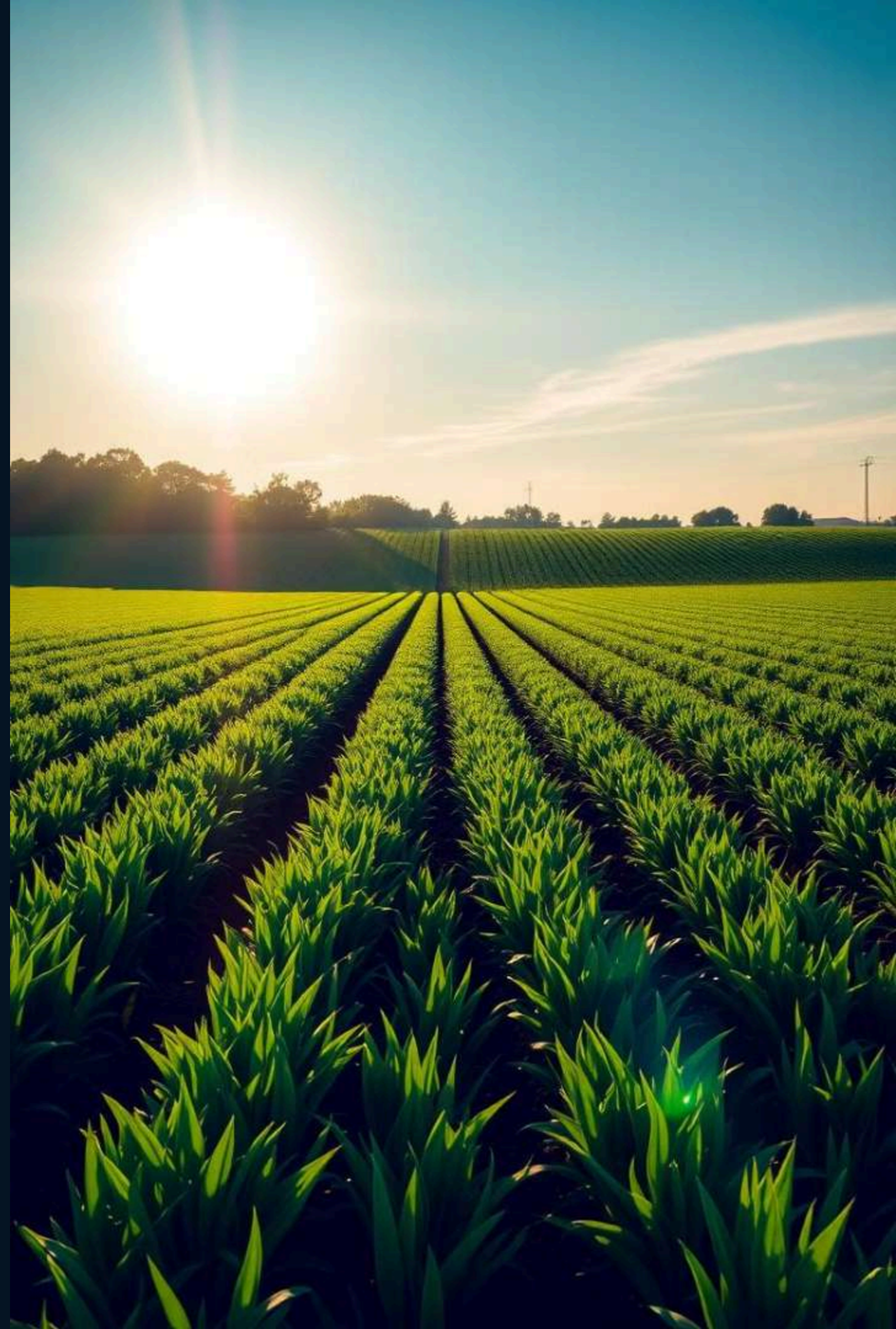
CROP PREDICT

GROUP NAME: KRISHI SEVA
GROUP NO. 7

- Shreya Saraf - RBA 115
- Sudhanshu Shekhar - RBA 119
- Jashwant Nayak - RBA 98
- Probhat Das - RBA 21
- Hijam Jitesor Singh - RBA 14



