

**Software Engineering Department**

**Braude College**

**Capstone Project Phase B**

**Smart Irrigation System**

**Smart Farm**

By:

Abed al kareem zeedan

Rami taha

**Advisor:**

**Dr. Naomi Unkelos Shpigel**

**Project Code:**

25-1-D-12

**Link to GitHub:**

[**https://github.com/SharkZeedan/Smart-Farm**](https://github.com/SharkZeedan/Smart-Farm)

**Table of Contents**

**1. *introduction*** *3*

**2. Literature Review** 3

**3. System Requirements**  4

3.1 FR Requirements 4

3.2 NFR Requirements 5

**4. System Software Architecture** [6](#4bk4up2ybsvc)

4.1. System Hardware Architecture [7](#n3s51df92vk)

**5. Features Implemented**  [8](#h9dsz49acov)

**6. Testing & Evaluation** [9](#2563yecbr4l)

6.1Testing Results 9

6.2 Insights & Validation from user’s feedbacks 9

6.3 How users rated the smart features 9

**7.Challenges & Solutions** 10

**8.User Manual** 10

**9.*Screens*** [12](#jw15amnulrwz)

**10.Maintenance Manual** 27

**11.References**  [34](#j1258uewzmjh)

***Abstract***

The Smart Irrigation System project presents a fully functional, web-based platform that enables farm managers to monitor environmental data and control irrigation remotely, based on real-time sensor readings and weather forecasts.

Unlike traditional irrigation practices, our system integrates underground soil moisture and temperature sensors deployed by a robotic mechanism, providing precise insights into soil conditions. These insights are transmitted via MQTT to a Firebase-backed cloud infrastructure, processed, and visualized through a responsive dashboard built with React. The system features customizable irrigation thresholds, alert notifications, manual override, and predictive analytics based on weather APIs.

The solution was tested in a real farm environment, offering reliable performance and scalable architecture. This work demonstrates how modern IIoT technologies can enable sustainable water usage and improve agricultural efficiency through smart automation.

***1.Introduction***

The escalating global water crisis—driven by climate change and unsustainable water management—poses a serious threat to agriculture [[11](#hcr4as3p361o)], which accounts for nearly 70% of global freshwater withdrawals. As droughts intensify and water scarcity increases, crop failures and food insecurity become more widespread. According to the UN [[12](https://www.un.org/en/)], by 2025 over 1.8 billion people will live in regions severely affected by water shortages. In response, global organizations such as the FAO [[13](#z2dzgall3uaj)] emphasize the adoption of water-efficient technologies and sustainable farming practices to enhance agricultural resilience.

One promising solution is the implementation of smart irrigation systems that adaptively manage water use based on real-time environmental data. These systems integrate underground IoT-based sensors to monitor critical parameters such as soil moisture, temperature, and weather conditions. By automating irrigation processes and enabling remote control via mobile applications, they ensure water is delivered precisely when and where it is needed—dramatically reducing waste and improving crop health [[15](#hquvcfqlwxqn)].

This project aims to design and implement a scalable, intelligent irrigation solution tailored to the needs of modern agriculture. The system combines Industrial Internet of Things (IIoT) , Industry 4.0 principles, and mechatronic design to enable real-time monitoring, adaptive decision-making, and accurate water delivery [[14](#v8fag1mgtmkv)] . Data is collected via robotic platforms equipped with humidity, temperature, light, and distance sensors, and uploaded to Firebase for secure storage. A predictive weather API enhances the system's ability to forecast irrigation needs, allowing for proactive water management.

In addition to automated logic, the platform includes a user-friendly website that enables manual intervention when alerts are triggered. The system is initially focused on optimizing irrigation for a single plant type but is built to scale. By addressing inefficiencies in traditional irrigation methods and replacing manual monitoring with automated intelligence, our solution aims to promote water conservation, improve agricultural productivity, and help secure the future of farming in a resource-constrained world.

***2******.Literature Review***

Smart irrigation systems are evolving to address the increasing demand for efficient water use in agriculture. Several commercial and research-based solutions demonstrate the integration of sensors, wireless communication, and automation. However, many lack a modular design that allows integration of custom sensors, predictive forecasting, and real-time manual control.

**Table 1**.Comparison of some of the features of the most relevant smart irrigation systems and the proposed solution.

| Reference | System Type | Technologies | Sensors & Actuators. |
| --- | --- | --- | --- |
| CropX [[19](https://docs.google.com/document/d/18HyPq1hXxdUoI7MgaYcWQu9LwaCnlm39JzCmP6hw5qE/edit?tab=t.0#bookmark=id.29sy1gscnn62)] | Smart Agriculture platform. | Cellular,Wifi. | Soil & Moisture Sensors. |
| Netafim [[17](https://docs.google.com/document/d/18HyPq1hXxdUoI7MgaYcWQu9LwaCnlm39JzCmP6hw5qE/edit?tab=t.0#bookmark=id.xy5ksa2yfb8u)] | Digital Farming Platform. | Cellular Networks. | Flow Measurement Systems. |
| Valley Irrigation [[18](https://docs.google.com/document/d/18HyPq1hXxdUoI7MgaYcWQu9LwaCnlm39JzCmP6hw5qE/edit?tab=t.0#bookmark=id.p35p1qidk428)] | Precision Irrigation Management. | Wifi,4G/LTE. | Weather & Pressure monitoring Systems. |
| Proposed Solution | Robotic-enabled. Smart Irrigation & Website. | MQTT,Cloud,Wifi. | Humidity & temperature sensors, auto valves, robotic deployment. |

While systems like Netafim and Valley 4.0 offer robust data collection, they often rely on proprietary hardware and closed infrastructure. In contrast, our project presents a modular, open-source solution that combines custom sensor deployment via robotics, real-time integration of weather forecasts with soil data, and a role-based web dashboard featuring alert notifications and manual overrides. Backed by Firebase for authentication and real-time updates, our system addresses key limitations in sensor flexibility, personalized irrigation, and practical field deployment.

***3******.System Requirements***

**3.1.*FR*** *Requirements****:***

The system we developed is based on a comprehensive set of functional requirements designed to enable the implementation of a smart, precise, and efficient irrigation solution. These requirements form the foundation for both the hardware and software components of the project, derived from system analysis, domain research, and insights from real-world agricultural practices. They include features such as real-time sensor monitoring, integration with weather APIs, AI-driven alerts, role-based user access, and a responsive dashboard interface. Together, these capabilities support informed irrigation decisions, water conservation, improved plant health, and an intuitive user experience, even for non-technical users.

With Firebase ensuring secure and scalable data management, and remote control features reducing the need for on-site presence, the system offers flexibility and ease of use. By tailoring irrigation to specific plant needs and supporting modular expansion, it lays the foundation for a sustainable and data-driven approach to modern agriculture.

Table 2 presents the system’s functional requirements along with their respective descriptions, outlining the key capabilities necessary for implementing an innovative and intelligent irrigation platform.

**Table 2. System functional requirements**

| # | FR Requirement | Description |
| --- | --- | --- |
| 1 | Real-Time sensor Data | Collect and display soil humidity and temperatures from field sensors. |
| 2 | WeatherForecast Integration | Integrate weather API to adjust irrigation predictions. |
| 3 | AI Based Weather forecast integration | Use external Weather API to enhance irrigation prediction through forecast-aware logic |
| 4 | Smart Alert System | Use AI informed thresholds to send alerts when irrigation is needed |
| 5 | Role-based Access | Support separate interfaces and permissions for both manager and worker |
| 6 | Dashboard Interface | Provide a responsive web interface for real-time monitoring |
| 7 | Plant Health Alert | Use AI to determine the health status of the plant and advice to maintain health |

**3.2. *NFR*** *Requirements****:***

The smart irrigation system was developed in alignment with a set of non-functional requirements (NFRs) that ensure high system quality, continuous uptime, rapid performance, and long-term scalability. These requirements address overall system behavior rather than specific features, focusing on stability, reliability, maintainability, and readiness for future AI-powered extensions. The system was built to operate reliably under field conditions, maintain consistent cloud communication, and deliver real-time dashboard updates. Its modular software architecture allows for seamless future enhancements and adaptability to evolving user and environmental needs.

In addition, emphasis was placed on security and data integrity to safeguard sensitive agricultural data and ensure trust in the platform’s automated decisions. Performance optimizations were incorporated to ensure smooth user experiences, even in low-bandwidth rural areas. Collectively, these non-functional considerations contribute to a robust, future-proof solution capable of supporting both small-scale and industrial farming operations.

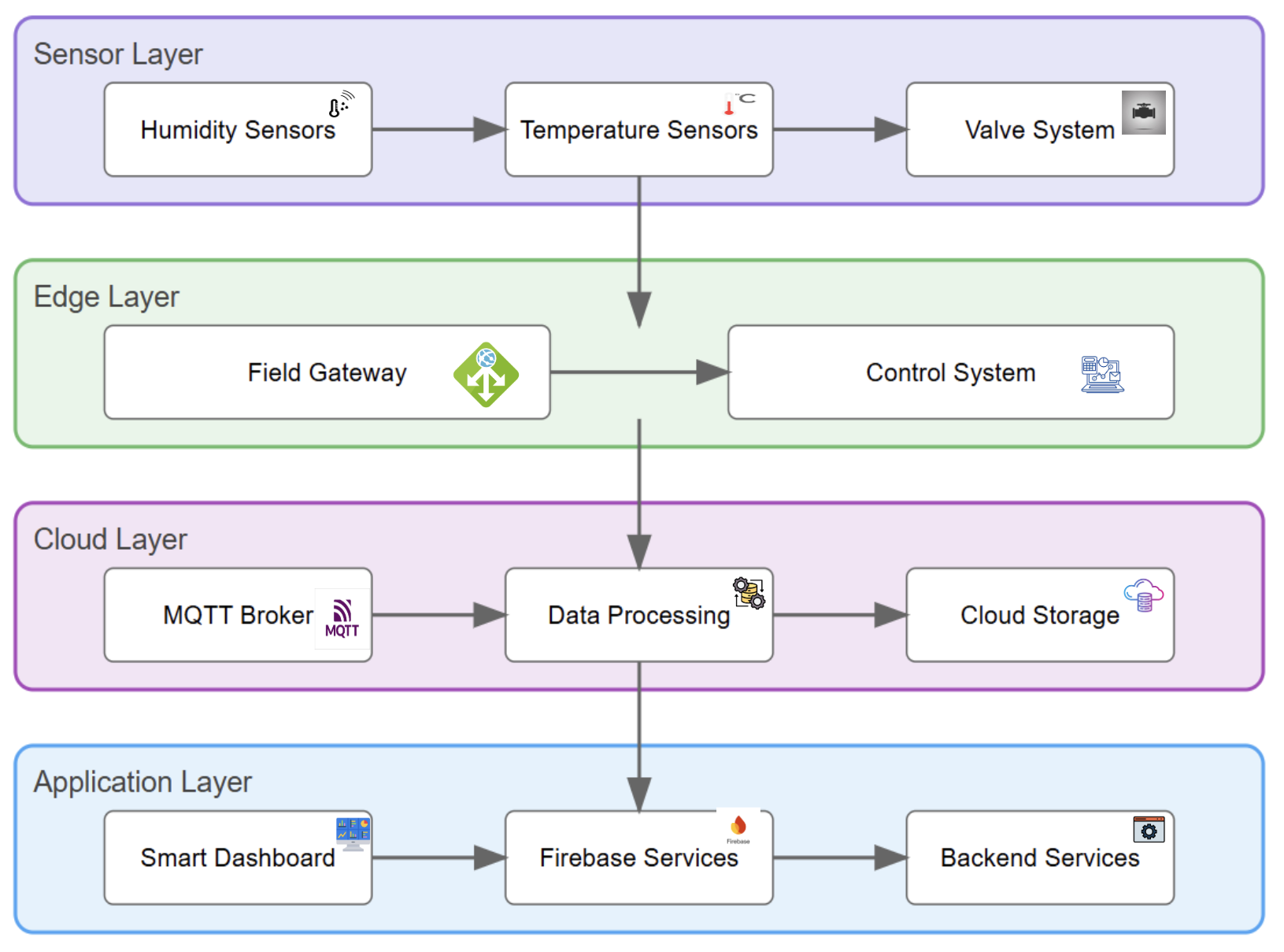
**Table 3 su**mmarizes the system’s non-functional requirements and their impact on performance, reliability, and overall robustness.

