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

**תרגיל בית 3**

שמות חברי הצוות:

ימית מורנו 207476102

עומר זילחה 211695440

קארין ארמה 209702299

נוי בן עזרא 211565130

בר אלרום 209121573

יובל כץ 207504952

**קישור למחברת קולאב:** <https://colab.research.google.com/drive/1KbjCNq8rkStciOdIXJJJAICe-JvvITug?usp=sharing>

**קישור לגיט:**

<https://github.com/KarynA19/Cloud-Computing>

**חלק ראשון**

**מהנדסת מערכת-קארין ארמה**

| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** | **Acceptance test –** בדיקת קבלה |
| --- | --- | --- | --- |
| קארין ארמה | הוספת צ'אט בוט לאתר | צ'אט בוט נוסף לאתר | **טעינה והצגה של הודעות בצ'אט**  מטרה: לבדוק שהצ'אט עובד. פעולה: להזין ידנית הודעה חדשה ב־Firebase תחת הצ'אט, ואז להריץ את תא הצגת ההודעות. תוצאה צפויה: ההודעה מופיעה ברשימה. |
| נוי בן עזרא | בדיקת חיבור ל firebase ומעבר על המשימות ב word. | נבדק חיבור למסד הנתונים והושלמו משימות ב word. | **טעינת גרפים גם כשאין נתונים**  מטרה: לבדוק שהקוד לא קורס כשאין קריאות מהחיישנים. פעולה: לוודא שהגרפים מציגים "No data" / גרף ריק בצורה תקינה. תוצאה צפויה: המערכת לא מציגה שגיאה גם אם רשומות החיישנים ריקות. |
| יובל כץ | בדיקת מקרי קצה ותיקון באגים. | המערכת עובדת במלואה ותוקנו באגים. | **התחברות ל־Firebase**  מטרה: לוודא שהקוד מתחבר לשירות Firebase. פעולה: להריץ את התא שמכיל את הגדרות Firebase והקישור למסד הנתונים. תוצאה צפויה: אין שגיאת התחברות; אם הנתונים חסרים, המערכת לא קורסת. |
| עומר זילחה | עיצוב המסכים. | המסכים עוצבו. | **בדיקה של עדכון ערך חיישן ב־Firebase והשפעתו על הגרף**  מטרה: לוודא שהגרף משתנה כשהנתון משתנה. פעולה: לעדכן ידנית ערך חיישן (למשל טמפ') ב־Firebase. תוצאה צפויה: בהרצת הגרף מחדש, הנתון החדש יוצג. |
| בר אלרום | בדיקת הפיצר באתר | הפיצר נבדק ועובד. | **הרצה מלאה של הקוד מתחילתו ועד סופו** מטרה: לבדוק את ה־flow המלא. פעולה: להריץ את כל התאים מלמעלה עד למטה. תוצאה צפויה: אין שגיאות; כל תא מסתיים בהצלחה גם אם אין נתונים חיצוניים. |
| ימית מורנו | הכנת סרטון המתאר את הפונקציונליות של המערכת | הוכן סרטון | בדיקה של זמן טעינה סביר לגרפים מטרה: לוודא שהגרפים נטענים תוך זמן קצר. פעולה: למדוד זמן מהרצת התא עד הופעת הגרף. תוצאה צפויה: פחות מ־5 שניות (בהתאם לחיבור). |

***חלק שני: בניית המערכת***

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

**שימוש ב Microservices :**

במהלך פיתוח המערכת יישמנו את עקרונות ה־ **Microservices** בכך שפיצלנו את הלוגיקה לשירותים מבודדים, כאשר כל שירות אחראי על פונקציונליות אחת בלבד. השירותים פועלים בתהליכים נפרדים ומתקשרים ביניהם.

**שירות 1: Sensor Writer Service - כתיבת נתוני חיישנים למסד הנתונים**

שירות עצמאי אשר אחראי על קליטת נתוני חיישנים מסוג indoor ו־outdoor משרת ה־MQTT ועיבודם לטובת עדכון שוטף במסד הנתונים Firebase .

**מאפיינים עיקריים:**

* פועל כתהליך נפרד שרץ ברציפות ללא תלות בשירותים אחרים במערכת.
* מקשיב לנושאים רלוונטיים במערכת ה־MQTT ומטפל בכל הודעה שנכנסת באופן מיידי.
* מבצע עדכון של ערכי החיישנים בפורמט מובנה לתוך Firebase .

