תרגיל בית 3 – מחשוב ענן צוות **Wolf**

מגישים:

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

**מהנדס המערכת:** דניאל לחמקוב

**חלוקת העבודה:** התפקידים בצוות הוגדרו בהתאם להעדפות ולחוזקות האישיות של כל אחד מהחברים, תוך שאיפה לנצל בצורה מיטבית את הכישורים של כל אחד. עם זאת, הקפדנו שכל חבר צוות ייחשף גם למשימות של שאר החברים, כדי לשמור על ראייה מערכתית רחבה ולהבטיח הבנה משותפת של כלל מרכיבי הפרויקט. לאורך כל העבודה שמרנו על תקשורת רציפה, תיאום מלא ושיתוף פעולה הדוק, אשר אפשרו חפיפה ותמיכה הדדית בין חברי הצוות לאורך כל שלבי הביצוע.

**איטרציה 1**

|  |  |  |  |
| --- | --- | --- | --- |
| שם חבר הצוות | משימות שהוקצו | משימות שהושלמו | Acceptence test – בדיקות קבלה |
| יובל כוגן | בניית צ'אטבוט | בניית צ'אטבוט | אינטרקציה בסיסית – הצ'אטבוט מגיב לשאלות כלליות.  מפתח Gemini-API מוחבא.  השמת Singleton לבוט כך שהשיחה איתו מופיעה בכל החלונות וההיסטוריה של השיחה נשמרת. |
| אלמוג רז | הרצה של בדיקות קבלה + כתיבה ותכנון של מסמך וורד למטלה 3. | הרצה של בדיקות קבלה + כתיבה ותכנון של מסמך וורד למטלה 3. | בדיקה של הבדיקות קבלה על מנת לוודא תקינות ולתת אישור סופי לקראת ההגשה.  יצירת סנכרון בין תוכן הפרויקט לבין התוכן שנכתב לתוך מסמך הוורד. |
| רון סלמה | רפקטורינג על ה-codebase | רפקטורינג על ה-codebase | שיפור קריאות הקוד.  שמירה על פונקציונליות קיימת.  הפחתת כפילויות בקוד היכן שניתן. |
| יניב בודאגה | UX & UI | UX & UI | הטמעה ויישום גורף של אלמנטים עיצוביים.  בדיקות שנעשו מול משתמשים אמיתיים על מנת לחוות מהם איך המערכת הרגישה להם וכך לבצע תיקונים היכן שאפשר. |
| דניאל לחמקוב | הטמעת אלגוריתמיקה למידע עתק במערכת | הטמעת אלגוריתמיקה למידע עתק במערכת | נכונות האלגוריתם בנתונים גדולים.  ביצועי האלגוריתם תחת עומס.  סקאלביליות האלגוריתם.  יעילות ניצול המשאבים. |
| ליאור דגש | בניית מיקרוסרוויסים | בניית מיקרוסרוויסים | פונקציות ליבה עבור כל מיקרוסרוויס.  עמידות וביצועים. |

1. **שימוש במיקרוסרוויסים:**

* **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 ELEMENTS:**

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.

**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.

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. 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.

***חלק שלישי : סגירת הפרויקט \_( 20 נקודות)***

*תיק למתכנת:* ***(ליאור)***

תיעוד פרויקט - המערכת שלנו

**קבצים:**

כל הקוד נמצא בתוך מחברת Google Colab בשם:

HW3\_WOLF.ipynb

המחברת כוללת את כל הרכיבים: סריקה, אינדוקס, 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.
* **: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.