**Table 3. System non-functional requirements**

| # | FR Requirement | Description |
| --- | --- | --- |
| 1 | Scalability | Design to support more sensors ,users and additional AI modules. |
| 2 | Reliability | System operates with minimal downtime and consistent sensor-cloud connection |
| 3 | Performance | Interface updates in real-time with response time <2ms. |
| 4 | Maintainability | Modular architecture enables easy integration of future ai modules |
| 5 | AI Integration Readiness | Backend structure supports future AI modules for predictive analytics. |

***4******.System software Architecture***

The following diagram illustrates the software architecture of the smart irrigation system, organized into four main layers: the Sensor Layer, Edge Layer, Cloud Layer, and Application Layer. Each layer is responsible for a different stage in the data processing flow – from collecting environmental data in the field to presenting insights through a smart management interface. This modular structure ensures smooth data transmission, informed decision-making, and efficient system operation.



**Figure 1** illustrates the overall architecture of the system, outlining the interaction between its hardware components, cloud infrastructure, and user interfaces.

***4******.1.System hardware Architecture***



**Figure 2** shows the robotic platform developed as part of the smart irrigation system. The robot was built in collaboration with a team of mechanical engineers who worked alongside us to design a structure tailored for precise integration of environmental sensors.  
 The sensors are mounted directly on the robot, allowing it to collect field data while in motion. The robot is controlled manually via an external joystick, and one of the controllers shown in the diagram is responsible for translating joystick commands into motor movement.

The key components labeled in the image include:

* Light Sensor – Measures ambient light intensity in the field.
* Temperature Sensor – Measures soil and air temperature for environmental monitoring and decision-making.
* Distance Sensor – Measures the distance in front of the robot and supports spatial awareness and accurate sensor placement.
* Humidity Sensor – Detects soil moisture levels, essential for determining irrigation needs.
* Electric Handle – A motorized mechanical arm that positions the sensors into the ground.
* Sensor Controller – Collects data from all sensors and transmits it to the cloud system.
* Joystick Motion Controller – Controls the robot’s movement based on input from the external joystick.

This integrated, modular setup enables efficient, accurate data collection in real agricultural environments, providing flexibility and control during field operation.

### ***5******.Features Implemented***

The system integrates smart agricultural technology with cloud computing and real-time control, enabling precise, efficient irrigation management. The following features were fully implemented and tested on a live farm environment, demonstrating the system’s robustness, usability, and impact.

* **Real-Time Sensor Monitoring** Live soil humidity and temperature data is displayed instantly on the dashboard via Firebase sync.
* **Robotic Sensor Deployment** Sensors are physically placed and adjusted in the field using a mobile robotic platform, ensuring accurate placement and flexible scalability.
* **Weather Forecast Integration (AI-Enhanced)** External forecast data is fetched and cross-referenced with real-time soil data to support **predictive irrigation planning**.
* **Smart Alert System** Alerts are triggered only when environmental thresholds are crossed *and* no rain is expected — leveraging AI logic for **intelligent decision-making**.
* **Manual Irrigation Control** Farm managers can activate or stop irrigation with a single click via the web interface, offering override capability even in AI-based decisions.
* **Role-Based Access** Two interfaces exist: one for farm managers (with full control and analytics) and one for workers (limited control and data visibility).
* **Secure Authentication** Firebase authentication ensures only authorized users can access system controls and sensitive data.
* **Live Dashboard Interface** Built with React and Tailwind CSS, the UI displays sensor data, weather forecasts, irrigation history, and alert logs in real-time.
* **Modular Design for Future AI Additions** The backend is structured to easily incorporate additional AI modules (e.g., water consumption prediction, anomaly detection).

### ***6******.Testing & Evaluation***

### The system was tested for usability, reliability, and performance through structured test cases and real-world interactions with users, including the farm owner and student testers. The evaluation focused on verifying real-time data synchronization between sensors and the dashboard, the accuracy of smart alerts that rely on forecast and threshold fusion, and the correct enforcement of user roles and permissions. Additional testing confirmed that manual override of irrigation functions worked as intended and that the system maintained stable operation during extended live use in a farm environment.

### ***6******.1.Testing Results***

Three users completed the SUS questionnaire, with a particular focus on the AI-related aspects of the system. The AI-specific average score was **90/100**, indicating excellent usability and clarity in how the AI-supported decisions were delivered. Participants reported that the system’s recommendations were transparent, context-aware, and helpful for effective decision-making. Overall, the feedback reflected a high level of trust in the system’s ability to combine sensor data and forecast information to support smart irrigation. The consistent real-time performance, intuitive interface, and meaningful alert logic contributed to positive user experiences across both technical and non-technical participants.

***6******.2.Insights & Validation from user’s feedbacks***

**🌱 *"****The system’s decisions felt transparent and easy to understand."*

📊 *"I trusted the AI recommendations — they made sense based on the conditions."*

⚙️ *"Everything worked smoothly — I didn’t need guidance to use the interface."*

☁️ *"Combining sensor readings with weather forecasts really helped reduce doubt."*

🔔 *"The alerts came at the right time and were easy to act on."*

🧠 *"I didn’t feel overwhelmed — the AI made it easier to decide, not harder."*

🌍 *"This could actually help real farms — not just a student demo."*

📱 *"The system worked well on my phone, not just desktop."*

💡 *"Even with no prior experience in agri-tech, I felt confident using it."*

***6******.3.How Users Rated the Smart Features***

User ratings for the system’s smart features are summarized in the chart below see Figure 3. The figure illustrates how participants evaluated key aspects such as the transparency of AI decisions, ease of use, support for decision-making, and accessibility via mobile devices. These results provide valuable insights into user satisfaction and help identify both the strengths of the system and areas for further improvement.

Based on these findings, future plans include more precise tuning of irrigation thresholds using larger datasets, improving the placement and depth of sensors for more accurate measurements, and integrating a user feedback mechanism to dynamically adjust irrigation recommendations. Additionally, expanding mobile support and enabling offline functionality are being considered to increase accessibility and efficiency in real-world field conditions.Figure 3 shows the average satisfaction ratings for key smart features of the AI-driven irrigation system, based on user feedback collected during testing

****

**Figure 3 .**Average satisfaction ratings for key smart features

Based on the feedback summarized in Figure 3, users responded positively to the system's smart features, particularly highlighting ease of use (4.9), transparent AI decisions (4.8), decision-making support (4.8), and confidence for new users (4.7). Trust in AI recommendations and timely alerts also received strong ratings, suggesting that the system effectively supports intuitive, informed, and reliable irrigation decisions. The integration of forecast data with real-time sensor readings was also well-received, reinforcing the platform’s perceived intelligence and responsiveness.

However, real-world applicability scored slightly lower (4.5), indicating that while users found the system useful in theory, some practical deployment challenges may still exist—such as sensor calibration or environmental variability. Minor concerns were also raised regarding mobile accessibility, which although rated 4.6, suggests room for further optimization.

To address these points and improve the system further, future development will focus on refining sensor placement and depth for more accurate data capture, and enhancing the AI model with larger, more diverse datasets to fine-tune irrigation thresholds. Plans also include introducing an in-app feedback loop to allow dynamic user input, and expanding mobile features with offline support to improve usability in rural areas with limited connectivity.

**7****.*Challenges & Solutions***

During the development of our smart irrigation system, we encountered several significant challenges, both technical and collaborative. One of the primary difficulties stemmed from our lack of mechanical engineering expertise, which limited our ability to understand and troubleshoot the robotic unit intended for autonomous sensor deployment. Despite multiple efforts to activate and test the robot, it remained non-functional, forcing us to simulate sensor data manually and proceed with software-side validation. This limitation, along with communication gaps between our team and the mechanical team, created integration delays and mismatched expectations that were eventually resolved through consistent meetings and shared diagrams.

On the software side, designing an AI-driven alert system that accurately combined real-time sensor data with weather forecasts required trial and error to prevent false triggers. We addressed this by implementing a rule-based logic that factored in both humidity thresholds and rain prediction probabilities. Selecting a reliable weather forecast source was another challenge — we tested both the official Meteorological Service API and Google’s Gemini AI-powered forecast tool, comparing accuracy and data consistency before finalizing our integration. Additionally, Firebase’s real-time data flow occasionally caused synchronization issues, which we mitigated by optimizing listeners and adding fallback logic. Sensor placement depth also posed problems, as deep readings often misrepresented actual root-zone dryness — a challenge we overcame by averaging data from both shallow and deep sensors. Other usability concerns included role confusion between workers and managers, and dashboard complexity for non-technical users. These were addressed by refining permission logic, simplifying the UI, and restructuring the dashboard for clarity.

***8******.User Manual***

This section provides guidance for farm managers and workers on how to interact with the Smart Irrigation System through the web interface.

The smart irrigation system was developed to help farmers manage irrigation intelligently, based on real-time data.

The system uses underground sensors to measure soil moisture and temperature, combined with weather forecasts, to recommend or automatically trigger irrigation.