**תועלת למערכת:**

* מבטיח שהנתונים המתקבלים מהחיישנים יהיו זמינים בזמן אמת לשירותים אחרים כגון הצגה גרפית, ניתוחים סטטיסטיים, או פעולות משתמש.
* מאפשר הפרדה בין איסוף נתונים לעיבוד והצגה, וכך תורם לארכיטקטורה מודולרית וניתנת להרחבה.
* מייעל את ביצועי המערכת על ידי כך שאינו חוסם או מעכב את פעולת הממשק או שירותים אחרים.

**שירות 2: Search Engine Service - בניית אינדקס תוכן מאתרים חיצוניים**

שירות עצמאי אשר אחראי על שליפת תוכן טקסטואלי מאתרי אינטרנט, ניתוח מילות מפתח, ובניית אינדקס חכם לצורכי חיפוש. התוצר נשמר במסד הנתונים Firebase ומאפשר שאילתות חיפוש על בסיס תוכן אמיתי שנמצא באתרי אינטרנט.

**מאפיינים עיקריים:**

* מבצע קריאה לאתרים חיצוניים, חילוץ טקסט וניתוח מילים באמצעות ביטויים רגולריים.
* מסנן מילים נפוצות (Stop Words), סופר הופעות, ומבנה אינדקס הכולל כתובת URL וכמות הופעות לכל מילה.
* שומר את האינדקס במסד הנתונים Firebase בפורמט מותאם לחיפוש.
* מאפשר שליפת תוצאות חיפוש מדויקות לפי מילות מפתח, כולל דירוג לפי כמות הופעות ודיוק.

**תועלת למערכת:**

* מאפשר שילוב מנוע חיפוש תוכן מותאם אישית המבוסס על מידע דינמי מהאינטרנט.
* תורם להפרדת תחומי אחריות - אינדוקס מתבצע כשירות עצמאי, ואינו תלוי ב UI או בנתוני חיישנים.
* ניתן להרחבה בקלות לשירותים מתקדמים כגון ניתוח סמנטי, זיהוי ישויות, או המלצות חכמות.

1. צ'אטבוט שישולב במערכת מבוסס AI (כפי שלמדתם בתרגול 8), העונה בצורה אינטליגנטית לשאלות המשתמש. (15 נקודות).

במערכת שולב צ'אטבוט חכם המבוסס על מודל השפה Gemini של Google, באמצעות שימוש ב־API. צ'אטבוט מקבל קלט מהמשתמש, שולח אותו למודל, ומחזיר תשובה אינטליגנטית בשפה טבעית.

התשובה מעובדת ומוצגת בצורה מקוצרת ונעימה למשתמש, עם אפשרות להצגת התשובה המלאה.

המימוש כולל שילוב בין תשתית ויזואלית (ipywidgets) לבין קריאות API בזמן אמת.

1. מהם ה -KPI הרלוונטיים בפרויקט שלכם? הסבירו?

ניתן להגדיר מספר מדדי ביצוע מרכזיים (KPI) הרלוונטיים לתפקוד ואיכות המערכת:

### אמינות ודיוק הנתונים (Data Accuracy KPI)

**הסבר:** כדי להבטיח שהמערכת תשקף את המצב האמיתי במעבדה – נדרש לבדוק עד כמה ערכי החיישנים המוצגים באפליקציה תואמים למדידות האמיתיות. דיוק גבוה מונע תגובה שגויה ומחזק את אמון המשתמשים במערכת.

* **מה נמדד:** רמת ההתאמה בין נתוני החיישנים באפליקציה לבין מדידות ידניות בפועל (למשל מד טמפרטורה פיזי במעבדה).
* **איך נבדק:** באמצעות השוואה של קריאות נבחרות מול מדידה ידנית, ובחינת האם הן נמצאות בטווח סטייה מותר ( ±0.5°C).
* **יעד (SLA):** לפחות **99%** מהקריאות יהיו מדויקות ויכנסו לטווח הסטייה שהוגדר מראש.

### זמינות המערכת (System Availability KPI)

**הסבר:** מערכת שאינה זמינה בזמן אמת אינה מאפשרת ניטור תקין של תנאים קריטיים – מה שעלול להוביל לנזק. זמינות גבוהה מעידה על המשכיות תפעולית ועל אמינות בשטח.

