Fהמכללה האקדמית להנדסה בראודה A black background with purple letters

Description automatically generated

**המחלקה להנדסת תוכנה - מבוא למחשוב ענן**

**פרויקט** - מועד הגשה: **10.2.26**

**הסעיף נלקח מתרגיל בית 1 הסעיף נלקח מתרגיל בית 2 הסעיף נלקח מתרגיל בית 3**

בסיום הסמסטר, מגיש כל צוות סטודנטים את הפרויקט עליו עבדו במהלך הסמסטר. הפרויקט ימומש בסביבת גוגל קולאב, עם מסד נתונים FireBase. ההרצה תהיה מהמחברת, וגיבוי של כל ההגשות יאוחסן בתיקיית גיט של הצוות.

על הפרויקט להכיל את כל האלמנטים הממומשים במלואם כפי שתוארו בתרגילי בית 2+3.

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

-

|  |  |
| --- | --- |
| **קישור למחברת ההגשה:** | https://colab.research.google.com/drive/1-AagDl5SsYEb\_1jCMTgpP86xNU6VXzuN?usp=sharing |
| **קישור לתיקיית גיט:** | https://github.com/GB2braude/PROJ\_ELEPHANT.git |

יש לוודא שניתן להריץ את הפרויקט מהמחברת בלבד , ללא כל צורך בהטענה של קבצים נוספים או יציאה למערכת חיצונית (כגון FLASK). עבודות שלא יעמדו בתנאי זה, יקבלו ציון 0 על אלמנט הקוד.

1. מהות המוצר ומרכיביו

**GREEN ELEPHANT - דשבורד חכם לניטור צמחים בענן**

האפליקציה פותרת את האתגר של מעקב רציף אחרי בריאות הצמחים וזיהוי מוקדם של בעיות, כפי שנדרש במערכות חקלאות מדייקת. היא מיועדת למגדלים ביתיים, לחקלאים, ולכל מי שמעוניין בניהול חכם של גינה או חממה. המערכת מאפשרת להעלות תמונות צמחים, לקבל נתוני חיישני IoT בזמן אמת, לבצע ניתוח AI למחלות, מזיקים ולחץ מים, ולעקוב אחר מגמות ונתונים בצורה ויזואלית ואינטואיטיבית. הענן תומך בפתרון באמצעות אחסון מאובטח של המידע, עיבוד AI מתקדם, סנכרון מתמיד בין החיישנים למשתמש, וגישה מלאה מכל מקום ומכל מכשיר – בכל רגע.

1.1 מודולים ופיצ'רים מעניינים

 אבחון מחלות בצמחים באמצעות תמונה (AI – HuggingFace)

 מנוע חיפוש אקדמי מבוסס RAG

 דשבורד נתוני חיישנים (IoT)

 מערכת Gamification (משימות, נקודות, רמות)

 צ'אטבוט חכם מבוסס Gemini

 המלצות טיפול אוטומטיות מבוססות AI

Microservices 1.2 שנעשה בהם שימוש

| **שירות** | **תפקיד** |
| --- | --- |
| DocumentService | טעינת PDF וחילוץ טקסט |
| TextProcessingService | ניקוי טקסט, stop-words, stemming |
| IndexService | בניית אינדקס הפוך |
| SearchService | חיפוש במסמכים |
| ResultService | דירוג והצגת תוצאות |
| GeminiRAGService | חיבור LLM לתוצאות |

**יתרונות:**

* הפרדה לוגית
* תחזוקה קלה
* סקיילביליות
* אפשרות החלפת שירותים

KPI1.3 מרכזיים

**AI:**

Accuracy

Confidence ממוצע

זמן תגובה

**RAG:**

אחוז שאילתות מוצלחות

דיוק שליפת מסמכים

אחוז תשובות ריקות

**IoT:**

Latency

הצלחת כתיבה ל-Firebase

עקביות נתונים

* 1. ניתוח נתוני עתק

בוצע ניתוח MapReduce על נתוני החיישנים:

* temperature
* humidity
* soil moisture

חישובים:

* min
* max
* average

הנתונים נשמרים ב-Firebase ומוצגים בגרפים בדשבורד.

1. ארכיטקטורת המערכת

2.1 תרשים ומאפיינים מרכזיים (מעודכן)

A diagram of a software company

AI-generated content may be incorrect.

