מסמך מסכם - **Panthers**

מגישים:

דניאל ארמגניאן – 209146943

צחי בקל – 315730176

מתן צוקרמן – 319003935

נדב גולדין – 316350768

אלון טרנרידר – 208635110

שקד עוז – 207475591

אביב בן עזרא – 313596355

**מהות המוצר ומרכיביו**

המוצר שלנו הוא מנוע חיפוש עבור האתר RedHat.

מטרתו היא לייעל ולהקל על חיפוש של מגוון נושאים באתר, ובפרט עבור נושאים הקשורים למחשוב ענן.  
לפרויקט יש מספר מרכיבים אשר מחולקים ל-2:

1. עבור משתמשים

* חיפוש – ביצוע חיפוש בתוך האתר תוך כדי קבלת מידע סטטיסטי על החיפוש.
* ChatBot – שיחה עם ChatBot וקבלת מענה במגוון נושאים מהאתר.

1. עבור מנהל

* סטטיסטיקות – צפייה במגוון סטטיסטיקות.
* עריכת אינדקסים – ניהול האינדקסים במסד הנתונים.

באופן כללי, המערכת מאפשרת למשתמשים להרחיב את הידע שלהם במגוון נושאים שהאתר RedHat מכיל בצורה קלה, יעילה ומהירה דרך החיפוש ודרך השיחה עם הChatBot.

**ארכיטקטורת המערכת**

תמונה שמכילה תרשים, קו, שרטוט טכני, עלילה

התיאור נוצר באופן אוטומטי

המערכת מאופיינת בכמה נקודות מרכזיות:

* מסד הנתונים שהשתמשנו בו הוא firestore (מכיוון שבמסד הקודם לא ניתן לשמור לינק כמפתח)
* המערכת כולה נבנתה בסביבת Google Colab, תוך ממשק עם Git.
* ממשק המשתמש נבנה בעזרת קבצי html, js, css כדי למקסמם את היבט העיצוב.
* בתוך סביבת העבודה, חילקנו כל שירות לחבילת קבצים נפרדת. הקובץ המרכזי של המנוע חיפוש השתמש וסנכרן בין כל החבילות.

שימוש בקבצי web כדי לעצב את ממשק המערכת מאפשר ממשק יותר יפה ומזמין מאשר הממשק שהיינו מקבלים בעיצוב דרך Colab.   
חלוקת המערכת לחבילות וקבצי הפעלה נפרדים שפועלים תחת קובץ ראשי אחד מורידה משמעותית את זמן ההמתנה והופכת את השימוש במערכת לחלק יותר.

בנוסף, המערכת מבצעת clone אוטומטי לריפו המתאים בgit, כלומר, באופן אוטומטי כל הקבצים הנדרשים להפעלת המערכת נמשכים מהתיקייה בgit.

**מאפייני SOA מרכזיים**

1. **Service Autonomy:** המאפיין המרכזי שבא לידי במערכת שלנו.

כל שירות במערכת (בניית אינדקס, chatbot, דף מנהל, GUI) נבנה כיחידה נפרדת, בחבילה נפרדת, עם הקבצים שרלוונטים עבורו. בכל חבילה קיים כל מה שצריך למימוש השירות הספיצפי, כך שכדי לשלב את כל השירותים תחת אותה מערכת יש צורך אך ורק בקובץ ראשי שמסנכרן ביניהם כנדרש.

1. **Interoperability:** מגוון השירותים שהמערכת מציעה עובדים יחדיו בצורה חלקה, למרות שהמימוש שלהם יכול להיות שונה (מבחינת שפת התכנות או היבטי פיתוח אחרים). למשל, פיתוח המערכת נעשה בסביבת Colab, בלומר בשפת python, אך העיצוב נעשה על ידי html, js, css . הסנכרון מתבצע על ידי פקודות ופונקציות מתאימות והתוצאה היא ריצה חלקה של המערכת.
2. **Loose Coupling:** כל שירות במערכת נבנה ביחידה נפרדת ולכן אם יידרש הוא יוכל לפעול גם כאחת כזאת. בעקבות זה, שינויים ובדיקות של חלקים מסוימים במערכת נעשו בצורה נוחה ויעילה יותר.
3. **Reusability**: ניתן להשתמש ביחידות השונות המערכת במקומות הנדרשים, למשל בניית האינדקס משמשת אותנו גם לחיפוש של המשתמש וגם לchatbot.  
   אם בעתיד המערכת תורחב לשירותים נוספים, יהיה ניתן לבצע שימוש חוזר ביחידות הקיימות בה עכשיו.
4. **Scalability**: הרחבת תשתית המערכת כך שתוכל לעמוד בעומסים גדולים יותר תתאפשר בצורה נוחה ויחסית קלה בשל הארכיטקטורה שלה – ניתן להרחיב ולחזק תשתית של כל יחידה (שירות) במערכת, ללא קשר ליחידות אחרות.
5. **Discoverability**: כלל הפיתוח של המערכת מלווה בהסברים והערות בקוד. בנוסף, המערכת מקושרת באופן ישיר לgit והclone מתבצע באופן אוטומטי כדי להבטיח את הנוחות של המשתמש/בודק.