Problem: Inefficient Agriculture

Resource Waste

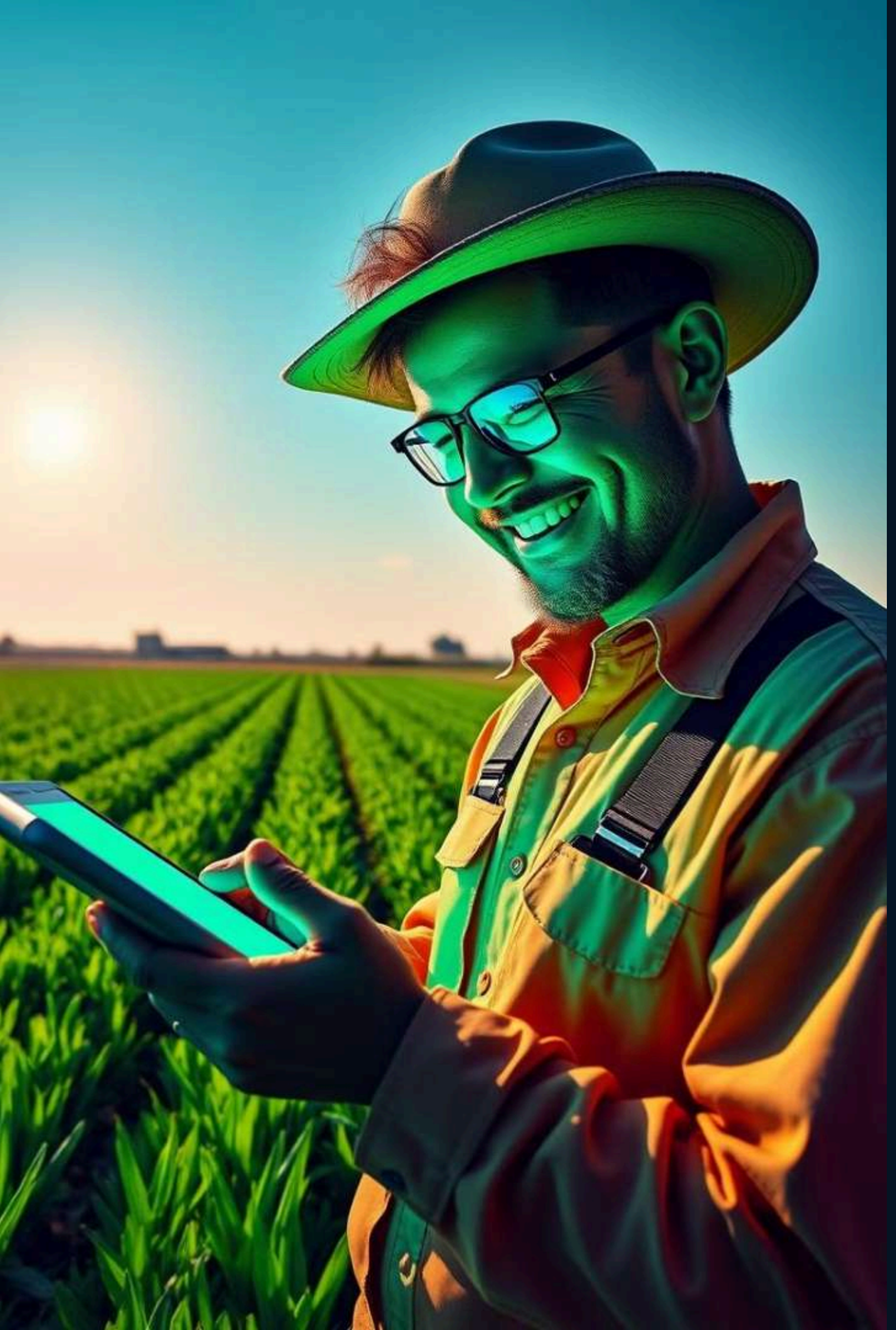
Farmers often over-use water and fertilizers, leading to financial losses and environmental damage.

Unpredictable Yields

Lack of data-driven insights makes it difficult to predict crop growth and optimize production.

Limited Access to Technology

Farmers in developing regions often lack access to affordable and user-friendly AI tools.



