Farmer X downloads my app.

Phase 1: First, he will select his preferred language (his own language). Then, the app language will switch to that particular language. This means if he selects Telugu, all the words and sentences will be in that language.

Phase 2: After selecting the language, he should enter an OTP because farmers can enter their phone number and not email and password, which would be difficult, you know. After entering the OTP (or auto-selecting from the messages app, which is common nowadays), following OTP validation, he should accept location permission so that we can get his live location. Then, the model should decide which crops can be cultivated here and which are not favorable. These should be mentioned on the home page only. If he changes the location, then he needs to go to settings and change it instead of asking every time, which becomes complex. If he changes location, then again all the favorable and unfavorable crops should be mentioned accordingly and respectively. He should be taken to the home page with an icon on the right-hand side, like a dashboard, where he can see the schemes applied, previous history, etc.

Phase 3: In the home page, first a scanner will be there using which he can upload a picture. After uploading the picture, the model should detect the disease and take us to the disease diagnosis or report of that disease. In that report, we should mention the disease, severity (if possible), and provide some pesticide companies who can solve the problem—like people who will come and cure the disease, you know. We should also mention the pesticides that can cure the disease along with contacts of pesticide cleaning people. If possible, mention the reason for the disease and precautions that should be avoided next time. If it is due to soil, then the model should mention it and also the crops that can be cultivated there. All these should be in text, and on the side, there should be a loudspeaker symbol. When he clicks it, he can listen to the text instead of reading (some people can read, some people can listen).

Phase 4: In the home page, below the scanner, we should add a microphone where instead of uploading a picture, he can speak to it. Then, using his audio, the model will detect the disease, and we should get the same report that is there for photo upload. That means the model for this should be that much efficient.

Phase 5: The next section in the home page is schemes available. If he clicks that, he should be taken to that page—the scheme application page—where a list of all the government scheme applications with the least deadline (like which is going to end soon) should appear first, from least to more time basically. Here, all the schemes that he can apply—we will provide only links to those official scheme pages, as we can't keep scheme applications here, you know—only links.

Phase 6: In the home page only, we will add a navbar. In that navbar, we will add emergency assistance. We will use a weather API to forecast weather and caution the farmer if there is any cyclone, heavy rains, etc., beforehand. The app should notify the user if there are any sudden hazards like cyclones in that particular area, just like how we get notifications saying "heavy cyclone and thunderstorm may strike in your area." It should mention this in the emergency assistance page too. We should also mention the precautions that he should take and helpline numbers for help.

🔷 OVERALL CORE STACK

📱 Mobile App

- **React Native (Expo)**
- React Navigation
- Axios
- AsyncStorage
- i18n (react-i18next) for multilingual UI

🧠 Backend

- **Python + FastAPI**
- Uvicorn
- MongoDB
- FAISS (Vector DB)
- Redis (optional caching)

☁ Deployment

- Local GPU / Laptop for hackathon
- AWS / Azure for scalability (optional mention)

🔷 PHASE 1 — LANGUAGE SELECTION (UI TRANSLATION)

Task:

Translate entire app dynamically to Telugu / Hindi / Kannada etc.

Models / Tools:

1. **react-i18next**
   - Static translation JSON files
   - Best for UI language switching
2. **Meta NLLB-200 (Pretrained Translation Model)**
   - Neural Machine Translation
   - Supports Indian languages
3. **Google Translate API**
   - Cloud-based translation

✅ BEST SUGGESTION:

Use **react-i18next + Pre-translated JSON files**
(Stable + Offline + Fast)

🔹 PHASE 2 — OTP LOGIN + LOCATION INTELLIGENCE

OTP Authentication

1. **Firebase Authentication (Phone OTP)**
2. **Twilio SMS OTP API**
3. **AWS SNS OTP**

✅ BEST SUGGESTION:

Firebase Authentication (Easy + Reliable)

Location-based Crop Suitability

You need:

1. GPS coordinates
2. Soil + climate mapping
3. Crop suitability prediction

Models for Crop Suitability

1. **Random Forest Classifier**
   - Based on soil, rainfall, temperature
2. **XGBoost**
   - Better accuracy
3. **Logistic Regression (baseline)**

Pretrained Model?

No pretrained global model — must train using dataset.

✅ BEST SUGGESTION:

Random Forest (Fast + Good accuracy + Simple)

🔹 PHASE 3 — IMAGE DISEASE DETECTION

This is core AI.

Model Options:

1. **ResNet50 (Pretrained on ImageNet)**
   - Fine-tuned CNN
2. **EfficientNet-B0**
   - Lightweight + High accuracy
3. **MobileNetV2**
   - Optimized for mobile
4. **DenseNet121**
   - Strong feature reuse

✅ BEST SUGGESTION:

EfficientNet-B0
(High accuracy + Mobile friendly)

Severity Detection

Options:

1. Image-based regression model
2. Use classification confidence score
3. Separate severity classifier

Best:

Use confidence score + rule-based severity

🔷 PHASE 4 — VOICE-BASED DIAGNOSIS

Speech-to-Text

1. **Whisper Small**
2. Whisper.cpp
3. Google Speech API
4. Vosk (offline STT)

✅ BEST:

Whisper Small

Intent Detection

1. BERT (fine-tuned)
2. DistilBERT
3. SentenceTransformers + Cosine similarity
4. RAG (Retrieval Augmented Generation)

✅ BEST:

SentenceTransformers + FAISS (RAG hybrid)

🔹 PHASE 5 — GOVERNMENT SCHEME SORTING

Task:
Sort by nearest deadline.

