המכללה האקדמית להנדסה בראודה A black background with purple letters

Description automatically generated

תרגיל בית מסכם – מחשוב ענן צוות **WOLF**

מגישים:

* **יובל כוגן - 207840042**
* **יניב בודאגה – 314700428**
* **ליאור דגש – 314855404**
* **אלמוג רז - 206458911**
* **רון סלמה – 207475146**
* **דניאל לחמקוב - 319041059**

קישור למחברת הקוד הסופי:[**HW3\_Notebook**](https://colab.research.google.com/github/KoganTheDev/cloud-computing-project/blob/main/HW3_WOLF.ipynb)

קישור ל-GitHub של הפרויקט:[**Project's git repository**](https://github.com/KoganTheDev/cloud-computing-project/tree/main)

יש להגיש את הפרויקט בדו"ח אשר אורכו לא יעלה על 15 עמודים , בכתב Arial 12. יש להתייחס לנקודות הבאות:

1. **מהות המוצר ומרכיביו:**

המערכת נועדה לעזור בניהול מעבדת הרובוטיקה, על ידי איסוף, ניתוח והצגה של נתונים חשובים מהמערכת הרובוטית. היא כוללת כלים לטעינת נתונים, סינון וניתוח שלהם, והפעלת אלגוריתמים של למידת מכונה לצורך זיהוי תקלות או חריגות. בנוסף, המערכת מציגה את המידע בצורה ברורה באמצעות גרפים ותוצאות ניתוח, כדי לאפשר למפעילים ולחוקרים להבין טוב יותר את מצב המערכת ולקבל החלטות מושכלות.

**מרכיבי המערכת כוללים:**

* טעינת נתונים גולמיים ממקורות שונים
* עיבוד מקדים של הנתונים (ניקוי, סינון, המרת פורמטים)
* ניתוח גרפי של המידע לצורך הבנה ויזואלית של החיישנים בשטח.
* בנייה והפעלה של מודלים.
* מדידת ביצועים והצגת תוצאות בצורה ויזואלית.
* ממשק נוח למשתמש להפעלה ולבדיקה של תרחישים שונים
  1. **מודולים ופיצ'רים מעניינים:**

### הסבר קצר על הספריות בהן השתמשנו:

**Matplotlib**  
שימשה אותנו ליצירת גרפים ודיאגרמות של נתוני החיישנים שנשלפים מה־Firebase. הגרפים מאפשרים להציג נתונים היסטוריים בצורה ויזואלית, לבצע ניתוח מגמות ולעקוב אחרי שינויים לאורך זמן.

**NLTK**  
ספריית עיבוד שפה טבעית (Natural Language Toolkit). נעזרנו בה לצורך ניתוח טקסטים שהורדנו מהאתר mqtt.org. באמצעותה ביצענו ניקוי טקסטים, הסרה של מילות עצירה (stop words) והחלת אלגוריתם Stemming (באמצעות PorterStemmer), כדי לבנות אינדקס מונחים יעיל לחיפוש.

**ipywidgets**  
אפשרה לנו ליצור ממשק משתמש אינטראקטיבי בתוך Google Colab. באמצעותה בנינו כפתורים, טפסים, טבלאות ושליטה דינמית בתצוגות — כך שהמשתמש יכול להפעיל פונקציות שונות במערכת ישירות מה־Notebook בצורה נוחה ומבלי לכתוב קוד.

**פיצ'ר מעניין:**

התוספת: **Alert** שמוצג במסך מנהל במידה ונמשך מידע חריג מהחיישנים, לדוג' טמפ' גבוהה מאוד.

**התוספת מיועדת לטפל במצב שבו המערכת מזהה סטייה חריגה בנתונים המגיעים מהמסד נתונים. במצב כזה, המערכת מייצרת הודעת שגיאה שמוצגת מיידית למנהל המערכת. ההודעה מדגישה את הצורך לפעול במהירות ולתפעל את התקלה כדי למנוע נזקים או המשך אי-תקינות.**

**היכן התוספת ממומשת בקוד:**

* **פונקציית alert** - יוצרת את הודעת השגיאה, מקבלת את פרטי הבעיה ומעבירה אותם הלאה לטיפול ולהצגה.
* **פונקציית update\_manager\_alert**  – **מעדכנת את ממשק המשתמש או את הנתונים במערכת עם הודעת השגיאה החדשה, ודואגת שהמידע יגיע ל־**UI **במצב עדכני.**
* **פונקציית show\_manager\_screen - אחראית על הצגת המסך למנהל, כולל הצגת הודעת השגיאה באופן בולט וגלוי כך שהמנהל לא יוכל לפספס אותה.**

**כיצד התוספת מתבטאת בחלק המוצג למנהל:**

**בחלק המוצג למנהל, הודעת השגיאה מופיעה בחלק העליון של מסך הניהול, עם טקסט ברור שמודיע על הבעיה החריגה ועל הצורך בפעולה מיידית.**

**הסבר על רכיב הצ'אטבוט (Gemini Chatbot)**

במערכת הוספנו רכיב צ'אטבוט מתקדם המבוסס על מודל השפה Gemini של Google.

הצ'אטבוט מאפשר למשתמשים לנהל שיחה חופשית ולקבל מענה לשאלות שונות הקשורות לפעילות המערכת. הוא משולב בתוך לוח הבקרה, ונותן תמיכה והכוונה למהנדסים תוך גישה לידע כללי וטכני רלוונטי. השימוש ב־Gemini מאפשר לצ'אטבוט להבין הקשר, לענות תשובות מפורטות, ואף לספק הסברים טכניים מורכבים בצורה נגישה. החיבור מתבצע באמצעות ממשק API חיצוני, והצ'אטבוט מוטמע כחלק מהאינטרקציה השוטפת של המשתמשים עם המערכת.

* 1. **Microsevices שנעשה בהם שימוש:**
* **שירות Crawler:**  
  אחראי על הורדת דפי אינטרנט, ניתוח ה-HTML ושליפת טקסטים וקישורים להמשך סריקה.
* **שירות Indexer:**

מקבל את הטקסטים מה-Crawler, מנקה את המידע (הורדת stop words, סטמינג), מחשב את שכיחות המילים ושומר את האינדקס למסד הנתונים Firebase.

**היתרונות בשירותים אלו:**

* הפרדה ברורה בין שליפת המידע לבין עיבוד הטקסטים.
* מאפשר לפתח ולשפר כל שירות בנפרד.
* קל להרחיב בעתיד — למשל להפעיל מספר Crawlers במקביל.

העבודה עם Microservices מאפשרת לנו לפצל את המערכת בצורה גמישה, מודולרית וניתנת לתחזוקה.

* 1. **KPI מרכזיים:**

ה-KPI אשר בהם עשינו שימוש בפרויקט הם:

* **User Experience:**

**שיעור שגיאות באפליקציה** קריטי ביותר למערכת שהיא דאשבורד אינטראקטיבי. נבדוק כמה פעמים ווידג'טים לא נטענים, גרפים נכשלים, החיבור לFirebase נופל או הצ'אטבוט מחזיר שגיאה? שיעור שגיאות נמוך מבטיח שהמהנדסים יכולים לסמוך על המידע המוצג.