הדיאגרמה מציגה ארכיטקטורה בשכבות (Layered Architecture) המתארת את זרימת המידע מהמשתמש ועד להצגת התוצאות. בשכבה העליונה נמצא ממשק המשתמש מבוסס Gradio, הכולל את כל מסכי המערכת: אבחון מחלות, מנוע חיפוש RAG, צפייה בנתוני חיישנים ודשבורד ויזואלי. פעולות המשתמש מנוהלות על ידי שכבת Event Handlers, המתווכת בין הממשק לשכבת הלוגיקה. שכבת הלוגיקה כוללת שלושה מודולים עיקריים: אבחון מחלות באמצעות HF ושילוב Gemini AI להמלצות טיפול, מודול RAG לשאילתות אקדמיות, ומודול IoT לניתוח והצגת נתוני חיישנים בגרפים. המערכת נשענת על שכבת נתונים הכוללת מאמרי PDF, Firebase, Gemini ו־API של חיישנים, ובסיום מציגה דו"חות אבחון, תשובות חכמות וגרפים ויזואליים. מבנה זה תומך במודולריות, סקיילביליות ותחזוקה קלה.

2.2 Use case מעודכן של פונקציונליות המערכת.

A diagram of a diagram

AI-generated content may be incorrect.

2.3 דרישות פונקציונליות

1. המערכת תאפשר למשתמש להעלות תמונת צמח ולקבל אבחון מחלה אוטומטי באמצעות מודל בינה מלאכותית.
2. המערכת תפיק דו"ח הכולל שם מחלה, הסבר והמלצות טיפול חכמות המבוססות על Gemini AI.
3. המערכת תאפשר למשתמש לשלוח שאילתות למנוע חיפוש אקדמי מבוסס RAG.
4. המערכת תחזיר תשובות מבוססות מאמרים ותציג את מקורות המידע לצורך שקיפות.
5. המערכת תמשוך נתוני חיישנים (טמפרטורה, לחות ולחות קרקע) משרת IoT חיצוני.
6. המערכת תציג נתוני חיישנים בצורה גרפית בדשבורד ויזואלי.
7. המערכת תשמור תוצאות ניתוח נתונים במסד הנתונים Firebase.
8. המערכת תכלול מנגנון משחוק (Gamification) הכולל משימות, ניקוד, רמות והישגים.
9. המערכת תשלב צ'אטבוט חכם מבוסס AI למענה על שאלות המשתמש.
10. כל הפיצ'רים יוצגו באפליקציה אחת ללא מעבר למערכות חיצוניות.
11. דרישות לא פונקציונליות - אתגרים איתם הפרויקט מתמודד (מסווגים לפי קישור WIKIPEDIA).

**Performance ביצועים**  
המערכת תספק תוצאות אבחון וחיפוש בתוך זמן מקסימלי של 10 שניות מרגע בקשת המשתמש.

**Usability שימושיות**  
הממשק יהיה אינטואיטיבי, ברור וידידותי למשתמשים ללא רקע טכנולוגי, תוך שימוש בצבעים, אייקונים והדמיות גרפיות.

**Reliability אמינות**  
המערכת תבצע בדיקות תקינות לנתונים (ולידציה) כדי למנוע הצגת תוצאות שגויות או חלקיות.

**Scalability יכולת התרחבות**  
המערכת תתמוך בעבודה עם מספר רב של משתמשים ונתונים במקביל ללא ירידה בביצועים.

**Availability זמינות**  
המערכת תהיה זמינה רוב שעות היממה ותכלול מנגנוני התאוששות במקרה של תקלה.

**Security אבטחת מידע**  
המערכת תשמור על פרטיות המשתמש ותגן על נתונים באמצעות הרשאות וחיבור מאובטח.

**Maintainability תחזוקתיות**  
הקוד יהיה מודולרי, קריא ומתועד, כדי לאפשר תיקונים והרחבות עתידיות בקלות.

**Portability ניידות**  
המערכת תרוץ בסביבת Google Colab ותהיה נגישה מכל דפדפן ללא התקנות מיוחדות.

**Interoperability יכולת שילוב**  
המערכת תדע לעבוד מול שירותים חיצוניים כגון Gemini API, HuggingFace ו־IoT Server.

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

A screenshot of a computer

AI-generated content may be incorrect.