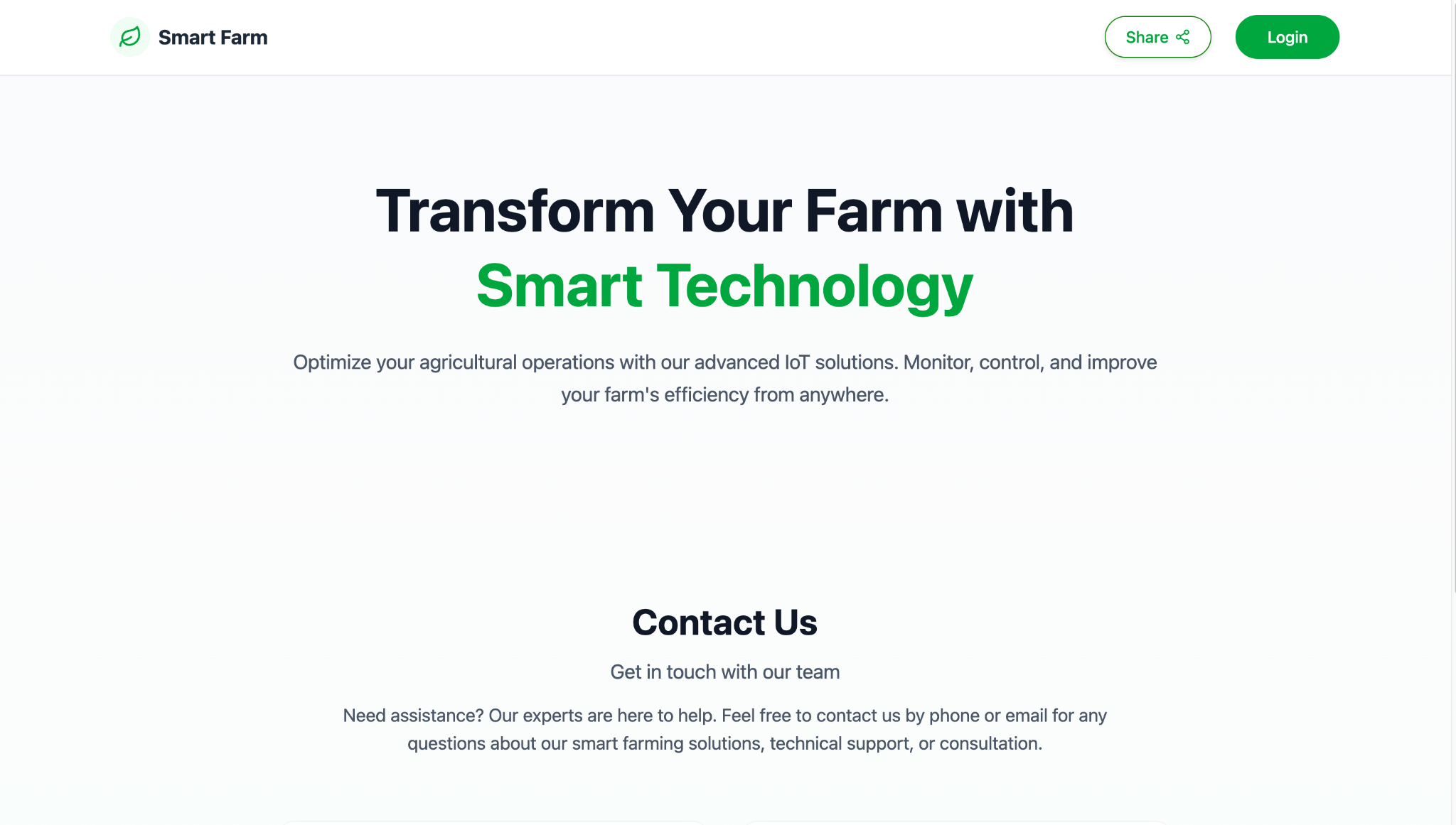
It includes a user-friendly web interface where users can monitor field conditions, control irrigation, and receive real-time alerts.

**Table 4** presents the main system screens, detailing their purpose and the user actions available on each interface

| Possible Actions | Description | Screen |
| --- | --- | --- |
| Enter email and password, then proceed to the dashboard. | Main login mode. Allows users to log in with different permissions (Manager/Worker) | Login screen |
| Quick view, receive alerts, and navigate to other screens. | Displays real-time sensor data: temperature, air and soil humidity, irrigation status, and weather forecast. | Dashboard |
| Track trends, detect anomalies, and compare data. | Displays graphs and data from all sensors — both real-time and historical. Includes filtering by date, area, and sensor. | Sensor Analytics |
| Identifies areas with high irrigation demand using distance-based and sensor-based analysis | Displays the results of PCA and Kriging analysis as an interactive heat map | Heatmap |
| Upload an image, receive a health report, and get actionable suggestions (irrigation, fertilization, shading) | Allows uploading a plant image, and the system analyzes its condition using AI and provides recommendations. | Plant Health Analyzer |
| Questions about irrigation, problem identification, general agronomic advice, and technical explanations. | An open chat agent (Gemini-based) that responds to the farmer in natural language. | Farmer Assistant AI |
| Edit details, access personal records, and log out. | Manage account details, log out, and change password. | User screen |

***9.Screens***

### 1. Home Page

****

Purpose:  
 This is the welcome screen of the Smart Farm system. It introduces users to the platform’s goals and capabilities.

What the user sees:

* A clear headline inviting users to optimize their farm using smart IoT technology.
* A short description of the platform’s benefits: monitoring, control, and efficiency from anywhere.
* A "Contact Us" section for users who may need support.
* A Login button (top-right corner) to access the system.
* A Share button for sharing the platform externally.

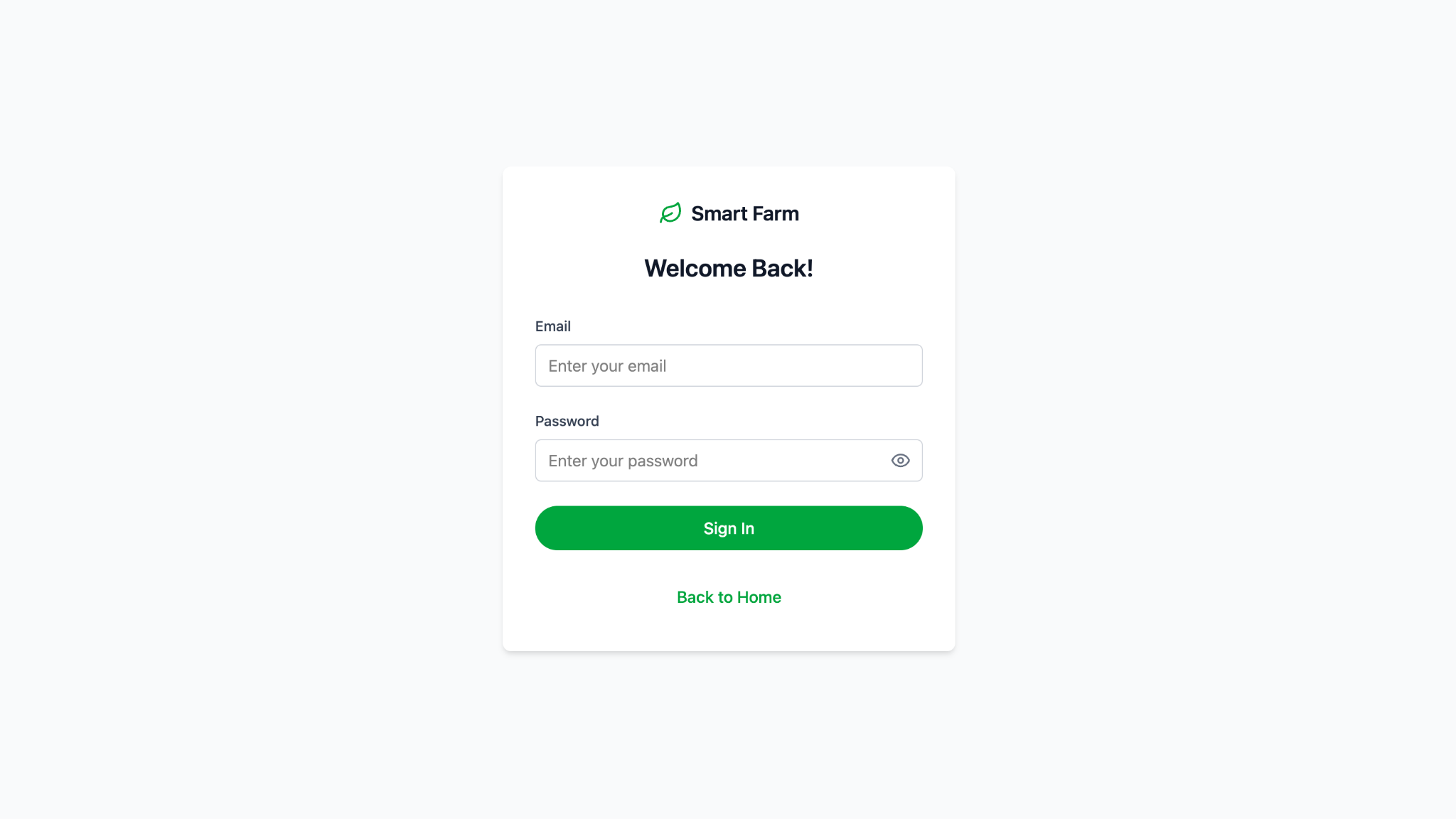
What the user is expected to do:

* Click Login to access the actual system functionality (based on their role).
* Alternatively, they can read more about the system or contact the team.

Navigation flow:  
 Clicking the Login button directs the user to the authentication screen.  
 This is the entry point to all other pages in the system.

Error states:  
 None appear on this screen.

2. Login Screen

****

Purpose:  
 Allows existing users (Manager/Worker) to securely access the system using their personal login credentials.