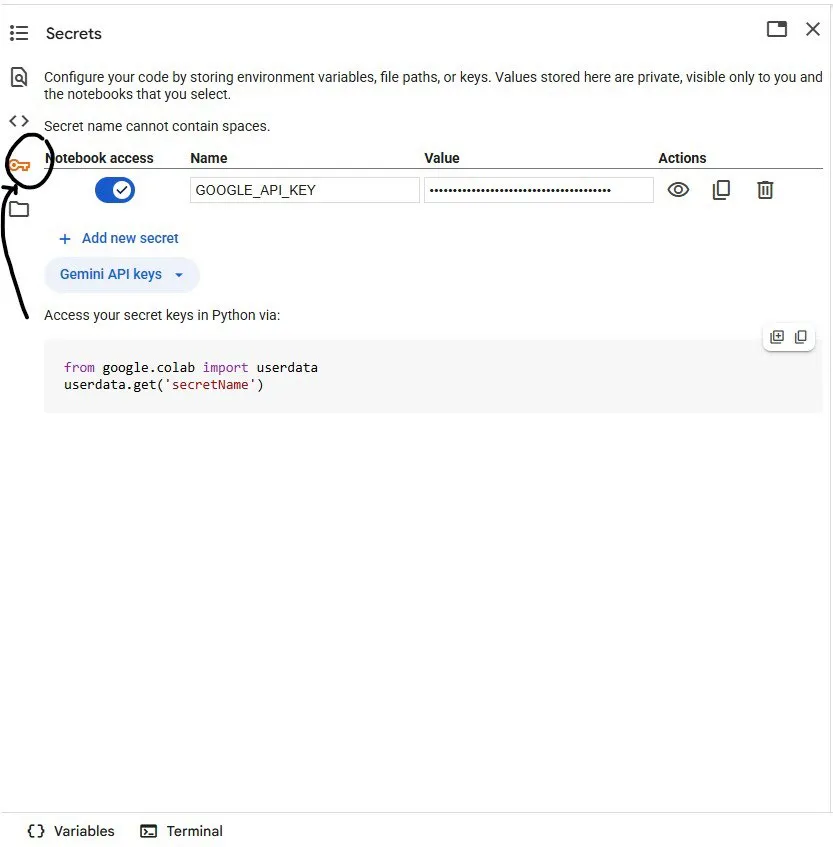


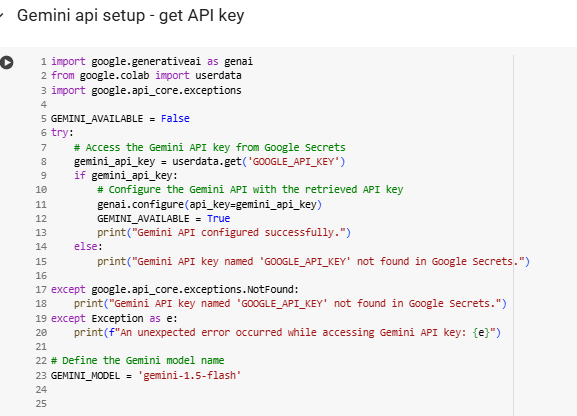
1. Chatbot

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





A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

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 דבשורד 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 ישנם 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.

