**Empowering Ethiopian Coffee Farmers with AI-Powered Leaf Disease Detection**

**1. Introduction**

In Ethiopia, coffee is more than a crop it’s a cultural cornerstone and a vital economic lifeline for millions of smallholder farmers. Yet, diseases like Rust, Phoma, and Miner threaten yields, with losses sometimes reaching 30-50% of harvests. These challenges, compounded by limited access to expert agricultural guidance, jeopardize farmers’ livelihoods and Ethiopia’s position as a global coffee leader. Our project tackles this problem head-on with a mobile app that uses artificial intelligence (AI) to detect coffee leaf diseases in real-time, empowering farmers to act swiftly and protect their crops. This is one step ahead of previous works which just focused on creating Machine Learning models to classify the diseases.

This initiative aligns with the United Nations Sustainable Development Goals (SDGs): SDG 2 (Zero Hunger) by reducing crop losses, SDG 8 (Decent Work and Economic Growth) by boosting farmers’ incomes, and SDG 13 (Climate Action) by promoting resilient agricultural practices. Imagine a farmer in a remote Ethiopian village scanning a coffee leaf with their phone and instantly receiving a diagnosis and actionable advice no experts, no delays, just results. This blog dives into how we built this solution, its impact, and its potential to transform agriculture.

**2. The Problem: A Threat to Coffee and Livelihoods**

Ethiopia, the birthplace of Arabica coffee, produces some of the world’s finest beans. However, leaf diseases like Rust (yellow-orange lesions), Phoma (tip die-off), and Miner (winding tunnels) spread rapidly, devastating crops. Smallholder farmers, who account for over 90% of Ethiopia’s coffee production, often lack access to timely diagnostics or expert advice. Traditional methods, such as manual inspections by agronomists or pathologists are neither scalable nor practical in remote areas. As a result, farmers face significant yield losses, threatening their income and food security.

The need for an accessible, real-time solution is clear. Our project leverages AI and mobile technology to bridge this gap, offering farmers a tool to diagnose diseases instantly using their smartphones, even offline. By addressing this challenge, we aim to enhance agricultural productivity and support sustainable development.

**3. The Process: Building the Solution**

**Methodology**

**Data Collection**

We sourced a diverse dataset of approximately 73,000 images from public repositories. First we had used the JMuBEN and JMuBEN2 datasets, which come preprocessed and cropped to only show the area of the leaf that shows the diseases. We trained our initial model with these images but noticed that the model, even though does a good job in classifying zoomed in pictures, it was very bad at classifying whole pictures. In addition, we thought it was impractical to always scan the leaves up close. Hence, we added 3 more data sets with whole leaf images, all while ensuring the use of only Arabica coffee leaf images as that is the coffee plant found in Ethiopia. We used the ETHIOPIAN COFFEE LEAF DISEASE, Coffee leaf diseases, and Disease and pest in coffee leaves datasets from Kaggle. After retraining our model, we faced another challenge. The model couldn’t classify non-leaf images resulting in classifying random scans as leaves. Due to this reason we added our final set of images, a randomly generated colorful noise images.

A collage of green leaves

AI-generated content may be incorrect.

**Model Development**

We implemented a transfer learning-based image classification model using TensorFlow and Keras. We initially designed a custom CNN from scratch but later agreed that utilizing transfer learning would be much better as it would result in better accuracy and generalization. We aimed to leverage pretrained architectures, namely: ResNet50, VGG16, InceptionV3, EfficientNetB0 and MobileNetV2, but were able to only train MobileNetV2 due to lack of sufficient computing resources. The other models couldn’t be trained within the maximum runtime that Google Colab allows which is 12 hours.

**Training setup included:**

* Pretrained Backbone: MobileNetV2 (with frozen base weights).
* Custom Classification Head: GlobalAveragePooling, Dense(128, ReLU), Dropout(0.5), and Dense(softmax) for multiclass classification.
* Image Input Size: 224×224 pixels.
* Loss Function: Categorical Crossentropy.
* Optimizer: Adam (default learning rate).
* Batch Size: 32.
* Epochs: Up to 7, with early stopping and model checkpointing.
* Class Weights: Automatically computed to handle class imbalance.
* Data Augmentation: Applied with Keras ImageDataGenerator (rotation, zoom, shifts, flips).

