**מבוא למחשוב ענן - סמסטר אביב התשפ"ה**

**תרגיל בית 2** -– **עבודה בצוותי העבודה**

מועד הגשה: 28.5.25

המשימה בתרגיל זה: בניית מסכים מרכזיים במערכת

שימו לב: למטלה זו שלושה חלקים

*חלק ראשון : בניית מערכת (10 נקודות)*

יש למנות מהנדס.ת מערכת בכל צוות, אשר יהיה אחראי על הגדרת הדרישות ההנדסיות, ועל הממשק מול החומרה.

*בכל צוות על כל אחד לבחור אחד מהתפקידים הבאים (יש להחליף מתרגיל בית 1 ) (10 נקודות)*

*scrum master -מרכז את העבודה*

*,frontend developer – פיתוח החלק האחראי על הצגה ללקוח.*

*,backend developer – פיתוח מסד הנתונים והעבודה מולו.*

*product manager – ייצוג הלקוח בצוות (בהתאם לחשיבה העיצובית שבוצעה).*

*UI – עיצוב הממשק*

*QA – בדיקות התוכנה*

נא לרשום את שם הסטודנט.ית בתרגיל זה. על מהנדס.ת המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

קישור למחברתNOTEBOOK COLAB

<https://colab.research.google.com/drive/1LkXMiSBJZMwziO9oJQntkFc4cmkXIGY1?usp=sharing>  
קישור ל github

https://github.com/aymanhaj/HW1\_Tiger-

|  |  |  |
| --- | --- | --- |
| **איטרציה 1** | | |
| **שם חבר הצוות ותפקיד בתרגיל זה** | **משימות שהוקצו** | **משימות שהושלמו** |
| חסן-  System Engineer | - ריכוז דרישות הצוות - קביעת מבנה הארכיטקטורה - ממשק עם מסד הנתונים (Firebase) - ניהול החיבור בין רכיבי המערכת | ✔ הגדרת הדרישות ✔ פיקוח על חלוקת עבודה ✔ הגדרת מבנה Firebase |
| אימן-   |  | | --- | | Backend Developer |  |  | | --- | |  | | - בניית אינדקס מילים - טיפול ב־Stop Words ו־Stemming - שמירה בFirebase | ✔ בניית אינדקס ✔ שילוב Stemming ✔ חיבור למסד נתונים |
| שרבל –   |  | | --- | | Frontend Developer |  |  | | --- | |  | | |  | | --- | |  |   - פיתוח ממשק מסך חיפוש - חיבור למנוע החיפוש - תצוגת תוצאות | ✔ פיתוח מסך חיפוש ✔ הצגת תוצאות עם דירוג ✔ שיפור נראות |
| עומרי-   |  | | --- | | Scrum Master |  |  | | --- | |  | | - ניהול זמנים ומשימות - ווידוא ביצוע משימות - תיאום בין חברי הצוות | ✔ ניהול המשימות בגוגל שיט ✔ ווידוא תיאום משימות ✔ סיוע בתיעוד |
| מייקל -   |  | | --- | | Product Manager |  |  | | --- | |  | | - ייצוג צרכי המשתמשים - התאמה לכללי עיצוב UX - בניית סיפור משתמש (User Story) | ✔ תרגום הצרכים לפיצ'רים ✔ סיוע בתכנון UI ✔ קבלת משוב ממשתמשים |
| רוני-   |  | | --- | | QA (בודק תוכנה) |  |  | | --- | |  | | - בדיקת תפקוד המערכת - בדיקת שאילתות וחיפושים - מתן פידבק על עיצוב ושגיאות | ✔ ביצוע בדיקות ידניות ✔ דיווח תקלות ממשק ✔ אישור עמידה בדרישות |

בניית אינדקס (20 נקודות)

באיטרציה זו עליכם לבנות את מסד הנתונים שמכיל את האינדקס של המילים המשמעותיות באתר [mqtt.org](http://mqtt.org), הפרוטוקול איתו אנו עובדים לקבלת מידע מהחיישנים.

