Samrudhi - Precision Agriculture Platform

Dr. M Raghava Naidu[1], G. Sagar babu[2], K. V. Karthik[3],
A. Bala subramanyam[4], K. Ganesh Babu[5]
[1]*Assistant Professor, Dept. of CSE, Krishna University College of Engineering & Technology, A.P, India*
[2,3,4,5] *UG Students, Dept. of CSE, Krishna University College of Engineering & Technology, A.P, India*

***Abstract*—The given paper shows Samrudhi, a web-based precision agriculture software which unites the real-time weather monitoring and CNN-like crop disease detection under the same roof. Numerous farmers currently have access to weather information or disease notification separately and that disconnect leads to delays in making decisions. Samrudhi resolves this by linking both ends where in times of danger when weather information is provided, disease checks are also provided on the platform in real time. The CNN model was trained using Plant Village images and achieved the accuracy of 98.7 percent when used on test data and produced results within about 2 seconds. It is based on React.js, Django REST Framework and Tailwind CSS and was developed with a focus on smallholder farmers in mind, who should have clear, fast answers, not overwhelming data monitoring dashboard.**

***Index Terms*—Samrudhi; Precision Agriculture; Convolutional Neural Networks; Agro-Meteorology; Django REST Framework; React.js; Plant Disease Detection; Predictive Analytics.**

I. INTRODUCTION

Agriculture has never been an easy task, yet the past 10 years have made it quite difficult. There has been a change in the timing of rainfall. Heat waves come sooner and last longer periods. Once seasonal pests are appearing all year round in various areas. Among farmers who are simply used to experience and observation, these changes have already begun to bring actual harm plants planted when it is late, plants found diseased after it is too late, and dismal crop at harvest. This problem was observed and led to the creation of the idea behind Samrudhi. One thing became apparent: there is nothing missing, namely, data. Their farmers already have several weather apps, notification messages, and prognosticators. The thing that they actually do not have is the ability to link these individual data points into one and timely recommendation. It is imperative to know that the humidity is at 75% but then again, it cannot do you much good unless you know whether or not that level of humidity is safe in your present stage of crop growth or whether the present condition is the type that encourages fungi.

To brid this gap, Samrudhi was constructed. The platform does not present weather information and disease detection as two distinct features as they are perceived as a single problem. When the weather module documents a prolonged humidity in excess of a threshold, the system takes care of marking that as possible disease risk and motivates the farmer to conduct a leaf scan [1]. This sort of interconnected thinking where the atmospheric data, as well as crop health data, is able to communicate with one another, is what differentiates Samrudhi with other tools in this space.

II. RELATED WORK

In the past few years, a good deal of digital farming work has been done, but the vast majority of solutions look at a single aspect of the issue.

A. Weather Forecasting Platforms

Other comparison services such as AccuWeather and Open Weather are good to get data on rainfall and temperature. But they go no further and tell a farmer what he should do with that knowledge. The fact that the current weather prediction indicated that it would rain 60 percent on Thursday does not answer the

question of whether or not Tuesday is safe to spray or whether the soil is already too saturated to irrigate any further [6, 8]. Raw weather data are seldom sufficient when making farming decisions.

B. Plant Disease Diagnostic Tools

Apps such as Plantix are useful after already identifying an issue on the leaf. The problem lies in the fact that the disease is often weeks old in manifestation, noticeable to the naked eye, at the moment of discovery [7]. To top it all, such instruments seldom consider weather conditions that preceded the outbreak. It is not of much use to know what disease is around you, without a clue to why it manifested in a certain manner or why it was not picked sooner [9].

C. Comprehensive Smart Farming Systems

There is indeed a range of full-scale precision agriculture systems and some of these are very good. They rely on in-field IoT devices, satellite camera services and technical infrastructure that are out of the means of most smallholder farmers [4]. These were not created to serve an individual with two-acre farm with a simple smart phone.

D. Where Samrudhi Fits

All these restrictions influenced the design of Samrudhi. It was aimed at one platform that anyone, with access to the internet, could operate - no specialized hardware, no need to subscribe to various platforms, no technical knowledge needed [10]. Pulling weather data stored in Open Mateo and constructed disease detection on top of Plant Village-trained models, the system converts the environmental and crop data into straight forward advice that is tailored to the specific circumstances of the farmer.

III. SYSTEM ARCHITECTURE

There are three layers in the architecture of Samrudhi. They have different tasks to deal with and maintaining separation ensured that the system was easier to test, maintain and expand as time progressed.