**דרישות לא פונקיוצנליות**

1. **Performance**: המערכת תבצע את כלל פעולותיה בצורה מהירה וחלקה כדי להבטיח חווית משתמש מקסימלית.
2. **Reliability:** המידע שיוצג למשתמש יהיה אמין ומעודכן. נבצע עדכונים רציפים מול האתר RedHat כדי להבטיח אמינות.
3. **Maintainability**: המערכת תיבנה בצורה חכמה באופן שבו ביצוע תחזוקה שוטפת ועדכונים יתבצעו בצורה נוחה וקלה.
4. **Scalability**: המערכת תאפשר עמידה בעומסים של כמויות משתמשים ומידע, ובאם יידרש תאפשר הרחבת התשתית כדי לעמוס ביותר עומסים.
5. **Accessibility:** המערכת תהיה זמינה תמיד, למעט בזמני עדכונים ותחזוקה שיוגדרו מראש.

**תיק תחזוקה**

**קבצים מרכזיים:**

* index.ipynb אחראי על יצירת האינדקס וניהול האינדקס מול המסד נתונים.
* chatbot.ipynb אחראי על מימוש הצ'אטבוט.
* administration.ipynb אחראי על מימוש פעולות של מסך האדמין ובנוסף לכך גם על הפעולות של המסך סטטיסטיקה.
* searchEngine.ipynb אחראי על מימוש פונקציונאליות של מנוע החיפוש ובנוסף לכך האינטגרציה של כל שאר הקבצים בשביל ריצה חלקה.
* קבצי html, css, js עבור ממשק משתמש נוח.

**פונקציות בקובץ searchEngine:**

* **switchToScreen(css\_filename\_, html\_filename\_, js\_filename\_)**
* הפונקציה מקבלת קבצי html, css, js או רק קובץ HTML.
* הפונקציה אחראית על ניקוי המסך והצגת הקובץ שהתקבל על ידי פונקציות מתאימות.
* **get\_title\_from\_link(link)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה מחזירה מחרוזת (כותרת h1 או h2 מתוך הקישור).
* הפונקציה אחראית על מציאת הכותרת של העמוד שאותו הלינק מייצג, בשביל להציג זאת במסך החיפוש.
* **check\_title\_list(lst)**
* הפונקציה מקבלת קבצי רשימה של מחרוזות.
* הפונקציה מחזירה רשימה.
* הפונקציה אחראית על בדיקת כפילויות של כותרות, ומשנה את הכותרות במידת הצורך.
* **receive\_search\_string(search\_string)**
* הפונקציה מקבלת מחרוזת חיפוש.
* הפונקציה מעבדת את מחרוזת החיפוש שהתקבלה לפי הקישורים והכותרות שהוחזרו מהפוקנציה get\_title\_links\_of\_search ומציגה תוצאות חיפוש ב-GUI.
* **back\_to\_home\_page()**
* הפונקציה אחראית על החלפת המסך הנוכחי חזרה לעמוד הראשי של המנוע חיפוש.
* **open\_chatbot\_page()**
* הפונקציה אחראית על החלפת המסך הנוכחי למסך הChatBot.
* **open\_indexeditor\_page()**
* הפונקציה אחראית על החלפת המסך הנוכחי למסך הAdmin.
* **open\_statistics\_screen()**
* הפונקציה אחראית על החלפת המסך הנוכחי למסך הסטטיסטיקות.
* **receive\_question\_chatbot(question)**
* הפונקציה מקבלת מחרוזת שאלה לChatBot.
* הפונקציה אחראית על החזרת מענה למשתמש מהChatBot.

**פונקציות בקובץ Index:**

* **get\_page\_text(url)**
* הפונקציה מקבלת קבצי מחרוזת - לינק.
* הפונקציה אחראית על השגת והחזרת הטקסט מעמוד מסויים תוך כדי תהליך יצירת האינדקס, על מנת לבדוק מופע של מילים באותו העמוד.
* **get\_links(url)**
* הפונקציה מקבלת קבצי מחרוזת - לינק.
* הפונקציה אחראית על השגת כל הקישורים בתוך העמוד שאותו הלינק שהפונקצייה קיבלה, בעזרת השגת כל אלמנטים הhref ומחזירה אותם.
* **trim\_words(words)**
* הפונקציה מקבלת קבצי רשימת מחרוזות.
* הפונקציה מוחקת את המילים הלא נחוצות מתוך רשימת המילים שנמצאה בעמוד, כחלק מתהליך יצירת האינדקס, ומחזירה את הרשימה החדשה.
* **apply\_stemming(words)**
* הפונקציה מקבלת קבצי רשימת מחרווזת.
* הפונקציה אחראית על החזרת כל המילים ברשימה למצבם הבסיסי.
* **create\_index(url)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה משמשת כסקריפט לזחלן, היא מוגבלת בעומק החיפוש שלה. לאחר סיום החיפוש במחרוזת שהתקבלה, מחזירה את האינדקס כמילון שאותו נעבד ונשמור במסד הנתונים.
* **update\_dict(word, link)**
* הפונקציה מקבלת מילון ומחרוזת - לינק.
* הפונקציה מקבלת את המילון שמחזירה create\_index ומחזירה מילון מסודר עם counter לכל מילה וגם id ייחודי לכל קישור.
* **create\_ranked\_words(index)**
* הפונקציה מקבלת את האינדקס כמליון.
* הפונקציה מדרגת את כל המילים באינדקס לפי כמות המופעים של אותה המילה באינדקס.
* **create\_data\_for\_db(final\_index)**
* הפונקציה מקבלת מילון שמכיל את האינדקס הסופי.
* הפונקציה יוצרת רשימת נתונים בפורמט המתאים למסד נתונים של Firestore מהאינדקס הסופי.
* **print\_collection\_from\_db(collection\_name)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה מדפיסה את כל האיברים שיש באינדקס שבמסד הנתונים לפי הסדר שנקבע בפונקציה.
* **upload\_to\_db(data\_list, collection\_name)**
* הפונקציה מקבלת רשימה ומחרוזת.
* הפונקציה מעלה את הרשימה שקיבלה למסד התונים, ספציפית לcollection שהפונקציה קיבלה.
* **get\_index\_from\_db(db\_con = None)**
* הפונקציה מקבלת חיבור אופציונלי למסד נתונים.
* הפונקציה מקבלת מהמסד נתונים את כל האינדקסים, מסדרת אותם לפורמט המתאים ומחזירה את האינדקס הסופי בפורמט מילון.

