המכללה האקדמית להנדסה בראודה A black background with purple letters

Description automatically generated

**המחלקה להנדסת תוכנה - מבוא למחשוב ענן**

**פרויקט סופי קבוצת BEE**

**Git Repository:** [**https://github.com/wadiea1/Cloud-computing**](https://github.com/wadiea1/Cloud-computing)

**Notebook:** [**https://colab.research.google.com/drive/1kSCisvnQMXc36otqAejDfEn0hfW5fRqR?usp=sharing**](https://colab.research.google.com/drive/1kSCisvnQMXc36otqAejDfEn0hfW5fRqR?usp=sharing)

|  |  |  |  |
| --- | --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** | **Acceptance test –** בדיקת קבלה |
| **הילאל עלי** | LEADERBOARD | הכל | המשתמש רואה את מיקומו בטבלה לפי ניקוד שמתקבל מהמשימות |
| **אוייס זועבי** | DASHBOARD. + MICROSERVICES+ LOGIN | הכל | ניתן לראות גרפים בזמן אמת, והמערכת קוראת ממיקרו־שירותים נפרדים |
| נואר דבאח | TASKS + פיצ'ר FAVORITES | הכל | **המשתמש יכול לסמן פריטים כמועדפים, והם נשמרים ומוצגים ברשימה** |
| **ודיע פראן** | SEARCH ENGINE | הכל | **חיפוש לפי מילת מפתח מחזיר תוצאות רלוונטיות תוך פחות מ־2 שניות** |
| **אוסאמה מאדי** | CHATBOT | הכל | ניתן לנהל משתמשים ולשלוח שאילתות לצ'אט ולקבל תגובות הגיוניות |
| **בדר בושנאק** | BACKEND + CONTROL PANEL+  (בניית DB והכנה ) + | הכל | משתמש מקבל קוד אימות למייל, מקליד אותו ונכנס בהצלחה למערכת |

|  |  |
| --- | --- |
| **Username = bader**  **Password = boshnak**  **Username = osama**  **Password = made** | **Admins data:** |
| **Username = ryan**  **Password = ryanpass**  **Username = nina**  **Password = ninapass** | **Users data:** |

1. **מהות המוצר ומרכיביו**

**1.1 מודולים ופיצ'רים מעניינים**

**AutoLine** היא מערכת דשבורד חכמה בענן לניהול קו ייצור אוטונומי. היא כוללת ניטור חיישנים בזמן אמת, דירוג לפי משימות וניקוד, ולמידה אתגרית לשיפור יעילות ובקרה.  
רכיבי המערכת: דשבורד חיישנים, טבלת דירוג, לוח ניהול, עמוד משימות, חיפוש MQTT וצ'אטבוט תעשייתי.

**מודולים ופיצ'רים מעניינים**

* **מנגנון דירוג מהנדסים**  
  טבלת דירוג של חמשת המובילים לפי משימות וניקוד, עם סטטוסים שמעודדים תחרות. הנתונים מתעדכנים אוטומטית.
* **לוח ניהול משימות**  
  סקירה עם פילטרים, גרפים לפי רמת קושי וסטטיסטיקות להנהלה.
* **חיפוש תיעוד MQTT**  
  דף לחיפוש מהיר במדריך MQTT באמצעות אינדקס הפוך.
* **צ'אטבוט BeeChat**  
  מסייע בביצוע משימות ונותן מענה מהיר לשאלות.
* **דשבורד חיישנים**  
  גרפים ונתוני סנסורים בזמן אמת (פנימי/חיצוני) עם סטטוס תקינות.
* **עמוד משימות אישי**  
  לוח משימות לפי קושי, עם ניקוד, תיאור, מועדפים וצבירת נקודות.
* **אימות דו־שלבי (OTP)**  
  קוד חד-פעמי למייל להגנה על חשבון המשתמש.
* **Favorites**  
  שמירת משימות מועדפות. מאפשר חזרה נוחה וניתוח נושאים מובילים.

**1.2 microservices שנעשה בהם שימוש**

בפרויקט שלנו, השתמשנו בארכיטקטורת מיקרו-שירותים (Microservices) כדי לחלק את המערכת לשירותים עצמאיים ונפרדים, כאשר כל שירות מתמקד בפונקציונליות מסוימתת משלו.

|  |  |
| --- | --- |
| **תיאור** | **שם הקובץ** |
| מייצר גרפים (קו, עמודות ועוגה) להצגת נתוני חיישנים, ביצועים, ומשימות בצורה ויזואלית. | ChartGenerationService |
| מנוע חיפוש חכם הסורק אתרי MQTT, בונה אינדקס עם Stemming , ושומר את התוצאות בפיירבייס לצורך חיפוש לפי מילות מפתח. | SearchService |
| שירות תקשורת גנרי לפעולות קריאה וכתיבה מול Firebase Realtime Database  בפורמט JSON. | FirebaseService |
| אחראי על הרשמה והתחברות של משתמשים עם אימות סיסמה,  כולל תמיכה בקוד אימות שנשלח למייל. | UserAuthService |
| UI דינמי שנבנה עם ipywidgets להצגת טופס התחברות,  קוד אימות, וחוויית משתמש מודרנית. | Login Interface Microservice |
| מאתחל את Firebase ומביא נתוני חיישנים חיים או  היסטוריים להצגה בדשבורד. | Firebase Initialization & Sensor Fetching |
| מחשב ביצועים יומיים של מהנדסים  ומציג את הדירוג השבועי בהתאם למשימות שבוצעו ונקודות שהושגו. | Leaderboard Performance Microservice |

**1.3 KPI מרכזיים:**

* **זמינות (Availability):**  
  אחוז הזמן שהמערכת פעילה ונגישה – יעד: מעל 99.5%, בזכות Firebase ושירותי ענן אמינים.
* **אמינות (Reliability):**  
  הצגת ושמירת נתונים (ניקוד, משימות, מועדפים) באופן עקבי וללא תקלות, כך שהמשתמשים סומכים על תקינות המערכת.
* **סקלאביליות (Scalability):**  
  יכולת המערכת להתמודד עם גידול במספר המשתמשים והנתונים בלי ירידה בביצועים, הודות לארכיטקטורת Firebase.

**1.4 ניתוח נתוני עתק**

A blue and black background with white text

AI-generated content may be incorrect.

לצורך ניתוח מידע עתק, השתמשנו במודל **MapReduce** על מאגר המשימות בפרויקט.