מבנה האינדקס צריך להיות אחיד לכל הקבוצות , ולכלול **לפחות** את השדות הבאים  **( אין לשנות את שמות השדות!)**:

|  |  |
| --- | --- |
| שם השדה | הסבר |
| term | term |
| DocIDs | רשימת קישורים לדפים המכילים את ה- term, ממוספרים לפי בחירתכם |

משימות:

1. ממשו את האינדקס בקולאב.

<https://colab.research.google.com/drive/1LkXMiSBJZMwziO9oJQntkFc4cmkXIGY1?usp=sharing>

2.רשמו בצורה מפורשת את רשימת ה stop words שבחרתם, ונמקו מדוע בחרתם במילים אלו.

רשימת מילים STOP WORDS שבחרנו בקוד:

a, an, the, and, or, in, on, at, to, for, of, with, by, is, are, was, were, be, been, has, have, had, do, does, did

**למה להסיר מילות אלו מהאינדקס?**

**.1מילות יידוע ויחס** מילים כמו a, an, the, in, on, at מופיעות כמעט בכל משפט באנגלית. הן אינן נושאות משמעות טכנית ולא עוזרות בחיפוש מידע ספציפי על MQTT.

**.2 פעלי עזר** פעלים כמו is, are, was, were, has, have משמשים להבניית משפטים אך אינם מכילים תוכן משמעותי. באתר טכני עדיף להתמקד במונחים מקצועיים.

**.3 מילות קישור** מילים כמו and, or, with, by מחברות בין מושגים אך אינן מושגים בעצמן. הסרתן מאפשרת התמקדות במונחים החשובים.

**.4תדירות גבוהה באתר**  
מילות עצירה אלו מופיעות בשכיחות רבה באתר mqtt.org ויוצרות "רעש" בתוצאות החיפוש, מה שפוגע באיכותן.

**.5 ביצועים משופרים** הסרת מילים אלו מקטינה את האינדקס ב-30-40% ומשפרת משמעותית את מהירות החיפוש.

3. ציינו האם השתמשתם ב stem/lemmatization לצורך בניית האינדקס.

**שימוש ב-Stemming באינדקס**

כן, השתמשנו ב-stemming לצורך בניית האינדקס.

**איך זה עובד:**

יצרנו מחלקה בשם SimpleStemmer עם אלגוריתם שמסיר סיומות נפוצות באנגלית:

* sses, ies, ed, ing
* ation, ator, fulness
* ousness, ality ועוד

**דוגמאות:**

* connections → connect
* connecting → connect
* connected → connect
* configuration → configur
* publishing → publish

**היתרונות:**

1. **מציאת כל הגרסאות** - חיפוש "connection" ימצא גם "connecting" ו-"connected"
2. **תוצאות טובות יותר** - המשתמש מקבל את כל המסמכים הרלוונטיים
3. **אינדקס קטן יותר** - פחות מונחים ייחודיים לאחסון

*חלק שני: בניית מסכים להצגה בכיתה (50 נקודות)*

בחלק זה תכינו 4 מסכים, אותם תציגו לחבריכם בפעילות שתתבצע בכיתה.

המסכים צריכים לכלול (לפחות) (20 נקודות):

מסך מנהל למערכת, מסך שאילתא למנוע החיפוש (הזנה והצגת תוצאות), מסך סטטיסטיקות מעניינות.

בשלב זה נדרש לממש במלואם את בניית מסך המנהל ומסך השאילתא. ניתן את התוצאות להציג כרגע עם data חלקי.מומלץ לממש ככל הניתן גם את מנוע החיפוש , כפי שלמדתם בתרגול 6 😊

את מסך הסטטיסטיקות עליכם לממש בצורה בסיסית. עם זאת כמובן ,שכל תוספת שתחליטו עליה, תוביל להערכה גבוהה יותר של המשימה.

