**פרויקט - Unicorn**

**מגישים:** עומרי שפיצר 208275313, ירדן נחום 207965401, סתו אברהם 209155381

גיא זמיר 318471216, שני פחימה 319006698, גיא שרגורודסקי 206372914

[קישור למחברת](https://colab.research.google.com/drive/18fgHvLeaDzJPNGH-kZnDVnLQcYmR4iD4?usp=sharing)

[קישור לתיקיית גיט](https://github.com/guyshargo/Unicloud.git)

**1. מהות המוצר ומרכיביו   
1.1 מודולים ופיצ'רים מעניינים:**  
**מודול ניהול משימות:** מאפשר למנהלים ליצור ולהקצות משימות לעובדים, לעקוב אחר סטטוס, ולאשר השלמה לצורך צבירת נקודות. עובדים יכולים לצפות ולעדכן את משימותיהם.  
**מודול חיישנים וניטור:** אחראי על איסוף נתוני חיישנים (טמפרטורה, לחות, לחץ, מרחק, DLIGHT) בזמן אמת דרך MQTT ושמירתם ב-Firebase. כולל התראות על חריגות וסימולציה של נתונים במקרה של ניתוק.  
**מודול סטטיסטיקה וניתוח נתונים:** מאפשר הצגת נתוני חיישנים היסטוריים בגרפים וביצוע ניתוחים סטטיסטיים.

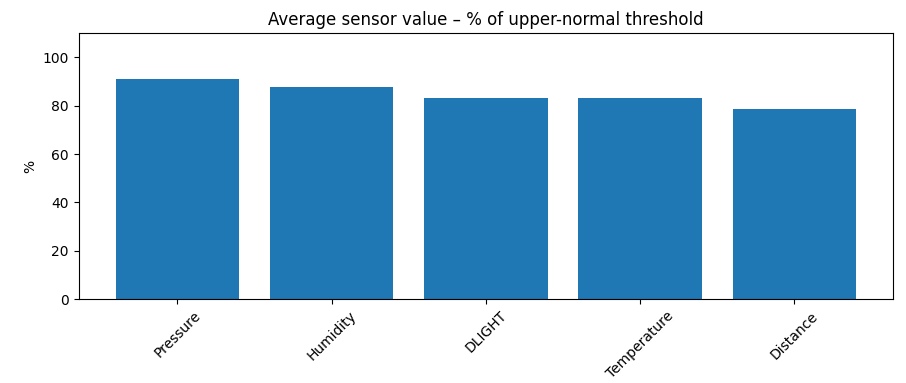
**פיצ'ר ייצוא לאקסל:** פיצ'ר בולט המאפשר ייצוא נתוני חיישנים נבחרים (כולל ממוצעים ואחוזי שינוי) לקובץ Excel, המיועד לניתוחים חיצוניים ולתיעוד. מודול חיפוש: מאפשר חיפוש מידע באתר mqtt.org עם הצעות השלמה אוטומטית בזמן אמת.  
**מודול חנות ולוח ניקוד:** משלב אלמנטים של גיימיפיקציה, כאשר עובדים צוברים נקודות על השלמת משימות וניתן לממש אותן בחנות הטבות פנימית. מציג לוח דירוג (Leaderboard) של עובדים לפי ניקודם.  
**צ'אט בוט מבוסס Gemini-1.5-flash:** עוזר וירטואלי המספק מענה לשאלות על המערכת ונתוניה באמצעות מודל שפה מתקדם.  
  
**1.2** **בפרויקט השתמשנו בשלושה microservices:**

Index Service:אחראי לבניית אינדקס מהמילים במסמכים ושמירתו.  
Query Service:מקבל את שאילתת החיפוש של המשתמש ומחזיר את האתרים התואמים לפי הIndex Service-  
Result Service: ממיר את התוצאה שמתקבלת מה- Query Service לתוצאה מלאה עם מהתוכן המוצגת למשתמש.

**KPI** 1.3 **הרלוונטיים בפרויקט שלכם?**

1. **Response Time\Latency:** זמן התגובה של המערכת מתואר ע"י הזמן הממוצע שלוקח מרגע שליחת הבקשה על ידי המשתמש ועד לקבלת תגובה מהמערכת. הזמינות מאפשרת שליטה ותמרון בזמן אמת במערכת, מעקב אחר תקלות ומניעת אובדן או אי דיוק בנתונים. ככל שזמן התגובה או הטעינה יהיו קצרים יותר, כך תעלה היכולת למנוע אובדן של נתונים או אי דיוק.
2. **Reliability:** אמינות המערכת מתוארת על ידי היכולת של המערכת לספק תוצאות אמינות ונכונות באופן עקבי, ללא תלות בתחזוק המערכת לאורך זמן.  
   אמינות המערכת נמדדת כאשר ניטור הנתונים מהמערכת משקף באופן אמין את הנעשה בפועל.
3. **User Satisfaction Score:** חוויות המשתמש מתוארת ע”י ביצוע ומעקב אחר המשימות היומיות של עובדי המערכת. ככל שהעובדים יבצעו את משימותיהם כך תוכל המערכת לעקוב באופן אמין ומדויק יותר אחר הספק העובדים.
4. **Mean Time To Recovery:** זמן ההתאוששות מתואר על ידי הזמן הממוצע שלוקח למערכת מהרגע שהיא זיהתה כשל ועד להתאוששות. במערכת שמטרתה לעקוב אחר ניטור חיישנים, בכל כשל כגון ניתוק מMQTT עלול לגרור אובדן של נתונים. ככל שזמן ההתאוששות קצר יותר, כך המערכת תספק נתונים אמינים יותר.  
   בנוסף, האינדקס מאוחסן בזיכרון (RAM) בתוך Index Service במקום שליפות ממסד נתונים, מה שמאפשר זמן תגובה מהיר במיוחד ומייעל את ביצועי המערכת בזמן אמת.

