

**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)

**Link of the videos that show how the system works:**[**https://drive.google.com/file/d/14u7yIb2ZD99b\_Cc4l8qTmZEe43qwHcxi/view?usp=sharing**](https://drive.google.com/file/d/14u7yIb2ZD99b_Cc4l8qTmZEe43qwHcxi/view?usp=sharing)

**Link github of the code :** [**https://github.com/ramitaha13/Phase2SmartFarm**](https://github.com/ramitaha13/Phase2SmartFarm)

**תיק למשתמש - user guide**

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.

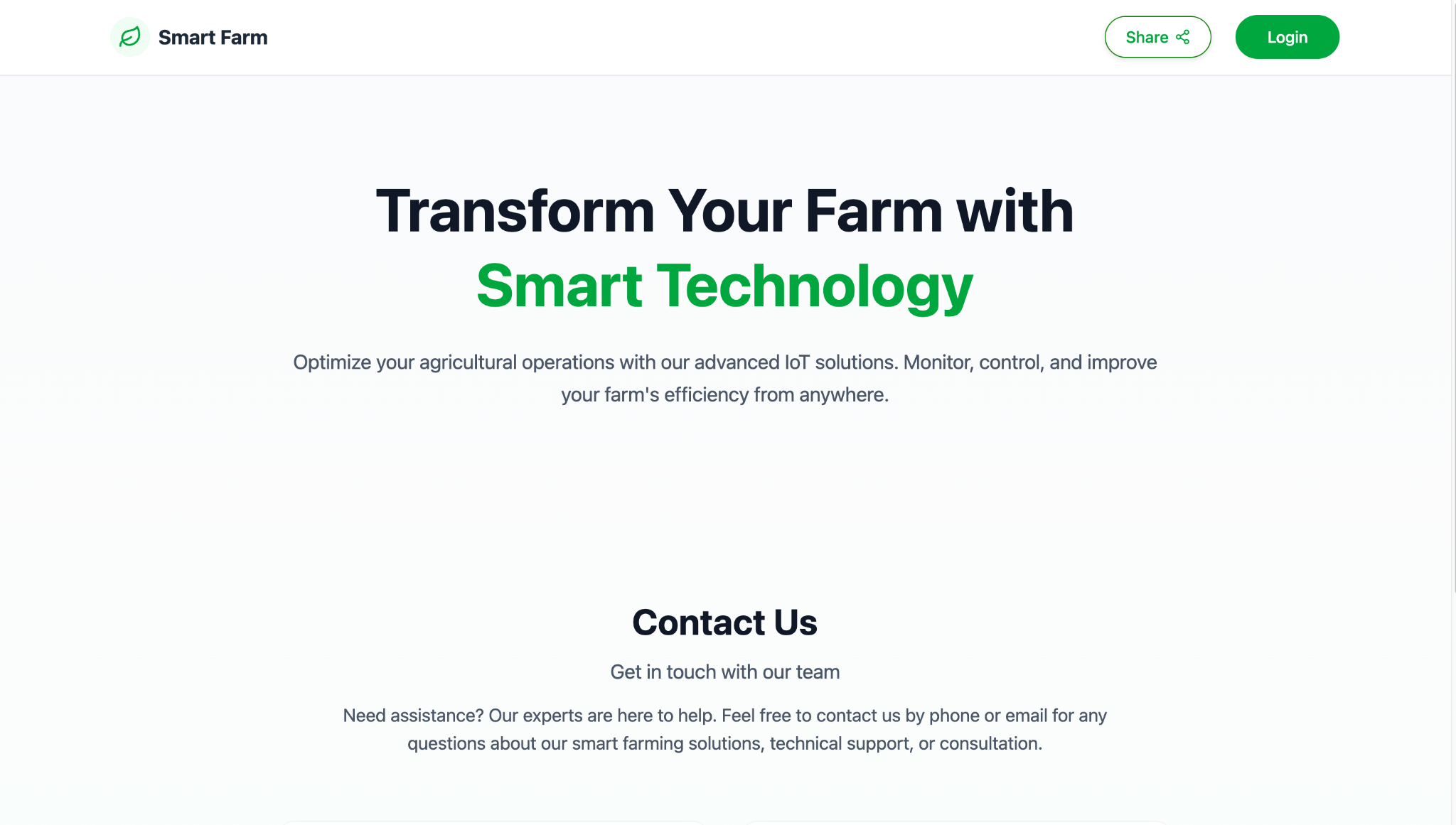
It includes a user-friendly web interface where users can monitor field conditions, control irrigation, and receive real-time alerts.

