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

**תרגיל בית 2**

שמות חברי הצוות:

ימית מורנו 207476102

עומר זילחה 211695440

קארין ארמה 209702299

נוי בן עזרא 211565130

בר אלרום 209121573

יובל כץ 207504952

**בר אלרום- מהנדס מערכת**

| **איטרציה 1** | | |
| --- | --- | --- |
| **שם חבר הצוות ותפקיד בתרגיל זה** | **משימות שהוקצו** | **משימות שהושלמו** |
| בר אלרום-Scrum Master | חיבור המסכים שנעשו על ידי חברי הצוות. | המסכים חוברו בהצלחה. |
| נוי בן עזרא-UI | עיצוב המסכים, שמירה על נראות ברורה ופשוטה. | המסכים עוצבו בצורה נוחה וידידותית למשתמש. |
| יובל כץ-Product Manager | מעבר על החשיבה העיצובית שהוגשה בתרגיל בית 1 והתאמה לפרויקט. | דגשים חשובים מהחשיבה עיצובית הותאמו למערכת. |
| עומר זילחה-QA | מעבר על כל הקוד והמסכים ותיקון באגים. | המסכים נבדקו בקפידה תוקנו באגים במידת הצורך. |
| קארין ארמה -Backend Developer | פיתוח מסד הנתונים ב firebase ובדיקה שהכל מתחבר למסד הנתונים. | מסד הנתונים נבנה וכל המערכת מחוברת אליו. |
| ימית מורנו-Frontend Developer | בדיקת המסכים והכפתורים שהמשתמש רואה במערכת. | בוצע ניסיון השתמשות במערכת ובדיקת כל הכפתורים והמסכים. |

**אינדקס**

**Stop Words** שנבחרו:

'a', 'an', 'the', 'and', 'or', 'is', 'are', 'was', 'were', 'be','to', 'from', 'of', 'in', 'on', 'at', 'as', 'with', 'for', 'it', 'this', 'that', 'these', 'those', 'which', 'such', 'also', 'can','will', 'may', 'should', 'could', 'would', 'have', 'has', 'had'.

**מילות קישור/מבנה תחבירי** למשל: and, or, as, with  
 לא מוסיפות מידע טכני, רק מחברות בין רעיונות.

**פעלים עזר נפוצים** –למשל: is, are, was, have, will  
 לא מתארים פעולה ממשית שקשורה ל-MQTT, רק משמשים להטיית זמנים.

**מילות ייחוס כלליות** – למשל: this, that, those, such  
 לא תורמות משמעות ברמת תוכן – לא מונחי מפתח.

**מילות מקום וזמן** – למשל: in, on, at, from  
 לא חשובות באינדוקס של מונחים טכנולוגיים.

עשינו שימוש ב־**stemming** באמצעות האלגוריתם PorterStemmer מספריית NLTK. בחרנו דווקא בשיטה זו מכיוון שהיא פשוטה, מהירה, ואינה דורשת מידע נוסף על ההקשר הדקדוקי של המילים. מכיוון שמדובר באינדוקס של טקסט טכני מתוך אתר mqtt.org, שבו מופיעות מילות מפתח קבועות יחסית כמו "client", "broker", "publish", לא היה צורך בדיוק לשוני גבוה אלא בעיקר בזיהוי אחיד של מונחים דומים. למשל, המילים "messages", "messaging" ו-"message" כולן התכנסו לשורש "messag", מה שחסך כפילויות ואפשר לבנות אינדקס פשוט ויעיל. לעומת זאת, lemmatization, שמחזירה את צורת היסוד הדקדוקית של מילה, דורשת עיבוד מורכב יותר ולעיתים איטי בהרבה. לכן, מכיוון שמהירות ויעילות היו חשובות יותר מדיוק לשוני, במיוחד עבור טקסט טכני, בחרנו ב־stemming.

