המכללה האקדמית להנדסה בראודה

המחלקה להנדסת תוכנה - מבוא למחשוב ענן

פרויקט

קישור למחברת הגוגל קולאב (יש לוודא שהיא פומבית) -[team\_snake\_dashboard\_main.ipynb](https://colab.research.google.com/drive/1c8DLA6vJmaewEA3-UYTmrZ1ZW9Kaw6uW?usp=sharing)

קישור לתיקיית הגיט של הצוות. -

<https://github.com/NahlaAboromi/intro_to_cloud_snake_group>

חברי צוות:

|  |  |
| --- | --- |
| שם חבר הצוות | ת"ז |
| נהלה אבו רומי | 212530828 |
| מוחמד עתאמנה | 213538606 |
| שהד שלבי | 324258367 |
| פאטמה דבאח | 213675549 |
| מנאל נעמה | 325393999 |

1. מהות המוצר ומרכיביו

1.1 מודולים ופיצ'רים מעניינים

המערכת שלנו היא פלטפורמה אינטראקטיבית לניטור, ניתוח וניהול של חיישנים, צוותים ומהנדסים בסביבת תפעול חכמה. המערכת משלבת בין איסוף מידע בזמן אמת, ניתוחים סטטיסטיים מתקדמים, תפעול צוותים, מערכת ניקוד מהנדסים, ושיחה אינטראקטיבית עם צ’אטבוט.

**מודולים בפרויקט**

* **Control Center-** ניהול ומעקב אחר מערכת, צוותים ומהנדסים, כולל התראות ודוחות.
* **Search-** חיפוש מונחים באתר MQTT עם הצגת תוצאות ותדירויות.
* **Statistics-** ניתוח נתוני חיישנים, סימולציות, ודוחות ביצועים.
* **Meetings-** ניהול פגישות צוות (יצירה, עריכה, מחיקה).
* **Rates-** דירוג ומעקב ביצועי מהנדסים.
* **Teams Overview-** סקירת התקדמות וביצועי צוותים.
* **Chatbot-** מתן מענה חכם לשאלות משתמשים.

**פיצ'רים מעניינים וייחודיים:**

* **סימולטור חיזוי תקלות**, מאפשר להזין ערכים ולחזות את רמת הסיכון לפגיעה תפעולית (ציון סיכון מ־0 עד 1).
* **התראות חכמות**, זיהוי והדגשה של חריגות משמעותיות בזמן אמת.
* **לוח המלצות חכם** המספק למשתמשים הצעות פרקטיות לשיפור ביצועי המערכת או לטיפול בבעיות פוטנציאליות.
* **ניתוח שורש תקלה (Root Cause Analysis),** מזהה את הגורמים האפשריים לבעיה שהתגלתה.
* **חישוב מדד אמינות לחיישנים**  המבוסס על דיוק, יציבות וזמן תגובה. עבור כל חיישן מחושב ציון כולל המאפשר לזהות חיישנים לא תקינים או כאלה שדורשים תחזוקה
* **מנגנון ניתוח מגמות**  של נתוני חיישנים — הן בזמן אמת בטווח של שעה אחרונה, והן בטווח ארוך של שבוע.
* **דו"ח סיכום תקלות שבועי** שמרכז את כלל הקריאות החריגות מכל החיישנים. בכל שבוע מחושבים אחוזי החריגה של כל פרמטר, והמערכת מציעה המלצות אוטומטיות בהתאם.
* **גרף ביצועי צוותים שבועי,** מציג לכל יום בשבוע את מספר התקלות שזוהו לעומת מספר המשימות שבוצעו, ומחשב את רמת הביצועים הכללית של הצוותים.
* **מערכת לניהול צוותים ומהנדסים**, הכוללת יצירה של צוותים, הוספת ומחיקת מהנדסים, הקצאת משימות עם ניקוד מוגדר מראש.
* **דירוג ביצועי מהנדסים** **וצוותים**, הערכת ביצועי מהנדסים וצוותים, כולל ציונים, אחוזי התקדמות ומהנדס מוביל.
* **ניהול פגישות צוותים**, הכולל יצירה, צפייה, עריכה ומחיקה של פגישות עתידיות, עם סינון לפי תאריכים.
* **מנוע חיפוש ייעודי** למונחים טכניים מהאתר MQTT.org, המאפשר חיפוש לפי מילים או ביטויים, ומחזיר מידע על מספר הופעות וקישורים לדפים רלוונטיים. הפיצ’ר כולל גם הסבר על דרך הפעולה של מנוע החיפוש, למען שקיפות וקלות שימוש.
* **צ'אטבוט אינטראקטיבי**, המאפשר שאילת שאלות חופשיות על המערכת — כגון התנהגות חיישן, פירוש שגיאות, מידע על צוותים ועוד. הצ'אטבוט משתמש במידע פנימי מהמערכת כדי לספק מענה רלוונטי ומיידי.

