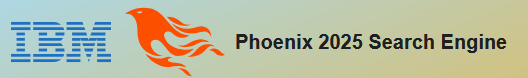
**פרויקט קורס מבוא למחשוב ענן – קבוצת Phoenix**



חברי הצוות

* שחר ברנסון
* עומר גולדשטיין
* ישראל אוחיון
* כפיר אמויאל
* שלומי פרידמן
* ולדי טרינטר
* [קישור לקוד ב-colab](https://colab.research.google.com/drive/15etxIQVNJB8syQAj_0bftT9qIyGDJHzD?usp=sharing)
* [קישור לפרויקט ב-github](https://github.com/ShlomiFridman/PhoenixProject2025)

תוכן העניינים

[סעיף 1 – מהות המוצר ומרכיביו 2](#_Toc189302236)

[סעיף 2 – ארכיטקטורת המערכת 3](#_Toc189302237)

[סעיף 3 – דרישות לא פונקציונליות 4](#_Toc189302238)

[סעיף 4 – ביקורת עמיתים וציון SUS 5](#_Toc189302239)

[סעיף 5 - תיק תחזוקה 6](#_Toc189302240)

[סעיף 6 – תיק משתמש 11](#_Toc189302241)

[סעיף 7 – שקיפות אלגוריתמית 12](#_Toc189302242)

[סעיף 8 – אתגרים במהלך העבודה 12](#_Toc189302243)

[סעיף 9 – משובים 12](#_Toc189302244)

[סעיף 10 – מקורות ושימוש בכלי AI 15](#_Toc189302245)

סעיף 1 – מהות המוצר ומרכיביו

1. מודלים ופיצ'רים מעניינים:

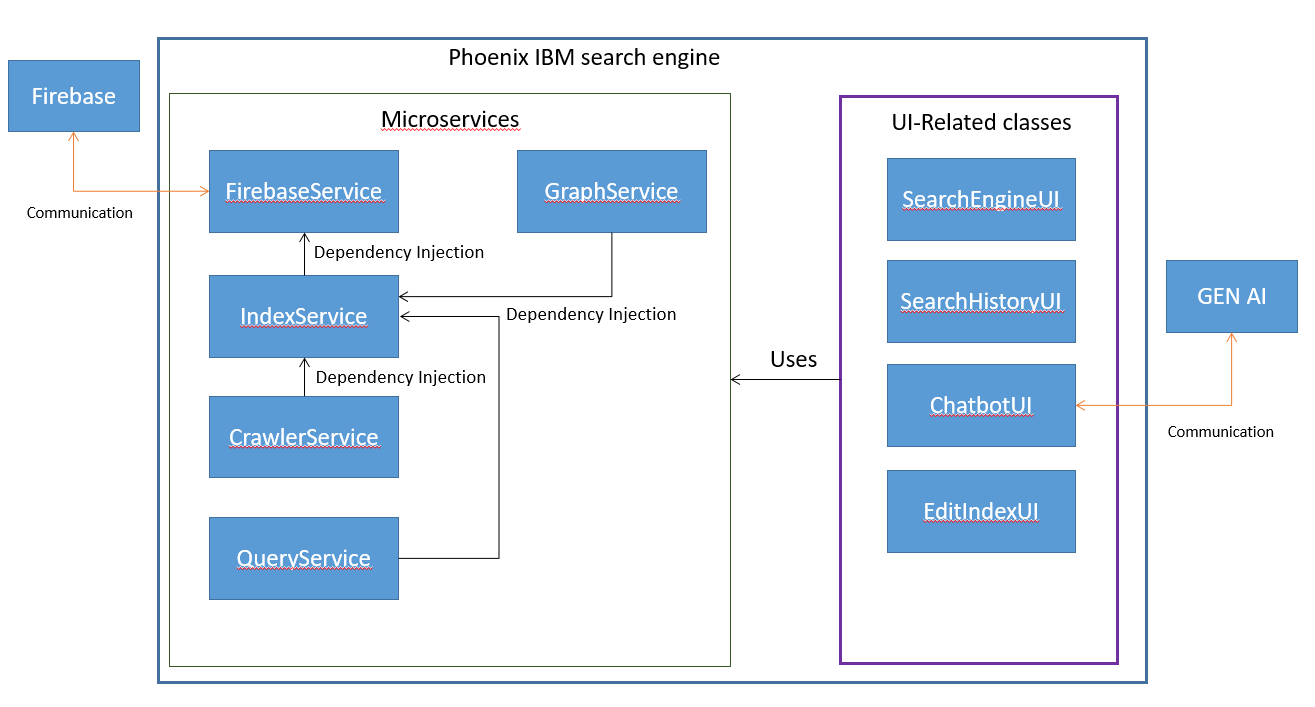
במערכת שלנו, בנוסף למנוע חיפוש, הוספנו 2 פיצ'רים נוספים:

1. שמירת היסטוריה – אנו שומרים את ה-5 חיפושים האחרונים של המשתמש, ועבור כל חיפוש את ה-10 תוצאות הראשונות שלו.
2. צ'אטבוט – מסך שמאפשר שיח עם צ'אטבוט על מנת לקבל מידע על מושגים, ותכנים הקשורים ל-IBM, הצ'אטבוט משתמש ב-API של Genai.
3. הצגת snippet – אנו מציגים snippet של כל תוצאת חיפוש, על מנת לתת למשתמש מידע נוסף על העמוד לפני שהוא נכנס עליו.
4. Microservices העיקריים שנעשה בהם שימוש:
5. FirebaseService - השירות שדרכו מתבצעת כל התקשורת עם ה-Firebase.
6. IndexService - השירות שמנהל את האינדקס במערכת, את המילים, הקישורים שלהן, וכמות המופעים בכל קישור.
7. CrawlerService - השירות שמבצע Crawling לפי עמוד בודד או על "עץ" של כתובת.
8. QueryService - השירות שאחראי על מתן תשובה לשאילתות, ודירוג עמודים. היתרון של שירות זה הוא שהוא עובד בנפרד משאר השירותים, הוא עדיין צריך שירות של index אבל הוא מקבל אותו על ידי dependency injection מה שמאפשר פיתוח מודולרי עם תלויות חלשות יותר.
9. GraphService - השירות שאחראי על בניית הגרפים במערכת (heatmap ו-bar), הוא מקבל dependency injection של שירות אינדקס ולפיו בונה את הגרפים ומציג אותם ב-output שהוגדר לו באתחול.
10. KPI מרכזיים:

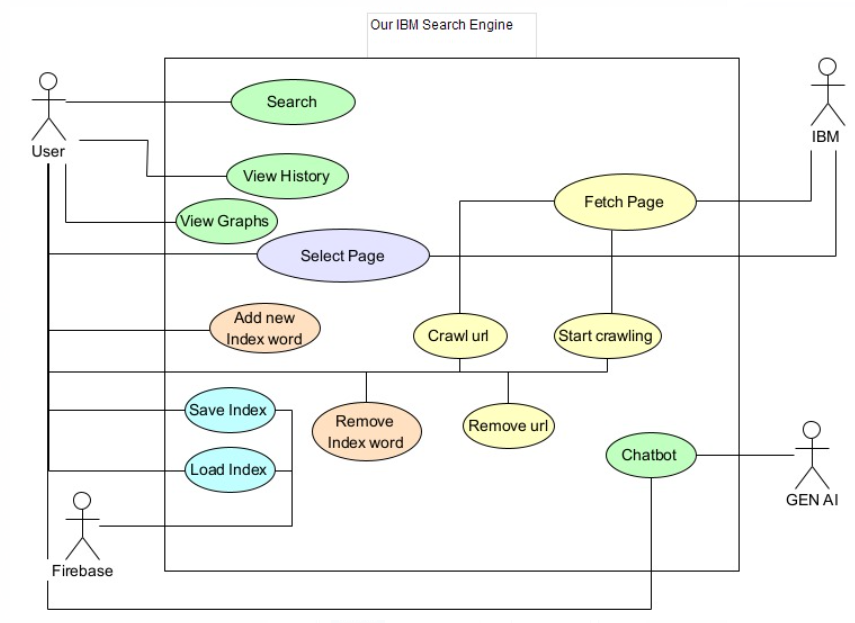
* User Satisfaction
* שהמשתמש יהיה מרוצה מתוצאת החיפוש, כלומר שהתוצאות חיפוש יכילו תוצאות נכונות.
* מדידה: שתהיה לפחות תוצאה אחד נכונה \ רלוונטית ב-5 התוצאות הראשונות.
* Performance
* שהמשתמש לא ימתין יותר מדי זמן לתוצאת החיפוש.
* מדידה: אחרי לחיצה על כפתור החיפוש לא יעברו יותר מ-3 שניות עד שתוצג תוצאה כלשהי.
* Scalability & Elasticity
* שהמערכת תהיה מודולרית, וקלה להרחבה.
* מדידה: פיתוח מודולרי, שכל פונקציה ומחלקה תהיה עצמאית, כלומר בלי תלות במשתנים גלובלים.