1. **תהליך הניתוח:** המודל סרק את כל המשימות, קיבץ אותן לפי רמת הקושי שלהן (Easy, Medium, Hard), ובסוף ספר כמה משימות יש בכל קבוצה.
2. **הצגת התוצאות:** תוצאות הספירה הוצגו ב**גרף עמודות**, כאשר כל עמודה מייצגת רמת קושי, והעמודה השנייה מראה את הכמות הכוללת של המשימות באותה הרמה בהתאמה.

**2. ארכיטקטורת המערכת**

**2.1 תרשים ומאפיינים מרכזיים (מעודכן)**

A diagram of a data flow

AI-generated content may be incorrect.

**ארכיטקטורת המערכת: :MVC (Model-View-Controller)**

בחרנו לבנות את מערכת "AutoLine" בארכיטקטורת MVC, המפרידה בין שכבת הנתונים והלוגיקה (Model), התצוגה למשתמש (View), והבקרה המקשרת ביניהן (Controller). הפרדה זו תורמת לקוד מאורגן וקל לתחזוקה.

**Model (מודל):**

ליבת המודל היא Firebase Realtime Database, האוצר את כל נתוני האפליקציה (משתמשים, משימות, ציונים, נתוני חיישנים). פונקציות ב-Cell 4, 6, 8, 10 וה-UserAuthService (ב-Cell 2) אחראיות על ניהול, שליפה ועדכון של נתונים אלו מול Firebase.

**View (תצוגה):**

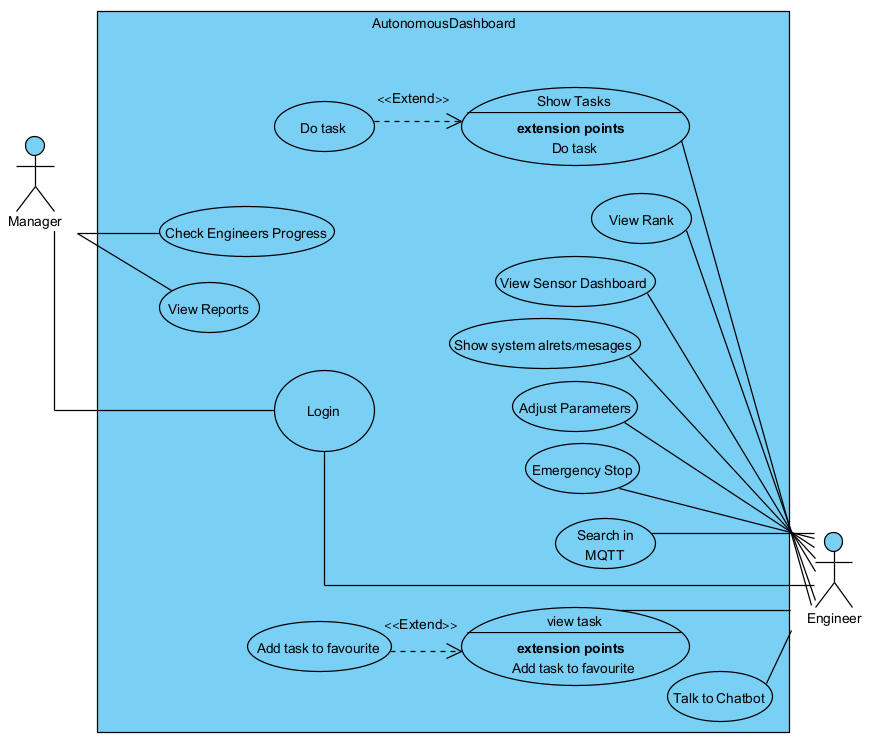
שכבת התצוגה כוללת את כל ה-HTML, CSS, ורכיבי JavaScript בצד הלקוח, הנוצרים דינמית על ידי Python ומוצגים בתאי הפלט של Colab. פונקציות ייעודיות בכל "עמוד" (כמו get\_full\_dashboard\_html ב-Cell 7, get\_tasks\_page\_html\_string ב-Cell 6) אחראיות על יצירת התצוגה הוויזואלית והאינטראקטיבית. main\_app\_output\_container משמש להצגת התצוגות.

**Controller (בקר):**

שכבת הבקרה מגשרת בין ה-View ל-Model. פונקציות ה-Callback ב-Cell 10 (כגון render\_dashboard\_view\_callback, handle\_finish\_task\_callback) הן ליבת הבקר בצד השרת, המופעלות על ידי אירועי JavaScript מה-View. הן מעבדות קלט משתמש, מתקשרות עם Firebase, ומורות על הצגת ה-View המתאים או עדכונו. גם לוגיקת ה-JavaScript בצד הלקוח המפעילה את ה-callbacks מהווה חלק מהבקר.

ארכיטקטורה זו מאפשרת הפרדה ברורה של אחריויות ופיתוח גמיש של כל רכיב במערכת.

**2.2 Use case מעודכן של פונקציונליות המערכת.**



**2.3 דרישות פונקציונליות**

1. המערכת מאפשרת למשתמשים להירשם עם שם משתמש, סיסמה, שם מלא ודוא"ל.
2. המערכת מאפשרת שליחת קוד אימות למייל בעת התחברות.
3. המערכת מאפשרת זיהוי סוג משתמש user או admin והצגת תפריט מותאם בהתאם.
4. המערכת מאפשרת למשתמש לצפות בנתוני חיישנים ממקום פנימי (indoor) או חיצוני (outdoor).
5. המערכת מאפשרת למשתמש לגשת לצ’אטבוט מובנה לקבלת עזרה.
6. המערכת מאפשרת הצגת הודעות שגיאה, הצלחה ומידע רלוונטי לאורך התהליך.
7. המערכת מאפשרת שמירת נתונים.
8. המערכת מאפשרת הצגת מידע היסטורי ממספר ימי עבר לצורך השוואת ביצועים.
9. המערכת מאפשרת למשתמש לשוחח עם צ’אטבוט לקבלת עזרה והכוונה במערכת.
10. המערכת מאפשרת ניתוח והצגת נתונים
11. המערכת תאפשר הצגת דוחות
12. המערכת מאפשרת חלוקת משימות למהנדסים
13. המערכת מאפשרת ביצוע משימה יומית
14. המערכת מאפשרת סינון הצגת המשימות

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

ביצועים (Performance):

המערכת מציגה נתונים ויזואליים במהירות גבוהה, גם כאשר נעשה שימוש במסדי נתונים בענן.

זמן התגובה בעת בחירת קטגוריה או בעת טעינת טבלת Leaderboard הוא מיידי ואפקטיבי.

זמינות (Availability):

השימוש ב־Firebase Realtime Database מאפשר זמינות גבוהה של המידע בכל זמן ומכל מקום.