בניגוד לתכנון הראשוני שצוין בתרגיל בית 1, **בחרנו לוותר על תצוגה אחידה של כל החיישנים במסך אחד**, ובמקום זאת יישמנו גרפים ממוקדים עם Dropdown לבחירת סוג החיישן – לתצוגה ברורה ונוחה יותר.

**גם צפייה היסטורית לפי תחנה ושעה לא יושמה במדויק** – העדפנו מבט כולל על השבוע האחרון עם גרפים מסכמים ודוח חריגות שבועי שמדגישים תובנות ולא רק פרטים נקודתיים.

1.2 microservices שנעשה בהם שימוש

המערכת מבוססת על שלושה מיקרוסרוויסים עיקריים, כאשר כל אחד מהם אחראי על שלב מרכזי בתהליך החיפוש והצגת התוצאות:

* **IndexService** – שירות זה מטפל בקליטת המסמכים ובבניית אינדקס הפוך, אשר מאפשר מיפוי מדויק של המילים המופיעות בכל מסמך יחד עם תדירות הופעתן. כך מושגת מהירות ויעילות בחיפוש.
* **QueryService** – שירות זה מקבל את מונחי החיפוש מהמשתמש, מבצע שאילתות על בסיס האינדקס ומחשב את תדירות ההופעה של כל מונח במסמכים המתאימים.
* **ResultService** – שירות זה מארגן ומעצב את תוצאות החיפוש להצגה ברורה ונוחה למשתמש, כולל פירוט של תדירויות ההופעה וקישורים ישירים למסמכים הרלוונטיים.

1.3 KPI מרכזיים

### **Performance – Average Page Load Time PLT** :

Average Page Load Time PLT הוא הזמן הממוצע שלוקח לדף אינטרנט להיטען באופן מלא בדפדפן המשתמש. כאשר השגת PLT של 1.5 שניות מצביעה על אופטימיזציה מוצלחת של ביצועי המערכת. זהו מדד חשוב בחוויית המשתמש, שכן טעינה איטית של דפים עלולה לגרום לעיכובים בקבלת המידע הקריטי ולהפחית את היעילות.

במערכת **InsightDash**, מדד PLT הוא מדד קריטי משום שהמערכת מציגה Dashboards, גרפים והתראות בזמן אמת, המשמשים את המהנדסים לזיהוי מהיר של חריגות בקווי הייצור. טעינה מהירה מבטיחה שהמשתמשים יקבלו את המידע החיוני באופן מיידי גם בזמני עומס, מה שתורם לרציפות תפעולית, אמינות וזמינות גבוהה של השירות. שמירה על PLT נמוך מאפשרת למנוע עיכובים קריטיים העלולים לפגוע בתהליכי הייצור ובקבלת החלטות בזמן אמת.

### **High Availability – Availability Rate:**

### זמינות גבוהה (High Availability) נמדדת על פי **Availability Rate** אחוז הזמן שבו המערכת זמינה לשימוש עבור המשתמשים לאורך פרק זמן מוגדר. השגת Availability Rate של 99.99% מבטיחה שירות רציף וללא הפסקות משמעותיות, גם בתקופות עומס גבוה, ומבטאת מערכת יציבה ואמינה.

מדד Availability Rate הוא קריטי בפרויקט שלנו, כיוון שהמערכת חייבת להבטיח גישה רציפה למידע החיוני בכל רגע נתון, במיוחד במצבי עומס ובזמן ניטור חריגות בתהליכי הייצור. המהנדסים משתמשים במערכת לצורך קבלת החלטות מיידיות, ולכן כל השהייה בגישה עלולה לגרום לעיכובים בתגובה ולאובדן שליטה על איכות התהליכים. זמינות גבוהה מאפשרת שמירה על רציפות תפעולית ותגובה בזמן אמת לכל אירוע חריג.

### **Reliability – System Uptime :**

### אמינות (Reliability) נמדדת על פי **System Uptime** אחוז הזמן שבו המערכת פעילה וזמינה לשימוש באופן תקין לאורך פרק זמן מוגדר. השגת System Uptime של 99.99% מצביעה על מערכת יציבה במיוחד, אשר חווה פחות משעה אחת של השבתה במהלך שנה שלמה. זהו מדד חשוב מאוד לאמינות המערכת, שכן זמינות נמוכה עלולה לגרום לאי יכולת לגשת לנתונים חיוניים ולפגוע בתפקוד השוטף של המשתמשים.

במערכת שלנו **InsightDash**, מדד System Uptime הוא מדד קריטי מכיוון שהמערכת מספקת Dashboards, גרפים והתראות בזמן אמת, בהם המהנדסים נעזרים לזיהוי וטיפול בחריגות בקווי הייצור. זמינות גבוהה מבטיחה שהמשתמשים יוכלו להסתמך על המערכת לקבלת המידע הנחוץ בכל רגע נתון, וכך נשמרת רציפות תפעולית, אמינות וזמינות השירות גם במצבי עומס או תקלות. זמינות זו מאפשרת למהנדסים לפעול בזמן אמת ולמנוע כשלים שעלולים לפגוע בייצור ובאיכות התהליכים.

