המכללה האקדמית להנדסה בראודה A black background with purple letters

Description automatically generated

המחלקה להנדסת תוכנה - מעבדה במודלים אקולוגיים

פרויקט – מעודכן 17.6

מועד הצגה: 17/6/25- בזמן ההרצאה

מועד הגשה: **10.7.25**

בסיום הסמסטר, מגיש כל צוות סטודנטים את הפרויקט עליו עבדו במהלך הסמסטר. הפרויקט יאוחסן בתיקיית גיט של הצוות.

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

- קישור למחברת הגוגל קולאב (יש לוודא שהיא פומבית)

- קישור לתיקיית הגיט של הצוות.

***Kareem Zeedan: 212101943***

***Rami Taha: 207410523***

[GitHub](https://github.com/ramitaha13/Ecological-models)

Link of WebSite :<https://phase2-smart-farm.vercel.app/>

Link of colab : <https://colab.research.google.com/drive/1pdcpeSIjJu6ejRpvNfY040uA2sb5DAvP?usp=sharing>

1. מבוא - הסבר האתגר, שאלת/שאלות המחקר. מה נעשה עד כה (כולל רפרנסים למאמרים).

אחד מהאתגרים המרכזיים כיום בעולם החקלאות הוא מחסור במים וניהול לא יעיל של השקיה, במיוחד באזורים חמים כמו צפון ישראל. חקלאים לעיתים משקים בצורה אחידה מבלי להתחשב בצרכים הספציפיים של כל גידול או בתנאי השטח המשתנים, דבר שעלול לגרום לבזבוז מים או חוסר השקיה.

### שאלת המחקר

כיצד ניתן לייעל את מערך ההשקיה בשטח חקלאי תוך שימוש בניתוח מרחבי של נתוני חיישנים בשילוב תחזיות מזג אוויר?

או - מתי כדאי להשקות ומתי כדי לעצור תוך כדי שמירה על בריאות הצמח

מערכות השקיה חכמות קיימות, כמו CropX ו-Netafim, מבוססות על חיישני לחות ותקשורת סלולרית, אך רבות מהן אינן מותאמות לתנאי שדה משתנים או חסרות יכולת חיזוי. בנוסף, חלקן לא כוללות ממשק משתמש נגיש או אינטגרציה עם תחזיות מזג אוויר, מה שמוביל לחוסר יעילות ולבזבוז מים. הפרויקט שלנו שואף להתגבר על מגבלות אלו באמצעות ניתוח מרחבי בזמן אמת וחיזוי השקיה מותאם אישית.

<https://cropx.com/>

<https://www.netafim.com/en/digital-farming>

<https://www.valleyirrigation.com/>

Froiz-Míguez, I. et al. (2020). *Design, Implementation, and Empirical Validation of an IoT Smart*

*Ndunagu, J. N., et al. (2022). Development of a Wireless Sensor Network and IoT‐based Smart*

2. סקירת ספרות :   
א. מה המחקרים שנעשו בעבר בתחום זה (ניתן להתבסס על מחקרים מהמטלה האישית)

המחקר בתחום ההשקיה החכמה משלב טכנולוגיות חדשניות לניהול מים מדויק ויעיל. להלן מחקרים מרכזיים שתרמו לביסוס התחום:

* **"Exploring the Integration of Industry 4.0 Technologies in Agriculture: A Comprehensive Bibliometric Review" – Fasciolo et al. (2024)**מציג סקירה רחבה של השילוב בין טכנולוגיות Industry 4.0 לחקלאות, כולל IoT, ענן, חיישנים ואוטומציה. המחקר מדגיש כיצד פתרונות טכנולוגיים יכולים לשפר את ניהול המים והיבול.
* **"Design, Implementation, and Empirical Validation of an IoT Smart Irrigation System for Fog Computing Applications Based on LoRa and LoRaWAN Sensor Nodes" – Froiz-Míguez et al. (2020)**מציע מערכת השקיה מבוססת IoT שנבדקה אמפירית, תוך שימוש בטכנולוגיות תקשורת מתקדמות וניתוח נתונים בשטח.
* **"Development of a Wireless Sensor Network and IoT‐based Smart Irrigation System" – Ndunagu et al. (2022)**מתאר הקמה של רשת חיישנים חכמה לניטור קרקע והשקיה, עם דגש על איסוף נתונים מרחוק ותגובה בזמן אמת.
* **"An Overview of Smart Irrigation Systems Using IoT" – Obaideen et al. (2022)**סקירה כללית של מערכות השקיה חכמות, כולל אתגרים, סוגי חיישנים, ופתרונות שונים שנבדקו במחקר.
* **"IoT and Machine Learning Approaches for Automation of Farm Irrigation System" – Vij et al. (2020)**עוסק בשילוב של IoT ולמידת מכונה על מנת לאפשר חיזוי אוטומטי של השקיה והתאמה לתנאי הקרקע בזמן אמת.
* **"Smart Irrigation System in Agriculture" – Vimal et al. (2021)**מציג פתרון פשוט ויעיל להשקיה מבוססת חיישנים, תוך התמקדות ביישום בשטח חקלאי פעיל.
* **"A Secure IoT-Based Irrigation System for Precision Agriculture Using the Expeditious Cipher" – Fathy & Ali (2023)**עוסק באבטחת מערכות השקיה חכמות, עם פרוטוקולי הצפנה לשמירה על שלמות המידע המועבר בין החיישנים לשרת.
* **"Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies" – Friha et al. (2021)**סוקר טכנולוגיות IoT מתקדמות לחקלאות חכמה, כולל תקשורת, ניתוח נתונים, ואינטגרציה עם מערכות קיימות.
* **"IOT Based Smart Irrigation System" – Ragab et al. (2022)**מתאר פיתוח של מערכת השקיה מבוססת IoT עם רכיבים זולים, מתאימה במיוחד לחוות קטנות ובינוניות.
* **"Economic Impacts of Climate Change on Vegetative Agriculture Markets in Israel" – Zelingher et al. (2019)**מדגיש את ההשלכות הכלכליות של שינויי אקלים על חקלאות בישראל, ותומך בצורך בפתרונות השקיה חכמים.
* **"Water Crisis: From Conflict to Cooperation" – Sivakumar (2011)**עוסק במשבר המים ברמה גלובלית, ומשמש כרקע תיאורטי לחשיבות של חדשנות ופתרונות אוטומטיים לניהול משק מים.