הגישה דרך Google Colab מהענן מבטיחה עבודה רציפה וללא תלות במכשיר מקומי.

שמישות (Usability):

הממשק פשוט, אינטואיטיבי וכולל תפריטים נפתחים, כפתורים ברורים, טבלאות וגרפים קריאים.

הניווט בין עמודים וקטגוריות מתבצע בקלות וללא מאמץ מצד המשתמש.

תחזוקה (Maintainability):

הקוד מאורגן בפונקציות נפרדות וברורות כמו get\_weekly\_leaderboard ו־generate\_sensor\_dashboard, מה שמקל על תיקונים, עדכונים והרחבות.

ישנה הפרדה ברורה בין שליפה, עיבוד והצגת הנתונים – דבר שמאפשר תחזוקה מהירה ונוחה.

תלות בשירותים חיצוניים (Dependency on Services):

המערכת עושה שימוש ב־Firebase וב־Google Colab – מה שמספק יציבות, עדכניות ויכולות מתקדמות, ללא צורך בתשתית מקומית.

ניידות (Portability):

ניתן להפעיל את המערכת מכל דפדפן עם חיבור לחשבון Google, ללא צורך בהתקנה – מה שמאפשר שימוש מכל מקום ובכל זמן.

קריאות קוד (Code Readability):

שמות הפונקציות והמשתנים ברורים ומשקפים את תפקידם, מה שמקל על מפתחים להבין את הקוד ולבצע בו שינויים.

סקלביליות (Scalability):

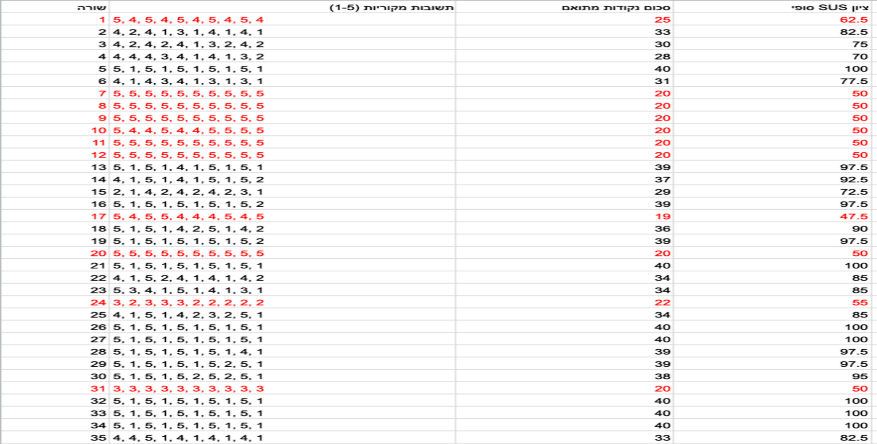
המערכת תומכת בהרחבה קלה של מספר המשתמשים, הקטגוריות והחיישנים – מבלי להשפיע על מבנה הנתונים הקיים

.**4.ביקורת עמיתים אשר ניתנה במהלך הסטודיו (שבוע 9) , וכיצד התמודדתם איתה. נא לחשב גם את ציון SUS ולהתייחס אליו.**  
עבור השאלות האי-זוגיות (1, 3, 5, 7, 9), יש להפחית 1 מהציון הניתן (ציון-1).

עבור השאלות הזוגיות (2, 4, 6, 8, 10), יש להפחית את הציון הנתון מ-5 (5-ציון).

נחשב את התוצאות של כל השאלות ונחבר אותן.

נכפיל את הסכום ב-2.5 כדי לקבל את הציון הסופי, מתוך 100.

*   
    
  הממוצעשלציוניה**-SUS הסופיים** הוא : **78.43**
* הממוצע של ציוני ה-SUS הסופיים, **לאחר החרגת השורות שבהן הציון הסופי היה נמוך מ- 70** **(הצבועות באדום)** ,הוא : **90.83**
* ציון ה־SUS שהתקבל מעיד שהמערכת נהנית מרמת שמישות גבוהה מאוד, עם ממוצע כללי של 78.43 המצביע על חוויית שימוש טובה, ואף מצוינת בקרב רוב המשתמשים. לאחר הסרת המשתמשים שחוו קושי (ציון נמוך מ־70), הממוצע מזנק ל־90.83 – נתון שמצביע על ממשק אינטואיטיבי, קל ללמידה ולתפעול, שאינו דורש הדרכה מוקדמת. כלומר, מרבית המשתמשים חוו את המערכת כנגישה, נוחה ויעילה – עדות לכך שהעיצוב והפונקציונליות תואמים היטב את צורכי קהל היעד.

**5. שקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?**  
1. תיאור ברור ליד כל ערך – כולל הסבר קצר ומשמעות של ערכים כמו טמפרטורה, לחות ולחץ.2. סימון ערכים חריגים – נתונים קריטיים מוצגים באדום עם אייקון התרעה.3. הצגת מקור הנתון – הבחנה בין חיישנים פנימיים (Indoor) לחיצוניים (Outdoor).4. כפתורים אינטואיטיביים – פעולות מלוות בכפתורים ברורים ותגובות ויזואליות להבהרת הפעולה.5. ויזואליזציה ברורה – גרפים להצגת מונחים, חיפושים ומועדפים בצורה פשוטה להבנה.

**.6 אתגרים אשר עלו במהלך העבודה, וכיצד התמודדתם איתם.**

* **נפח נתונים גדול:**  
  שמרנו רק מידע מעובד וחילקנו לפי ימים לשיפור ביצועים.
* **ויזואליזציה מורכבת:**  
  השתמשנו בצבעים, סינון וגרפים אינטראקטיביים להבהרת המידע.
* **ניתוקי חיישנים:**  
  הוספנו זיהוי ניתוק והצגת "N/A" במקום נתונים שגויים.
* **עומס מסך:**  
  עברנו לעיצוב מודולרי עם קוביות מידע נפרדות.