### **Maintainability Code Maintainability Index :**

תחזוקה (Maintainability) נמדדת על פי **Code Maintainability Index** מדד מספרי המעריך את רמת תחזוקת הקוד על סמך פרמטרים כמו מורכבות, בהירות, כפילויות ואיכות הכתיבה. השגת ציון של 90 במדד זה מצביעה על קוד נקי, ברור וקל לתחזוקה לאורך זמן.

במערכת **InsightDash**, מדד Code Maintainability Index הוא מדד קריטי מכיוון שהמערכת מבוססת על עדכונים ושדרוגים מתמידים של דשבורדים, גרפים ורכיבי ממשק בזמן אמת. קוד שקל לתחזק מבטיח יכולת להוסיף תכונות חדשות, לתקן באגים במהירות ולשמור על יציבות המערכת גם כאשר דרישות הייצור והמשתמשים משתנות. כך אנו יכולים להבטיח שמערכת InsightDash תמשיך לספק שירות איכותי ואמין לאורך זמן.

1.4 ניתוח נתוני עתק

במהלך העבודה עם נתוני הפרויקט, התמודדנו עם כמויות גדולות של רשומות שהצטברו משתי תחנות עיקריות פנימית וחיצונית (Indoor ו Outdoor) ומסוגים שונים של חיישנים, כמו טמפרטורה, לחות, לחץ, מרחק . הנתונים נשמרו ב Firebase במבנה היררכי ומסודר, כך שהמידע מכל תחנה (Indoor ו Outdoor) ומכל סוג חיישן נשמר בצורה ברורה ומופרדת. המבנה שבו השתמשנו איפשר לארגן את הנתונים באופן גמיש, להתאים את האחסון לסוגי מידע מגוונים ולהמשיך לצבור נתונים לאורך זמן מבלי להידרש למבנה קשיח. הגישה הזו גם אפשרה להתמקד בכל פעם בחלק הרצוי של הנתונים לצורך העיבוד והניתוח, בדומה לאופן שבו מנוהלים נתונים במאגרים מרכזיים המיועדים לאיסוף ושמירה של כמויות גדולות של מידע ממקורות שונים.

במהלך העבודה במערכת יישמנו תהליך עיבוד בזמן אמת **Real Time Analytics** שאיפשר לזהות תקלות ומגמות מיד עם הופעתן. הנתונים שהתקבלו מהחיישנים הועברו לדשפורד הראשי ו לדף הסטטיסטיקה כך שהמידע הוצג בצורה עדכנית ורציפה, מה שאיפשר לעקוב אחר ערכים חריגים של חיישנים או ביצועים יוצאי דופן בתחנות העבודה תוך כדי פעילותם. התהליך כלל שליפה שוטפת של הנתונים, עיבוד מהיר באמצעות חישובים סטטיסטיים בזמן אמת והצגת ההתראות והמגמות באופן מיידי, מה שתמך באיתור חריגות וקבלת החלטות בזמן אמת.

כחלק מתהליך העבודה, הנתונים שעובדו הוצגו בצורה גרפית ברורה ומובנת. נעשה שימוש בגישה של **Data Visualization**, שבאה לידי ביטוי ביצירת גרפים אינטראקטיביים ודוחות צבעוניים שאיפשרו להבין את המידע במהירות ובקלות. הנתונים הוצגו באמצעות גרפי עמודות, גרפי קו, boxplots שהמחישו את מגמות החיישנים, הפיצול בין תחנות indoor ו outdoor, התפלגות הערכים והחריגות לאורך זמן. השילוב בין הכלים הוויזואליים והנתונים המעובדים סייע למשתמשים במערכת לזהות דפוסים, מגמות חריגות והזדמנויות לשיפור בזמן קצר ובאופן אינטואיטיבי.

בנוסף, בוצע ניתוח נתונים בגישת **MapReduce**, באמצעות השימוש ב־PySpark לצורך עיבוד כמויות גדולות של מידע. הנתונים חולקו לקבוצות לפי מאפיינים משותפים, לדוגמה לפי יום בשבוע, ובכל קבוצה חושבו ממוצעים וסטיות תקן לערכי החיישנים השונים. התהליך כלל חלוקה ראשונית של הנתונים (Map), עיבוד וסיכום סטטיסטי של כל קבוצה (Reduce), ולאחר מכן המרה לגרפים שהמחישו את המגמות והתבניות לאורך זמן. גישה זו אפשרה לנתח את המידע בצורה יעילה ולהפוך את הנתונים הגולמיים לתובנות שימושיות.

בסופו של התהליך, השילוב בין שיטות העיבוד וההצגה השונות תרם להפקת תובנות מעמיקות מהמידע שנאסף. כך הצלחנו להפוך את הנתונים הגולמיים לכלי משמעותי לקבלת החלטות מבוססות מידע.