(72.5+75+75+92.5+65+92.5+92.5+95+92.5+70+97.5)/11=83.6

**פירוש הציון**

* **שימושיות טובה מאוד עד מצוינת**
* **מעל הממוצע (68)**

5. מערכת מלאה ועובדת, הכוללת הצגת נתוני החיישנים שאתם קולטים מאיתנו, מנוע חיפוש RAG, וצאטבוט. יש לוודא שכל המסכים תקינים ועובדים כראוי, **בקולאב**, ולעבוד עם firebase לצורך שמירת   
 האינדקס ונתוני החיישנים.

- על כל המסכים להופיע באפליקציה יחידה, תא בודד. אין לשים פיצ'רים שונים בתאים שונים.

- יש לממש מנוע חיפוש מבוסס RAG המבוסס על מאמרים הרלוונטיים לאפליקציה שלכם. מנוע זה צריך להשתמש ב- LLM חיצוני אשר יחזיר תשובות אמיתיות המבוססות על המאמרים.

- אין להשאיר קישורים או כפתורים למסכים שלא קיימים.

- יש לממש צטבוט שישולב במערכת, מבוסס AI (כפי שלמדתם בתרגול 9), העונה בצורה   
 אינטליגנטית לשאלות המשתמש.

- יש לוודא שניתן להריץ את הפרויקט מהמחברת בלבד , ללא כל צורך בהטענה של קבצים נוספים   
 או יציאה למערכת חיצונית (כגון FLASK). ניתן לעשות שימוש ב gradio, יש לוודא כי האתר עולה ופועל כיאות.

- שימוש ב -microservices (תרגול 8)- עליכם לעשות שימוש לפחות בשניים. פרטו באילו שירותים עשיתם שימוש, ומה היתרון בשימוש בשירותים אלו. (10 נקודות)

שימוש ב Microservices בפרויקט שלנו:

בפרויקט שלנו, השתמשנו בארכיטקטורת מיקרו-שירותים (Microservices) כדי לחלק את המערכת לשירותים עצמאיים ונפרדים, כאשר כל שירות מתמקד בפונקציונליות מסוימתת.

דוגמאות Microservices בפרויקט שלנו:

דוגמאות למיקרו־שירותים בפרויקט שלנו:

1. DocumentService

מה השירות עושה:

שירות זה אחראי על טעינת מסמכים (קבצי PDF), חילוץ הטקסט מהם, וניהול מאגר המסמכים במערכת. השירות מספק גישה לתוכן המסמכים ולשמותיהם עבור שירותים אחרים.

יתרונות:

הפרדה ברורה בין ניהול מסמכים לשאר הלוגיקה של המערכת.

מאפשר החלפה עתידית של מקור הנתונים (למשל מסד נתונים או אחסון בענן) ללא שינוי בשאר השירותים. שיפור תחזוקת הקוד וקריאותו.

2. TextProcessingService

מה השירות עושה:

שירות זה אחראי על עיבוד טקסט, כולל ניקוי נתונים, הסרת מילות עצירה (stop words), ביצוע stemming וסינון מונחים רלוונטיים.

יתרונות:

מימוש עקרון האחריות היחידה (Single Responsibility).

מאפשר שימוש חוזר בעיבוד טקסט עבור אינדוקס, חיפוש ושאילתות

שיפור איכות תוצאות החיפוש והאינדוקס.

3. IndexService

מה השירות עושה:

שירות זה אחראי על בניית אינדקס הפוך (Inverted Index) על בסיס המסמכים שעברו עיבוד טקסט. האינדקס משמש בסיס יעיל לביצוע חיפושים.

יתרונות:

שיפור משמעותי בביצועי החיפוש.

הפרדה בין שלב האינדוקס לשלב החיפוש.

מאפשר הרחבה עתידית (כגון TF-IDF או דירוג מתקדם).

4. SearchService

מה השירות עושה:

שירות זה אחראי על ביצוע חיפושים על בסיס שאילתת המשתמש, תוך שימוש באינדקס הקיים וחישוב ציוני רלוונטיות למסמכים.

יתרונות:

ביצוע חיפושים מהירים ויעילים.

ניתן לשדרוג בקלות לתמיכה באופרטורים לוגיים (AND / OR).

הפרדה ברורה בין לוגיקת חיפוש ללוגיקת הצגה.

5. ResultService

מה השירות עושה:

שירות זה אחראי על דירוג התוצאות, בחירת המסמכים הרלוונטיים ביותר והצגת תוצאות החיפוש בפורמט ידידותי למשתמש.

