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

**תרגיל בית 3**- **עבודה בצוותי העבודה**

מועד הגשה: 26.1.25

המשימה בתרגיל זה: סיום בניית המערכת, ביצוע קוד ריוויו לצוות אחר

איטרצית פיתוח מס 2

שימו לב: למטלה זו שלושה חלקים

קישור לתיקיית גיט ציבורי:

<https://github.com/SagiYosofov/WOLF.git>

קישור למחברת קולב:

[Colab - Wolf](https://colab.research.google.com/drive/1RLhVI3BrmWjSfd2o-26i6k4pDdsYOZnk#scrollTo=N9iBXjrADxh4)

**חלק ראשון.**עליכם להגדיר את בעלי התפקידים לאיטרציה זו (יש להחליף מתרגיל בית 2) יש לרשום מי מהנדס.ת המערכת, האחראי.ת בתרגיל זה.

|  |  |  |  |
| --- | --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** | בדיקת קבלה (acceptance test) – האם הבדיקה עברה? |
| דניאל פלדמן, מהנדס המערכת scrum master | ווידוי תקינות מסך מנהל ומסך מידע ודירוג וניקוי קוד רלוונטי למסכים אלו  פירוט על שימוש באינדקס סרוויס ורובוטס צ'קר, כתיבת KPI רלוונטיים, כתיבת אתגרים שעלו במהלך העבודה,  עריכת והגשת המטלה. | ווידוי תקינות מסך מנהל ומסך מידע ודירוג וניקוי קוד רלוונטי למסכים אלו  פירוט על שימוש באינדקס סרוויס ורובוטס צ'קר, כתיבת KPI רלוונטיים, כתיבת בדיקות קבלה לכל חלקי המערכת.  עריכת והגשת המטלה. | עברה |
| ארתור צ'רני  QA | ווידוי תקינות מסכים וניקוי קוד  שימוש באינדקס סרביס, כתיבת KPI רלוונטיים,  כתיבת אתגרים שעלו במהלך העבודה, בניית תיק למשתמש, התייחסות לשקיפות אלגוריתמית בתיק משתמש, הכנת סרטון המתאר את השימוש במערכת | ווידוי תקינות מסכים וניקוי קוד, שימוש באינדקס סרביס, כתיבת KPI רלוונטיים,  כתיבת אתגרים שעלו במהלך העבודה, בניית תיק למשתמש, התייחסות לשקיפות אלגוריתמית בתיק משתמש,  כתיבת אתגרים שעלו במהלך העבודה,  הכנת סרטון המתאר את השימוש במערכת | עברה |
| שגיא יוסופוב Frontend + UI | ווידוי תקינות מסכים וניקוי קוד (דף חיפוש)  שימוש בסרביס, מימוש צ'טבוט, כתיבת אתגרים שעלו במהלך העבודה, העלאת המטלה לגיט | ווידוי תקינות מסכים וניקוי קוד (דף חיפוש)  שימוש בסרביס, מימוש צ'טבוט, כתיבת אתגרים שעלו במהלך העבודה, העלאת המטלה לגיט | עברה |
| איליה לזרב Product Manager | ווידוי תקינות מסכים וניקוי קוד (דף צ'טבוט),  שימוש בסרביס,  מימוש צ'טבוט, כתיבת אתגרים שעלו במהלך העבודה, שקיפות אלגוריתמית | ווידוי תקינות מסכים וניקוי קוד (דף צ'טבוט),  שימוש בסרביס,  מימוש צ'טבוט, כתיבת אתגרים שעלו במהלך העבודה, שקיפות אלגוריתמית | עברה |
| אביב רז Backend Developer | ווידוי תקינות מסכים וניקוי קוד (דף סטטיסטיקות), כתיבת KPI רלוונטיים, כתיבת אתגרים שעלו במהלך העבודה, בניית תיק מתכנת | ווידוי תקינות מסכים וניקוי קוד (דף סטטיסטיקות), כתיבת KPI רלוונטיים, כתיבת אתגרים שעלו במהלך העבודה, בניית תיק מתכנת | עברה |

**בדיקות קבלה**

**דף מנהל**

**דניאל פלדמן**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comment | Precondition | Expected Result | Description | Test ID | # |
| Test to see if the front end reacts well to no index. | No entries in the index folder in the database | System displays a message saying the index is empty | Open manager page | Empty/Non Existing Index | 1 |
| Test to see if words can be added to the database | The word doesn’t exist in the database | System displays a message saying the word was added successfully, the word is added to the database under index folder with the entered url and count.  System clears all fields.  System refreshes the table to show the newly added word. | Open manager page  Type in a word: testword  Select count: 5  Type in a url: https://example.com/  Click “Add Word” button. | Successful “add new word” | 2 |
| Test to see if the system reacts well to invalid input in add new word (count field) | - | System displays a message saying the count is not valid. | Open manager page  Type in a word: testword  Select count: -1 or 0  Type in a url: https://example.com/  Click “Add Word” button. | Invalid add new word count | 3 |
| Test to see if the system reacts well to invalid input in the add new word (Link field) | - | System displays a message saying the url is not valid. | Open manager page  Type in a word: testword  Select count: 5  Type in a url: abc  Click “Add Word” button. | Invalid add new word url | 4 |
| Test to see if links are added to a word successfuly | The word already exists in the index folder in the DB | System displays a message saying the word was updated and refreshes the table, and updates the DB and clears all fields. | Open manager page  Type in a word: testword  Select count: 5  Type in a url: https://example2.com/  Click “Add Word” button. | Successful add new link to word | 5 |
| The word is in the index and has the given url in its list. | The word already exists in the index folder in the DB with the given url in its url list | System displays a message saying the word was updated and refreshes the table, and updates the DB and clears all fields. | Open manager page  Type in a word: testword  Select count: 5  Type in a url: https://example.com/  Click “Add Word” button. | Successful add count to existing word link | 6 |
| Test to see if the system reacts well to fields left empty in add new word | - | System displays a message saying please enter a word | Open manager page  Select count: 5  Type in a url: https://example.com/  Click “Add Word” button. | Empty word in add new word | 7 |
| Test to see if the system reacts well to fields left empty in add new word | - | System displays a message saying please enter a link. | Open manager page  Type in a word: testword  Select count: 5  Click “Add Word” button. | Empty link in add new word | 8 |
| Test to see if editing a word’s frequency in a url works well | There is at least one word in the index to edit inside the DB index folder. | System displays a message saying the word was updated successfully, refreshes the table and updates the database, disables the edit form.  (Both the word’s total frequency and frequency in the url are updated) | Open manager page  Click on “Edit” button next to a link in the table  Select count : 5  Click “Save Edit” button | Successful edit word url | 9 |
| Test to see if the system handles invalid frequency while editing it in a url | There is at least one word in the index to edit inside the DB index folder. | System displays a message saying the count is not valid. | Open manager page  Click on “Edit” button next to a link in the table  Select count : -5  Click “Save Edit” button | Invalid count edit | 10 |
| Try to click on save edit without changing count. | There is at least one word in the index in the DB with at least one url with the count of 5 appearances | System displays a message saying the count is the same | Open manager page  Click on “Edit” button next to a link in the table  Select count : 5  Click “Save Edit” button | Same count edit | 11 |
| Test to see if removing a url from a word’s url list works well when the word has more than 1 url | There is at least one word in the index with more than one url in its list. | System displays a message saying the link was removed successfully from the word | Open manager page.  Click on “Remove” button next to a link in the table. | Successful remove link (more than 1 link) | 12 |
| Testing that removing all links to a word removes the word. | The word testword exists in the index with only 1 url. | System displays a message saying the word was removed successfully from the database and removes it from firebase | Open manager page.  Select word: testword  Click on “Remove” button next to a link in the table. | Successful remove link (1 link left) | 13 |
| Testing that the remove word button actually removes a selected word. | The word testword exists in the index. | System displays a message saying the word was removed successfully.  System removes the word from both the local index and the database | Open manager page.  Select word: testword  Click on “Remove Current Word” button next to a link in the table. | Successful remove current word | 14 |

