנכונות של הבקשה מהמשתמש והאם הוא רשאי לעשות זאת, השרת גם בודק את הזמן שהלקוח אמור להיות מסוגל לצבוע פיקסל, אם הוא השרת מגלה את הזמן כתקין, הוא יחזור זו לשרת אחרת הוא יחזור undefined. | next\_time |

### הודעה של השרת ללקוח:

|  |  |  |  |
| --- | --- | --- | --- |
| הודעה | שם האירוע | הסבר | פרמטרים |
| התחלת מקום | start-place | השרת משיב ללקוח על החיבור ומודיע לו על פרמטרים שנחוצים לתחילת המסך. | Board  Current\_time |
| קביעת פיקסל | set-board | לאחר בדיקת הנכונות של בקשת הלקוח לצבוע פיקסל השרת שולך הודעה לכל המשתמשים לצבוע את הפיקסל. | X, Y, color |

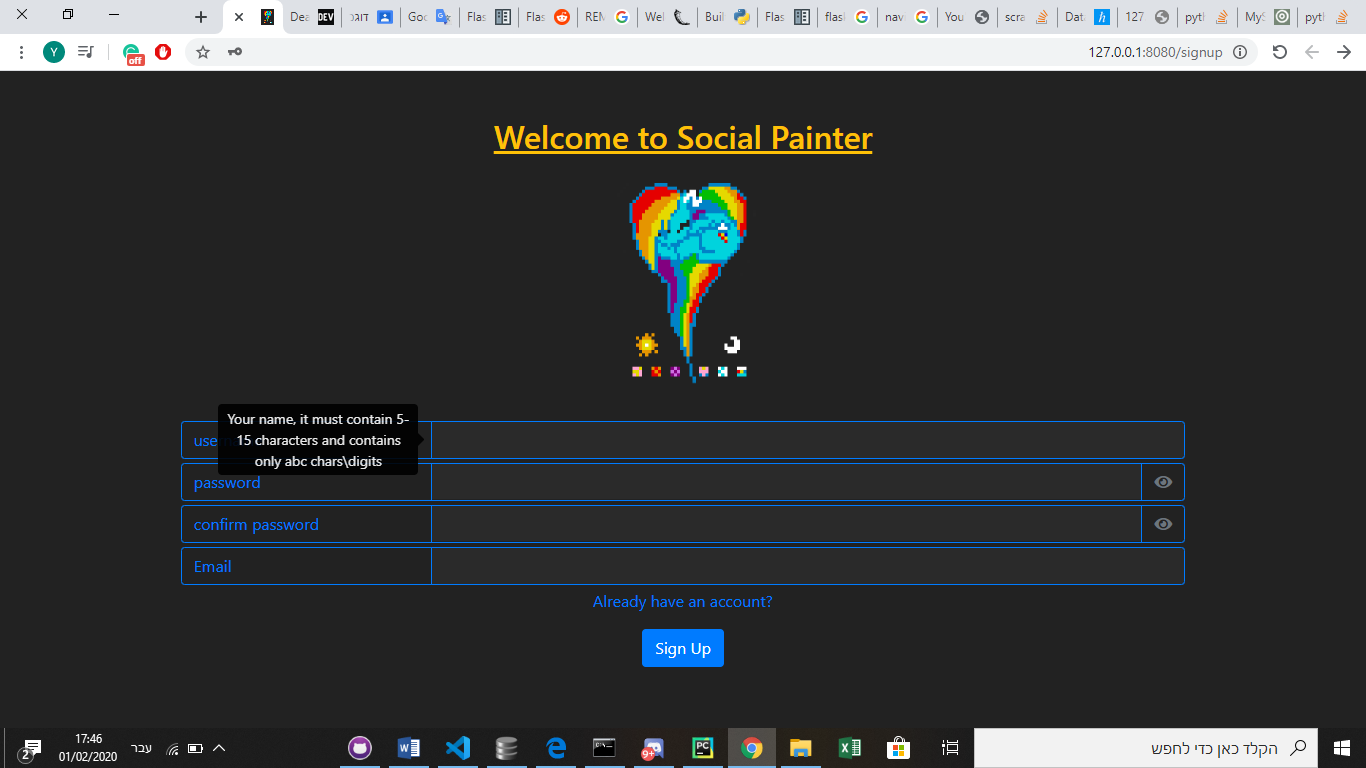
מסכים

## מסך פתיחה

מציג למשתמשים את האפשרות כניסה שלו למערכת, הרשמה או התחברות רגילה.  