**פונקציות בקובץ chatbot:**

* **getLinkFromString(w, linkID)**
* הפונקציה מקבלת 2 מחרוזות.
* הפונקציה מחזירה קישור עם המזהה הייחודי linkID שבתוכו יש את המילה.
* **get\_first\_paragraph\_from\_link(url)**
* הפונקציה מקבלת קישור כמחרוזת.
* הפונקציה מחזירה מחרוזת טקסט.
* הפונקציה מחפשת את התשובה ב3 מקומות (h2, div, div) שונים בעמוד שבהם הסבירות לקבל את התשובה היא הגבוהה ביותר.
* **patterns\_to\_dict(patterns)**
* הפונקציה מקבלת רשימת דפוסים מוכרים בChatBot.
* הפונקציה מחזירה מילון מותאם לפורמט הFirestore.
* הפונקציה מסדרת את רשימת הדפוסים בהתאם לפורמט שמקובל בFirestore.
* **create\_patterns ()**
* הפונקציה יוצרת מילון שיכיל את כל התגובות של הChatBot בהתאם לכל מילה שיש בIndex.
* **get\_patterns\_from\_db(db\_collection\_name)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה מחזירה רשימה.
* הפונקציה מחזירה מהמסד נתונים את התגובות של הChatBot, ומפרמטת אותם בחזרה לרשימה שתתמוך בביטויים רגולריים.
* **delete\_response\_from\_db(bot\_response)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה מוחקת מהמסד נתונים תגובה של הChatBot בהינתן המחרוזת שהפונקציה מקבלת (פונקציה זו לא בשימוש – נוצרה למקרה ויש תגובה לא מתאימה).

**פונקציות בקובץ administration:**

* **update\_or\_add\_stsatistics(statistic\_name)**
* הפונקציה מקבלת מחרוזת.
* הפונקציה משמשת כעדכון מופע של מילה אשר חיפשו במנוע חיפוש.
* **get\_indexes\_keys()**
* הפונקציה מקבלת את כל המילים שיש בIndex וממלאה את הCombo-box במסך סטטיסטיקות עם המילים.
* **get\_indexes\_values()**
* הפונקציה מקבלת את כל המילים שיש בIndex וממלאה את הCombo-box במסך סטטיסטיקות עם הקישורים של אותה המילה.
* **get\_random\_indexes\_with\_counters()**
* הפונקציה לוקחת 5 מילים בצורה רנדומלית מתוך הIndex ומציגה אותם במסך הסטטיסטיקות יחד עם כמות המופעים של אותם המילים.
* **get\_key\_with\_min\_counter ()**
* הפונקציה לוקחת מהמסד נתונים את המחרוזת שאותה חיפשו הכי פחות במנוע, ומציגה במסך סטטיסטיקות.
* **get\_key\_with\_max\_counter ()**
* הפונקציה לוקחת מהמסד נתונים את המחרוזת שאותה חיפשו הכי הרבה במנוע, ומציגה במסך סטטיסטיקות.
* **delete\_index\_from\_db(index\_to\_delete)**
* הפונקציה מקבלת את האינדקס שאותו רוצים למחוק מהמסד נתונים.
* פונקציה זו מוחקת ערך במסד הנתונים של Firestore בהתבסס על האינדקס שסופק.
* **delete\_link\_from\_db(link\_index, link\_to\_delete)**
* הפונקציה מקבלת את הקישור שאותו רוצים למחוק מהמסד נתונים ואת האינדקס שהקישור שייך אליו.
* פונקציה זו מוחקת ערך במסד הנתונים של Firestore בהתבסס על הקישור והאינדקס שסופקו.