ב.אילו אלגוריתמים/שיטות נחקרו

* Fasciolo et al. (2024) – "Exploring the Integration of Industry 4.0 Technologies in Agriculture"
  + השיטות שנחקרו: שילוב של בינה מלאכותית, אוטומציה, ניתוח ביג דאטה, ותקשורת IoT במסגרת מערכות חקלאיות.
  + מודגש השימוש ב־predictive analytics ובמערכות הסתגלות חכמות.
* Froiz-Míguez et al. (2020) – "Design and Empirical Validation of an IoT Smart Irrigation System"
  + השיטות שנחקרו: תקשורת LoRa / LoRaWAN, שימוש ב־Fog computing לעיבוד מידע בקצה הרשת, ועיבוד real-time sensor data.
  + הדגש הוא על ארכיטקטורת מערכת יעילה מבחינת עיבוד וזמני תגובה.
* Ndunagu et al. (2022) – "Development of a Wireless Sensor Network and IoT-based Smart Irrigation System"
  + נעשה שימוש ב־רשת חיישנים אלחוטית (WSN) ובמערכת בקרת סף (threshold-based control) פשוטה.
  + הדגש הוא על שליטה על השקיה על סמך ערכי חיישנים בסיסיים – ללא למידת מכונה.
* Obaideen et al. (2022) – "An Overview of Smart Irrigation Systems Using IoT"
  + סקירה כללית של שיטות כמו שליטה אוטומטית לפי חיישנים, שימוש ב־תחזיות מזג אוויר, ויישום מערכות חכמות מבוססות ענן.
  + מדווחים על שימוש בטכנולוגיות כמו MQTT ו־RESTful APIs.
* Vij et al. (2020) – "IoT and Machine Learning Approaches for Automation of Farm Irrigation"
  + מחקר זה משלב למידת מכונה (Machine Learning) עם IoT לצורך חיזוי השקיה (predictive irrigation scheduling).
  + האלגוריתמים כללו מודלים של Classification & Regression Trees, Random Forest, ו־Neural Networksלחיזוי צרכי מים.
* Vimal et al. (2021) – "Smart Irrigation System in Agriculture"
  + מבוסס על אלגוריתם פשוט של בקרת ערך סף (threshold-based), כאשר ההשקיה מופעלת או מושבתת לפי קריאת לחות.
  + השיטה מדגישה יעילות בצריכת מים תוך שימוש בטכנולוגיות בסיסיות ונגישות.
* Fathy & Ali (2023) – "A Secure IoT-Based Irrigation System Using the Expeditious Cipher"
  + הדגש כאן הוא על אבטחת המידע – שימוש באלגוריתם Expeditious Cipher להצפנת נתונים בין החיישנים לבין הענן.
  + בנוסף, השימוש בפרוטוקול MQTT עם TLS encryption לתקשורת מאובטחת.
* Friha et al. (2021) – "Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey"
  + מציינים שימוש בשיטות כמו Cloud-based AI, Remote Sensing, Decision Support Systems, edge computing, ועוד.
  + מחקר זה סוקר שיטות רבות אך לא מתמקד באלגוריתם מסוים.
* Ragab et al. (2022) – "IoT Based Smart Irrigation System"
  + שימוש באלגוריתמים פשוטים מבוססי בקרת סף עם שילוב של בקרי מיקרו (microcontrollers)וחיישנים לחותיים.

3. שיטה וממצאים :

א. אילו אלגוריתמים נבחרו (ניתן להתבסס על תרגיל בית 2)

בפרויקט נעשה שימוש באלגוריתם **PCA (Principal Component Analysis)** כדי לזהות אילו פרמטרים סביבתיים (כגון טמפרטורת אוויר, לחות באדמה, ולחות באוויר) משפיעים בצורה המשמעותית ביותר על הצורך בהשקיה. לאחר מכן יושם מודל קריגינג (**Kriging)** – טכניקת אינטרפולציה מרחבית, שמייצרת מפת חום על בסיס נתוני החיישנים, ומציגה אזורים בשטח החקלאי לפי רמת הצורך בהשקיה (מאדום = צורך גבוה ועד ירוק = אין צורך בהשקיה).

ב. אילו שיטות איסוף מידע התבצעו בפרויקט.

המידע נאסף באמצעות חיישנים תת-קרקעיים שהוצבו בעומקים שונים בשדה החקלאי. כל חיישן מדד את לחות הקרקע והטמפרטורה, ובנוסף נאספו נתוני לחות אוויר. המידע נשלח בזמן אמת דרך פרוטוקול **MQTT** ונשמר במסד נתונים **Firebase**. כמו כן, בוצעה אינטגרציה עם **API** לחיזוי מזג אוויר, אשר איפשרה ניתוח תחזיות גשם לשבוע הקרוב, לשם קבלת החלטות השקיה חכמות.

ג. מה היו הממצאים המרכזיים - מספרית וגרפית.

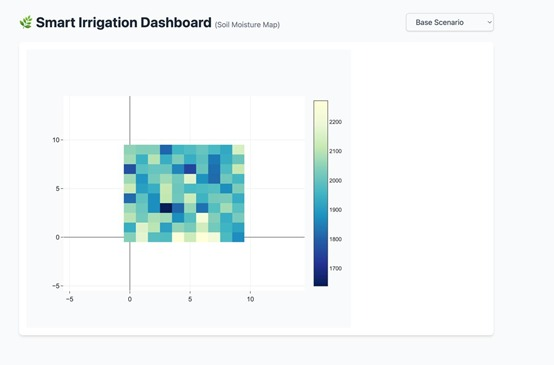
נמצא כי **PC1** (הרכיב העיקרי הראשון) הסביר מעל **60%** מהשונות בנתונים והושפע בעיקר מלחות הקרקע והאוויר.