**קישור למחברת קולאב:**

<https://colab.research.google.com/drive/15j5G35BjGidPqRa7-_L9zhXNc9iEkTcT?usp=sharing>

**קישור לgit של הפרויקט:**

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

**8 כללי הזהב של שניידרמן:**

**1. עקביות (Strive for consistency)** עיצוב הממשק עקבי בין המסכים השונים, הכפתורים דומים בגודל ובצבע, אותם מונחים משמשים בכל המסכים.

**2. קיצורי דרך למשתמשים מתקדמים (Enable frequent users to use shortcuts)** משתמשים יכולים לחפש מידע בכפתור החיפוש ולקבל תשובה מהירה לשאלתם.

**3. משוב מיידי (Offer informative feedback)** המערכת מציגה בזמן אמת ערכים מהחיישנים  
  
**4. סיום ברור לפעולות (Design dialogs to yield closure)** לאחר ביצוע פעולה מוצגת הודעת הצלחה.  
  
**5. מניעת שגיאות (Prevent errors)** יש הגבלות על קלט לא תקין כמו טמפרטורה שלילית.

**6. אפשרות לבטל (Permit easy reversal of actions)** ניתן לבטל פעולות או לחזור אחורה במסכים.

**7. שליטת המשתמש (Support internal locus of control)** המשתמש בוחר אילו חיישנים להציג, הוא מחליט מתי להפעיל את המשחקים או לחפש מידע.

**8. הפחתת עומס על הזיכרון (Reduce short-term memory load)** כל הנתונים החשובים מוצגים על המסך, אין צורך לזכור ערכים או מיקומים ממסכים קודמים.

**משובים**

| **הערת משוב** | **האם התבצע שינוי באפליקציה בעקבות ההערה?** | **נימוק** |
| --- | --- | --- |
| יש צורך לשלב את כל הדפים יחד. | כן | בוצע איחוד של כלל המסכים למסך אחד אחיד, עם ניווט בין התכנים באמצעות כרטיסיות (TABs). |
| להוסיף סטטיסטיקות (לפצל את הגרף שיציג כל חיישן בנפרד) | כן | בוצע פיצול תצוגה שמאפשר לראות את כל נתוני החיישנים בגרף אחד מרוכז. |
| הסברים כיצד לבצע פעולות | לא | התקבלה החלטה קבוצתית שלא להוסיף הסברים טקסטואליים, מתוך כוונה לשמור על ממשק DASHBOARD נקי ואינטואיטיבי שאינו דורש הדרכה נוספת. |
| העיצוב קטן בבחירת מסכים | כן | בוצע עדכון של עיצוב הממשק הכולל הגדלת המסך והפונטים, לשיפור הקריאות ונוחות המשתמש. |

**מסמך SUS של המערכת**  


**ציון SUS**

מקומות זוגים: (4-1)+(4-1)+(5-1)+(5-1)+(5-1)=18   
 מקומות אי זוגיים: (5-2)+(5-1)+(5-1)+(5-2)+(5-1) =18  
  
 חישוב סופי: (18+18)\*2.5=90  
  
 ציון ה־SUS שקיבלנו אינו מפתיע אותנו, במיוחד לאור העובדה שתחילת העבודה התאפיינה באי־בהירות באיזו תוכנה ושפת תכנות נדרשות לבניית המערכת. בעקבות שעת ייעוץ עם המרצה

נטלי והבנה שהמערכת שבנינו אינה תואמת את דרישות המשימה, החלטנו להתחיל את המשימה מחדש. שינוי זה גרר אילוצים בלוחות הזמנים, והגענו לסדנה כשאנחנו מודעים לכך שהתוצאה אינה מושלמת, אך משקפת את המקסימום שיכולנו להפיק במסגרת הזמן והנסיבות.