**7 . תיק תחזוקה – תיאור של כל הקבצים והאובייקטים המרכזיים, ותיעוד קצר של כל פונקציה בקוד.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **תא (Cell)** | **קובץ (שם ייצוגי)** | **מטרה עיקרית** | **פונקציות / מחלקות מרכזיות** | **תבנית עיצוב / קונספט מרכזי** |
| **1** | 01\_Setup\_Installation.sh | התקנת כל התלויות הנדרשות לפרויקט. | !pip install | **הגדרת סביבה נקייה:** שימוש ב---quiet להסתרת פלט טכני. |
| **2** | 02\_Chart\_Generation\_Service.py | שירות מרכזי ליצירת גרפים דינמיים. | ChartGenerationService | **יצירת תמונה בזיכרון (Base64):** המרת גרפים לטקסט להטמעה ישירה ב-HTML. |
| **3** | 03\_Search\_Service.py | מימוש מנוע חיפוש טקסט מבוסס אינדקס הפוך. | SearchService | **אינדקס הפוך ועיבוד שפה (NLP):** מימוש מנוע חיפוש יעיל עם ניקוי ונירמול טקסט. |
| **4** | 04\_Authentication\_Logic.py | ניהול תהליכי התחברות, הרשמה ואימות דו-שלבי. | UserAuthService, send\_verification\_email | **אימות דו-שלבי מבוסס זמן (2FA):** שליחת קוד למייל עם תאריך תפוגה לאבטחה מוגברת. |
| **5** | 05\_App\_UI\_Components.py | הגדרת רכיבי UI וסגנונות (CSS) משותפים. | get\_common\_app\_styles, get\_navbar\_html | **שיפור חווית משתמש (UX):** הצגת אנימציית טעינה בזמן פעולות רקע. |
| **6** | 06\_Firebase\_Data\_Access\_Layer.py | שכבת גישה לנתונים (DAL) לשליפת מידע מ-Firebase. | get\_all\_data\_for\_playback, get\_historical... | **שליפה מוקדמת לביצועים (Pre-fetching):** שליפה חד-פעמית של מידע לסימולציית עדכון חי. |
| **7** | 07\_Leaderboard\_Page\_Logic.py | בניית עמוד ה-Leaderboard, כולל טבלאות וגרפים. | get\_leaderboard\_page\_html\_content | **אגרגציית נתונים מורכבת בשרת:** חישוב מצטיין יומי הדורש עיבוד של כל המשתמשים. |
| **8** | 08\_Tasks\_Page\_Logic.py | בניית עמוד המשימות האינטראקטיבי. | get\_tasks\_page\_html\_string, JS: acceptChallenge | **ניהול מצב בצד לקוח (State Management):** סנכרון ממשק המשתמש עם השרת לאחר כל פעולה. |
| **9** | 09\_Dashboard\_Page\_Logic.py | בניית עמוד ה-Dashboard הראשי. | get\_full\_dashboard\_html, JS: updateCardsForPlayback | **יישום חוקים עסקיים בצד לקוח:** חישוב סטטוס "תקין/אזהרה" בדפדפן למשוב מיידי. |
| **10** | 10\_Admin\_Control\_Panel\_Logic.py | בניית פאנל ניהול עם אנליטיקות ועיבוד נתונים. | run\_mapreduce\_task\_difficulty\_analysis | **עיבוד נתונים עם MapReduce:** הדגמת פיצול בעיה גדולה לשלבי Map, Shuffle ו-Reduce. |
| **11** | 11\_Search\_UI.py | יצירת ממשק המשתמש של מנוע החיפוש. | display\_search\_engine\_ui | **חיבור UI-Backend:** קישור ווידג'טים של ipywidgets למחלקת שירות בפייתון. |
| **12** | 12\_Chatbot\_Logic.py | מימוש צ'אטבוט רב-שכבתי (NLTK, פונקציות, AI). | chatbot\_reply\_logic, handle\_special\_query | **שרשרת אחריות (Chain of Responsibility):** מתן מענה מדורג - מפונקציה ספציפית עד למודל AI כללי. |
| **13** | 13\_Application\_Entrypoint.py | נקודת הכניסה לאפליקציה, רישום כל ה-Callbacks. | render...view\_callback, handle...callback | **מערכת רישום Callback:** הגשר המרכזי בין אירועים ב-JavaScript לפונקציות בפייתון. |

**8. תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.**

### **תיק משתמש למערכת AutoLine**

#### **? AutoLine הקדמה: מהי מערכת**

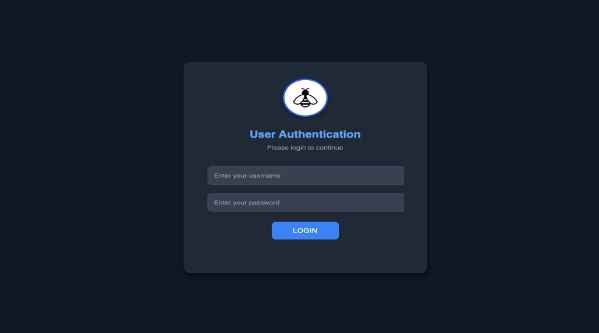
מערכת זו נועדה לספק תמונת מצב מלאה וכלים לניהול קו הייצור האוטונומי. היא מאפשרת לכם, כמהנדסים ומנהלים, לעקוב אחר תקינות המערכת בזמן אמת, לבצע משימות אופטימיזציה, ולעקוב אחר ביצועים ותפוקה. המערכת מחולקת למספר מסכים, כל אחד עם מטרה ייעודית, כדי להפוך את עבודתכם ליעילה ופשוטה יותר.

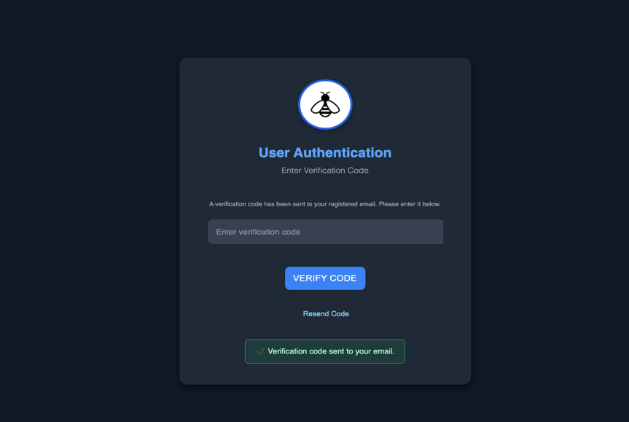
**הממשק יותר ברור בפועל, המסכים נראים קטנים או אי נוחים בגלל צילומי המסך**

#### **2. פירוט המסכים המרכזיים**

##### **Login Screen א. מסך התחברות**

* **מה רואים במסך?** זהו שער הכניסה המאובטח למערכת. תתבקשו להזין שם משתמש וסיסמה. המערכת מוודאת שהקלט שהוזן תקין לפני שהיא ממשיכה בתהליך.