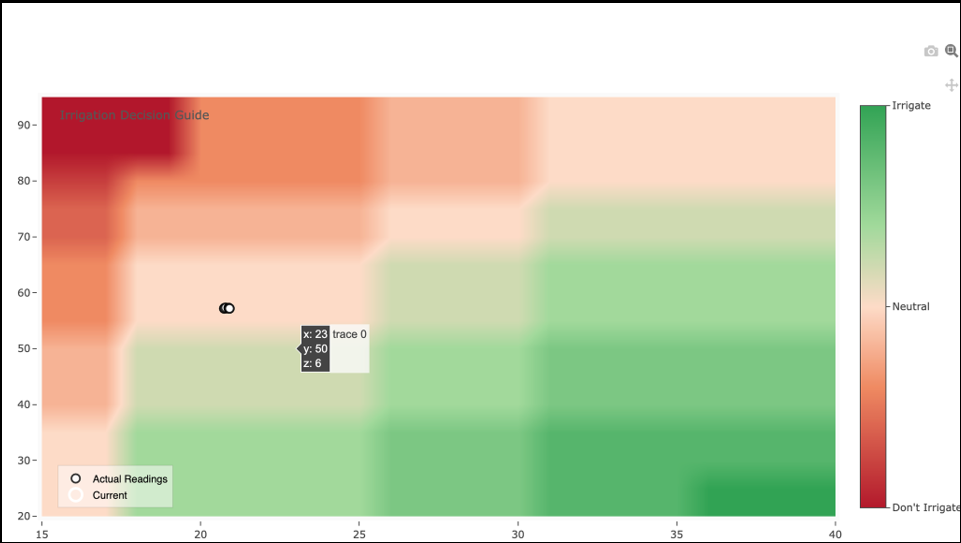
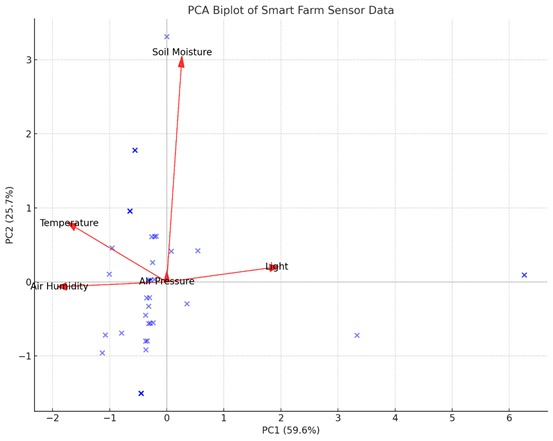
**PC2** הסביר כ-25% נוספים, והבליט את התרומה של טמפרטורת האוויר.

מפת הקריגינג הציגה אזורים ברורים בשדה עם צורך מיידי בהשקיה (אזורים אדומים), לעומת אזורים רוויים יותר (אזורים ירוקים).

כמו כן, ניתן היה לראות מגמה עונתית בהתפלגות הלחות בהתאם לחיזוי הגשם (לפי נתונים חודשיים).

המידע הגרפי הוצג למשתמש באתר האינטרנט בפורמט **Heatmap**, גרפים יומיים, והתראות השקיה.





4. דיון:

א. מענה על שאלות המחקר - מה המסקנות שניתן להסיק מהממצאים?

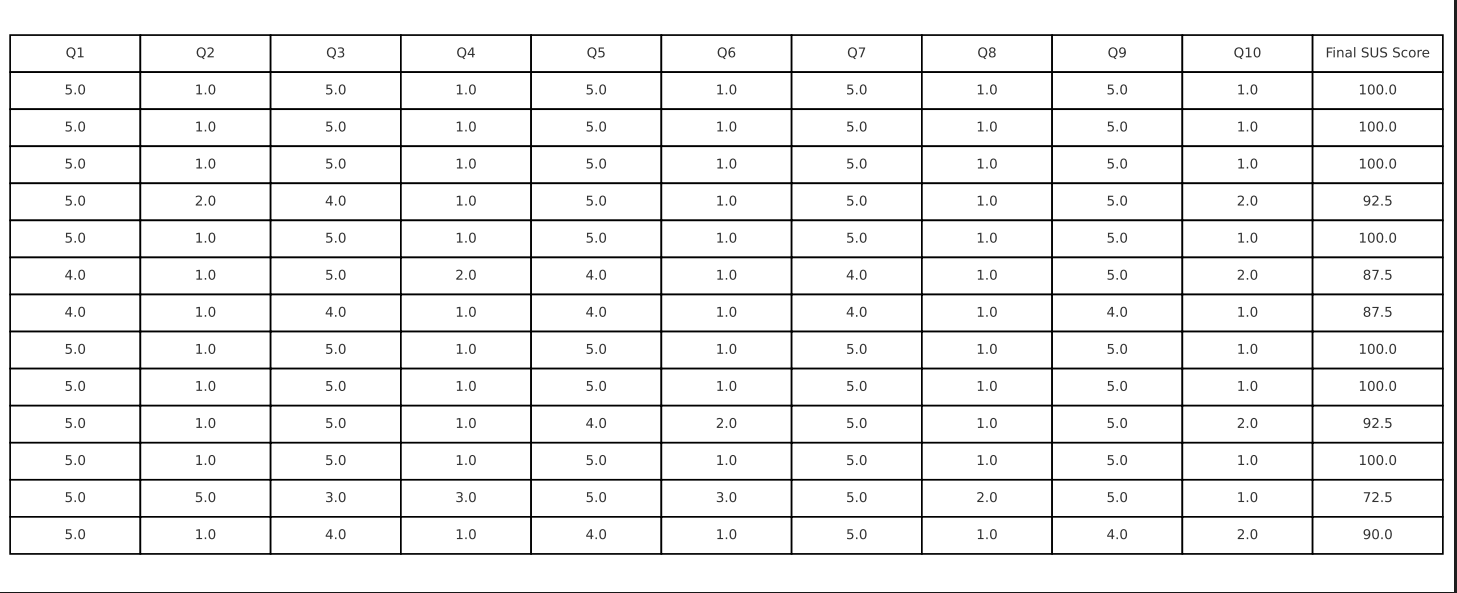
הממצאים מאשרים כי ניתן לייעל את מערך ההשקיה בשטח חקלאי בעזרת שילוב של ניתוח מרחבי (קריגינג) ו-ניתוח סטטיסטי (**PCA)** של נתוני חיישנים תת-קרקעיים. בעזרת חלוקה מדויקת של השדה לאזורים עם רמות השקיה משתנות, ניתן:

* להפחית בזבוז מים משמעותית,
* להשקות רק בעת הצורך ובמיקום המדויק, בהתאם לנתונים הסביבתיים.
* לשלב תחזיות מזג אוויר לצורך השקיה חזויה מראש – דבר שחוסך השקיה מיותרת בימים גשומים.  
   בנוסף, חוויית המשתמש שנבנתה בממשק הדיגיטלי הייתה אינטואיטיבית, ואפשרה שליטה ובקרה מלאה בזמן אמת על כלל המערכת.

ב. כיוונים להמשך.

* פיתוח אלגוריתם מבוסס בינה מלאכותית שיבצע אופטימיזציה אוטומטית של לוחות הזמנים להשקיה לפי התנהגות היסטורית של הקרקע, סוג הצמח והתחזיות.
* הוספת סוגי חיישנים נוספים, כגון חיישני אור או לחות עלים, לקבלת תמונה רחבה יותר של מצב הצמח בפועל.
* הרחבת הפרויקט למספר שדות שונים והשוואת ביצועים ביניהם.
* חיבור למערכת השקיה קיימת כמו GSI באופן אוטומטי כדי להפוך את ההשקיה כולה לאוטונומית.
* יצירת אפליקציה לנייד למעקב ושליטה מרחוק.

5 חישוב והסבר של ציון SUS (מסטודיו שבוע 12)



94/100

ביצענו חישוב של ציון השימושיות (SUS) על סמך 10 שאלות דירוג. התשובות עובדו לפי שיטת הניקוד הסטנדרטית של SUS, כאשר ציונים גבוהים בשאלות חיוביות מקבלים תוספת, ובשליליות – הפחתה. כל ציון הוכפל ב-2.5, והתוצאה הממוצעת של כל המשתמשים היא **94**, מה שמעיד על מערכת מאוד נגישה, נוחה, ואינטואיטיבית לשימוש.

**Here we focus mainly on the course related screens**

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

### ��️ קובץ: **GeminiPlantAnalyzer.jsx**

#### �� תיאור כללי:

* רכיב React המשמש כדף המרכזי לניתוח תמונות של צמחים.
* המשתמש יכול להעלות תמונה של צמח ולהוסיף הוראות נוספות (אופציונלי).
* הרכיב מתקשר עם **Gemini API** של **Google** לקבלת ניתוח שכולל:
  + זיהוי סוג הצמח
  + הערכת נזק
  + צרכי השקיה
  + מצב בריאותי כללי
  + המלצות לטיפול
* הממשק תומך בשתי שפות: אנגלית ו־עברית, כולל התאמות לכיוון טקסט (RTL).

### �� אובייקטים ו־**State** מרכזיים:

* selectedImage – התמונה שהמשתמש העלה.
* imagePreview – תצוגה מקדימה של התמונה בפורמט base64.
* language – שפת הממשק הנוכחית ("english" או "hebrew").
* uiText – אובייקט שמכיל את כל טקסטי ה־UI בשתי השפות.
* plantIdentification, damageAssessment, wateringNeeds, overallHealth, recommendations – תוצאות ניתוח הצמח.
* API\_KEY, API\_URL – הגדרות חיבור ל־Gemini API.
* additionalPrompt – הוראות נוספות מהמשתמש (אם סופקו).
* isAnalyzing, analysisError – סטייטים המנהלים מצב טעינה ושגיאות.

### ⚙️ פונקציות ומה כל אחת עושה:

* **handleImageChange** קוראת את קובץ התמונה, יוצרת תצוגה מקדימה ומאפסת את תוצאות הניתוח הקודמות.
* **resetAnalysis()** מאפסת את כל שדות הניתוח ומנקה שגיאות.
* **handleBackClick()** מחזיר את המשתמש לעמוד הקודם דרך useNavigate.
* **handleLanguageToggle()** מחליף בין עברית ואנגלית, ומאפס את תוצאות הניתוח.
* **parseStructuredResponse(text)** מפענחת את תגובת Gemini לפי תגים (כגון [PLANT\_TYPE]), ומעדכנת את ה־state. אם אין מבנה מסודר – מפצלת לפסקאות.
* **formatText(text)** מעצב את הטקסט להצגה: ממיר נקודות לרשימות, מדגיש טקסט מודגש, ומטפל בכיוון טקסט לפי שפה.
* **handleAnalyzeImage()** הפונקציה המרכזית:
  + ממירה את התמונה ל־base64
  + בונה prompt מותאם שפה + הוראות המשתמש
  + שולחת בקשה ל־Gemini
  + מפענחת את התשובה או מציגה שגיאה
* **convertFileToBase64** ממירה את קובץ התמונה למחרוזת base64.

## �� קובץ: **SmartFarmAdvisor.jsx**

### �� תיאור קצר:

רכיב React מרכזי לאפליקציה "Smart Farm Advisor" המציג נתוני מזג אוויר עדכניים מכרמיאל, כולל טמפרטורות קרקע מרובות עומקים, ומספק המלצות חקלאיות מותאמות אישית על סמך ניתוח AI באמצעות Google Gemini API. הרכיב תומך בדו-לשוניות (עברית/אנגלית) ובחירת סוג צמח.