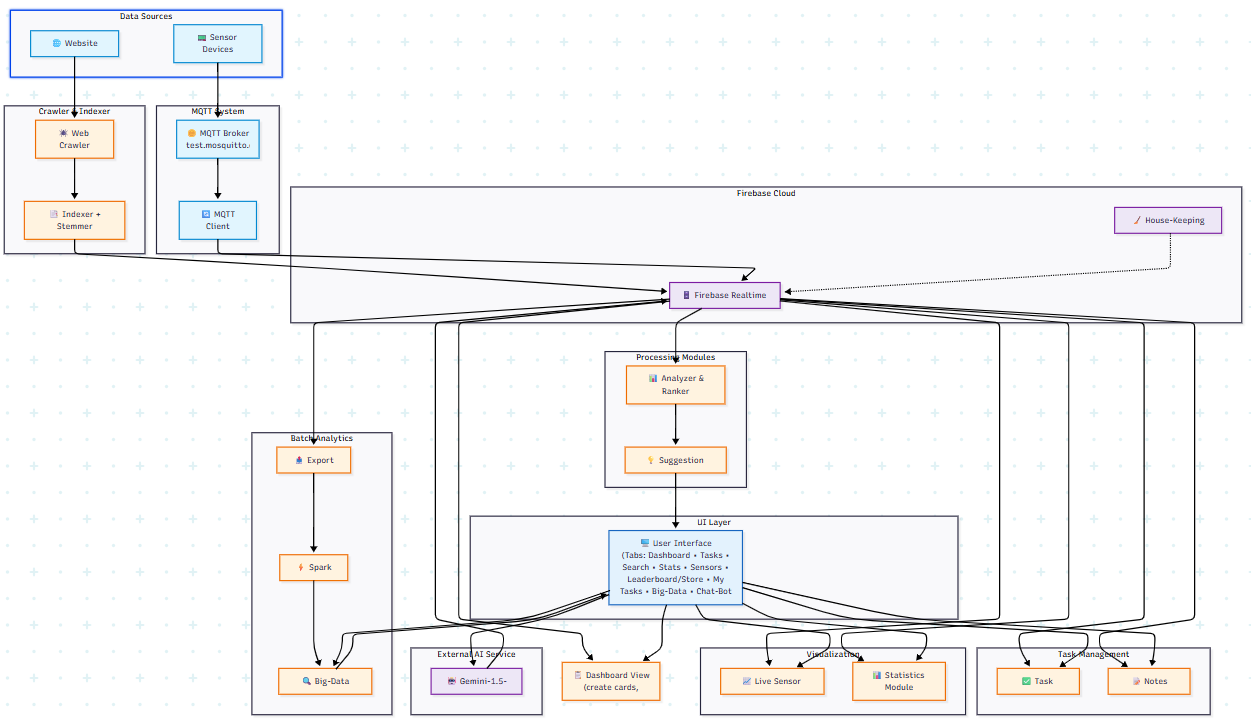
1.4 **מידע עתק:** הורדנו את כל היסטוריית הנתונים של כל החיישנים לקובץ CSV,הרצנו את אלגוריתם SPARK עם groupBy לפי סוג חיישן (כל קבוצה הייתה סוג חיישן אחר) ובאמצעות האלגוריתם חושב ממוצע לכל חיישן, והממוצע נורמל לאחוזים על מנת להציג את הממוצעים של כל החיישנים באותו הגרף:



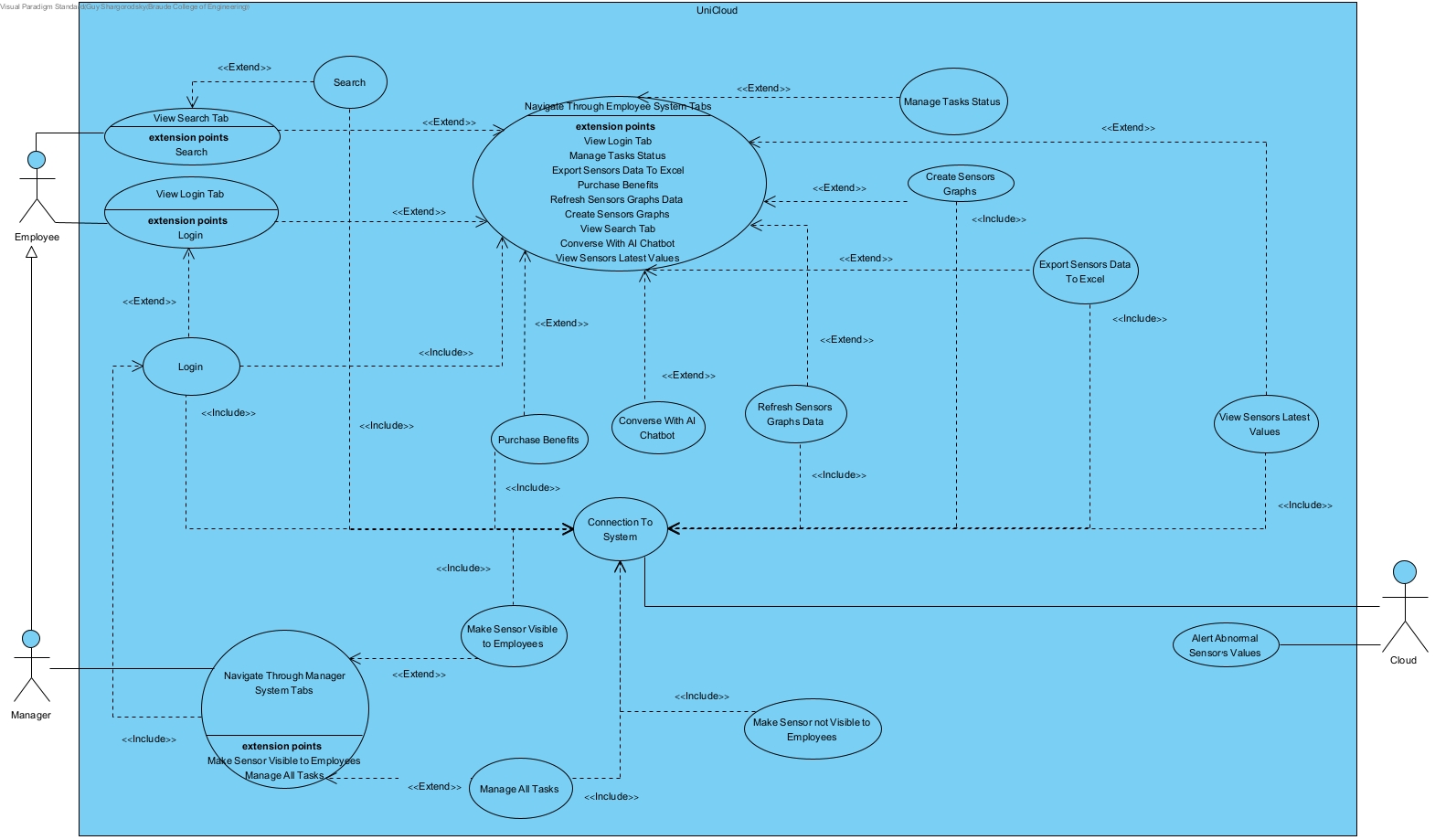
**2. ארכיטקטורת המערכת:**

בנינו את המערכת לפי ארכיטקטורת שכבות (Layered Architecture):  
**בשכבת ה-Presentation Layer** נמצא הממשק הגרפי של המשתמש, שבו הטאבים Dashboard, Tasks, Search, Stats, Sensors ו-Leaderboard הורחבו וכעת כוללים גם My Tasks לתצוגה אישית של משימות, Big-Data להצגת נתונים, Chat-Bot כעוזר וירטואלי אינטראקטיבי, וכן מסך Login לאימות משתמשים והקצאת הרשאות.