**דף חיפוש**

**איליה לזרב**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comment | Precondition | Expected Result | Description | Test ID | # |
| Validates search logic, stemming, ranking, and database updates. | Index contains the terms “azur” or “storage” | System displays a list of all relevant URLs with ranked results.  Results are displayed in descending order by rank.  Query “Azure storage” is added to Firebase under “Queries” folder.  Words “Azure” and “storage” are added to Firebase under “Words” folder. | Open Search Page  Enter query: Azure storage  Click “Search” button. | Successful Search | 1 |
| Makes sure we don’t try to process empty strings. | - | System does nothing and doesn’t waste computing power. | Open Search Page  Click “Search” button | Empty Search | 2 |
| Ensures unmatched queries are handled gracefully and no false database entries are created. | "NonexistentTermXYZ" doesn’t exist in the index. | System displays no results found message. | Open Search Page  Enter: “NonExistentTermXYZ”  Click “Search” button. | Query words not in index | 3 |
| Ensures we still get results when there is at least one word in the query that exists in the index. | The word “azur” exists in the index, the word “icecream” doesn’t exist in the index | System displays a list of all relevant URLs with ranked results.  Results are displayed in descending order by rank.  Query “icecream azure” is added to Firebase under “Queries” folder.  Words “icecream” and “azure” are added to Firebase under “Words” folder. | Open Search Page  Enter: “icecream azure”  Click on “Search” button | One word in index one isn’t | 4 |
| Confirms safe failure when the index is missing or empty. | Index is empty or doesn’t exist in the DB. | System displays an error message saying the index is empty or doesn’t exist | Open Search Page | Empty/Non Existing Index | 5 |
|  |  |  |  |  | 6 |

**דף סטטיסטיקות**

**אביב רז**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comments | Precondition | Expected Result | Description | Test ID | # |
|  |  |  |  | Bar Plot: |  |
| Bar Plot – Top 10 Words (S) | There is at least 1 word in the Words folder in the Database. | System displays a barchart, showing the correct top 10 searched words and their counts. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Searched Words” from the dropdown. | BP Successful display Top 10 Searched Words. | 1 |
| Bar Plot – Top 10 Words (F) | There are no words in the Words folder in the Database | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Searched Words” from the dropdown. | BP No words for Top 10 Searched Words | 2 |
| Bar Plot – Top 10 Queries (S) | There is at least 1 query in the Queries folder in the Database. | System displays a barchart, showing the correct top 10 searched queries and their counts. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Searched Queries” from the dropdown. | BP Successful display Top 10 Searched Queries. | 3 |
| Bar Plot – Top 10 Queries (F) | There are no entries in the Queries folder in the Database | System displays a message saying there are no queries to display yet. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Searched Queries” from the dropdown. | BP No words for Top 10 Searched Queries. | 4 |
| Bar Plot – Top 10 Common Words (S) | There is at least 1 entry in the Index folder in the Database. | System displays a barchart, showing the correct top 10 most common words in Azure and their counts. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | BP Successful display Top 10 Common Words. | 5 |
| Bar Plot – Top 10 Common Words (F) | There are no entries in the Index folder in Firebase | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Bar Plot” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | BP No words for Top 10 Common Words | 6 |
|  |  |  |  | Pie Chart: |  |
| Pie Chart – Top 10 Words (S) | There is at least 1 word in the Words folder in the Database. | System displays a Pie Chart, showing the correct top 10 searched words and their counts. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Searched Words” from the dropdown. | PC Successful display Top 10 Searched Words. | 7 |
| Pie Chart – Top 10 Words (F) | There are no words in the Words folder in the Database | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Searched Words” from the dropdown. | PC No words for Top 10 Searched Words | 8 |
| Pie Chart – Top 10 Queries (S) | There is at least 1 query in the Queries folder in the Database. | System displays a Pie Chart, showing the correct top 10 searched queries and their counts. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Searched Queries” from the dropdown. | PC Successful display Top 10 Searched Queries. | 9 |
| Pie Chart – Top 10 Queries (F) | There are no entries in the Queries folder in the Database | System displays a message saying there are no queries to display yet. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Searched Queries” from the dropdown. | PC No words for Top 10 Searched Queries. | 10 |
| Pie Chart – Top 10 Common Words (S) | There is at least 1 entry in the Index folder in the Database. | System displays a Pie Chart, showing the correct top 10 most common words in Azure and their counts. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | PC Successful display Top 10 Common Words. | 11 |
| Pie Chart – Top 10 Common Words (F) | There are no entries in the Index folder in Firebase | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Pie Chart” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | PC No words for Top 10 Common Words | 12 |
|  |  |  |  | Histogram |  |
| Histogram – Top 10 Words (S) | There is at least 1 word in the Words folder in the Database. | System displays a Histogram, showing the correct top 10 searched words and their counts. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Searched Words” from the dropdown. | H Successful display Top 10 Searched Words. | 13 |
| Histogram – Top 10 Words (F) | There are no words in the Words folder in the Database | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Searched Words” from the dropdown. | H No words for Top 10 Searched Words | 14 |
| Histogram – Top 10 Queries (S) | There is at least 1 query in the Queries folder in the Database. | System displays a Histogram, showing the correct top 10 searched queries and their counts. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Searched Queries” from the dropdown. | H Successful display Top 10 Searched Queries. | 15 |
| Histogram – Top 10 Queries (F) | There are no entries in the Queries folder in the Database | System displays a message saying there are no queries to display yet. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Searched Queries” from the dropdown. | H No words for Top 10 Searched Queries. | 16 |
| Histogram – Top 10 Common Words (S) | There is at least 1 entry in the Index folder in the Database. | System displays a Histogram, showing the correct top 10 most common words in Azure and their counts. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | H Successful display Top 10 Common Words. | 17 |
| Histogram – Top 10 Common Words (F) | There are no entries in the Index folder in Firebase | System displays a message saying there are no words to display yet. | Open Statistics Page.  Press “Show Histogram” button.  Select “Top 10 Common Words in Microsoft Azure” from the dropdown. | H No words for Top 10 Common Words | 18 |
|  |  |  |  | Refresh |  |
| Checks if refresh button works correctly. | There is at least 1 entry for the selected diagram and data type in the Database | System refreshes the current displayed plot and shows new data if it was added | Open Statistics Page  Press “Show {Bar Plot / Pie Chart / Histogram}”  Select “Top 10 {Searched Words / Queries / Common Words in Microsoft Azure}”  Press “Refresh” button. | Refresh button success | 19 |

**דף מידע ושקיפות אלגוריתמית**

**ארתור צ'רני**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comments | Precondition | Expected Result | Description | Test ID | # |
|  | - | System displays a thank you message and the chosen scores, updates firebase with the new rating entry and disables the Submit button | Open Information and Rating Page  Select satisfaction rating : 10  Select accuracy rating : 10  Select response time rating : 10  Click “Submit” button | Submit success |  |
|  | No connection to firebase | System displays an error message saying the feedback couldn’t be sent to the database | Open Information and Rating Page  Select satisfaction rating : 10  Select accuracy rating : 10  Select response time rating : 10  Click “Submit” button | Submit Failure |  |

**דף צ'טבוט**

**שגיא יוספוב**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comments | Precondition | Expected Result | Description | Test ID | # |
|  | - | System displays a reply from the gemini API | Open ChatBot page  Enter “Hello”  Click the ‘Send’ button | Send Success |  |
|  | - | System displays a message saying please enter a message. | Open ChatBot page  Click “Send” button | Send Empty |  |

***חלק שני: בניית המערכת*** (80 נקודות)

עליכם לסיים את בניית המערכת.