### �� אובייקטים עיקריים:

* **useState, useEffect** – לניהול מצב וטעינה ראשונית.
* **useNavigate** – לניווט בין דפים.
* **weatherData** – אובייקט שמכיל את כל הנתונים המטאורולוגיים המעובדים של כרמיאל.
* **language** – מצב שפה (עברית / אנגלית).
* **uiText** – אובייקט לתרגום טקסטים.
* **commonPlants** – רשימת צמחים לבחירה ע"י המשתמש.
* **Gemini API** – משמש לשליחת פרומפטים ולקבלת המלצות AI.
* **sectionTags** – מזהים עבור חלקי התשובה של ה-AI.

### ⚙️ שמות הפונקציות ומה כל אחת עושה:

* **fetchKarmielWeatherData** – שואבת נתוני מזג אוויר בזמן אמת מממשק Open-Meteo, כולל טמפרטורת קרקע מרובת עומקים.
* **getMoistureStatus** – מחזירה סטטוס לחות קרקע (לדוגמה: "Very Dry", "Wet") לפי ערך מספרי.
* **getAiAdvice** – יוצרת פרומפט מפורט לפי נתונים קיימים, שולחת ל-Gemini ומעבירה את התוצאה לעיבוד.
* **parseStructuredAdvice** – מפענחת את תגובות ה-AI ומעדכנת את החלקים השונים של ההמלצות.
* **formatAdviceText** – מעצבת את טקסט ההמלצות לפורמט HTML (רשימות מודגשות וכו').
* **handlePlantSelect** – משנה את הצמח הנבחר לפי בחירת המשתמש.
* **getSelectedPlantName** – מחזירה את שם הצמח הנבחר לפי השפה הנוכחית.
* **handleGetAdvice** – מפעילה את הפונקציה לקבלת המלצות AI.
* **handleBackClick** – מחזירה את המשתמש לדף הקודם.
* **handleRefresh** – מרעננת את נתוני מזג האוויר מכרמיאל.
* **handleLanguageToggle** – מחליפה בין עברית לאנגלית ומנקה את ההמלצות הקיימות.

📄 קובץ: **SensorAnalytics.jsx**

🧩 תיאור קצר:  
רכיב React להצגת גרפים וסטטיסטיקות חכמות של נתוני חיישני סביבה בחווה חקלאית, כולל טמפרטורה, אור, לחות, לחץ ולחות קרקע. הרכיב שואב נתונים מ־Firebase Firestore, מחשב מגמות וסטטיסטיקות, מציג מידע בגרפים דינמיים עם אפשרות לבחור טווחי זמן ולהוריד את הנתונים כקובץ CSV. כולל ממשק משתמש עשיר עם אפשרות לבחור אילו חיישנים יוצגו ובחירת טווח תאריכים מותאם אישית.

🧱 אובייקטים עיקריים:

* useState, useEffect – לניהול מצב הרכיב וטעינה ראשונית.
* useNavigate – משמש לניווט חזרה לעמוד קודם.
* firebase/firestore – לאחזור נתונים ממסד הנתונים בענן.
* combinedData – מערך נתונים משולב לכל החיישנים.
* temperatureData, lightData, soilData, humidityData, pressureData – מערכים נפרדים לכל חיישן.
* stats – אובייקט המרכז מידע סטטיסטי (נוכחי, ממוצע, מינימום, מקסימום, מגמה) עבור כל חיישן.
* customDateRange – מאפשר לבחור טווח תאריכים מותאם אישית.
* selectedSensors – מערך חיישנים שנבחרו להצגה על ידי המשתמש.
* recharts, LineChart, Line, YAxis, Tooltip, Legend – ספריית הגרפים להצגת המידע בצורה ויזואלית.

⚙️ שמות הפונקציות ומה כל אחת עושה:

* **fetchSensorData** – שואבת נתוני חיישנים לפי טווח זמן נבחר, מעבדת את המידע, מחשבת סטטיסטיקות ומעדכנת את המצב.
* **processCombinedData** – מאחדת נתוני חיישנים שונים לפי זמן כדי ליצור גרף מאוחד.
* **calculateTrend** – מחשבת מגמה (עולה / יורדת / יציבה) על בסיס שיפוע של רגרסיה ליניארית.
* **handleBackClick** – מחזירה את המשתמש לדף הקודם.
* **handleRefresh** – טוענת מחדש את הנתונים מהמסד בטווח הזמן הנוכחי.
* **handleTimeRangeChange** – משנה את טווח הזמן הנבחר ושואבת בהתאם את הנתונים.
* **handleSensorToggle** – מוסיפה או מסירה חיישן מהרשימה הנבחרת.
* **handleCustomDateSubmit** – מפעילה שאילתה לפי טווח תאריכים מותאם אישית.
* **exportData** – ממירה את נתוני החיישנים לפורמט **CSV** ויוזמת הורדה למשתמש.
* **getTrendColor** – מחזירה צבע טקסט לפי מגמת הנתונים (עלייה, ירידה, יציב).
* **getTrendIcon** – מחזירה אייקון לפי מגמת החיישן.
* **getSensorColor** – מחזירה צבע ייחודי לכל חיישן לצורך גרפים.
* **getSensorUnit** – מחזירה את יחידת המידה עבור כל חיישן (°**C, %, hPa** וכו').
* **getSensorLabel** – מחזירה תווית טקסט לכל חיישן לצורך הצגה למשתמש.
* **CustomTooltip** – מציג מידע מפורט בעת ריחוף על נקודה בגרף (כולל צבע, שם ויחידת מידה).
* 📄 קובץ: SensorsPage.jsx
* 🧩 תיאור קצר:
  + רכיב React המציג לוח בקרה (Dashboard) של נתוני חיישנים בזמן אמת.
  + מבצע שליפה מ־Firebase Firestore ומציג ערכים עדכניים של חיישני טמפרטורה, אור, לחות קרקע, לחות אוויר ולחץ.
  + כולל חיווי סטטוס (רגיל, אזהרה, קריטי) על בסיס ערכים שנמדדו.
  + כולל רשימת התרעות חכמות בהתאם לערכים שנמדדו.
  + תומך ברענון נתונים ידני ובניהול גישת משתמשים לפי תפקידים.
* 🧱 אובייקטים עיקריים:
  + useState, useEffect – לניהול מצב טעינה, משתמש, חיישנים והתרעות.
  + useNavigate – משמש לניווט אחורה מהדף.
  + db, collection, query, where, getDocs, orderBy, limit – שיטות של Firebase Firestore לשליפת מידע.
  + sensors – אובייקט המרכז נתונים עבור כל חיישן (ערך, יחידה, סטטוס, זמן עדכון).
  + alerts – מערך התרעות מוצגות למשתמש בהתאם לסטטוס החיישנים.
  + userData – נתוני המשתמש המחוברים כולל תפקיד.
  + isRefreshing – משתנה בוליאני לציון האם מתבצעת רענון נתונים.
* ⚙️ שמות הפונקציות ומה כל אחת עושה:
  + useEffect – בעת טעינה ראשונית, טוען נתוני משתמש מ־localStorage ואז טוען נתוני חיישנים בהתאם לתפקידו.
  + determineSensorStatus – מחשבת סטטוס ("normal", "warning", "critical") לפי סוג החיישן והערך הנמדד.
  + fetchSensorData – שואבת את הנתונים האחרונים ממסד הנתונים, מחשבת סטטוס לכל חיישן, יוצרת התרעות רלוונטיות, ומעדכנת את הממשק.
  + formatTimestamp – ממירה חותמת זמן לתצוגת שעה קריאה למשתמש.
  + formatAlertDate – ממירה חותמת זמן לתצוגה אנושית עם Today / Yesterday וכו'.
  + handleBackClick – מחזירה את המשתמש לעמוד הקודם.
  + handleRefresh – מרעננת את הנתונים דרך fetchSensorData.
  + getStatusColor – מחזירה צבע טקסט תואם לסטטוס החיישן.
  + getStatusIcon – מחזירה אייקון חזותי תואם לסטטוס החיישן.