**פונקציות בקובץ script.js:**

* **PerformSearch()**
* הפונקציה שולחת את המחרוזת שהמשתמש הקליד בחיפוש, לקולאב - למימוש החיפוש.
* **performSearch\_again()**
* הפונקציה שולחת את המחרוזת שהמשתמש הקליד בחיפוש במסך התוצאות, לקולאב - למימוש החיפוש בשנית.
* **backToHomePage()**
* הפונקציה שולחת בקשה לקולאב לחזור למסך הראשי.
* **backToAdmin()**
* הפונקציה שולחת בקשה לקולאב לחזור למסך המנהל.
* **openChatBot ()**
* הפונקציה שולחת בקשה לקולאב לפתוח את מסך הצ'טבוט.
* **openAdminPage ()**
* הפונקציה שולחת בקשה לקולאב לפתוח את מסך המנהל אם הוקלדה סיסמא נכונה
* **askChatbot ()**
* הפונקציה מעדכנת את מסך ההתכתבות עם הבוט ושולחת את מחרוזת המשתמש לקולאב למימוש.
* **ChatbotResponse ()**
* הפונקציה מעדכנת את מסך ההתכתבות בצ'אטבוט בתשובה של הבוט.
* **renderResults\_title\_link(titles, links, times\_appeared)**
* הפונקציה מוסיפה למסך התוצאות את תוצאות החיפוש ומציגה סטטיסטיקות שונות של החיפוש.
* **addSearchEventListener(inputId, func)**
* הפונקציה מבצעת חיפוש בלחיצה על מקש האנטר.

**פונקציות בקובץ admin.html:**

* **deleteIndex()**
* הפונקציה שולחת בקשה לקולאב למחוק את האינדקס הנבחר מהדטבייס ומעדכנת את הGUI.
* **deleteLink()**
* הפונקציה שולחת בקשה לקולאב למחוק את הלינק הנבחר מהדטבייס ומעדכנת את הGUI.
* **getIndexesKeys()**
* הפונקציה שולחת בקשה לקולאב לקבל את הkeys של האינדקסים.
* **openStatisticsScreen()**
* הפונקציה שולחת בקשה לקולאב לפתוח את מסך הסטטיסטיקות.
* **populateFirstCombo(indexes)**
* הפונקציה מייצרת את הcombo-box של בחירת האינדקס לעריכה ומציגה למנהל.
* **getLinks()**
* הפונקציה שולחת בקשה לקולאב ליצירת הcombo-box של הלינקים של האינדקס הנבחר לעריכה.
* **populateSecondCombo(indexes)**
* הפונקציה מייצרת את הcombo-box של הלינקים של האינדקס הנבחר לעריכה ומציגה למנהל.

**פונקציות בקובץ chatbot.html:**

* **addSearchEventListener()**
* הפונקציה מאפשרת שליחת הודעה לבוט על ידי לחיצה על מקש האנטר.

**פונקציות בקובץ statistics.html:**

* **getMinMaxKeys()**
* הפונקציה שולחת בקשה לקולאב לקבל את מילת החיפוש הנפוצה ביותר ואת מילת החיפוש הכי פחות נפוצה לצורך הצגה בסטטיסטיקות.
* **updateStatisticsBlock(min\_key, min\_counter, max\_key, max\_counter)**
* הפונקציה מקבלת את מילת החיפוש הנפוצה ביותר ואת מילת החיפוש הכי פחות נפוצה עם כמות ההופעות בהתאמה ומציגה למסך הסטטיסטיקות.
* **getRandomIndexes()**
* הפונקציה שולחת בקשה לקולאב לקבל אינדקסים רנדומלים עם מספר המופעים של כל אחד מהם בהתאמה.
* **generateGraph()**
* הפונקציה מקבלת אינדקסים ומספר ההופעות שלהם ומציגה למשתמש גרף מתאים.

**ביקורת עמיתים:**

לאחר מעבר על הערות עמיתנו בהצגת הפרויקט ראינו כי עניין עיצוב האתר היה במחלוקת. יש אנשים כי אהבו את העיצוב וחלק אמרו שיש לשפר אותו. לאחר דיון בין חברי הקבוצה החלטנו להשאיר את העיצוב כמו שהוא וזאת משום שלדעתנו העיצוב נעים לעיני המשתמש. בנוסף, העמיתים הציעו להוסיף סטטיסטיקות מעניינות. לפי דרישות הפרויקט, מסך הסטטיסטיקות נתון לשיקולנו. מכיוון שגם קיבלנו משובים טובים על הסטטיסטיקות החלטנו להשאיר שגם את זה נשאיר כמו שהוא.

לסיכום, קיבלנו בסך הכל משובים טובים מעמיתנו בהצגת הפרויקט. ההערות שקיבלנו התחייחסנו בכובד ראש ואחר דיון קבוצתי ראינו לנכון להשאיר את המצב כקדמותו.

**אתגרים**

במהלך בניית המערכת, התמודדנו כקבוצה עם מספר אתגרים.

* קבוצה גדולה של 7 סטודנטים, נדרש מאיתנו להיות יותר מסונכרנים ולחלק את העבודה בצורה מתאימה כך שהעבודה תהיה כמה שיותר יעילה ומסונכרנת.
* העבודה בסביבת Colab לא הייתה מוכרת לאף אחד מאיתנו והיה נדרשת מאיתנו "ללמוד תוך כדי תנועה"
* הבחירה בעיצוב המערכת על ידי קבצי web דרשה מאיתנו למצוא פתרון לסנכרון המסכים והצגתם ללא פגיעה בזמני התגובה.
* מבנה האינדקס שלנו כולל בתוכו קישורים כמפתחות, מה שלא איפשר לנו לעשות עם מסד הנתונים שקיבלנו, ולכן נאלצנו להחליף.
* האתר הראשון שקיבלנו חסם את פעולת הcrawling ולכן נאלצנו להחליף כמה אתרים עד שמצאנו אתר שמאפשר זאת.
* סנכרון העבודה בין כולם ורצון שהמערכת תהיה פתוחה לשימוש מבלי תלות בשיתופים בdrive, הובילה אותנו לממשק את הgit עם הColab.