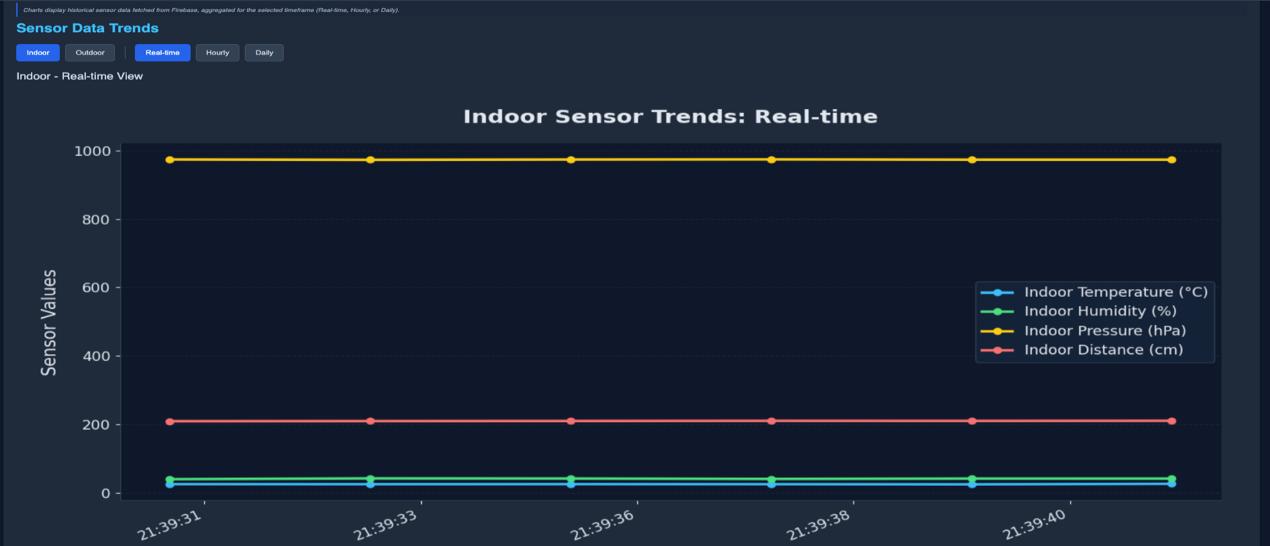
* **מה אפשר לעשות?**
  + להזין פרטי התחברות.
  + לאחר הזנת פרטים נכונים, המערכת שולחת **קוד אימות חד-פעמי** למייל שלכם.
  + יש להזין את הקוד שהתקבל במייל כדי להשלים את הכניסה.
  + אם הקוד לא הגיע או שפג תוקפו, ניתן לבקש קוד חדש באמצעות כפתור "Resend Code".

##### **Dashboardב. לוח דשבורד**

* **מה רואים במסך?** זהו המסך הראשי והמרכזי של המערכת. הוא מציג תמונת מצב חיה של קו הייצור באמצעות "כרטיסיות חיישנים". כל כרטיסייה מציגה ערך עדכני מחיישן אחר (טמפרטורה, לחות, מרחק וכו'), ומסמנת את מצבו בצבע (ירוק – תקין, צהוב – אזהרה, אדום – חריגה). בנוסף, מוצגים גרפים המראים את מגמת הנתונים לאורך זמן.



* **מה אפשר לעשות?**
  + לצפות בנתוני החיישנים המתעדכנים כל מספר שניות.
  + לעבור בין תצוגת חיישני **פנים** (Indoor) לחיישני **חוץ** (Outdoor).
  + לשנות את טווח הזמן של הגרפים כדי לראות נתונים היסטוריים (זמן אמת, שעתי, יומי).



##### **Leaderboard ג. טבלת מובילים**

* **מה רואים במסך?** מסך זה מציג את דירוג המהנדסים במערכת על בסיס הביצועים שלהם. למשתמש רגיל יוצגו הנתונים האישיים שלו (ניקוד כללי, מספר משימות שביצע, והדירוג הנוכחי). למנהל יוצגו נתונים כלליים על ביצועי העובד המצטיין בכל יום.

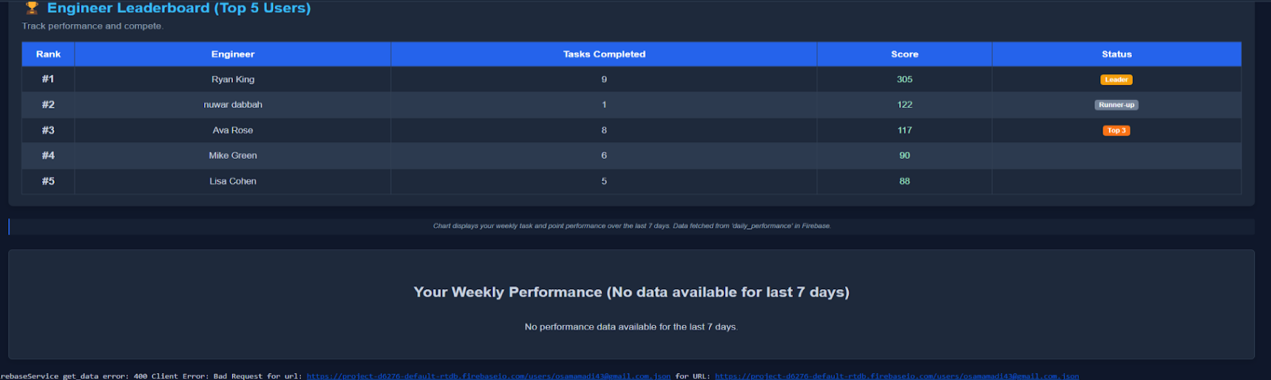
**🔹 מבנה העמוד:**

1. **טבלת דירוג חכמה – Top 5 Users:**

מציגה את חמשת המהנדסים המובילים לפי ניקוד כולל (Score).

כוללת מידע נוסף כמו מספר משימות שבוצעו (Tasks Completed) ומיקום בדירוג (Rank).