📄 קובץ: SpatialModelDashboard.jsx

🧩 תיאור קצר:

* רכיב React המציג לוח מחוונים (Dashboard) חכם להשקיה, כולל מדדי טמפרטורה ולחות אוויר, המלצה חכמה על השקיה, והדמיית Heatmap אינטראקטיבית באמצעות Plotly.
* מבצע שליפת נתוני חיישנים בזמן אמת מ-Firebase Firestore, מחשב המלצות השקיה, ומציג את הנתונים על גבי מפת צבעים דו-ממדית.
* תומך בבחירת טווח זמן (24 שעות, 7 ימים, 30 ימים), רענון נתונים, והצגת מדדים אחרונים.

🧱 אובייקטים עיקריים:

* useState, useEffect – לניהול מצבים פנימיים (נתוני משתמש, חיישנים, גרף, טווח זמן ועוד).
* useNavigate – לניווט חזרה לעמוד הקודם.
* sensorData, latestReadings, plotData, plotLayout – משתנים לניהול מדדי חיישנים והגרף.
* db, collection, query, orderBy, limit, where, getDocs – פונקציות Firestore לשליפת נתוני משתמש וחיישנים.
* Plot – ספריית Plotly להצגת גרף Heatmap עם שכבת נתוני חיישן וקריאות עדכניות.

⚙️ שמות הפונקציות ומה כל אחת עושה:

* useEffect – מאתחלת את טעינת נתוני המשתמש מ-localStorage והפעלת טעינת נתוני חיישנים אם המשתמש מורשה.
* fetchSensorData – שואבת את נתוני החיישנים מ-Firestore לפי טווח הזמן שנבחר, מחשבת המלצה להשקיה, ומעדכנת את המשתמש עם הקריאה העדכנית ביותר.
* generateHeatmapData – מייצרת מטריצת ציון השקיה (0–10) לפי טמפרטורה ולחות, מוסיפה נקודות קריאה בפועל ומגדירה את מבנה ה-layout לגרף Plotly.
* handleBackClick – מחזירה את המשתמש לדף הקודם.
* handleRefresh – מפעילה רענון נתונים באמצעות fetchSensorData.
* handleTimeRangeChange – משנה את טווח הזמן ומרעננת את הנתונים בהתאם.
* getRecommendationInfo – מחזירה אובייקט הכולל טקסט, צבע ואייקון להמלצת ההשקיה הנוכחית (השקה / לא להשקות / ניטרלי).

📄 קובץ: WeatherPage.jsx

🧩 תיאור קצר:

* רכיב React להצגת תחזית מזג אוויר גלובלית מבוססת על Open-Meteo API. מאפשר חיפוש לפי ערים בעולם, מציג תנאים נוכחיים, תחזית יומית ושעתית, מדדי רוח מתקדמים ומידע ממוקד הכולל Sunrise/Sunset ו-UV. כולל תמיכה בערים נפוצות בישראל (כרמיאל, תל אביב, ירושלים), טעינה אוטומטית חוזרת, והצגת מפה מכל עיר שנבחרה.

🧱 אובייקטים עיקריים:

* useState, useEffect – לניהול מצבי חיפוש, טעינה, שגיאות, נתוני מזג אוויר ועדכון אוטומטי.
* useNavigate – משמש לניווט חזרה מהעמוד.
* cities, weatherData, commonLocations – מערכים ואובייקטים המייצגים ערים נבחרות, נתונים שהתקבלו ומיקומים מוגדרים מראש.
* EnhancedWindDetails – רכיב פנימי להצגת מידע מפורט על רוח, כולל מדדי Beaufort, כיוון רוח והשפעות.
* Open-Meteo API, OpenStreetMap Nominatim API – משמשים לשליפת נתוני מזג אוויר ולקבלת קואורדינטות ממחרוזת חיפוש.

