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

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

מועד הגשה: 19.6.25

המשימה בתרגיל זה: סיום בניית המערכת, - איטרצית פיתוח מס 2

**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)

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

שימו לב: למטלה זו שלושה חלקים

**חלק ראשון.**(10 נקודות).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** | **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 |

***חלק שני: בניית המערכת*** (70 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 להוסיף ממשק כלשהו של משחוק כרצונכם.

יש לוודא שניתן להריץ את הפרויקט מהמחברת בלבד , ללא כל צורך בהטענה של קבצים נוספים או יציאה למערכת חיצונית (כגון FLASK). ניתן לעשות שימוש ב gradio/

1. שימוש ב -microservices (תרגול 7)- עליכם לעשות שימוש לפחות בשניים. פרטו באילו שירותים עשיתם שימוש, ומה היתרון בשימוש בשירותים אלו.(10 נקודות)

בפרויקט נעשה שימוש בארכיטקטורה מודולרית המדמה Microservices. שני השירותים המרכזיים הם:

### ****(1**** ChartGenerationService ****(שירות יצירת גרפים)****

שירות זה אחראי באופן בלעדי על יצירת כל הגרפים באפליקציה (קו, עמודות ועוגה) והמרתם לתמונה.

**יתרונותיו:**

* **שימוש חוזר (Reusability):** אותו קוד משמש ליצירת גרפים במספר עמודים שונים (Dashboard, Leaderboard, Control Panel) ומונע שכפול.
* **תחזוקתיות גבוהה (Maintainability):** שינוי עיצוב של כל הגרפים באפליקציה דורש עריכה במקום אחד בלבד.
* **הפרדת אחריויות (Decoupling):** לוגיקת יצירת הגרף מופרדת לחלוטין מהעמודים המציגים אותה.

### ****(2**** SearchService ****(שירות חיפוש)****

שירות זה מתפקד כמנוע חיפוש עצמאי, האחראי על בניית אינדקס ממקורות מידע וביצוע שאילתות חיפוש עליו.

**יתרונותיו:**

* **מודולריות ואנקפסולציה:** כל הלוגיקה המורכבת של החיפוש (אינדוקס, עיבוד שפה) "ארוזה" בתוך יחידה עצמאית, המסתירה את המורכבות משאר האפליקציה.
* **קוד נקי:** ממשק המשתמש של החיפוש לא עוסק בלוגיקה, אלא רק קורא לפונקציה פשוטה של השירות (query\_index), מה ששומר על הקוד הראשי פשוט וברור.

1. צטבוט שישולב במערכת מבוסס AI (כפי שלמדתם בתרגול 8), העונה בצורה אינטליגנטית לשאלות המשתמש. (15 נקודות).
2. מהם ה -KPI הרלוונטיים בפרויקט שלכם? הסבירו? (10 נקודות)  
     
   **1. זמן עיבוד ממוצע לנתון חיישן (End-to-End Data Processing Time)**

**מה מודד:**  
הזמן הכולל מרגע קבלת המידע מה־API ועד שהוא מוצג על הדשבורד.

**יעד:**  
פחות מ־2 שניות (רצוי מתחת לשנייה בנתונים קריטיים).  
**נוסחה:**זמן קבלת הנתון – זמן הצגה

**חשיבות:**  
מדד קריטי לביצועים בזמן אמת, בעיקר אם מהנדסים צריכים להגיב לשינויים מהירים.

**2.** **שיעור הצלחה בשליפת מידע מה־API (API Data Retrieval Success Rate)**

**מה מודד:**  
כמה מתוך קריאות ה־API הסתיימו בהצלחה (200 OK) ולא בשגיאה (404 וכו').

**יעד:**  
מעל 99% הצלחה.  
**נוסחה:**100 x קריאות מוצלחות / סכ"ה קריאות.

**חשיבות:**  
בעיות בשליפה עלולות להשאיר את המערכת בלי מידע עדכני – ולפגוע באמינות.

**3.** **דיוק בזיהוי חריגות (Critical Value Detection Accuracy)**

**מה מודד:**

עד כמה המערכת מדויקת בזיהוי נתונים חריגים (ללא false alarms או פספוסים).

**יעד:**

דיוק מעל 95%.

**נוסחה:**  
מספר המקרים שבהם המערכת זיהתה נכון ערך חורג, מתוך כלל המקרים שבהם הייתה אמורה לזהות או זיהתה בטעות.

**חשיבות:**

התרעות שווא פוגעות באמון המהנדסים במערכת; פספוס של ערכים חריגים מסוכן תפעולית.

**4. שביעות רצון המשתמש (User Satisfaction)**

**מה מודד:** מדד זה נמדד באמצעות סקר **SUS (System Usability Scale)** או מדד דומה, כאשר משתמשים מדרגים את המערכת על בסיס נוחות, יעילות ואינטואיטיביות.  
 **יעד**:  
 ציון SUS מעל **67**, המעיד על חוויית שימוש מצוינת.  
**נוסחה**:

סקר בן 10 שאלות, כל אחת מדורגת מ 1 עד 5 , ציון מנורמל ל־100.  
 **חשיבות**:  
 מדד זה מספק משוב ישיר מהמשתמשים ומסייע בזיהוי נקודות לשיפור בממשק ובפונקציונליות.

\* מדדי KPI אלו מספקים תמונה כמותית ומדויקת של ביצועי המערכת, זמינותה – מה שמאפשר לנו לשפר את המערכת באופן מתמיד ולהבטיח שהיא נותרת יעילה, אמינה ונגישה.

1. שקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם? (5 נקודות)

במערכת שלנו דאגנו לשקיפות מלאה על ידי הצגת הנתונים והחישובים בצורה נגישה, ויזואלית ומובנת:

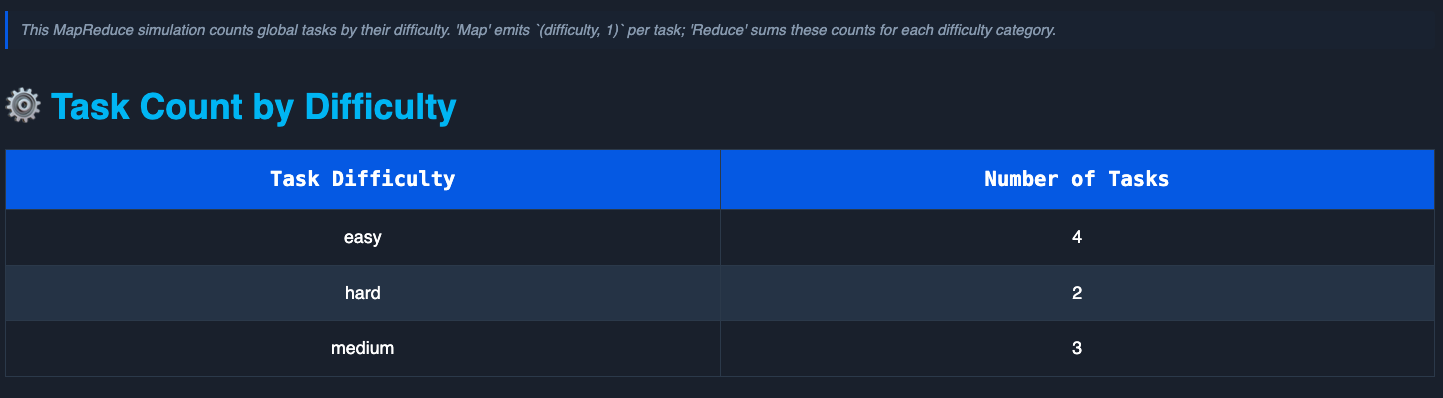
1. **הצגת שמות החיישנים והסברים ליד כל ערך** – כל נתון שמוצג בדשבורד (כמו טמפרטורה, לחות, לחץ וכו') כולל תיאור קצר בצמוד, שמבהיר מה המשמעות שלו ומה נחשב ערך תקין.

2. **סימון ויזואלי לערכים חריגים** – ברגע שהמערכת מזהה ערך קריטי (למשל טמפרטורה חריגה), הנתון מוצג בצבע אדום עם אייקון התרעה ברור, כך שהמשתמש מבין מיידית שמדובר בבעיה.  
3. **הדגשה של מקור הנתון** – בכל רכיב של הנתונים מצוין האם הוא מקורו ב־ Indoor ו Outdoor Sensors כדי שהמהנדסים יידעו מה ההקשר.

4. **כפתורים ברורים עם תוויות מדויקות ותגובות ויזואליות** – כל פעולה שהמשתמש מבצע, כמו 'שמור', 'סכם', או 'עדכן', מוצגת עם כפתורים ברורים ותגובות כמו שינוי צבע או גדילה כאשר מצביעים עליהם, כדי להסביר למשתמש את מהות הפעולה.

5. **הצגת נתונים בצורה גרפית ומובנת** – גרפים שמוצגים במערכת מציגים למשתמשים מידע על חיפושים נפוצים, מונחים שסוכמו, ומועדפים בצורה קלה להבנה.

1. מידע עתק: השתמשו במודלים שהוצגו בשבוע 9 ובצעו ניתוח על המידע שלכם. הציגו גרף נתונים של חיתוך לבחירתכם



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.

1. אתגרים שעלו במהלך העבודה , וכיצד התמודדתם איתם (לדוגמא- DB גדול מדי, יצירת ויזואליזציה, מבנה הקוד וכו).

**1. אתגר: נפח נתונים גבוה ב DB**  
ככל שהמערכת קיבלה יותר נתונים מחיישנים בזמן אמת, מסד הנתונים ב־Firebase הלך וגדל, מה שהוביל לעיכובים בביצועים ושליפות איטיות.

**פתרון:**  
במקום לשמור את כל הנתונים הגולמיים, בחרנו לשמור רק את המידע הרלוונטי והמעובד (כמו ערכים קריטיים או ממוצעים תקופתיים). בנוסף, חילקנו את המידע לפי ימים בתתי קבוצות כדי לארגן את הנתונים בצורה נוחה ונגישה יותר.

**2. אתגר: יצירת ויזואליזציה ברורה למשתמש**  
הצגת המידע באופן מובן ולא עמוס הייתה מורכבת, במיוחד כשיש מספר חיישנים וסוגי נתונים במקביל (טמפ', לחות, תנועה וכו').

**פתרון:**  
יצרנו דשבורד מבוסס צבעים וסינונים – לדוגמה, נתונים קריטיים בצבע אדום, חיתוך לפי סוג חיישן/תאריך, וגרפים אינטראקטיביים (bar/line chart) שאפשר לנתח דרכם מגמות.  
**3**. אתגר: הפסקות חשמל שהשביתו את החיישנים  
לפעמים הייתה הפסקת חשמל באתר שבו ממוקמים החיישנים – וזה פשוט ניתק אותם. התוצאה: במשך שעות לא התקבלו נתונים, והמערכת הציגה "שקט" מזויף.

**פתרון:**  
הוספנו מנגנון זיהוי ניתוק – אם לא התקבל מידע מהחיישן בפרק זמן מסוים (למשל 10 דקות), המערכת ומציגים: "N/A". כך המשתמש מבין שלא מדובר בערכים רגילים, אלא בהפסקה זמנית.

**4.** **אתגר: ניסינו להכניס הכל למסך אחד**בהתחלה רצינו שהמשתמש יראה את *כל* הנתונים במבט אחד. התוצאה? כאוס מוחלט.

**פתרון:**עברנו לעיצוב מודולרי: כל סוג נתונים בקובייה מסודרת עם כותרת וצבע קבוע. הדשבורד הפך לנעים, נוח, ואפילו קצת כיף לגלול בו.