* **מה נמדד:** אחוז הזמן שבו המערכת פועלת כראוי ונגישה למשתמשים, לאורך זמן (למשל במהלך חודש).
* **איך נבדק:** ניטור רציף של זמני פעילות מול זמני השבתה, לדוגמה על ידי לוגים או כלי ניטור שרתים (Firebase או מקומי), והשוואה מול זמן כולל.
* **יעד (SLA):** זמינות של לפחות 99% מהזמן החודשי – כלומר מקסימום 7 שעות השבתה בחודש.

### עדכניות הנתונים (Data Freshness KPI)

**הסבר:** נתונים שמתעדכנים באיחור פוגעים ביכולת לזהות חריגות בזמן אמת. KPI זה בודק את מהירות ההגעה של נתוני החיישנים לאפליקציה.

* **מה נמדד:** הזמן שעובר בין הרגע שהחיישן מייצר נתון חדש – לבין רגע ההצגה שלו באפליקציה בפועל.
* **איך נבדק:** על ידי חישוב הפרש בין Timestamp של שליחת הנתון ל־Firebase לבין זמן הרינדור/הצגה באתר, באמצעות לוגים או כלים מובנים בקוד.
* **יעד (SLA):** לפחות **95%** מהנתונים יוצגו למשתמש תוך עד 5 שניות מרגע ההפקה.

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

**שקיפות אלגוריתמית (Algorithmic Transparency):**

במהלך הפיתוח שמנו דגש על עקרון השקיפות על מנת שהמשתמש יבין מה המערכת עושה, אילו נתונים נאספים, ואיך האלגוריתם משפיע על התוצאה המוצגת.

**1. הצגת נתוני חיישנים בזמן אמת**

· הנתונים מהחיישנים (כגון טמפרטורה, לחות, לחץ אויר) מוצגים למשתמש בממשק בצורה ברורה, מופרדים לפי אזור פנימי (indoor) וחיצוני (outdoor).

· כל ערך מלווה בחותמת זמן המציינת את מועד העדכון האחרון.

· המשתמש יכול לראות בכל רגע אילו נתונים נאספים, מתי הם התקבלו, ואילו חיישנים סיפקו אותם.

**2. הסבר ודירוג תוצאות חיפוש**

· תוצאות החיפוש מוצגות עם מידע מפורט על כמות הופעות של כל מילה במסמך.

· כל תוצאה כוללת גם דירוג לפי מספר מילים תואמות ומספר הפעמים שכל מילה הופיעה.

· המשתמש מבין מדוע תוצאה מסוימת מופיעה בראש הרשימה, ומקבל הבנה על קריטריונים של דירוג האלגוריתם.

**3. שקיפות במערכת הניקוד של המשתמשים (Game Mode)**

· כל פעולה שמבצע המשתמש (כגון שינוי ערך או הפעלת מנגנון) מוצגת לו עם מספר הנקודות שהוענקו.

· מוצג גם הציון המצטבר לאחר כל פעולה, כך שהמשתמש רואה את ההשפעה הישירה של פעולותיו.

· המידע מוצג מיידית ובאופן ישיר, ללא חישובים נסתרים או הפתעות.

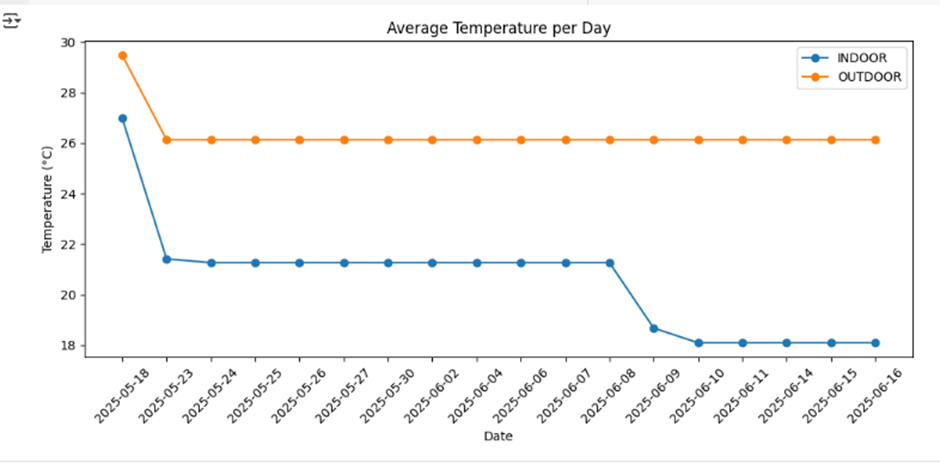
**4. הסברים טקסטואליים ומילוליים בממשק**