בפרט, יש לוודא קיום של כלל האלמנטים הבאים:

1. כל הפיצ'רים שהתחלתם לממש בתרגיל בית 2 - יש לוודא שכל המסכים תקינים ועובדים כראוי, **בקולאב**,ולעבוד עם firebase לצורך שמירת נתונים .אין להשאיר קישורים או כפתורים למסכים שלא קיימים.

יש לוודא שניתן להריץ את הפרויקט מהמחברת בלבד , ללא כל צורך בהטענה של קבצים נוספים או יציאה למערכת חיצונית (כגון FLASK). (30 נקודות)

1. שימוש ב -microservices (תרגול 7)- עליכם לעשות שימוש לפחות בשניים. פרטו באילו שירותים עשיתם שימוש, ומה היתרון בשימוש בשירותים אלו.(10 נקודות)
2. צטבוט שישולב במערכת (כפי שלמדתם בתרגול 8), הכולל מידע אשר מגיע מה - DB שלכם, ועונה בצורה אינטליגנטית לשאלות המשתמש. (15 נקודות).
3. מהם ה -KPI הרלוונטיים בפרויקט שלכם? הסבירו? (10 נקודות)
4. שקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם? (5 נקודות)
5. אתגרים שעלו במהלך העבודה , וכיצד התמודדתם איתם (לדוגמא- DB גדול מדי, יצירת ויזואליזציה, מבנה הקוד וכו) (10 נקודות)

***חלק שלישי : סגירת הפרויקט \_( 20 נקודות)***

בחלק זה תכינו מסמכים המתארים את הפרויקט שלכם:

1. יש לבנות תיק למתכנת הכולל את שמות כל הקבצים המרכזיים, פונקציות מרכזיות, קטעי קוד/תבניות עיצוב מעניינים שהשתמשתם בהם.
2. יש לבנות תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.
3. התייחסו בתיק המשתמש לשקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?
4. יש להכין סרטון קצר של 30-60 שניות, המתאר את השימוש במערכת. הסרטון משמש כ – elevator pitch  למערכת שלכם, כלומר יש לכלול בו הסבר מקצועי ועם זאת שיווקי , המדגיש את האלמנטים המיוחדים של המערכת שבניתם. יש להגיש את הסרטון בפורמט mp4.

הוראות הגשה:

1. יש להגיש את התרגיל בצוותים, בתיקיית ה –GIT שלכם, וכן בתיקייית התרגיל ב moodle. **חובה לכלול קישור לתיקיית הגיט בקובץ במוודל.**
2. יש להגיש במוודל קובץ זיפ הכולל קובץ וורד ובו מענה לשאלות, וקישור ל- notebook ובו הקוד שלכם (יש לוודא שהקישור פומבי ונגיש).
3. כותרתו של הקובץ תהיה HW3\_TEAMNAME
4. שימו לב כי כל העבודות חייבות להיות שונות זו מזו.אנו מריצות תוכנה לבדיקת עבודות זהות. עבודות שייראו דומות ייפסלו ויינתן עליהן ציון 0.

בהצלחה!

**חלק שני: בניית המערכת:**

2. פירוט על מיקרו-שירותים:

א. שירות המיקרו הראשון בו השתמשנו הוא שירות האינדקס מתרגול 7, לצורך יצירת האינדקס ורשימת המסמכים. היתרון בשימוש בו היה שלא היה צורך להמציא את הגלגל מחדש ולכתוב קוד שיעבור מילה מילה וימפה את המילים לדפים.

ב. שירות המיקרו השני בו השתמשנו הוא מיקרו שירות בשם RobotsChecker שכתבנו בעצמנו – השירות מקבל base url, בדוגמא שלנו "https://azure.microsoft.com" ומוסיף לו את הסיומת "/robots.txt", מוריד את הקובץ ומפענח אותו בעזרת ספריית urllib.robotparser, ולאחר מכן נותן את האפשרות לבדוק האם ניתן לבצע סקרייפ ללינק מסוים והאם לינק מסוים עדיין שייך לאתר שכתובתו היא הbase url.

היתרון בשימוש בו הוא שכעת יש לנו מיקרו שירות גנרי שנוכל להשתמש בו לכל פרויקט בו מצטרך לבצע scraping לאתר מסוים ללא כל חשש שיחסמו אותנו בגלל ניסיון לגשת לכתובת אסורה לשימוש ושבמקום לחשוב איך להכניס את שורות הקוד הרלוונטיות למחלקה בה השתמשנו להורדת תוכן האתר, כל מה שהיינו צריכים לעשות הוא ליצור מופע של RobotsChecker ולקרוא ל2 מתודות על כל לינק שרצינו לבדוק.

ג. מיקרו שירות שלישי הוא DatabaseService אשר משמש אותנו לתקשורת עם מסד הנתונים בFirebase לצורך העלאה והורדה של נתונים, חשבנו על הרעיון רק לקראת סוף הפרויקט ולכן בחרנו להשתמש בו רק במספר חלקים בפרויקט ולא בכל מחלקה.

היתרון בשימוש בו הוא שניתן לחסוך כתיבה מחדש של פקודות להעלאה והורדה של קבצים, ובעתיד נוכל רק להחליף את הקישור למסד הנתונים למסד נתונים אחר בפיירבייס ולהשתמש בו שוב. בנוסף חוסך בכוח חישובי מכיוון שמוגדר כסינגלתון ולכן בעצם לא ניצור עוד ועוד מופעים של החיבור למסד.

4. הKPI הרלוונטיים בפרויקט שלנו הם:

א. זמן תגובה – לדעתנו תוכנה שמגיבה במהירות היא תוכנה יעילה וטובה למשתמש, ניסינו בכל שלב לבצע כל פעולה בדרך הכי יעילה כדי לחסוך בזמן חישוב ולראות למשתמש נתונים כמעט בזמן אמת.

כדי לעמוד ביעד זה, מדדנו את זמן התגובה של המערכת על ידי שימוש בטיים סטמפים בכל אינטראקציה קריטית, כגון מרגע לחיצת המשתמש על כפתור ועד להצגת התוצאות על המסך. נתוני זמן התגובה הודפסו אך ורק לצרכים טכניים ונגישים אך ורק למפתחים (נמדדו בשלב הפיתוח ולאחר נמחקו מהמערכת בשלב release).



A computer screen shot of a computer code

Description automatically generated

יעדינו הוא פחות מ2 שניות והצלחנו לעשות זאת בשנייה אחת.

ב. שיעור שגיאות – פעלנו קשות כדי לוודא שהמשתמש יקבל את התוצאות המדויקות ביותר והרלוונטיות ביותר לחיפוש שלו (ראו אתגר מס' 2) ולכן שיעור השגיאות באפליקציה הוא מדד להצלחה שלנו.

מכיוון שאין לנו הכלים לנתח את תוכן הדף, בחרנו למדוד את שיעור השגיאות בעזרת דירוג של המשתמש, בלשונית מידע ודירוג, נתנו את האופציה למשתמש לשלוח לנו דירוג מספרי של עד כמה תוצאות החיפוש תאמו לשאילתה שלו, 10 הוא 100% ו1 הם 0%.

שיעור השגיאות הכללי הוא הממוצע של כל הדירוגים של הלקוחות.

היעד שהצבנו לעצמנו הוא 80% ומעלה.