**מדדי הצלחת המערכת:**

### 1. אמינות ודיוק הנתונים

המערכת תשקף נאמנה את המצב האמיתי במעבדה כדי למנוע תגובה שגויה או חוסר תגובה. דיוק גבוה שומר על אמון המשתמשים ועל קבלת החלטות נכונה בזמן אמת.

* מה נמדד: עד כמה הנתונים באפליקציה תואמים את המדידות בפועל.
* איך נבדק: השוואה בין קריאות החיישנים באפליקציה למדידות ידניות.
* יעד (SLA): לפחות 99% מהקריאות צריכות להיות בטווח סטייה מותר (למשל ±0.5°C בטמפרטורה).

**2. זמינות המערכת**

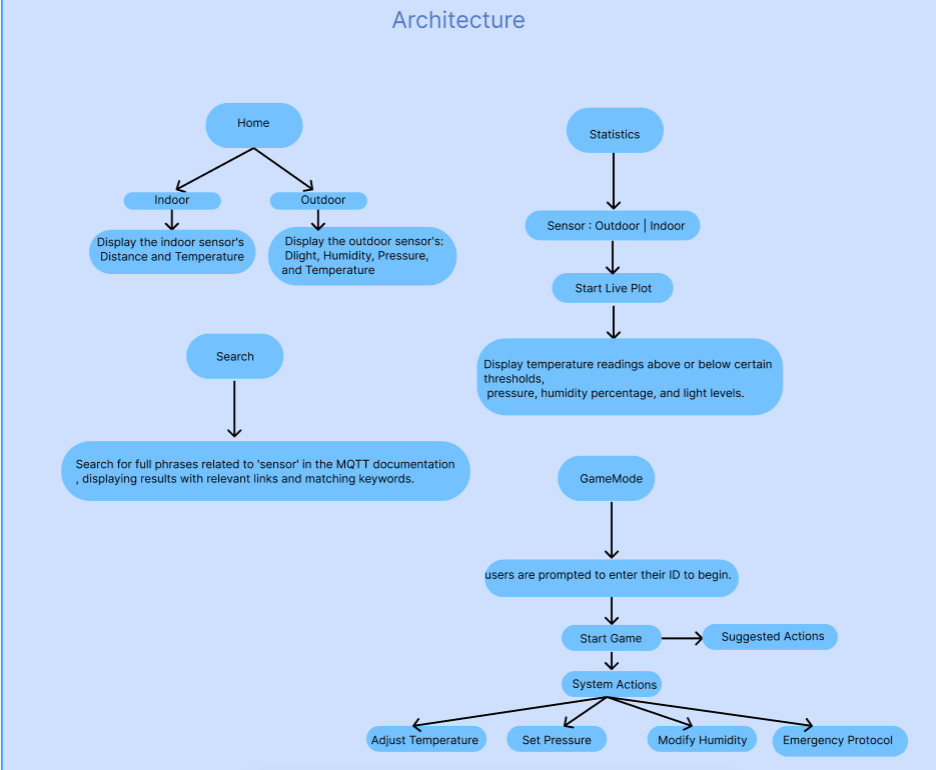
כאשר המערכת לא זמינה, לא ניתן לנטר תנאים קריטיים דבר שעלול להוביל לנזקים. לכן SLA גבוה בזמינות מבטיח המשכיות תפעולית ואמינות במעבדה.

* מה נמדד: אחוז הזמן שבו המערכת זמינה ונגישה למשתמשים.
* איך נבדק: ניטור רציף של זמן פעילות מול זמן השבתה (למשל המקסימום בחודש שהמערכת מושבתת 7 שעות).
* יעד (SLA): זמינות של לפחות 99% מהזמן החודשי.

### 3. עדכניות הנתונים

נתונים שמתעדכנים באיחור עלולים למנוע זיהוי בזמן של בעיות חמורות. עדכניות גבוהה תורמת לשקיפות, לתגובה מהירה ולמניעת כשלים במעבדה.