**שקיפות אלגוריתמית**

המטרה בשקיפות אלגוריתמית היא לקשר כמה שיותר בין המשתמש לבין האלגוריתמים שפועלים במערכת, יישמו זאת בכמה דרכים:

* תיעוד – כלל הקוד מתועד בצורה שמאפשר להבין כיצד המערכת עובדת ומה תפקידו של כל חלק.
* נתונים – הסבר קצר על הנתונים שנאספו במערכת ועל בסיסן היא פועלת, כמו כן גם הסבר על האתר שעליו המערכת מבצעת את השירותים.
* עיצוב – ממשק המשתמש הוא נוח וברור להבנה כך שגם אם משתמש ינסה להשתמש באתר ללא שום הקדמה, הוא יצליח.
* הסברים במהלך השימוש – תוך כדי השימוש בשירותי המערך יינתנו למשתמש הסברים קצרים (משפט) על מה שהוא רואה.
* סרטון המחשה – סיפקנו סרטון קצר שממחיש כיצד משתמשים בשירותים השונים במערכת.

**קישורים ומקורות**

קישור למחברת:

* <https://colab.research.google.com/github/ndvp39/CloudComputing-tirgul/blob/main/Project/searchEngine.ipynb>

קישור לGitHub:

* <https://github.com/ndvp39/CloudComputing-tirgul>

מקורות וקישורים אחרים:

* <https://www.redhat.com/en>
* <https://en.m.wikipedia.org/wiki/Non-functional_requirement>
* <https://firebase.google.com/>
* <https://colab.google/>
* <https://canvasjs.com/docs/charts/basics-of-creating-html5-chart/>
* <https://www.michaelagreiler.com/code-review-checklist-2>
* <https://medium.com/@anshulgarwal45/case-study-on-netflixs-successful-cloud-migration-4cde43b4ed63>
* <https://muditmathur121.medium.com/aws-netflix-case-study-40eea075c3f9>
* <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>
* מצגות הקורס

חלק ב:

חלק קבוצתי:

|  |  |
| --- | --- |
|  | acceptance test |
| Passed – connecting, updating, and retrieving data from FireBase. | Ensure that the database connection is established successfully and that data retrieval and storage operations function as expected |
| Passed – searching queries and getting relevant results. | The application correctly retrieves and displays relevant search results matching the query |
| Passed – all buttons are clickable | Check the functionality of the screen features |
| Passed – all the actions lead to the expected (correct) information. | Verify that the information presented is indeed accurate |
| Passed – the search results are sorted based on their category. | Verify that the search functionality now includes a scoring mechanism that accurately ranks search results |
| Passed – the chat results and links are relevant for the user questions. | Confirm that the project's chatbot functions effectively by utilizing relevant and significant words extracted directly from the website |
| Passed – after checking with the team, we got their requirements, and all are matched. | Verifying that the implemented feature accurately matches the requirements specified in the project |

חלק אישי:

שם הסטודנט: דניאל ארמגניאן

|  |  |
| --- | --- |
| קריטריון | הסבר |
| מימוש | הקוד מבצע את הנדרש. הצ'אטבוט מגיב ומתפקד, החיפוש עובד וניתן לערוך את הIndex ולצפות בסטטיסטיקות. |
| יעילות | היה ניתן לפרק את הקוד ליותר קבצים ולשבור תלויות בין מרכיבים, עשוי לשפר את היעילות. |
| פשטות | חלוקה לקבצים מתאים היה מפשט את הקוד. |
| מודולריות | יש לא מעט פונקציות, אך שוב - חסרה חלוקה לקבצים. |
| תבניות עיצוב | אין תיעוד בקוד על תבניות עיצוב, לא ראיתי שיש. |
| OOP | ניתן להשתמש בהורשה. ישנם מספר עקרונות של OOP בשימוש. |
| באגים וטעויות | לא בטעויות בקוד או באגים באפליקציה. |
| טיפול בשגיאות | אין טיפול נרחב בשגיאות. ישנם מספר מקומות בקוד שיש תפיסה של השגיאה אך לא בצורה נרחבת (try-expect). |
| בדיקות | ניתן לבצע בדיקות תחת עומס על המנוע חיפוש ולראות אם מתפקד בצורה הצפויה. |
| שימושיות | הממשק נוח לשימוש והחוויה כולה אינטואיטיבית. |
| תיעוד | הקוד מתועד בצורה מספקת. |
| אתיקה | הקוד אתי. אין פגיעה בפרטיות וגם אין ניצול התנהגות או חולשה אנושית. |
| אבטחה | הקוד מאובטח. אין פרטים אישיים גלויים. |
| ביצועים | מאחר וכל הפרויקט רשום תחת קובץ יחיד ולא מספיק מודלרי, שינויים מסויימים יכולים לפגוע בביצועים. |
| ביצועים | המערכת מתפקדת בצורה טובה ומגיבה מהר. תמיד ניתן לשפר אך לדעתי המימוש מספק. |
| קריאות - readability | בחלק בו הקוד כתוב בפיתון הקוד מובן ומתועד. בחלקים של js היה ניתן לתעד יותר. בנוסף כפי שנאמר מספר רב פעמים במשוב, חלוקה לקבצים הייתה עוזרת גם לקריאות. |
| chatbot | הצ'אטבוט ממומש בצורה מעולה. העיצוב נוח ואינטואיטיבי. לדעתי מהמרכיבים המעניינים במערכת. |
| Admin page | במסך זה ניתן לערוך את האינדקסים ולראות את כל הקישורים פר מילה, כפי שהתבקש ובצורה מובנת לאדמין. |