# Function to create model

def create\_model(base\_model, num\_classes):

    x = base\_model.output

    x = GlobalAveragePooling2D()(x)

    x = Dense(128, activation='relu')(x)

    x = Dropout(0.5)(x)

    predictions = Dense(num\_classes, activation='softmax')(x)

    model = Model(inputs=base\_model.input, outputs=predictions)

    model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

    return model

# Pretrained models

models = {

    'ResNet50': ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3)),

    'VGG16': VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3)),

    'InceptionV3': InceptionV3(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3)),

    'EfficientNetB0': EfficientNetB0(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3)),

    'MobileNetV2': MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3)),

}

# Dictionary to store training history

history\_dict = {}

# Train each model

for name, base\_model in models.items():

    print(f"\n📦 Training {name}...")

    base\_model.trainable = False  # Freeze base

    model = create\_model(base\_model, num\_classes=train\_data.num\_classes)

    # Callbacks

    early\_stop = EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True, verbose=1)

    checkpoint = ModelCheckpoint(

        filepath=f'/content/drive/MyDrive/best\_model\_{name}t4.h5',

        monitor='val\_loss',

        save\_best\_only=True,

        verbose=1

    )

    # Fit model

    history = model.fit(

        train\_data,

        validation\_data=val\_data,

        epochs=7,

        callbacks=[early\_stop, checkpoint],

        class\_weight=class\_weights,

        verbose=1

    )

    history\_dict[name] = history

# Plot Accuracy and Loss

plt.figure(figsize=(14, 6))

# Accuracy

plt.subplot(1, 2, 1)

for name, history in history\_dict.items():

    plt.plot(history.history['accuracy'], label=f'{name} - Train')

    plt.plot(history.history['val\_accuracy'], label=f'{name} - Val')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

# Loss

plt.subplot(1, 2, 2)

for name, history in history\_dict.items():

    plt.plot(history.history['loss'], label=f'{name} - Train')

    plt.plot(history.history['val\_loss'], label=f'{name} - Val')

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.tight\_layout()

plt.show()

**Evaluation:**

* The model was tested on a separate dataset using a confidence threshold (0.8) to reject uncertain predictions.
* We achieved an overall test accuracy of approximately 95–98%, depending on the dataset and class distribution.
* Additional evaluation was done using confusion matrices and classification reports to assess per-class performance.

A graph with numbers and a chart

AI-generated content may be incorrect.

precision recall f1-score support

Cerscospora 0.91 0.96 0.93 1027

Healthy 0.99 0.98 0.99 2167

Leaf rust 0.92 0.98 0.95 1139

Miner 0.95 0.99 0.97 1760

Phoma 0.99 0.79 0.88 935

unknown\_images 1.00 1.00 1.00 497

accuracy 0.96 7525

macro avg 0.96 0.95 0.95 7525

weighted avg 0.96 0.96 0.96 7525

**The Application**

**A cartoon of a person holding a staff

AI-generated content may be incorrect.**

**Meet Keffa, our AI-Powered Coffee Leaf Disease Detection app. Keffa is a region found in Ethiopia, which is believed to be where coffee was first discovered. Our application’s logo is a young Ethiopian shepherd.**

**Intergration**

To make the model accessible, we converted it to **TensorFlow Lite (TFLite)** format, reducing size and latency for mobile deployment. The app, built with **Flutter**, supports Android, ensuring broad accessibility. The tflite\_flutter plugin enables on-device inference, allowing farmers to scan leaves without internet access. The implementation code snippet below illustrates model loading:

\_interpreter = await Interpreter.fromAsset(

'assets/models/plant\_disease\_model.tflite',

options: InterpreterOptions().threads = 4,

);

\_interpreter!.run(reshapedBuffer, outputBuffer);