* מה נמדד: תוך כמה זמן נתון מהחיישן מופיע באפליקציה.
* איך נבדק: חישוב הפרש זמנים בין המדידה להצגה.
* יעד (SLA): כ95% מהנתונים יוצגו תוך עד 5 שניות.

**דיאגרמת ארכיטקטורה** 

הארכיטקטורה של המערכת שלנו היא מסוג **Layered Architecture**, אשר מחלקת את המערכת למספר שכבות לפי תחום אחריות:

שכבת הצגה (Presentation Layer) - אחראית על הצגת הנתונים למשתמש והאינטראקציה (לדוגמה: מסכי הבית, גרפים, חיפוש).

שכבת הלוגיקה העסקית (Business Logic Layer) - מטפלת בפעולות כמו ניתוח חיישנים, הפעלת מצב המשחק, ניתוח נתוני MQTT.

שכבת הגישה לנתונים (Data Access Layer) - מתקשרת עם Firestore ו-MQTT כדי לשלוף נתונים על חיישנים ולבצע שאילתות.

דיאגרמת הארכיטקטורה ממחישה את הזרימה בין שכבות אלו, כאשר כל שכבה תלויה רק בשכבה מתחתיה, דבר המקל על תחזוקה ושדרוגים.

עם זאת, ברמה הטכנית, קיים גם ממד של Client-Server – שבו ה-Client (דפדפן/ממשק) שולח בקשות ל- Firebase - אבל זהו מימוש טכני בתוך הארכיטקטורה הלוגית שהיא Layered.

| **Home** | מסך ראשי שמאגד את כלל הרכיבים של המערכת ומציג את מצב החיישנים בזמן אמת. מאפשר למנהל לעבור בקלות בין החלקים השונים (Statistics, Search, Game Mode) באמצעות כפתורי ניווט. | * render\_sensor\_status() – מציג תיבת מידע מעוצבת עם ערכי החיישנים Indoor ו־Outdoor. * fetch\_current\_sensor\_values () – קורא את הערכים העדכניים של החיישנים ממסד הנתונים (/sensor\_values). * switch\_view() – מחליף את התוכן שמוצג בהתאם ללחיצה בתפריט (לדוגמה: מעבר ל־Search). * admin\_view() – מכיל את כלל התצוגות במסך Home כולל החיישנים והטופס לצפייה בהיסטוריה. * mng\_ui() – טופס צפייה בנתוני היסטוריה לפי טווח תאריכים. * show\_history\_table() – מציג טבלת היסטוריית מדדים לפי תאריכים שנבחרו. |
| --- | --- | --- |
| **Statistics** | הצגת מידע גרפי חי ממסד הנתונים, לצד טבלה סטטיסטית של המדדים – כולל ממוצע, מקסימום, סטיית תקן ואיתור חריגות. | * live\_plot() – מריץ גרפים חיים מהחיישנים כולל חישוב אנומליות והצגת טבלה סטטיסטית. * fetch\_sensor\_data(sensor\_type='indoor') – מביא עד 15 קריאות אחרונות מחיישנים לפי Indoor/Outdoor. * create\_stats\_table(data\_dict, thresholds) – מחשב ממוצע, סטיית תקן, מקסימום, מינימום, ואנומליות לכל מדד. * thresholds – מגדיר את טווחי הערכים התקינים (למשל: טמפרטורה בין 20 ל־30 מעלות). * sensor\_dropdown – תפריט בחירה בין סוגי חיישנים. * start\_button – כפתור להפעלת הגרף החי. * graph\_output, table\_output – אזורים גרפיים להצגת גרף וטבלת סטטיסטיקה בהתאמה. |
| **Search** | לאפשר למשתמש לחפש ביטויים מתוך מסמכי אתר mqtt.org. המנוע משתמש באינדקס מילים שנבנה מראש ומחזיר קישורים לפי רלוונטיות. | search\_sentence\_links(sentence) – מחפש ביטוי שלם, מפעיל חיפוש על כל מילה ומדרג תוצאות לפי התאמה.  search\_word(word) – מחפש מופעים של מילה בודדת במסד הנתונים /word\_index.  on\_search\_click(b) – מופעל בלחיצה על כפתור החיפוש ומציג את התוצאות בצורה מעוצבת.  search\_input – תיבת הקלט של המשתמש.  search\_button – כפתור לביצוע החיפוש.  results\_output – תיבת תצוגת התוצאות.  search\_ui – מאגד את כל רכיבי החיפוש למסך אחד. |
| **Game Mode** | לאפשר למהנדסים לפעול לפי מצבים הנדסיים שונים (למשל: חריגה בטמפרטורה) באמצעות ממשק המדמה סביבה תעשייתית. כל פעולה מקנה ניקוד אישי. | display\_login\_screen() – טופס התחברות עם הזנת מזהה משתמש.  load\_points\_for\_id(user\_id) – טוען את מספר הנקודות הנוכחי של המשתמש ממסד הנתונים.  update\_points\_for\_id(user\_id, points) – שומר את הניקוד החדש של המשתמש תחת מזהה אישי.  display\_game\_mode() – בונה את מסך המשחק שכולל פעולות מערכתיות והצעות לפעולה.  make\_cb() – יוצרת דינמית כפתור, שדה קלט וכפתור אישור לכל פעולה הנדסית (כמו "Adjust Temperature").  FBconn.post('/engineerActions/...') – רושם את הפעולה שבוצעה (כולל ערך, פעולה, ניקוד) במסד.  submit\_sugg.on\_click() – שומר את ההצעות שסומנו (Suggested Actions) ומעדכן את הניקוד.  current\_id – מזהה המשתמש הנוכחי.  current\_points – סך הנקודות של המשתמש. |