What appears on the screen:

* Email input field
* Password input field (with eye icon to toggle visibility)
* Sign In button to submit credentials
* Back to Home link to return to the main landing page

What the user is expected to do:

* Enter a valid email and password
* Click Sign In to log into the system

System behavior after login:

* If credentials are valid → the user is redirected to their dashboard (Manager or Worker view)
* If credentials are incorrect → an error message appears, e.g.:  
   *"Email or password is incorrect. Please try again."*

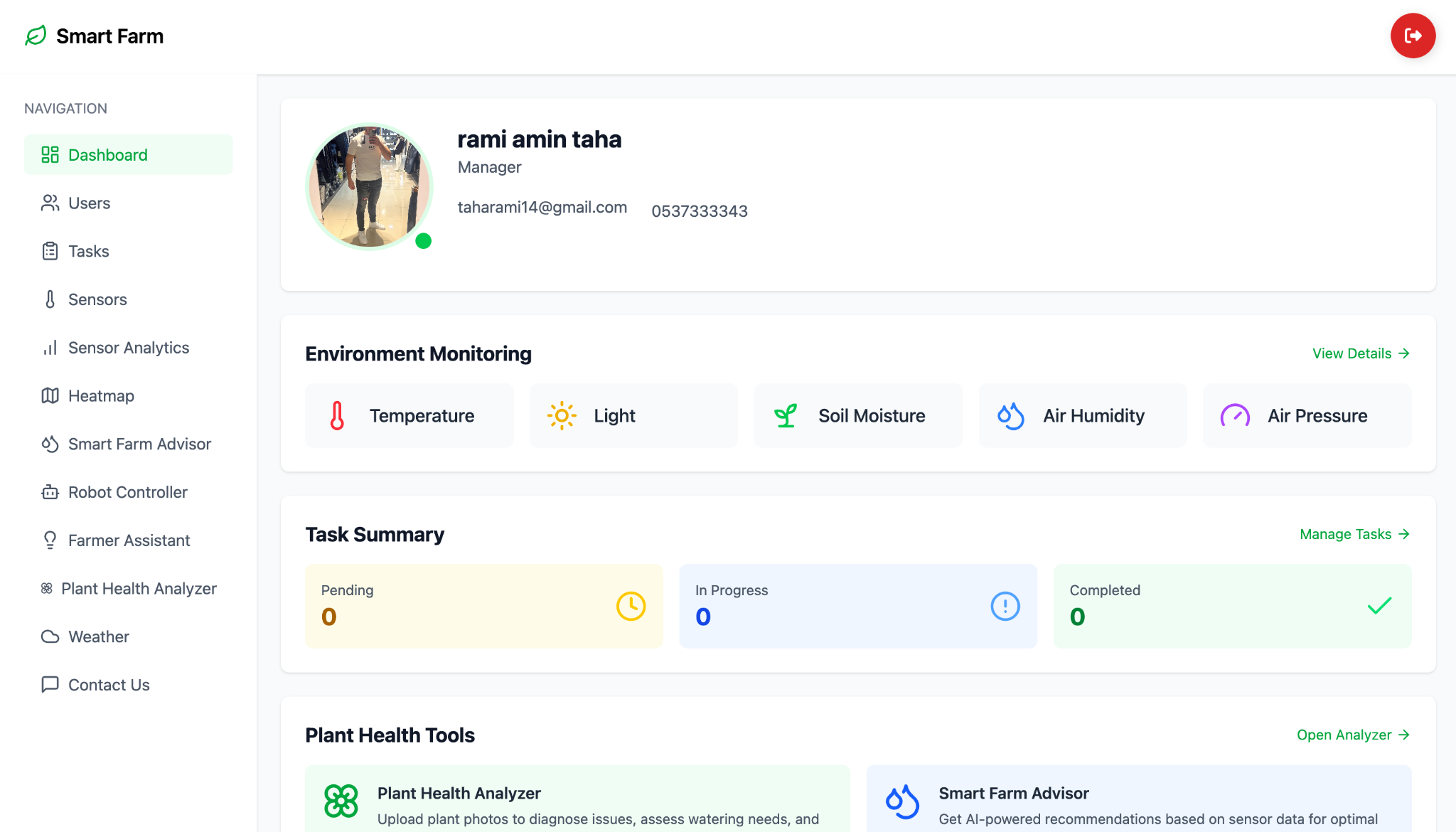
Navigation options:

* Redirect to the dashboard upon successful login
* Return to the home page via the link at the bottom

Common errors:

* Empty fields → inline messages such as *"Email is required"*, *"Password is required"*
* Invalid credentials → general error message shown under the input fields

3. Dashboard – Manager Screen



Purpose:  
 The dashboard provides the manager with an overview of all system activities: environmental monitoring, task status, AI-based tools, and quick access to all required modules.

What is displayed on the screen:

* Logged-in user details (Name, Role, Email, Phone number)
* Environment Monitoring – Live display of sensor data:  
  + Temperature
  + Light
  + Soil Moisture
  + Air Humidity
  + Air Pressure
* View Details link for in-depth monitoring of environmental data
* Task Summary – Task status overview:  
  + Pending
  + In Progress
  + Completed
* Manage Tasks link to access the task management page
* Plant Health Tools – Tools for analyzing plant condition:  
  + Plant Health Analyzer – Upload plant images for AI-based health analysis
  + Smart Farm Advisor – Get irrigation recommendations based on sensor data
* Sidebar navigation to all system modules: Sensors, Analytics, Heatmap, Robot, Weather, Smart Advisor, Farmer Assistant, Contact, and more

What the user is expected to do:

* Monitor key real-time metrics
* Access and manage assigned tasks
* Use smart tools for plant analysis or irrigation guidance
* Navigate to any system module as needed

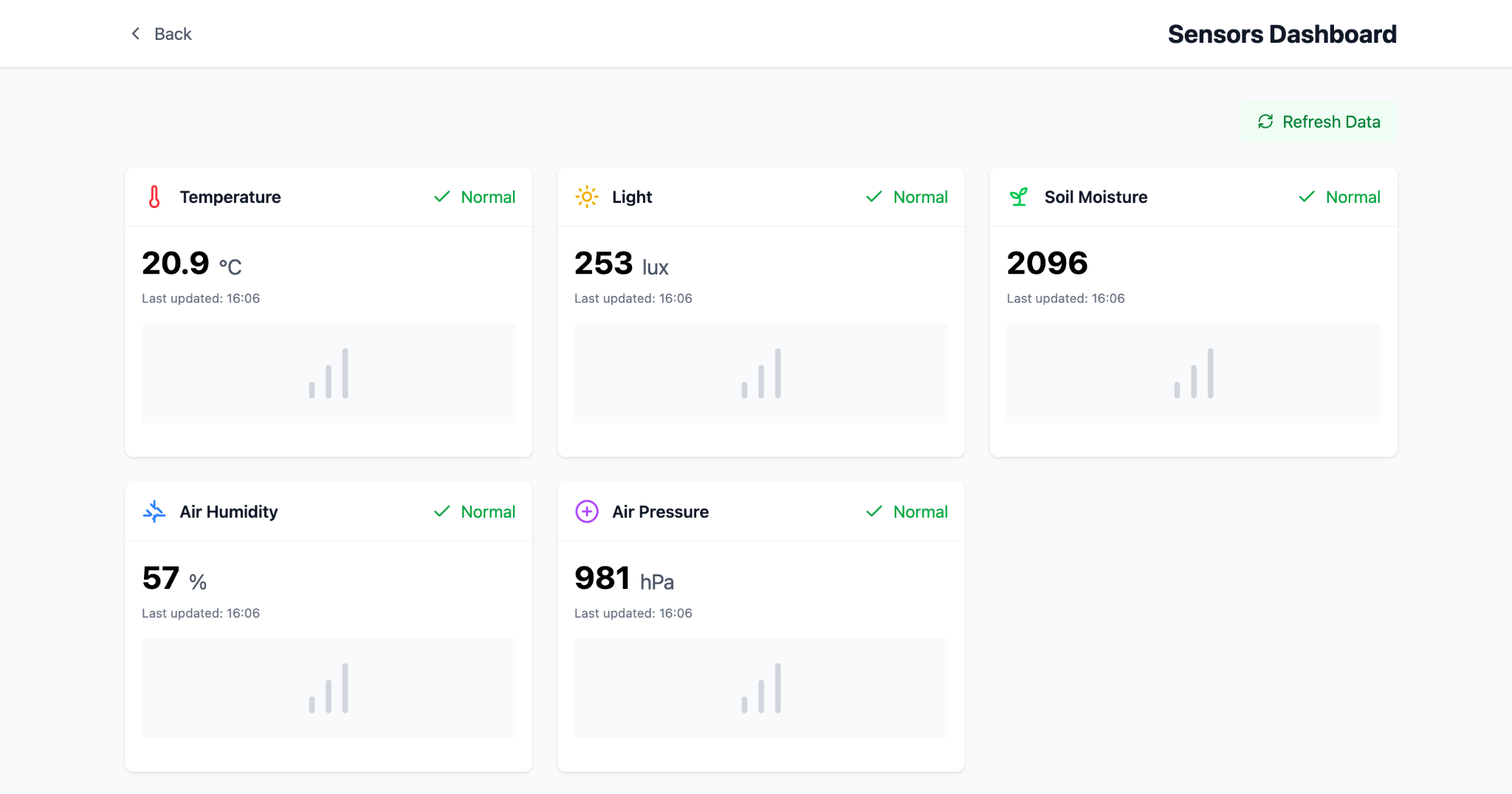
Behavior:

* Each card leads to a dedicated screen (e.g., View Details opens an expanded chart view)
* Data is updated in real time

Possible Errors:

* Sensors not updated → “Offline” status or missing values may appear

4. Sensors Dashboard

****

Purpose:  
 To allow users to monitor real-time environmental conditions through live readings from field-deployed smart sensors, with an option to manually refresh data.

Screen Elements:

* "Back" button to return to the main dashboard
* Title: Sensors Dashboard
* "Refresh Data" button to update the sensor readings
* Display of the five key environmental sensors:  
  + Temperature
  + Light
  + Soil Moisture
  + Air Humidity
  + Air Pressure

Each sensor card includes:

* Current value (with appropriate units: °C, lux, %, hPa)
* Health status indicator (e.g., "Normal" in green)
* Last updated timestamp
* Mini bar chart showing the reading trend

User Actions:

* Review the environmental metrics to assess current field conditions
* Click "Refresh Data" to manually update the readings
* Check for anomalies or errors in the values and sensor health status

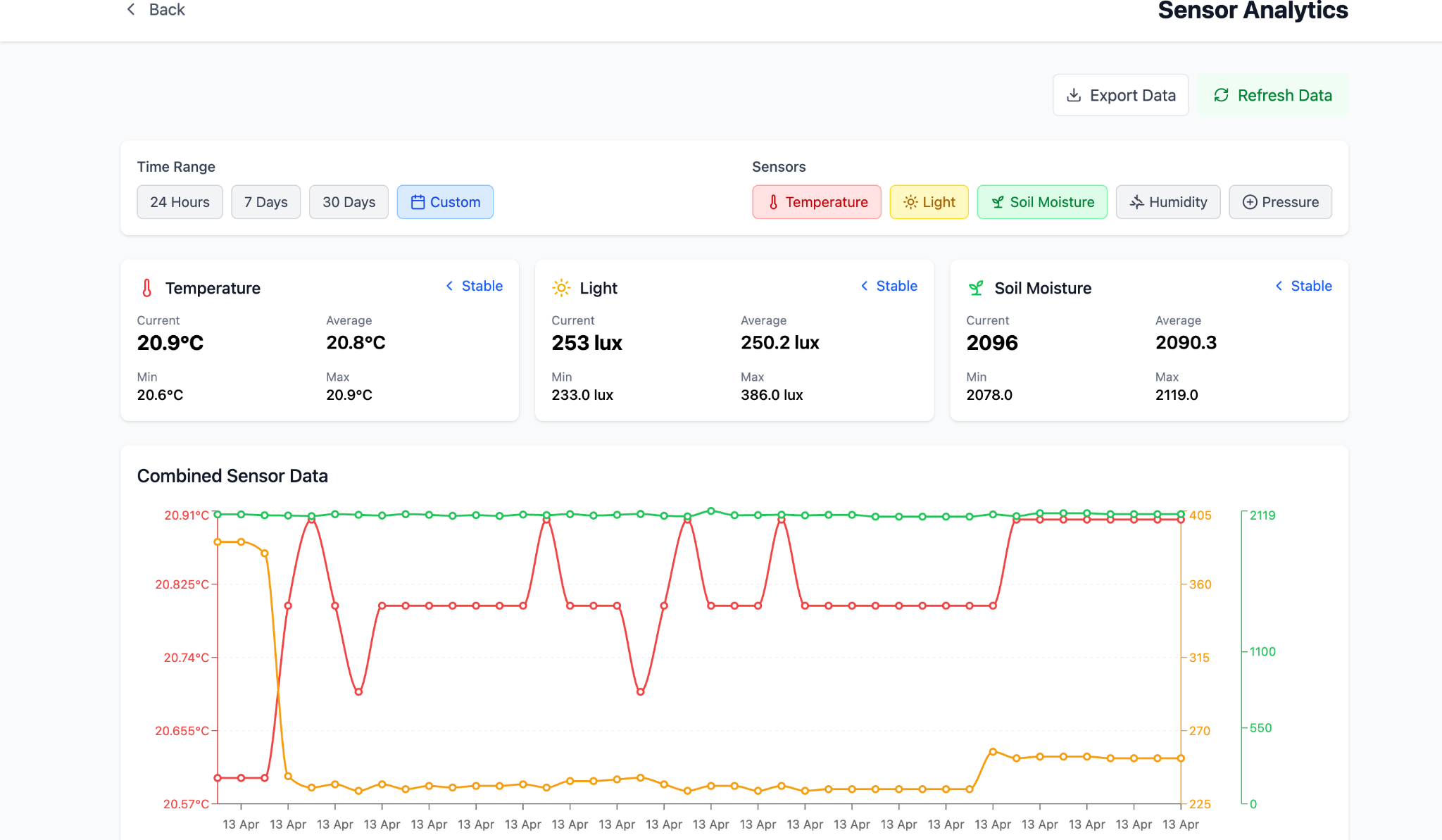
Behavior:

* Each sensor card automatically updates after refresh
* “Normal” status is shown in green if the value is within acceptable range

Possible Errors:

* Missing or abnormal values → chart not shown or warning displayed
* Disconnected sensor → health status changes from “Normal” to “Error” or warning alert appears

5. Sensor Analytics

****

Purpose:  
 This screen enables users to analyze sensor trends over time, helping to detect anomalies, assess field conditions, and make data-driven irrigation decisions.

Screen Elements:

* Back button to return to the previous view
* Title: Sensor Analytics
* Export Data button to download the sensor records
* Refresh Data button to update the graphs
* Time Range Selector:  
  + 24 Hours
  + 7 Days
  + 30 Days
  + Custom range
* Sensor Filters:  
  + Temperature
  + Light
  + Soil Moisture
  + Humidity
  + Pressure
* Current, Average, Min, and Max Values displayed for each selected sensor:  
  + Temperature (in °C)
  + Light (in lux)
  + Soil Moisture (unitless or based on system design)
* Combined Sensor Data Graph displaying synchronized data trends for all selected sensors over the chosen time range

User Actions:

* Select a time range to view historical trends
* Toggle between sensors to focus the analysis
* Export sensor data for offline use or further processing
* Refresh data to update values and graphs

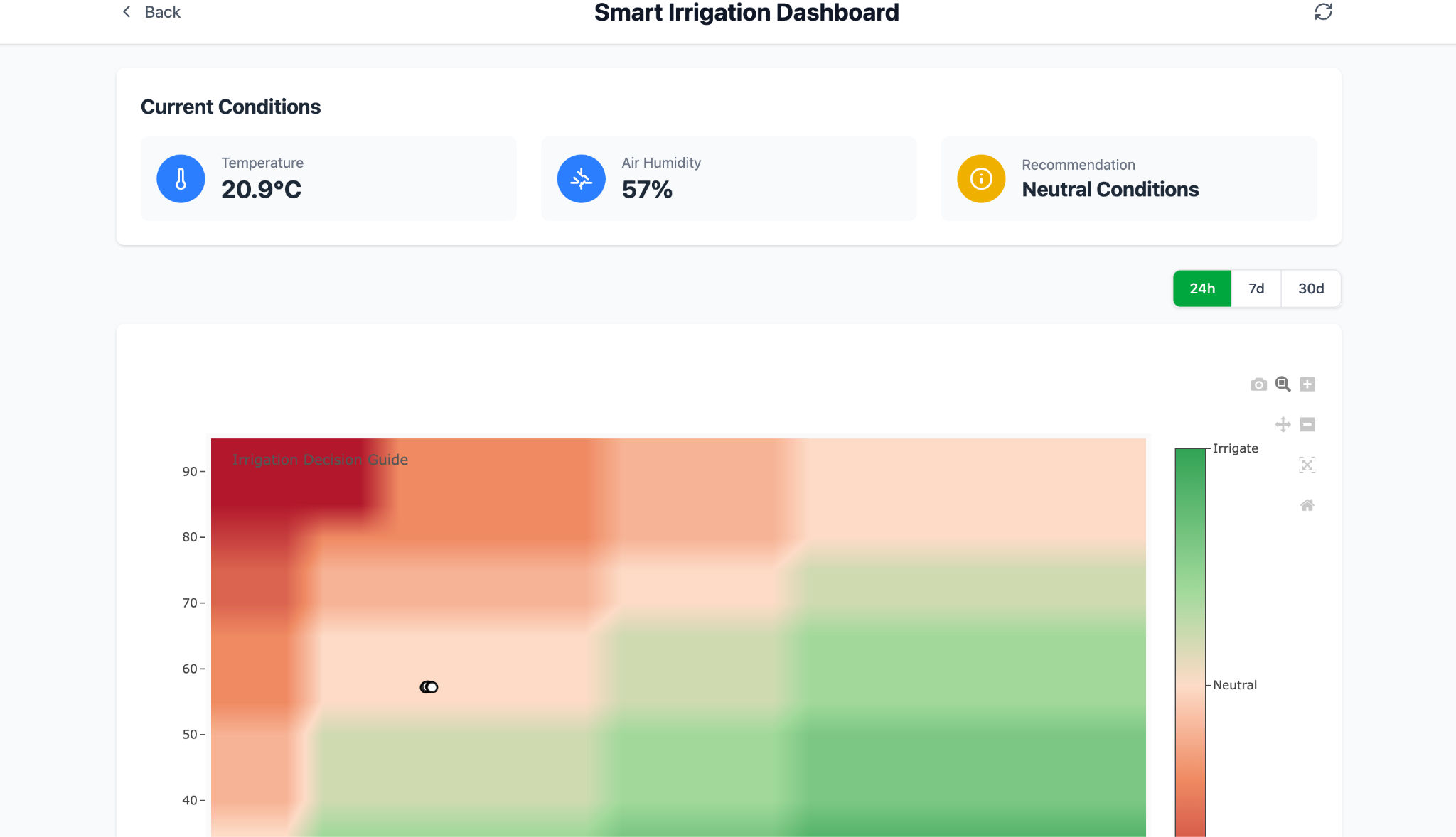
Behavior:

* Graphs update dynamically based on selected filters and time
* Tooltip appears on hover to show detailed readings per date
* Status indication such as "Stable" appears if values are within expected thresholds

Possible Errors:

* If data is missing or sensors are offline, the graph will show gaps or error messages

6. Smart Irrigation Dashboard

****

Purpose:  
 This screen provides real-time irrigation recommendations based on current environmental conditions. It also displays a heatmap that helps guide decision-making regarding irrigation needs.

Screen Components:

* Top Title: Smart Irrigation Dashboard
* Back button to return to the previous screen
* Current Conditions display:  
  + Temperature
  + Air Humidity
  + Recommendation: Neutral Conditions
* Time Range Selection:  
  + 24 hours
  + 7 days
  + 30 days
* Heatmap with Irrigation Decision Guide:  
  + Vertical axis: Environmental indicators (likely humidity/temperature)
  + Color gradient from red (no irrigation needed) to green (irrigation required)
  + A marker indicates the current soil condition

User Instructions:

* Monitor real-time environmental data
* Follow recommendations to decide whether irrigation is needed
* Switch between time ranges to observe trends
* Use the heatmap to understand environmental and agronomic context

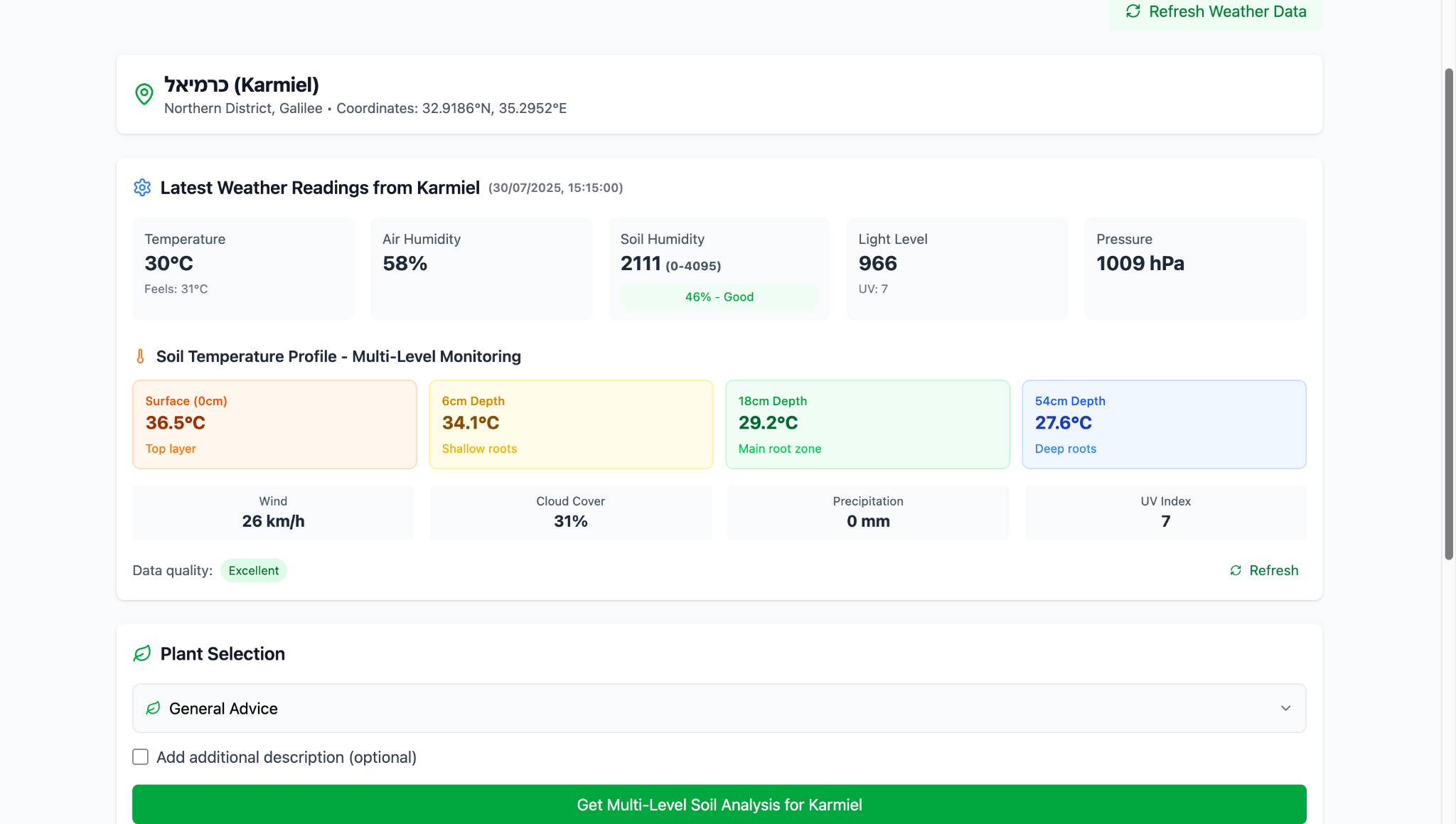
Behavior:

* The recommendation is updated automatically based on sensor data
* Changing the time range refreshes the heatmap view
* Users can zoom in/out or download the heatmap

Possible Errors:

* If no sensor data is available, the recommendation may be missing or an error message may appear

7. Weather and Soil Analysis Dashboard



Purpose:  
 This screen displays up-to-date weather data and a multi-level soil temperature profile for a selected location (e.g., Karmiel). It aims to give farmers a comprehensive view of environmental conditions to support accurate crop management decisions.