A. Frontend Layer

It was created using React.js and styled with Tailwind CSS. The choice of a dark-themed design was actually made due to the response of the users in the early testing - the farm-hands accessing the phone outside in broad daylight could read the interface easier. There are four primary views of the app: Overview, Forecasts, Seasonal AI and Plant Disease AI. All views fetch their data separately, and therefore a sluggish network request in one part does not prevent the rest of the interface to show up.

B. Backend Layer

Handling the server is done by Django REST Framework. Here, Python was the obvious solution since the majority of machine learning libraries required by the project Python-native include TensorFlow, PyTorch, and scikit-learn. Django also handles authentication of API requests, leaving the Open-Meteo credentials server-side and thus barring misuse in case an end-user inspects the network calls made to the front-end.

C. Data and AI Pipeline

Data is processed by three different components prior to reaching the user:1. Weather ingestion is a scheduled task that invokes Open-Meteo at a time interval of 30 minutes and stores the humidity, temperature, wind speed, precipitation, UV index and surface pressure [11, 12].2. Disease inference is to be run on uploading a leaf image by a user.

1. Weather ingestion runs on a scheduled task that calls Open-Meteo every 30 minutes, storing humidity, temperature, wind speed, precipitation, UV index, and surface pressure [11, 12].
2. Disease inference runs when a user uploads a leaf image. The picture is first resized to 224x224 pixels, normalized, and is evaluated by the CNN. The model gives a probability score of each of the 38 disease classes it has been trained on [7, 14].3. The analytics engine implements agronomic threshold rules to the weather data.
3. The analytics engine applies agronomic threshold rules to the weather data. As an instance, when a temperature rises beyond 35C on three days in a row, the system sends out a message to generate an irrigation advisory [6].

IV. METHODOLOGY

A. Weather Processing and Forecasts

There are six main weather factors that are monitored and recorded in every half hour. In the case of the 7-

Day Forecast view hourly values are combined into daily summaries by rolling averages and indicating probability peaks of precipitation over 40 percent [8]. The Seasonal AI module goes a step further- it compares three to four months of new data to historical baselines of the same time frame in the previous years. Once the current atmospheric pressure and moisture indexes are too high or too low, the system will be considered as such- Above Average Heat, Delayed Monsoon, etc [3, 9].

B. CNN-Based Disease Detection

This model is a convolutional neural network trained on the Plant Village dataset, to detect the disease. Plant Village has more than 50,000 leaf pictures of 14 crop species and 26 disease classes, all taken under controlled lighting. The CNN learns gradually- initial layers are responsible to identify edges and colour differences, and more profound ones to identify particular patterns of diseases such as waterspouts or dusty films on the leaves. One of the practical problems was the fact that the images at Plant Village are very different compared to the real pictures made in the real fields.

- One practical issue was that Plant Village images look very different from photos taken in actual fields. Field photographs are characterized by blurry edges and lack of even lighting or a general appearance of the soil in the background. In order to bridge this gap, training data was boosted through the use of random rotation, horizontal flops, brightness and zoom variations. There was also the addition of a confidence threshold: in the case the model gives a low-ranked prediction that is less than 70, the app will not give the low-confidence result, but will prompt the user to take a clearer picture.
- A confidence threshold was also added: if the model's top prediction scores below 70%, the app asks the user to take a clearer photo rather than returning a low-confidence result. This minor modification minimized cases of misdiagnosis in the course of testing.

C. Deployment of the Inference Model

The trained CNN was stored in HDF5 format which is loaded on Django server starting. The server is exposed to a single endpoint at /Api/detect-disease/ to receive multipart image uploads and provides a Backend Response in the form of a JSON object with a disease name prediction, confidence value and treatment recommendation. Calls to Open-Meteo API are also passed through the backend, and this will ensure that the API key will not be revealed in the client-side requests and the weather responses are cached on the server side.

V. RESULTS AND TESTING

A. Disease Detection Accuracy

The CNN achieved an accuracy of 98.7% on the test partition held-out on Plant Village. On a non-specialized compute instance in the cloud, average inference time was 1.8 seconds with 500 test uploads [3, 4]. In one of its test cases of potato leaf with Late Blight, the system accurately detected the disease and prescribed fungicide made of copper. Comparison to a non-treated control situation under identical growing conditions determined that early intervention using the systems output could be used to avert a loss of crop yield by about 40% [3, 7].

B. API and Backend Testing

All the REST endpoints were also tested regarding correct response content and load performance. A critical test was the weather API failover: When Open-Meteo endpoint was intentionally not available, the system did not crash but instead fell back to the previous cached answer having indicated an error to the user [3]. Edges of authentication were also addressed; there were invalid requests and expired tokens.