### פיצ’ר: הצגת תחזית מזג אוויר לכרמיאל והקשר לחיישני Outdoor

בפרויקט הוספנו פיצ’ר פונקציונלי המציג תחזית מזג אוויר עדכנית ליום הנוכחי ולשלושת הימים הקרובים עבור העיר כרמיאל — שם ממוקמת המעבדה בה מותקנים חיישני החוץ (Outdoor sensors).

#### מטרת הפיצ’ר:

#### מטרת התחזית היא לאפשר למהנדסים ולמנהלים לצפות בתנאי מזג האוויר הנוכחיים והצפויים, ולוודא שהנתונים המתקבלים מהחיישנים החיצוניים תואמים את התנאים בסביבה החיצונית בפועל. כך ניתן לגלות חריגות אפשריות בטמפרטורה או בלחות ולהבין אם הן נובעות מהחיישן עצמו או מהשפעות חיצוניות (כגון עומס חום או רוחות חריגות).

#### כיצד ממומש בקוד:

* הנתונים נמשכים מממשק Open-Meteo API באמצעות הפונקציות:  
  + fetch\_weather\_forecastb() – תחזית ל־4 ימים קדימה.
  + fetch\_weather() – מזג אוויר נוכחי.
* התצוגה למנהל מתבצעת באמצעות:  
  + render\_weather\_widget() – קוביית מזג אוויר נוכחי.
  + render\_forecast\_widget() – תחזית יומית גרפית.
* הפיצ'רים משולבים בלוח הצד של ממשק מנהל (admin\_ui\_tabs) מתחת ללוגו ולטאבי הניווט.

#### מיקום בקוד:

הפונקציות והווידג'טים מופיעים בסוף בלוק # admin functions, ושולבו לתוך nav\_colum:

nav\_column = widgets.VBox([logo] + nav\_buttons + [weather\_widget, forecast\_widget], layout=widgets.Layout(padding='20px'))

התחזית מוצגת בזמן אמת לצד כפתורי הניווט — באופן ברור ונגיש למהנדסים.