***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.

### תיק למתכנת ל****פרויקט AutoLine****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. |

1. יש לבנות תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.
2. התייחסו בתיק המשתמש לשקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?

### **תיק משתמש למערכת **AutoLine****

#### **1. הקדמה: מהי מערכת **AutoLine**?**

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.

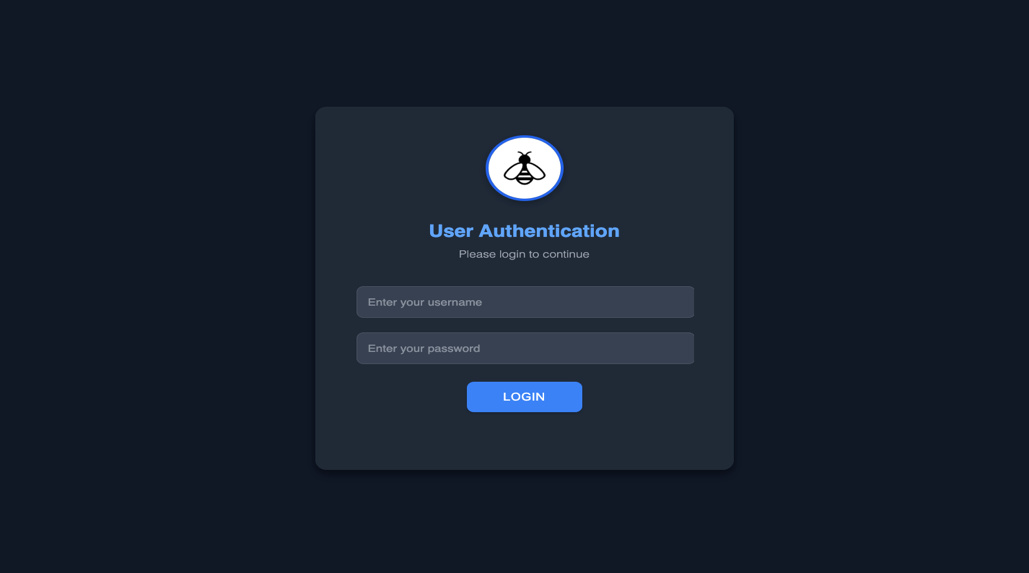
המערכת מחולקת למספר מסכים, כל אחד עם מטרה ייעודית, כדי להפוך את עבודתכם ליעילה ופשוטה יותר.