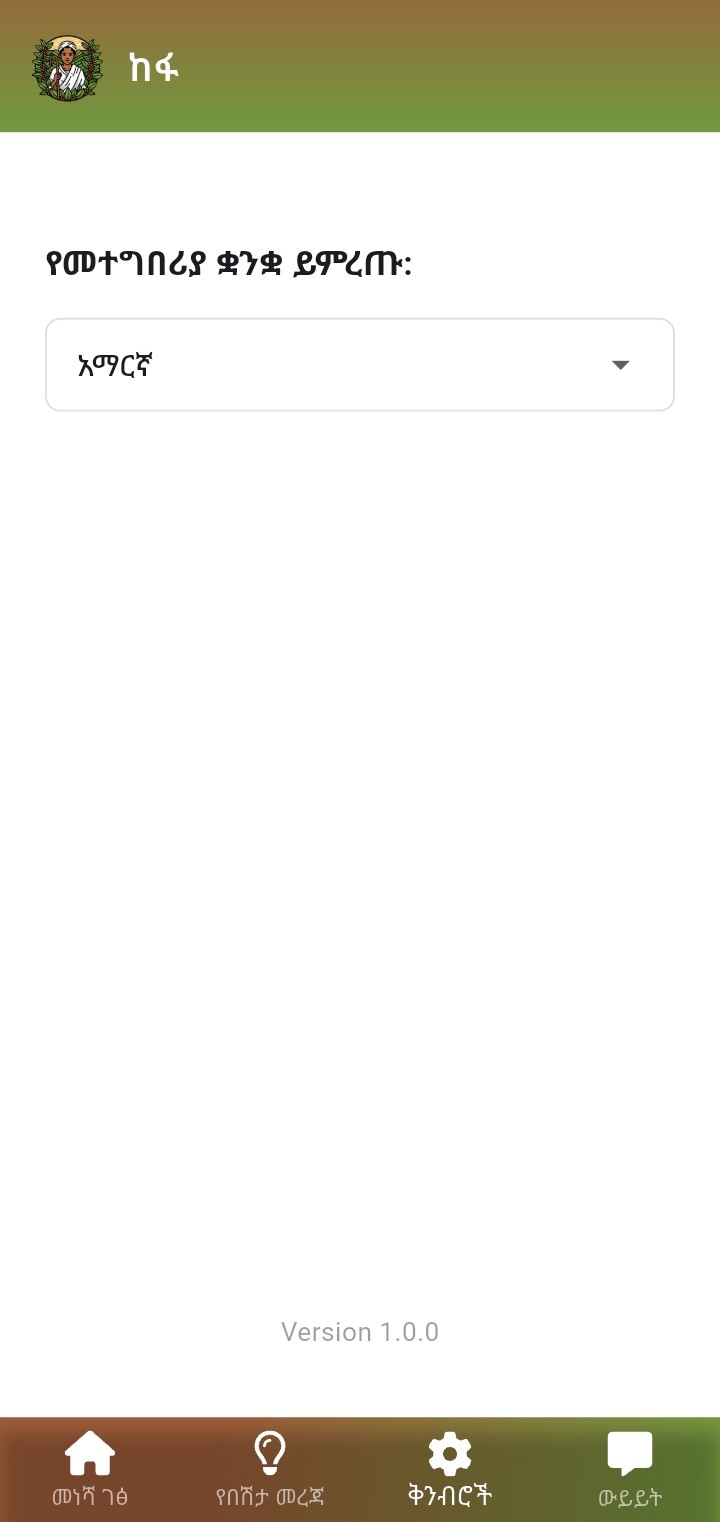
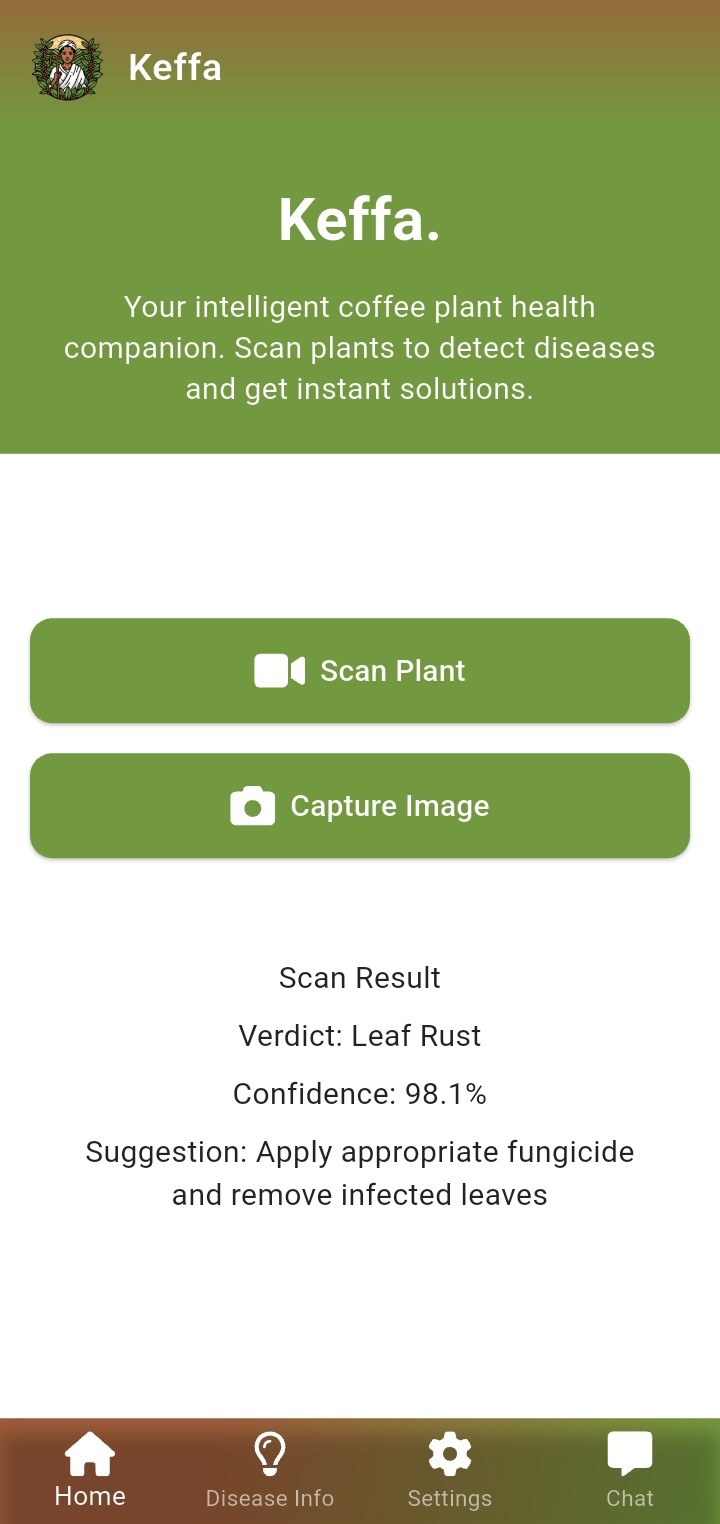