2. ארכיטקטורת המערכת

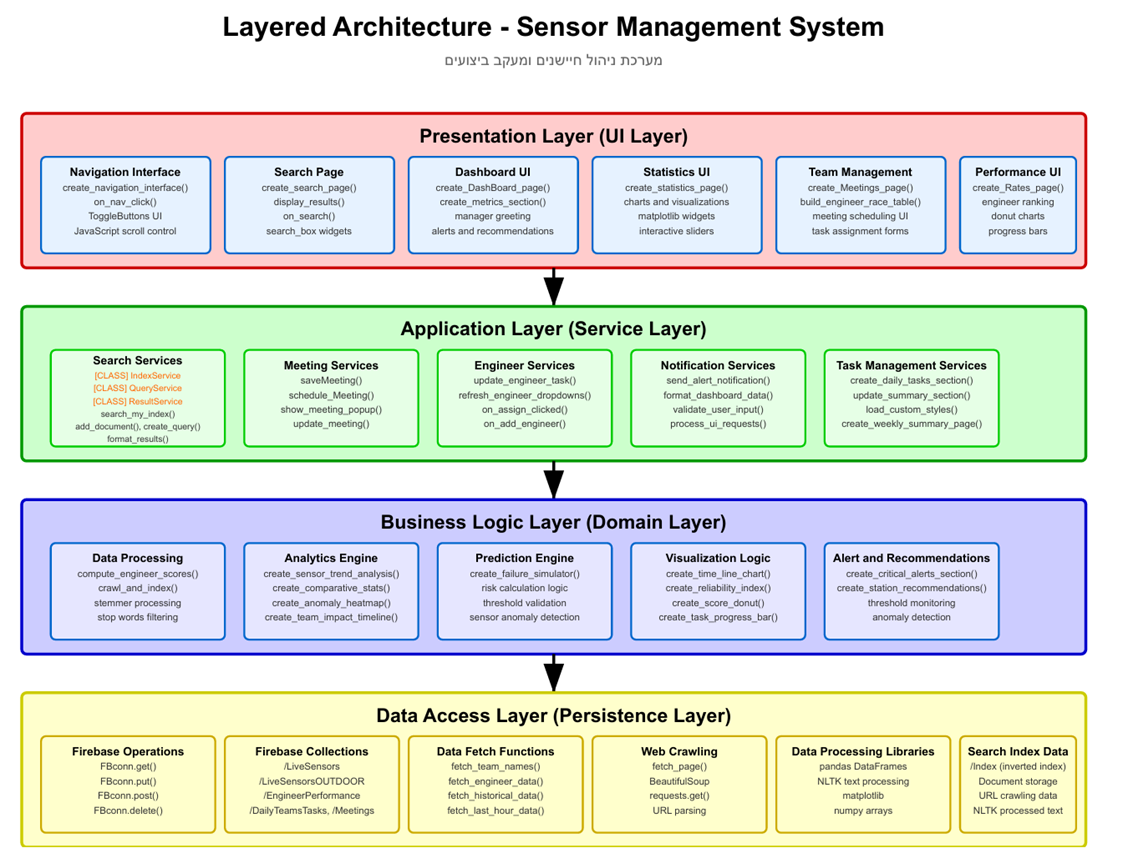
2.1 תרשים ומאפיינים מרכזיים (מעודכן)

בפרויקט שלנו השתמשנו בארכיטקטורת **Layered Architecture Pattern** (שכבות). כל שכבה אחראית על תחום אחר במערכת, מה שמאפשר הפרדה בין הלוגיקה העסקית, השירותים, ממשק המשתמש והנתונים עצמם. ככה קל יותר לנהל, לתחזק ולהרחיב את המערכת בעתיד.

**שכבת הממשק (UI Layer):** אחראית על התקשורת הישירה עם המשתמש והצגת המידע. מאפשרת אינטראקציה נוחה עם המערכת באמצעות רכיבי ממשק שונים. כל הפעולות הגלויות למשתמש מתבצעות דרכה.

**שכבת השירותים (Service Layer):** מתאמת בין ממשק המשתמש ללוגיקה העסקית. אחראית על ביצוע הפעולות הנדרשות ועיבוד הבקשות שמגיעות מה-UI. מבטיחה זרימה תקינה של הנתונים בין השכבות.

**שכבת הלוגיקה העסקית (Business Logic Layer):** מכילה את כלל החישובים, העיבודים וההחלטות של המערכת. אחראית על ניתוח הנתונים, עיבוד מידע וגיבוש תובנות עסקיות. מהווה את הליבה האינטליגנטית של המערכת.