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

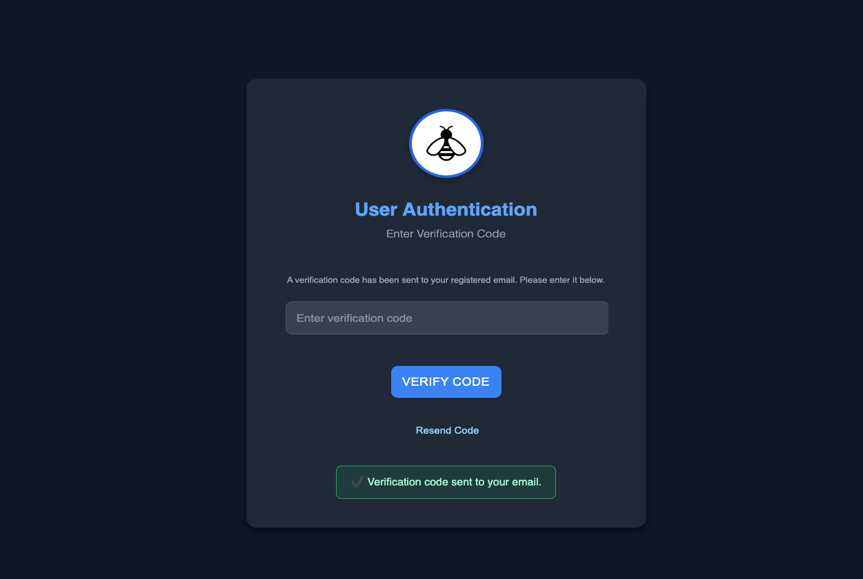
#### **2. פירוט המסכים המרכזיים**

##### **א. מסך התחברות (**Login** **Screen**)**

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

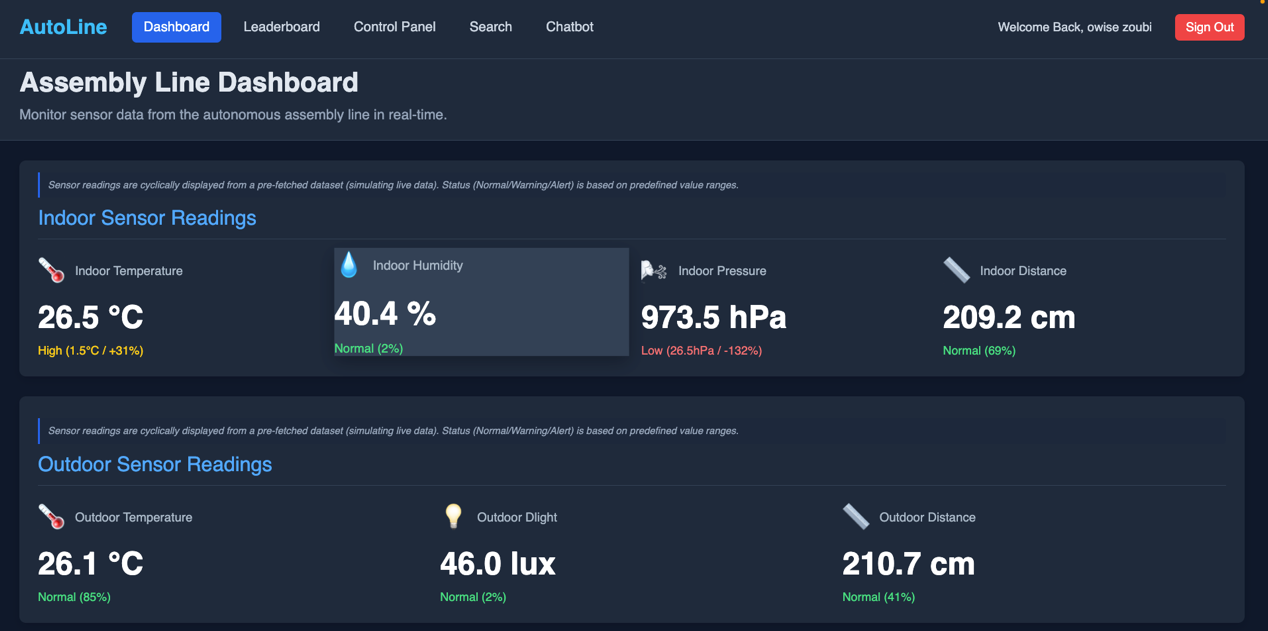


* **מה אפשר לעשות?**
  + להזין פרטי התחברות.
  + לאחר הזנת פרטים נכונים, המערכת שולחת **קוד אימות חד-פעמי** למייל שלכם.



* + 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".

##### **ב. לוח דשבורד (**Dashboard**)**

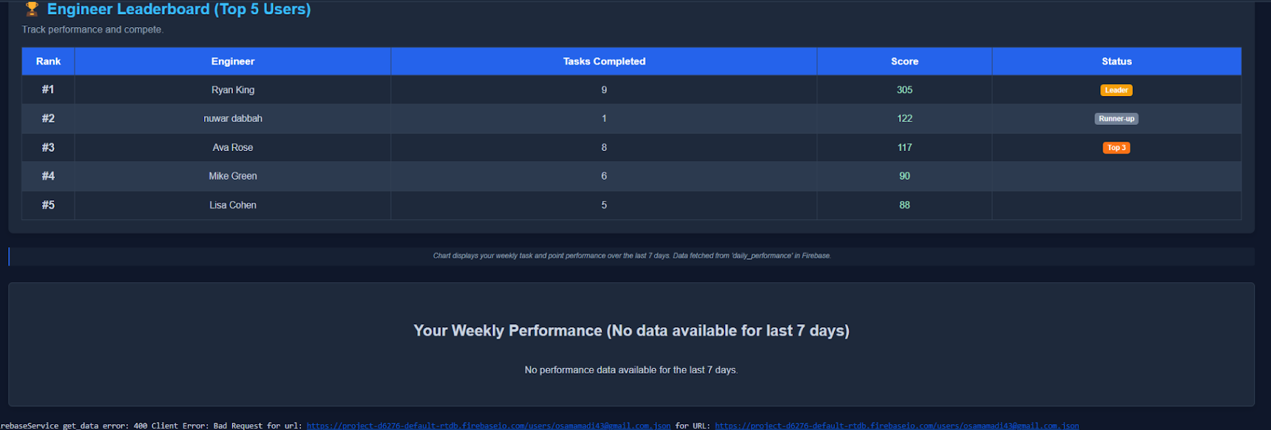
* **מה רואים במסך?** זהו המסך הראשי והמרכזי של המערכת. הוא מציג תמונת מצב חיה של קו הייצור באמצעות "כרטיסיות חיישנים". כל כרטיסייה מציגה ערך עדכני מחיישן אחר (טמפרטורה, לחות, מרחק וכו'), ומסמנת את מצבו בצבע (ירוק – תקין, צהוב – אזהרה, אדום – חריגה). בנוסף, מוצגים גרפים המראים את מגמת הנתונים לאורך זמן.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

##### **ג. טבלת מובילים (**Leaderboard**)**

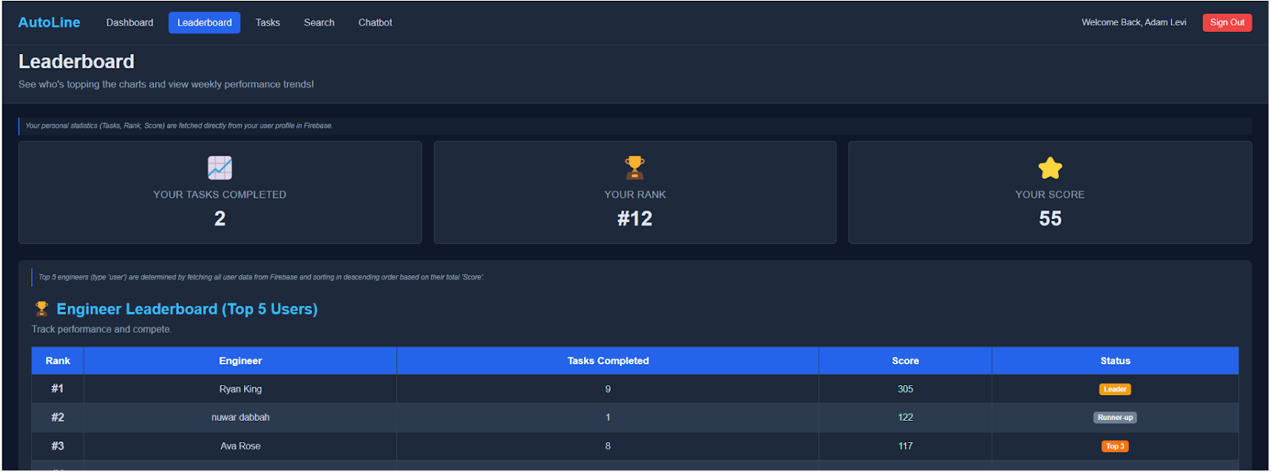
* **מה רואים במסך?** מסך זה מציג את דירוג המהנדסים במערכת על בסיס הביצועים שלהם. למשתמש רגיל יוצגו הנתונים האישיים שלו (ניקוד כללי, מספר משימות שביצע, והדירוג הנוכחי). למנהל יוצגו נתונים כלליים על ביצועי העובד המצטיין בכל יום.