יתרונות:

שליטה מלאה על אופן הצגת התוצאות.

מאפשר שינוי אלגוריתם הדירוג ללא השפעה על החיפוש עצמו.

שיפור חוויית המשתמש.

6. GeminiRAGService

מה השירות עושה:

שירות זה אחראי על שילוב מודל LLM (Gemini) עם תוצאות החיפוש, יצירת הקשר (context) מתוך המסמכים, ומתן תשובה חכמה ומבוססת מידע למשתמש.

יתרונות:

מאפשר חיפוש מתקדם בסגנון RAG (Retrieval Augmented Generation).

הפרדה בין מנוע החיפוש למנוע ה-AI.

גמישות בהחלפת מודל שפה בעתיד.

היתרונות בשימוש במיקרו־שירותים בפרויקט:

עצמאות שירותים:

כל שירות פועל באופן עצמאי, כך שניתן לתחזק, לעדכן או להרחיב שירות מסוים מבלי להשפיע על שאר המערכת.

שיפור ביצועים:

אינדוקס וחישוב רלוונטיות מתבצעים מראש, מה שמפחית עומס בזמן החיפוש ומשפר זמני תגובה.

גמישות והרחבה עתידית:

הארכיטקטורה מאפשרת הוספת שירותים חדשים (כגון המלצות, סטטיסטיקות או Serverless) בקלות.

התמודדות עם עומסים:

ניתן להקצות משאבים נוספים לשירותים קריטיים כמו חיפוש או RAG, מבלי להשפיע על יתר השירותים.

1. מהם ה -KPI הרלוונטיים בפרויקט שלכם? הסבירו. (10 נקודות)

בפרויקט זה הוגדרו מדדי KPI ברמה **תיאורית ואיכותית**, אשר נועדו לשקף את תחומי הביצוע המרכזיים של המערכת, אך אינם נמדדים בצורה כמותית מלאה במסגרת הפרויקט. מדדים אלו משמשים כבסיס להערכה רעיונית של הצלחת המערכת ולזיהוי נקודות לשיפור עתידי.

**1. מדדי AI ואבחון מחלות**

מדדים אלו מתארים את איכות פעולת רכיב הבינה המלאכותית:

* **דיוק המודל (Accuracy)** – מדד תיאורטי להערכת יכולת המערכת לזהות מחלות בצורה נכונה.
* **רמת ביטחון ממוצעת** – משקפת את תחושת הוודאות של המודל בתחזיותיו, כפי שמוצגת למשתמש.
* **זמן תגובה** – מדד חווייתי המתאר את פרק הזמן מרגע העלאת התמונה ועד קבלת האבחון.

**2. מדדי RAG ומנוע החיפוש האקדמי**

מדדים אלו משקפים את איכות המענה המחקרי של המערכת:

* **אחוז שאילתות מוצלחות** – מדד תיאורי להערכת מספר השאילתות שמובילות לתשובה שימושית עבור המשתמש.
* **דיוק שליפת מסמכים** – הערכה איכותית של התאמת המסמכים שנשלפו לשאלת המשתמש.
* **אחוז תשובות ריקות** – אינדיקציה תיאורית למצבים בהם לא נמצאו מסמכים רלוונטיים.

**3. מדדי IoT ונתוני חיישנים**

מדדים אלו מתמקדים באמינות וזרימת הנתונים:

* **זמן השהיה (Latency)** – מדד תיאורי לזמן שעובר מהתקבלות נתון חיישן ועד הצגתו בדשבורד.
* **הצלחת כתיבה ל-Firebase** – מדד איכותי המתאר את יציבות שמירת הנתונים במסד הנתונים.
* **עקביות נתונים** – בחינה רעיונית של התאמה בין הנתונים שנקלטו מהחיישנים לבין הנתונים המאוחסנים והמעובדים.

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

1. **אתחול ותשתיות**

* **Cell 1**: התקנות + imports.
* **Cell 2**:
  + הורדת firebase-key.json מ־Google Drive לפי DRIVE\_JSON\_ID
  + firebase\_admin.initialize\_app(...) + בדיקת כתיבה: db.reference("test").set({"status":"ok"})
  + drive.mount(...)
  + הורדת תיקיות מ־Drive מקישורים (PDF\_LINK, IOT\_LINK) עם:
    - drive\_folder\_to\_local(url, local, clean)

**2) Gamification**

* **Class GamificationSystem**
  + complete\_task(task\_id) – מסמן משימה, מוסיף נקודות, מעלה רמות ומוסיף achievements.
  + get\_status() – מחזיר Markdown להצגה ב־UI.
  + reset\_daily\_tasks() – מאפס משימות יומיות.