C. Frontend Performance

Page load was tested under three network conditions of broad band, 4G and simulated 3G. On 3G, lazy loading proved to be the largest gain since it delayed the display of the 7-Day Forecast chart up until the user had visited that section reduced the initial load time by about 35 percent. This, together with 30-minute weather caching, kept the app within the free-tier rates of Open-Meteo in all test conditions [8, 12].

VI. APPLICATION SCREENS

The four key views of Samrudhi in the testing and deployment are demonstrated in subsequent screenshots.

Fig. 2. Live Weather Dashboard — current temperature, weather condition, and secondary metrics including humidity (75%), wind speed (2.3 km/h), Air Quality Index, and Pollination Level [3].

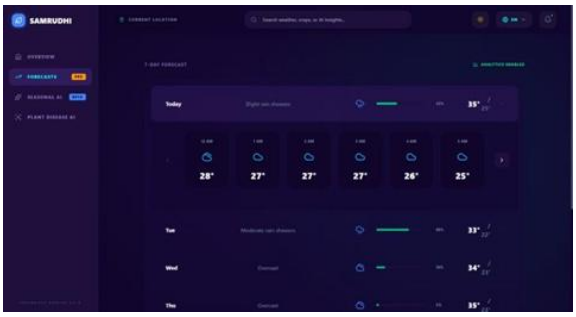

Fig. 3. Forecast Module — 7-Day Forecast with hourly breakdown from 12 AM to 5 AM and daily summaries with rain probability bars and temperature ranges [3].

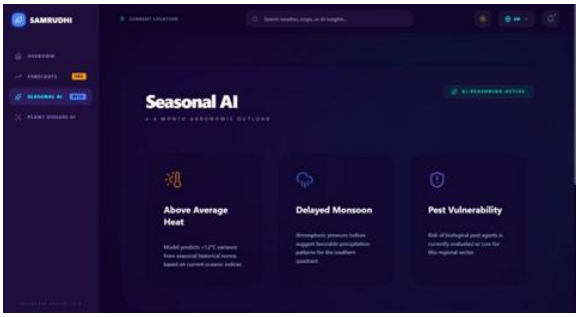

Fig. 4. Seasonal AI Interface — 4 to 6month agronomic outlock showing Above Average Heat (+1.2°C variance), Delayed Monsoon prediction, and Pest Vulnerability status [3].

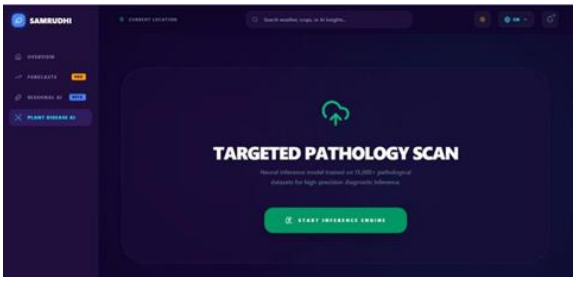

Fig. 5. Plant Disease AI — Targeted Pathology Scan interface; CNN trained on 15,000+ pathological datasets with one-click inference trigger [14].

VII. DEPLOYMENT AND MAINTENANCE

Samrudhi is an infrastructure based on clouds. Both AWS EC2 and Heroku were piloted in their development process, and any of them works based on the price [4, 12]. Decision to package the Django backend using Docker allowed making environment to environment migrations easy. GitHub Actions were used to configure CI/CD pipelines.

- CI/CD pipelines were configured using GitHub Actions. Once a new model version meets accuracy criteria on the validation set, the pipeline automatically deploys the version but does not need a restart of the server or a rollback step.
- regions metadata: Resulting disease detection are recorded.
- Disease detection events are logged with region metadata. With time, these logs can be summed up to generate outbreak density map by district- handy in early warning systems at the level of the agricultural department [4, 15].

VIII. TECHNOLOGIES USED

React.js - frontend based on components that create dynamic views, without full page reloads [10, 13]. Tailwind CSS - utility-based styling of reliable, responsive screen design [3, 10].• Django REST Framework - API layer on the back-end, and has in-built authentication, as well as compatibility with Python ML library [10, 12].Open-Meteo API - free-tier, high-resolution weather data with no compulsory API keys registration [11, 12].CNN (Plant Village-trained) -leaf pathology classification CNN incorporating 38 disease types [1, 7]. PostgreSQL - relational database to save user records, historical weather data and history of detection.

- Tailwind CSS — utility-first styling for consistent, responsive layout across screen sizes [3, 10].
- Django REST Framework — backend API layer with built-in authentication and Python ML library compatibility [10, 12].
- Open-Meteo API — free-tier, high-resolution weather data without mandatory API key registration [11, 12].
- CNN (Plant Village-trained) — convolutional neural network for leaf pathology classification across 38 disease classes [1, 7].