Solution: AI-Powered Crop Management Platform

1 **Predictive Insights**

The platform uses machine learning to predict crop growth, water, and fertilizer needs.

2 **Resource Optimization**

Farmers can make data-driven decisions to minimize waste and maximize yields.

3 **User-Friendly Interface**

The platform is designed to be accessible and easy to use for farmers of all experience levels.

SWOT Analysis

Strengths

- AI-driven predictive analytics for yield, water, and fertilizer optimization.
- Affordable freemium model, accessible to rural users.
- Focus on environmental sustainability – promotes eco-friendly farming practices.
- Easy localization for regional languages and markets.

Weakness

- Relatively new entrant – lacks brand recognition.
- Limited on-ground presence and distribution networks.
- Dependence on accurate data (soil, weather, etc.) which might be lacking in some regions.
- Scalability may be slower without significant government/NGO partnerships.

Opportunities

- Huge market in India and other agri-dependent economies.
- Government push for AgriTech (Digital India, PM-KISAN, etc.) provides support.
- Partnerships with IoT vendors, co-operatives, agri-input companies.

Threats

- Competition from well-funded players like CropIn, Fasal, and DeHaat.
- Risk of technological redundancy if not continuously updated.
- Climate unpredictability could affect prediction accuracy and farmer trust.

Value Proposition and Sustainability Analysis

a) Utility / Value Proposition:

- Accurate crop yield forecasts.
- Optimal resource usage → Reduced costs.
- Timely alerts → Efficient farming schedules.
- Improved productivity for small and marginal farmers.

b) Scalability:

- India has over 140 million farmers: Huge TAM (Total Addressable Market).
- Targeting digitally literate farmers first, then expanding via government partnerships.
- Easy language localization for regional rollouts.

c) Economic Sustainability:

- Revenue Model: Subscription-based platform.
- Government & NGO collaboration for large-scale adoption.
- IoT partnerships with agri-tech hardware manufacturers.
- Highly viable due to growing demand for Agri-Tech and AI-based farming.

d) Environmental Sustainability:

- Reduces fertilizer and water overuse, leading to lower pollution and soil degradation.
- Promotes responsible resource consumption.
- Encourages sustainable farming as a habit via insights and education.

Porter’s 5 Forces

Threat of New Entrants	- Moderate to High: Entry barriers in AgriTech are lowering due to open-source AI tools and accessible cloud platforms. - However, building trust among farmers, acquiring agricultural domain knowledge, and collecting quality data remain difficult for new players.
Bargaining Power of Suppliers	- Low to Moderate: Main inputs are datasets (weather, soil, satellite) and cloud computing infrastructure. - With many open-source and low-cost data/APIs, supplier power is low unless tied to exclusive data partnerships.
Bargaining Power of Buyers (Farmers)	- High: Farmers have many free or subsidized advisory apps. - Price-sensitive market, and if predictions are inaccurate, they will quickly switch to alternatives or traditional methods. - Building trust and providing real value is critical.
Threat of Substitutes	- Moderate: Traditional methods (local advisors, experience-based farming, government advisories) still dominate in many regions. - Other digital platforms like WhatsApp groups or YouTube videos also serve as informal substitutes.
Existing Industry Rivalry	- High: Market is crowded with players like CropIn, Fasal, Plantix, and DeHaat. - Many are well-funded and already have government & corporate partnerships. - Differentiation must come from usability, affordability, and hyper-local insights.