סעיף 2 – ארכיטקטורת המערכת

1. תרשים ומאפיינים מרכזיים



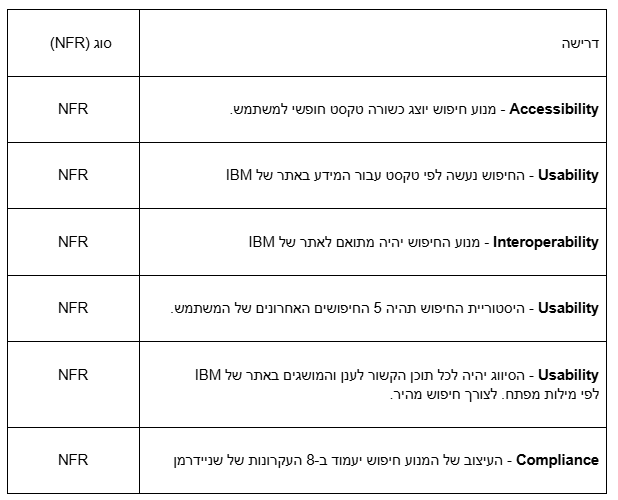
1. Use-Case



1. דרישות פונקציונליות



סעיף 3 – דרישות לא פונקציונליות



סעיף 4 – ביקורת עמיתים וציון SUS

1. ביקורת עמיתים – מהסדנא בשבוע 9

|  |  |  |
| --- | --- | --- |
| **הערת משוב** | **האם התבצע שינוי באפליקציה בעקבות ההערה?** | **נימוק** |
| לשפר את העיצוב | כן | הוספנו בצע ריקע, ועיצוב משופר למסכים ובפרט לרכיבים בכל מסך. |
| זמן תפעול מהיר יותר \ יותר פירוט | כן | נוסיף טקסט שלמשתמש יהיה יותר ברור איך המערכת עובדת. |
| הכנסת כמות דפים לקרולר כפרמטר | לא | כבר יש פיצ'ר להוספת דף חדש במסך עריכת האינדקס. |
| הצגת האינדקסים במערכת בפלט החיפוש | לא | אנו לא רוצים לחשוף את האינדקס שלנו במסך החיפוש, על מנת שלא יהיה עומס קוגנטיבי על המשתמש, המשתמש יוכל לראות את האינדקס במסך עריכת האינדקס, ולקבל מידע עליו שם ובמסכי סטטיסטיקה. |
| זמני ריצת החיפוש, אולי כדאי לייעל את החיפוש מראש ולשמור בDB | לא | כבר מוצג למשתמש זמן החיפוש במסך התוצאות, ואנו לא רוצים לשמור את התוצאות בבסיס נתונים כדי למנוע עומס (לאחר בדיקה ה-Firebase לא אוהב את זה) בנוסף האינדקס יוכל להתעדכן בהמשך (בין במילים או בדפים חדשים \ נוספים) ולכן אנו מעדיפים שעבור כל שאילתה יהיה חיפוש חדש ועדכני. |
| יש לשפר את האינדקס חיפוש | כן | נוספו מילים חדשות לאינדקס |
| הגדל הכתב | כן | למרות שבדפדפן יש אפשרות מובנת להגדלת הכתב, הגדלנו את הכתב במקומות שלאחר בדיקה הרגשנו שכן יש צורך, כגון בתקציר של עמוד בתוצאת החיפוש. |
| הגרף עם המספרים היה קצת מסורבל ללא הסבר מפורמט מהאחראי של האתר | כן | הוספנו הסברים לגרפים, צמצמנו את כמות המידע שמוצג בכל גרף. |