**3) מנוע חיפוש אקדמי (PDF Indexing)**

* **Class AcademicSearchEngine**
  + load\_documents(folder) – טוען PDFים לתוך self.documents.
  + read\_pdf(path) – חילוץ טקסט.
  + process\_text(text) – ניקוי + stemming + סינון לפי KEY\_TERMS.
  + build\_index() – אינדקס מונחים → מסמך → שכיחות.
  + retrieve\_documents(query, top\_k) – דירוג מסמכים לפי התאמת מונחים.

**4) IoT MapReduce + כתיבה ל־Firebase**

* load\_iot\_json(folder) – טוען רשומות JSON.
* map\_sensor\_data(records) – ממפה לערכים מספריים: temperature/humidity/soil.
* reduce\_sensor\_data(mapped) – סטטיסטיקות: count/min/max/avg.
* run\_mapreduce(folder) – מריץ ושומר ל־Firebase תחת iot\_analysis.

**5)RAG עם Gemini + TFIDF Retriever**

* **Class GeminiRAG**
  + create\_context(ranked\_docs, max\_chars\_per\_doc) – יוצר הקשר ממסמכים.
  + query(question, top\_k, ...) – תשובה רק מהמסמכים + מחזיר sources.
* **Class TfidfRetriever**
  + search(query, top\_k) – דמיון קוסיני על TF-IDF.
* PDF chunking helpers:
  + pdf\_to\_text(path), chunk\_text(text, size, overlap), build\_pdf\_retriever(folder)
* HF README chunking:
  + load\_hf\_chunks(model\_id) – מוריד README.md של מודל מה־HF ומחלק ל־chunks.

**6) מודלי AI בפועל**

* Image classification:
  + clf = pipeline("image-classification", model=IMAGE\_MODEL, device=device)
* Text generation להסבר קצר:
  + llm = pipeline("text2text-generation", model="google/flan-t5-small", device=device)
* Gemini משמש ל־RAG וגם ל־treatment bullets

**פונקציות מרכזיות (Core Functions)**

* parse\_label(label) – פיצול תווית "Plant\_\_\_Disease" ל־(plant, raw, clean).
* rag\_explain(plant, disease\_raw, disease\_clean)
  + אם “healthy” → תשובה קצרה.
  + אחרת: בונה שאילתה, מושך הקשר מ־PDF TFIDF ומ־HF TFIDF, ומפיק הסבר באמצעות llm (Flan-T5).
* gemini\_treatment(plant, disease) – מבקש מג’מיני 5–7 נקודות טיפול.
* diagnose(img) – צינור מלא: תמונה → ניבוי → הסבר → טיפול → נקודות משימה.
* IoT UI:
  + get\_sensor\_data(feed, limit) – קריאה מהשרת BASE\_URL/history.
  + plot\_all\_feeds(limit) – גרפים ל־temperature/humidity/soil.
* Research UI:
  + query\_handler(q, top\_k) – מפעיל rag\_system.query + gamification.
* Chatbot:
  + helper\_bot\_reply(msg, history) – “עוזר אישי” דרך Gemini, עם היסטוריה.

**קטעי קוד/תבניות UI מעניינים (שווה לציין בתיק למתכנת)**

**1) Gradio עם Tabs + חלוקה פונקציונלית**

* 5 טאבים: Diagnosis / Research / Sensor Data / Dashboard / Daily Missions
* כל Tab מחובר לפונקציות דרך .click(...) ו־.submit(...).

**2) CSS מותאם + Floating Chat**

קטעים בולטים:

* .floating-chat-btn כפתור עגול קבוע למטה־ימין.
* .chat-window חלון צ’אט “צף” עם visible=False ו־gr.State לשליטה.
* toggle\_chat(visible) מחזיר עדכון gr.update(visible=...).

**3) Dashboard עם gr.Plot**

פונקציה שמחזירה **רשימת figures** לפי סדר, וממופה ל־3 רכיבי Plot שונים.