תגים צבעוניים לפי מיקום:  
 🥇 Leader – מקום ראשון  
 🥈 Runner-up – מקום שני  
 🥉 Top 3 – מקום שלישי

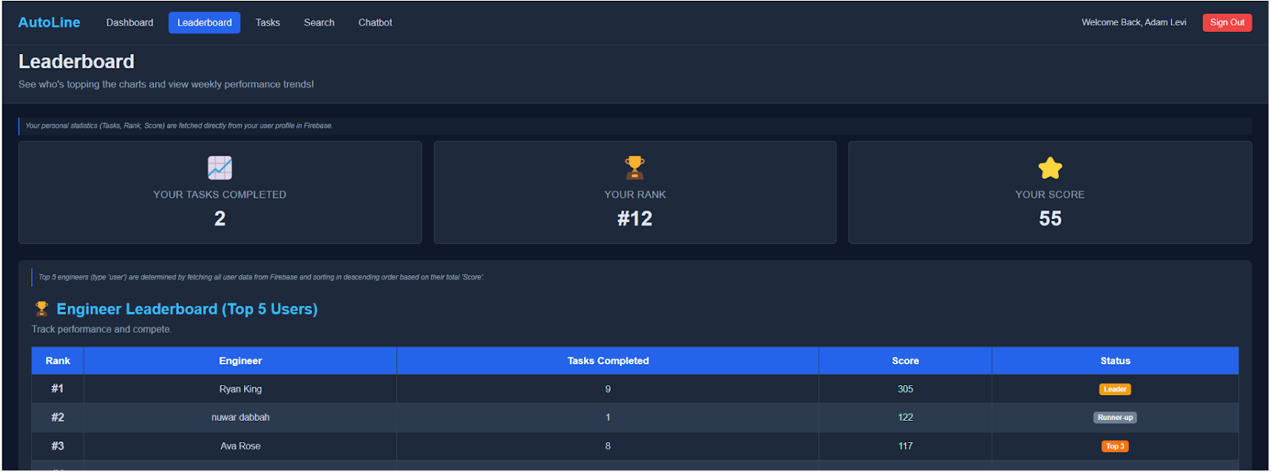
****

1. **כרטיסי סיכום אישיים (למשתמש רגיל בלבד):**

מוצגים בראש העמוד כשמשתמש רגיל מחובר.

מציגים:

* + - מספר משימות שביצע.
    - הדירוג הכללי שלו.
    - הניקוד האישי שלו.

****

1. **גרף ביצועים שבועי (Weekly Performance):**

עבור משתמש רגיל: מציג את מספר המשימות והנקודות שהושגו בכל אחד משבעת הימים האחרונים.

עבור מנהל (Admin): מציג את הביצועים של "מהנדס היום" (Top Engineer of the Day) עבור כל יום בשבוע, כולל טבלת סיכום טרנספוזיציונית עם שם המהנדס המוביל**.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **הצגת הודעות במידה ואין נתונים:**

למשל: "No performance data available for the last 7 days" – מוצג במקרה ואין פעילות לאחרונה.

**🔹 שימוש בטכנולוגיות:**

* הנתונים נשלפים בזמן אמת מ־Firebase Realtime Database.
* העיבוד והעיצוב מתבצעים באמצעות Python, pandas, HTML ו־CSS מותאם אישית (מצב כהה).
* הגרפים נוצרים באמצעות ChartGenerationService מותאם.
* מאפשר למהנדסים להבין את מיקומם היחסי במערכת.
* עוזר למנהל לזהות מהנדסים מובילים לפי ימים ולפי ביצועים מצטברים.
* מציג את הדירוג והסטטוס של כל אחד – בצורה ויזואלית, אינטואיטיבית ונעימה לעין.

##### **TASKS** **ד. משימות (עבור מהנדסים בלבד )** -

* **מה רואים במסך?** מסך זה מרכז את כל משימות האופטימיזציה והתחזוקה שיש לבצע בקו הייצור. הוא מחולק לרשימת משימות זמינות, רשימת המשימות שכבר לקחת על עצמך ("משימות בביצוע"), ורשימת מועדפים.
* **תיאור המסך:**
* כותרת עליונה
* סינון לפי קושי:   
  כפתורים לבחירת רמת קושי: All, Easy, Medium, Hard.  
  ברירת המחדל היא "All".
* רשימת משימות זמינות: כל משימה מוצגת ככרטיס מידע וכוללת:  
  שם המשימה (למשל: "Basic Temperature Check"  
  תיאור קצר  
  אייקון קושי (לדוג' Easy)  
  מספר נקודות שניתן להרוויח  
  כפתור Accept Challenge  
  כפתור Add to Favourites (סימן כוכב)

**סטטיסטיקות (צד ימין):** מציגות נתונים על המשימות שהמהנדס בצע , כמה ניקוד צבר , המשימה עם הציון הכי גבוה

**Ongoing Tasks :** מציג את המשימות שהמשתמש כבר קיבל אך טרם סיים.  
אם אין משימות פעילות: מוצגת הודעה "No tasks accepted yet".

**Favourites :** מציג משימות שסומנו כמועדפות.  
אם אין: "No active tasks in favourites".

לחיצה על Accept Challenge , המשימה מועברת ל-Ongoing Tasks.

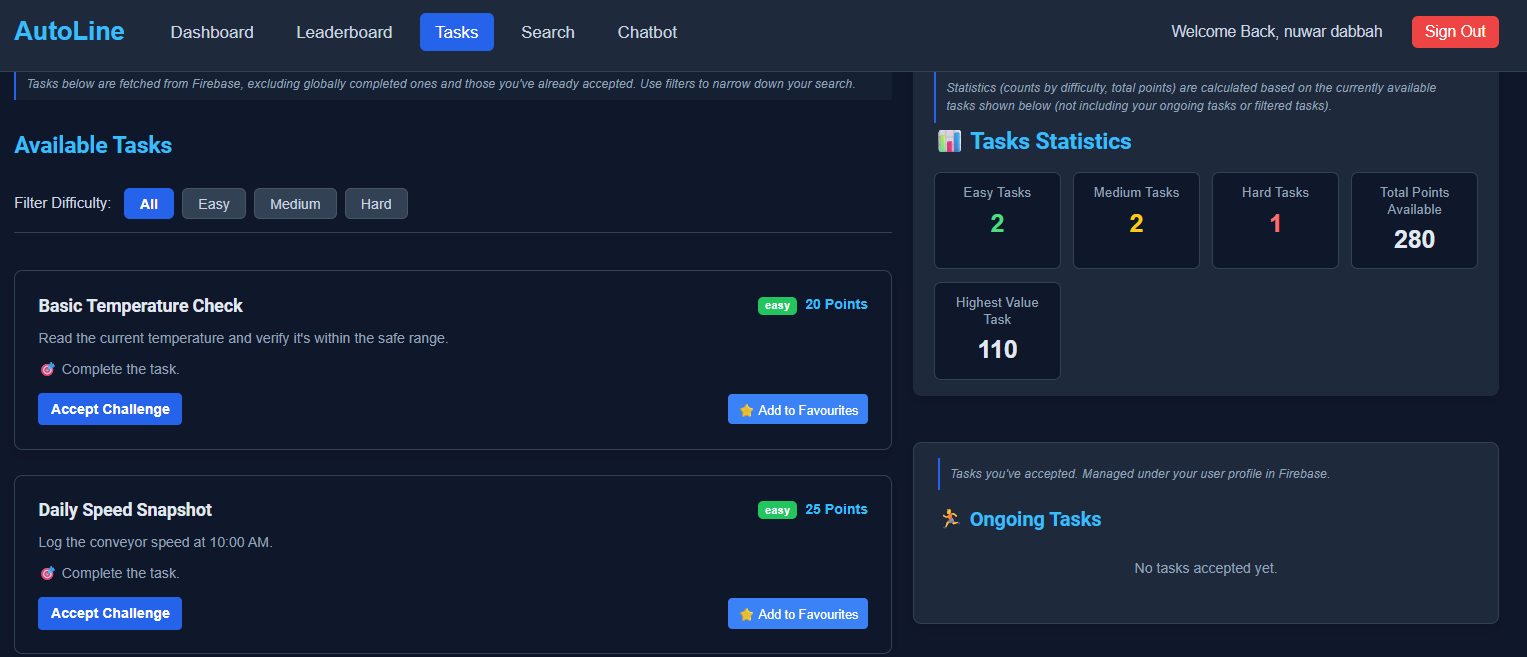
לחיצה על Add to Favourites, המשימה מופיעה ברשימת המועדפים.