1. ציון SUS - מהסדנא בשבוע 9

לקחנו את התוצאות במשוב ה-SUS מהסדנא ועשינו את השלבים הבאים:

* חישבו ממוצע של התוצאות לכל שאלה.
* עבור הממוצעים של השאלות במקומות האי-זוגיים החסרנו אחד.
* עבור הממוצעים של השאלות במקומות הזוגיים לקחנו את הערך 5 פחות הממוצע.
* חישבנו את הסכום של התוצאות לאחר החיסורים, והכפלנו אותו ב-2.5.

קבלנו ציון SUS: 77.64 מתוך מאה.

לאחר מעבר על המשובים של שאלון ה-SUS מהסדנא, ראינו שהרבה הרגישו שהמערכת קשה לתפעל ולהבין. לכן הוספנו יותר הסברים עבור המשתמש בכל מסך.

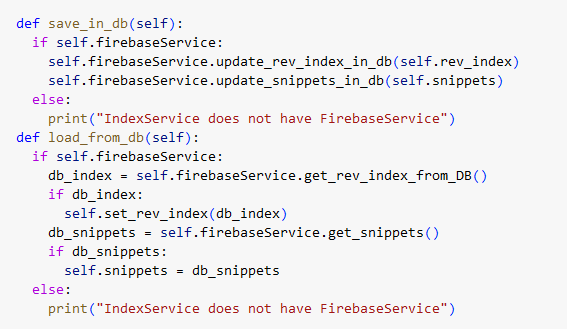
סעיף 5 - תיק תחזוקה

* הקובץ המרכזי הוא הקוד בגוגל colab.
* ניתן למצוא פירוט נוסף בקובץ DeveloperKit.docx

תיאור של האובייקטים המרכזיים:

|  |  |  |
| --- | --- | --- |
| **FirebaseService** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | db\_url: str = 'default\_db\_url' | אתחול המחלקה עם כתובת URL של Firebase. |
| get\_rev\_index\_from\_DB | אין | שליפת האינדקס ההפוך ממסד הנתונים. |
| update\_rev\_index\_in\_db | index\_p: dict | עדכון האינדקס ההפוך ב-Firebase. |
| get\_snippets | אין | שליפת ה-snippets ממסד הנתונים. |
| update\_snippets\_in\_db | snippets\_p | עדכון ה-snippets ב-Firebase |

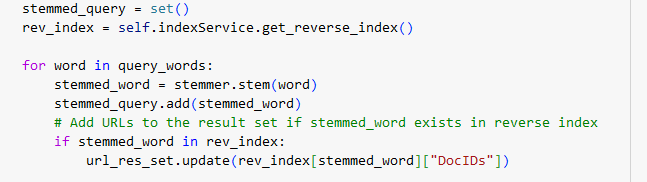
דוגמת שימוש:



|  |  |  |
| --- | --- | --- |
| **IndexService** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | index: dict = None, firebaseService: FirebaseService | אתחול אינדקס ושירות Firebase. |
| process\_soup | url: str, soup: BeautifulSoup | עיבוד תוכן HTML כדי להוסיף מונחים לאינדקס. |
| get\_reverse\_index | אין | מחזיר את האינדקס ההפוך. |
| get\_url\_index | url: str | מחזיר מידע באינדקס עבור URL מסוים. |
| set\_rev\_index | newRevIndex: dict | טוען אינדקס חדש. |
| add\_new\_word | word: str | מוסיף מילה חדשה לאינדקס. |
| remove\_word | word: str | מסיר מילה קיימת מהאינדקס. |
| remove\_url | url: str | מסיר URL מהאינדקס. |
| save\_in\_db | אין | שומר את האינדקס ב-Firebase. |
| load\_from\_db | אין | טוען אינדקס מ-Firebase. |
| get\_snippet | url: str | מחזיר את Snippet עבור URL מסוים. |
| index\_toString | minView: bool = True | מחזיר את ה-toString בצורה מורחבת או מינימלית |
| resetService | אין | מאפס את השירות |