Screen Components:

* Location Header:  
  + City name
  + Coordinates
  + Geographic region
* Weather Data:  
  + Temperature
  + Air Humidity
  + Soil Humidity
  + Light Level
  + Air Pressure
  + UV Index
* Soil Temperature Profile – Multi-Level Monitoring:  
  + Surface (0 cm) – Top layer
  + Intermediate Depth (6 cm) – Shallow roots
  + Main Root Zone (18 cm)
  + Deep Roots (54 cm)
* Additional Environmental Parameters:  
  + Wind Speed
  + Cloud Cover
  + Precipitation
  + Data Quality
* Plant Selection Section:  
  + Dropdown to select plant type
  + General recommendation based on current conditions
  + Optional field for additional description
  + Button to run advanced soil analysis

User Actions:

* View current environmental data
* Examine soil temperature at different depths
* Select desired plant type for tailored recommendations
* Refresh data if needed

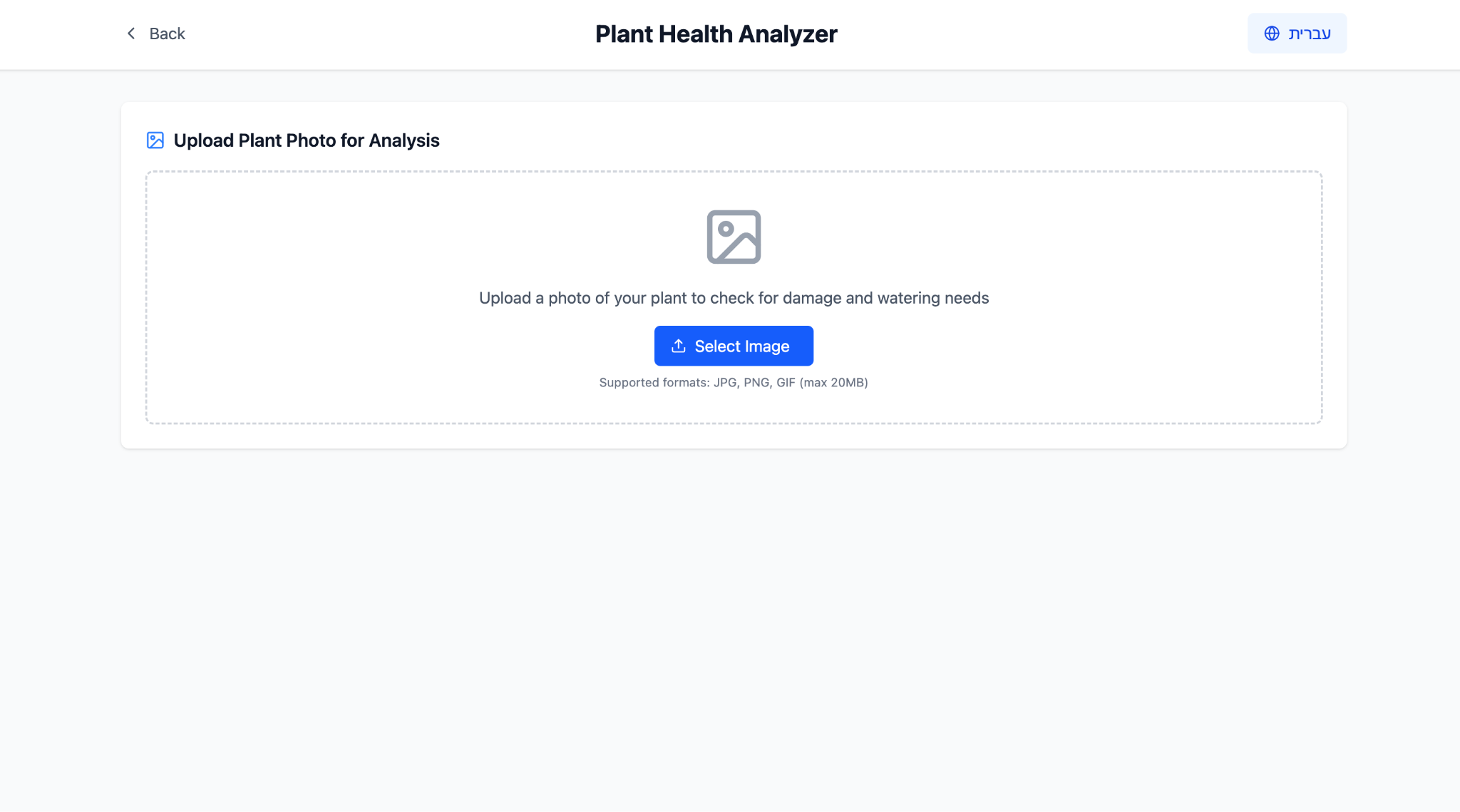
Behavior:

* Data is updated in real time or via the "Refresh" button
* Recommendations change based on the selected plant type
* Color indicators represent temperature levels at different soil layers

Possible Errors:

* Disconnection of the weather station or sensors may result in incomplete data
* Choosing the wrong plant type may lead to inaccurate recommendations

8. Plant Health Analyzer Screen



Purpose:  
 This screen allows the farmer to upload a photo of a plant from the field so that the system can analyze its condition using artificial intelligence. The result includes diagnosis of potential issues (such as leaf damage, dryness, or disease) and tailored irrigation recommendations.

What appears on the screen:

* Header: *Plant Health Analyzer*
* Instruction: Upload a photo of your plant for analysis
* File upload button: *Select Image*
* Supported formats: JPG, PNG, GIF (max 20MB)

What the user is expected to do:

* Take a photo of the plant (or choose an existing one)
* Upload the image via the button
* Wait for the system to analyze and provide feedback

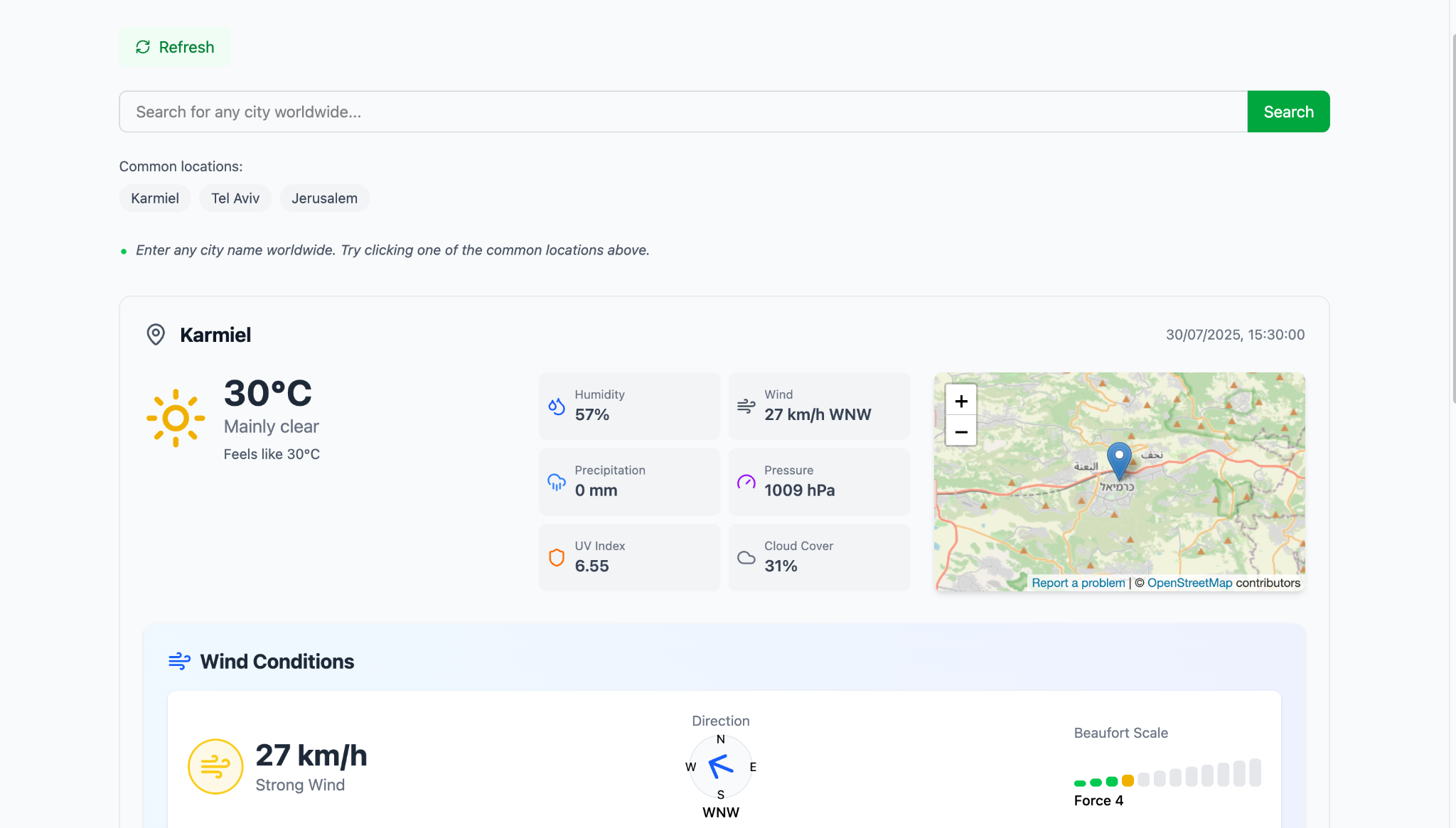
Behavior:

* After uploading, the system automatically performs an AI analysis and displays the result on the next screen (not shown here)
* If the format or file size is invalid – an error message is shown

Possible Errors:

* Unsupported file (wrong format or too large)
* Analysis failure (e.g., blurry image or unclear background)

9. General Weather Conditions Screen



Purpose:  
 This screen allows users to search and monitor live weather data from any city in the world, with a focus on agricultural relevance such as wind, humidity, UV index, and more.