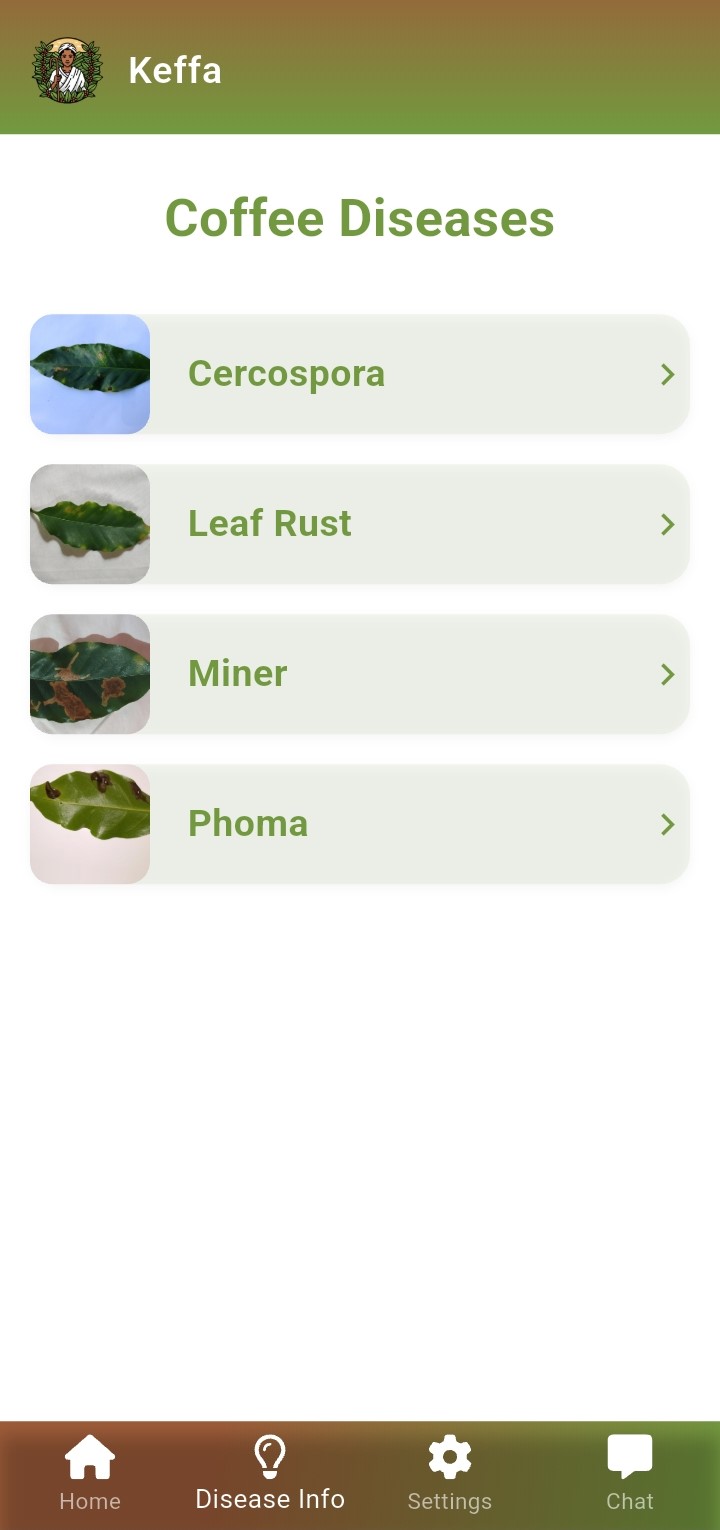