· לצד טבלאות הנתונים והגרפים מוצגים הסברים לגבי טווחים תקינים, חריגות ערכים ומשמעות מדדים סטטיסטיים.

· המשתמשים מקבלים הנחיה כיצד לפרש את המידע, מה נחשב חריג, ומה משמעות סטייה מהטווח התקין.

· השילוב בין ויזואליזציה לבין הסבר טקסטואלי מעודד הבנה ובקרה על תהליכים בזמן אמת.

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



ביצענו ניתוח על נתוני חיישני טמפרטורה Indoor ו־Outdoor שנשמרו במסד הנתונים.  
 לצורך כך השתמשנו ב Spark – על מנת לטעון את הנתונים ולבצע עליהם פעולת MapReduce ובאמצעות groupBy חישוב ממוצע יומי לכל מיקום.

**ניתוח הגרף:**

הגרף מתאר את ממוצע הטמפרטורות היומי שנמדדו בתוך מבנה (INDOOR) ומחוץ לו (OUTDOOR) לאורך תקופה של כחודש.

* ביום הראשון (18.5), נרשמות טמפרטורות גבוהות במיוחד בשני המיקומים – מעל 29° בחוץ וכמעט 27° בפנים. ניתן להניח שמדובר ביום חריג שאינו מייצג את התנהגות שאר החודש, ולכן יש להתייחס אליו כאל קיצון חד־פעמי.
* החל מ־23.5, הטמפרטורות ב־OUTDOOR מתייצבות על ערך קבוע של הטמפרטורה, מה שמעיד ככל הנראה על חוסר בשינויים דרמטיים במזג האוויר החיצוני.
* ב־INDOOR לעומת זאת, לאורך תקופה ארוכה הטמפרטורה נשמרת קבועה, אך החל מ 8.6 נרשמת ירידה פתאומית בטמפרטורות. ייתכן שהדבר נובע מהפעלת מערכת מיזוג אוויר בתוך החדר, שכן מזג האוויר החיצוני נותר יציב.
* ההפרש בין טמפרטורת החוץ לפנים נותר יציב יחסית עד ה־9.6, ואז גדל בצורה ברורה. שינוי זה עשוי להעיד על השפעה אנושית על תנאי החדר, ולא על תנודות סביבתיות טבעיות.

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

### האתגרים שעלו במהלך העבודה וכיצד התמודדנו איתם:

1. **שילוב טכנולוגיות רבות (MQTT, Firebase, NLTK, BeautifulSoup וכו')** הפרויקט כלל עבודה עם מגוון רחב של ספריות וטכנולוגיות, שכל אחת מהן דורשת קונפיגורציה שונה. לדוגמה, חיבור ל-MQTT ו-Firebase דרש קריאה לתיעוד, טיפול בהרשאות, ותחזוקה של פורמטים תואמים בין החלקים השונים של הקוד.  
   **התמודדות**: ביצענו ניסויים בשלבים, בדקנו את תקינות החיבורים בנפרד, השתמשנו בלוגים וב-printים לאיתור תקלות בשלב הפיתוח.
2. **קבלת מידע בזמן אמת וניהול עדכונים (MQTT live streaming)** נדרשה קליטה של נתונים חיה משני חיישנים שונים (indoor ו-outdoor) באמצעות פרוטוקול MQTT. הדבר חייב טיפול יעיל בנתונים מתעדכנים והתמודדות עם אפשרות לשגיאות בזמן ריצה.  
   **התמודדות**: יצרנו callbacks מובנים ל-on\_connect ו-on\_message, והוספנו טיפול בשגיאות באמצעות try-except כדי למנוע קריסה של המערכת במקרה של שגיאה בנתונים הנכנסים.
3. **שימוש במבנה קוד קריא ושמורכבות נמוכה** בגלל היקף הפונקציונליות, היה צורך לארגן את הקוד כך שיהיה ברור ונוח לתחזוקה – במיוחד לאור ריבוי מקורות המידע (חיישנים, firebase, נתונים מאתרים).  
   **התמודדות**: מיקמנו את הפונקציות המרכזיות בצורה מודולרית והקפדנו על שמות משתנים ברורים. השתמשנו בהערות ובפירוק לקבצי קוד לפי תפקידים.
4. **אתגרי ויזואליזציה של נתונים** מאחר שמדובר בנתונים מתעדכנים, הצגת המידע בצורה ברורה ויזואלית הייתה מאתגרת – במיוחד באקוסיסטם של Jupyter Notebook שבו כל cell מרונדר מחדש.  
   **התמודדות**: השתמשנו ב־matplotlib וב־ipywidgets כדי לאפשר עדכונים אינטראקטיביים ויצירת ממשק משתמש בסיסי. כמו כן, נעשה שימוש ב־clear\_output כדי למנוע הצפה של מידע מיותר.
5. **אבטחת מידע בחיבור ל-Firebase** בעת עבודה עם Realtime Database ב-Firebase, יש להתחשב בזכויות גישה ובמפתחות API על מנת להימנע מחשיפה של מידע רגיש.  
   **התמודדות**: נעשה שימוש ב-credentials וב־firebase admin כדי לוודא שהגישה מתבצעת בצורה מאובטחת ומבוקרת.