What appears on the screen:

* Search Bar: Enter any city name worldwide to view weather conditions
* Quick Access Buttons: Common cities (e.g., Karmiel, Tel Aviv, Jerusalem)
* Refresh Button: To reload the latest weather data

Weather Data Displayed:

* Temperature (and "Feels like" temperature)
* Weather status (e.g., Mainly clear)
* Humidity
* Wind (Speed + Direction)
* Precipitation
* Air Pressure
* UV Index
* Cloud Cover
* Map showing the selected location

Wind Conditions Section:

* Wind speed (with intensity indication: e.g., Strong Wind)
* Wind direction (Compass-style)
* Beaufort Scale (visual indicator and numeric value)

What the user is expected to do:

* Search for any city to retrieve real-time weather conditions
* Use the data to assess field readiness or plan agricultural activities
* Track wind and UV intensity for safety or irrigation decisions

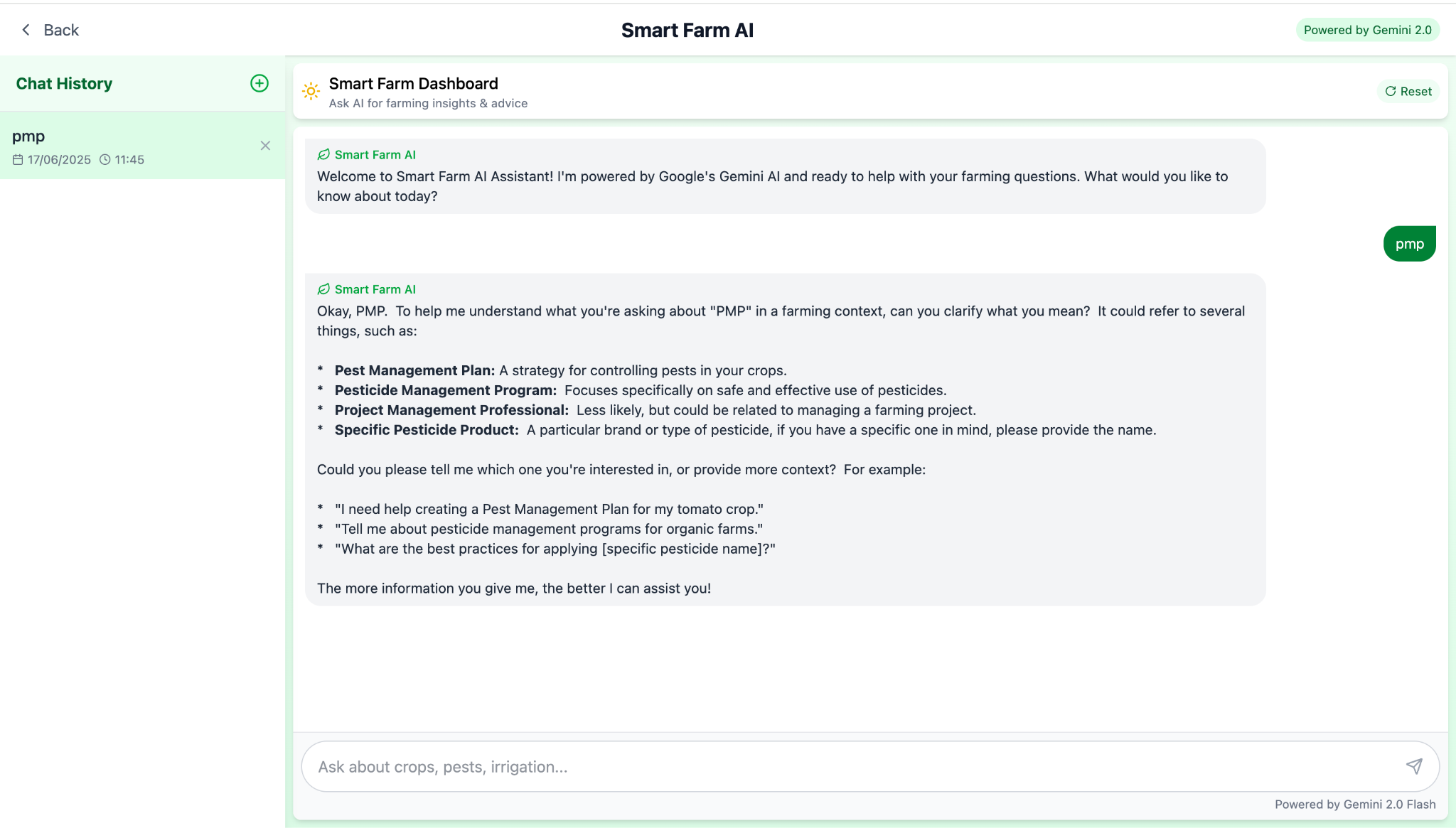
Behavior:

* Data updates every few minutes
* Clicking on a location shows its exact coordinates and updates the values
* All components are clickable for deeper analytics (in other modules)

Possible Errors:

* Invalid city name → shows error or no results
* Network issues → failure to load weather data

10.AI-Based Farming Assistant (Smart Farm AI)

****

Purpose:  
 This screen allows farmers to interact with an AI assistant and receive personalized agricultural advice on topics such as pests, pesticides, irrigation, and more.

What appears on the screen:

* Title: Smart Farm AI
* Description: “Ask the AI for insights and agricultural advice”
* Recent Conversation: Displayed on the left side with a timestamp
* Chat Window: Shows the user’s questions and the assistant’s responses
* Input Field: Allows users to type new questions at the bottom of the screen

What the user is expected to do:

* Type any question related to agriculture such as pests, irrigation, pesticide use, crop health, field conditions, etc.
* Continue the conversation with the assistant to receive deeper, tailored recommendations

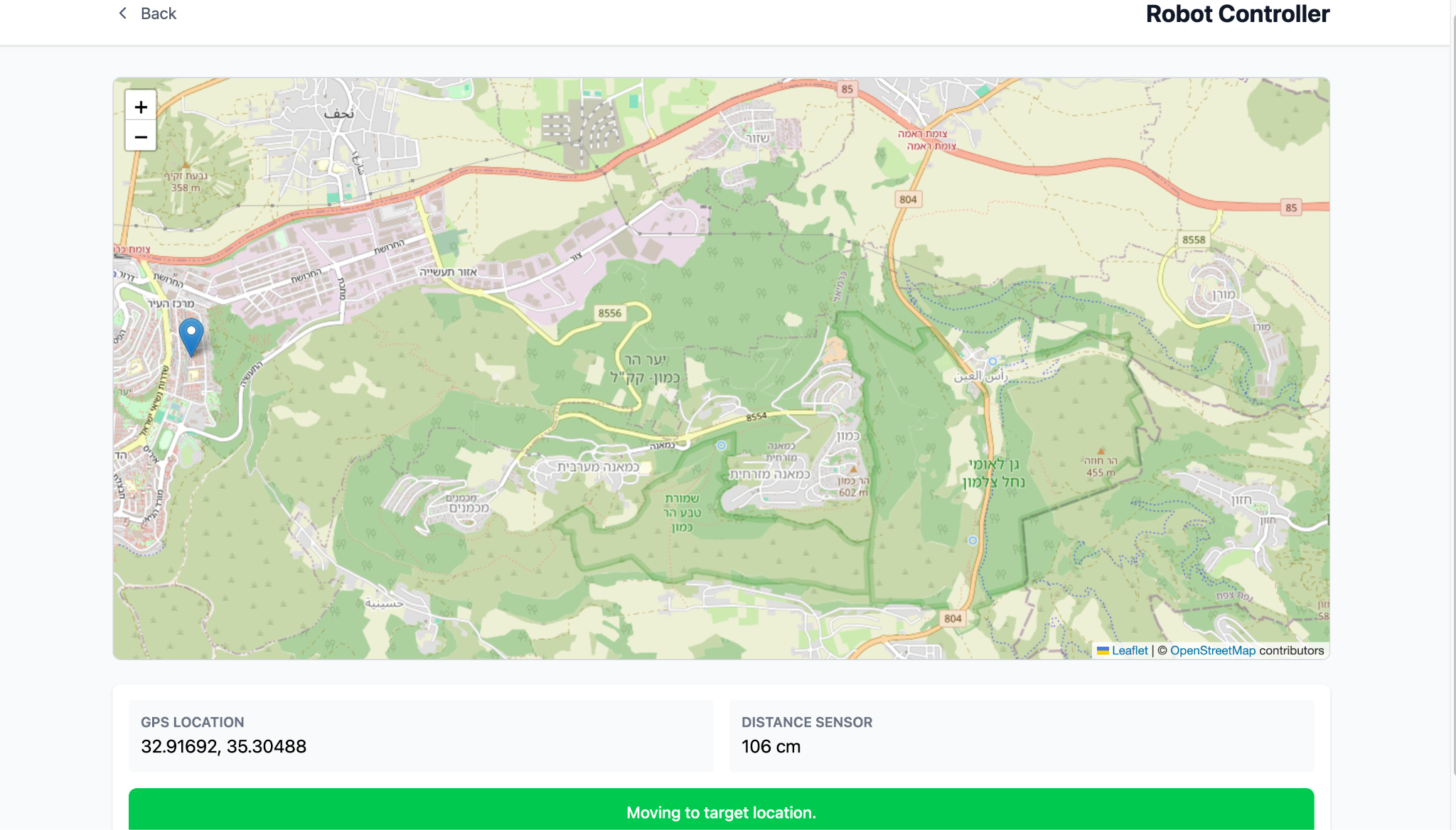
Behavior:

* The smart assistant replies to user questions and offers relevant advice
* If additional information is needed, the system asks follow-up questions
* The conversation flows naturally, similar to a chat interface

Main Uses:

* Receive personalized agricultural guidance powered by AI
* Get real-time support for on-field decision-making
* Obtain smart explanations and recommendations on topics like pests, irrigation, plant diseases, pesticide application, and environmental conditions

11. Robot Controller

****

Purpose:  
 This screen allows the farmer to monitor the real-time location and movement of the field robot. The system uses GPS coordinates and distance sensors to display the robot’s position on the map and to guide it toward its target location. This is essential for tasks such as precision irrigation, monitoring, or field navigation.

What appears on the screen:

* Header: Robot Controller
* Map: Interactive map showing the robot’s current location with a pin
* GPS Location: Displays the robot’s current coordinates
* Distance Sensor: Indicates the measured distance from the robot to an object or target
* Status Message: A green bar at the bottom shows the robot’s current activity – e.g., “Moving to target location.”