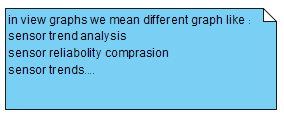
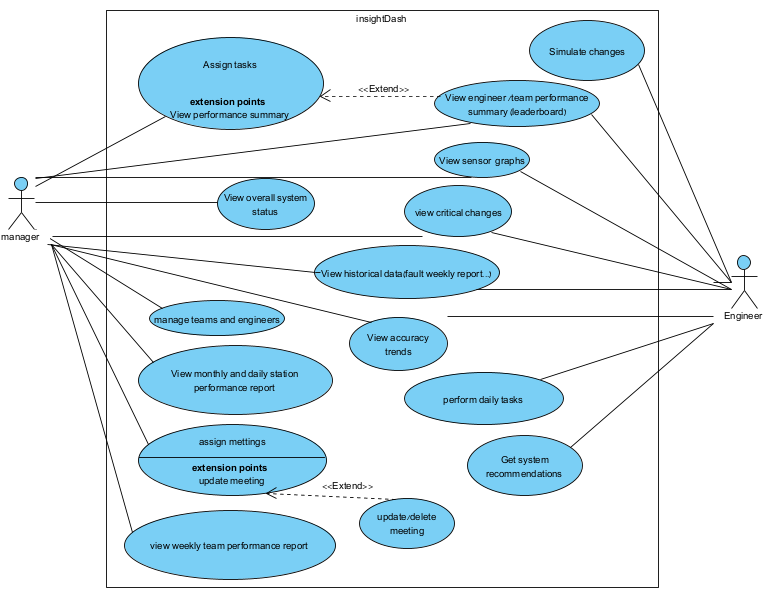
**שכבת הגישה לנתונים (Data Access Layer):** מנהלת את הגישה למקורות הנתונים ושומרת על תקינות הקריאות והכתיבות. אחראית על שליפת נתונים ואחסונם באופן מאורגן ובטוח. מקשרת את המערכת אל בסיסי הנתונים החיצוניים.

### **מאפיינים מרכזיים של הארכיטקטורה**

* **הפרדה לרמות אבסטרקציה שונות:** כל שכבה פועלת על רמת פירוט שונה ומטפלת בתחום אחר במערכת.
* **העברת שירותים בין שכבות:** כל שכבה מספקת שירות לשכבה שמעליה וצורכת שירותים מהשכבה שמתחתיה.
* **מודולריות:** ניתן להחליף, לשפר או לעדכן שכבה בודדת מבלי להשפיע על שאר המערכת.
* **תחזוקה פשוטה וניהול קוד נוח:** החלוקה הברורה בין השכבות מאפשרת תחזוקה קלה וזיהוי מהיר של תקלות.
* **שימושיות וגמישות:** הארכיטקטורה מתאימה למערכות שונות וניתן להוסיף יכולות חדשות בקלות.
* **ניהול נתונים עקבי ומאובטח:** שכבת הגישה לנתונים מבטיחה גישה מבוקרת, עקבית ובטוחה למקורות המידע.

לצורך נוחות הקריאה ניתן לגשת לתרשים דרך קישור ה־GitHub הבא:

<https://github.com/NahlaAboromi/intro_to_cloud_snake_group/blob/main/System_Architecture_Diagram_InsightDash.pdf>

2.2 Use case מעודכן של פונקציונליות המערכת.

2.3 דרישות פונקציונליות

**1.** המערכת מאפשרת צפייה בנתונים בזמן אמת ממגוון חיישנים.

**2.** המערכת מאפשרת קבלת התראות בזמן אמת במקרה של תקלה או חריגה מערכי סף.

**3.** המערכת מאפשרת לסנן נתונים לפי תחנה מסוימת, לצורך מיקוד הניתוח.

**4.** המערכת מאפשרת להציג דירוג מהנדסים לפי מספר הנקודות שצברו.

**5.** המערכת מאפשרת למנהלים להקצות למהנדסים משימות יומיות.

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

|  |  |
| --- | --- |
| **קטגוריה** | **אתגר / פתרון** |
| **Availability** | פיצול קליטת MQTT למחברות נפרדות כדי להבטיח זמינות גבוהה גם בזמן עומס. |
| **Maintainability** | חלוקת הקוד לפונקציות קטנות, שימוש במיקרו-שירותים, קוד קריא ומודולרי. |
| **Usability** | פיצול מסכים, תפריט ניווט קבוע, הסברים, צבעים ברורים – שיפור חוויית המשתמש. |
| **Performance** | שמירה על PLT נמוך, שימוש בגרפים דינמיים וטעינה חכמה. |
| **Reliability** | ניהול עומסים ב-Firebase, ארגון היררכי של הנתונים, מפתחות אחידים. |
| **Transparency** | הצגת תהליך החיפוש למשתמש (Stemming, תדירות, לוגיקה). |
| **Scalability** | מבנה נתונים גמיש, שימוש ב-PySpark לניתוח כמויות מידע גדולות. |

4. ביקורת עמיתים אשר ניתנה במהלך הסטודיו (שבוע 9) , וכיצד התמודדתם איתה. נא לחשב גם את ציון SUS ולהתייחס אליו.