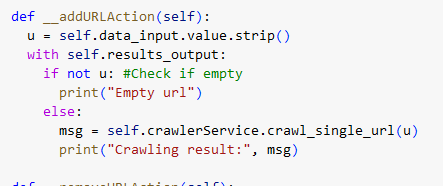
דוגמת שימוש:

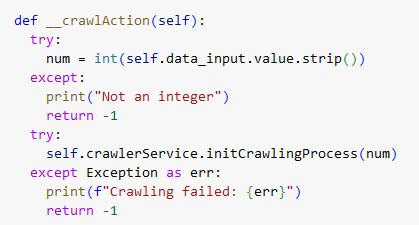




|  |  |  |
| --- | --- | --- |
| **CrawlerService** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | indexService: IndexService, baseURLs: set,  maxDepth: int = 100 | אתחול המחלקה עם הגדרות לזחילה. |
| initCrawlingProcess | max\_depth: int = 10 | מתחיל תהליך זחילה. |
| crawl\_website | base\_url: str, max\_pages: int = 100 | זוחל אתר בודד ומחזיר רשימת כתובות. |
| crawl\_single\_url | url: str | זוחל כתובת URL בודדת. ומחזיר הודעה עם התוצאה |

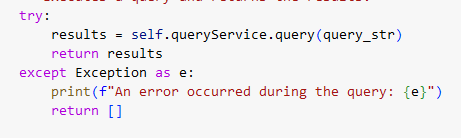
דוגמת שימוש:





|  |  |  |
| --- | --- | --- |
| **QueryService** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | indexService: IndexService | אתחול המחלקה עם שירות אינדקס. |
| query | query: str | מחפש ומחזיר תוצאות ממוינות לפי ה-rank בסדר יורד, התוצאות בפורמט: |
| get\_history | אין | מחזיר היסטוריית חיפושים. |
| rank\_url | url: str, query\_words: set | מדרג כתובת URL על סמך מילות חיפוש. |

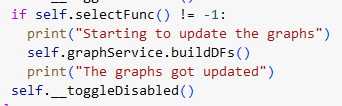
דוגמת שימוש:



|  |  |  |
| --- | --- | --- |
| **GraphService** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | indexService: IndexService, heatmap\_output: widgets.Output,  bar\_output: widgets.Output  max\_heatmap\_items: int = 15, max\_bar\_items: int = 15 | אתחול המחלקה ליצירת גרפים. |
| buildDFs | אין | בונה את מסגרות הנתונים לניתוח ומציג אותם מחדש ב-output שהתקבל באתחול. |
| build\_heatmap | אין | מחזיר גרף חום של מונחים משותפים. |
| build\_barChart | אין | מחזיר גרף עמודות של שכיחות מונחים. |

* הנתונים שמוצגים בגרפים עוברים סינון לפי כמות מנמלית של url-ים משותפים \ כמות מופעים באתר.

דוגמת שימוש:



מחלקות ה-UI:

מחלקות אלו מקבלות קלט מהמסכים ובעזרת השירותים מציגות למשתמש את התוצאות.