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

**א. הסבר כללי על המערכת**

**Plant Disease Research & Monitoring System** היא פלטפורמה שמטרתה:

1. לזהות מחלות בצמחים לפי תמונת עלה/צמח (AI Image Classification).
2. להסביר על המחלה ולהציע טיפול (שילוב RAG + מודלי שפה).
3. להציג נתוני חיישנים (IoT) בזמן אמת + לוח מחוונים גרפי.
4. להוסיף משחקיות (משימות יומיות, ניקוד, רמות).

**ב. פירוט מסכים (Tabs) + מה עושים בכל מסך**

**1) אבחון מחלה**

* **מה רואים:** העלאת תמונה + כפתור Analyze + תיבת Report.
* **מה עושים:** מעלים תמונת צמח ולוחצים “Analyze”.
* **מה מתקבל:**
  + שם הצמח והמחלה (או Healthy)
  + הסבר קצר על בסיס מקורות
  + “Treatment” עם 5–7 נקודות
  + נקודות משחק (10pts) על השלמת משימה.

**2) עוזר מחקר**

* **מה רואים:** תיבת שאלה + Slider לכמות מסמכים + כפתור Search + Answer + Sources.
* **מה עושים:** כותבים שאלה (למשל “symptoms of tomato late blight”) ובוחרים Docs.
* **מה מתקבל:**
  + תשובה (אמורה להתבסס על המסמכים שניטענו מה־PDFים)
  + רשימת מקורות שהיו רלוונטיים
  + נקודות משחק (20pts).

**3) נתוני חיישנים**

* **מה רואים:** Dropdown לבחירת feed + slider לכמות דגימות + כפתור Fetch + תיבת Values.
* **מה עושים:** בוחרים humidity/soil/temperature ולוחצים Fetch.
* **מה מתקבל:** רשימת דגימות כפי שהשרת מחזיר + נקודות (15pts).

**4) לוח מחוונים**

* **מה רואים:** Slider “Points” (כמות נקודות/דגימות בגרף) + כפתור Refresh + 3 גרפים.
* **מה עושים:** בוחרים כמה דגימות להציג ולוחצים Refresh.
* **מה מתקבל:** שלושה גרפים: טמפרטורה, לחות, אדמה.

**5) משימות יומיות**

* **מה רואים:** הסבר “איך לשחק” + סטטוס משימות + כפתורי Status/Reset.
* **מה עושים:** משלימים פעולות בטאבים אחרים כדי לצבור נקודות.
* **Reset:** מאפס משימות יומיות.

**6) צ’אט בוט**

* **מה רואים:** כפתור 💬 קבוע במסך. לחיצה פותחת חלון צ’אט.
* **מה עושים:** שואלים שאלות על המערכת, איך להשתמש, מה משמעות תוצאות וכו’

**ג. מעברים בין מסכים (User Flow)**

* המשתמש “מנווט” בין טאבים לפי צורך:
  + אם יש תמונה → Diagnosis.
  + אם רוצה מידע ממסמכים → Research Assistant.
  + אם רוצה נתוני חיישנים → Sensor Data / Dashboard.
  + כדי לראות התקדמות וניקוד → Daily Missions.
  + לעזרה בכל רגע → כפתור הצ’אט הצף.

אין “מסכי מעבר” קלאסיים; הכל באותו דף דרך Tabs

**ד. טעויות אפשריות והסברים למשתמש (Error Handling)**

**Diagnosis**

* **לא העלית תמונה:** מתקבל "⚠️ Upload image first".
* **מודל נכשל / בעיית טעינה:** עלול להחזיר שגיאה כללית או תשובה חלקית.

**Research Assistant**

* **שאלה ריקה:** "⚠️ Enter question".
* **אין מסמכים רלוונטיים:** "❌ No relevant documents found."

**Sensor Data / Dashboard**

* **השרת לא עונה / תקלה ברשת:** מתקבל "❌ <error>".
* **מבנה JSON לא צפוי מהשרת:** "❌ Error"כאשר "data" לא קיים.

**Firebase / Drive**

* בעיות הרשאות/מפתח לא תקין: כשל בחיבור Firebase או הורדה מ־Drive.

9. בשבוע 12 בקורס הצגתם את המערכת שלכם.