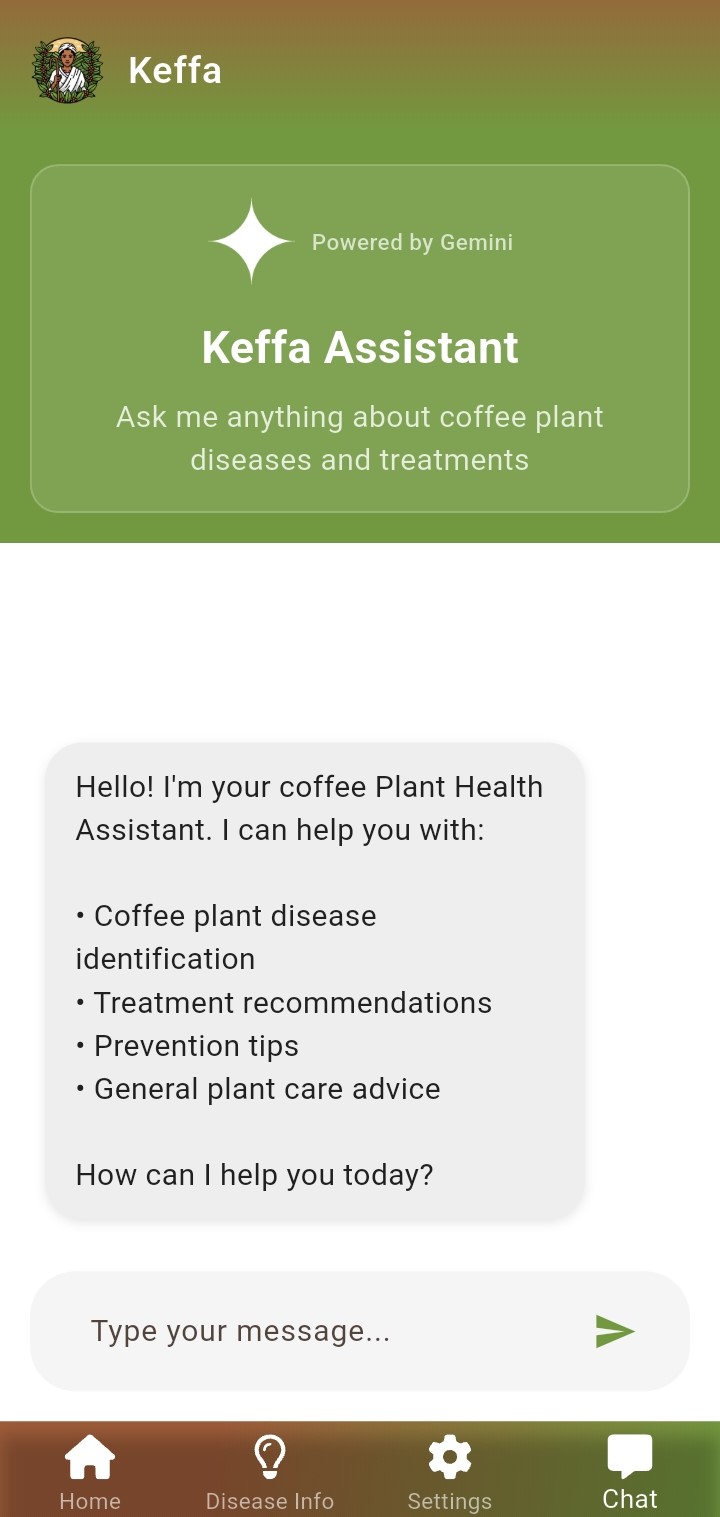
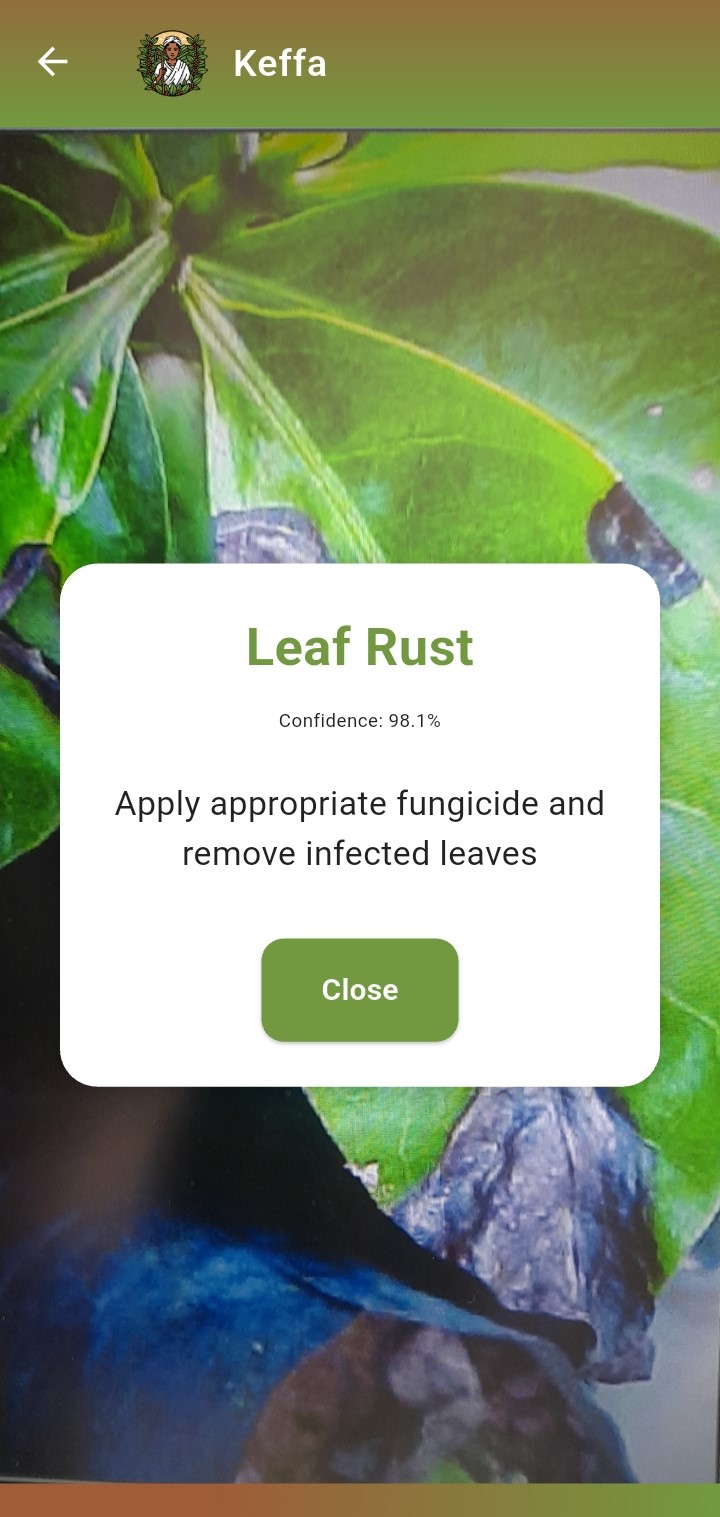