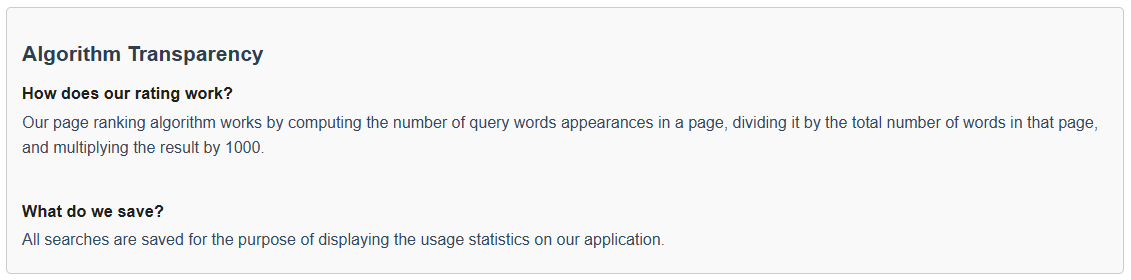
ג. שביעות רצון הלקוח - שביעות רצון הלקוח עמדה במרכז העשייה שלנו. חוויית המשתמש הייתה לנגד עינינו מהרגע הראשון, והנחתה אותנו בכל שלבי הפיתוח והעיצוב. עבורנו, אם המשתמש לא נהנה מהשימוש באפליקציה זה סימן שלא עמדנו בסטנדרטים שהצבנו לעצמנו. רצון זה שלנו בא ליידי ביטוי מעיצוב המסכים שהחלטנו לעשות בצורה מינימליסטית שתחסוך מן המשתמש עומס קוגניטיבי לא נחוץ, פשטות השימוש כמו שבמסך החיפוש יש רק תיבת טקסט וכפתור אחד, ועד הרספונסיביות של עדכון מיידי של טבלת הלינקים במסך המנהל לאחר כל שינוי.

את שביעות רצון הלקוח בחרנו למדוד באותו אופן בו מדדנו את שיעור השגיאות. ע"י דירוג המשתמש.

יעדינו הוא שהממוצע יהיה 8 ומעלה.

5. שקיפות אלגוריתמית:

בחרנו שלהציג למשתמש טקסט קצר ובו הסבר על איך נעשה דירוג הדפים שלנו כדי לספק לו תוכן רלוונטי לחיפוש ואת כך שהחיפושים נשמרים לצורך סטטיסטיקות.



6. אתגרים:

האתגר הראשון שהתמודדנו איתו היו במהלך בניית האינדקס, הוא שבאתר MS Azure היו הרבה מאוד מילים והאינדקס היה ממש גדול, לא יכולנו להעלות אותו בבת אחת למסד, לכן תחילה פירקנו אותו למספר חתיכות וניסנו לצמצם את המספר החתיכות כמה שאפשר כדי לחסוך בגישות למסד, לאחר מכן בצד החיפוש הצוות העדיף להשתמש באופן שונה במסד, ולשמור כל מילה כרשומה, לכן החלטנו שלעבור על כל מילה באינדקס ולהעלות אותה עם רשימת הלינקים הממוינת שלה לפי מספר הופעות.

אתגר נוסף שהתמודדנו איתו היה שבאינדקס היה חוסר דיוק בין כמות ההופעות לכמות הפעמים שניתן למצוא את המילה בדף, איתרנו את מקור הבעיה בכך שבעצם היו כפילויות של לינקים כאשר אחד הסתיים עם סיומת "/" והאחר ללא, ומבחינת הקוד היו אלו אותם לינקים שגרמו לעדכון של כמות ההופעות לכל לינק פעמיים, כלומר אם מילה מסוימת הופיעה בדף מסוים 200 פעמים, קיבלנו כי המילה הופיעה 400 פעם בכל אחד מן הדפים (עם "/" ובלי). התמודדנו עם כך שאיתרנו את החלק בקוד בו יצרנו את האינדקס ודאגנו לוודא שלא יהיו הכפילויות האלו.

אתגר שלישי היה שתחילה האינדקס היה מלא בהרבה מאוד מילים חסרות כל ערך עם סימנים מרובים ומספרים, אותן סיננו בעזרת ביטוי רגולרי שייתן לנו מילים רק שהן א"ב ולא סימנים או מספרים שלא מועילים.

**חלק שלישי: סגירת פרויקט:**

**תיק מתכנת:**

**Index Creation**

**Classes**

|  |  |
| --- | --- |
| Explanation | Name |
| Responsible for parsing a given site’s robots.txt file and checking if given links are in the site’s domain and if they’re allowed to scrape from. (Uses the library robotparser from urllib) | RobotsChecker |
| Responsible for fetching Microsoft Azure’s pages content, stemming it and cleaning it of meaningless words like stopwords and numbers. | AzureSearchEngine |
| In charge of creating the inverted index, iterates through the pages content, for each word saves all the links it appears in and the amount of times it appears in them. | IndexService |
| Responsible for communication with Firebase database, uploading and downloading data. | DatabaseService |

**Methods**

|  |  |
| --- | --- |
| Explanation | Name |
| Fetches the robots txt and parses it | load\_robots\_txt |
| Checks if a given link is allowed to be scraped | Is\_allowed |
| Checks if a given link is in microsoft azure’s domain | is\_in\_site |
| Stems a given word using PorterStemmer | stem\_word |
| Transverses all the links in the site, scraping processing and saving the content for future use. | fetch\_azure\_pages |
| Receives a string and clears it from all substrings that are not alphabet letters, turns it to lowercase and removes stopwords. | clean\_text |
| Recreates the index by fetching the pages and adding all the documents to the index. | restore\_index |
| Recieves a document’s data, creates a unique id for it, runs through all of it’s words, checking the count, updating it’s frequency and total count. | Add\_document |
| Uploads a given user ratings dictionary | Upload\_rating |
| Sorts the index by sorting each word’s url list. | Sort\_index |
| Uploads all urls and their total amount of words as separate entries in the database. | Upload\_doc\_word\_count |
| Uploads the index in a loop, so that each word is an entry | Upload\_index |
| Gets and returns the index from the database | Download\_index |

**Manager Page**

**Functions**

|  |  |
| --- | --- |
| Explanation | Name |
| Receives a message and it’s type (both as strings) and displays it with the right color accordingly on the message output. | display\_message |
| Renders and re-renders the page’s html code, rebuilding the table and displaying all buttons. | refresh\_html |
| Removes a given link locally and, refreshes the page and then saves the changes to firebase.  In case there are no longer links in the selected word, removes the word completely. | remove\_link |
| Receives a term, link and the terms appearance count in that link and activates all buttons related to editing the frequency, basically allowing the user to edit the term’s frequency in that link | edit\_link |
| Removes a word entirely from the local index and from firebase, then refreshes the page. | Remove\_word |
| Updates the term’s frequency in the edited link, updating the total count aswell, refreshing the page and saving to firebase. | on\_save\_edit\_btn\_click |
| Uploads the selected term and it’s doc list to firebase, updating the existing entry or uploading it as new. | save\_to\_firebase |
| Activated when the user selects a word in the combo box, refreshes the page with the newly selected word’s data | on\_word\_select |
| Adds a word to the local index with the values given in the add word text boxes, handles all types of situations such as the case in which the word is already in the index, only updating it’s url list and count, saves the new word to firebase, adds it to the dropbox and refreshes the page with the new word’s data. | Add\_word |

דף זה הוא דף עריכת האינדקס שלנו, בו ניתן לבחור מילה, לבצע שינויים כמו למחוק או להוסיף לינק לאותה מילה או לשנות בלינק ספציפי את כמות ההופעות של המילה, להוסיף מילה, למחוק מילה.