1. המעבר על הקוד של קבוצה זו נתן פרספקטיבה שונה למימוש של מרכיבים שונים בפרויקט זה וצורת מחשבה חדשה ומעניינת. בחלקים מסוימים אהבתי את הרעיונות של הקבוצה.
2. לדעתי מבחינת הפונקציונליות הקוד ממומש בצורה מספקת. כפי שציינתי בטבלה, בעיקר היה כדאי לעמוד יותר בקונבנציות: חלוקה לקבצים מתאימים, יותר תיעוד.

שם הסטודנט: צחי בקל:

|  |  |
| --- | --- |
| קריטריון | הסבר |
| מימוש | הקוד מבצע את הנדרש. לאחר בדיקה מעמיקה החיפוש, הצ'אט בוט ומסך עריכת אינדקס וסטטיסטיקות קיימים ועובדים כנדרש. |
| יעילות | הקוד כתוב באופן יעיל. |
| פשטות | היה אפשר לכתוב את קבצי הHTML בנפרד לקוד הראשי. |
| מודולריות | הקוד מחולק לפונקציות שניתן להבין בבירור מה תפקידה של כל פונקציה. |
| תבניות עיצוב | לא מצאתי תבניות עיצוב בקוד. |
| OOP | ישנם שימושים שהיה כדאי להשתמש בהם, לדוגמה הורשה. |
| באגים וטעויות | אין מקרים שבהם נתקלתי. |
| טיפול בשגיאות | יש שימוש ב try-catch לצורך תפיסת שגיאות. ההודעות למשתמש ברורות ומובנות מה התוכן שלהן. |
| בדיקות | מחיקת אותו אינדקס משני משתמשים במקביל. |
| שימושיות | הממשק נוח ונעים למשתמש. איזור הצ'אטבוט נותן רצון להשתמש בו הרבה. |
| תיעוד | הקוד מתועד כראוי. |
| אתיקה | הקוד לא מפר פרטיות ולא מנצל חולשה כלשהי אצל המשתמש. |
| אבטחה | אין הכנסת פרטים אישיים שעלולים לגרום להפרת פרטיות כלשהי. עם זאת, ניתן למצוא בקוד את שם המשתמש והסיסמא לכניסה עם האדמין. |
| ביצועים | עקב חוסר מודלריות של הקוד, הוא חשוף יותר לפגיעות גם כאשר השינויים קטנים. |
| ביצועים | הממשק עובד מהר וזמן התגובה נמוך. לדעתי, הביצוע מספיק למשתמש. |
| קריאות - readability | הקוד מובן, אך חלוקת קוד היה מועילה להבנתו, לדוגמה קוד לחיפוש, קוד לצ'אטבוט וכו'. |
| chatbot | לדעתי, זהו החלק העיצובי היפה ביותר בקוד. מסך הצ'אטבוט מזכיר מסכים כמו באתרים פופולריים וגם התגובות שלו מעניינות ועונות על השאלות כראוי. |

1. במעבר על הקוד של קבוצה זו נחשפתי לרעיונות נוספים למימוש הפרויקט שנדרש. לא עלה בדעתי לממש את הצ'אטבוט כמו שהם מימשו וזה פתח לי רעיונות נוספים לעתיד איך לבצע ממשקים של אינטראקציות בין אדם למכונה(מחשב). בנוסף, כתיבת הקוד היה שונה משלנו וגם שם ניתן היה למצוא רעיונות למימוש עתידי כמו מיון לפי קטגוריות.
2. לדעתי מבחינת הפונקציונליות הקוד ממומש בצורה מספקת. כפי שציינתי בטבלה, בעיקר היה כדאי לעמוד יותר בקונבנציות: חלוקה לקבצים מתאימים, יותר תיעוד.