***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.

#### קליטת חיישנים ועדכון בסיס נתונים:

* on\_message(client, userdata, msg – מקבלת נתוני MQTT
* push\_sensors\_to\_firebase() – מעדכנת את הנתונים העכשוויים ב־Firebase
* save\_current\_to\_history() – שומרת את הנתונים להיסטוריה

#### מנוע חיפוש:

* build\_word\_index\_multiple\_urls(urls, stop\_words – בונה אינדקס מילים מאתרים
* save\_index\_to\_firebase(index – שומר את האינדקס ב־Firebase
* search\_word(word – מחפש מילה באינדקס
* search\_sentence\_links(sentence – מחפש ביטוי שלם
* on\_search\_click() – מחבר בין כפתור החיפוש להצגת תוצאות

#### תצוגת היסטוריית חיישנים:

* fetch\_sensor\_data(sensor\_type, from\_dt, to\_dt– שליפה לפי טווח זמן
* show\_history\_table(...) – הצגת טבלה עם נתונים בטווח נבחר
* create\_stats\_table(data\_dict, thresholds – יצירת סטטיסטיקות עם זיהוי חריגות

#### משחק:

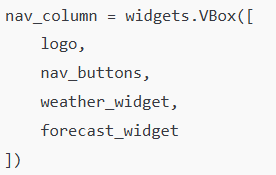
* display\_login\_screen() – מסך כניסה עם מזהה
* display\_game\_mode() – תפריט פעולות עם ניקוד
* update\_points\_for\_id(user\_id, points) – שמירה על ניקוד ב־Firebase
* leaderboard\_widget() – הצגת טבלת מובילים

#### תחזית מזג אוויר:

* fetch\_weather\_forecast() – שולפת תחזית מ־Open-Meteo API
* render\_forecast\_widget() – מציגה תחזית ל־4 ימים הקרובים עם אייקונים

תבניות עיצוב מעניינות:

**סרגל ניווט (תפריט צדדי):**



#### עיצוב:

* תפריט צדדי שבנוי בעזרת ספריית ipywidgets זוהי ספרייה גרפית לפייתון – מציגה כפתורים, תיבות טקסט, תפריטים ועוד
* כל רכיב (כפתור / תיבה) מופיע בתוך VBox – עמודה אנכית.
* העיצוב נובע מהמבנה של Colab ולא מעיצוב חיצוני.

**טבלת סטטיסטיקה עם צבעים ו־hover:**

****

#### עיצוב:

* טבלה דינמית שנוצרת מפייתון (pandas.DataFrame.to\_html).
* CSS מוסיף הדגשת שורה במעבר עכבר (hover)- כשהעכבר עובר על כל שורה (<tr>) בתוך גוף הטבלה (<tbody>), צובע את השורה בכחול-בהיר.
* תא עם ערכים חריגים נצבע בצבע ע"י תנאים לוגיים

#### ספריות:

* שילוב בין pandas + CSS

**תחזית מזג אוויר ל־4 ימים הקרובים:**

****

עיצוב:

* אייקון מזג אוויר לפי weather\_code (שמש, גשם וכו')
* טווח טמפרטורות: {min}°C - {max}°C
* האייקונים נבחרים מתוך מילון weather\_code\_icons לפי קוד מזג האוויר שהגיע מה־API.