AI Model Development

Data Collection

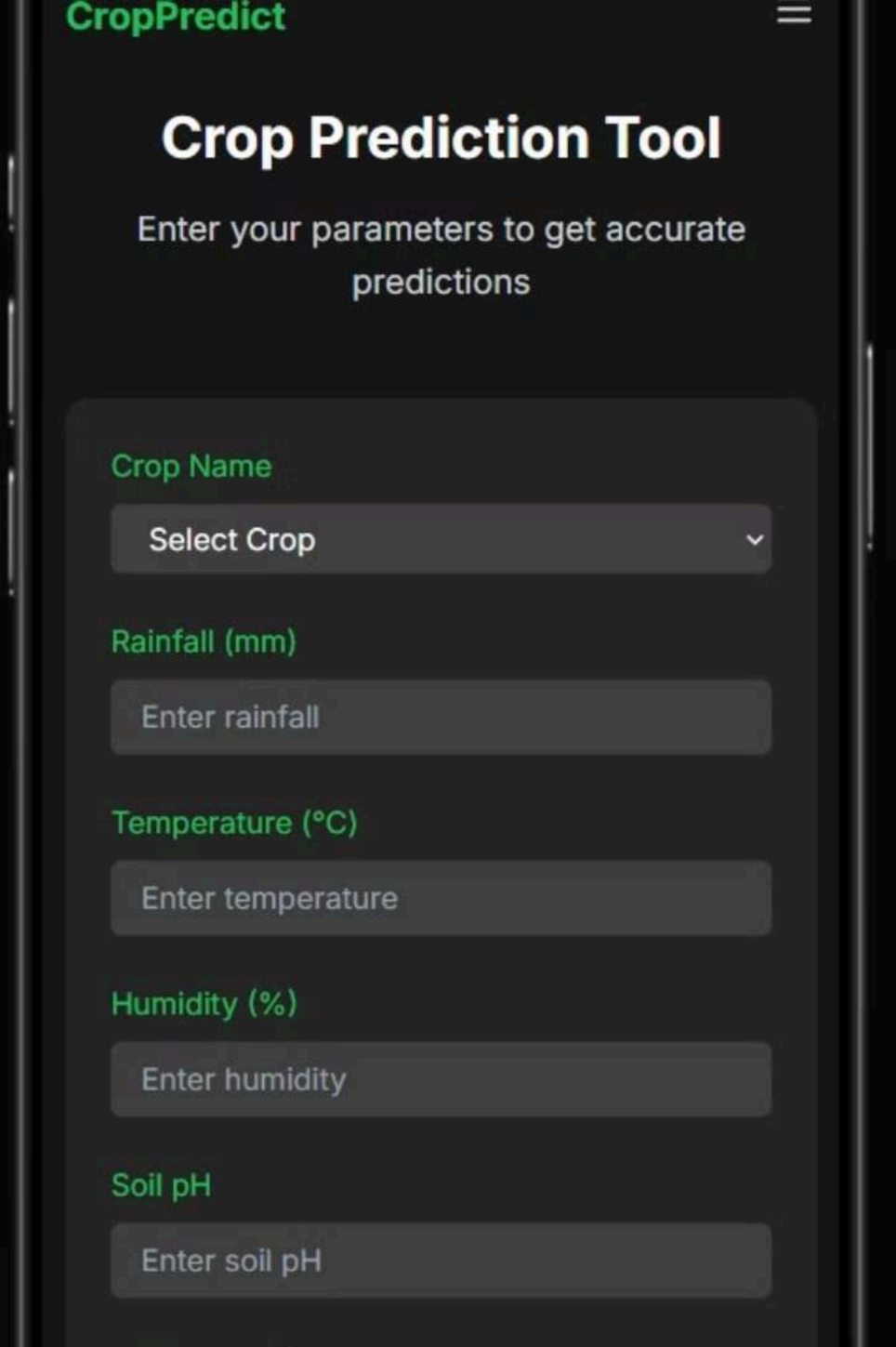
Secondary data from various sources was used to train the model.

Model Training

A Random Forest model was trained to predict crop growth, water, and fertilizer needs.

Model Evaluation

Metrics such as MSE, R^2, and RMSE were used to assess model performance.