סינון לפי קושי, מציג רק משימות בהתאם לרמת הקושי שנבחרה.

המשימות שהושלמו לא מוצגות במסך זה.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.הניקוד מעודד ביצוע משימות בצורה עקבית ושיטתית.



##### **פאנל ניהול(למנהלים בלבד) Control Panel**ה.

* **תמונה שמכילה צילום מסך, טקסט, תוכנה, תכונות מולטימדיה

  תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.מה רואים במסך?**   
  בחלק זה של מערכת הניהול יצרתי טבלה אינטראקטיבית המציגה את כלל המשתמשים מסוג "user" (מהנדסים) מתוך מסד הנתונים של Firebase. הטבלה כוללת מידע כמו שם, שם משתמש, ניקוד (Score), מספר משימות שבוצעו, וסוג המשתמש. בנוסף, הוספתי כפתורי סינון חכמים המאפשרים למנהל לסנן את הטבלה לפי סף ניקוד: "כולם", "מעל 0", "מעל 50", "מעל 100". עיצוב הטבלה בוצע בסגנון מודרני כהה, כולל שורות מודגשות לסריקה נוחה, ואפקטים של ריחוף (hover) להבהרת שורות נבחרות.
* **סטטוס השלמת משימות – תרשימי עוגה לפי רמת קושי**

בפרויקט יצרתי הדמיה גרפית של סטטוס המשימות בעזרת תרשימי עוגה, המחולקים לפי רמות קושי: קל, בינוני וקשה. כל תרשים מציג את אחוז המשימות שבוצעו מול אלו שעדיין לא הושלמו, כשהצבעים נבחרו לפי הקטגוריה (ירוק לקל, כתום לבינוני, אדום לקשה). הנתונים נשלפים ממסד Firebase בזמן אמת ומעובדים באמצעות Python וספריית Matplotlib. התרשימים מוטמעים בעמוד הניהול באופן דינמי ומעוצבים באופן שמבליט הבנה ויזואלית מהירה של מצב המשימות במערכת.

* **ניתוח משימות לפי רמת קושי (MapReduce)**

**תמונה שמכילה טקסט, צילום מסך, תרשים, גופן

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**במערכת הניהול יישמתי סימולציה של תהליך MapReduce לצורך ניתוח כמותי של משימות לפי רמות קושי. בשלב ה־Map, כל משימה "פולטת" את רמת הקושי שלה עם הערך 1; בשלב ה־Reduce, מתבצעת סכימה של כלל המשימות עבור כל רמת קושי (easy, medium, hard). התוצאה מוצגת בטבלה המסכמת את מספר המשימות בכל קטגוריה. מטרת הניתוח היא לספק למנהל תמונת מצב ברורה של פיזור המשימות ולהבין היכן מרוכז העומס או הקושי במערכת.

**תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**

##### **Search ו.חיפוש**

# **מאפשרת למשתמשים למצוא מידע בדפי התיעוד המרכזיים של MQTT.org. היא עושה שימוש באינדקס הפוך (inverted index) בנוי מראש כדי לאחזר במהירות תוצאות רלוונטיות, אשר מדורגות לאחר מכן לפי רלוונטיות.**

# **זהו תיאור של מנגנון חיפוש תיעוד MQTT. הוא פועל בשני שלבים עיקריים: אינדוקס, שבו תוכן נבחר מ-MQTT.org נאסף, מנוקה, מעובד למילים בסיסיות ונשמר כאינדקס הפוך ב-Firebase; ושאילתה, שבה שאלת המשתמש מעובדת באופן דומה, משמשת לאיתור מסמכים רלוונטיים מהאינדקס, ולדירוג התוצאות לפי מספר מונחי החיפוש המשותפים. השימוש פשוט: מזינים מילות מפתח בלשונית "Search", לוחצים על "Search", ומקבלים רשימת תוצאות מדורגות עם ציוני רלוונטיות.**

בדוגמה הספציפית הזו, התוצאות הן:

* Faq (שאלות נפוצות):
  + כתובת URL: https://mqtt.org/faq/
  + ציון רלוונטיות: 3 (זהו הציון הגבוה ביותר, המעיד על כך שהוא מכיל את המונחים הרבים ביותר התואמים מתוך "iot standard worth").
* Mqtt.Org:
  + תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

    תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.כתובת URL: https://mqtt.org/
  + ציון רלוונטיות: 2
* Software (תוכנה):
  + כתובת URL: https://mqtt.org/software/
  + ציון רלוונטיות: 2

**(Chatbot)צ'אטבוט ז.**

* **מה רואים במסך?** חלון שיחה עם עוזר וירטואלי (בוט).
* **מה אפשר לעשות?**
  + לשאול שאלות בשפה חופשית על תפעול המערכת (למשל, "איפה אני רואה את המשימות שלי?", "מי העובד המצטיין?").
  + תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

    תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.להשתמש בכפתורי ההצעות כדי לשאול שאלות נפוצות.