#### ספריות:

* requests – לקריאת תחזית מ־Open-Meteo API

1. יש לבנות תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.

**תיק משתמש**

**הסבר כללי על המערכת**

המערכת שפותחה נועדה לאסוף, להציג ולנתח נתונים המתקבלים מחיישנים מסוג Indoor ו־Outdoor, במקביל להפעלת מנוע חיפוש אינטרנטי ואלמנטים של ממשק משחק (Gamemode) עבור משתמשים טכניים. המערכת מספקת למשתמשים תמונה חיה של מצב הסביבה באמצעות נתוני חיישנים, לצד כלים סטטיסטיים מתקדמים, יכולות חיפוש חיצוניות, וחוויית משתמש המדמה משימות מהעולם התפעולי לצורכי תרגול, למידה ואיסוף ניקוד אישי.

המערכת בנויה לפי עקרונות Microservices – כל רכיב אחראי לתפקוד עצמאי, עם ממשקי תקשורת ברורים ביניהם. מבנה זה מאפשר גמישות והתאמה נוחה לתחזוקה עתידית או הרחבות מערכת.

בכניסה למערכת מוצג מסך הבית הכולל תפריט ניווט ראשי עם ארבעה כפתורים:  
 HOME, STATISTICS, GAMEMODE, ו־SEARCH.  
כל כפתור מעביר את המשתמש למסך ייעודי, כשהמעברים מהירים ושומרים על עקביות עיצובית לשם חוויית שימוש רציפה.

**פירוט מסכים**

:HOME

במסך הבית מוצגים נתוני החיישנים (INDOOR/OUTDOOR) בזמן אמת.   
בין הנתונים: מרחק, טמפרטורה, לחות, לחץ ועוד.

STATISTICS:

במסך זה המשתמש בוחר בין חיישני Indoor או Outdoor. המערכת מציגה גרף בזמן אמת של ערכים שנאספו, יחד עם טבלה סטטיסטית הכוללת ממוצע, סטיית תקן וערכים קיצוניים. המידע כולו מתעדכן בזמן אמת מתוך מסד הנתונים Firebase.

GAMEMODE:

המשתמש מזדהה בעזרת מזהה UID ומקבל משימות תחזוקה טכניות המדמות סביבה תפעולית. כל משימה מבוססת על הנתונים הנוכחיים מהחיישנים, וביצועה מזכה בניקוד. הנתונים נשמרים במערכת לצורכי השוואות עתידיות.

SEARCH:

מאפשר למשתמש להזין ביטוי חופשי ולקבל קישורים לאתרים שבהם נמצאו התאמות רלוונטיות. מנוע חיפוש פנימי סורק אתרי אינטרנט, יוצר אינדקס הפוך (Inverted Index), ומחזיר תוצאות מדורגות לפי מספר הופעות של מילים רלוונטיות בביטוי.

**מעברים בין מסכים**

המערכת בנויה כך שכל ממשק מוצג במסך ייעודי ונפרד, ומעבר בין מסכים מתבצע באמצעות תפריט ניווט ראשי, הממוקם בראש או במרכז הדף, בהתאם לעיצוב. הכפתורים המוצגים במסך זה מאפשרים ניווט ישיר לארבעה אזורים מרכזיים: Home, Statistics, GameMode, ו-Search.

לחיצה על כפתור Home, המשתמש חוזר למסך הראשי של המערכת - שם ניתן לקבל מידע כללי, התמצאות ראשונית ובחירה בפעולה הבאה.   
  
לחיצה על כפתור Statistics מעבירה את המשתמש למסך ניתוח החיישנים, שם הוא בוחר את סוג החיישן (indoor או outdoor), מריץ גרף חי של הנתונים, וצופה בטבלת סטטיסטיקה מתקדמת.