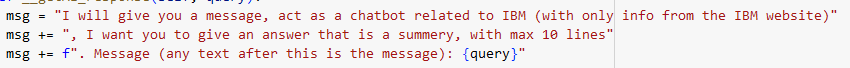
|  |  |  |
| --- | --- | --- |
| **SearchEngineUI** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | queryService: QueryService, history\_service: SearchHistoryUI | אתחול עם שירות שאילתות והיסטוריה. |
| display | אין | מציג את ממשק המשתמש. |
| perform\_search | b | מבצע חיפוש ומציג תוצאות. |

|  |  |  |
| --- | --- | --- |
| **SearchHistoryUI** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | אין | מאתחלת את ממשק המשתמש לצפייה בהיסטוריית החיפושים, כולל מידע טקסטואלי ותיבת פלט ריקה. |
| save\_search | Query: str,  results [{}] | שומרת שאילתת חיפוש ותוצאותיה להיסטוריה, שומרת רק את 5 החיפושים האחרונים ומעדכנת את התצוגה. |
| display\_history | אין | מציגה את חמשת החיפושים האחרונים (לפי הסדר), עם מספר תוצאות לכל חיפוש, בתצוגה אינטראקטיבית. |
| get\_output\_widget | אין | מחזירה את ה-Widget של היסטוריית החיפושים שניתן לשלב בטאב או ממשק אחר. |

|  |  |  |
| --- | --- | --- |
| **ChatbotUI** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | indexService: IndexService, | מאתחלת את האובייקט, כולל קונפיגורציה של מודל ה-AI והכנה של ממשק המשתמש הגרפי |
| display | אין | מציגה את ממשק המשתמש הגרפי של הצ'אטבוט. |

הערות:

* הצ'טבוט משתמש ב-API של GenAI על מנת לבנות תשובה להודעה של המשתמש, עם prompt שמדריך את ה-AI שהוא אמור לענות בתור צ'טבוט של אתר IBM ושהוא צריך לספק תשובה קצרה במקסימום 10 שורות. ה-prompt הוא:



|  |  |  |
| --- | --- | --- |
| **EditIndexUI** | | |
| פונקציה | פרמטרים | תיאור |
| \_\_init\_\_ | indexService: IndexService, crawlerService: CrawlerService, graphService: GraphService | אתחול עם שירותי אינדקס, זחילה וגרפים. |
| display | אין | מציג את התפריט לעריכת אינדקס. |

סעיף 6 – תיק משתמש

הסבר כללי על המערכת:  
מנוע החיפוש פניקס 2025 מציע מספר שימושים למשתמש במספר מסכים.

* המשתמש יכול לעבור בין המסכים על ידי לחיצה על ה-tab הרצוי.
* בכל מסך מוצג למשתמש מידע מה תפקיד המסך, ומה הוא יכול לבצע בו.

המסכים הם:

* מנוע חיפוש עבור האתר של IBM: על מנת לבצע חיפוש, הכנס טקסט לתיבת ה-Query ולחץ על כפתור Search התוצאות יוצגו על המסך, בכל עמוד יש 10 תוצאות, וניתן לדפדף בין העמודים על ידי הכפתורים Previous ו-Next.
  + טעות אפשרית: בגלל פגם בחבילת ה-widgets אם לוחצים על כפתור ה-search מהר מדי הערך של שדה השאילה לא עדכני ויתבצע חיפוש עם הטקסט הלא נכון.

עשינו תיקון זמני של ביטול האפשרות חיפוש על ידי לחיצה של enter בתיבת הטקסט, וביטול הכפתור עד שהמשתמש גמר להכניס את הטקסט בשדה.

* מסך של ההיסטורית החיפוש שמציג את 5 השאילתות האחרונות ואת ה-10 תוצאות עבור כל שאילתה. על מנת לראות תוצאות של חיפוש ישן יותר (כגון Query No.2) לחץ על השורה והתוצאות יוצגו.
* צ'טבוט עם שימוש ב- GEN AI: על מנת לבצע לתקשר עם בצ'טבוט הכנס טקסט לתיבת ה-Message ולחץ על כפתור ה-Send, הצ'טבוט יציג את התשובה על המסך.
* מסך של מפת חום המציגה את כמות הקישורים המשותפים בין מילים באינדקס.
  + טעות אפשרית: אם יש יותר מדי מילים באינדקס, הטבלה יכולה להיראות עמוסה, על מנת לפתור זאת אנו מציגים רק אינדקסים שיש להם כמות מינימלית משותפת עם אינדקס אחר (רק אז המילה תקבל שורה\עמודה).
* מסך של תרשים עמודות המציג את כמות המופעים של המילים באינדקס באתר של IBM.
  + טעות אפשרית: אם יש יותר מדי מילים באינדקס, הטבלה יכולה להיראות עמוסה, על מנת לפתור זאת אנו מציגים רק אינדקסים עם כמות מינימלית של מופעים.
* מסך תפריט עריכה לאינדקס המאפשר את הפעולות:
  + הצגת האינדקס.
  + הוספת \ מחיקת מילים באינדקס.
  + הוספת \ הסרת קישור מהאינדקס.
  + שמירה \ טעינה של האינדקס מהבסיס נתונים.
  + ביצוע Crawling מחדש.