**Information and Rating Page**

**Functions**

|  |  |
| --- | --- |
| Name | Explanation |
| on\_submit\_clicked | Activated when the submit button is clicked, uploads the selected values for user reviews to the database using the database service. |

דף זה משמש למתן שקיפות אלגוריתמית וקבלת משוב מהמשתמש, הוא מכיל קטע HTML בו מפורט ההסבר לצורת דירוג הלינקים שלנו, שלושה סליידרים לדירוג חוויית המשתמש, זמן התגובה ודיוק החיפוש, וכפתור שליחה.

**Interesting Design Patterns**

תבניות עיצוב מעניינות:

Singleton: המחלקה DatabaseService היא singleton, מכיוון שנרצה לשמור רק מופע אחד של חיבור למסד בכל התוכנית במקום לבנות הרבה כאלה ממקומות שונים. (על רעיון זה חשבנו רק לקראת הסוף ולכן השתמשנו בכך רק במקומות האחרונים בהם ביצענו שינויים בקוד).

Facade: תבנית עיצוב זו היא תבנית בה למעשה אנחנו נותנים ממשק פשוט יותר למערכת מסובכת יותר, את התבנית הזו ניתן למצוא גם כן במחלקה DatabaseService, אשר מפשטת שימושים במחלקה IndexService והעלאת נתונים למסד. אז למעשה חלקים אחרים בקוד לא צריכים לדאוג לגבי אינטקרציה ישירה עם IndexService אלא רק להשתמש במתודות הפשוטות שמציע DatabaseService.

**תיק למתכנת  
Search Part  
functions and methods**

|  |  |
| --- | --- |
| **Method/Component** | **Explanation** |
| ***global\_url\_count\_map*** | A global dictionary that stores the total number of words for each URL. It is populated by **URL\_Count\_mapping()** and used to calculate relevance scores for search results. |
| ***URL\_Count\_mapping()*** | Fetches the total word count for each URL from the Firebase database under the /Links path. It saves the overall amount of words from each web page into global\_url\_count\_map for ranking search results. |
| ***add\_single\_word(word)*** | Tracks how often users search for a specific word by updating its count in the Firebase database under the /Words path. If the word already exists, it increments the count; if not, it creates a new entry with a count of 1 (indicating the first search). This data is later used for search statistics and analytics. |
| ***analyze\_query(text)*** | Splits the user's search query into individual words and processes each word using add\_single\_word(). This ensures that every word in the query is counted in the Firebase database, tracking how often users search for specific terms. |
| ***update\_query\_count\_in\_db(query)*** | Tracks how often a specific query is searched by updating its count in the Firebase database under the /Queries path. If the query is new, it initializes the count to 1; if it already exists, it increments the count. This data is used for search statistics and analytics. |
| ***stem\_query\_words(query)*** | Processes a query by reducing each word to its root form (stemming) using the PorterStemmer. This ensures that variations of the same word (e.g., "running" and "run") are treated as identical for searching. |
| ***search(stemmed\_query\_words, index)*** | Checks if each stemmed word from the query exists in the search index. If a word is found in the index, it retrieves all associated data (e.g., links, word counts within those links) and adds it to the results. This method returns a dictionary containing the search results for each word. |
| ***get\_intersection\_of\_urls(query\_data)*** | Finds URLs that are common to **all words** in the query. It iterates through the search results for each word, retrieves the associated URLs from the DocIDs field, and performs a **set intersection** to identify URLs that appear in every word's result. If no common URLs exist, it returns an empty set. |
| ***run\_search\_page()*** | Renders the search interface, including the search input, button, and results container. It also attaches event listeners to handle user interactions, such as button clicks and "Enter" key presses. |
| ***process\_search(query)*** | Orchestrates the entire search process. It takes a user query, stems the words, searches the index, finds common URLs, calculates relevance scores, and displays the results in a table. It also updates query and word statistics in Firebase for analytics. |

**Variables**

|  |  |
| --- | --- |
| **Variable (Grouped by Func name)** | **Explanation** |
| **add\_single\_word(word)** |  |
| word\_path\_in\_db | Firebase path for the word under /Words/. |
| word\_dict | Dictionary retrieved from Firebase containing the word's count. |
| **analyze\_query(text)** |  |
| words | List of words obtained by splitting the query text. |
| **update\_query\_count\_in\_db(query)** |  |
| query\_path\_in\_db | Firebase path for the query under /Queries/. |
| query\_dict | Dictionary retrieved from Firebase containing the query's count. |
| **search(stemmed\_query\_words, index)** |  |
| results | Dictionary storing search results for each stemmed word. |
| word | A single stemmed word from the query. |
| index | Search index containing word-to-URL mappings. |
| **get\_intersection\_of\_urls(query\_data)** |  |
| common\_urls | Set of URLs common to all words in the query. |
| word\_data | Data associated with a word from the query (retrieved from query\_data). |
| urls | Set of URLs associated with a specific word. |
| **run\_search\_page()** |  |
| index | Search index used for retrieving results. |
| js | JavaScript code for attaching event listeners to the search interface. |
| **process\_search(query)** |  |
| **stemmed\_query\_words** | List of stemmed words from the query. |
| **query\_data** | Dictionary of search results for each stemmed word. |
| **common\_urls** | Set of URLs common to all words in the query. |
| **url\_scores** | Dictionary mapping URLs to their relevance scores. |
| **count\_of\_query\_words\_in\_url** | Total count of query words found in a specific URL. |
| **total\_number\_of\_words\_in\_url** | Total word count of a specific URL (from global\_url\_count\_map). |
| **relevance\_score** | A score calculated for a URL based on the ratio of query words found in the URL to the total words in the URL, scaled by 1000 for better readability. |
| **sorted\_urls** | List of URLs sorted by relevance score (descending order). |
| **results** | List of dictionaries containing formatted search results. |
| **results\_html** | HTML string for displaying the search results in a table. |

**ChatBot Part**

**Methods/Functions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Method/Function** |  |  | | --- | |  | | | **Explanation** |  | | --- | --- |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | run\_ChatBot() |  |  | | Sets up and displays the chatbot interface. Integrates a Gemini Generative Model to handle user input and generate bot responses dynamically. |
| |  | | --- | | handle\_message(message) | | Processes user messages. Displays the user input and generates a response  using the Gemini model. Updates the chatbot interface with the bot's response. |
| |  |  | | --- | --- | | on\_send\_clicked(b) |  | | Callback function for the "Send" button. Invokes handle\_message with the user's input and clears the input field after sending. |
| |  |  | | --- | --- | | text\_input.on\_submit() |  | | Sets up an event listener for the Enter key to trigger the same functionality as the "Send" button. |

**Variables**

|  |  |
| --- | --- |
| **Variable** | **Explanation** |
| **genai** | |  | | --- | | The library used to interact with the Gemini generative model. |  |  | | --- | |  | |
| **model** | |  | | --- | | Instance of genai.GenerativeModel configured with the "gemini-1.5-flash" model. |  |  | | --- | |  | |
| **messages\_output** | |  | | --- | | Output widget to display the chat history, including user messages and bot responses. |  |  | | --- | |  | |
| **text\_input** | |  | | --- | | Input widget for users to type their messages. |  |  | | --- | |  | |
| **send\_button** | |  | | --- | | Button widget for sending user messages. |  |  | | --- | |  | |
| **interface** | |  | | --- | | Layout for the chat interface, combining the chat history, input field, and send button. |  |  | | --- | |  | |
| **message** | |  | | --- | | The text input provided by the user. |  |  | | --- | |  | |
| **response** | |  | | --- | | The bot's response generated by the Gemini model based on the user input. |  |  | | --- | |  | |
| **bot\_message** | HTML-formatted string containing the bot's response, used to update the chat interface. |

**Interesting Code Insights and Optimizations**

While there may not be any explicit design patterns visible in the search or chatbot code, one particularly compelling and efficient aspect of the system is the **ranking mechanism**. The ranking system calculates the relevance score of each URL by dividing the number of query words found within it by the total number of words in the URL, then multiplying the result by 1,000. This simple yet effective formula provides a clear and normalized representation of each URL's relevance to the search query, making the ranking both intuitive and easy to interpret.