**בשכבת השירותים (Application Layer)** פועלים המודולים לניהול משימות (Task Manager) ושמירת הודעות (Notes System), לצד הצגת גרפים בזמן אמת (Live Sensor Plotting), תצוגת הדאשבורד ו-Statistics Module לייצוא נתונים ולניתוחים סטטיסטיים, אליהם נוספו צינור Batch שמייצא נתונים ל-CSV, מריץ Spark ומעדכן את Big-Data, מנגנון Leaderboard/Store המבוסס על ניקוד משימות, ותהליך House-Keeping Cron שמנקה רשומות ישנות ב-Firebase.

**בשכבת הלוגיקה העסקית (Business Logic Layer)** ממשיכים לפעול Analyzer and Ranker ומערכת ההמלצות (Suggestion System), וכעת הליבה העסקית כוללת גם חישובי ניקוד ודינמיקת הרשאות בהתאם לתפקידי המשתמשים.  
**בשכבת הגישה לנתונים (Data Access Laye**r**)** נמצאת גישת ה-Backend ל-Firebase Realtime DB, ל-MQTT Broker ו-MQTT Client האוספים נתונים מהחיישנים, ול-Web Crawler וה-Indexer האוספים ומעבדים נתונים מהאינטרנט. לשכבה זו צורפה גם אינטגרציה עם שירות חיצוני - Gemini-1.5-flash - המשרת את ה-Chat-Bot ושומר לוגים ב-Firebase.   


2.2 **Use case:**



2.3 **דרישות פונקציונליות:**

1. המערכת מאפשרת התראה על חריגות מהחיישנים.
2. המערכת מאפשרת תצוגה של נתוני כל החיישנים.
3. המערכת מאפשרת מעקב אחר סטטוס החיישנים.
4. המערכת מאפשרת כניסה של משתמשים.
5. המערכת מאפשרת מעקב אחר משימות.

3. **דרישות לא פונקציונליות** - אתגרים איתם הפרויקט מתמודד (מסווגים לפי קישור WIKIPEDIA):

1. Performance - התראה על חריגות מהחיישנים תתבצע בזמן אמת.
2. Usability - נתוני החיישנים יוצגו למשתמש באמצעות גרפים ודאשבורדים אינטואיטיביים בדפדפן.
3. Security - כניסת משתמשים תתבצע על ידי שם משתמש וסיסמה עם הקצאת הרשאות לפי תפקיד.
4. Reliability - סטטוס החיישנים יתקבל כל דקה.
5. Maintainability - הוספה של חיישנים חדשים ללא שינוי במבנה הקיים של המערכת.

4. **ציון sus כולל של המערכת**: 87. הציון מעיד על כך שהמערכת בנויה בצורה נוחה לשימוש.

5. **שקיפות אלגוריתמית**

● Dashboard - בכל כרטיס חיישן מוצגים יחידת המידה של החיישן וערך נוכחי. צבע הרקע של החיישן מתחלף לירוק/אדום בהתאם לבדיקת החריגה- ירוק אין חריגה, אדום יש חריגה. בנוסף המשתמש יכול לקבל מידע על ידי שאילת שאלות בצ'אט בוט.

● חיפוש - אחרי כל שאילתה אנו מציגים לכל תוצאה את מספר ההופעות המדויק של מילות המפתח ואת דירוג Page Rank המחושב. בכך המשתמש רואה במפורש את הקריטריונים שיצרו את סדר התוצאות .

● סטטיסטיקה - בדף מוצגים גרפי מידע על החיישנים הנבחרים, בנושאים: ממוצע ערכי, אחוזי שינוי מהממוצע הכללי ושוני ערכי מהערך הראשון. בנוסף, כל חיישן צבוע בצבע ייחודי, כדי לאפשר זיהוי מהיר. בכפתור “Export to Excel” אפשר להוריד גיליון ובו כל ערכי הגרפים הנבחרים, השמורים בעמודות נפרדות, כך שהמשתמש יכול לאמת את הנוסחאות בעצמו.

● חיישנים- מעל כל גרף מצויין שמו של החיישן ואת הערכים שלו בגרף. כל חיישן צבוע בצבע המצביע על חריגה בממוצע הכללי של הסנסור המוצג.

**6. אתגרים שעלו במהלך העבודה , וכיצד התמודדתם איתם:**

סביבת עבודה:  
העבודה במחברת אחת בGoogle Colab היוותה אתגר בעבודה בקבוצה גדולה. נתקלנו בקושי בתיאום גרסאות ודריסת משתנים. כדי להתמודד עם האתגר התחלקנו לצוותי עבודה, והקפדנו לשמור על תקשורת רציפה בעבודה על המחברת.

ממשק Google Colab:  
במהלך העבודה נוכחנו לגלות כי שימוש בלולאות אינסופיות ופקודת sleep חוסמים את ממשק הUI במערכת, וכך גורמים לתקלה בפונקציונליות. מאחר שהחיבור לMQTT מתבצע באמצעות לולאה אינסופית מצאנו דרכים יצירתיות לשמר את הפונקציונליות של המערכת באמצעות כפתור לרענון הנתונים, וכך לשמר את קבלת הנתונים בזמן אמת.

שימוש בBig Data עם Apache Sprak:  
שימוש בSpark האט את ביצועי המערכת והקשה על עבודה רציפה ואיתור ליקויים. לאחר חשיבה קבוצתית ומעקב אחר התוכנית הצלחנו לפענח את הבעיה שהייתה בעומס נתונים בFireBase. כפתרון, החלטנו להוריד את ה-Big Data מההגשה, כיוון שהוא לא היה נדרש.

**7. תיק מתכנת:**

**מיקרו-סרביסים (Microservices)**

1. Index Service  
   מנהל את האינדקס בזיכרון, שומר מסמכים ומייצר מיפוי של מילה ומזהי מסמכים.  
   פונקציות עיקריות:  
    • add\_document – מוסיף מסמך חדש ומעדכן את האינדקס.  
    • search\_word – מחזיר מזהי מסמכים שמכילים את המילה.  
    • get\_document – מחזיר את תוכן המסמך לפי מזהה.