**שביעות רצון המשתמשים:** למרות שאין מנגנון מובנה למדידה, חשוב לאסוף משוב מהמשתמשים (מהנדסים ומנהלים) לגבי קלות השימוש, בהירות הצגת הנתונים והתועלת של הכלים השונים.

* **Performance:**

**זמן תגובה** נחשב כמדד מרכזי. כמה זמן לוקח לדאשבורד להיטען? כמה מהר מתעדכנים הגרפים והסטטיסטיקות לאחר בחירת סנסור או מאפיין אחר? מהירות התגובה של הצ'אטבוט ושל מסך החיפוש משפיעה ישירות על חווית המשתמש.

**זמן השהיה** (Latency) חשוב במיוחד עבור נתוני הסנסורים. מהו פער הזמן בין קריאת נתון על ידי החיישן הפיזי לבין עליית הנתונים למערכת? עבור "מרכז בקרה", יש לשאוף לזמן השהיה נמוך ככל האפשר כדי להבטיח שהנתונים משקפים את המצב הנוכחי במעבדה.

* **זמינות ואמינות:**

**System Uptime:** האם הדאשבורד זמין ומתפקד כשהמהנדסים זקוקים לו? זהו מדד חשוב שכן רצוי לדעת האם ניתן להשתמש במערכת באופן יום יומי או שמא היא קורסת ותחבל בעבודתם של אנשי המעבדה.Z

**זמן ממוצע בין תקלות:** באיזו תדירות המערכת נתקלת בכשל קריטי למשל, התרסקות של המחשב שמריץ את התוכנה בענן.

**עלות וניצול משאבים:**

**Cost by service** הינו מדד חיוני לניהול. מהן העלויות החודשיות של שירותי הענן שבהם אתה משתמש? זה כולל:

* שימוש ב-Firebase
* עלות זמני הריצה של המערכת על גבי הפלטפורמה של Collab.
* עלות השימוש ב-Gemini API עבור הצ'אטבוט לפי כמות התוקנים.

**שיעור ניצול משאבים:** האם המערכת יעילה? לדוגמה, האם היא מבצעת קריאות מיותרות מ-Firebase ביצוע ניטור זה יכול לעזור באופטימיזציה והורדת עלויות.

* 1. **מידע עתק:**

בפרויקט השתמשנו בנתונים שנאספו ממספר חיישנים, והחלטנו להטמיע big data במערכת על ידי ביצוע ניתוח אנליטי של מקדם המתאם בין החיישנים בחוץ ולחיישנים שבתוך המעבדה.

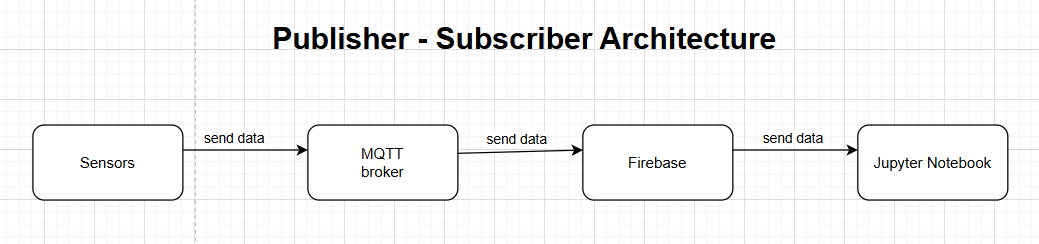
1. **Publisher-Subscriber Architecture:**

**בדפוס ה- Publisher-Subscriber, ישנו מנגנון תיווך הודעות שמנהל את רשימת המנויים.**

**כאשר מפיק (Producer) שולח אירוע, המנגנון שולח את ההודעה לכל מנוי שרלוונטי.**

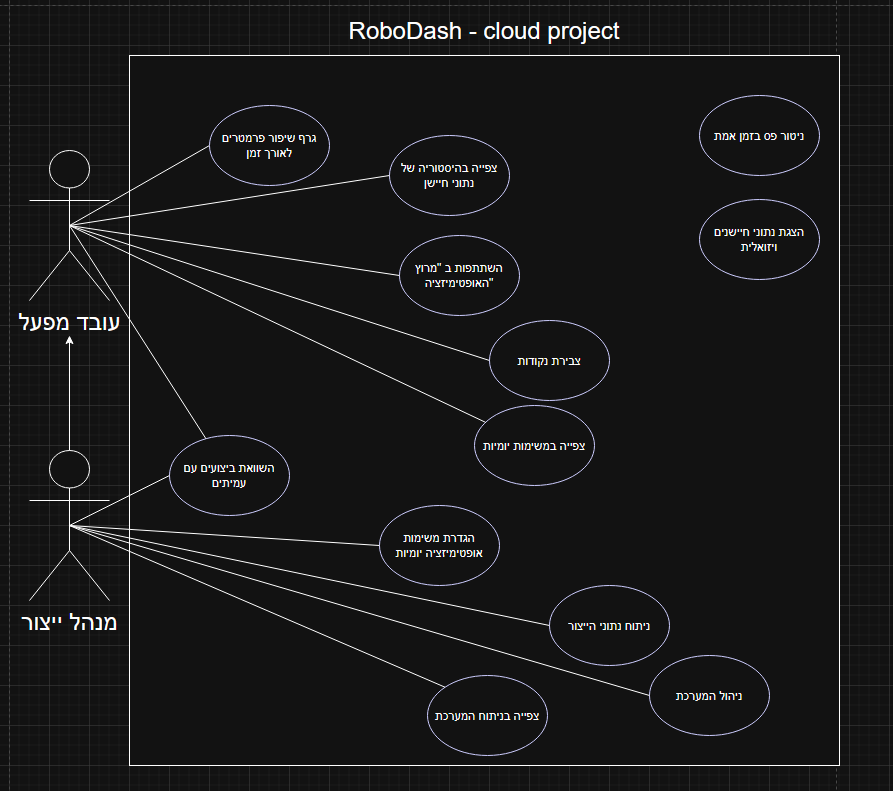
**לאחר שליחת ההודעה, היא מוסרת מהמנגנון – כך שלא ניתן לשלוח אותה שוב, ומנויים חדשים לא יראו את האירוע הקודם.**  
**באמצעות כך, מתווך ההודעות מנתק את התלות הישירה בין המפיק לצרכני ההודעות – כל צד פועל באופן עצמאי.**

**בתא העליון נמצא הקוד העוסק ב- Backend + Frontend**

**בתא התחתון ישנה התעסקות עם החיישנים + ה-MYQTT**

2.1 תרשים ומאפיינים מרכזיים

2.2 Use Case:



2.3 **דרישות פונקציונליות**

כל מה שהמערכת עושה בפועל:

1. **איסוף נתוני חיישנים**
   * שליפת נתונים מחיישנים פנימיים וחיצוניים מה־Firebase.
   * עיבוד והצגה של ערכי חיישנים בזמן אמת ובגרפים.
2. **ניהול משימות יומיות**
   * הצגת רשימת משימות יומית.
   * עדכון סטטוס משימות (השלמה, הקצאה למהנדס, תאריך השלמה).
   * חישוב ניקוד לכל משימה.
3. **מעקב אחר ביצועי מהנדסים**
   * הצגת טבלת ניקוד אישי לכל מהנדס.
   * חישוב סכום נקודות ומשימות שהושלמו.
4. **צ’אטבוט מבוסס Gemini AI**
   * מאפשר למשתמשים לקבל מענה לשאלות טכניות ולבצע אינטראקציה עם המערכת דרך שיחה חופשית.
   * עושה שימוש במודל Gemini לשם הפקת תשובות חכמות ומותאמות להקשר התעשייתי וההנדסי.
   * מבצע חיבור בין נתונים מתוך המערכת (כגון חיישנים, משימות וכו') לבין יכולות ניתוח של מודל השפה.
   * נבנה רכיב ממשק משתמש אינטראקטיבי עבורו, כולל קליטת שאילתות טקסט והצגת תשובות.
5. **ניהול ועיבוד נתונים מ־mqtt.org**
   * עיבוד מחרוזות טקסט, כולל ניקוי טקסט. Stemming (Porter Stemmer).
   * בניית אינדקס מילים עם DocIDs (עבור שאילתות חיפוש).
6. **הצגת מידע גרפי**
   * שימוש ב־ Matplotlib ליצירת גרפים של נתוני חיישנים היסטוריים.
   * הצגת גרפים בזמן אמת.
7. **שמירה על תיעוד זמנים**
   * שימוש ב־ Timezone מדויק (Asia/Jerusalem).
   * המרת תאריכים לתצוגה מקומית נכונה.
8. **ממשק משתמש אינטראקטיבי**
   * שימוש ב־ ipywidgets להצגת כפתורים, טבלאות, גרפים, טפסים אינטראקטיביים.
   * תמיכה בפעולות קריאה, עדכון ומחיקה על נתונים.
9. **דרישות לא פונקציונליות - אתגרים איתם הפרויקט מתמודד:**

**הסביבה הפיזית של המערכת**

המערכת רצה בסביבת Google Colab בענן, בתצורת notebook. היא מבוססת על חיבור לשירות Firebase Database של Google Cloud. אין דרישות חומרה מיוחדות מלבד מחשב עם דפדפן אינטרנט פעיל וחיבור לרשת.

**אבטחה (מערכת המשתמשים)**המערכת אינה כוללת מנגנוני ניהול משתמשים או אימות זהות. הגישה ל־Firebase מתבצעת באמצעות מפתחות גישה שמוסתרים מהקוד הראשי ונשמרים בקבצים חיצוניים מאובטחים. אין שימוש או אחסון של נתונים אישיים של משתמשים במערכת.

**ביצועים**

המערכת קוראת נתונים מתוך Firebase בתדירות גבוהה, ולעיתים מבצעת שליפות חוזרות על כלל הנתונים. עבור נפח נתונים קטן ובסביבת עבודה מצומצמת, הביצועים טובים ועונים על דרישות העבודה. אין מנגנוני cache פנימיים.

**עלויות**

המערכת מבוססת על שימוש בגרסאות חינמיות של Google Colab ושל Firebase Realtime Database. כל עוד היקף הנתונים והפעילות נשארים מצומצמים, אין עלויות משמעותיות להפעלתה.

**ניידות המערכת**

ניתן להריץ את המערכת מכל מחשב עם גישה לאינטרנט ול־Google Colab. הקוד ניתן להעברה בקלות כקובץ Notebook וניתן להפעלה בסביבות שונות ללא צורך בתשתיות מיוחדות.

**אמינות**

המערכת יציבה בתפעול השוטף. בבדיקות שבוצעו לא זוהו באגים מהותיים או תקלות. קיימים מנגנוני טיפול בשגיאות בסיסיים בממשק המשתמש המונעים פעולות לא חוקיות.

**בטיחות**

המערכת אינה מתממשקת עם רכיבים פיזיים במעבדה ואינה שולטת ברובוטים ישירות. היא עובדת מול נתונים בלבד ולכן אין סיכונים בטיחותיים ישירים בשימוש בה.

* 1. **ציון SUS שהתקבל:** 80

ציון של 80 במבחן SUSמעיד על כך שהמערכת שיצרנו הינה שמישה מאוד. זה אומר שהמשתמשים שלנו חוו את המערכת כידידותית, נוחה וקלה לשימוש. מבחינת התמודדות למול הביקורת עמיתים ביצענו שיפורים במערכת ובעיצוב בעקבות הביקורת שניתנה לנו.

5. **שקיפות אלגוריתמית – הושמה בקוד שלנו בדרכים הבאות**:

* **תיעוד פנימי מפורט**

הקוד עושה שימוש נרחב בתאי Markdown עבור כל חלק עיקרי בקוד.

התאים מכילים הסבר על מטרת כל סעיף ומה הוא מכיל.

כיצד זה תורם לשקיפות:

תיעוד פנימי מסוג זה מאפשר למפתחים אחרים, או למשתמשים עם רקע טכני, להבין את הלוגיקה שמאחורי הקוד. הוא אינו מיועד למשתמש הקצה הלא-טכני, אך הוא מהווה בסיס חשוב לשקיפות פנימית ופיתוח.

matplotlib, pandas, numpy, firebase\_admin ועוד.

* **שימוש בספריות מוכרות**

הפרויקט עושה שימוש בספריות פיתון נפוצות ובדוקות כמו matplotlib, numpy, firebase & pandas ספריות אלה ידועות, מתועדות היטב, וברובן הן בקוד פתוח.

כיצד זה תורם לשקיפות:

נגישות לבדיקה: העובדה שהספריות הן בקוד פתוח מאפשרת לכל מי שמעוניין לבדוק את אופן פעולתן הפנימית, כולל האלגוריתמים בהם הן משתמשות.

היכרות ותקן: משתמשים בעלי ידע בתכנות יכולים להניח בביטחון יחסי כי הספריות הללו פועלות באופן צפוי ומתועד.

קהילת מפתחים גדולות: לרוב לספריות הללו יש קהילות מפתחים גדולות, מה שאומר שניתן למצוא מידע רב על אופן פעולתן ועל שימוש בנתונים.

* **ארכיטקטורת הקוד ובניית אחריות**

ארכיטקטורת המערכת נעשתה בצורה מודולרית כך שלכל קטע קוד פונקציונליות שונה במערכת.

כיצד זה תורם לשקיפות:

מבנה ברור: קוד מובנה ומחולק היטב מאפשר הבנה קלה יותר של הפונקציונליות של כל חלק.

הפרדת קונספטים: הפרדה בין ייבוא ספריות, איסוף נתונים, עיבוד נתונים והצגה מאפשרת לזהות באופן ברור איזה חלק בקוד אחראי על איזה תהליך אלגוריתמי או איסוף נתונים. לדוגמה, אם יש סעיף המוקדש לאיסוף נתונים, ברור שבחלק זה מתבצע איסוף הנתונים.