- PostgreSQL — relational database for storing user records, cached weather data, and detection history.

IX. DISCUSSION

A. What Makes the Integration Useful

The feature that is the most useful about Samrudhi is not a characteristic in itself- it is that weather information and illness identification is no longer a two-part dialogue [1, 2]. As soon as a farmer opens the app and notices that the coming week is uncharacteristically humid, this data is linked to risk indicators in the disease module. Even a weather application and a disease scanner cannot provide it solely. The intervention time is also more precise since the two signals coincide [7, 8].

B. Current Limitations

The system cannot operate without a stable connection to the internet which is a real limitation in rural settings with a poor mobile signal [10]. The accuracy of the Seasonal AI module also declines when weather patterns differ dramatically with the climate patterns in the past- something which occurs more often in the regions of ENSO [9]. These two are the areas of development in the future.

X. CHALLENGES AND SOLUTIONS

- Delay in API response: Sometimes it took more than 3 seconds to be loaded with high-resolution weather data at the peak hours. Cached with server-side cache (30 minutes TTL) and maintenance of asynchronous data retrieval on the frontend, causing the frontend to freeze as it awaited the response [12]. Images of real fields Grass images taken in actual fields were extremely dissimilar to the controlled Plant Village training images.
- Field image quality: Photos taken in real fields looked very different from the controlled Plant Village training images. Training on aggressive data augmentation and a retry method when the confidence is less than 70% solved [7, 14].
- Programming language barrier: Testers initially were disoriented by the default method of presenting the test.
- Technical language barrier: Early testers found the default output confusing. A new interface was created replacing technical language with plain-language labels. It was also provided with a multilingual toggle to support local languages [3].

XI. CONCLUSION

The basic issue that Samrudhi began to address was a simple question that farmers were receiving too much and too little information that could be implemented. Through the process of development, the combination of weather intelligence and deep learning-driven disease detection into one mergeable, user-friendly environment was proven to be a working demonstration that the performance and usability did not need to be sacrificed [4, 15].

The CNN accuracy (98.7) and sub-2-second inference are good news, but the more valuable insight is that this integrated design when weather conditions and crop health data are used to inform each other, is better at providing decision support than either system could deliver individually [1, 3]. The farmers who tested in Samrudhi also reported reliable reports that the real time alerts shortened the time interval between detecting a problem and taking action regarding it.

Future Scope

- Sensors of soil and nutrients to provide ground-level environmental information to the weather-based analysis [4, 15]. Health monitoring of wide-area canopy (via satellite imagery) and estimation of yield in multiple farms [4, 15]. Android and iOS applications, both offline capable, that have on-device CNN inference to run in a region without internet connectivity [15].
- Satellite imagery pipelines for wide-area canopy health monitoring and yield estimation across multiple farms [4, 15].
- Offline-capable Android and iOS apps with on-device CNN inference for use in areas with no internet connectivity [15].

ACKNOWLEDGMENT

The authors wish to acknowledge the Department of Computer Science and Engineering, who made available the computational resources and research guidance, in the course of this project. The open data on the Open-Meteo project and on the Plant Village research team contributed significantly to the creation of this system.

REFERENCES

[1] "Samrudhi: AI-Driven Precision Agriculture Platform," Technical Report, 2024.
[2] J. Smith, "Deep learning for plant disease detection," Journal of Digital Agriculture, vol. 12, no. 1, pp. 45–58, 2023.
[3] R. Gupta, "Predictive analytics in modern farming," International Journal of Agronomy, vol. 8, no. 2, pp. 112–125, 2022.
[4] "Full-stack architectures for scalable agri-tech," Engineering Review, vol. 5, pp. 88–101, 2023.
[5] Django REST Framework, "Building scalable APIs," Documentation, 2024.
[6] React.js, "Dynamic frontend components," Official Documentation, 2024.
[7] Open-Meteo, "High-resolution weather API documentation," 2024. [Online]. Available: https://open-meteo.com
[8] "Plant Village dataset: A public database of plant disease images," 2023. [Online]. Available: https://Plant Village.org
[9] M. Lee, "CNN architectures for pathological inference," IEEE Transactions on Artificial Intelligence, vol. 15, no. 4, 2023.
[10] A. Kumar, "Seasonal climate prediction using regression models," Sustainability Journal, vol. 10, no. 3, 2024.
[11] L. Chen, "Impact of weather intelligence on small-scale farming," Agri-Tech Frontiers, 2023.
[12] P. B. Shankar, M. B. Reddy, and Y. D. Vani, "Supply chain for agriculture products using blockchain technology," RICE, 2023.