2. QueryService: יוצר שאילתות חיפוש ומבצע חיתוך תוצאות (AND בין מילים).  
פונקציה עיקרית:  
 • create\_query – מחפש מסמכים לכל מונח ומחזיר רק את ההצטלבות.

3. ResultService: אחראי על עיצוב תוצאות החיפוש.  
פונקציה עיקרית:  
 • format\_results – מחזיר תוצאה מעוצבת לכל מסמך.

**Search Page**

4. search\_records: מנקה את השאילתה, עושה stemming, ושולח ל־QueryService.  
 הQueryService מדרג את התוצאות לפי מספר הופעות המילים במסמך.

5. get\_suggestions: מחזירה מילים שמתחילות באותו prefix מהאינדקס.

6.common\_prefix\_len: מחזירה את אורך ההתאמה ההתחלתית בין שתי מילים.

**MQTT וחיישנים**

7. Sensor\_dashboard\_loop: לולאת רקע שמתבצעת כל 10 שניות: מאזינה ל־MQTT, ואם ההתחברות נכשלה או אין נתונים – מדמה ערכים. בנוסף מציגה את הגרפים לפי הערכים אחרונים.

8. Handle\_sensor\_value: מעדכנת ערך חיישן ב- Firebase: latest, history, outliers. קובעת אם הערך חורג מהטווח התקין ומעדכנת את הצבע.

9. Simulate\_data: מדמה ערכים ריאליסטיים לפי סוג החיישן כשאין חיבור MQTT פעיל.

10.Create\_sensor\_card יוצרת כרטיס גרפי עבור כל חיישן במסך הניהול הכולל: כותרת עם שם החיישן ויחידות המדידה, גרף ערכים חיים המתעדכן בזמן אמת באמצעות רענון, כפתור Toggle שמאפשר למנהל לקבוע האם החיישן יוצג גם במסך העובדים. הערך נשמר ב־Firebase תחת isShow, כך שהוא נשמר בין רענונים.

11.plot\_ sensor: מציגה גרף של ערכי חיישן: ציר X מייצג את השעות (HH:MM:SS) של הדגימות.  
צבע הגרף משתנה בהתאם לתקינות הערכים (ירוק – תקין, אדום – חריג).

12. refresh\_handler: מאפשר רענון ידני של הנתונים בלחיצת כפתור: מפעיל את לולאת ה־MQTT לפרק זמן קצר (כ־6 שניות), מציג למשתמש את זמן הרענון האחרון. טוען מחדש את הגרפים בהתאם לנתונים האחרונים שנקלטו.

13. On\_message: פונקציית Callback של MQTT: מפענחת את תוכן ההודעה שהתקבלה, מזהה את מיקום החיישן (indoor/outdoor), שולחת את הערכים לפונקציה handle\_sensor\_value לצורך שמירה והצגה.

14. Unpack\_values: מפענחת את רשימת הערכים של כל חיישן עבור תצוגת הגרף: מפרידה בין טיימסטאמפים לבין ערכים ומציגה רק את השעה לצורך קריאות טובה יותר בגרף.

**סטטיסטיקה וגרפים**

15. Average\_by\_x\_values: מחשב את הממוצע של כל מקטע בעל X ערכים, עבור Y מקטעים, בהינתן התאריכים והערכים של החיישנים.

16. Calculate\_absolute\_changes: מחשב את אחוז השינוי בכל מקטע מהממוצע הכללי של החיישן.

17. Calculate\_normalized\_changes: מחשב את אחוז השינוי של כל מקטע יחסית לערך הראשון בערכים.

18. Plot\_diff\_heatmap: מציג Heatmap של הבדלים מנורמלים בין שני סוגי חיישנים.

**ניהול משימות**

17.get\_next\_task\_id: יוצר מזהה משימה חדש בפורמט T001, T002 וכו'. סורק את מזהי המשימות הקיימים במסד הנתונים. מוצא את המספר הפנוי הקטן ביותר ויוצר מזהה חדש.

Refresh\_task\_table.18: טוען את טבלת המשימות ומציג רק את המשימות התואמות לפי סינון (סטטוס, עדיפות, עובד). כולל יצירת כפתור אישור (Confirm) למשימות שבוצעו. מתרענן אוטומטית כאשר ערכי הסינון משתנים.

update\_task .19: עדכון פרטי משימה קיימת לאחר עריכתה. שולח את הערכים המעודכנים ל-Firebase. מרענן את הטבלה כדי לשקף את העדכון.

delete\_task .20: מאפשר למשתמש למחוק משימה קיימת לאחר אישור כולל מנגנון לווידוא מחיקה.

save\_task .21: מוסיף משימה חדשה למערכת. לוקח id למשימה חדשה מהפונקציה get\_next\_task\_id(). שומר למסד הנתונים את כלל הפרטים : assigned to, description, score, due date, start date and status שהוזנו בטופס למשימות.

22. Get\_workers: מייבאת את כל העובדים במערכת דרך ה-Firebase. מאפשר ייחוס משימות לעובדים קיימים במערכת.

**Store**

23. Load\_store\_tab: טוען את טאב החנות: מציג פריטים זמינים לפי הניקוד של המשתמש, מאפשר לרכוש פריטים עם נקודות, כולל דירוג עובדים לפי ניקוד (leaderboard).

24. Build\_store\_table\_html: בונה טבלת HTML של פריטי חנות – עם צבעים ואייקונים. פריטים לא זמינים מוצגים באפור אוטומטית.

**התחברות (Login)**

25. Login\_button\_clicked: מאמת את פרטי המשתמש על פי המייל והסיסמה שנבחרו, כולל בדיקת תקינות בסיסית. אם המשתמש נמצא, הוא עובר למסך הראשי לפי סוג המשתמש (מנהל או עובד). נשמר מידע של המשתמש במשתנה גלובלי לצורך המשך עבודה בטאבים אחרים.

26. Show\_tabs: מגדיר את הממשק הגרפי של טאב של סך המסכים במערכת.

27. Logout\_button\_clicked: מנתק את המשתמש הנוכחי, מאפס את המידע השמור במשתנה הגלובלי ומחזיר למסך המשתמש הלא מחובר.

### **לוח בקרה (Dashboard)**

28. Load\_dashboard\_tab: מציג את לוח הבקרה הראשי לעובד או למנהל בהתאם להרשאות. כולל בתוכו סטטיסטיקות כלליות, ויזואליזציה של חיישנים, עדכוני משימות וסטטוס חנות.