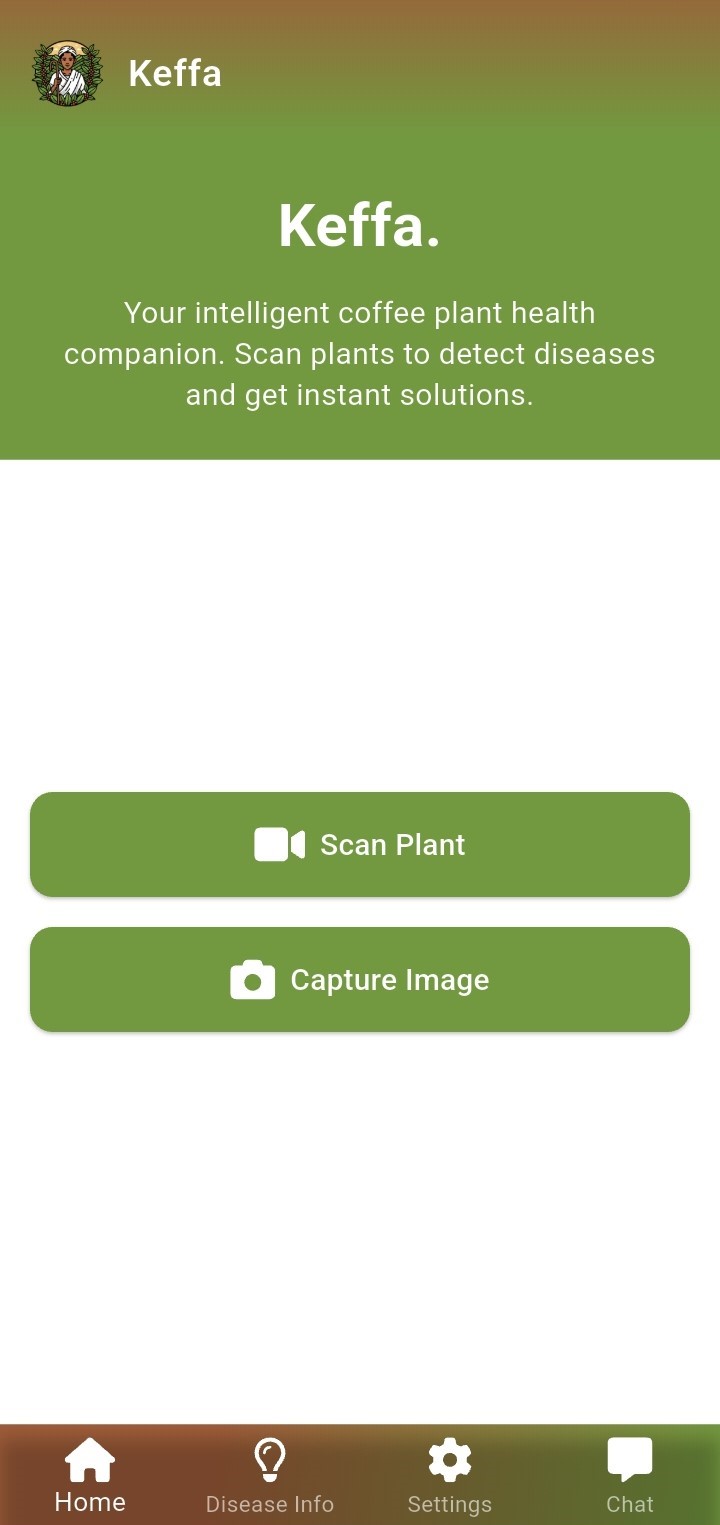
**Implementation**