|  |  |  |
| --- | --- | --- |
| **הערת משוב** | **האם בוצע שינוי?** | **נימוק** |
| **פונט בהיר** | כן | שונה לפונט כהה וברור יותר לטובת קריאות ונגישות. |
| **גרפים מבולגנים** | כן | פוצלו לגרפים נפרדים, עיצובם שופר. |
| **עיצוב** | כן | שופר העיצוב הכללי – אחידות צבעוניות ושיפורי קריאות. |
| **כמות מופעים ליד כל תוצאה במנוע חיפוש** | כן | נוספה תצוגת ספירה ליד כל תוצאה לחיפוש, להגברת הרלוונטיות. |
| **Navigation bar** | לא | כבר קיים Navbar פעיל, ולכן לא נדרש שינוי. |
| **דפים עמוסים** | כן | פוצלו לדפי Rates ו‑Teams – לצמצום עומס וניווט נוח יותר. |
| **קושי בהבנה** | כן | הוספו הסברים טקסטואליים ויזואליים, ייעול תוכן וניווט. |
| **הרבה כפתורים ללא הסברים** | כן | נוסף טקסט עזר ליד כל כפתור עיקרי. |
| **צבעים לא בולטים לחריגות** | כן | הדגשת חריגות בצבעים בולטים לשיפור זיהוי. |
| **צריך לפצל מסך עם הרבה גרפים** | כן | המסך נחלק לשניים + נוספה אפשרות בחירה דרך Dropdown לצפייה ממוקדת. |
| **חלון סטטיסטיקה חסר** | לא | כבר קיים, לא נדרש שינוי. |
| **Navbar (שופרו)** | כן | ה־Navbar הקיים שופר לשמירה על עקביות ממשק משתמש. |

לגבי ציון SUS :

מספר הסטודנטים שמילאו את כל 10 שאלות ה **SUS** : הוא 39 אז נחשב הממוצע שלהם :

סכום=50+35+50+50+50+100+50+100+70+82.5+95+60+85+32.5+50+50+52.5+50+72.5+50+92.5+95+70+85+90+100+47.5+67.5+97.5+97.5+100+90+100+82.5+100+100+100+100+77.5=2927.5

אז ממוצע 

לאחר חישוב ציוני ה SUS לכל אחד מהמשיבים, התקבל סכום כולל של 2927.5 עבור 39 סטודנטים שמילאו את השאלון. חישוב הממוצע (2927.5 חלקי 39) נותן ציון ממוצע של 75.06.

ציון זה גבוה מהממוצע העולמי המקובל לשאלון SUS, שעומד על 68. משמעות הדבר היא שרמת השמישות הנתפסת של המערכת על ידי המשתמשים גבוהה, והם חוו את המערכת כנוחה, ידידותית וקלה לשימוש. ציון כזה מעיד על כך שהמערכת עונה היטב על הצרכים של המשתמשים, ומספקת חוויית שימוש חיובית.

ככל שציון ה SUS גבוה יותר, כך ניתן להסיק שהמערכת שמישה יותר בעיני המשתמשים. במקרה זה, ציון 75.06 מציב את המערכת ברמה טובה מאוד מבחינת שמישות, ומרמז על כך שאין בעיות עיקריות בחוויית המשתמש.

5. שקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?

בחרנו להציג למשתמשים בצורה שקופה את כל שלבי העיבוד של מנוע החיפוש. המשתמש יכול להבין בדיוק איך השאילתה שלו מתורגמת, מה קורה למילים שהוא מקליד, ואיך מתבצא תהליך ההתאמה לתוכן הקיים במערכת.

* **פירוק מילים**: השאילתה מתפצלת למילים בודדות כדי לבדוק כל מילה בנפרד.
* **השוואה בלי תלות באותיות גדולות/קטנות**: "המערכת הופכת את כל האותיות ל־lowercase – כדי ש'Cat' ו־'cat' יתאימו."
* **סטמינג**: המילים עוברות תהליך שנקרא Stemming – כמו להפוך 'running' ל־'run'. זה עוזר למצוא התאמות גם כשיש הטיות מילים.
* **AND logic**: המערכת מחפשת רק מסמכים שמכילים את כל המילים – לא מספיק מילה אחת.
* **תדירות**: אנחנו מחשבים כמה פעמים כל מילה מופיעה במסמך – זה נותן ערך לכמה תוכן רלוונטי.
* **תוצאה עשירה**: המשתמש מקבל לא רק קישור – אלא גם כותרת, מקור, מספר התאמות, ותדירויות.

**למה זה חשוב:**

שקיפות כזו מגבירה אמון. המשתמש לא מקבל תוצאה 'קסומה' – אלא מבין את ההיגיון מאחורי המנוע, מה שעוזר לו לשפר את החיפוש בעצמו.

