EPICS Project :- "SmartAgro: A disease detection model with Human Interaction"

Introduction:

Hello everyone very warm greetings to all, as you all knows we have EPICS Project and, in this project, we have to develop a Web application which can detect the Diseases of plants and crops with a human like interaction and I have named it "SmartAgro". I have taken the liberty of compiling this word file which incorporate all the important things that is needed in order to complete our project successfully and smoothly.

Please go through Description, and understand the basic work that we have to perform and based on that please select a part that fits you well, and this selection is first come first serve, so whichever part you like just mention it in the group. And if you have any problem with the name, please suggest your own name in the Group.

Description:

"SmartAgro" is an AI-powered system that helps farmers detect plant diseases and get remedies through both image and text inputs. A farmer can **log in** and either **upload a photo or enter a text query**. The system then processes the input using **CNN/Random Forest models** (for images) or **NLP** (for text) to identify the disease. Remedies are first checked in the **local database**, with fallback to **Wikipedia/ChatGPT** if no match is found. The result is then displayed on a **user-friendly web interface**, along with a **history of recent queries** for easy reference.

SmartAgro – System Flow Overview: (for better understanding please refer the flowchart)

1) **User Login/Registration** – Farmer logs into the system.
2) **Input Choice** – User submits either:
   a) **Image** (plant leaf photo) → processed by CNN/Random Forest, or
   b) **Text** (symptoms/crop info) → processed by NLP pipeline.
3) **Disease Identification** – System predicts the plant disease.
4) **Remedy Retrieval** – Remedies are fetched from the **local database**; if not found, fallback to **Wikipedia/ChatGPT**.
5) **Result Display** – Final disease info + remedies shown on the **Result Page**.
6) **History Tracking** – Last 10 queries stored and accessible on the **History Page**.

(Most Important)

Team Role Assignment:

🟦 1. NLP Developer(main) + Pipeline Integrator(helper)

Tasks:

- Implement nlp_process.py for extracting crop/disease symptoms from text input.
- (Helper) Design the query pipeline: Text → NLP → Database → ChatGPT fallback (refer to UML decision block *"Is it related to Plant?"*).
- Provide deliverables: NLP code, pipeline diagram, and short documentation.
  References: NLP in agriculture【Plant_Disease_Detection_and_Classification_by_Deep_LearningA_Review.pdf】

🟩 2. ML Developer 1 – CNN Model Developer

Tasks:

- Train a CNN on the PlantVillage dataset for image-based plant disease detection.
- Save the trained model as cnn_model.h5 and evaluate test accuracy.
- Provide documentation of training process, hyperparameters, and results.
  References: CNN achieving 99% accuracy on PlantVillage【applsci-12-06982-v2.pdf】

🟨 3. ML Developer 2 – Random Forest Model Developer

Tasks:

- Train a Random Forest classifier on pre-processed image features (flattened or CNN-extracted).
- Save model as rf_model.pkl and implement rf_predict.py.
- Compare metrics with CNN (accuracy, confusion matrix).
  References: Comparative studies of ML vs DL for plant disease detection【plants-08-00468-v3.pdf】

🟧 4. Backend Developer – Flask + Model Selector+ Pipeline Integrator

Tasks:

- Develop Flask app (routes.py) with endpoints for:
  - Text input → NLP
  - Image input → CNN/Random Forest (based on toggle)
- Integrate final result handler combining disease prediction + remedy retrieval.

- Design the query pipeline: Text → NLP → Database → ChatGPT fallback (refer to UML decision block *"Is it related to Plant?"*).
- Integration of NLP + CNN + RF + DB into one pipeline.
- Provide documentation of backend architecture.
  References: UML flow "Use Model → Fetch Remedies → Show Result"

🟥 5. Frontend Developer – UI/UX

Tasks:

- Create all HTML pages (Login, Dashboard, Query Input, Result Page, History).
- Add model toggle (CNN vs RF) on the dashboard.
- Ensure responsive design using CSS/Bootstrap.
- Provide screenshots and flowchart of navigation.
  References: UML diagrams – User login, dashboard, and history flow

---

🟫 6. Database Engineer + Knowledge Base Fetcher

Tasks:

- Build disease information database (disease_data.csv).
- Implement data_lookup.py for local search + Wikipedia fallback.
- Implement query history management (store last 10 queries).
- Document schema and provide screenshots of DB operations.
  References: Studies on integrating ML with expert knowledge bases【fpls-14-1158933.pdf】

Note:

If you have any query or any kind of problems, please feel free to discuss in the group or contact me or anyone via phone. And I have also attached the Important links word file which consists the relevant links for the project, please refer to them. And I will also send the reference research paper after 2-3 days when all the selection have been completed.

Thank you Friends,

Best regard,

Tarman Singh Sohal