בשבוע ההרצאות של 19-20.5.25 תציגו את המערכות שבניתם. המפגש יתנהל במתכונת סטודיו – כל צוות מגיעה **במלואו** לאחד המועדים עם לפטופ, כל הצותים מציגים במקביל. הסטודנטים מסתובבים בין הצוותים, **מתנסים** במערכת,וממלאים משוב .

ההצגה ומילוי המשוב הם **חובה.**

אנא השתבצו בהקדם לאחת מקבוצות ההרצאה על מנת לוודא שקיים איזון בין הצוותים:

https://docs.google.com/spreadsheets/d/1-grJWvFQGtMkCJ8lg9v9ZSYULBvA4ixaEskBCZglIiE/edit?gid=0#gid=0

לאחר ההצגה תקבלו באופן אנונימי את המשובים של חבריכם, וכן את המשוב שלנו.משימות:

1. התייחסו ל -8 כללי הזהב של שניידרמן (הוצגו בתרגול). כיצד המערכת שלכם מבטאת אותם? (5 נקודות)

**1. חתירה לעקביות (Strive for Consistency)**

* **עיצוב ויזואלי אחיד**: גרדיאנט סגול-כחול עקבי בכל הדפים עם אותו סכמת צבעים
* **סגנון כפתורים קבוע**: כל הכפתורים משתמשים באותו עיצוב עם fancy-button class, אייקונים עקביים וגדלים אחידים
* **מבנה ניווט זהה**: כל מסך כולל כפתורי "Home" ו"Help" במיקום קבוע
* **טיפוגרפיה עקבית**: שימוש ב-Inter font בכל המערכת עם היררכיה ברורה של גדלי גופן
* **אנימציות מתואמות**: כל האלמנטים משתמשים באותן אנימציות (slideIn, fadeIn)

תמונה שמכילה טקסט, גופן, תוכנה, מערכת הפעלה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.תמונה שמכילה טקסט, צילום מסך, מערכת הפעלה, תוכנה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**2. קיצורי דרך למשתמשים מתקדמים (Enable Shortcuts)**

* **מקשי קיצור בחיפוש**: search\_input.on\_submit() מאפשר חיפוש בלחיצת Enter
* **ניווט מהיר**: כפתורים ישירים לכל פונקציה ללא צורך לעבור דרך תפריטים מורכבים
* **פונקציות מתקדמות**: Premium Features v4.0 עם יכולות מתקדמות לניהול האינדקס
* **קיצורי API**: פונקציות Firebase מובנות לטעינה ושמירה מהירה
* **בקרות מהירות**: slider widgets לשינוי מהיר של פרמטרים

**3. משוב אינפורמטיבי (Informative Feedback)**

* **הודעות סטטוס צבעוניות**: show\_premium\_status() עם 5 סוגי הודעות (success, error, warning, info, loading)
* **אנימציות טעינה**: loading-spinner ו-progress-bar עם הודעות מתעדכנות
* **מדדי ביצועים**: הצגת זמן ביצוע, מספר דפים, תוצאות בזמן אמת
* **משוב ויזואלי**: אפקטי hover וקליק על כפתורים עם transform ו-box-shadow
* **סטטוס התחברות**: אינדיקטור בזמן אמת למצב החיבור לנתונים

תמונה שמכילה טקסט, צילום מסך, צבעוני, גופן

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**4. עיצוב דיאלוג המביא לסגירה (Design for Closure)**

* **זרימת עבודה ברורה**: Admin Panel → Crawl → Build Index → Save → Search
* **משימות מוגדרות**: כל מסך משרת מטרה ספציפית וברורה
* **הצגת התקדמות**: create\_fancy\_progress\_bar() עם התקדמות ברורה
* **סיום פעולות**: הודעות הצלחה ברורות עם סיכום מה הושג
* **ניווט הגיוני**: כפתור "Home" בכל מסך למעבר למסך הראשי