* **גישה לנתונים**

הקוד מייבא את הספרייה firebase\_admin ומכך ניתן להסיק כי קיימת אינטגרציה בקוד מול המסד נתונים Firebase

כיצד זה תורם לשקיפות:

הצהרה על מקור הנתונים: עצם הייבוא של ספריית Firebase מציין למשתמשים ולמפתחים שמקור הנתונים הוא Firebase.

מדיניות נתונים של צד שלישי: ניתן להפנות את המשתמש למדיניות הפרטיות של Firebase עצמה לגבי איסוף ואחסון נתונים, ובכך להעביר את נטל השקיפות לחברה מוכרת ומוסדרת.

**6.** אתגרים שעלו במהלך הפרויקט:

**התמודדות עם כמויות גדולות של נתונים וביצועים איטיים**

* **האתגר שעלה:** בשלבים מסוימים, כמות הנתונים שרצינו לאחזר ולעבד החלה להיות גדולה, והדבר האט משמעותית את זמן הריצה של המחברת. לא יכולנו להרשות לעצמנו לחכות זמן רב לכל שאילתה.
* **איך התמודדנו:** למדנו להשתמש בשאילתות חכמות יותר ב-Firebase במקום למשוך את כל הנתונים, ביקשנו רק את מה שבאמת היה נחוץ לנו באותו רגע.

**עיצוב וייצוג חזותי ברור של הנתונים**

* **האתגר שעלה:** להציג נתונים זה דבר אחד, אבל להציג אותם בצורה שתהיה ברורה, אסתטית ומעבירה את המסר בצורה אפקטיבית זה אתגר בפני עצמו. לא תמיד היה קל לבחור את סוג הגרף המתאים ביותר, או לעצב אותו כך שיהיה קריא.
* **איך התמודדנו:** כדי לשמור על אחידות וקלות עבודה, הגדרנו מראש קבועים לעיצוב - צבעים, גופנים, גדלים - והשתמשנו בהם באופן עקבי בכל הגרפים. יצרנו גם פונקציות קטנות שעוזרות לנו ליצור גרפים נפוצים במהירות, וכך יכולנו להתנסות ולשפר את התצוגה עד שהיינו מרוצים.

**תיק למתכנת:**

**קבצים:**

כל הקוד נמצא בתוך מחברת Google Colab בשם:

HW3\_WOLF.ipynb

המחברת כוללת את כל הרכיבים: סריקה, יצירת אינדקס, חיפוש, עיבוד טקסט וסטטיסטיקות מילים.

**רכיבים ופונקציות עיקריות:**

**סורק אתרים ואינדיקס:**

* **:fetch\_page\_text(url)** מוריד את תוכן ה-HTML של כתובת האתר שניתנה.
* **:extract\_links(self, base\_url, soup)** הפונקציה מחלצת קישורים פנימיים מעמוד אינטרנט, מוודאת שהם שייכים לאותו דומיין ומסירה מזהים פנימיים כמו #. היא ממירה קישורים יחסיים לאבסולוטיים, מסננת כתובות שאינן תקינות, ושומרת קישורים ייחודיים בקבוצה. לבסוף, היא מחזירה את רשימת הקישורים התקפים לשימוש בסריקה נוספת.

**search - הפעלת מנוע החיפוש :**

* **perform\_search:** הפונקציה מחפשת מונח שהוזן על ידי המשתמש במסד נתונים, מזהה את הגרסה המותאמת שלו, ובודקת האם הוא מופיע באינדקס, תוך הצגת מידע על מספר הפעמים שנמצא וקישורים רלוונטיים.
* **: handle\_search** הפונקציה מטפלת בלחיצה על כפתור חיפוש, שולפת את הערך שהוזן על ידי המשתמש, מחפשת תוצאות באמצעות perform\_search, מנקה את התצוגה הקודמת ומציגה את התוצאות החדשות בפורמט HTML.

**:Sensor Data**

* **: generate\_sensor\_statistics**הפונקציה יוצרת תצוגה HTML המציגה את הנתונים האחרונים של חיישן מסוים יחד עם חישובים סטטיסטיים של

נתונים היסטוריים מאותו חיישן ב-5 הדקות האחרונות.

* **: update\_sensor\_data\_and\_plot**הפונקציה מעדכנת את תצוגת הסטטיסטיקות והגרפים בהתאם לחיישן ולמאפיין שנבחרו, טוענת נתונים חדשים, מציגה סטטיסטיקות מעודכנות, ומייצרת גרף היסטורי חדש.
* **: update\_sensor\_status**הפונקציה מעדכנת את תצוגת מצב החיישנים במסך הניהול, בודקת זמינות נתונים מהחיישנים, ומציגה את הסטטוס שלהם בהתאם.
* **: on\_sensor\_selector\_change**הפונקציה מעדכנת את אפשרויות הגרף בהתאם לחיישן שנבחר, מוודאת שהמאפיין שנבחר תקף, ואז מעדכנת את הסטטיסטיקות והגרף בהתאם.

**ממשק משתמש – :UI**

* **: create\_html\_widget** יוצרת רכיב HTML מותאם אישית להצגת תוכן בפורמט דינמי עם אפשרויות עיצוב.
* **: create\_button**יוצרת כפתור אינטראקטיבי עם תיאור, סגנון ועיצוב מותאם אישית לממשק המשתמש.
* **: create\_text\_input**יוצרת שדה קלט טקסט אינטראקטיבי עם תיאור והגדרות עיצוב מותאמות אישית.
* **: create\_output**יוצרת רכיב פלט דינמי לתצוגת מידע מתעדכן בממשק.
* **: create\_dropdown**יוצרת רשימה נפתחת עם אפשרויות בחירה ותיאור, תוך שימוש בהגדרות עיצוב מותאמות.
* **: create\_vbox**מארגנת ומציגה רכיבים בתיבה אנכית, תוך התאמה אישית של הסגנון והעיצוב.
* **: create\_hbox**מארגנת ומציגה רכיבים בתיבה אופקית, עם אפשרויות עיצוב מותאמות.