שם הסטודנט: נדב גולדין:

|  |  |
| --- | --- |
| קריטריון | הסבר |
| מימוש | הקוד מבצע את הנדרש, החיפוש והצ'טבוט נותנים תוצאות טובות. |
| יעילות | עובד טוב. אולי ניתן להפריד לקבצים |
| פשטות | הכל נראה יחסית מובן. |
| מודולריות | ניתן להפריד את הקוד לקבצים. |
| תבניות עיצוב | לא מצאתי תבנית עיצוב. |
| OOP | ניתן להשתמש בהורשה. |
| באגים וטעויות | לא נתקלתי בבאגים ובבעיות בקוד. |
| טיפול בשגיאות | יש טיפול ותפיסה של שגיאות. |
| בדיקות | ניתן להעמיס על החיפוש עם כמות גדולה של חיפושים שונים. |
| שימושיות | נוח מאוד לשימוש, מבינים מה כל דבר עושה. |
| תיעוד | הקוד מתועד. |
| אתיקה | הקוד אתי. |
| אבטחה | הקוד מאובטח. המנהל צריך סיסמא ומשתמש על מנת להכנס לאזור שלו. |
| ביצועים | חלוקה לקבצים תשפר את הביצועים. |
| ביצועים | עובד מהר |
| קריאות - readability | קוד מחולק לקבצים היה מאפשר קריאה יותר מובנת של הקוד |
| chatbot | מאוד נוח לשימוש, קל ונראה טוב. |

1. היה מעניין לראות מימוש שונה של אותה מערכת. סה"כ המערכת מאוד נוחה לשימוש, נראית טוב ומזמינה את המשתמש.
2. כפי שנאמר, הפרדת קבצים הייתה מועילה לגמישות ויעילות הקוד.

שם הסטודנט: אלון טרנרידר:

|  |  |
| --- | --- |
| קריטריון | הסבר |
| מימוש | הקוד מבצע את הנדרש, ניתן לבצע חיפוש באתר שהקבוצה קיבלה ולקבל תוצאות מדויקות שבעזרתן אפשר להרחיב וללמוד על נושאי מחשוב ענן.  ניתן לעשות זאת גם על ידי שיחה עם הchatbot וכן למנהל ישנם האפשרויות הנדרשות. |
| יעילות | הקוד יעיל והמערכת עובדת בצורה חלקה. |
| פשטות | אולי רק במסך הסטטיסטיקות שיהיה קצת יותר קריא. |
| מודולריות | היה ניתן לעשות את הקוד יותר מודולרי על ידי חלוקה לקבצים כך שכל קובץ מכיל שירות מסוים.. |
| תבניות עיצוב | לאחר מעבר על הקוד והסבר מחברי הקבוצה, ניתן להבחין במימוש של סינגלטון. |
| OOP | ישנו יישום של מספר עקרונות OOP |
| באגים וטעויות | לא. |
| טיפול בשגיאות | יש טיפול בשגיאות של משתמש ויש טיפול על ידי תפיסת שגיאות בקוד. |
| בדיקות | ניתן לבצע המון הרצות חיפושים לראות איך המערכת פועלת תחת עומס לאורך זמן. |
| שימושיות | ממשק מעוצב ובנוי בצורה מאוד נוחה לשימוש.  ברור מה הפונקציונליות של כל דף ומה צריך לעשות, העיצוב נורא מזמין. |
| תיעוד | קיים תיעוד של הקוד. |
| אתיקה | אין הפרת פרטיות או פגיעה אתית כלשהי. |
| אבטחה | יש אבטחה עבור מסך המנהל, רק מי שיודע את הפרטים רשאי להיכנס, כנדרש. |
| ביצועים | בגלל שאין חלוקה לקבצים אם נרצה להגדיר את המערכת ולשפר ביצועים אולי יהיה קשה יותר. |
| ביצועים | המערכת עובדת בצורה מעולה, זמן תגובה נורא קטן וריצה חלקה. |
| קריאות - readability | בגלל שרובו כתוב בjs כמחרוזת, קצת מאתגר אבל מעבר לזה קוד קריא ומובן. |
| chatbot | כל מה שקשור לchatbot נעשה בצורה מעולה, הממשק שלו מעולה ונוח לשימוש, התשובות שהוא מספק איכותיות והכפתור שמוביל אליו ופותח מסך קטן ולא עוד מסך גדול נורא נוח ומזמין. |

1. נהניתי להתנסות בפרויקט של קבוצה אחרת ולקבל נקודת מבט שונה על המערכת שנדרשנו לעשות. השימוש בקוד שלהם היה מהנה ומאוד נוח ומזמין.

2.רק לחלק לקבצים כדי שיהיה יותר יעיל ויהיה יותר גמיש ופיתוחים נוספים.

שם הסטודנט: צוקרמן:

|  |  |  |
| --- | --- | --- |
| הקוד מבצע את הנדרש | מימוש | |
| היה ניתן לחלק לקבצים, עשוי לשפר מהירות. | יעילות | |
| הקוד נראה די פשוט ומובן | פשטות | |
| אין חלוקה לקבצים, ניתן היה לחלק. | מודולריות | |
| נראה כי לא היה שימוש בתבניות כלל. | תבניות עיצוב | |
| יש שימוש בעקרונות ה OOP ניתן היה להשתמש בהורשה | OOP | |
| לא נתקלתי בבאגים | באגים | |
| אין שימוש בטיפול בשגיאות למעט ביצירת האינדקס. | טיפול בשגיאות | |
| ניתן לנסות בדיקות עומס תחת סקריפט שמריץ המון חיפושים | בדיקות | |
| ממש שימושי ונוח | | שימושיות |
| הקוד מתועד בצורה יחסית נוחה ומובנת. | | תיעוד |
| אין פגיעה בפרטיות או ניצול התנהגות כלשהי. הקוד אתי | | אתיקה |
| הקוד מאובטח, לא גלויים פרטים אישיים | | אבטחה |
| חלוקה לקבצים עשוי לשפר ביצועים | | ביצועים |
| חסר תיעוד בחלקים של ה JS בחלקים של הפייתון הקוד קריא ומובן.  יש צורך בתיעוד הקוד הhtml וה JS כמו כן, חלוקה לקבצים היו יכולים להקל על הבנת הארכיטקטורה. | קריאות | |
|  |  |  |