**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.

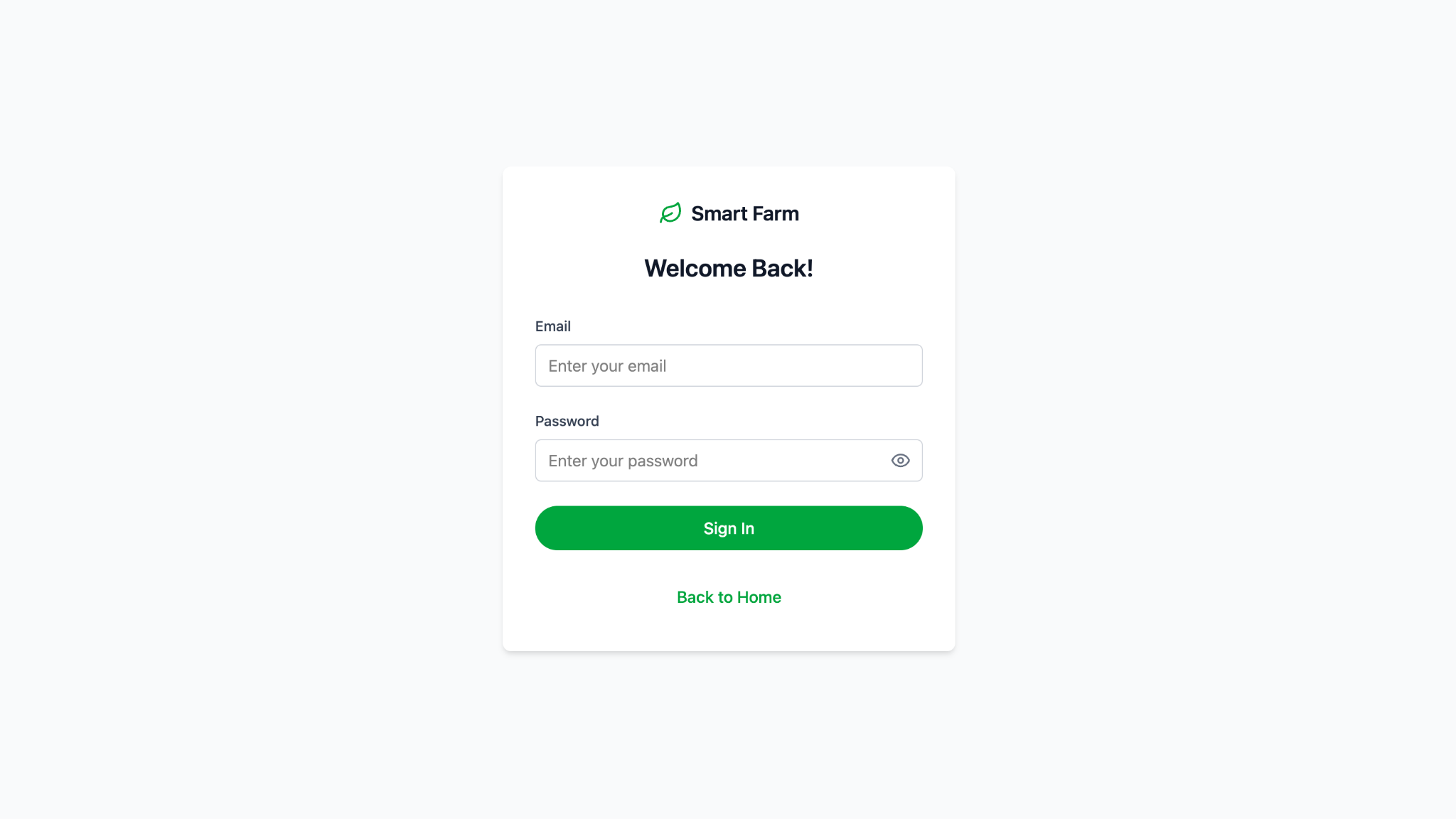
What the user is expected to do:

* 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."*

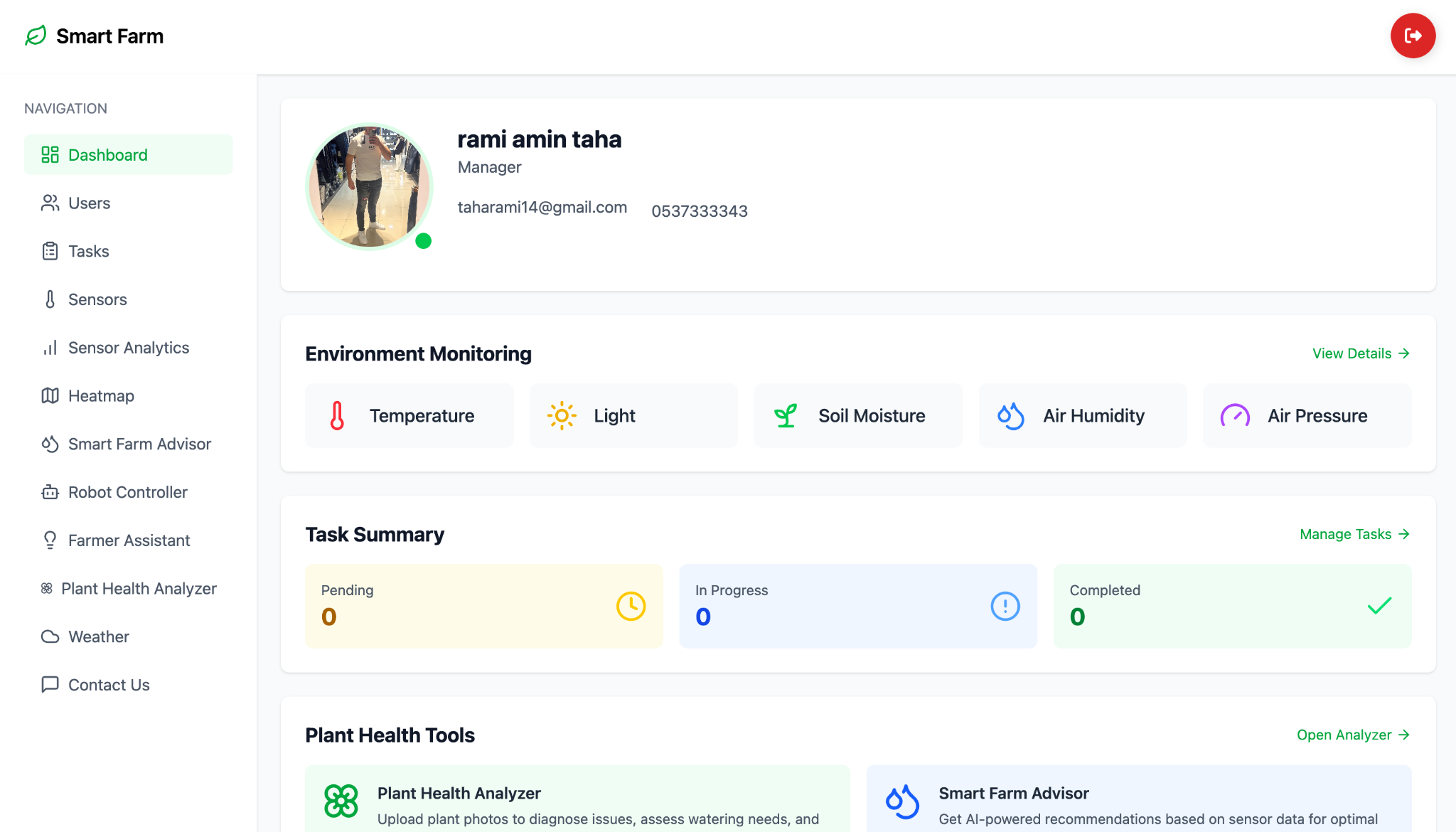
Navigation options:

* 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

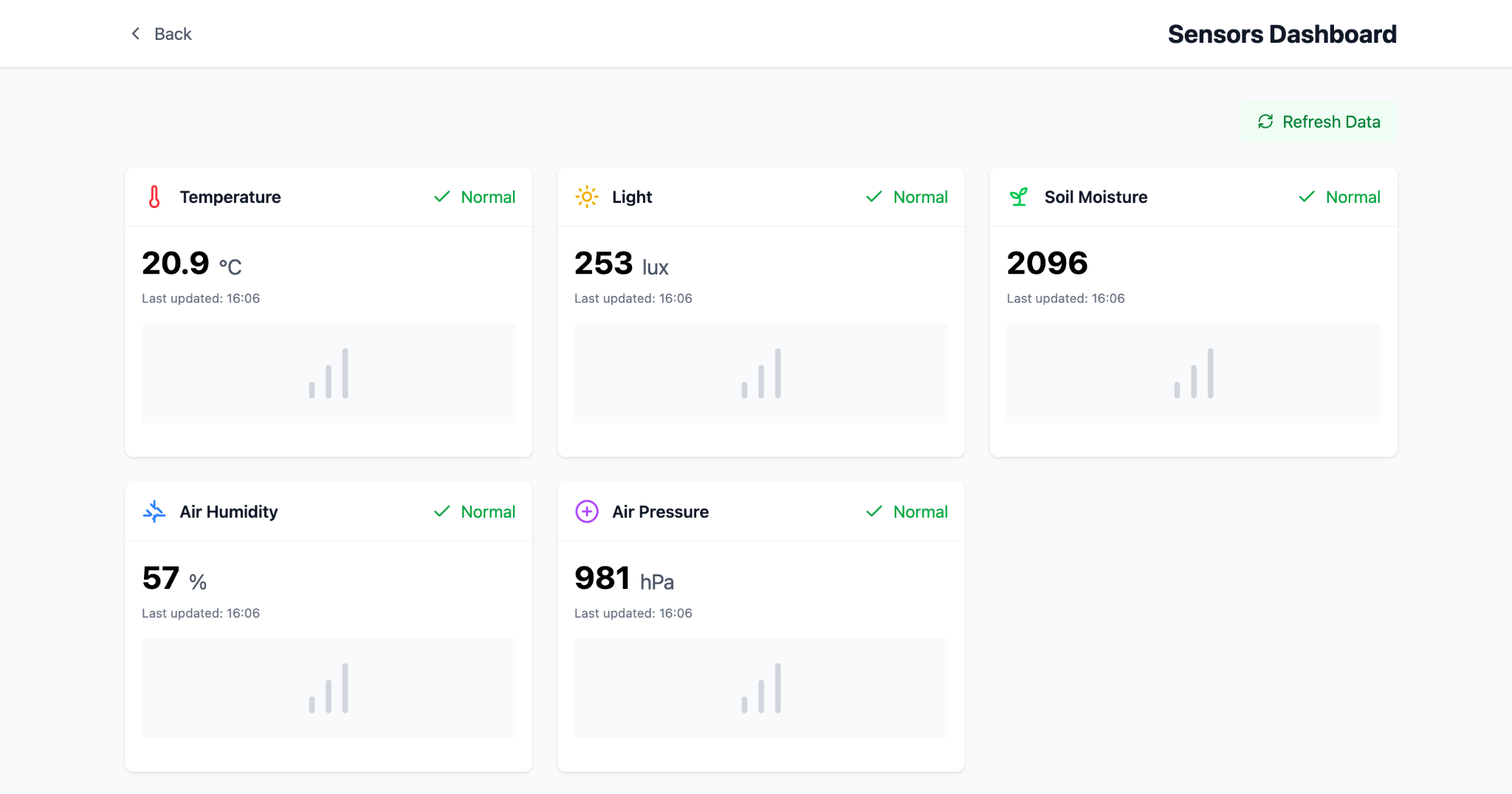
Behavior:

* 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

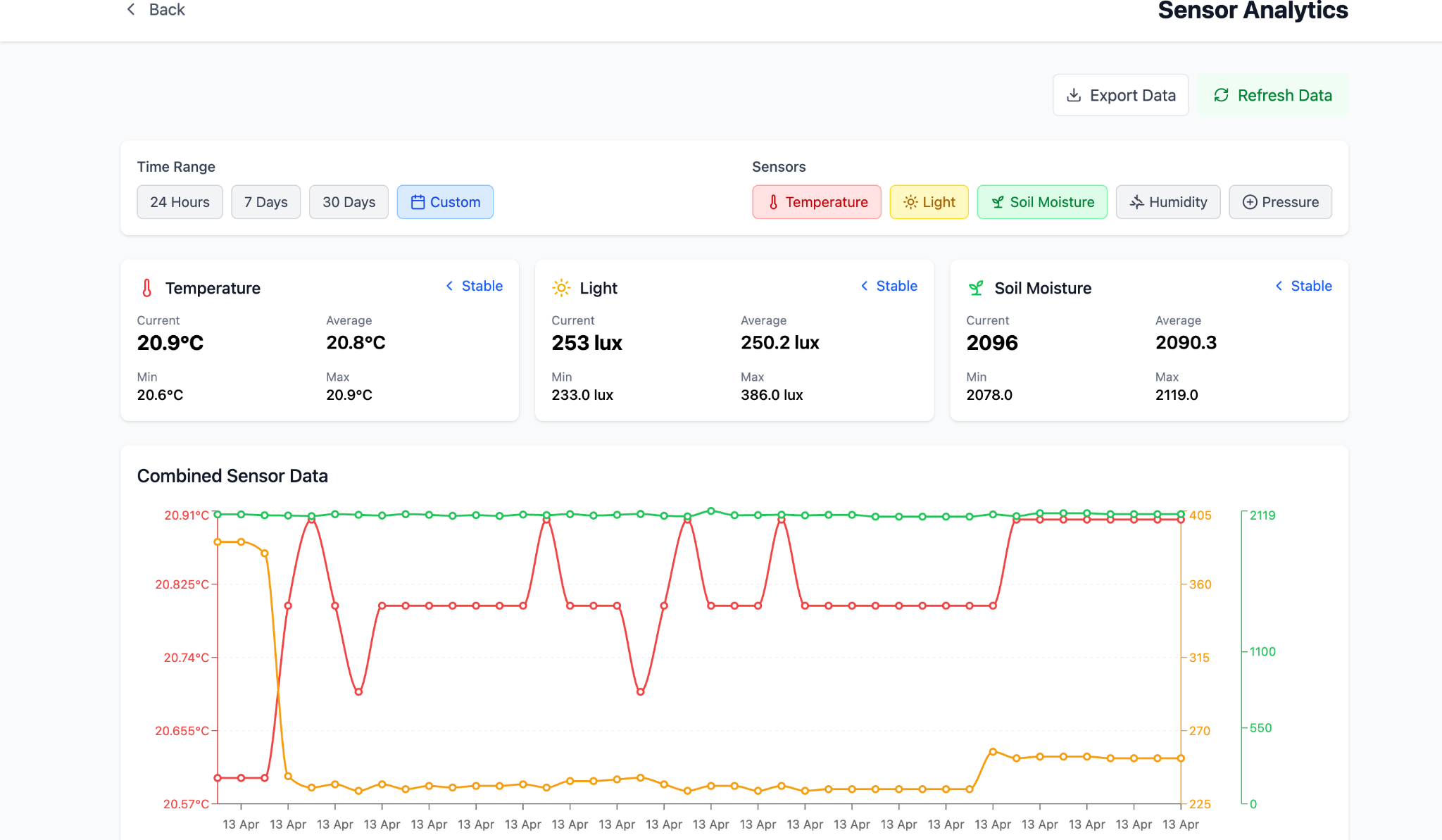
Behavior:

* 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

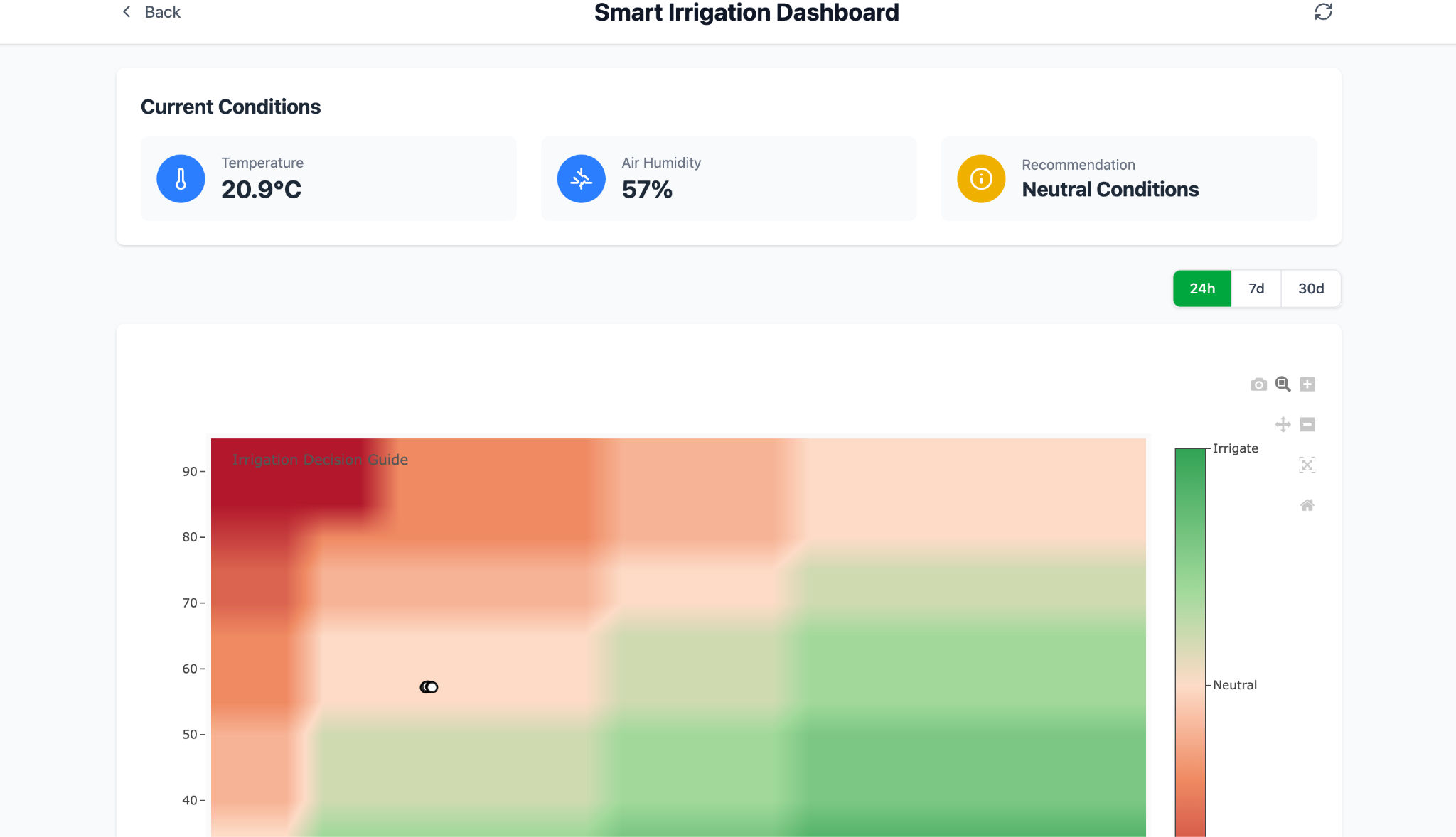
Behavior:

* 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

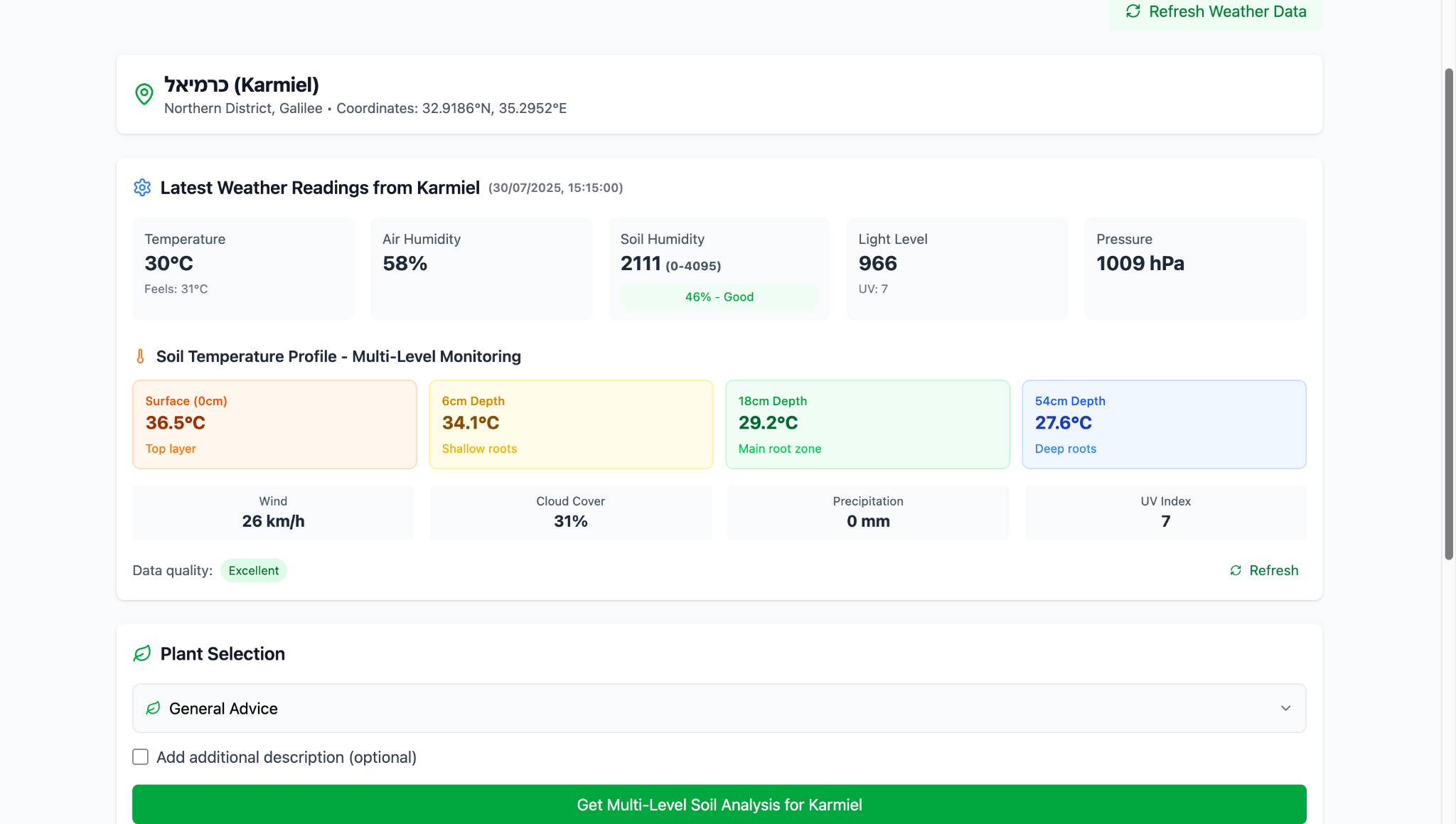
Behavior:

* 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

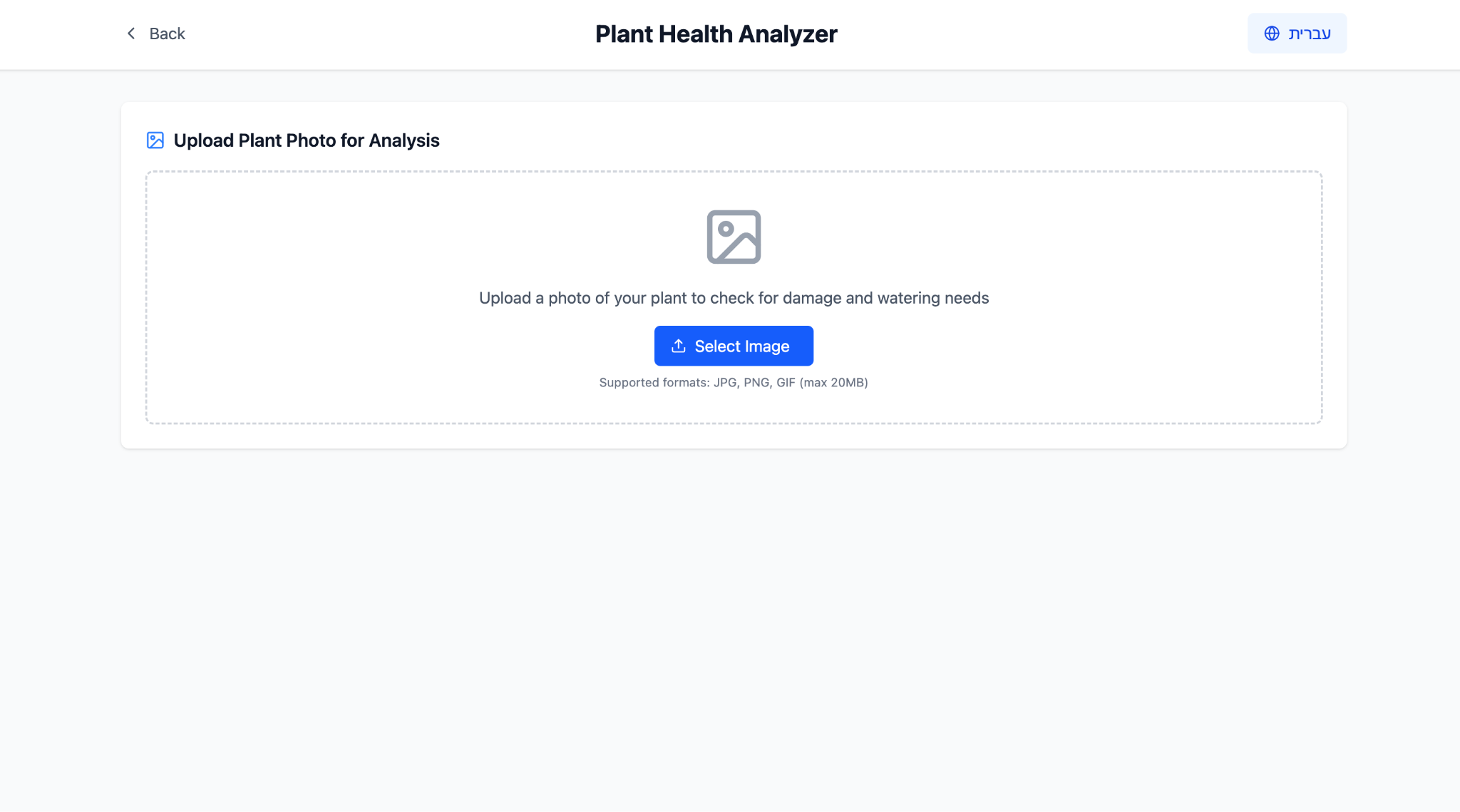
Behavior:

* 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

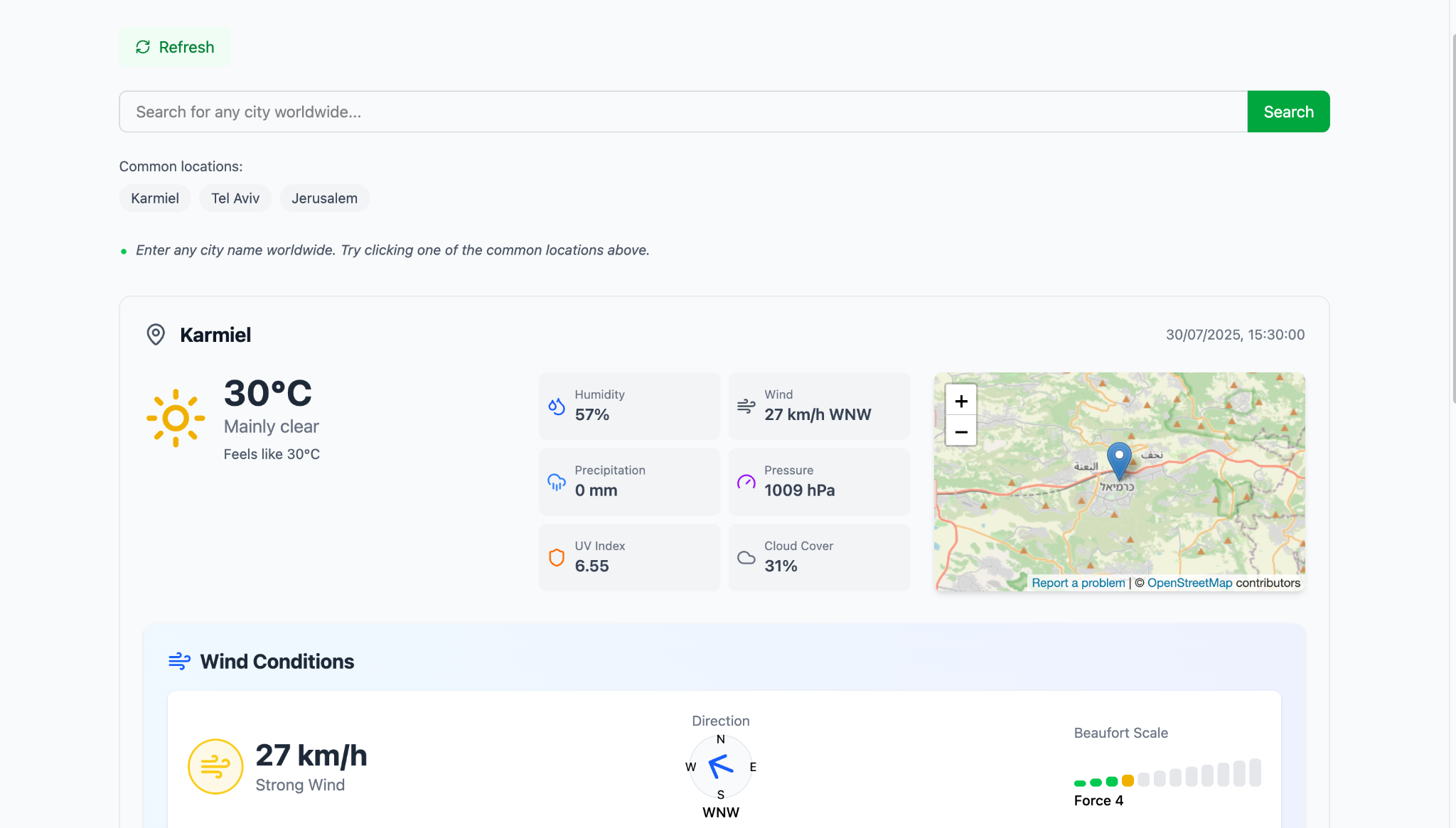
Behavior:

* 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

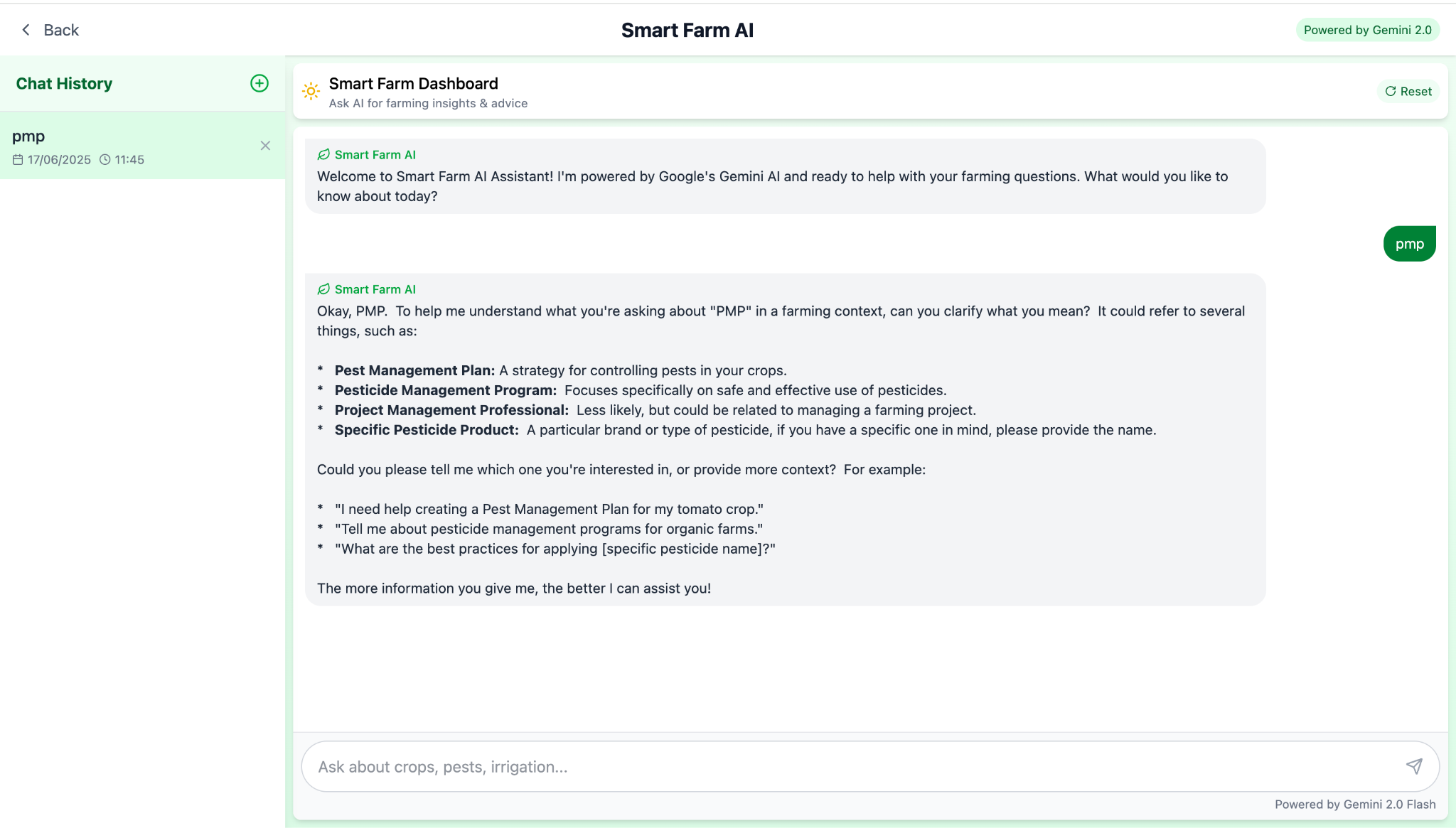
Behavior:

* 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

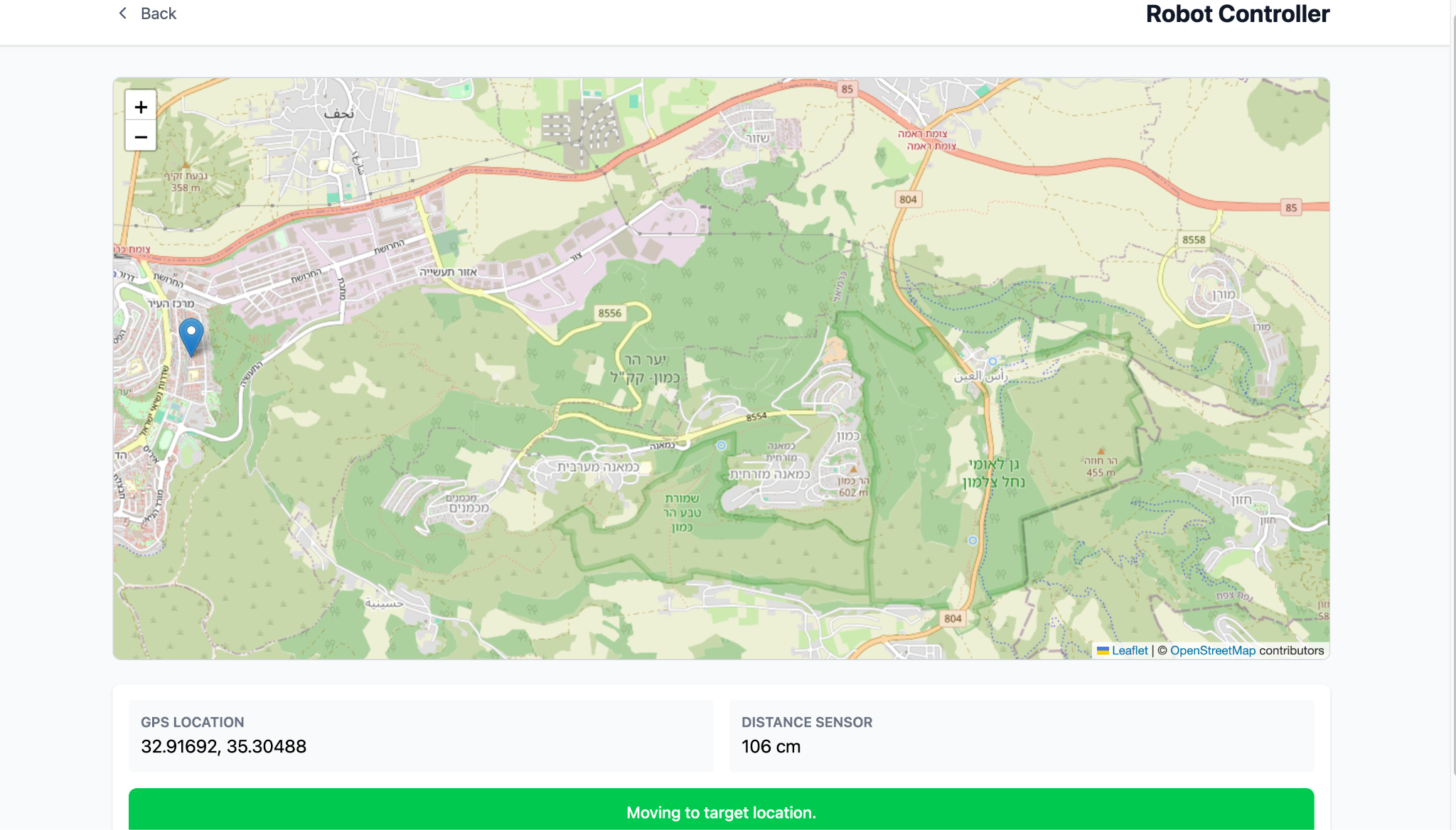
Behavior:

* 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

**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