**🔹 מבנה העמוד:**

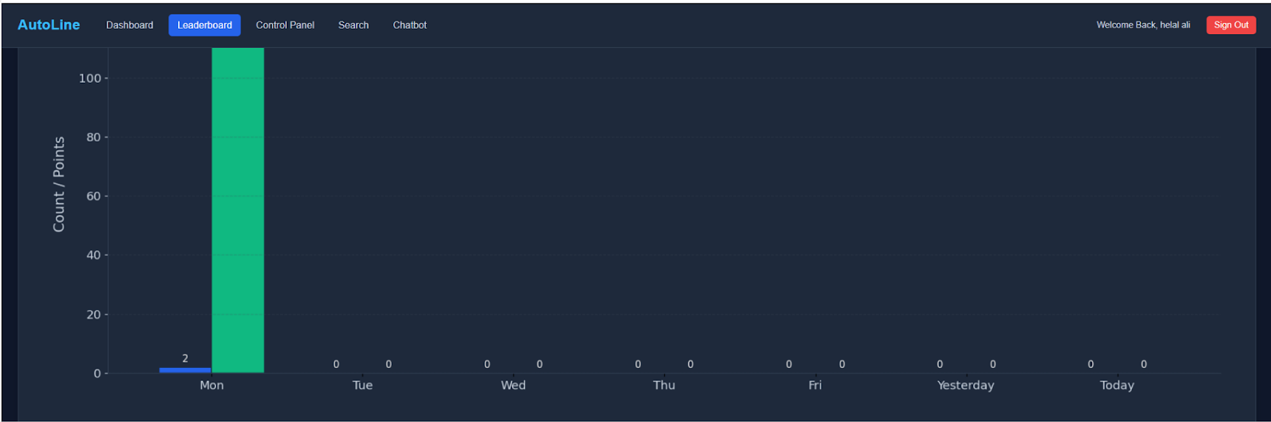
1. **טבלת דירוג חכמה – Top 5 Users:**
   * **מציגה את חמשת המהנדסים המובילים לפי ניקוד כולל (Score).**
   * **כוללת מידע נוסף כמו מספר משימות שבוצעו (Tasks Completed) ומיקום בדירוג (Rank).**
   * **תגים צבעוניים לפי מיקום:  
      🥇 Leader – מקום ראשון  
      🥈 Runner-up – מקום שני  
      🥉 Top 3 – 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.**

****

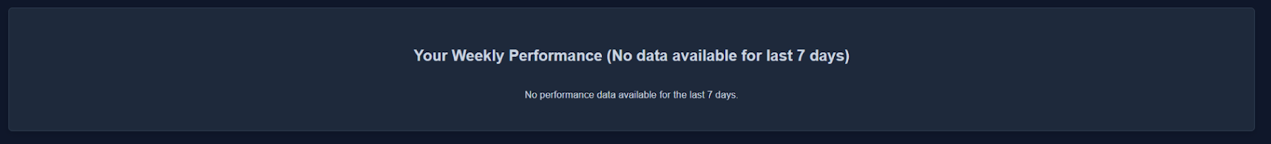
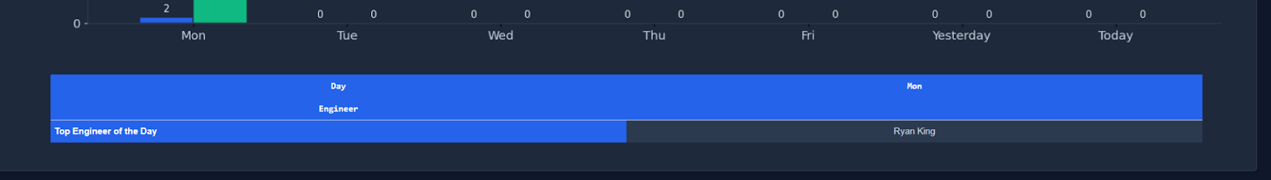
1. **הצגת הודעות במידה ואין נתונים:**
   * **למשל: "No performance data available for the last 7 days" – מוצג במקרה ואין פעילות לאחרונה.**

**🔹 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.**

**🔹 שימושיות:**

* **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.**

**  
**

##### **ד. משימות (**Tasks**) - (למהנדסים 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:

"Daily Optimization Tasks" – עם הסבר קצר על מטרת המשימות.

* 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".