What truly sets this approach apart is the clever handling of the total word count in each URL. Instead of recalculating the word count on each query—an operation that would be computationally expensive—we take advantage of **precomputation**. When initializing the database, the total word count for each URL is calculated once and stored. The **URL\_Count\_mapping function** then creates a global dictionary (global\_url\_count\_map) that maps each URL to its precomputed word count by fetching this data from the database. This precomputation ensures that the mapping process is extremely fast—taking only seconds—eliminating the need for repetitive calculations each time a query is processed.

This strategy not only optimizes performance but also scales effortlessly as the dataset grows. By leveraging precomputed data, the system remains both fast and accurate, even with large volumes of information. This optimization makes the ranking process significantly more efficient, ensuring that the system delivers high performance without sacrificing the quality or clarity of the results.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

The **URL\_Count\_mapping** function creates a global dictionary mapping each URL to its precomputed word count, fetched from the Firebase database for fast access during queries.

**Relevance Rank Calculation**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

|  |  |
| --- | --- |
| **Method/Component** | **Explanation** |
| **get\_count(item)** | Extracts the 'count' value from a dictionary-like item. Used as a key function for sorting data by count. |
| **real\_time\_data()** | Fetches and processes data from Firebase. Prepares top-10 lists for words, queries, and common words in Azure index. |
| **run\_pie\_chart()** | Displays a bar chart of the selected dataset. |
| **run\_seaborn\_histogram()** | Displays a histogram showing the distribution of counts in the selected dataset. |
| **run\_statistics()** | Main function to handle interactive buttons (Plot, Pie Chart, Histogram, Refresh) and orchestrate the visualization display. |
| **Dropdown** | Provides an interactive way for users to select between data types (Words, Queries, Index). |
| **Buttons** | - Plot: Displays a bar chart.  - Pie Chart: Displays a pie chart.  - Histogram: Displays a histogram.  - Refresh: Updates the current graph with refreshed data. |

**תיק למתכנת  
Statistics Part**

**functions and methods**

**Variables**

|  |  |
| --- | --- |
| Variable | Explanation |
| |  | | --- | | top\_10\_words |  |  | | --- | |  | | List of the top 10 words from the Words node in Firebase, sorted by count. |
| |  | | --- | | counts\_words |  |  | | --- | |  | | |  | | --- | | Corresponding counts for top\_10\_words. |  |  | | --- | |  | |
| top\_10\_queries | |  | | --- | | List of the top 10 queries from the Queries node in Firebase, sorted by count. |  |  | | --- | |  | |
| counts\_queries | |  | | --- | | Corresponding counts for top\_10\_queries. |  |  | | --- | |  | |
| top\_10\_index\_keys | |  | | --- | | List of the top 10 index keys from the Index node in Firebase, sorted by count. |  |  | | --- | |  | |
| counts\_index | Corresponding counts for top\_10\_index\_keys. |

**Interesting Code Insights and Optimizations**

**Dynamic Data Visualization**

The code dynamically updates the data and visualizations in real-time, ensuring the user always sees the most current statistics. This is accomplished using the real\_time\_data() function, which fetches and processes data from Firebase whenever a plot or graph is rendered.

**תיק משתמש**

הסבר כללי על המערכת:

המערכת שלנו היא מנוע חיפוש שמיועד במיוחד לאתר Microsoft Azure. בדומה למנועי חיפוש מוכרים כמו גוגל, היא מאפשרת למשתמשים לחפש מידע שהם צריכים, אבל במקום לחפש בכל האינטרנט, היא מתמקדת רק בתוכן שנמצא באתר הזה. המטרה היא לעזור למשתמשים למצוא מידע במהירות ובקלות, על ידי הכוונה למאמרים ולדפים הכי רלוונטיים באתר לפי מילות החיפוש שלהם.

האפליקציה מבוססת על מאגר מידע שאספנו מכל האתר של Microsoft Azure ומשתמשת באלגוריתם חכם שיעזור למקד את תוצאות החיפוש בצורה מדויקת. בנוסף, שילבנו צ'אט עם בינה מלאכותית (LLM) שמתכתב בשפה פשוטה וברורה.

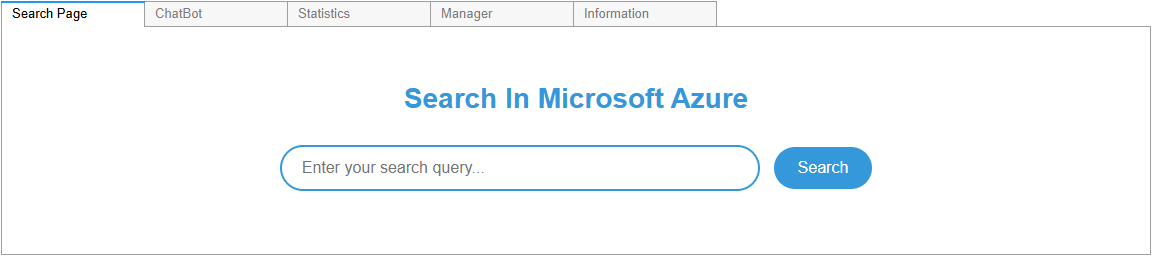
בנוסף, יש גם דף סטטיסטיקה שמציג את מילות החיפוש הכי פופולריות והכי שימושיות, כדי לעזור למשתמשים להבין מה עוד אפשר לחפש ולגלות מידע חדש בקלות.

פירוט מסכים

(פירוט על כל פעולה במסכים נמצא בהמשך)

1. מסך חיפוש מידע

* הזנת מילות חיפוש במנוע החיפוש לקבלת תוצאות ממוקדות מאתר Microsoft Azure
* הצגת תוצאות החיפוש בצורה נוחה ומאורגנת



1. מסך צ'אט בינה מלאכותית

* אפשרות לשוחח עם צ'אטבוט חכם המספק הסברים, המלצות והכוונה
* תמיכה בשפה טבעית לנוחות המשתמש

A screenshot of a computer

Description automatically generated

1. מסך סטטיסטיקות

* בחירה של סוג נתונים להצגה
* צפייה במילות החיפוש הפופולריות והרלוונטיות ביותר בזמן אמת.
* הבנת טרנדים ומגמות עכשוויות בחיפושים.

A screenshot of a graph

Description automatically generated

1. מסך ניהול המערכת

* הוספת מונח חדש
* מחיקת מונח
* מחיקת קישור מהרשימה של המונח
* עריכת כמות מופעים של מונח בקישור
* הוספת קישור למונח עם כמות הופעות מוגדרת

A screenshot of a computer

Description automatically generated

1. מסך אינפורמציה על האפליקציה

* פירוט האלגוריתם
* ממשק שביעות רצון משתמש

A screenshot of a computer

Description automatically generated

הסבר על מעבר בין מסכים:

המעבר בין המסכים מתבצע בעזרת הלשוניות שלמעלה, רק צריך לבחור בלשונית של הדף המתאים וכך נגיע אליו:



טעויות אפשריות:

במסכי חיפוש וצ'טבוט:

חיפוש ארוך ומורכב: הכנסת ביטוי חיפוש ארוך או מסורבל יכולה לבלבל את המנוע ולהחזיר תוצאות שאינן מדויקות.