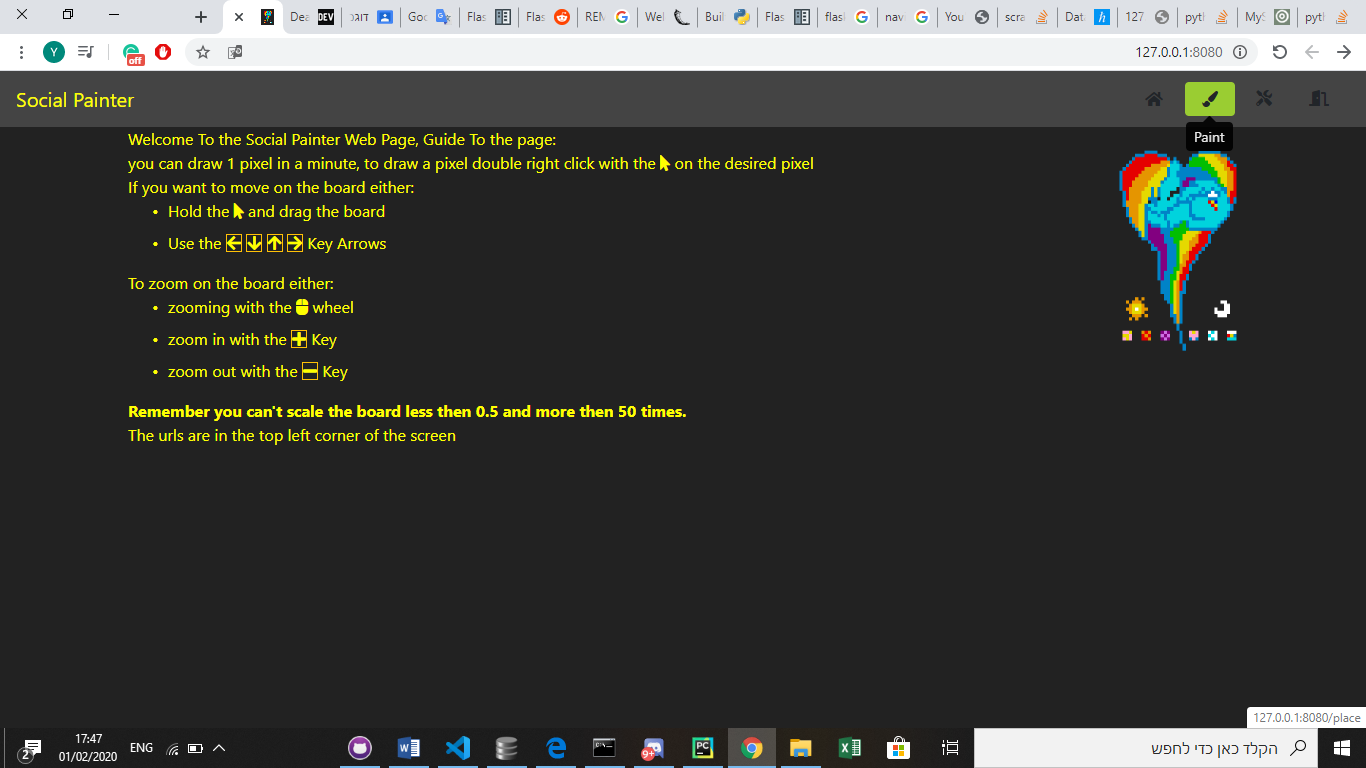
מסך הרשמה

במקרה שלמשתמש אין עדיין חשבון, הוא יכול לפתוח אחד במסך הזה.



בית

השחקן מועבר למסך הזה לאחר ההתחברות, הוא מסביר לשחקן כיצד לפעול בתוכנה.



צייר

האפליקציה עצמה, הצייר השיתופי בו ניתן לראות את כל האפשרויות של השחקן, ללכת למסך הבית, לשנות את צבע המסך, הזמן שנותר עד לסיום הזמן שצריך לחכות.