טעויות אפשריות:

* מילים שלא מופיעות באינדקס שהמשתמש יחפש בשאילתא במנוע חיפוש לא ימצאו בעמודים למרות שיש עמודים שמופיע בהם מילים אלו. על מנת שיופיעו, על המשתמש להיכנס לתפריט אינדקס, להוסיף את המילים לאינדקס ולהפעיל את הזחלן כדי לחפש את המופעים של מילים אלו באתר.
  + בחרנו שלא לבצע את ה-crawling באופן אוטומטי לאחר הוספת מילה חדשה על מנת לא "לנעול" את המשתמש במצב crawling אם הוא ירצה להוסיף עוד מילים.
* עקביות בגודל של המסכים בגלל השימוש בחבילת pyplot - מסכים גדולים מזיזים את המבט אל אמצע הגרף ומשנים את המיקום של הדף בלי החלטת המשתמש לזוז לאמצע הגרף.

הסבר יותר מפורט על המסכים ניתן למצוא בקובץ UserKit.docx המצורף.

סעיף 7 – שקיפות אלגוריתמית

שקיפות אלגוריתמית במסך החיפוש: אנו מציגים למשתמש את הדרך בה מתבצע חישוב ה-rank עבור כל קישור, כמות הקישורים שעונים על השאילתה של המשתמש, ומשך זמן החיפוש. בנוסף רשום עבור כל קישור בתוצאת החיפוש את "אחוז ההתאמה" שלו לשאילתה (לדוגמא match 98.18%) אחוז זה הוא "נרמול" של הדירוג לצורה ברורה יותר עבור המשתמש.

שקיפות אלגוריתמית במסך ההיסטוריה: אנו מציגים למשתמש את השאילתות האחרונות שלו בצורה ממוספרת על מנת שידע מהי השאילתה האחרונה שעשה.

סעיף 8 – אתגרים במהלך העבודה

בניית האינדקס: בהתחלה רצינו שהאינדקס יבנה באופן דינאמי במהלך תהליך ה-crawling, אבל ראינו שהוא בנה אותו עם יותר מדי מילים (למרות שהוצאנו את ה-stopwords) שלא היה ניתן לשמור אותו ב-firebase. אז הוחלט שנגדיר את האינדקס הראשוני עם מונחים מהקורס ומהאתר של IBM ונבנה את האינדקס רק עבור אותן מילים. ובהמשך במידת הצורך ניתן להוסיף מילים חדשות על ידי תפריט האינדקס.

תהליך ה-Crawling: בהתחלה עשינו crawling על כ-200 עמודים, אבל ראינו שאנו שלא מגיעים לעמודים שציפינו, כגון עמודים שמכילים מידע על מונחים. אז שיפרנו את ה-CrawlerService על מנת שיתעלם מעמודים שאינם רלוונתים לנו, כגון עמודי הורדות, עמודי mp3 וכו'.

קוד מודולרי: לא היה לנו ניסיון רב בפייטון, ולכן בהתחלה רשמנו את כל הפעולות העיקריות בפונצקיות משלהן, מה שגרם לעבודה מבולגנת כאשר פונקציה אחת הייתה תלוייה בשנייה. לאחר מכן הוחלט שנלמד עצמאית את החלק של Class-ים בפייטון ובנינו את השירותים עם כמות תלויות מינימלית (ואיפה שיש אנו משתמשים ב-dependency injection). כתוצאה מכך יצא לנו מבנה יותר מסודר של המערכת, עם שימוש בשירותים, ובלי קוד "גלובלי" בסקריפט אלה רק קריאה לפונצקיית ה-main.