⚙️ שמות הפונקציות ומה כל אחת עושה:

* getWeatherIcon – מחזירה אייקון מתאים לפי קוד מזג אוויר.
* getWeatherCondition – מחזירה תיאור מילולי לפי קוד מזג אוויר.
* formatTime – ממירה מחרוזת תאריך מה-API לשעת תצוגה.
* fetchWeatherDataForCities – שולחת בקשת API ל־Open-Meteo לפי מיקום, ומחשבת תחזית יומית ושעתית עבור כל עיר.
* useEffect – מפעילה טעינה ראשונית של הנתונים וקובעת רענון אוטומטי כל 15 דקות.
* handleBack – מחזירה את המשתמש לעמוד הקודם.
* handleInputChange – מעדכנת את טקסט החיפוש ומנקה שגיאות.
* handleSearch – מבצעת חיפוש עיר לפי OpenStreetMap, שואבת קואורדינטות ומעדכנת את העיר שנבחרה.
* getWindDirection – מחזירה כיוון רוח מילולי (N, NE וכו') לפי מעלות.
* getBeaufortScale – מחשבת עוצמת הרוח לפי קנה מידה Beaufort.
* EnhancedWindDetails – מציג לוח מידע רוחני: מהירות, כיוון, מדדי עוצמה, תחושת רוח, גורמי השפעה אפשריים, ומאפיינים בטיחותיים.

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

מערכת ההשקיה החכמה פותחה כדי לסייע לחקלאים לנהל את ההשקיה בצורה חכמה, מבוססת נתוני אמת.

המערכת משתמשת בחיישנים תת-קרקעיים למדידת לחות וטמפרטורה, בשילוב תחזית מזג אוויר, כדי להמליץ או להפעיל השקיה באופן אוטומטי.

המערכת כוללת ממשק אינטרנטי ידידותי, בו ניתן לראות את מצב השדה, לשלוט על ההשקיה, ולקבל התראות בזמן אמת.

|  |  |  |
| --- | --- | --- |
| מסך | תיאור | פעולות אפשריות |
| מסך התחברות | עמוד הכניסה הראשי. מאפשר כניסת משתמשים עם הרשאות שונות (מנהל/עובד). | הזנת מייל וסיסמה, מעבר ללוח בקרה. |
| לוח בקרה (**Dashboard)** | מציג נתוני חיישנים בזמן אמת: טמפרטורה, לחות באוויר ובקרקע, סטטוס השקיה ותחזית. | צפייה מהירה, קבלת התראות, ניווט לשאר המסכים. |
| **Sensor Analytics** | מציג גרפים ונתונים של כלל החיישנים – בזמן אמת ובעבר. כולל סינון לפי תאריך, אזור וחיישן. | מעקב אחר מגמות, בדיקת חריגות, השוואת נתונים. |
| מסך שליטה בהשקיה | שליטה ידנית על מערכת ההשקיה. ניתן לבחור אזור או צמח ולהפעיל השקיה יזומה. | הפעלה/כיבוי השקיה, תיעוד הפקודה, בדיקת לחות טרום-השקיה. |
| **Heatmap** – מפת חום חכמה | מציג תוצאה של ניתוח PCA ו-Kriging בצורת מפת חום אינטראקטיבית. | זיהוי אזורים עם צורך גבוה בהשקיה, ניתוח מבוסס מרחק וחיישנים. |
| **Plant Health Analyzer** | מאפשר להעלות תמונה של צמח, והמערכת מנתחת את מצבו בעזרת AI ונותנת המלצות. | העלאת תמונה, קבלת דוח בריאות, הצעות לפעולה (השקיה, דישון, הצללה). |
| **Farmer Assistant** (צ'אט חכם) | סוכן שיחה פתוח (מבוסס Gemini) שנותן מענה לחקלאי בשפה חופשית. | שאלות על השקיה, זיהוי בעיות, ייעוץ אגרונומי כללי, הסברים טכניים. |
| מסך משימות (**Task Manager)** | מאפשר למנהל להוסיף משימות לעובדים, שמקבלים אותן כהתראות לביצוע. | יצירת משימה, הקצאה לפי שם משתמש/אזור, מעקב סטטוס, אישור סיום. |
| מסך משתמש | ניהול פרטי חשבון, יציאה מהמערכת, שינוי סיסמה. | עריכת פרטים, מעבר לרשומות משלו, יציאה מהמערכת. |

נא לכלול גם סרטון קצר של הרצת המערכת .

<https://drive.google.com/file/d/14u7yIb2ZD99b_Cc4l8qTmZEe43qwHcxi/view?usp=sharing>

שגיאות נפוצות והסברים

|  |  |  |
| --- | --- | --- |
| שגיאה / מצב | סיבה אפשרית | פתרון מוצע |
| **"Invalid credentials"** (שגיאת התחברות) | הוזנו מייל או סיסמה שגויים. | לבדוק את הנתונים שהוזנו, לנסות שוב או לפנות למנהל לקבלת שחזור. |
| **"No sensor data available"** | אין קריאה זמינה מהחיישנים בזמן אמת – ייתכן ניתוק תקשורת או שאין קליטה מהחיישנים בשטח. | לוודא שהחיישנים פעילים ומחוברים לחשמל, לבדוק חיבור MQTT, ולרענן את הדף. |
| במסך **Sensor Analytics** לא מוצגים נתונים בגרף | המשתמש בחר תאריך שבו לא נאסף מידע בפועל. | יש להזין כתאריך התחלה את 13.4.2025 – זהו הדגימה האחרונה שנשמרה בענן. לא בוצעו דגימות חדשות לאחר תאריך זה, ולכן הגרף יופיע רק החל ממנו. |
| העלאת תמונה ב-**Plant Health Analyzer** נכשלה | ייתכן שפורמט התמונה לא נתמך או שהתמונה לא תקינה. | לבדוק שהקובץ בפורמט .jpg/.png ושגודלו קטן מ-5MB. אם הבעיה נמשכת – לנסות תמונה אחרת. |

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

#### 1. פערי תקשורת בין צוותי התוכנה וההנדסה המכאנית

* האתגר: היינו צוותים מתחומים שונים (תוכנה ומכונות) עם שפה טכנית שונה, מה שיצר קושי בהבנה הדדית של צרכים ויכולות.
* הפתרון: קיימנו מפגשים תכופים – הן בזום והן פיזית – שכללו הסברים מפורטים, תרשימים ודוגמאות מעשיות. בנוסף, נעזרנו בשפה פשוטה וממוקדת תוך תיאום מושגים בסיסיים.