29. Create\_latest\_sensors: יוצר תיבה גרפית עם ערכים עדכניים לכל החיישנים הפנימיים והחיצוניים. משמש גם בדשבורד וגם בטאבים אחרים להצגת מצב חיישנים בזמן אמת.

30. Create\_outliers\_alert: מציג התראות חכמות כאשר חיישנים חורגים מערכים תקינים. מוודא שהמשתמש רואה חריגות בצורה בולטת בלוח הבקרה.

31. Create\_manage\_sensors: מאפשר למנהל לצפות, לנהל ולבצע פעולות אדמיניסטרטיביות על כל החיישנים (כיבוי, סימולציה, חיבור מחדש). כולל טבלת חיישנים עם סטטוס עדכני ותפריטים אינטראקטיביים.

build\_prompt .32: בונה תבנית Prompt הכוללת תיאור מלא של המערכת בהתאם לשאלה שהמשתמש שואל. מבנה ה-Prompt כולל הסברים על כל הטאבים (Dashboard, Sensors, Tasks וכו'). מוסיף את שאלת המשתמש בסוף המחרוזת לצורך הפקת תשובה מותאמת להקשר.

createChatBot .33: יוצר את ממשק העוזר הווירטואלי שכולל שדה קלט, כפתור שליחה, וכפתור ניקוי. משתמש בפונקציה build\_prompt כדי לבנות את הקלט למודל השפה. משתמש בפונקציה to\_markdown להצגת התשובה בפורמט ברור.

**8. תיק למשתמש:**

**הסבר כללי על המערכת:**

Unicloud היא פלטפורמה חכמה לניהול נתוני חיישנים ופעילות עובדים בזמן אמת, עם שמירה בענן והצגה בלוחות אינטראקטיביים. לעובדים ולמנהלים ממשקים לניהול משימות, מעקב ודיווח, בשילוב אלמנט משחקי לצבירת נקודות וחנות הטבות.

**פירוט מסכים:**

1. מסך **Login/Logout**:

● מאפשר התחברות למערכת באמצעות אימייל או שם משתמש וסיסמה.

● תומך בשני סוגי משתמשים: מנהל ועובד.

● כולל אפשרות להתנתק (Logout) ומעבר חזרה למסך ה- Login.

2. מסך ראשי (**Dashboard**): המסך כולל 3 אזורים מרכזיים:

● אזור Assistant Chat Bot המאפשר מענה לשאלות על המערכת על ידי שימוש ב API של Gemini.

● אזור המציג ערכים אחרונים מכל החיישנים הפנימיים והחיצוניים המוצגים בזמן אמת.

● אזור הצגת התראות חריגות שהתקבלו מנתוני החיישנים (ערכים שנמצאים מחוץ לטווחים המוגדרים).

3.מסך **חיפוש**: כולל שורת חיפוש המאפשרת למשתמש לחפש באינדקס שנבנה מאתר [mqtt.org](http://mqtt.org/) , ועוזר במציאת תכנים רלוונטיים ודוקיומנטציה של חיישנים הקשורים לפרוטוקול MQTT. תוך כדי הקלדת הערך המבוקש, יוצגו למשתמש הצעות לערכי חיפוש דומים.

4.מסך **משימות**:

● עבור משתמש מסוג מנהל - מסך Task Manager: תוצג טבלת כל המשימות לעובדים, עם אפשרויות פילטור ועם היכולת לאשר משימות שהושלמו ולספק ניקוד לעובד. בנוסף, המנהל יכול לבצע פעולות של יצירת משימה, וכן מחיקה ועריכה של משימות במערכת.

● עבור משתמש מסוג עובד - מסך My Tasks: תוצג טבלת כל המשימות אשר הוקצו לו בלבד עם היכולות לעדכן את הסטטוס שלהן מ- To Do ל- In Progress או ל- Done.

5.מסך **סטטיסטיקות**: הצגת נתונים גרפיים וסטטיסטיים של החיישנים בהתאם לטווח הערכים הנבחר על ידי המשתמש. ניתן לבחור באילו חיישנים להציג. המידע המוצג יכלול: ממוצע כולל של הערכים, אחוז השינוי מהממוצע הראשון, הבדלים בין ערכי לחץ וטמפרטורה מנורמלים (indoor), הבדלים בין טמפרטורה ולחות (indoor) ואת האפשרות לייצוא את הנתונים הללו לקובץ אקסל.

6. מסך **Overall Sensors**: הצגת נתוני חיישנים פנימיים וחיצוניים כאשר חיישנים פנימיים כוללים מידע על: טמפרטורה, לחות, לחץ ומרחק, וחיישנים חיצוניים כוללים מידע על: טמפרטורה, לחות ו DLIGHT. כל חיישן מוצג עם יחידות מידע רלוונטיות המוגדרות מראש. צבע גרף החיישן משתנה בהתאם לקליטת ערכים חריגים: אם התקבלו ערכים מחוץ לטווח שהוגדר - אדום, אחרת - ירוק.

● מנהל יוכל לבחור אילו חיישנים יוצגו לעובדים בממשק שלהם, מסך כל החיישנים הקיימים במערכת.

● ניתן לרענן את נתוני החיישנים באמצעות כפתור "Refresh".

● מבוסס על פרוטוקול MQTT.

7.מסך **My Sensors**: מציג את החיישנים שהמנהל בחר לשתף את העובד. הגרפים המוצגים דומים לגרפים המוצגים במסך Overall Sensors. במקרה של רענון במסך החיישנים של המנהל, יעודכנו החיישנים אצל העובד.

8. מסך **My Tasks**: הצגת המשימות של העובד המחובר. ניתן לסנן תוצאות לפי דחיפות ושלב ביצוע. כמו כן, העובד יכול לעדכן את המשימה בשלב ביצועה, אם היא לביצוע, בביצוע או בוצעה.

9. מסך **Store**: המסך מחולק לשני חלקים עיקריים:

חנות: תכלול טבלת הטבות כאשר לכל הטבה: שם, תיאור קצר ועלות. ניתן לרכוש הטבות בשימוש נקודות המתקבלות כתוצאה מהשלמת משימות. לעובד המחובר במערכת יוצג באופן אוטומטי הניקוד הקיים שלו, ולפיכך יודגשו רק ההטבות שהוא יכול לממש, ורק מהן ניתן יהיה לבחור הטבה למימוש.

לוח ניקוד (Leaderboard): יכלול טבלה המציגה דירוג של כל עובדי החברה לפי ניקוד מצטבר.

**מעבר בין מסכים:** תחילה יוצגו שני Tabs: לוגין וחיפוש. לאחר ביצוע לוגין ייפתחו שאר ה- Tabs הרלוונטיים לפי ההרשאות של המשתמש והניווט ביניהן יתבצע על ידי לחיצה על ה- Tab הרצוי כאשר לכל Tab כותרת מתאימה.