קיבלתם מאיתנו את המשובים

יש להגיש את הטבלה הבאה , תוך התיחסות למשובים שקיבלתם:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **הערת משוב** | **האם נדרש שינוי במערכת בעקבות ההערה?** | **נימוק** |
| **"גרפים בסיסיים שאפשר לשפר אותם או הנראות שלהם, לא משהו מיוחד"** | כן | למרות שהגרפים פועלים ומציגים נתונים נכונים, המשוב מצביע על נראות בסיסית. נדרש שיפור ויזואלי (עיצוב, צבעים, גודל). |
| **"מעט קטנים בתצוגה אבל סהכ מוצגים יפה"** | כן | הגרפים קיימים ומתפקדים, אך גודל התצוגה פוגע בקריאות. ניתן להגדיל ולהתאים למסכים שונים. |
| **"איך לטפל במחלה כשמעלים תמונות!"** | לא | המערכת אכן מספקת הסבר וטיפול לאחר העלאת תמונה. הפונקציונליות קיימת ועובדת. |
| **"מימוש הhugging face במערכת"** | לא | השימוש ב-Hugging Face מיושם במערכת ומוצג בסרטון. המשוב מצביע על הערכה, לא על חוסר. |
| **"לדעתִי לא היה משהו מעניין במיוחד"** | לא (אך לשקול שיפור חווייתי) | מדובר בדעה סובייקטיבית כללית ללא הצבעה על רכיב חסר. ניתן לשקול חיזוק חוויית המשתמש או הדגשת ערך המערכת. |

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** | **תגובה** | **האם בכוונת הצוות לבצע שינוי בעקבות ההערה? יש לפרט. אם כן, מה השינוי. אם לא, מדוע** |
| פשטות | כל שירות ממוקד בתפקיד אחד: לדוגמה, שירות אחד . זה מקל על קריאה והבנה של הלוגיקה. | לא נדרש שינוי נוסף, פשטות כבר השתפרה בעקבות החלוקה. |
| מימוש | כל שירות ניתן לפריסה נפרדת, קל יותר להוסיף תכונות חדשות או לשנות שירות מסוים בלי להשפיע על אחרים. | לא, המימוש כבר מותאם למיקרוסרביסים |
| מודלריות | כל Microservice הוא מודול עצמאי עם API מוגדר. | לא נדרש שינוי נוסף, אך כדאי לחלק אותו לFOLDERES ותיקיות |
| יעילות | ניתן להריץ שירותים שונים על שרתים שונים או בקונטיינרים נפרדים | כן, ניתן להוסיף caching ושיפור בקריאות I/O בין שירותים. |
| באגים | חלוקת השירותים מקטינה את הסיכוי לבאגים בין מודולים | כן, ניתן לשפר עם בדיקות אינטגרציה בין שירותים ובדיקות חוזרות |
| טיפול בשגיאות | ניתן לטפל בשגיאות בצורה ממוקדת בכל Microservice, ולהחזיר סטטוסים ברורים ל־API. | כן, כדאי להוסיף centralized logging ו־monitoring לכל השירותים כדי לזהות תקלות בזמן אמת. |

12.בונוס 2 נק- רפלקציה **אישית** למילוי על ידי כל סטודנט בקורס על הפרויקט. נא למלא את השאלון בטופס:

https://docs.google.com/forms/d/e/1FAIpQLSfuoqW69m6aZDNnqOGHnwwffjUVEIxXcV7TMFF7Qng-aCTuZA/viewform

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

רוב השימושים בכלי AI של Chatgpt ו Gemini הייתה בשביל שיפור ה UI של המערכת, היו מלא באגים, לפעמים ביצענו שינויים בעיצוב בקוד אך לא השתנה שום דבר במערכת, העבודה עם ה UI ב google colab מאוד קשה, ולכן רוב הפרומפטים היו מהצורה הבאה: “I just updated this part of the code {Code} to have this new styling {styling part} but nothing changed in the UI, why is that?”

רוב התשובות היו לשנות מ Vbox או Hbox למשל ל HTML כי יש לו יותר אופציות עיצוב וכו.

השתמשנו גם באתר זה: <https://www.w3schools.com/python/default.asp> כרענון לפייתון.

רוב המימוש של הקוד לקינו עזרה מחקי הקוד נמצאים בתרגולים השבועים

הדו"ח יוגש לאחר סיום הסמסטר.

הנחיות:

1. יש להגיש את התרגיל בצוותים שנקבעו.
2. חובת הצגה בקורס על כל הסטודנטים בקבוצה. סטודנט אשר לא יציג, לא יקבל ציון בקורס.
3. שימו לב כי כל העבודות חייבות להיות שונות זו מזו. עבודות שייראו דומות ייפסלו ויינתן עליהן ציון 0.

בהצלחה!