#### 2. חוסר בהירות לגבי מטרות ותכולת הפרויקט בתחילתו

* האתגר: בתחילת הדרך לא הייתה הגדרה מדויקת מה המערכת צריכה לכלול – האם מדובר רק בתצוגה של נתונים, או גם בשליטה אקטיבית על השקיה.
* הפתרון: לאחר פגישה עם אוזי, מרצה בקורס מחשוב ענן ומנחה של הצוות המכאני, התחדדה ההבנה כי יש לשלב גם יכולת שליטה ממשית במערכות ההשקיה דרך האתר – מה שהוביל להרחבת הפונקציונליות.

#### 3. אתגרים טכנולוגיים בתקשורת בין הרכיבים

* האתגר: נדרשנו ליצור תקשורת יציבה בין החיישנים לבין השרת בענן, בתנאי שטח חקלאיים שבהם יתכנו הפרעות או נפילות רשת.
* הפתרון: נעשה שימוש בפרוטוקול **MQTT**, המתאים במיוחד לרשתות דלות רוחב פס. בנוסף, הטמענו מנגנוני גיבוי ושמירה בענן (Firebase) במבנה פשוט ויעיל.

#### 4. מיקום עומק החיישנים בקרקע והשפעתו על דיוק הנתונים

* האתגר: גובה החיישנים משפיע על רמת הלחות הנמדדת – ככל שהחיישן עמוק יותר, מתקבלת קריאה לחה יותר, מה שעלול להוביל לטעויות.
* הפתרון: יישמנו שתי רמות של חיישנים (עמוקים ושטחיים), וחישבנו את ממוצע הקריאות כדי לקבל תמונה מדויקת של מצב הקרקע האמיתי.

#### 5. איסוף צרכים מדויק מהמשתמשת – קרן, בעלת החווה

* האתגר: המידע הראשוני שקיבלנו מקרן היה כללי, ולא כלל פרטים טכניים מספקים לגבי מגבלות המערכת הקיימת (GSI).
* הפתרון: ערכנו פגישה ייעודית עם קרן, שבמהלכה עלו צרכים ברורים: הודעות השקיה מחוץ לאפליקציה, אפשרות לשליטה ידנית, חשיבות לעומק החיישנים, ותיעוד כמויות המים. מידע זה שימש כבסיס לאפיון הפתרון המותאם.

#### 6. קושי בתעדוף ועיצוב הדשבורד והנתונים באתר

* האתגר: עמדנו בפני החלטות אילו נתונים חשובים להצגה, אילו פעולות יהיו זמינות למשתמשים שונים, וכיצד לעצב את החוויה.
* הפתרון: עבדנו בשיתוף פעולה עם הממשק של Firebase, והגדרנו בבירור תפקידים למנהל ולפועל. בנינו דשבורד ברור, כולל התרעות, גרפים, ותיעוד נתונים היסטוריים.

9. בשבוע 13 בקורס תציגו את המערכת שלכם.

ההצגה תכלול הרצת המערכת/סרטון, וכן מצגת המכילה את סעיפים 1-4 (כ-15 דקות) , אשר תוצג בשיעור האחרון, בה יוצגו הממצאים העיקריים.המצגת תאוחסן בתיקיית גיט של הקבוצה. ניתן לעצב אותה כפוסטר.

כל חברי הצוות חייבים להשתתף במצגת, וכן יש להקפיד על מצגת אסתטית ונקייה משגיאות איות.

בונוס:

בשבוע 12 הצגתם סטודיו.

קיבלתם באופן אנונימי את המשובים של חבריכם, וכן את המשוב שלנו.

יש להגיש את הטבלה הבאה , תוך התיחסות למשובים שקיבלתם:

There were no bad or negative feedbacks from the other groups, seems like they love us so much 🙂 . regardlessly we put some of the feedbacks we’ve got .

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| המערכת מחזירה תגובה לפי תמונה של צמח – מרשים. | לא | זוהי נקודת חוזקה של המערכת, ואין צורך בשינוי – להפך, כדאי להדגיש אותה יותר בהדגמה. |
| אהבתי את העיצוב ורמת הבקרה הגבוהה. | לא | המשוב מחזק את החוויה החיובית – חשוב לשמר את רמת הבקרה והנגישות למשתמש. |
| ה-AI Agent היה שימושי. | לא | מדובר בפיצ'ר מתקדם שהוסיף ערך למערכת – כדאי להשאיר ולשפר בהמשך את הבנת ההקשר שלו. |
| לא מצאתי מה לשפר / אין הערות. | לא | חיזוק לכך שהמערכת ברורה ונוחה, אך נמשיך לוודא שהיא נגישה גם למשתמשים חדשים. |
| מצוין, דיוק ועיצוב טוב. | לא | אין הצעה לשיפור – משוב חיובי שמאשר שהשפה החזותית ברורה ונעימה. |

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

הנתונים שלנו באופן עיקרי ממשק התחזית של השירות המטאירולוגי ו Open Meteo ו OpenStreetMap

<https://ims.gov.il/he>

<https://open-meteo.com/>

[https://www.openstreetmap.org/#map=7/31.438/35.074](https://www.openstreetmap.org/%23map=7/31.438/35.074)

Open Street map was used to include the map

השתמשנו ב chatGPT ללא שום AI אחר.

להלן הפרומפטים מעכשיו ות"ב קודמים גם:

* Help me create a maintenance manual for a smart irrigation project, including key file descriptions, functions, and data flow.
* I have sensor data and want to create a heatmap using PCA and Kriging. Explain what to do and how to analyze it.
* Give me a clear explanation for each function in my JavaScript/React code so I can add it to the developer's manual.
* Give me a clear explanation for each function in my JavaScript/React code so I can add it to the developer's manual.
* Explain how to calculate the SUS (System Usability Scale) score based on 10 user responses. Show the calculation step by step, and explain what the final score means in terms of usability. Provide example interpretation for scores like 72, 85, and 94.

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!