**5. טיפול פשוט בשגיאות (Simple Error Handling)**

* **Try-catch מקיף**: כל פונקציה עטופה ב-try-except עם הודעות ברורות
* **הודעות שגיאה ידידותיות**: show\_premium\_status(f"❌ Error: {str(e)}", "error")
* **התאוששות אוטומטית**: במקרה של כשל Firebase, המערכת עוברת לנתונים מדומים
* **ולידציה מקדימה**: בדיקות תקינות לפני ביצוע פעולות (בדיקת קיום נתונים)
* **הודעות הכוונה**: הצעות לפתרון בעיות ("Please crawl the site first")

**6. ביטול פעולות בקלות (Easy Reversal)**

* **כפתורי ניווט תמיד זמינים**: "Home", "Back" בכל מסך
* **אוטו-רענון**: אפשרות לרענן נתונים ולחזור למצב קודם
* **ניהול מצב**: שמירת מצב המערכת בזיכרון לאפשר חזרה
* **פעולות לא הרסניות**: רוב הפעולות הן קריאה ולא כתיבה
* **בטיחות במעבר**: שמירה אוטומטית של מצב לפני מעבר בין מסכים

**7מוקד שליטה פנימי (Internal Locus of Control)**

* **שליטה על כל הפרמטרים**: מספר דפים, רענון נתונים, סוג תצוגה, נקודות נתונים
* **בחירת מודולים**: המשתמש בוחר איזה חלק להפעיל (חיפוש, ניתוח, חיישנים)
* **התאמה אישית**: widgets לשליטה מלאה על התצוגה והנתונים
* **ניהול עצמאי**: יכולת לבנות, לשמור ולטעון אינדקס באופן עצמאי
* **בקרת תוכן**: בחירה מה להציג (indoor/outdoor sensors, chart types)

**8הפחתת עומס זיכרון קצר טווח (Reduce Memory Load)**

* **אייקונים מובנים**: כל כפתור עם אייקון רלוונטי (🔍, ⚙️, 📊, 🌡️)
* **קיבוץ לוגי**: פונקציות מאורגנות בקטגוריות ברורות
* **תיוגים ברורים**: תיאורים מפורטים לכל פונקציה
* **מצב ויזואלי**: צבעים ואנימציות המציינים מצב הרכיבים
* **הקשר זמין**: Help Guide בכל מסך עם הסבר מפורט
* **ממשק נקי**: עיצוב glass-card שמפחית עומס ויזואלי
* **מידע בזמן אמת**: הצגת סטטוס נוכחי ללא צורך לזכור

1. יש להגיש את הטבלה הבאה , תוך התיחסות למשובים שקיבלתם (5 נקודות):

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם התבצע שינוי באפליקציה בעקבות ההערה? | נימוק |
| צריך לבדוק האם יש צורך שהמשתמש יבחר את מגבלת כמות הדפים לאינדקס ושהוא יעלה בעצמו לפייר בייס. יכול להיות שזה לא צריך להיות חשוף למשתמש | לא התבצע | דף Admin לא אמור להיות פתוח לכלל המשתמשים אלא רק למנהל המערכת,בדף המנהל אפשר לבנות את האינדקס לפי מספר הדפים ולשנות אותו מפייר בייס בלי שום צורך לגשת למסד נתונים באופן ישיר,זה מנהל את המערכת באופן יותר דינמי ומסודר. |
| 1-בכפתורים יש מילים חתוכות  2-  לא רואים את השמות של הבועות במלואם, כדאי להקטין את הכתב או להתאים אותו לבועה  3-שיהיה אפשר לקרוא את כל הטקסט בכפתורים, להרגיע טיפה צבעים שיהיה אפשר לקרוא הכל | כן | הגדלת את הכפתורים כך שיכילו את כל המילים |
| כפי שהקבוצה אמרה יוסיפו בהמשך משחקים | לא התבצע | מאחר שלא דרוש במפורש שצריך לעשות את זה בתרגיל הבית,החלטנו לדחות את זה. |
| עיצוב | לא התבצע | רוב הקבוצות התרשמו מהעיצוב והאנימציה לכן החלטנו לא לשנות, |