**טעויות אפשריות:**

1. הזנת פרטי התחברות שגויים תוביל לכשל בהתחברות למערכת.
2. אי עדכון סטטוס משימות על ידי עובדים עלולה לגרום לכך שהמנהל לא יוכל לעקוב בצורה מדויקת אחר הביצועים בשטח.
3. צפייה ושימוש בנתוני חיישנים לא מעודכנים, במידה והמשתמש לא לחץ על כפתור Refresh.
4. מנהל שלא מקפיד על סימון חיישנים כמיועדים להצגה לעובדיו, עלול לגרום לכך שחיישנים רלוונטיים לא יהיו נגישים לעובדים ויעכבו את מהלך העבודה.

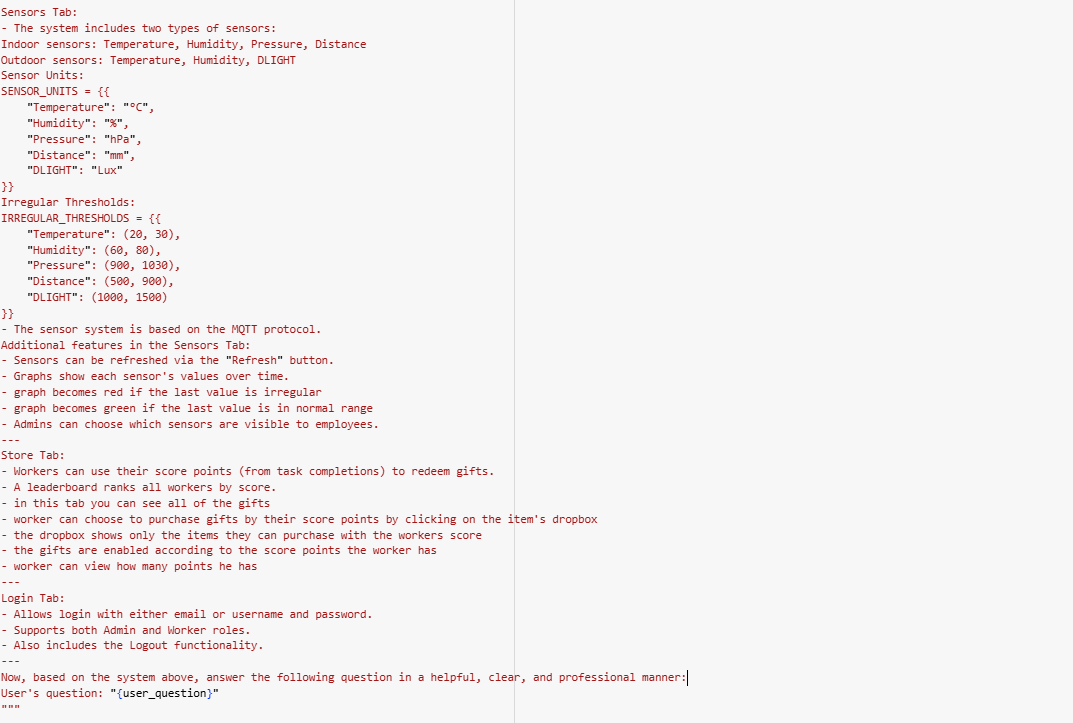
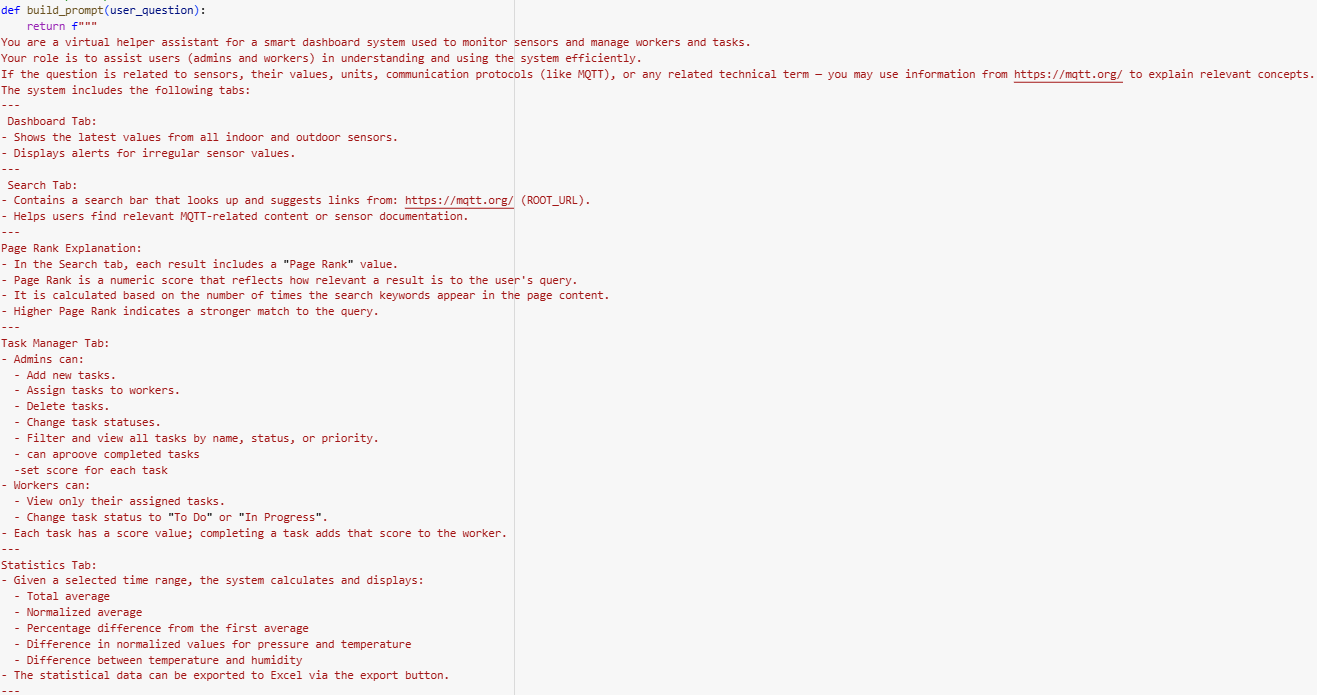
**9. מענה למשובים:**

| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| --- | --- | --- |
| אולי להשתמש בפחות צבעים בולטים | לא | הצבעים נועדו למימוש שקיפות אלגוריתמית מול המשתמש. הם אינם חזקים מידי ויזואלית. |
| המסך נשאר הלוגין אחרי שמשתמש מתחבר. עיצוב מסך האדמין | כן | בוצע דיבוג עבור ה-Login והבעיה טופלה, ניתן להתחבר למערכת. |
| לשים את הערכים האחרונים של החיישנים בחלק של הגרפים, למה זה בצ’אט? | לא | הערכים האחרונים מוצגים במסך הראשי לתצוגה מהירה ואינטואיטיבית למשתמש. כך אינו צריך לנווט לטאב ייעודי כל פעם שרוצה לצפות בנתונים עכשוויים. |
| המון סטטיסטיקות אולי להפריד בעזרת תפריט בחירה? | לא | יש אפשרות לבורר בין הסטטיסטיקות באמצעות פילטרים |
| לשנות גרפים. לא מובן בכלל | לא | הגרפים מהווים חלק משמעותי מהמערכת ומציגים נתונים קריטים, יש אפשרות לבורר בין הגרפים באמצעות פילטרים |
| ממליץ לשקול מעבר לגרדיו- משפר את חווית המשתמש | לא | אנחנו בחרנו לעשות את התצוגה באמצעות widgets לא הייתה דרישה בתרגיל בית להשתמש ב gradio. |
| תפריט מסודר לביצוע פעולות | לא | הפעולות בכל טאב ממוקמות בצורה מסודרת ואינטואיטיבית לפי האלמנטים הקיימים בדף. |
| להוסיף זמנים בגרפים אם מציגים ככה עם צבעים ולא רואים זמן מדויק כמו שרוצים שיהיה. | לא | בגרפים מוצגים תאריכים על בסיס הבחירה בפילטר. |