The app’s key features include:

* **Real-time Detection**: Scans leaves via the phone’s camera, delivering diagnoses in under a second.
* **Image Support**: Users can also take pictures and get diagnosis results
* **Offline Functionality**: Operates without internet, critical for rural areas.
* **Multilingual Support**: Offers text outputs in English and Amharic. Adding all 5 of Ethiopia’s official languages was planned, but only Amharic could be added.
* **Treatment aid**: Provides treatment suggestions for all the disease types. If internet access is available, an AI chatbot using Gemini’s API is available to undergo interactive treatment.

The user interface, designed for simplicity, includes:

* **Home Screen**: Scanning of the leaves take place here.
* **Treatment Aid:** Detailed disease descriptions and treatment recommendations are available immediately on the platform.
* **Settings Screen**: Here language can be changed.
* **Chatbot**: Provides guidance on disease treatment.



**4. Discussion**

**Reflections**

The project’s success lies in its blend of advanced AI and user-centric design. The CNN’s high accuracy reflects the robustness of our dataset and training strategy. Offline functionality and multilingual support make the app accessible to Ethiopia’s diverse, often remote farming communities. However, some limitations still exist including:

* **Leaves with 2 diseases**: Scanning the leaves result in detection of one disease, while in real life scenarios we see that leaves contain more than 2 diseases.
* **Environmental Factors**: The model does not yet account for stressors like drought or nutrient deficiencies. We couldn’t find datasets that account for that. Even though meta data was tried to use to infer that, it was unsuccessful.