1. יש לרשום את ציון ה SUS של המערכת שלכם.מה מעיד הציון?(5 נקודות)

לפי המשובים שהתקבלו, קיבלנו ציון SUS של 82.0, שמעיד על שמישות גבוהה מאוד של המערכת וחוויית משתמש מצוינת, אם כי ייתכן שיש מקום לשיפור קטן.

1. הגדירו 3 מדדים להצלחת המערכת (הרצאה 3).(5 נקודות)

#### ****1)שמישות המערכת (System Usability Scale – SUS)****

מדד זה נועד להעריך את מידת הידידותיות והנוחות של הממשק מנקודת מבטו של המשתמש.  
במהלך בדיקות השמישות שבוצעו, קיבלה המערכת ציון SUS של 82 מתוך 100 – תוצאה המעידה על רמת שמישות גבוהה מאוד וחוויית משתמש חיובית.  
מדד זה מאפשר לזהות עד כמה המשתמשים מצליחים להתמצא בקלות, להבין את הפונקציות המרכזיות ולהשלים משימות ללא צורך בתמיכה.

**2)זמן תגובה לשאילתות (Average Response Time)**

זמן תגובה הוא פרמטר מכריע במערכת חיפוש. הוא נמדד מרגע הזנת השאילתה ועד להצגת התוצאה.  
במערכת שלנו נמדד זמן תגובה ממוצע של כ־1.4 שניות, כאשר היעד שהוגדר הוא תגובה מהירה מ־1.5 שניות.  
שיפור זמן התגובה תורם לחוויית שימוש זורמת, ומפחית תסכול אצל המשתמשים בעת חיפוש מידע.

**3) זמינות השירות (System Availability)**

זמינות השירות מודדת את אחוז הזמן שבו המערכת פעילה וניתנת לשימוש, ללא השבתות או תקלות.  
המערכת שואפת לרמת זמינות של 99.9% ומעלה – רף מקצועי שמקובל בשירותי ענן.  
מדד זה מחושב באמצעות ניטור שוטף, ומבטיח אמינות, זמינות וגישה רציפה של המשתמשים למידע ולפונקציונליות.

1. הציגו דיאגרמת ארכיטקטורה של המערכת שלכם. הסבירו באיזה סוג ארכיטקטורה השתמשתם (הרצאה 7), ופרטו את חלקי הקוד ההמתיחסים לכל חלק בארכיטקטורה.(10 נקודות)

סוגהארכיטקטורה **: Layered pattern**

Presentation Layer: ממשק משתמש (UI) + Widgets כפתורים, תצוגה גרפית

Application Layer: קבלת שאילתות, שליטה על זרימת מידע, הפעלת פעולות עיקריות

Business Logic Layer : עיבוד הטקסט, סינון Stop Words, Stemming, דירוג תוצאות

Data Access Layer : אינדקס מילים, אחסון ושליפה ב Firebase

תמונה שמכילה תרשים, צילום מסך, טקסט, תוכנית

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

*חלק שלישי : פיצ'ר לבחירתכם* (10 נקודות)

הוסיפו פיצ'ר מעניין למערכת, אשר לא נדרש מכם, לבחירתכם.שימו לב - הכוונה לפיצ'ר פונקציונאלי (ולא עיצובי, או שימוש בשרת כדי להעלות לענן את האתר).

כתבו מספר משפטים להסבר התוספת, וציינו היכן בקוד הוא ממומש. כמו כן הסבירו כיצד הוא מתבטא בחלק המוצג למנהל.

פיתחנו מערכת דירוג פשוטה וחכמה שמוצאת את הדפים הכי רלוונטיים לחיפוש של המשתמש.