App/Website Concept

User Dashboard

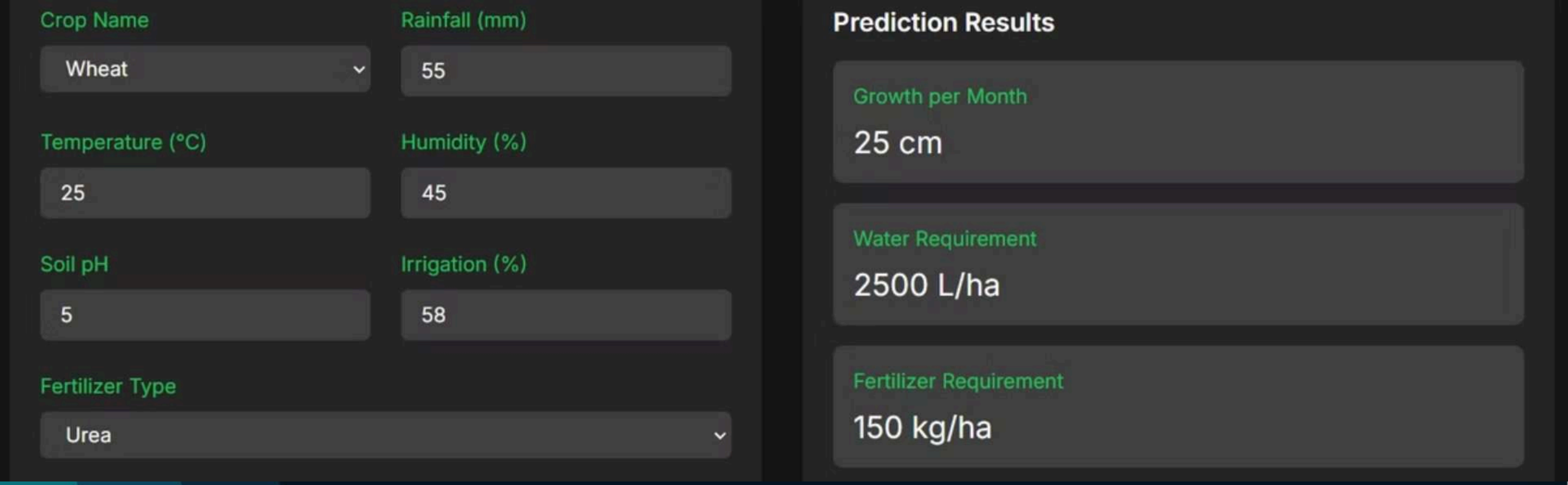
Input crop and environmental data for personalized predictions.

Custom Alerts

Receive timely alerts for irrigation and fertilizer applications.

Mobile Compatibility

Access the platform on any device for real-time insights.



Get Complete Report for the following: Link

- Growth per Month
- Water Requirement
- Fertilizer Requirement

Prototype video link: crop-predict

Empowering Farmers

Reduce Costs

Optimize resource usage to minimize water and fertilizer expenses.

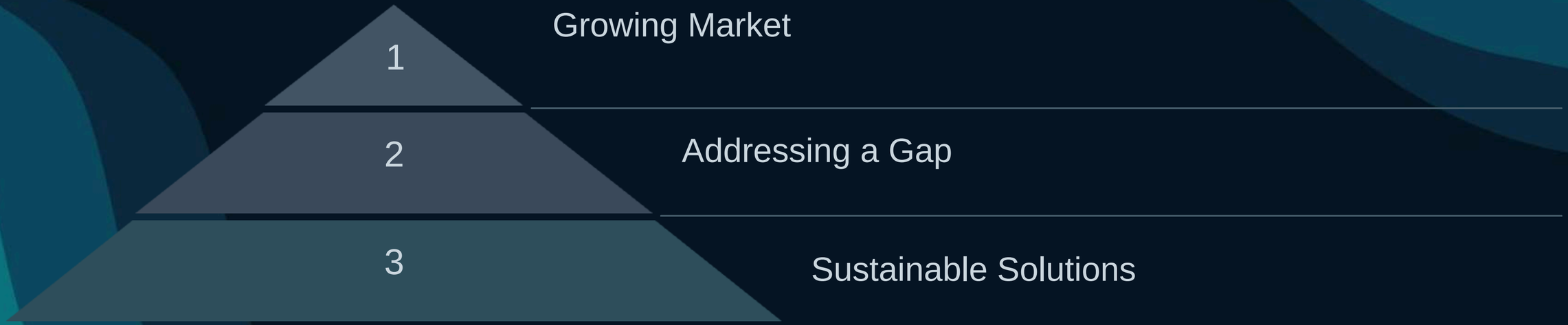
Increase Yields

Data-driven insights lead to improved crop cycles and higher productivity.

Promote Sustainability

Encourage responsible farming practices that conserve natural resources.

Market Opportunity



The market for agricultural AI/ML solutions is expanding rapidly, creating a significant opportunity for our platform.