What the user is expected to do:

* View the robot’s location and movement on the map
* Use the GPS and distance sensor data to track the robot’s path
* Respond or intervene if needed based on its navigation (e.g., obstacle detection)

Behavior:

* The map updates dynamically as the robot moves
* The distance sensor displays live data in centimeters
* The robot status message informs the user of the current action (moving, idle, etc.)

Possible Errors:

* GPS signal loss (resulting in no location update)
* Sensor failure or inaccurate reading
* Delayed map refresh due to network issues

**Demonstration Video :**

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

***10.Maintenance Manual***

### **📄 File: GeminiPlantAnalyzer.jsx**

🧩Short Description:  
 A React component that serves as the main page for analyzing plant images.  
 Users can upload an image of a plant and optionally provide additional instructions.  
 The component communicates with Google’s Gemini API to perform an analysis that includes:

* Identifying the plant type
* Assessing damage
* Determining watering needs
* Evaluating general health
* Providing care recommendations  
   The interface supports both English and Hebrew, including text direction adjustments (RTL).

🧱 Key Objects and State:

* selectedImage: The uploaded image
* imagePreview: Base64 preview
* language: Current interface language ("english" / "hebrew")
* uiText: Contains all UI texts in both languages
* plantIdentification, damageAssessment, wateringNeeds, overallHealth, recommendations: AI results
* API\_KEY, API\_URL: Gemini API connection settings
* additionalPrompt: User-added instructions
* isAnalyzing, analysisError: States for loading and error tracking

⚙️ Functions and What They Do:

* handleImageChange: Loads the image, creates preview, resets results
* resetAnalysis(): Clears all analysis data and errors
* handleBackClick(): Navigates to the previous page
* handleLanguageToggle(): Switches between English and Hebrew and resets analysis
* parseStructuredResponse(text): Parses Gemini's structured tags ([PLANT\_TYPE]) or breaks into paragraphs if unstructured
* formatText(text): Formats text: bullets, bold, and language alignment
* handleAnalyzeImage(): Converts image to base64, builds prompt + user instructions, sends to Gemini, parses response
* convertFileToBase64: Converts image file to base64 string

### **📄 File: SmartFarmAdvisor.jsx**

🧩 Short Description:  
 Main React component for the “Smart Farm Advisor” app.  
 Displays up-to-date weather data from Karmiel, including multi-depth soil temperatures, and provides personalized agricultural recommendations based on AI analysis from Google Gemini API.  
 Supports bilingual UI (Hebrew/English) and plant type selection.

🧱 Key Objects:

* useState, useEffect: For state and initial loading
* useNavigate: For page navigation
* weatherData: All processed meteorological data
* language, uiText: Language and translations
* commonPlants: Plant list for user selection
* Gemini API: Sends prompts and receives advice
* sectionTags: Identifiers for AI response parts

⚙️ Functions:

* fetchKarmielWeatherData: Gets real-time weather via Open-Meteo (includes multi-depth soil temps)
* getMoistureStatus: Converts numeric values to labels like “Very Dry”
* getAiAdvice: Builds prompt from data, sends to Gemini, processes result
* parseStructuredAdvice: Parses AI response into structured fields
* formatAdviceText: Styles the AI advice for HTML view
* handlePlantSelect: Updates selected plant
* getSelectedPlantName: Returns plant name based on selected language
* handleGetAdvice: Triggers AI recommendation function
* handleBackClick: Navigates back
* handleRefresh: Refreshes weather data
* handleLanguageToggle: Switches language and clears old advice

### **📄 File: SensorAnalytics.jsx**

🧩 Short Description:  
 React component that displays graphs and smart stats of environmental sensor data on a farm.  
 Covers temperature, light, humidity, pressure, and soil moisture.  
 Pulls data from Firebase Firestore, computes trends, and shows them in dynamic graphs.  
 Includes custom time ranges, sensor toggles, and CSV export.

🧱 Key Objects:

* useState, useEffect: Component state and initial load
* useNavigate: Navigation control
* Firebase Firestore functions: collection, getDocs, etc.
* combinedData, temperatureData, lightData, soilData, humidityData, pressureData: Data arrays per sensor
* stats: Holds stats like current, average, min/max, and trend
* customDateRange, selectedSensors: For filters and display
* Recharts (LineChart, Line, etc.): For graph display

⚙️ Functions:

* fetchSensorData: Gets data for selected time, calculates stats, updates UI
* processCombinedData: Merges sensor timelines for unified display
* calculateTrend: Uses linear regression to determine trend (up/down/stable)
* handleBackClick: Navigates back
* handleRefresh: Reloads sensor data
* handleTimeRangeChange: Updates time window and refetches
* handleSensorToggle: Shows/hides selected sensors
* handleCustomDateSubmit: Loads data for custom range
* exportData: Converts data to CSV for download
* getTrendColor, getTrendIcon, getSensorColor, getSensorUnit, getSensorLabel: Helpers for UI styling
* CustomTooltip: Shows detailed sensor info on graph hover

### **📄 File: SensorsPage.jsx**

🧩 Short Description:  
 React component showing a real-time sensor dashboard.  
 Fetches live data from Firebase Firestore: temperature, light, soil/air humidity, and pressure.  
 Includes status indicators (normal, warning, critical) and smart alerts.  
 Supports manual refresh and role-based access.

🧱 Key Objects:

* useState, useEffect: State for loading, user, sensors, alerts
* Firebase Firestore methods: collection, query, where, etc.
* sensors: Object with sensor value, unit, status, update time
* alerts: Alert list based on current sensor status
* userData: Info about logged-in user and role
* isRefreshing: Boolean to show refresh status

⚙️ Functions:

* useEffect: Loads user data from localStorage, then loads sensors based on role
* determineSensorStatus: Assigns “normal”, “warning”, or “critical” based on sensor type and value
* fetchSensorData: Fetches recent sensor data, determines status, raises alerts, updates UI
* formatTimestamp, formatAlertDate: Converts timestamps for display
* handleBackClick, handleRefresh: Navigation and manual refresh
* getStatusColor, getStatusIcon: Display helpers for sensor state

### **📄 File: SpatialModelDashboard.jsx**

🧩 Short Description:  
 Smart irrigation dashboard React component showing temperature, air humidity, and irrigation recommendation via an interactive heatmap using Plotly.  
 Fetches real-time sensor data from Firebase, computes recommendations, and overlays it on a color-coded 2D map.  
 Supports time range selection (24h/7d/30d), live updates, and trend display.

🧱 Key Objects:

* useState, useEffect: Internal states for user data, sensors, graphs
* Firebase Firestore: For user and sensor data
* sensorData, latestReadings, plotData, plotLayout: Graph and reading states
* Plotly (Plot): For heatmap visualization

⚙️ Functions:

* useEffect: Loads user data and sensors if authorized
* fetchSensorData: Gets data for time range, computes irrigation scores, updates map
* generateHeatmapData: Builds 0–10 irrigation score matrix based on temperature + humidity
* handleBackClick, handleRefresh, handleTimeRangeChange: Standard controls
* getRecommendationInfo: Returns text, color, and icon for current irrigation recommendation

### **📄 File: WeatherPage.jsx**

🧩 Short Description:  
 React component for global weather forecasts using Open-Meteo API.  
 Allows searching cities, displays current conditions, daily/hourly forecasts, wind indicators, and extras like sunrise/sunset and UV index.  
 Includes support for major Israeli cities and auto-refresh.

🧱 Key Objects:

* useState, useEffect: For loading, errors, data updates
* Firebase and Open-Meteo APIs
* cities, weatherData, commonLocations: City info and weather data
* EnhancedWindDetails: Internal component for wind metrics
* OpenStreetMap + Open-Meteo APIs: For coordinates and weather queries

⚙️ Functions:

* getWeatherIcon, getWeatherCondition: Translates codes to visuals/text
* formatTime: Formats date strings
* fetchWeatherDataForCities: Pulls forecast per city from Open-Meteo
* useEffect: Sets auto-refresh every 15 min
* handleBack, handleInputChange, handleSearch: For navigation and search
* getWindDirection, getBeaufortScale: Calculates and names wind power
* EnhancedWindDetails: Displays wind speed, direction, safety factors

***11.Refrences***

1. **Google Gemini API** – Used for AI-based plant health analysis and smart advisory system.  
   <https://ai.google.dev/gemini>
2. **Firebase Firestore** – Real-time NoSQL database for storing and retrieving sensor data.  
    [*https://firebase.google.com/docs/firestore*](https://firebase.google.com/docs/firestore)
3. **Open-Meteo API** – Open-source weather forecast API used for integrating real-time weather predictions.  
    [*https://open-meteo.com*](https://open-meteo.com)
4. **Plotly.js** – JavaScript graphing library used for rendering interactive heatmaps.  
    [*https://plotly.com/javascript/heatmaps*](https://plotly.com/javascript/heatmaps)
5. **React.js** – JavaScript library for building user interfaces. Used to develop all dashboard components.  
    [*https://reactjs.org*](https://reactjs.org)
6. **Vercel Deployment Platform** – Used for deploying and hosting the web application frontend.  
    [*https://vercel.com*](https://vercel.com)
7. **Google Maps Platform** – Used for geo-location and city search functionalities in the weather module.  
    [*https://developers.google.com/maps*](https://developers.google.com/maps)
8. **Design Thinking Framework** – Methodology followed to understand user needs and define system features.  
    [*https://www.interaction-design.org/literature/topics/design-thinking*](https://www.interaction-design.org/literature/topics/design-thinking)
9. **Kriging & PCA Methods** – Applied in spatial modeling for identifying irrigation needs.  
    *Wackernagel, H. (2003). Multivariate Geostatistics. Springer Science & Business Media.*
10. **SUS Usability Evaluation** – System Usability Scale (SUS) was used to evaluate user experience.  
     *Brooke, J. (1996). SUS: a “quick and dirty” usability scale. Usability Evaluation in Industry.*

1. *Sivakumar, B. (2011). Water crisis: from conflict to cooperation—an overview. Hydrological Sciences Journal, 56(4), 531-552.‏ [*[*CrossRef*](https://www.tandfonline.com/doi/full/10.1080/02626667.2011.580747)*]*

1. [*https://www.un.org/en/*](https://www.un.org/en/)

1. [*https://www.fao.org/home/en*](https://www.fao.org/home/en)

1. *Ragab, M. A., Badreldeen, M. M. M., Sedhom, A., & Mamdouh, W. M. (2022). IOT based smart irrigation system. International Journal of Industry and Sustainable Development, 3(1), 76-86. [*[*CrossRef*](https://ijisd.journals.ekb.eg/article_252796.html)*].*

1. *Vij, A., Vijendra, S., Jain, A., Bajaj, S., Bassi, A., & Sharma, A. (2020). IoT and machine learning approaches for automation of farm irrigation system. Procedia Computer Science, 167, 1250-1257 [*[*CrossRef*](https://www.sciencedirect.com/science/article/pii/S1877050920309078)*].*