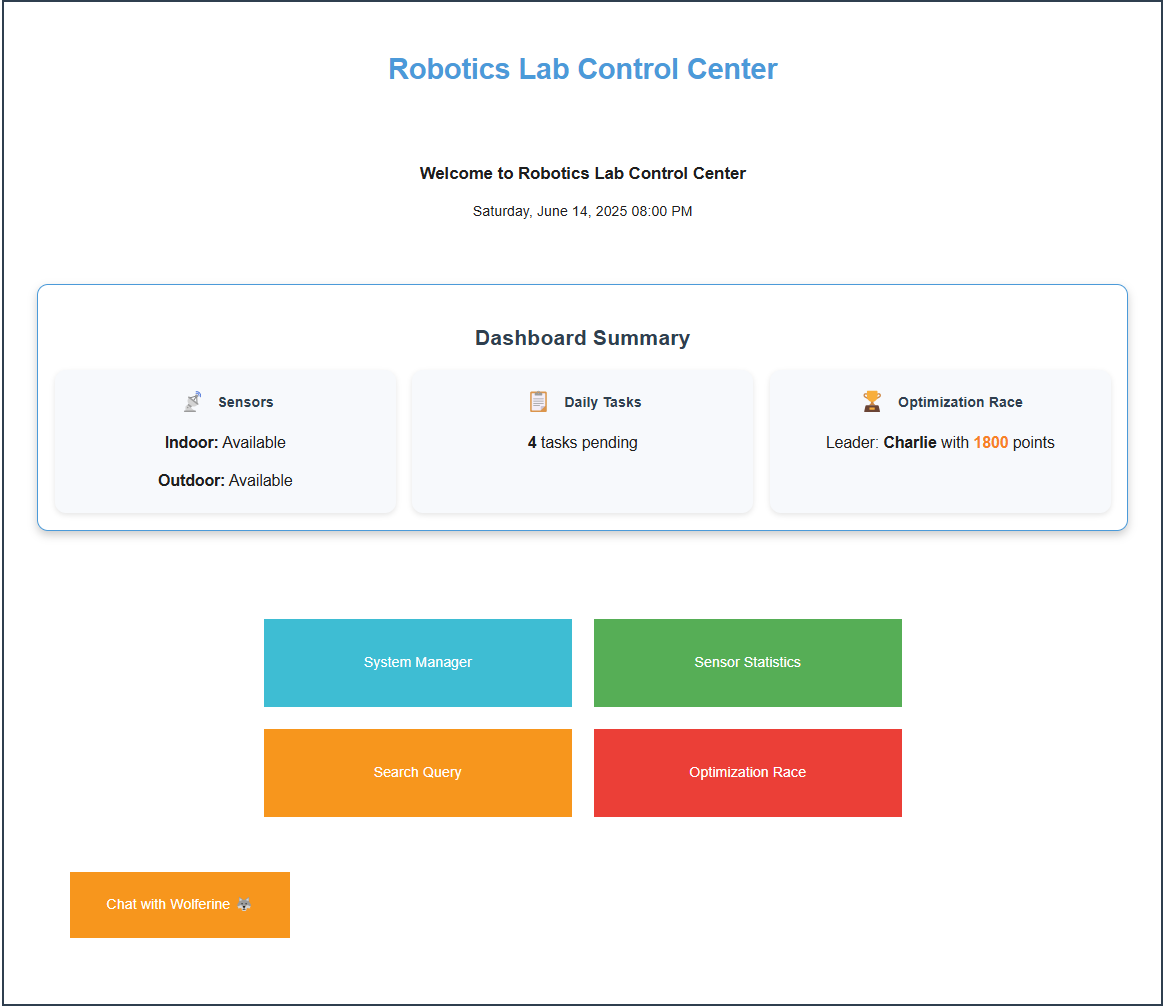
לחיצה על **system manager** מעבירה אותנו לחלון **system manager.**

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 **ניתן לנהל בו 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 ספיציפי 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 ב5 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.

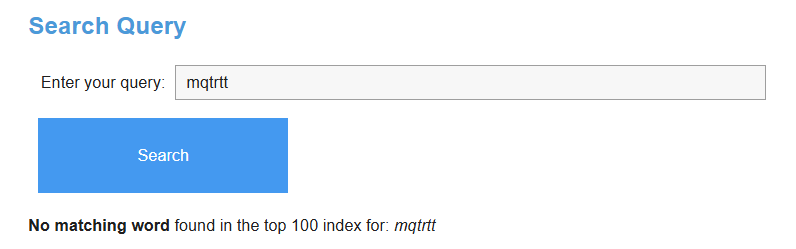
וכפתור chat with Wolferine אשר פותח לנו חלון שבו נוכל לתקשר עם Gemini.