Revenue Model

1. Subscription Based Web app

- Directly to the farmers
- Collaborating with the government

2. Collaborating with Farm equipment manufactures to implement the IoT sensors

Next Steps: Bringing the Solution to Life

Real-Time Data Source from IoT Devices

To enhance prediction accuracy and enable proactive decision-making, our platform would integrate real-time data streams from IoT sensors deployed in the field. These sensors will capture:

- Soil moisture, pH, and temperature
- Ambient weather conditions (humidity, rainfall, sunlight)
- Irrigation system status and performance

This real-time integration ensures that the AI model adapts continuously to changing farm conditions, delivering up-to-date recommendations and alerts to farmers.

Audio Information for Output (Vernacular Support)

Understanding the linguistic diversity among Indian farmers, our platform includes audio outputs in regional languages. Key features:

- Voice-based recommendations for water/fertilizer usage
- Vernacular language support to ensure accessibility (e.g., Hindi, Bengali, Odia, Tamil)
- Text-to-speech integration for illiterate or semi-literate users

This feature enhances usability and adoption, especially among smallholder farmers.

Python Code Snippets

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('/content/Updated_Crop_Price_Prediction_Dataset.csv')

# Selecting relevant columns for prediction
relevant_columns = [
    "Crop Name", "Rainfall (mm)", "Temperature (°C)", "Humidity (%)",
    "Soil pH", "Irrigation Availability (%)", "Fertilizer Type",
    "Growth per Day (cm/day)", "Fertilizer Cost (₹/kg)"
]
data_filtered = data[relevant_columns]

# Encoding categorical variables
data_encoded = pd.get_dummies(data_filtered, columns=["Crop Name", "Fertilizer Type"], dr

# Splitting the data into features and target variable
X = data_encoded.drop(columns=["Growth per Day (cm/day)"])
y_growth = data_encoded["Growth per Day (cm/day)"]

# Adding computed targets for water and fertilizer requirements
data_encoded['Water Requirement (L/ha)'] = data_encoded['Rainfall (mm)'] * 10  # Example
data_encoded['Fertilizer Requirement (kg/ha)'] = data_encoded['Fertilizer Cost (₹/kg)'] *
y_water = data_encoded["Water Requirement (L/ha)"]
y_fertilizer = data_encoded["Fertilizer Requirement (kg/ha)"]

# Standardizing the features to improve model performance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
# Splitting into training and testing sets
X_train_scaled, X_test_scaled, y_growth_train, y_growth_test = train_te
X_train_water, X_test_water, y_water_train, y_water_test = train_test_s
X_train_fertilizer, X_test_fertilizer, y_fertilizer_train, y_fertilizer

# Hyperparameter tuning using GridSearchCV
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10]
}

# Model for growth prediction
model_growth = RandomForestRegressor(random_state=42)
grid_search_growth = GridSearchCV(model_growth, param_grid, cv=5, scori
grid_search_growth.fit(X_train_scaled, y_growth_train)
best_model_growth = grid_search_growth.best_estimator_

# Model for water prediction
model_water = RandomForestRegressor(random_state=42)
grid_search_water = GridSearchCV(model_water, param_grid, cv=5, scoring
grid_search_water.fit(X_train_water, y_water_train)
best_model_water = grid_search_water.best_estimator_

# Model for fertilizer prediction
model_fertilizer = RandomForestRegressor(random_state=42)
grid_search_fertilizer = GridSearchCV(model_fertilizer, param_grid, cv=
grid_search_fertilizer.fit(X_train_fertilizer, y_fertilizer_train)
best_model_fertilizer = grid_search_fertilizer.best_estimator_
```

```python
# Evaluate models
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    rmse = mse ** 0.5
    return mse, r2, rmse

# Evaluate each model
mse_growth, r2_growth, rmse_growth = evaluate_model(best_model_growth, X_test_scaled, y_growth_test)
mse_water, r2_water, rmse_water = evaluate_model(best_model_water, X_test_scaled, y_water_test)
mse_fertilizer, r2_fertilizer, rmse_fertilizer = evaluate_model(best_model_fertilizer, X_test_scaled, y_fe