**תבניות עיצוב:**

השתמשנו בתבניות עיצוב של wrapper, singleton ו- factory

**קטעי קוד מעניינים:**

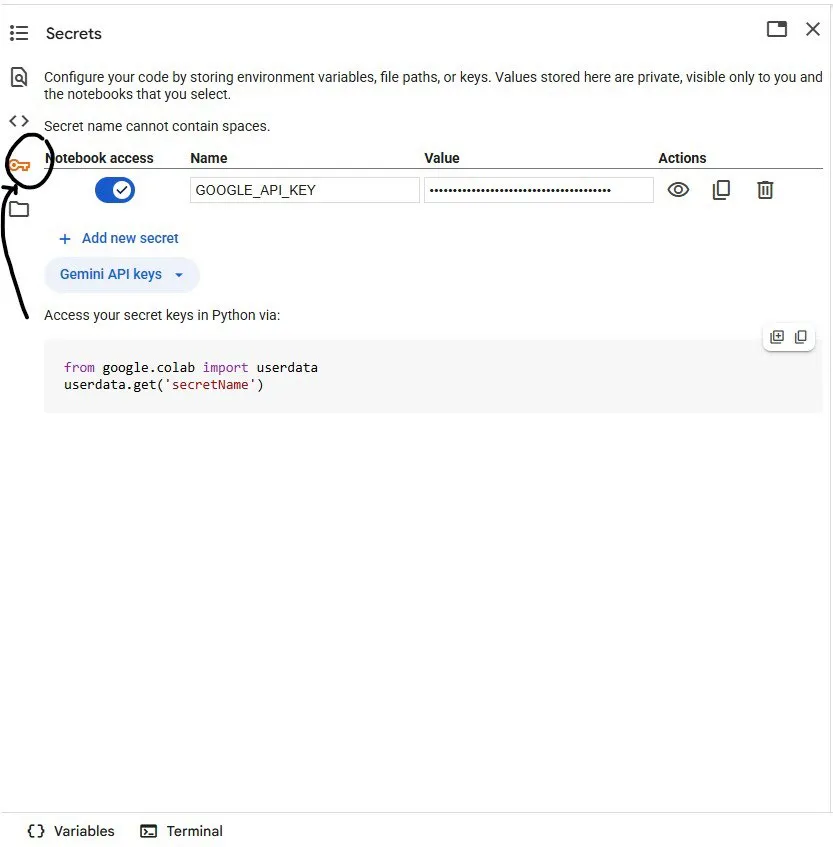
1.  - Anomaly detectorהפונקציה בודקת ערכי חיישנים ומזהה חריגות על ידי השוואתם לטווחים נורמליים מוגדרים מראש, אם הערך מחוץ לטווח היא מוסיפה הודעה לרשימת החריגות.
2. Chatbot

בתוכנה יש להטמיע מפתח API עבור 'gemini-1.5-flash'

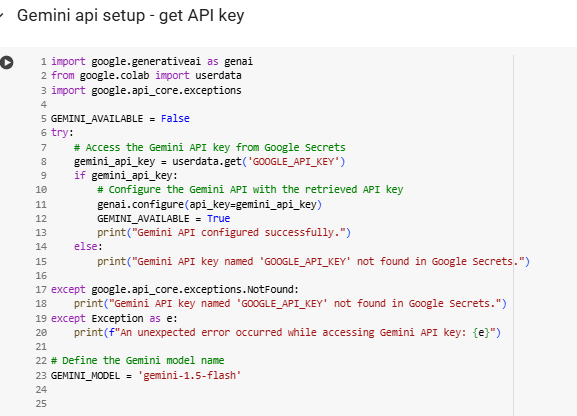
את המפתח יש להפיק מהקישור הבא <https://aistudio.google.com/prompts/new_chat>

ולהכניס אותו תחת google secrets בשם google\_API\_key

או לחלופין להשתמש במפתח הדיפולטיבי שמוטמע במערכת.



A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.A white screen with blue and purple lines

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**תיק למשתמש:**

הסבר כללי על המערכת:

המערכת היא אפליקציית דשבורד אשר מיועדת למהנדסים העובדים עם פס הייצור האוטונומי במעבדת הרובוטיקה.

האפליקציה מספקת ממשק מקצועי לניטור, ניתוח ושליטה בזמן אמת בתהליכי הייצור, תוך הצגת נתונים מחיישנים שונים (טמפרטורה, מהירות, דיוק, צריכת אנרגיה) בצורה ויזואלית.

בנוסף באפליקציה ישנה תחרות של נקודות בין המהנדסים על מנת להגדיל את המוטיבציה לביצוע המשימות.

במערכת קיימים המסכים הבאים:

**מסך ראשי:**

מציג הודעת welcome עם התאריך והשעה הנוכחית.

מציג חלון שמציג את מצב החיישנים בזמן אמת האם פעיל/לא פעיל בהתבסס על האם קיים מידע מהחיישן במסד.

מציג את מספר המשימות היומיות שנותרו לביצוע.

מציג את מי שמוביל כרגע את מירוץ המהנדסים.

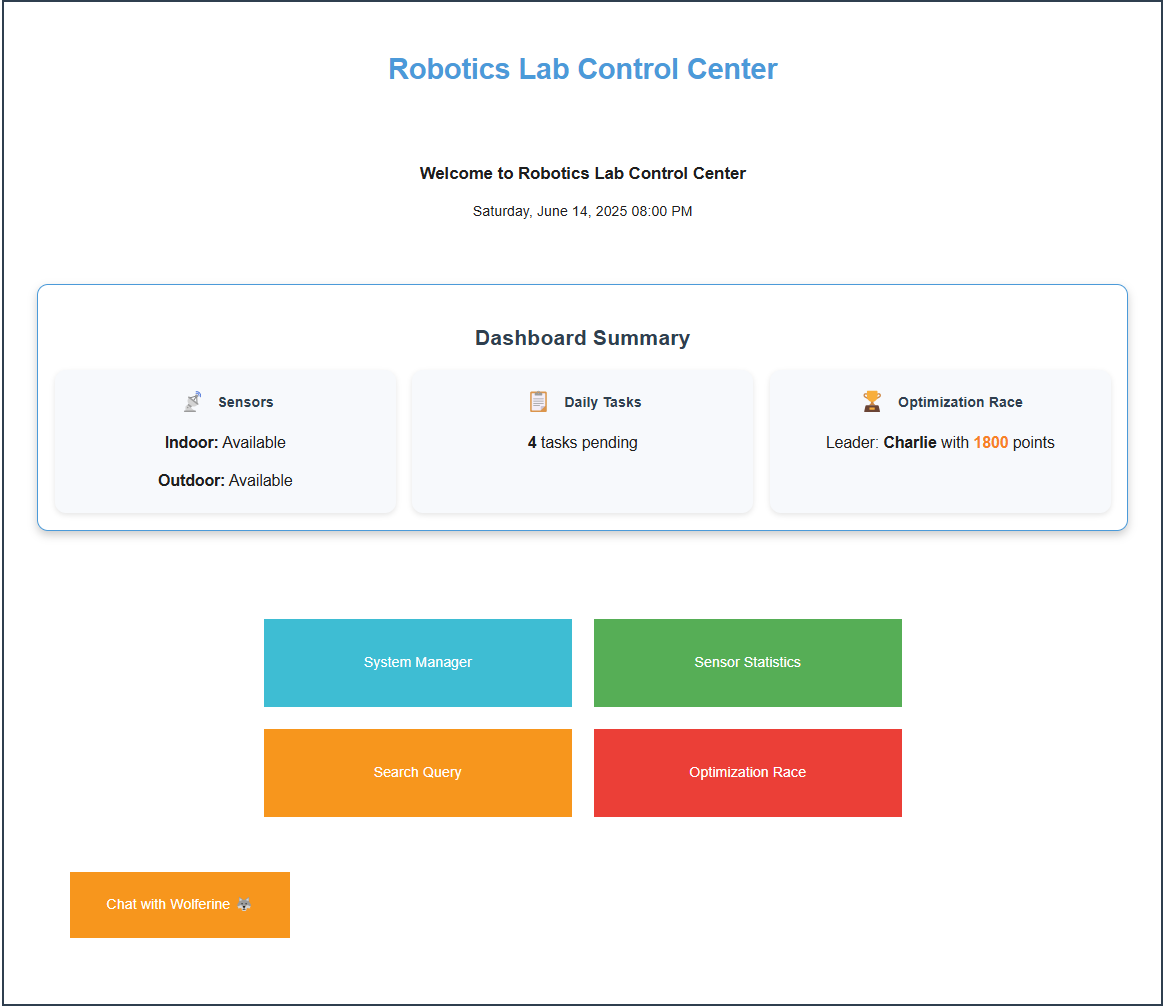
בנוסף החלון הראשי מציג כפתורים שמהם ניתן לעבור לכל אחד מהמסכים האחרים במערכת באמצעות לחיצה על הכפתור עם שם המסך הבא.