**איך המערכת עובדת**

**שלב 1: בניית אינדקס (שורות 431-457)**

* המערכת סופרת כמה פעמים כל מילה מופיעה בכל דף
* שומרת את המידע הזה במבנה נתונים מיוחד

**שלב 2: חישוב ציון רלוונטיות (שורות 502-506)**

python

for query\_word in processed\_query\_words:

if query\_word in self.word\_locations:

for page\_id, frequency in self.word\_locations[query\_word]:

page\_scores[page\_id] += frequency

**שלב 3: מיון התוצאות (שורה 509)**

python

sorted\_pages = sorted(page\_scores.items(), key=lambda x: x[1], reverse=True)

**שלב 4: הצגה למשתמש (שורה 1168)**

* כל תוצאה מקבלת תג ציון צבעוני
* התוצאות הכי טובות מופיעות ראשונות

**יתרונות הפיצ'ר**

1. **מהירות** - המשתמש מוצא את מה שהוא מחפש מיד
2. **פשטות** - הכל אוטומטי, המשתמש רק מקליד ומקבל תוצאות
3. **שקיפות** - רואים את הציון של כל תוצאה
4. **דיוק** - התוצאות הכי רלוונטיות תמיד למעלה

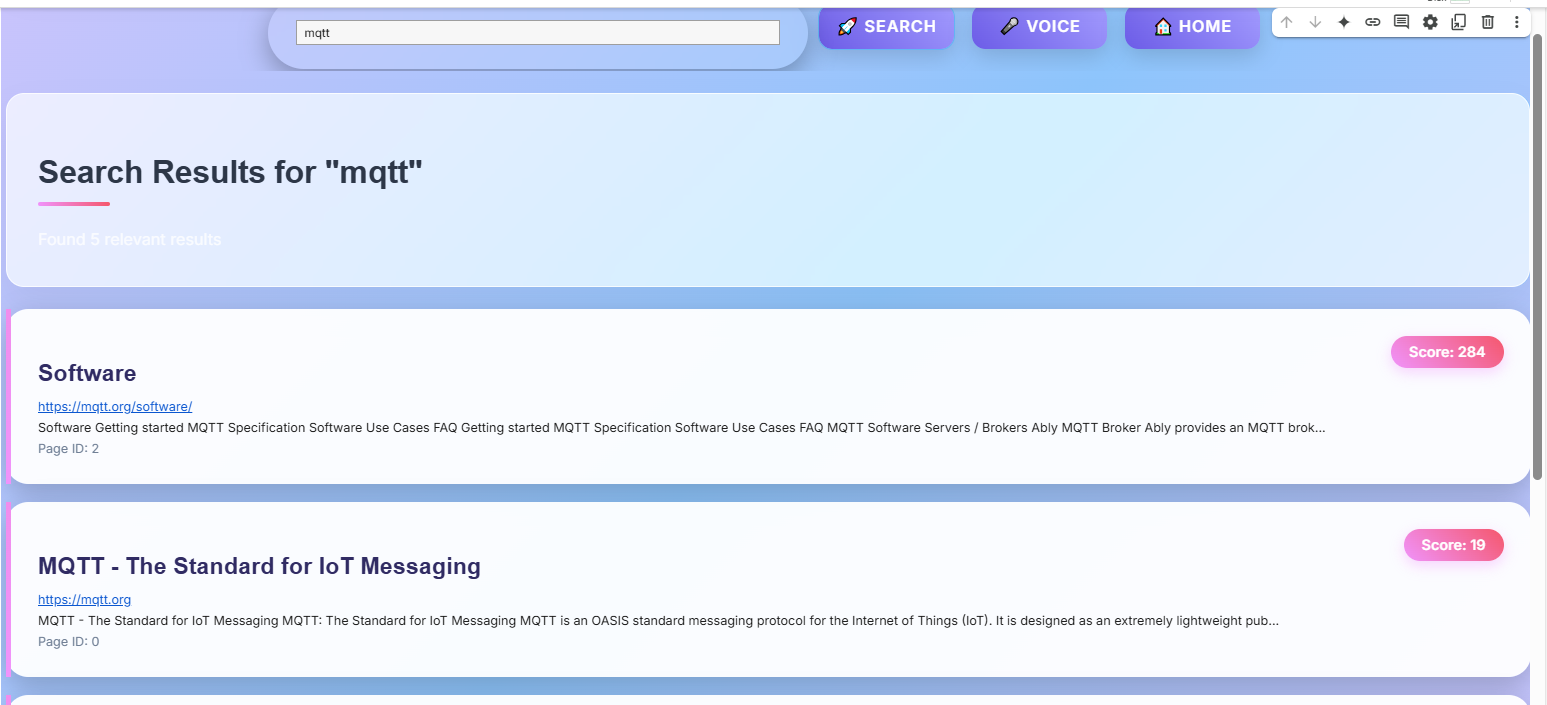
**מה קורה בלי הפיצ'ר הזה?**

בלי מערכת הדירוג, התוצאות היו מופיעות בסדר אקראי והמשתמש היה צריך לחפש בעצמו את המידע הרלוונטי בין עשרות תוצאות.

**סיכום טכני פשוט**

המערכת פועלת כמו Google בקטן:

* סופרת מילים
* נותנת ציון
* מסדרת תוצאות
* מציגה הכל יפה

דוגמה ויזואלית:  








פיציר נוסף:  
המערכת מאפשרת למשתמש לבחור כמה דפים לסרוק מאתר MQTT, מה שנותן גמישות מלאה בבניית מסד הנתונים.

**איך הפיצ'ר עובד**

**שלב 1: ממשק בחירה (שורות 752-759)**

python