# Print evaluation results
print("Growth Prediction - MSE:", mse_growth, "R2:", r2_growth, "RMSE:", rmse_growth)
print("Water Requirement Prediction - MSE:", mse_water, "R2:", r2_water, "RMSE:", rmse_water)
print("Fertilizer Requirement Prediction - MSE:", mse_fertilizer, "R2:", r2_fertilizer, "RMSE:", rmse_fert
```

```
Growth Prediction - MSE: 0.15715651585389737 R2: -0.16851291875508068 RMSE: 0.39642971111395947
Water Requirement Prediction - MSE: 3372.196476124991 R2: 0.9953370556451456 RMSE: 58.070616288489575
Fertilizer Requirement Prediction - MSE: 4.763449173000234 R2: 0.9987937133887964 RMSE: 2.1825327427097707
```

```python
# Function for making predictions
crop_types = data["Crop Name"].unique()
fertilizer_types = data["Fertilizer Type"].unique()

def predict_crop_growth_and_requirements(crop_name, rainfall, temperature, humidity, soil_pH, irrigation, fertilizer_type):
    input_data = pd.DataFrame({
        "Rainfall (mm)": [rainfall],
        "Temperature (°C)": [temperature],
        "Humidity (%)": [humidity],
        "Soil pH": [soil_pH],
        "Irrigation Availability (%)": [irrigation]
    })

    # One-hot encoding for the crop and fertilizer type
    for crop in crop_types:
        input_data[f"Crop Name_{crop}"] = 1 if crop == crop_name else 0

    for fert in fertilizer_types:
        input_data[f"Fertilizer Type_{fert}"] = 1 if fert == fertilizer_type else 0

    # Ensuring column alignment with training data
    missing_cols = set(X.columns) - set(input_data.columns)
    for col in missing_cols:
        input_data[col] = 0

    input_data = input_data[X.columns]

    # Scaling input features
    input_scaled = scaler.transform(input_data)

    # Predicting growth per day and converting to cm/month
    growth_per_day = best_model_growth.predict(input_scaled)[0]
    growth_per_month = growth_per_day * 30

    # Predicting water and fertilizer requirements per hectare
    water_requirement = best_model_water.predict(input_scaled)[0]
    fertilizer_requirement = best_model_fertilizer.predict(input_scaled)[0]

    return {
        "Growth per Month (cm)": round(growth_per_month, 2),
        "Water Requirement (L/ha)": round(water_requirement, 2),
        "Fertilizer Requirement (kg/ha)": round(fertilizer_requirement, 2)
    }
```

```python
# Dynamic user input
crop_name = input("Enter the crop name (e.g., Wheat, Rice): ")
rainfall = float(input("Enter rainfall in mm: "))
temperature = float(input("Enter temperature in °C: "))
humidity = float(input("Enter humidity in %: "))
soil_pH = float(input("Enter soil pH level: "))
irrigation = float(input("Enter irrigation availability in %: "))
fertilizer_type = input("Enter fertilizer type (e.g., Urea, DAP): ")

# Predict based on user input
predicted_values = predict_crop_growth_and_requirements(
    crop_name=crop_name,
    rainfall=rainfall,
    temperature=temperature,
    humidity=humidity,
    soil_pH=soil_pH,
    irrigation=irrigation,
    fertilizer_type=fertilizer_type
)

print("Predicted Values:", predicted_values)
```

```
Enter the crop name (e.g., Wheat, Rice): maize
Enter rainfall in mm: 124
Enter temperature in °C: 42
Enter humidity in %: 45
Enter soil pH level: 7.5
Enter irrigation availability in %: 65
Enter fertilizer type (e.g., Urea, DAP): urea
Predicted Values: {'Growth per Month (cm)': np.float64(33.27), 'Water Requirement (L/ha)': np.float64(1205.3), 'Fertilizer Requirement (kg/ha)': np.float64(25.79)}
```

CONCLUSION

Our AI-powered platform offers a comprehensive, accessible, and intelligent farming assistant. By blending predictive analytics with real-time IoT data and vernacular audio support, we empower farmers to make smarter decisions, reduce costs, and increase productivity.

This innovation stands at the intersection of technology and social impact, addressing food security, resource conservation, and economic growth in the agricultural sector.

Thank You