יומן פרויקט:

ביבליוגרפיה

|  |  |
| --- | --- |
| תאריך | מחקר |
| 6.11.19 | בחירת הפרויקט  https://redditblog.com/2017/04/13/how-we-built-rplace |
| 7.11.19 | למידת:  שרת http  aiophttp  [**https://aiohttp.readthedocs.io/en/stable/**](https://aiohttp.readthedocs.io/en/stable/)  למידת python socket.io - המלצת המורה  socketio python  [**https://python-socketio.readthedocs.io/en/latest/**](https://python-socketio.readthedocs.io/en/latest/) |
| 8.11.19 | השלמת חומר js, css' html שלא ידעתי בw3schools |
| 15.11.19 | מעבר לvscode כי לpycharm אין תמיכה טובה של html  סביבת פייתון  [**https://code.visualstudio.com/docs/python/environments**](https://code.visualstudio.com/docs/python/environments) |
| 19.11.19 | למידת svg - לשימוש בפרויקט:  [**https://css-tricks.com/using-svg/**](https://css-tricks.com/using-svg/)  [**https://mid.as/kb/00134/configuring-server-to-handle-svg-images**](https://mid.as/kb/00134/configuring-server-to-handle-svg-images)  https://he.wikipedia.org/wiki/SVG |
| 23.11.19 | שימוש סופי בgrid  [**https://css-tricks.com/snippets/css/complete-guide-grid**](https://css-tricks.com/snippets/css/complete-guide-grid) |
| 24.11.19 | למידת דרכים יעילות למציאת xy על קנבס  [**https://codepo8.github.io/canvas-images-and-pixels/#display-colour**](https://codepo8.github.io/canvas-images-and-pixels/#display-colour)  [**https://developer.mozilla.org/en-US/docs/Web/API/UIEvent/layerX**](https://developer.mozilla.org/en-US/docs/Web/API/UIEvent/layerX) |
| 29.11.19 | שימוש באובייקט image במקום canvas - ירד  [**https://stackoverflow.com/questions/15541183/canvas-vs-img-performance**](https://stackoverflow.com/questions/15541183/canvas-vs-img-performance)  המשך מציאת דרכים חדשות למציאת xy על קנבס  [**https://stackoverflow.com/questions/17130395/real-mouse-position-in-canvas**](https://stackoverflow.com/questions/17130395/real-mouse-position-in-canvas) |
| 30.12.19 | שימוש באנימציה:  [**https://stackoverflow.com/questions/18023859/possible-to-reverse-a-css-animation-on-class-removal**](https://stackoverflow.com/questions/18023859/possible-to-reverse-a-css-animation-on-class-removal)  כפתור כקישור:  https://stackoverflow.com/questions/2906582/how-to-create-an-html-button-that-acts-like-a-link |
| 5.12.19 | קריאה על bootstrap: [**https://getbootstrap.com/docs/4.4/getting-started/introduction/**](https://getbootstrap.com/docs/4.4/getting-started/introduction/)  (לא שומש עד למאוחר יותר) |
| 10.12.19 | אתרים לביצוע progress bar:  <https://css-tricks.com/css3-progress-bars/>  <https://codepen.io/pmk/pen/emBErK>  <https://www.w3schools.com/w3css/w3css_progressbar.asp> |
| 15.12.19 | חקירת פרוטוקול smtp לשליחת מיילים.  [**https://he.wikipedia.org/wiki/Simple\_Mail\_Transfer\_Protocol**](https://he.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol)  חקירת smptlib לשליחת פקטות smtp  [**https://docs.python.org/3/library/smtplib.html**](https://docs.python.org/3/library/smtplib.html)  חקירת aiosmtplib לשליחת פקטות smtp במצב asynchronize  [**https://aiosmtplib.readthedocs.io/en/stable/overview.html**](https://aiosmtplib.readthedocs.io/en/stable/overview.html) |

ביביליוגרפיה:

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

<https://flask-socketio.readthedocs.io/en/latest/>

<https://www.palletsprojects.com/p/flask/>

<https://redditblog.com/2017/04/13/how-we-built-rplace/>

<https://stackoverflow.com/>

<https://josephg.com/blog/rplace-in-a-weekend/>