**9. משובים :**

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| הצבעים בהירים מדי | כן | התייחסנו לנושא הצבעים על ידי בחירת פלטת צבעים מאוזנת הכוללת גוונים כהים וניגודיות מספקת, כך שנשמרת נוחות ראייה ונגישות טובה למשתמשים בכל תנאי תאורה. |
| עיצוב לידבורד | כן | נעשו שיפורים בעיצוב הלוח כדי להתאימו לשאר דפי המערכת מבחינת מראה ועיצוב: שימוש בצבעים מותאמים, כותרות ברורות, תגיות סטטוס עם עיצוב ויזואלי אחיד, וסידור נתונים בטבלאות עם עיצוב אחיד ונגיש, מה שמגביר את בהירות הלוח והופך אותו לאינטואיטיבי יותר למשתמשים. |
| לוותר על האופציה של להציג כשמספר המשימות גדול מ-0 | לא | הצגת מידע גם כשהמשימות מעל 0 היא חלק מהשקיפות למשתמש, ומסייעת למעקב ובקרה. יש לבדוק את ההצעה לעומק לפני שינוי כזה, כי ייתכן שזה מפחית מידע חשוב למשתמש. |
| יש להוסיף גרפים מסוג קווים | לא | מערכת כוללת כבר גרפים מובנים ומותאמים לפי הצורך. הוספת גרפי קווים צריכה להיעשות רק אם יש דרישה פונקציונלית ברורה, כרגע לא נדרש שינוי זה. |
| יותר הסבר עבר המשתמש הפשוט מה קורה | לא | במערכת קיימת שקיפות אלגוריתמית ברורה הכוללת תיאורים ליד ערכי החיישנים, סימוני צבעים לערכים חריגים, הדגשת מקור הנתונים, כפתורים ברורים עם תגובות ויזואליות וגרפים מובנים. בנוסף, בכל דף במערכת יש הסברים וטקסטים שמבהירים את תפקיד הדף ותהליכי העבודה בו, המסייעים למשתמש להבין טוב יותר את המידע והפעולות. |

**11.**

|  |  |
| --- | --- |
| **קריטריון** | **הסבר** |
| מימוש | האם הקוד מבצע את הנדרש? פרטו.  תשובה: כן, הקוד מבצע את הכי מינימום שנדרש , כלומר נטו פונקציונאלי |
| יעילות | האם ניתן לשפר את יעילות הקוד?כיצד? תשובה:  לא, מבחינה פונקציונלית הקוד פשוט ויעיל ככל האפשר |
| פשטות | האם ניתן לפשט את הקוד?כיצד?  לא , הקוד מאוד פשוט |
| מודולריות | האם הקוד מודולרי מספיק? (חלוקה תקינה לקבצים, פונקציות ומחלקות)  כן החלוקה ברורה בתוך כל cell אך רק צריך לתת לcells שמות משמעותיים, למשל בעזרת @title לדפדוף יותר מהיר בקוד בקולאב |
| באגים וטעויות | האם ישנם מקרים בהם הקוד לא מתנהג כצפוי? אילו?  לא, אין באגים, הכל עובד תקין |
| 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?  טיפול בשגיאות הוא טוב ונראה במסך ה LOGIN כאשר מזינים קלט שגוי,וגם print statements נמצאים לדיבוג בקוד |
| בדיקות | האם יש בדיקות שניתן להוסיף? לא שידוע לנו |
| שימושיות | האם הממשק שימושי? (usability)? האם המערכת מעוררת רצון להמשיך להשתמש בה?  לא , חוסר עיצוב , חוסר אינטראקציה עם המערכת לא מעודדת המשך להשתמש בה באף אופן |
| תיעוד | האם הקוד מתועד?הקוד מתועד היטב . |
| 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?  יש שקיפות מילולית בממשק לכל tab מה יש ומציג למשתמש, אין algorithmic bias הכל מנותח לרוב הנתונים |
| אבטחה | האם ישנו מידע אבטחה (שמות משתמש, ססמאות) גלויים?  הכל נמצא במסד הנתונים וזה מצויין ! |
| ביצועים | האם שינוי עתידי בקוד יכול לגרום לפגיעה בביצועים?כיצד ניתן לשפר? בגדול הכל בסדר אבל הנתונים של החיישנים מצריכים זמן רב בכדי לטעון |
| קריאות - readability | האם הקוד מובן בקלות?אילו חלקים בקוד היו לא ברורים עבורך?כיצד ניתן לשפר את קריאות הקוד?   כן הקוד קריא , אפשר להוסיף section לכל cell |

התייחסות למשוב שקיבלנו במסגרת code review בשבוע 13:

הצוות השני ביצע בדיקה מקיפה והמשוב כולו היה חיובי.

* **שימושיות ושמישות:** ממשק ברור וקל להבנה.
* **בדיקות:** קיימות ותקינות.
* **תיעוד וקריאות:** הקוד מתועד, קריא ומובנה היטב.
* **אמינות והיעדר הטיות:** הקוד מתנהג כמצופה, ללא בעיות של אפליה אלגוריתמית.
* **קריטריונים טכניים:** ציון לשבח על פשטות, יעילות, מודולריות ועמידות.

**סיכום:** לא בוצעו שינויים – כל ההערות היו חיוביות בלבד.

**13. מקורות.**

ברוב שלבי הפרויקט נעזרנו בכלי בינה מלאכותית מתקדמים, כגון ChatGPT ו-Gemini AI, במטרה לשפר ולייעל את ממשק המשתמש ועיצוב המערכת. כלים אלו סייעו לנו בעיקר ביצירת קוד ברור יותר, הצעות לעיצוב מודרני ופתרונות טכניים שונים לשיפור חוויית המשתמש.

במהלך הפיתוח, עבדנו רבות עם ספריית ipywidgets ב-Python, אשר מאפשרת יצירת ממשקים אינטראקטיביים בסביבת Jupyter Notebook ו-Google Colab. עם זאת, נתקלו במספר אתגרים טכניים, כולל באגים ובעיות שבהן שינויים בקוד העיצוב לא השתקפו בממשק המשתמש כפי שציפינו. בעיות אלו גרמו לנו לבחון דרכי עבודה אלטרנטיביות, כמו שימוש ברכיבי HTML לצורך עיצוב גמיש ומדויק יותר.

בנוסף, לצורך הבנה מעמיקה יותר של שפת Python ושל סביבת הפיתוח בה אנו עובדים, הסתמכנו על מקורות לימוד מקוונים אמינים ומעודכנים. בין המקורות המרכזיים היו אתרי לימוד פופולריים כמו w3schools, שמציעים הסברים ברורים, דוגמאות קוד ותרגולים, וספרות מקצועית המתעדכנת תדיר, אשר סייעו לנו להתמודד עם בעיות וללמוד בצורה ממוקדת ויעילה.

השילוב בין כלי AI מתקדמים לבין למידה מתמדת ממקורות מהימנים אפשר לנו לפתח מערכת בעלת ממשק משתמש איכותי, אמין ונגיש.

<https://www.w3schools.com/python/>