שימוש במונחים לא רלוונטיים: הכנסת ביטוי שפחות רלוונטי לאתר טכנולוגי כמו "גלידת שוקולד" לא יציג תוצאות.

שימוש בסימני פיסוק: הכנסת ביטוי עם סימני פיסוק תבלבל את המנוע ולא תציג תוצאות.

שימוש בשפה שגויה: המערכת עובדת בשפה האנגלית, כל ניסיון להזין חיפוש שאינו באנגלית לא יציג תוצאות.

לחיצה על כפתור שליחה/חיפוש ללא הזנת טקסט: במקרה זה המערכת תדאג להציג הודעה מתאימה.

במסך מנהל:

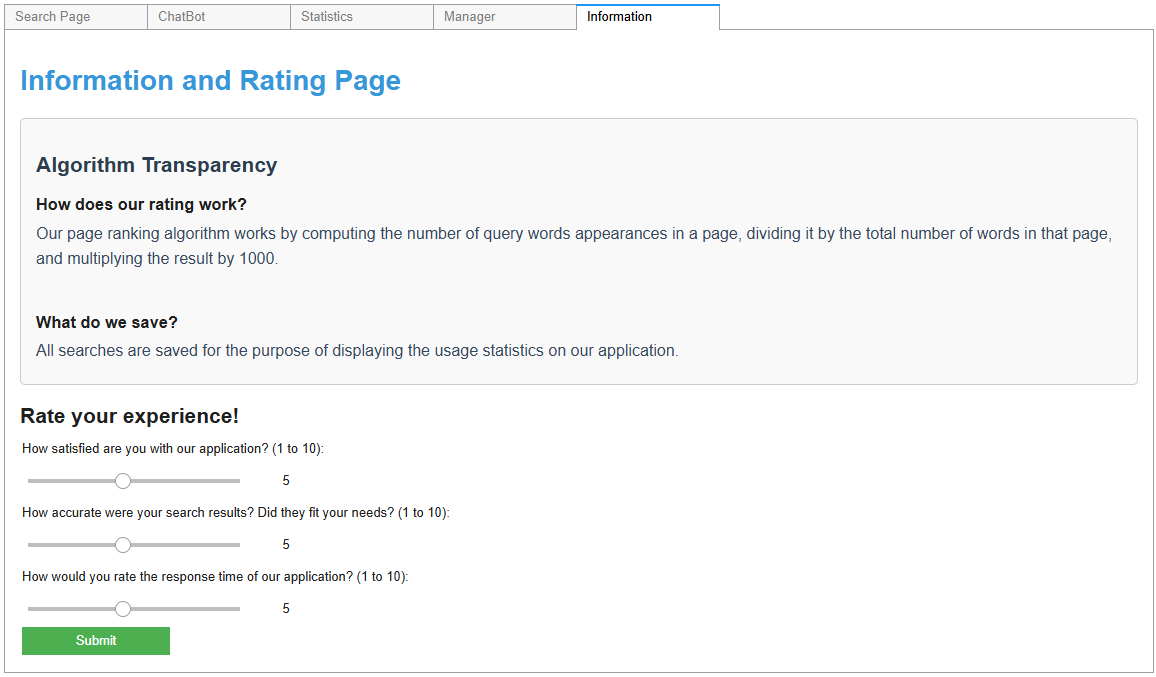
השארת שדות ריקים שנחוצים לפעולות: במידה ונרצה להוסיף מילה ולא נמלא את השדות הרלוונטים, המערכת תציג הודעה מתאימה.

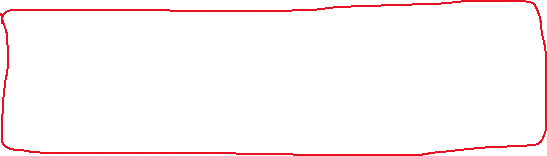
הזנת קלט לא חוקי: במידה ונזין קלט לא חוקי כמו כמות הופעות קטנה או שווה ל0 או לינק מפורמט לא תקין, המערכת תודיע על כך עם הודעה מתאימה.

שקיפות אלגוריתמית:

בלשונית המידע, ניתן לראות את אופן פעולת האלגוריתם, ואת אופן דירוג הדפים.

המידע שנאסף מהמשתמש הוא אנונימי, השאילתות נשמרות במערכת, ונאגרות לצורך יצירה ותחזוק הסטטיסיקה, שירות שחשוב לנו לספק למשתמשים.





גישה לאפליקציה

**הערה:** יש לוודא חיבור יציב לרשת לפני ביצוע השלבים.

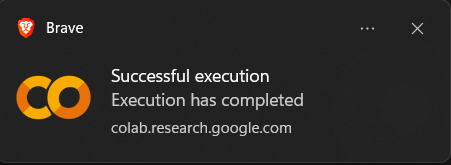
כדי לגשת לאפליקציה, יש לבצע את השלבים הבאים:

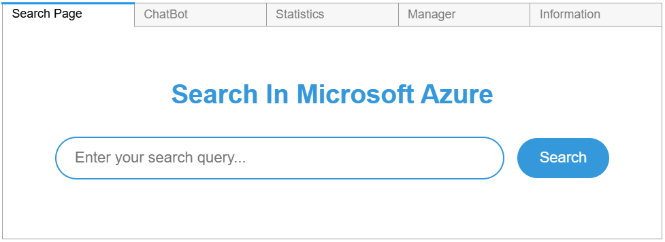
1. נלחץ על **הקישור כניסה**
2. הפעלת המערכת
   1. ישנם שתי אפשרויות
      1. בתפריט העליון נלחץ על לשונית Runtime
      2. נבחר ב Run all   
         A screenshot of a computer program

         Description automatically generated
      3. נמתין לסיום הריצה
   2. אפשרות שניה
      1. נלחץ על הצירוף CTRL+F9



1. נמתין לסיום ההרצה

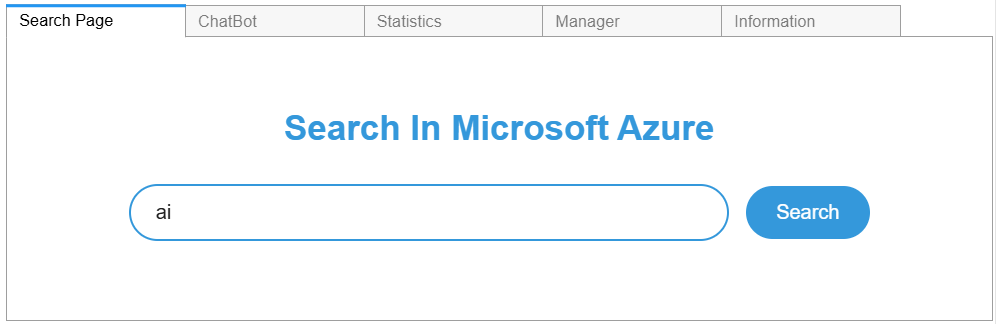


1. נגלול לתחתית עד שנגיע למסך הממשק.  
   