סעיף 9 – משובים

משובים של שבוע 11

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| עשיתם תפריט של CLI, אבל אתם באפליקציה עם frontend, זה לא הגיוני | לא | המערכת איננה frontend "טהור" ולכן בחרנו לעשות עיצוב מינימליסטי, ובפרט עיצוב פשוט עבור תפריט עריכת האינדקס, היות וקבלנו הרבה משובים חיוביים לגבי העיצוב, בהצגה וגם בביקורת עמיתים, בחרנו להשאיר את העיצוב כמו שהוא |
| הכיתוב קטן. הגרפים מאוד קטנים ועמוסים (כדאי להציג גג 10 פריטים בגרף). | כן | הגדלנו את הכתב בכמה מקומות במערכת, ועשינו סינון נוסף עבור הפריטים בגרפים כך שיראו פחות עמוסים |
| אולי להגדיל קצת את הכתב של הsnippet של מה מופיע באתר | כן | הגדלנו את הכתב של ה-snippet-ים |
| הuse case קצת מסורבל | לא | מוצג ב-Use-Case כל הפונקציונליות העיקרית של המערכת, ועל מנת שהוא יכנס בעמוד אחד, עם כתב מספיק גדול. היינו צריכים למקם את כל הבועות עם מעט רווחים. לכן חילקנו אותם לקבוצות עם אותו הקשר, ולכל קבוצה צבע משלה (כגון בועות הקשורות ל-DB הן בצבע כחול) על מנת להקל על ההבנה של התרשים. |
| עיצוב, הסבר מה זה match | כן | הוספנו הסבר נוסף במסך החיפוש, איך הדפים מדורגים |

משובים של שבוע 12

|  |  |  |
| --- | --- | --- |
| **הערה** | **תגובה** | **האם בכוונת הצוות לבצע שינוי בעקבות ההערה? יש לפרט. אם כן, מה השינוי. אם לא, מדוע** |
| פשטות | הקוד מבצע הנדרש כל הפונקציות עובדות כמו שצריך | לא, תודה רבה. אין מה לשנות. |
| מימוש | הקוד מספק את מה שנדרש באופן מהיר וישיר | לא, תודה רבה. אין מה לשנות. |
| מודלריות | הקוד מפושט לתאים שונים וכך שומר על פשטות ובהירות | לא, תודה רבה. אין מה לשנות. |
| יעילות | יש חלוקה טובה וברורה. כל תא בקולאב מכיל קטע קוד שאחראי על פונקציונליות שונה. | לא, תודה רבה. אין מה לשנות. |
| באגים | הכל רץ באופן תקין ומהיר | לא, תודה רבה. אין מה לשנות. |
| טיפול בשגיאות | יש טיפול בשגיאות, בעת חיפוש לא תקין יש הודעה למשתמש שהיא ברורה מאוד. לא מחפש stop words | לא, אין חיפוש של stops-words כי אנו לא רואים צורך לשמור אותם באינדקס. |

סעיף 10 – מקורות ושימוש בכלי AI

מקורות:

* <https://docs.python.org/3/>
* <https://ipywidgets.readthedocs.io/en/stable/>
* <https://www.ibm.com>
* <https://firebase.google.com/docs>
* <https://seaborn.pydata.org/>
* <https://matplotlib.org/stable/>
* מצגות הקורס

שימוש בכלי AI:

* שימוש ב-ChatGPT
  + הסבר איך לבנות מחלקות ב-python

Python: how to create classes, give example of Student class with private and public methods\properties

* + הסבר איך ליישם פונקציית onClick עבור רכיב ב-ipywidgets

Python, ipwidgets: how to add function that will trigger on field enter and button click

* שימוש ב-Genmini
  + הסבר איך לשלב את genai בצ'אטבוט, עם מפתח API ייחודי

Python: how to create a simple google.generativeai chatbot, with unique key