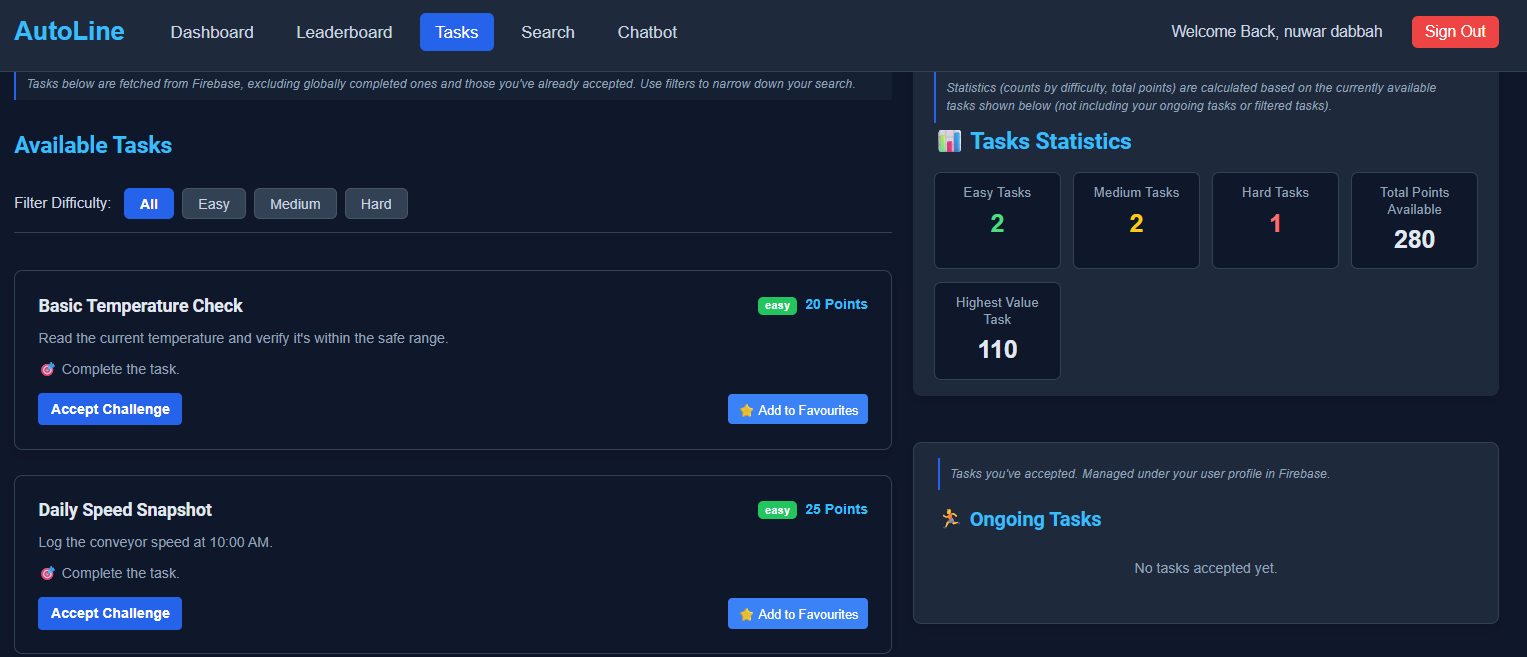
לחיצה על Accept Challenge , המשימה מועברת ל-Ongoing Tasks.

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.

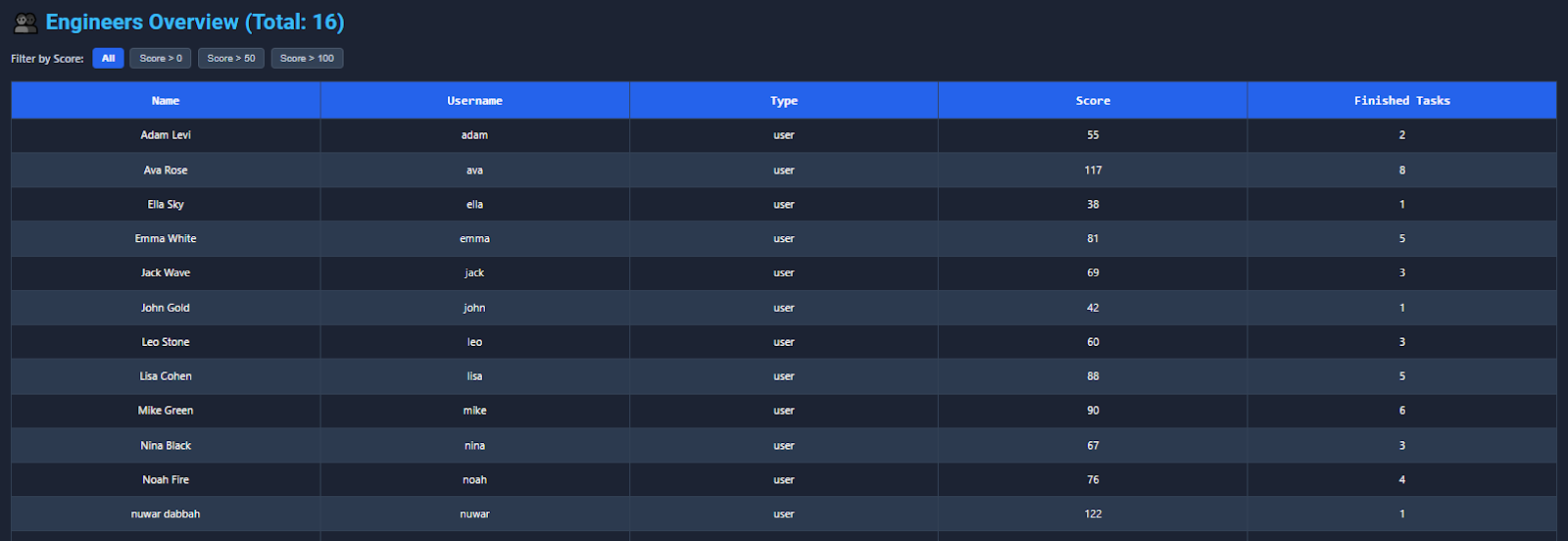
הניקוד מעודד ביצוע משימות בצורה עקבית ושיטתית.