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

* במהלך העבודה על הפרויקט נתקלנו באתגר משמעותי בשילוב רכיב קליטת נתוני החיישנים בזמן אמת עם ממשק המשתמש הגרפי ב-Google Colab. מאחר שסקריפטי ה MQTT לקליטת נתונים מהחיישנים פועלים בלולאות אינסופיות (loop\_forever()), לא ניתן היה להריץ אותם כחלק מהמחברת הראשית מבלי לשתק את הממשק האינטראקטיבי של המשתמש – אחת ממגבלות Colab הידועות, שאינה מאפשרת הרצת תהליכים ממושכים או מקבילים באותה מחברת. כדי להתמודד עם אתגר זה, פיצלנו את המערכת למחברות נפרדות: מחברות ייעודיות אחראיות לקליטת נתוני החיישנים, העלאתם בזמן אמת ל-Firebase, בעוד שהמחברת הראשית מתמקדת בניהול הדשבורד, הוויזואליזציה והאינטראקציה עם המשתמש. מבנה זה אפשר לנו להבטיח זרימה תקינה של נתונים חיים, להציג ויזואליזציות בזמן אמת, ולשמור על מערכת אינטראקטיבית, יציבה ונוחה לשימוש – למרות מגבלות סביבת Colab.
* אתגר נוסף שבו נתקלנו במהלך העבודה היה מגבלת נפח האחסון של Firebase Realtime Database. החיישנים פעלו לאורך כל שעות הבוקר ואספו כמויות גדולות של נתונים חיים, שנשמרו ב-Database. רק בשעות הצהריים שמנו לב לפתאום לחסימה בכתיבה ל-Firebase – לאחר חקירה מצאנו כי עברנו את גבול האחסון המותר במסגרת המנוי שלנו. מגבלה זו גרמה לעצירת קליטת נתונים חדשים ולשיבוש זמני בפעולת המערכת.

כדי להתמודד עם האתגר, נדרשנו לעקוב באופן שוטף אחרי ניצול המשאבים של ה-Database, למחוק נתונים ישנים שאינם נחוצים.

* העבודה עם כמויות גדולות של נתוני חיישנים (כולל חיישני indoor ו outdoor, נתוני צוותים , ביצועי מהנדסים ועוד) יצרה עומס על מסד הנתונים ודרשה ארגון נכון של הנתונים כדי למנוע עיכובים בשליפה. כדי להתמודד עם האתגר הזה, ארגנו את הנתונים במבנה היררכי ברור תחת קיבוצים ייעודיים כמו /LiveSensors ו /EngineerPerformance, מה שאיפשר שליפות ממוקדות ויעילות יותר. בנוסף, הפרדנו את נתוני החיישנים לשני NODE ים נפרדים: /LiveSensors עבור חיישנים פנימיים (indoor) ו /LiveSensorsOUTDOOR עבור חיישנים חיצוניים (outdoor). נעשה גם שימוש במפתחות אחידים המבוססים על timestamp, שאיפשרו ניהול מסודר של הנתונים והקלו על פעולות הסינון והשליפה.
* הצורך להציג את הנתונים בצורה ברורה, אינטראקטיבית ונוחה לשימוש יצר אתגר משמעותי, במיוחד כאשר היה עלינו לשלב גרפים, דוחות, כרטיסיות מהנדסים ודוחות התראות חכמות וכל זאת תוך שמירה על זמני טעינה סבירים גם כשכמות הנתונים גדלה. אתגר זה דרש תכנון מוקפד של הממשק והכלים שבהם נשתמש. כדי להתמודד עם האתגר, יצרנו ממשק ויזואלי עשיר ודינמי באמצעות שילוב של כלים כמו ipywidgets, matplotlib ו HTML מותאם אישית. כך הצלחנו לפתח גרפים אינטראקטיביים ודוחות צבעוניים שניתן לעדכן בזמן אמת, לייצר טבלאות עם הדגשת ערכים חריגים, ולהציג תרשימים מתקדמים כמו donut charts בצורה אסתטית ונגישה. כל אלה תרמו לשיפור חוויית המשתמש ולאפשרות לנתח את הנתונים בקלות וביעילות.
* ניהול הקוד במסגרת הפרויקט יצר אתגר ממשי, בעיקר בשל הצורך לשלב רכיבים רבים ושונים כמו מנוע חיפוש, ניהול נתוני חיישנים ולוח מחוונים באותו פרויקט. המורכבות דרשה לארגן את הקוד כך שיהיה קל להבנה, לתחזוקה ולשדרוג, מבלי שהחלקים השונים יתערבבו אלה באלה. כדי להתמודד עם האתגר הזה, בשלב ביצוע משימת ה Index בחרנו לבנות את מנוע החיפוש במבנה של מיקרו-שירותים ייעודיים, כך שלכל רכיב יש תפקיד ברור לדוגמה, IndexService שמנהל את האינדקס, QueryService שאחראי על ביצוע השאילתה ו ResultService שמעבד את התוצאות. בנוסף, בנינו את הקוד כך שכל חלק יתמקד בפונקציות ממוקדות וברורות, מה שהקל על התחזוקה, צמצום השגיאות והוספת פיצ'רים חדשים לאורך הדרך.

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

מאחר ותיק התחזוקה כולל פירוט מלא על הקבצים, האובייקטים המרכזיים והתיעוד של הפונקציות, הוא צורף כקובץ נפרד.