לחיצה על **system manager** מעבירה אותנו לחלון **system manager.**

לחיצה על **Sensor Statistics** מעבירה אותנו לחלון **Sensor Statistics.**

לחיצה על **search query** מעבירה אותנו לחלון **search query.**

לחיצה על **optimization race** מעבירה אותנו לחלון **optimization race.**

לחיצה על **chat with Wolferine** מוסיפה לנו חלון אשר בו **ניתן לנהל שיחה עם GEMINI.**

**חלון system manager:**

בראש המסך של חלון המנהל מופיעות חריגות של החיישנים מן הערכים התקינים שנקבעו במערכת.

A white background with red text

AI-generated content may be incorrect.

לאחר מכן ישנו חלון אשר מופיע בו סטטוס החיישנים.

A screenshot of a computer

AI-generated content may be incorrect.

בנוסף ישנה רשימת המשימות היומית ניתן לראות לכל משימה את הסטטוס ואת כמות הנקודות שהיא שווה. אם משימה מוגדרת כהושלמה ניתן לראות גם את פרטי מי שהשלים ואת זמן הסיום המדויק.

ולבסוף ניתן לראות את סטטוס מירוץ המהנדסים.

A screenshot of a computer

AI-generated content may be incorrect.

בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

וכפתור chat with Wolferine אשר פותח לנו חלון שבו נוכל לתקשר עם Gemini.

**חלון :Sensor Statistics**

בראש המסך של החלון ישנם 2 Drop box הראשונה משמשת לבחירת הסנסור הרצוי (פנימי/חיצוני)

השנייה משמשת לבחירת attribute ספציפי בתלות בסנסור שנבחר בתיבה הראשונה.

לסנסור הפנימי ישנם התכונות הבאות: לחות, טמפרטורה ,לחץ ,מרחק

לסנסור החיצוני ישנם התכונות הבאות: לחות, טמפרטורה, עוצמת אור.

בנוסף לשני הסנסורים ישנה האפשרות לבחור בקורולציה בין הפנימי לחיצוני על מנת לראות את יחס הטמפרטורות בין הסנסור הפנימי לחיצוני.

לאחר בחירת הסנסור והתכונה הרצויה מתקבל גרף המציג את ערכי הסנסור הנבחר ביחס לתכונה הנבחרת ב-5 הדק האחרונות.

בנוסף מוצג הערך האחרון שנמדד והשעה שבה הוא נמדד.

לאחר מכן מוצגים נתונים סטטיסטים ביחס למידע של ה5 דק האחרונות.

יוצגו הנתונים הבאים : ממוצע, חציון ,ערך מינימלי, ערך מקסימלי וסטיית התקן.

אם התכונה הנבחרת היא קורולציה בין הפנימי לחיצוני המידע שיוצג יהיה הקורולציה בין סנסורי הלחות הפנימי והחיצוני וסנסורי הטמפרטורה הפנימי והחיצוני.

לאחר מכן יוצג הגרף שהופק החלון יראה באופן הבא:

A white background with black and white clouds

AI-generated content may be incorrect.

A graph with blue lines

AI-generated content may be incorrect.

בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

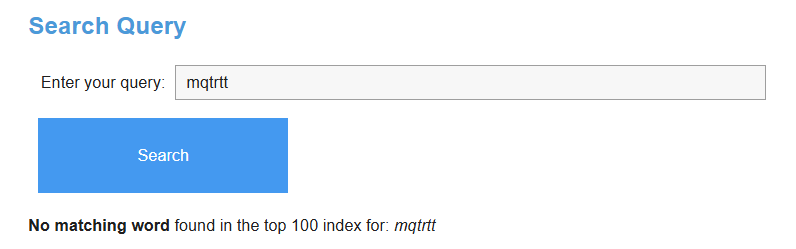
וכפתור chat with Wolferine אשר פותח לנו חלון שבו נוכל לתקשר עם Gemini.

**חלון search query:**

בעת פתיחת החלון מופיע תיבת טקסט שבה ניתן להכניס כל מילה רצויה.

באם המילה מופיעה באינדקס,כאשר נלחץ על כפתור הsearch נקבל רשימה של קישורים שהמילה מופיעה בה מהאתר של **MQTT: The Standard for IoT Messaging.** ליד כל קישור מופיע כמות הפעמים שהמילה מופיעה בעמוד הרלוונטי.

אם המילה לא מופיעה באינדקס נקבל הודעת שגיאה לדוגמא אם אני מזין mqtrtt מתקבלת ההודעה הבאה:



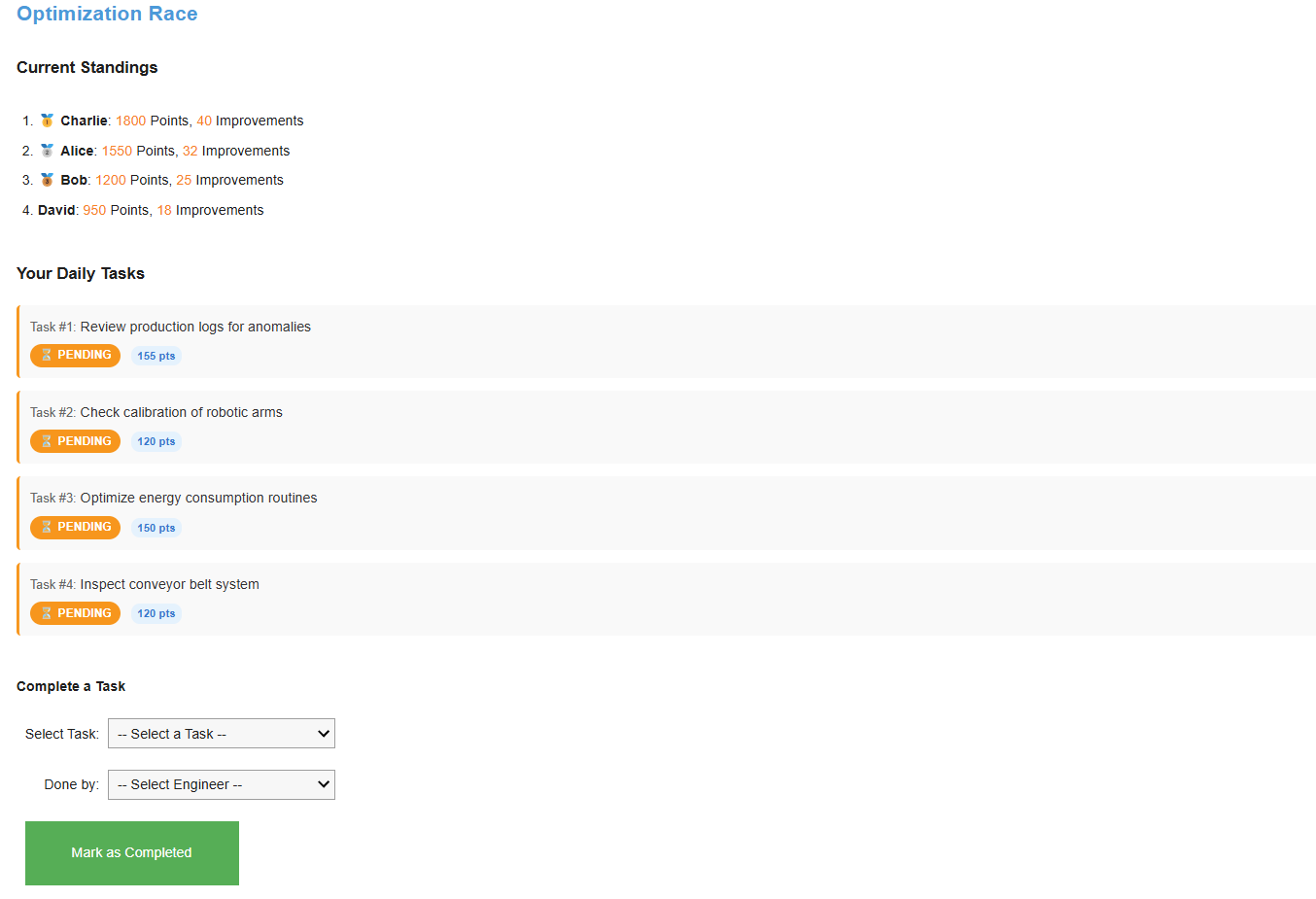
בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

וכפתור chat with Wolferine אשר פותח לנו חלון שבו נוכל לתקשר עם Gemini.

**חלון optimization race:**

בעת פתיחת החלון מופיע בחלק העליון המצב העדכני של מירוץ המהנדסים.

לאחר מכן מופיע רשימת המשימות היומיות עם הסטטוס של כל משימה והניקוד עבורה.



לאחר מכן ישנם 2 drop box הראשון עבור בחירת משימה שאותה אנו רוצים לעדכן שסיימנו והשני עבור שם המהנדס שסיים אותה.

בתיבה הראשונה ניתן לראות את כל המשימות שהסטטוס שלהן ממתין לביצוע על מנת לסיים משימה נדרש לבחור אחת מהמשימות מהרשימה.

בתיבה השנייה ניתן לראות רשימה של כל ההמהנדסים שעובדים במעבדה. על מנת לסיים משימה ולקבל את הנקודות עבורה נדרש לבחור באחד המהנדסים.

לאחר מכן יש לנו את הכפתור mark as completed באם כל הפרטים הוזנו כמו שצריך בעת הלחיצה על הכפתור מתקבלת הודעה הבאה עם המהנדס שסיים אותה וקיבל נקודות.



כמו כן טבלת מירוץ המהנדסים וטבלת המשימות היומיות מתעדכנת באותו הרגע.

המשימה שהושלמה הסטטוס שלה הופך ל completed עם שם המהנדס שביצע אותה והשעה המדויקת שבה הסתיימה.

בנוסף אם חל שינוי ומישהו חדש עלה למקום הראשון שמו והנקודות שלו יוצגו בחלון הראשי של המערכת כמו גם כמות המשימות החדשה שתרד ב1.

באם משתמש מנסה לללחוץ על mark as completed מבלי שנבחרה משימה בתיבה הראשונה מתקבלת ההודעה הבאה:



באם לא נבחר מהנדס בתיבה השנייה מתקבלת ההודעה הבאה:



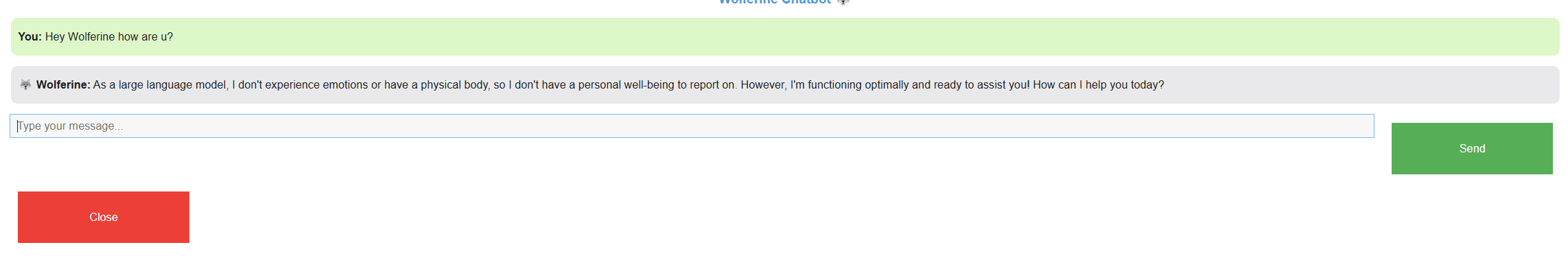
בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

וכפתור chat with Wolferine אשר פותח לנו חלון שבו נוכל לתקשר עם Gemini.

**חלון chat with wolferine:**

בעת פתיחת החלון מופיע לנו box שבו ניתן להכניס טקסט על מנת לנהל שיחה עם Gemini.

אחרי שמקלידים משפט ניתן ללחוץ על כפתור send על מנת לשלוח את ההודעה לGemini.



בתחתית החלון בצד שמאל יש כפתור close אשר סוגר את החלון הנוסף שנפתח עבור השיחה עם Gemini.

**שקיפות אלגוריתמית:**

* **תיעוד פנימי מפורט**

הקוד עושה שימוש נרחב בתאי Markdown עבור כל חלק עיקרי בקוד.

התאים מכילים הסבר על מטרת כל סעיף ומה הוא מכיל.

כיצד זה תורם לשקיפות:

תיעוד פנימי מסוג זה מאפשר למפתחים אחרים, או למשתמשים עם רקע טכני, להבין את הלוגיקה שמאחורי הקוד. הוא אינו מיועד למשתמש הקצה הלא-טכני, אך הוא מהווה בסיס חשוב לשקיפות פנימית ופיתוח.

matplotlib, pandas, numpy, firebase\_admin ועוד.

* **שימוש בספריות מוכרות**

הפרויקט עושה שימוש בספריות פיתון נפוצות ובדוקות כמו matplotlib, numpy, firebase & pandas ספריות אלה ידועות, מתועדות היטב, וברובן הן בקוד פתוח.

כיצד זה תורם לשקיפות:

נגישות לבדיקה: העובדה שהספריות הן בקוד פתוח מאפשרת לכל מי שמעוניין לבדוק את אופן פעולתן הפנימית, כולל האלגוריתמים בהם הן משתמשות.