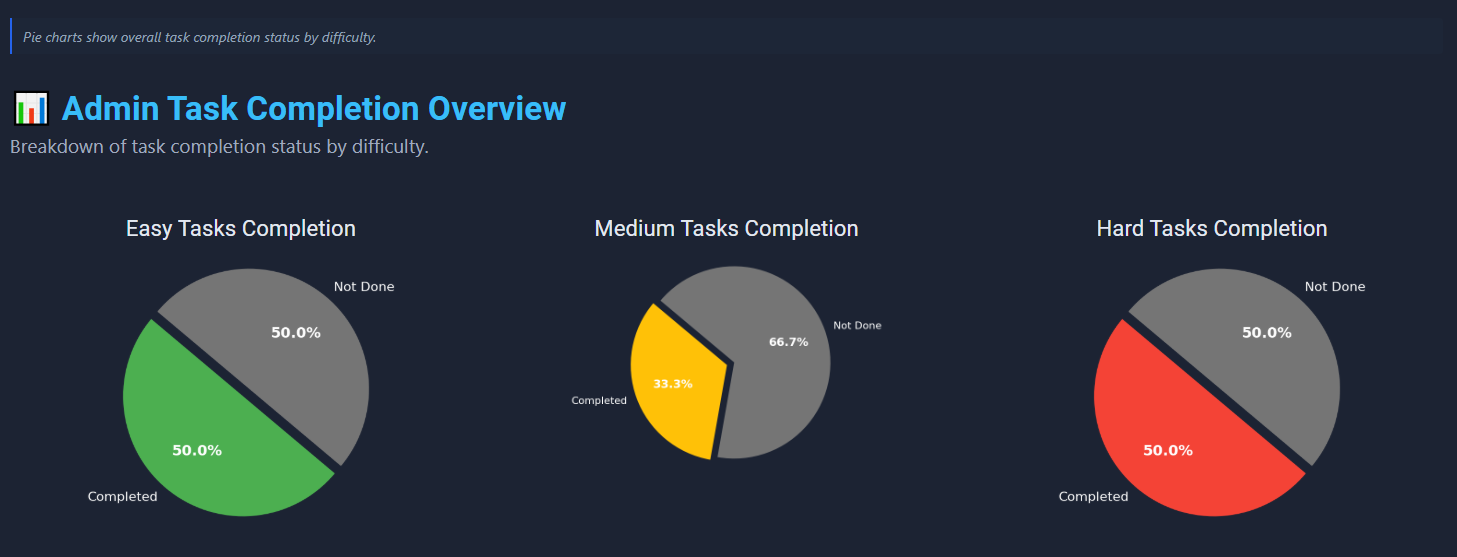
תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

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

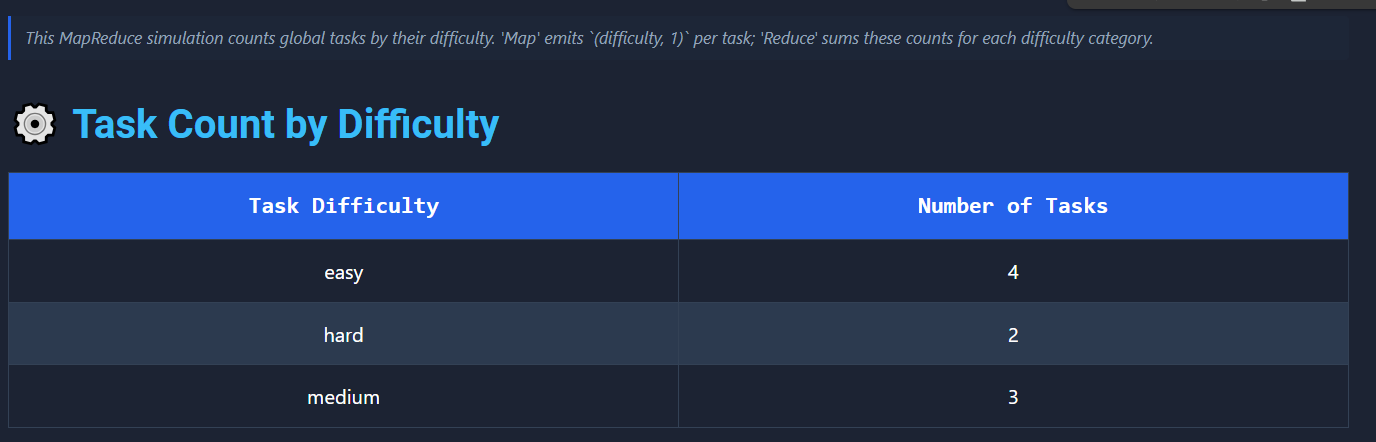
* **מה רואים במסך?** בחלק זה של מערכת הניהול יצרתי טבלה אינטראקטיבית המציגה את כלל המשתמשים מסוג "user" (מהנדסים) מתוך מסד הנתונים של Firebase. הטבלה כוללת מידע כמו שם, שם משתמש, ניקוד (Score), מספר משימות שבוצעו, וסוג המשתמש. בנוסף, הוספתי כפתורי סינון חכמים המאפשרים למנהל לסנן את הטבלה לפי סף ניקוד: "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)**

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.