מדריך שימוש

1. חיפוש מידע

נכניס לתיבת הטקסט את המידע שרצוי



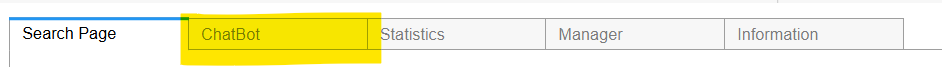
* 1. נלחץ על כפתור Search

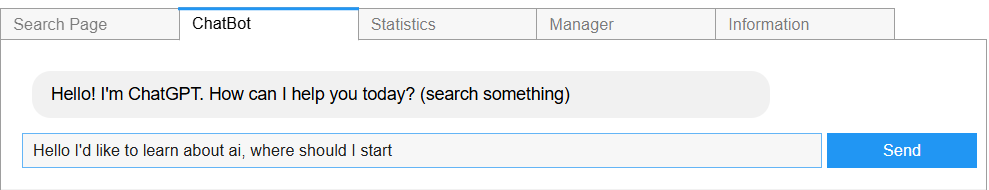
A blue rectangle with white text

Description automatically generated

* 1. נעיין במסך התוצאות   
     A screenshot of a search results

     Description automatically generated  
     נשים לב שהתוצאה הראשונה הינה הרלוונטית ביותר ע"פ האלגוריתם ולכן נפתח את הקישר המוצע
  2. לו הקישור לא סיפק את הצרכים שלנו, נעבור לתוצאה הבאה ונעיין בדף החדש, נחזור על התהליך עד שיתקבל דף שסיפק את הצורך שלנו.

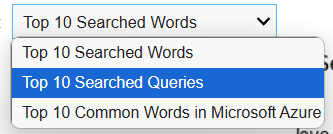
1. שימוש בצ'אט בוט
   1. בתצוגת הלשוניות נלחץ על לשונית ChatBot , המסומנת בצהוב בתמונה  
      
   2. נוודא שהגענו למסך שנראה כמו שיחת צ'אט  
      A screenshot of a computer

      Description automatically generated
   3. נכתוב לבוט הודעה בשפה טבעית כפי שהיינו עושים עם נציג שירות אנושי  
      
   4. נלחץ כפתור send  
      A blue rectangle with white text

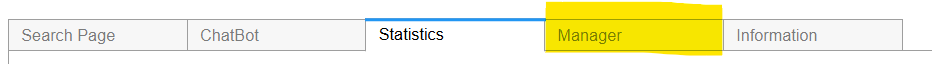
      Description automatically generated
2. שימוש בסטטיסטיקה
   1. נעבור ללשונית הסטטיסטיקה  
      A yellow box with black text

      Description automatically generated
   2. בפנינו שלושה תצוגות של גרפים נבחר אחת באופן שרירותי, ההוראות זהות לשני הגרפים האחרים.  
      A screenshot of a video game

      Description automatically generated
   3. נבחר איזה סטטיסטקיה נרצה לראות באופן הבא, נלחץ על תיבת האפשרויות  
      A yellow rectangular sign with black text

      Description automatically generated
   4. נבחר באפשרות המועדפת  
      
   5. כעת יוטען בפנינו הגרף שרצינו  
      A colorful pie chart with text

      Description automatically generated
   6. **הערה:** חשוב להבהיר שאלו מילים שעברו stemming ולכן עלולות להיראות קטועות, זה נובע מאופן פעילות האלגוריתם, עליו נפרט בפרק שקיפות האלגוריתם.
   7. כדי לקבל את המידע הכי עדכני לזמן זה, נלחץ על כפתור הרענון  
      A blue rectangle with black text

      Description automatically generated
3. שימוש בכלי המנהל
   1. נעבור ללשונית המנהל   
      
   2. **הוספת מילה**
      1. נביט בצד תחתון ימני  
         A screenshot of a computer

         Description automatically generated
      2. נזהה את הטופס הוספה
      3. שדה Word נכניס את המונח שנצרה להוסיף  
         שדה Count נכניס כמות הופעות בדף  
         שדה Link נכניס קישור לדף
      4. נלחץ על כפתור Add Word  
         A screenshot of a computer

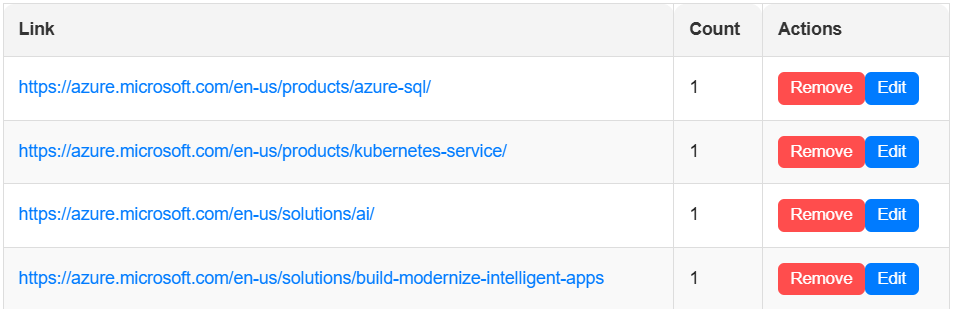
         Description automatically generated
   3. **מחיקת מונח**
      1. נזהה את ממשק בחירת המונחים   
         A red and white box with black text

         Description automatically generated
      2. נלחץ על כפתור בחירת המילה ונבחר במילה הרצויה  
         A screenshot of a computer

         Description automatically generated
      3. נלחץ על הכפתור האדום  
         A red sign with white text

         Description automatically generated
   4. מחיקת קישור מהרשימה של המונח
      1. נזהה את ממשק בחירת המונחים   
         A red and white box with black text

         Description automatically generated
      2. נלחץ על כפתור בחירת המילה ונבחר במילה הרצויה  
         A screenshot of a computer

         Description automatically generated
      3. נזהה בטבלת התוצאות, איזה קישור נרצה למחוק  
         
      4. נלחץ כפתור מחיקה  
         A close up of a computer screen

         Description automatically generated
   5. עריכת כמות הופעות של מילה
      1. נזהה את ממשק בחירת המונחים   
         A red and white box with black text

         Description automatically generated
      2. נלחץ על כפתור בחירת המילה ונבחר במילה הרצויה  
         A screenshot of a computer

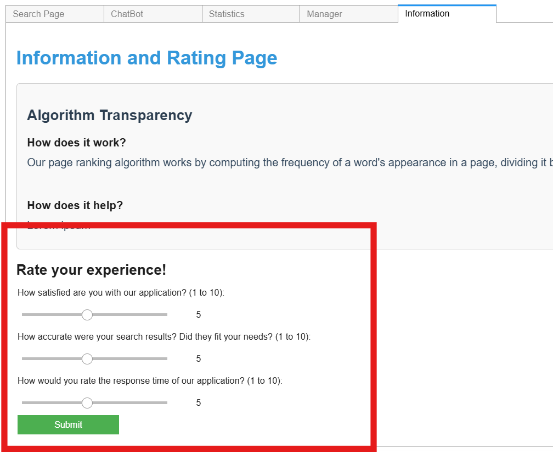
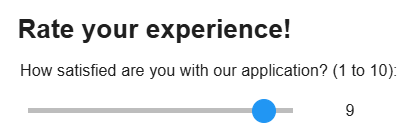
         Description automatically generated
      3. נזהה בטבלת התוצאות, איזה קישור נרצה לערוך  
         A screenshot of a phone

         Description automatically generated
      4. נלחץ כפתור עריכה  
         A close-up of a white background

         Description automatically generated
      5. נזהה את תבנית העריכה   
         A screenshot of a computer

         Description automatically generated
      6. נכניס לשדה השני Count את כמות ההופעות המעודכנת  
         A screenshot of a computer

         Description automatically generated
      7. נלחץ Save Edit  
         A blue rectangle with white text

         Description automatically generated
4. שימוש בלשונית המידע
   1. נלחץ על לשונית information  
      
   2. כעת נוכל
      1. לעיין בהסבר האלגוריתם
      2. לשלוח סקר שביעות רצון
         1. 
         2. נחזיק את העכבר על העיגול במרכז, ונגרור ימינה לניקוד גבוה, ושמאלה לניקוד נמוך. עבור כל שדה  
            
         3. נלחץ על כפתור submit  
            