No heavy AI needed.

Models:

- Rule-based sorting
- Simple filtering logic

Optional:

- NLP intent detection (same as Phase 4)

🔹 PHASE 6 — WEATHER + EMERGENCY SYSTEM

Weather API

1. OpenWeatherMap
2. WeatherAPI.com
3. IMD Open Data

✅ BEST:

OpenWeatherMap

Cyclone Prediction Alerts

Use:

- Weather alerts API endpoint
- Hazard feed parsing

GIS Location Intelligence

1. Google Maps API
2. OpenStreetMap
3. Mapbox

✅ BEST:

Google Maps API

✅ PART 2 — DATASETS (PHASE-WISE WITH LINKS)

🔹 PHASE 2 — Crop Suitability Dataset

1️⃣Crop Recommendation Dataset (Kaggle)

https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset

Contains:

- Nitrogen
- Phosphorus
- Potassium
- Temperature
- Humidity
- Rainfall
- pH

✅ BEST DATASET:

Crop Recommendation Dataset (Above)

🔷 PHASE 3 — Plant Disease Dataset

1️⃣ PlantVillage Dataset

https://www.kaggle.com/datasets/emmarex/plantdisease

38 classes.

2️⃣ Rice Disease Dataset

https://www.kaggle.com/datasets/minhhuy2810/rice-diseases-image-dataset

---

3️⃣ PlantDoc Dataset

https://www.kaggle.com/datasets/pratikkayal/plantdoc-dataset

✅ BEST:

PlantVillage + Rice Disease Dataset combined.

🔷 PHASE 4 — Voice Model Dataset

Whisper is pretrained.
No dataset needed.

If training intent:

1️⃣ Indic NLP Dataset

https://indicnlp.ai4bharat.org/

2️⃣ Indian Language Speech Dataset

https://openslr.org/

🔹 PHASE 5 — Government Scheme Data

Official Portals:

https://www.india.gov.in
https://pmkisan.gov.in
https://data.gov.in

You must manually scrape / curate.

🔹 PHASE 6 — Weather Data

https://openweathermap.org/api

New Features:

1) in dashboard he should get schemes applied, if possible status

2) in disease report we should mention pesticides needed to cure the disease along with contacts of pesticides cleaning ppl and if possible reason for the disease and precautions that should be avoided next time
If it is due to soil then model should mention it and also the crops that can be cultivated there

3) after accepting location model should decide which crops can be cultivated here and which are not favarable and these should be mentioned in home page only, if he changes the location then he need to go to settings and change instead of asking everytime and it becomes complex and if he changes location then agin all the favourable and unfavorable crops hould be mentioned accordingly and respectively

4) we will add navbar in home page there again we will mention home, scheme application,

✅ PART 3 — WHAT IS MISSING ACCORDING TO JUDGING CRITERIA

Important.

🔹 1. Business Understanding (15%)

Missing:

- Market size analysis
- Number of farmers affected
- Economic loss data
- Rural digital penetration stats

🔹 2. Scalability (10%)

Missing:

- Load balancing architecture
- Database indexing strategy
- Cloud deployment design

🔹 3. Security (10%)

Missing:

- Data encryption strategy
- Secure API tokens

- Rate limiting
- Location privacy protection

🔹 4. Innovation (20%)

Currently innovation = integration

To increase:

- Risk scoring model
- Village-level outbreak heatmap
- AI confidence meter

🔹 5. Implementation Feasibility (10%)

Need:

- Hardware requirements
- RAM usage estimation
- Model size documentation

✅ PART 4 — INNOVATIVE ADDITIONS (STICKING TO PROBLEM STATEMENT)

🌾1️⃣Disease Outbreak Map

If 10 farmers detect same disease → show heatmap.

Very innovative.

🌦️2️⃣Weather + Disease Correlation Warning

If humidity high + rice → fungal alert.

Predictive system.

📊3️⃣Crop Health Score

0–100 score.

🧠4️⃣AI Confidence Meter

“Model Confidence: 92%”

Adds trust.

🏥5️⃣Emergency Smart Routing

Auto-route to nearest:

- Hospital
- Agriculture officer
- Disaster relief

📶6️⃣Low Bandwidth Mode

Auto compress images before sending.

🚀 FINAL TECH SUMMARY (BEST STACK)

Component	Best Model
Image Model	EfficientNet-B0
Crop Suitability	Random Forest
Speech-to-Text	Whisper Small
Intent	SentenceTransformers
Vector DB	FAISS
Weather	OpenWeather
OTP	Firebase