****

##### **ו. חיפוש (**Search**)**

# **מה רואים במסך?** 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.

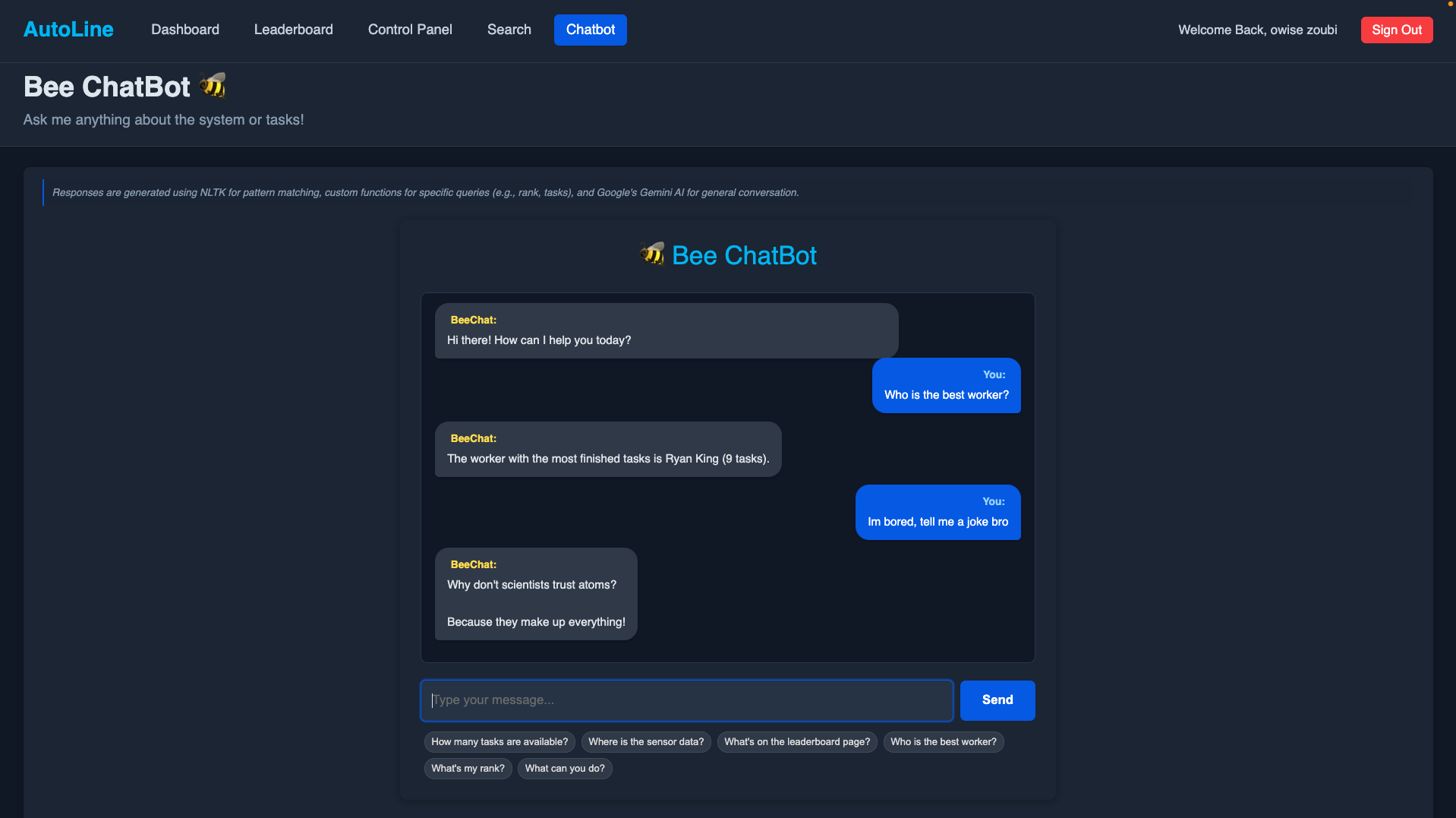
# 

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:
  + כתובת URL: https://mqtt.org/
  + ציון רלוונטיות: 2
* Software (תוכנה):
  + כתובת URL: https://mqtt.org/software/
  + ציון רלוונטיות: 2

##### **ז. צ'אטבוט (**Chatbot**)**

* **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?").
  + להשתמש בכפתורי ההצעות כדי לשאול שאלות נפוצות.



#### **3. מעברים בין מסכים**

הניווט במערכת מתבצע בעיקר באמצעות **סרגל הניווט העליון** שמופיע בראש כל מסך (לאחר ההתחברות).

* לחיצה על כל אחד מהקישורים בסרגל ("Dashboard", "Leaderboard", "Tasks" וכו') תוביל אותך ישירות למסך המתאים.
* הקישור למסך שבו אתה נמצא יודגש בצבע כחול כדי שתדע תמיד היכן אתה.
* בצד ימין של הסרגל נמצא שמך וכפתור **"Sign Out"** להתנתקות בטוחה מהמערכת.

#### **4. טעויות אפשריות ופתרונן**

|  |  |  |
| --- | --- | --- |
| **השגיאה / הבעיה** | **הסיבה האפשרית** | **מה לעשות?** |
| **הודעת שגיאה: "Username and password are required"** | המשתמש לחץ על כפתור "Login" מבלי למלא את שדה שם המשתמש או הסיסמה. | יש למלא את שני השדות, שם המשתמש והסיסמה. |
| **הודעת שגיאה: "Username is required"** | המשתמש מילא את שדה הסיסמה אך השאיר את שדה שם המשתמש ריק. | יש למלא את שדה שם המשתמש. |
| **הודעת שגיאה: "Password is required"** | המשתמש מילא את שדה שם המשתמש אך השאיר את שדה הסיסמה ריק. | יש למלא את שדה הסיסמה. |
| **הודעת שגיאה: "Incorrect username or password"** | הקלדה שגויה של שם המשתמש או הסיסמה. | יש לבדוק היטב את האיות, לשים לב לאותיות גדולות/קטנות, ולוודא שמקש ה-Caps Lock אינו לחוץ. |
| **הודעת שגיאה: "Verification code has expired"** | עברו יותר מ-5 דקות מרגע שליחת הקוד למייל ועד לרגע הזנתו. | יש ללחוץ על כפתור **"Resend Code"** כדי לקבל קוד חדש ותקף. |
| **הכרטיסיות ב-Dashboard מציגות "No Data" ולא מתעדכנות.** | אין נתונים חדשים שמוזנים למערכת מקו הייצור, או שישנה תקלה בשליפת הנתונים ממסד הנתונים. | זו אינה טעות של המשתמש. יש לפנות למנהל המערכת או לתמיכה הטכנית כדי שיבדקו את מקור הנתונים. |
| **משימה שאני רוצה לבצע לא מופיעה ברשימת "Available Tasks".** | 1. המשימה כבר הושלמה על ידי מהנדס אחר  2. המשימה כבר נמצאת אצלך ברשימת **"Ongoing Tasks"**. | יש לבדוק תחילה את רשימת "משימות בביצוע" שלך. אם היא לא שם, סביר להניח שהיא כבר אינה רלוונטית. |

1. 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.

**הסרטון מצורף בקובץ ההגשה**

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!