לחיצה על כפתור GameMode מעבירה את המשתמש למסך ה-Gamemode, ומוביל לאזור המיועד למשתמשים שמעוניינים לבצע סימולציה של משימות תחזוקה, לצבור נקודות ולבצע פעולות בהתאם לנתוני המערכת בפועל. לאחר הזנת UID, המשתמש מקבל משימות המתאימות לו, ויכול לבצע אותן בלחיצת כפתור.

בלחיצה על כפתור Search, נפתח מסך החיפוש אשר מאפשר הזנת משפט חופשי בשדה טקסט, מריצה את אלגוריתם החיפוש, ומחזיר קישורים רלוונטיים על פי ניתוח המילים במשפט.

**טעויות אפשריות ומנגנוני טיפול**

המערכת ערוכה לטיפול במצבים חריגים ושגיאות פרשנות מצד המשתמש, תוך הצגת הודעות ברורות והנחיות לפעולה. להלן דוגמאות עיקריות:

חיפוש שלא הניב תוצאות:  
 אם המשתמש מזין מונח שלא קיים באינדקס, תוצג הודעה בהתאם.

הזדהות שגויה במצב המשחק:  
 אם מוזן UID שגוי או ריק, תוצג הודעה מפורשת עם אפשרות להזדהות מחדש או להירשם.

פעולה ללא נתוני חיישן זמינים:  
 המערכת מונעת ביצוע פעולות שאין עבורן מידע עדכני, ומציגה הודעת הסבר בהתאם.

שגיאות מערכת:  
 במקרה של תקלות מערכתיות (למשל ניתוק מ־Firebase), מוצגת הודעה זמנית עם ניסיון חיבור חוזר אוטומטי.

כל הודעות השגיאה נכתבות בשפה ברורה, ללא מונחים טכניים, ומאפשרות למשתמש להמשיך לעבוד בצורה רציפה ובטוחה.

1. התייחסו בתיק המשתמש לשקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?

**שקיפות אלגוריתמית (Algorithmic Transparency):**

במהלך הפיתוח שמנו דגש על עקרון השקיפות על מנת שהמשתמש יבין מה המערכת עושה, אילו נתונים נאספים, ואיך האלגוריתם משפיע על התוצאה המוצגת.

**1. הצגת נתוני חיישנים בזמן אמת**

· הנתונים מהחיישנים (כגון טמפרטורה, לחות, לחץ אור) מוצגים למשתמש בממשק בצורה ברורה, מופרדים לפי אזור פנימי (indoor) וחיצוני (outdoor).

· כל ערך מלווה בחותמת זמן המציינת את מועד העדכון האחרון.

· המשתמש יכול לראות בכל רגע אילו נתונים נאספים, מתי הם התקבלו, ואילו חיישנים סיפקו אותם.

**2. הסבר ודירוג תוצאות חיפוש**

· תוצאות החיפוש מוצגות עם מידע מפורט על כמות הופעות של כל מילה במסמך.

· כל תוצאה כוללת גם דירוג לפי מספר מילים תואמות ומספר הפעמים שכל מילה הופיעה.

· המשתמש מבין מדוע תוצאה מסוימת מופיעה בראש הרשימה, ומקבל הבנה על קריטריונים של דירוג האלגוריתם.

**3. שקיפות במערכת הניקוד של המשתמשים (Game Mode)**

· כל פעולה שמבצע המשתמש (כגון שינוי ערך או הפעלת מנגנון) מוצגת לו עם מספר הנקודות שהוענקו.

· מוצג גם הציון המצטבר לאחר כל פעולה, כך שהמשתמש רואה את ההשפעה הישירה של פעולותיו.

· המידע מוצג מיידית ובאופן ישיר, ללא חישובים נסתרים או הפתעות.

**4. הסברים טקסטואליים ומילוליים בממשק**

· לצד טבלאות הנתונים והגרפים מוצגים הסברים לגבי טווחים תקינים, חריגות ערכים ומשמעות מדדים סטטיסטיים.

· המשתמשים מקבלים הנחיה כיצד לפרש את המידע, מה נחשב חריג, ומה משמעות סטייה מהטווח התקין.

· השילוב בין ויזואליזציה לבין הסבר טקסטואלי מעודד הבנה ובקרה על תהליכים בזמן אמת.

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.

**קישור לסרטון:**   
<https://drive.google.com/file/d/1osrb2PPakMFm9yhx5ypHOv4tesxIJEq_/view?usp=sharing>

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!