**Future Improvements**

* **Use of YOLOv8 for object detection**: This is a method that can be used to solve the problem of not being able to detect multiple diseases in one leaf.
* **Seasonal Analysis**: Incorporate temporal data to model disease patterns across seasons.
* **Prescriptive Analytics**: Add severity assessments
* **Interactive Tutorials**: Guide users on capturing high-quality images to improve diagnosis accuracy.
* **Feedback Loop**: Enable farmers to rate predictions, refining the model over time.

**6. Conclusion**

Our AI-powered coffee leaf disease detection app is a game-changer for Ethiopian farmers, offering a scalable, accessible tool to combat crop losses. By integrating CNNs with a user-friendly Flutter app, we’ve created a solution that empowers farmers, boosts productivity, and supports sustainable agriculture. This project not only addresses immediate agricultural challenges but also sets a foundation for broader applications in crop health monitoring.

**7. Recommendations**

* **Researchers**: Make sure your data set is diverse to ensure real world usage. Explore YOLOv8 or ensemble models for enhanced detection. Use Transfer Learning for better accuracy and generalization. In order to prevent the model from classifying non-leaf images as leafs, add an additional non leaf category and increase the probability threshold.
* **Developers**: Prioritize offline-first designs and multilingual interfaces for rural users.
* **Policymakers**: Support digital infrastructure to scale such solutions across Africa.

By fostering collaboration and innovation, we can extend this technology to other crops and regions, driving global agricultural resilience.

## 8. References

* Taye, B. G., & Goel, N. (2024). Coffee Leaf Plant Disease Identification. Research Square. <https://doi.org/10.21203/rs.3.rs-3720115/v1>
* Abdalla, A., et al. (2023). An Advanced Deep Learning Models-Based Plant Disease Detection. Frontiers in Plant Science. <https://doi.org/10.3389/fpls.2023.1158933>
* Kaggle. (n.d.). Coffee Leaf Diseases Dataset. <https://www.kaggle.com/datasets/badasstechie/coffee-leaf-diseases>
* Jepkoecha, J., et al. (2021). Arabica Coffee Leaf Images Dataset. Mendeley Data. <https://data.mendeley.com/datasets/tgv3zb82nd/1>
* Cerda, R., et al. (2017). Effects of Shade, Altitude and Management on Multiple Ecosystem Services in Coffee Agroecosystems. European Journal of Agronomy, 93, 69-78. <https://doi.org/10.1016/j.eja.2017.11.007>
* Kumulachew, A., et al. (2016). Review on Coffee Production and Constraints in Ethiopia. Journal of Food, Agriculture & Environment, 14(2), 13-18.
* World Bank. (2021). Ethiopia’s Coffee Sector: A Pathway to Prosperity. <https://www.worldbank.org/en/news/feature/2021/04/20/ethiopian-farmers-triple-coffee-yields-with-sustainable-tree-stumping>
* Akundi, P. (2023, March 28). Unknown-unknowns: How to train a CNN that’s ready for anything with softmax thresholding. Medium. <https://medium.com/@prathyuakundi/unknown-unknowns-how-to-train-a-cnn-thats-ready-for-anything-with-softmax-thresholding-20cba0496990>
* Ketiwe Kaluma. (2021). Data set on Outcomes of Participatory Fisheries Management in Zambia's Mweru- Luapula Fishery. Article S2352340921004261. <https://doi.org/10.1016/j.dib.2021.107161>
* Yoseph, B. (n.d.). Ethiopian coffee leaf disease. Kaggle. <https://www.kaggle.com/datasets/biniyamyoseph/ethiopian-coffee-leaf-disease>
* AlvaroLE. (n.d.). Coffee leaves disease. Kaggle. <https://www.kaggle.com/datasets/alvarole/coffee-leaves-disease>
* Jepkoech, jennifer; Kenduiywo, Benson; Mugo, David; Chebet, Edna (2021), Coffee leaf disease dataset. Mendeley Data. <https://data.mendeley.com/datasets/t2r6rszp5c/1>
* Bayile Getu Taye; Neeraj Goel. (2024). Coffee leaf plant disease identification through image processing and machine-learning techniques in Ethiopia. ResearchGate. https://www.researchgate.net/publication/377181128\_Coffee\_Leaf\_Plant\_Disease\_Identification\_through\_Image\_Processing\_and\_Machine-Learning\_Techniques\_in\_Ethiopia