1. מעבר על הקוד נתן לי השקפה נוספת על הפרוייקט מעיניהם של אחרים, מרגיש שלמדתי מעט מהמעבר על הרעיונות של הקבוצה האחרת.
2. כמו שאמרתי קודם, נדמה כי היה עדיפות לחלוקה לקבצים בנוסף היערות בקוד היו עשויים להועיל עם קורא חיצוני.

שם הסטודנטית: שקד עוז:

|  |  |
| --- | --- |
| **קריטריון** | **הסבר** |
| מימוש | הכל עובד, ניתן לבצע חיפוש, הצ'אטבוט עובד וניתן לנהל שיחה, ניתן לצפות בסטטיסטיקות ולערוך את האינדקס. |
| יעילות | הקוד נראה יעיל. |
| פשטות | הקוד פשוט להבנה, היה יותר פשוט אם היה מחולק לקבצים. |
| מודולריות | כל הקוד בקובץ אחד, היה ניתן לחלק לקבצים קטנים יותר. |
| תבניות עיצוב | לא ראיתי שימוש בתבניות עיצוב. |
| OOP | יש שימוש בעקרונות של OOP . |
| באגים וטעויות | הכל עבד בצורה תקינה. |
| טיפול בשגיאות | יש בקוד שימוש במנגנון תפיסת שגיאות בחלק של הטיפול באידנקס. |
| בדיקות | אולי ניתן לבצע בדיקת ביצועים. |
| שימושיות | הממשק מובן וקל לשימוש, קל להבין איך להשתמש במנוע החיפוש |
| תיעוד | הקוד מתועד מעט אך בצורה מובנת. |
| אתיקה | אין התייחסות למידע אישי של המשתמש בקוד, הקוד אתי. |
| אבטחה | הקוד מאובטח. |
| ביצועים | ביצועי המערכת עובדים טוב, אם הערכת תשונה בעתיד אז אולי יהיה קל יותר להוסיף ולבצע שינויים כאשר יהיה חלוקה לקבצים. |
| קריאות - readability | הקוד היה סך הכל קריא ומובן, מה שהיה קשה זה שהקוד לא מחולק לקבצים, זה יצר קוד ארוך וזה גרם לקושי במעבר על הקוד. |
| chatbot | הצ'אטבוט עובד בצורה נוחה ומהירה, מספק תשובות רלוונטיות, אהבתי שניתן להגיע לצ'אטבוט מכל מסך במנוע החיפוש. |

1. במעבר על הקוד הרגשתי שהקוד ברור ונוח לקריאה, בנוסף היה נחמד לראות מימוש שונה למערכת דומה וללמוד מכך דברים נוספים.
2. לדעתי עבודה משותפת בכיתה עם הקבוצה שעליה ביצענו את הcode review הייתה יותר מלמדת עבור שני הצדדים וזה היה משפר את התרגיל הזה.

שם הסטודנט: אביב בן עזרא:

|  |  |
| --- | --- |
| קריטריון | הסבר |
| מימוש | החיפוש תקין ומציג תוצאות הקשורות לענן. הצ'אט בוט עובד ומנהל שיחה. |
| יעילות | פקודת ה-sleep עלולה להאט את זמן התגובה. |
| פשטות | הקוד מובן וקריא. |
| מודולריות | לא קיימת מודולריות- כל הקוד בקובץ אחד. |
| תבניות עיצוב | לא ראיתי שימוש בתבניות עיצוב. |
| OOP | אין שימוש בעקרונות. הקוד פרוצדורלי. קיימת מחלקה HTML. |
| באגים וטעויות | לא נמצאו. |
| טיפול בשגיאות | קיים טיפול בשגיאות. למשל בשגיאה שעלולה להתרחש במהלך הסרת ערך מאינדקס, על ידי רישום הודעת השגיאה והפרטים למסוף למטרות ניפוי באגים. |
| בדיקות | ניתן לבצע בדיקות ולידציה. למשל בדיקת ה-url |
| שימושיות | קל להבין את השימוש בתוכנה. ניתן לטעות מה שמאפשר להתנסות בפונקציות השונות. |
| תיעוד | קיים תיעוד שורות ספציפיות. יעזור תיעוד לפונקציות. |
| אתיקה | הקוד אינו שומר מידע על המשתמש |
| אבטחה | האתר מאובטח. |
| ביצועים | קיים sleep בכל פניה לאתר. אולי מוטב להשתמש בחוטים. |
| קריאות - readability | הקוד קריא ומובן. מוטב יותר פונקציות. |
| chatbot | מנהל שיחה קולחת ומעניינת. |

1. אני חושב שלראות קוד שונה תרם לי בראיה ביקורתית יותר.
2. עבודה משותפת בכיתה עם קבוצת הביקורת הייתה מלמדת עבור שני הצדדים וזה שיפר את התרגיל.