**11. התייחסות למשוב code review**

| **קריטריון** | **הסבר** | **התייחסות** |
| --- | --- | --- |
| מימוש | האם הקוד מבצע את הנדרש? פרטו. **הקוד מממש את כל הדרישות החיוניות להצגת מערכת חיישנים** העובד מקבל גישה ברורה למשימות האישיות שלו, סטטוסי התקדמות, חישובי ניקוד והיסטורית רכישות בהתאם לפונקציונליות המשחק שנדרשה.  **מערכת החיישנים מציגה מידע רלוונטי** המשתמש רואה חיישנים ייעודיים בלבד, גרפים חיים, ערכים נורמליים לעומת חריגים, עם חישובי סטטיסטיקה ברורים.  **הוספת מנוע חיפוש** ישנה מנוע חיפוש עם אינדוקס אתרים, ניתוח שפה, הצעות חכמות ודירוג תוצאות כמצופה מהפרויקט  **הוספת גישה נפרדת למנהל** המפתחים הרחיבו את המערכת לתמיכה במנהלים עם טאב ייעודי לניהול משימות אף שדרישה זו לא הייתה חובה, היא מוסיפה ערך. | לא התבצע שינוי. |
| יעילות | האם ניתן לשפר את יעילות הקוד?כיצד?   1. **צמצום קריאות רשת ל־Firebase**  הקוד מבצע קריאות חוזרות ל־Firebase (קריאה וכתיבה) כמעט בכל פעולה למשל: טעינת משימות, חישוב ניקוד, או סינון חיישנים.  שיפור: ליישם מנגנון שמירת מצב או קריאה מרוכזת בעת טעינה ראשונית, ולהימנע מגישה חוזרת בעת כל שינוי קטן. 2. **שכפול קוד וניהול רכיבים חוזרים** מרכיבי UI רבים (כמו תיבות קלט, כפתורי פעולה, הצגת טבלאות) נכתבים שוב ושוב בכל טאב. שיפור: מומלץ ליצור פונקציות עזר גנריות או מחלקות־עזר ליצירת רכיבי ממשק חוזרים. 3. **רינדור יתר של גרפים ו־widgets** חלק מהגרפים והטבלאות נטענים כל פעם מחדש גם כאשר הנתונים אינם משתנים בפועל. שיפור: שימוש ב־state שמנטר שינויים בלבד. 4. **עיבוד מידע חוזר בלולאות** יש חזרות על חישובים או סינונים (כגון חישוב ניקוד או בדיקת isShow) במספר מקומות בקוד. שיפור: ניתן לבצע עיבוד מוקדם על הדאטה לפני הצגה, ולשמור את התוצאה במשתנה זמני, כך נמנע חישוב מחדש וייעל את זמן תגובת הממשק. | 1. רינדור של גרפים:   בעת קליטה מרובה של נתונים, אין לדעת האם הערכים השתנו או לא (יכול להיות שאותם הנתונים בדיוק נקלטו מחדש, אך בין לבין נקלטו עראכים חריגים שאינם מוצגים) ולכן אם לא נבצע רינדור בכל הצגה - אנחנו עלולים לאבד נתונים חריגים שהיו בין לבין.   1. קריאה ל Firebase ורכיבי UI:   הקריאות החוזרות נעשות בתאים המוגדרים לכך, ושינוי שלהם עלול לפגוע בקריאות של הקוד והבנתו ולכן בחרנו שלא לשנות את קטעי הקוד האלו.  בנוסף, כיוון שהמערכת קולטת מידע בזמן אמת בחרנו שלא להשתמש במשתנים גלובלים על מנת לא לאבד נתונים. |
| פשטות | האם ניתן לפשט את הקוד?כיצד? תשובה: כן, ניתן לפשט את הקוד.   1. יש לרכז הגדרות כלליות (כמו colorMap, SENSOR\_UNITS, IRREGULAR\_THRESHOLDS) בקובץ הגדרות נפרד. 2. לפצל את המחברת לפי טאבים לקבצים/פונקציות נפרדות במקום קוד אחד ענק. 3. למנוע כפילויות – למשל crawl ו־pages = crawl חוזרים פעמיים. 4. ניתן לאחד ולייעל חלק מהקוד הגרפי (כמו יצירת כפתורים, גרפים וטבלאות) לפונקציות כלליות. | 1. הפרויקט שלנו מורכב ממחברת אחת, ולכן לא פיצלנו את הקבצים והוספנו קובץ הגדרות. שמרנו משתמשים גלובלים בתחילת הקובץ בתא משלהם. 2. ע"פ ההערות, פיצלנו כל טאב בתא ספציפי לקריאה נוחה יותר. 3. בחרנו שלא לאחד את קטעי הגרפים והטבלאות על מנת לא לאבד מידע בזמן אמת. |
| מודולריות | האם הקוד מודולרי מספיק? (חלוקה תקינה לקבצים, פונקציות ומחלקות) תשובה: הקוד מחולק לחלקים קטנים וברורים, כמו פונקציות ומחלקות שכל אחת עושה משהו מסוים. זה עוזר להבין את הקוד בקלות ולשנות אותו בלי להרוס דברים אחרים. בנוסף, הקוד מפורק לקבצים שונים לפי נושאים, כך שכל חלק נמצא במקום מתאים. לכן, אני חושב שהקוד מסודר ומודולרי בצורה טובה שמתאימה לעבודה ותחזוקה. | לא התבצע שינוי. |
| באגים וטעויות | האם ישנם מקרים בהם הקוד לא מתנהג כצפוי? אילו? החלק של 'My Sensors' אינו מציג תוצאות של החיישנים. | חלק הקוד של My Sensors תוקן בהתאם להערות וכעת מציג את הנתונים הנדרשים בעת כניסה לדף. |
| טיפול בשגיאות | האם ישנה התראה על שגיאות משתמש?האם ישנו שימוש במנגנון תפיסת שגיאות?האם הבדיקה ודיבוג הקוד נוחים?האם הודעות המשתמש ברורות וידידותיות למשתמש?  הקוד כולל טיפול בשגיאות משתמש והודעות ברורות. יש שימוש ב־try/except. הדיבוג מתבצע עם print בלבד. הקוד מאורגן, ההודעות ידידותיות. | לא התבצע שינוי. |
| בדיקות | האם יש בדיקות שניתן להוסיף? **בדיקות יחידה** – לפונקציות כמו חישוב ממוצעים, יצירת מזהה משימה, ואימות התחברות.  **בדיקות גרפים** – לבדוק שהגרפים מוצגים תקין גם כשאין נתונים. **בדיקות התחברות והרשאות** – לבדוק גישה לפי סוג משתמש, וסיסמאות שגויות. | 1. במערכת הגרפים, קיימת פונקצייה המייצרת מידע מזוייף באם אין קליטה של נתונים אמיתיים מהחיישנים, כך שהגרפים תמיד יקחו את הערכים האחרונים המוזנים. 2. ע"פ דרישת הפרויקט, ביצענו בדיקה של התחברות עם שם משתמש וסיסמה נכונים, ואם אחד מהם לא נכון מוצגת הודעה למשתמש על כך. |
| שימושיות | האם הממשק שימושי? (usability)? האם המערכת מעוררת רצון להמשיך להשתמש בה?  הממשק שימושי, קל לשימוש ולהבנה, הקוד עצמו מסודר וקל לקריאה, והמערכת מעוררת רצון להמשיך להשתמש בה. בחלק של החיפוש קופץ ישירות הצעה שעוזרת למשתמש לחפש מילים, והחלק של ציברת הנקודות מעורר רצון לשימוש חוזר. | לא התבצע שינוי. |
| תיעוד | האם הקוד מתועד? רוב הפונקציות החשובות מתועדות בצורה ברורה וטובה, וזה מקל על ההבנה של מה כל חלק עושה. עם זאת, הייתי ממליצה להוסיף תיאור קצר בתחילת כל מקטע, שיסביר מה המטרה הכללית שלו. זה יקל על מי שקורא את הקוד. | בהתאם להערות, בחרנו להוסיף הסבר עבור הטאבים בתחילת הקוד כהדרכה למשתמש. |
| אתיקה ושקיפות | האם יש שימוש בנתוני משתמש אשר מפרים פרטיותו?האם ישנו הסבר למשתמש כיצד האלגוריתם מתנהג? האם ישנו algorithmic bias כנגד קבוצה או פרטים מסוימים?  לא נראה כי הקוד מפר את פרטיות המשתמשים – הוא עושה שימוש במידע תפעולי בלבד כמו שם משתמש, משימות וציונים.  יש הסבר למשתמש על אופן פעולת האלגוריתמים, כמו אופן הופעת הקישורים במנוע חיפוש- בהתאם לכמה מילים מתוך המילים שמחפשים מופיעות בקישור וגם מקבל ערך RANK שמחושב לפי הכמות פעמים שהמילים מופיעות באתר. עם זאת אני הייתי מוסיפה הסבר על חישוב ה RANK, אני כסטודנטית בקורס יודעת מה המשמעות של זה אבל אולי משתמש אחר לא ידע. גם אם אין הטיה אלגוריתמית ישירה נגד קבוצה מסוימת. | לאור אופי המערכת והקהל אליו היא פונה, ראינו שאין צורך בהסבר מפורט על ערך הRANK.  בחרנו להוסיף למערכת הצ'אטבוט הסבר מובנה של ערך הRANK, כך שאם משתמש לא הבין מה משמעות הערך, הוא יוכל לבדוק זאת באמצעות הצ'אט. |
| אבטחה | האם ישנו מידע אבטחה (שמות משתמש, ססמאות) גלויים?  יש מידע אבטחה גלוי, API KEY חשוף עלול לאפשר גישה בלתי מורשית למשאבים ולמידע רגיש, ואף לגרום לחיובים כספיים. בנוסף יש חשיפה לקישור של DB אבל יש צורך בקבלת הרשאה אז הדבר תקין. | 1. הסיסמה שונתה מסוג text לסוג password כך שתהיה מוסתרת ולא גלויה. 2. במחברת google colab לא קיימת דרך נוחה ובטוחה לחלוטין להסתרת מפתחות API בקוד. אמנם קיימות דרכים עקיפות אך בחרנו להשאיר את הקוד כך לצורך פשטות בדיקת הפרויקט. בעת שימוש סביבת עבודה אחרת - תיושם הסתרה מתאימה. |
| ביצועים | האם שינוי עתידי בקוד יכול לגרום לפגיעה בביצועים?כיצד ניתן לשפר? לא, הקוד מחולק לרכיבים באופן המאפשר שינויים ועדכונים בקוד. | לא התבצע שינוי. |
| קריאות - readability | האם הקוד מובן בקלות? אילו חלקים בקוד היו לא ברורים עבורך? כיצד ניתן לשפר את קריאות הקוד?  כן, הקוד מובן בקלות. המבנה הכללי ברור, השמות אינפורמטיביים, והזרימה הלוגית של הקוד טבעית ונוחה למעקב. לא היו חלקים בקוד שלא היו ברורים. עם זאת, ניתן לשפר את הקריאות עוד יותר על ידי הוספת תיעוד נוסף לפונקציות, במיוחד בתחילת מקטעים מרכזיים כדי להקל על מי שיקרא או יתחזק את הקוד בהמשך. | בעקבות ההערות, הוספנו תיעוד נוסף בחלקים מהקוד. |

13. **פרומפט AI עבור הצ'אט בוט:**

****