**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:



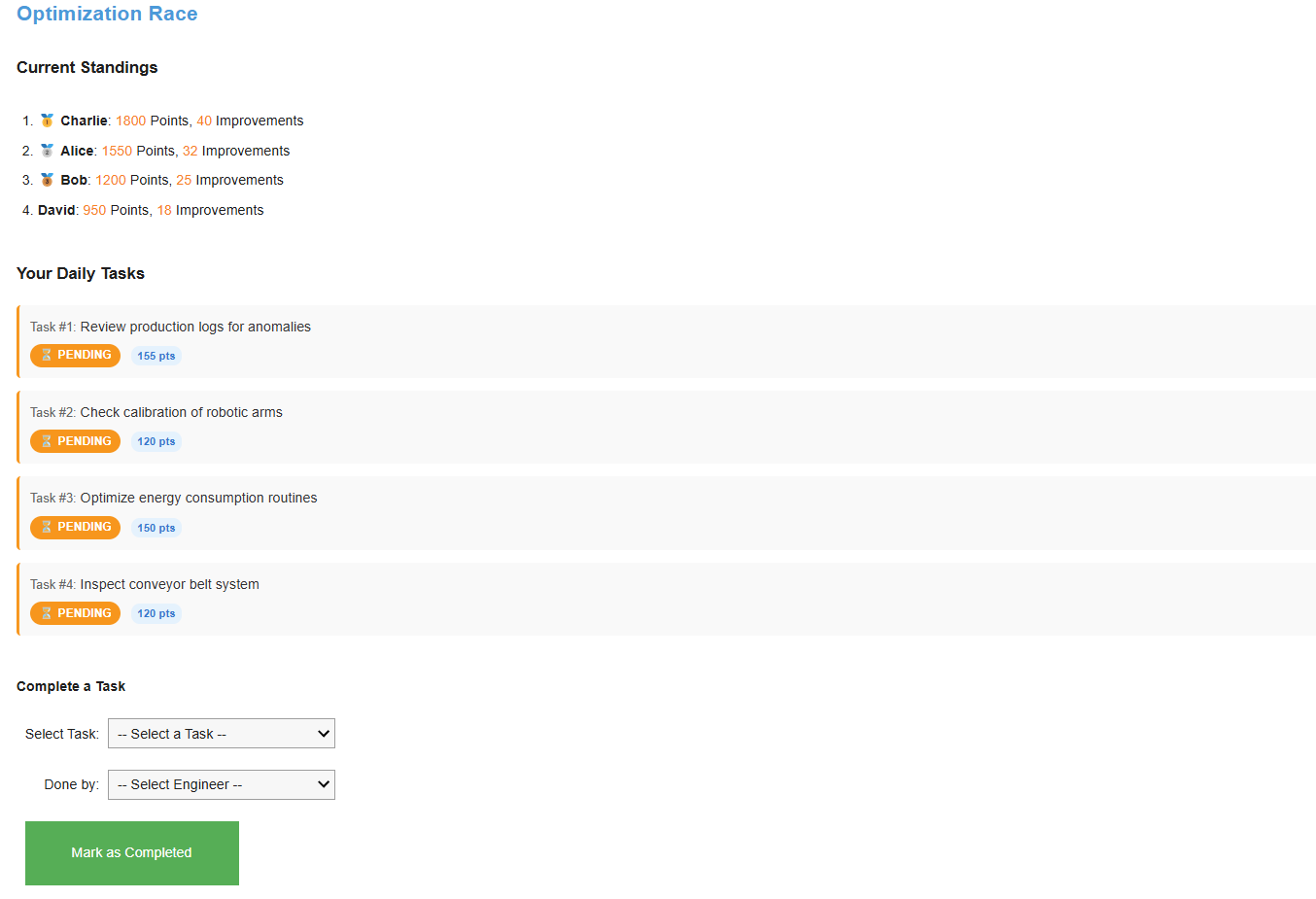
בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

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:



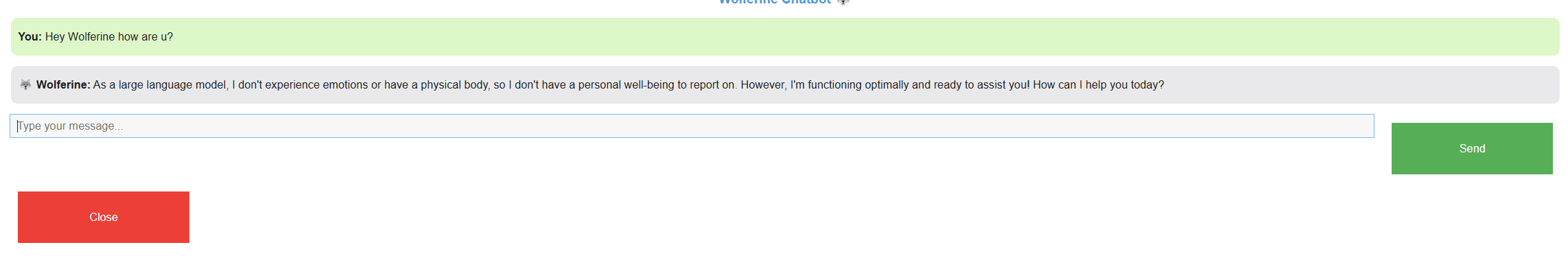
בחלק התחתון של החלון ישנם כפתורי back to main menu אשר לחיצה עליו מנתבת אותנו למסך הראשי של המערכת.

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.

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

בהצלחה!