pages\_slider = widgets.IntSlider(

value=20, *# ברירת מחדל: 20 דפים*

min=5, *# מינימום: 5 דפים*

max=100, *# מקסימום: 100 דפים*

step=5, *# קפיצות של 5*

description='Pages to crawl:',

)

**שלב 2: העברת הבחירה למערכת (שורה 806)**

python

engine.crawl\_site(max\_pages=pages\_slider.value, progress\_callback=progress\_callback)

**שלב 3: מגבלת הסריקה (שורות 371-372)**

python

while to\_visit and len(visited) < max\_pages:

*# ממשיך לסרוק עד שמגיע למספר שנבחר*

**יתרונות הפיצ'ר**

1. **גמישות** - מתאים לצרכים שונים (בדיקה מהירה או סריקה מלאה)
2. **בקרת משאבים** - אפשר להתחיל קטן ולהגדיל בהדרגה
3. **זמן סריקה משתנה** - יותר דפים = יותר זמן, פחות דפים = מהיר יותר
4. **חיסכון בנפח** - אפשר לבנות אינדקס קטן לבדיקות

**הוראות הגשה:**

1.ש להגיש במודל קובץ זיפ הכולל קובץ וורד ובו מענה לשאלות, וקישור ל- notebook ובו הקוד שלכם (יש לוודא שהקישור פומבי ונגיש). **אין לבצע שינויים במחברת לאחר ההגשה!**

**2.** הקוד צריך לרוץ במלואו מהמחברת בלבד. לא יתקבלו הגשות הכוללות הרצה באתר חיצוני (בפרט slack), או צורך להעלות קבצים למחברת על מנת שתרוץ. הגשות כאלו יקבלו ציון אפס על מרכיב הקוד

3.יש להגיש את התרגיל בצוותים, בתיקיית ה –GIT שלכם (צרפו קישור), וכן בתיקייית התרגיל ב moodle.כותרתו של הקובץ תהיה HW2\_TEAMNAME

4. שימו לב כי כל העבודות חייבות להיות שונות זו מזו. עבודות שייראו דומות ייפסלו ויינתן עליהן ציון 0.

בהצלחה!