ניתן לעיין בו בקישור הבא:

[Developer\_Guide\_Package.docx – תיק תחזוקה](https://github.com/NahlaAboromi/intro_to_cloud_snake_group/blob/main/Developer_Guide_Package.docx)

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

תיק המשתמש כולל הסבר כללי על המערכת, פירוט מסכים, מעברים והסבר על תקלות אפשריות.  
 מאחר והמסמך מפורט וארוך אז גם הוא צורף כקובץ נפרד.  
 ניתן לעיין בו בקישור הבא:  
 [תיק משתמש.docx – מדריך למשתמש](https://github.com/NahlaAboromi/intro_to_cloud_snake_group/blob/main/%D7%AA%D7%99%D7%A7%20%D7%9E%D7%A9%D7%AA%D7%9E%D7%A9.docx)

התיחסות למשובים שקיבלנו:

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| צבעים בהירים יותר | לא | קיבלנו יותר משובים חיוביים מאשר שליליים בנוגע לעיצוב. |
| צ'אטבוט | כן | ביצענו שיפור בהתמקדות התשובות של הצ'אטבוט כדי שיהיה מדויק ורלוונטי יותר. |
| מסכים מאוד עמוסים.  מסכים מאוד ארוכים ודורשים מלא גלילה. | לא | הערה זו התקבלה בעבר ובעקבותיה פיצלנו חלק מהמסכים. לגבי מסך הסטטיסטיקות – בחרנו לרכז את כל הנתונים בדף אחד לצורך שלמות המידע. |
| איפה המשחקיות? | לא | אלמנטים של משחקיות **קיימים כבר**, וניתן למצוא אותם במסך ה־**Rates** ו־**Team Rates**. |
| גרפים לא ברורים או שימושיים במסך מנהל. | כן | הוספנו הסבר קצר מעל כל גרף שמבהיר מה הוא מציג, כדי לשפר את ההבנה והשימושיות. |

11. התייחסות למשוב שקיבלתם בשבוע 13 – code review – יש להגיש את המשוב שקיבלתם מהצוות השני בחלק הקבוצתי , ולכתוב התייחסות מתאימה:לכל הערה, יש לרשום אותה ולציין האם נעשה שינוי בקוד בעקבות זאת.

|  |  |
| --- | --- |
| **הערה** | **האם בוצע שינוי בעקבות ההערה?** |
| **פשטות -** הוספת פונקציות עזר לקוד וארגון מחדש של קוד הצ’אט־בוט יכולים לשפר את הפשטות. | כן. הקוד פוצל למספר תאים (Cells), כאשר כל תא מכיל את המימוש של מסך נפרד. בנוסף, הקוד כבר היה קריא, מחולק לפונקציות עזר, וללא כפילויות – ולכן לא נדרש שינוי נוסף מבחינת הפשטות. |
| **מימוש -** הקוד עומד בדרישות ומבצע את הנדרש. | לא. הקוד כבר עמד בדרישות וביצע את כל הפעולות הנדרשות, ולכן לא היה צורך בהתאמות נוספות. |
| **מודולריות -** מומלץ לחלק את הקובץ הגדול למחברות נפרדות על מנת לממש micro service | כן. נדרשנו לממש את הפרויקט במחברת יחידה, ולכן לא בוצע פיצול למחברות נפרדות. עם זאת, כן חילקנו את הקוד לתאים (Cells) לשם שיפור המבנה, הקריאות והארגון. |
| **יעילות -** מומלץ להציג את הנתונים בצורה יעילה יותר יותר מידי גרפים | כן. הוספנו תיאור מילולי מעל כל גרף כדי להסביר מה הוא מייצג, במטרה להנגיש את המידע בצורה יעילה וברורה יותר. |
| **באגים -** לא נצפו באגים. | לא. לא נתגלו באגים, ולכן לא נדרשה כל התערבות. |
| **טיפול בשגיאות-** יש טיפול חלקי, אך לא שלם. | לא, לא בוצע שינוי, מכיוון שכבר קיים טיפול בכל מקרי השגיאה. |

12. מקורות. יש לצטט את המקומות מהם אתם לוקחים את הנתונים השונים. יש לכלול פרומפטים לכלי AI, במידה והשתמשתם בהם.

* חומרי הקורס: הרצאות 1–10 בקורס Cloud Computing + תרגולים
* תרגילי בית שעשינו במהלך הסמסטר
* A Detailed Exploration with KPIs and Examples - <https://medium.com/@awadhesh.p.gupta/unveiling-non-functional-requirements-in-web-applications-a-detailed-exploration-with-kpis-and-8d152544f9a1>
* Non-functional requirement - Wikipedia
* <https://en.wikipedia.org/wiki/Non-functional_requirement>
* chatGPT : ארכיטקטורת שכבות <https://chatgpt.com/share/68514414-7264-8006-93a7-2c3348767ca9>
* chatGPT : שקיפות אלגוריתמית <https://chatgpt.com/share/685146c2-5444-8006-9a36-b5e620fb4080>
* נתוני עתק <https://he.wikipedia.org/wiki/Big_data>