היכרות ותקן: משתמשים בעלי ידע בתכנות יכולים להניח בביטחון יחסי כי הספריות הללו פועלות באופן צפוי ומתועד.

קהילת מפתחים גדולות: לרוב לספריות הללו יש קהילות מפתחים גדולות, מה שאומר שניתן למצוא מידע רב על אופן פעולתן ועל שימוש בנתונים.

**הצגה:**

|  |  |  |  |
| --- | --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק | השינוי שבוצע בפועל |
| פשטות - לעיתים הקוד לעיתים מרוכז מידי בקובץ יחיד, מה שמקשה על ההתמצאות. כדאי לפצל אותו לקבצים לפי תפקידים: למשל קובץ אחד לממשק המשתמש (UI), אחד ללוגיקה של המשימות, אחד למודול הצ'אט ועוד. כך כל חלק יהיה פשוט יותר לתחזוקה והרחבה. | כן | כן. בכוונתנו לבצע פיצול לחלק מהפונקציות הגדולות ולאחד קוד שחוזר על עצמו, אבל תוך שמירה על הבהירות והזרימה של הקוד. לא נרצה לפשט יותר מדי כדי לא לפגוע בקריאות. | הקוד חולק למקטעים קטנים יותר ול-Cells ספציפיים וייעודיים עבור כל קטע. |
| מימוש - הקוד כולל לוח בקרה שלם למעבדה רובוטית: סטטיסטיקות חיישנים, זיהוי חריגות, ניהול משימות, תחרות מהנדסים, צ'אטבוט מבוסס Gemini, ויכולת חיפוש במידע מהאתר mqtt.org | לא | כרגע אנחנו מרוצים מהמימוש ואיך שהפיצ'רים עובדים בפועל | - |
| מודלריות - ברובו כן. יש חלוקה טובה למחלקות כמו ChatbotComponent ו־TaskManager, וגם לפונקציות שמייצרות ווידג'טים. | לא | כרגע, החלוקה מספקת ונוחה לתחזוקה | - |
| יעילות - בחלק מהפונקציות מתבצעת שליפה חוזרת של כלל הנתונים מה־Firebase בכל קריאה, דבר שעלול לפגוע בביצועים כאשר כמות הנתונים גדלה. כדי לשפר זאת, אפשר ליישם מנגנון cache מקומי או לעבוד עם timestamp ולשלוף רק נתונים חדשים. זה יפחית משמעותית את זמן הטעינה ויעמיס פחות על השרת. | כן | ננסה לצמצם את הקריאות ולעבוד רק עם שינויים. | המעטנו את שליפות הנתונים מה-DB במקטעים שאמרו לנו ששלפנו יותר מדיי. |
| באגים - בבדיקה שלנו לא נתקלנו בבאגים משמעותיים או בהתנהגות חריגה של המערכת. כל הכפתורים עבדו כמצופה והתגובות לפעולות היו תקינות | לא | לא זיהינו כרגע תקלות או בעיות תפקוד, ולכן אין צורך בשינויים מיידיים. | - |
| טיפול בשגיאות - כן. לדוגמה, כאשר לא נבחר מהנדס או משימה, מתקבלת הודעת שגיאה ברורה למשתמש. כמו כן, קיימים תנאים שמונעים לחיצה כפולה או שליחה ריקה. | לא | כרגע כל תרחישי הקצה החשובים מכוסים, אך פתוחים להוסיף שיפור קטן בעתיד | - |
| בדיקות - מה קורה אם יש 500 משימות? האם הדף קורס? שהמשתמש מקבל הודעה במקרה של שגיאה (למשל, כשאין נתונים). | כן | נבצע בדיקות עומס נוספות ונוסיף טיפול ייעודי במקרים של חוסר נתונים או עומסים חריגים, כדי לוודא שהמערכת יציבה גם בתרחישים קיצוניים. | בוצעו בדיקות עומס נוספות על המערכת, ומצב המערכת נמצא תקין. |
| שימושיות - כן. העיצוב מושך ונעים, הכפתורים ברורים, הסגנון מודרני ויש תחושה שהממשק מזמין ומעודד שימוש חוזר. שילוב הצ’אטבוט מוסיף נוחות וחוויית משתמש מתקדמת. | לא | בשלב זה אין צורך בשינויים מהותיים, אך נשמח לשפר בהמשך על סמך פידבק נוסף מהמשתמשים. | - |
| תיעוד - כן. רוב הפונקציות כוללות docstrings ברורים שמסבירים מה הן עושות, כולל הסברים על פרמטרים והחזרות. | לא | התיעוד הנוכחי מספק להבנת הקוד ולתחזוקה. | - |
| אתיקה ושקיפות - אין שימוש בנתונים אישיים, ולכן פרטיות המשתמש לא נפגעת. מצד שני, לא קיים הסבר ברור למשתמש כיצד הצ’אט או האלגוריתם של תחרות המהנדסים פועל. | כן | נוסיף הסבר מפורט יותר בעמוד העזרה או בתיעוד המשתמש לגבי אופן פעולת הצ’אט והאלגוריתמים, לשם שקיפות מלאה. | הוספנו הסבר מפורט בכל הקוד המסביר מה קורה עם נתוני המשתמש בכל מקטע שעוסק בו. |
| אבטחה - לא מצאנו מפתחות API גלויים או מידע אישי שמופיע בקוד. המפתחות מוסתרים בקבצים חיצוניים בצורה נכונה. | לא | האבטחה הנוכחית עומדת בסטנדרטים הנדרשים. נבצע בדיקות תקופתיות לוודא שאין פרצות חדשות. | - |
| ביצועים - ככל שהמערכת תגדל, צפוי עומס על קריאות נתונים, במיוחד אם יהיו הרבה משתמשים בו זמנית. | לא | המערכת פועלת בצורה תקינה עם נפח הנתונים והמשתמשים הנוכחי, והביצועים עומדים בדרישות שהוגדרו לפרויקט. | - |
| קריאות - – readability כן. שמות הפונקציות והמשתנים נבחרו היטב, מה שעוזר להבין את זרימת הקוד. חלק מהפונקציות מעט ארוכות וניתן לפרק אותן כדי לשפר את הקריאות, במיוחד באזורים בהם יש הרבה תנאים או לולאות. | לא | הקריאות של הקוד טובה ומאפשרת לנו לעבוד עליו בצורה שוטפת. הפיצול לפונקציות קטנות יותר יכול להיות רלוונטי בהמשך, אך בשלב זה הקוד קריא מספיק ואין בעיות שמצריכות טיפול מיידי. | - |

**13. מקורות:**

כלי AI שונים כגון: ChatGPT, Gemini, ClaudeAI ואססיסטנטים שונים לדוגמה MS Copilot ו-Gemini בקולאב.

ויקיפדיה עבור מונחים ומושגי בסיס: <https://www.wikipedia.org/>

אתר הקורס במודל עבור מושגים הגדרות וקטעי קוד שניתן לעבוד עימן בפרויקט.