Capstone Project Concept Note and Implementation Plan

Project Title: AI-Powered Coffee Leaf Disease Detection

Team Members

1. Abdulkerim Issa Osman
2. Luel Fikadu
3. Rabiya Abdulnassir

Concept Note

1. Project Overview

- This project aims to develop a mobile app that helps Ethiopian coffee farmers detect leaf diseases in real-time by scanning leaves with their phones.

- It supports SDG 2 (Zero Hunger) by boosting yields, SDG 13 (Climate Action) by enhancing resilience, and SDG 8 (Decent Work and Economic Growth) by improving farmers' income.

2. Objectives

- Develop a user-friendly mobile application that uses image analysis to detect coffee leaf diseases in real-time.
- Empower Ethiopian coffee farmers with accessible technology to identify and manage plant diseases early.
- Reduce crop losses by enabling timely and accurate disease diagnosis.
- Improve productivity and income by supporting healthier coffee crops.
- Contribute to sustainable agriculture through smart farming solutions that align with global development goals.
- This project addresses the lack of timely disease detection among farmers by offering a practical, tech-driven tool that enhances decision-making and boosts agricultural outcomes.

3. Background

Ethiopia is renowned as the birthplace of coffee and remains one of the world's leading producers. However, many smallholder coffee farmers face significant challenges in maintaining healthy crops due to limited access to agricultural training and expert guidance. One major issue is the inability to accurately and promptly identify coffee leaf diseases, such as the Rust, Miner or Phoma Disease. These diseases can spread rapidly and cause severe crop losses if not detected early.

Existing solutions often rely on manual inspection by experts or agricultural extension services, which are not always accessible or scalable in rural areas. Printed guides or general awareness campaigns may help but lack the precision and speed needed for timely action.

A machine learning-based mobile application offers a powerful solution. By using computer vision models trained on thousands of diseased and healthy leaf images, farmers can instantly identify issues with a simple scan. This approach is scalable, cost-effective, and provides real-time, location-independent support, empowering farmers to make informed decisions and protect their livelihood.

4. Methodology

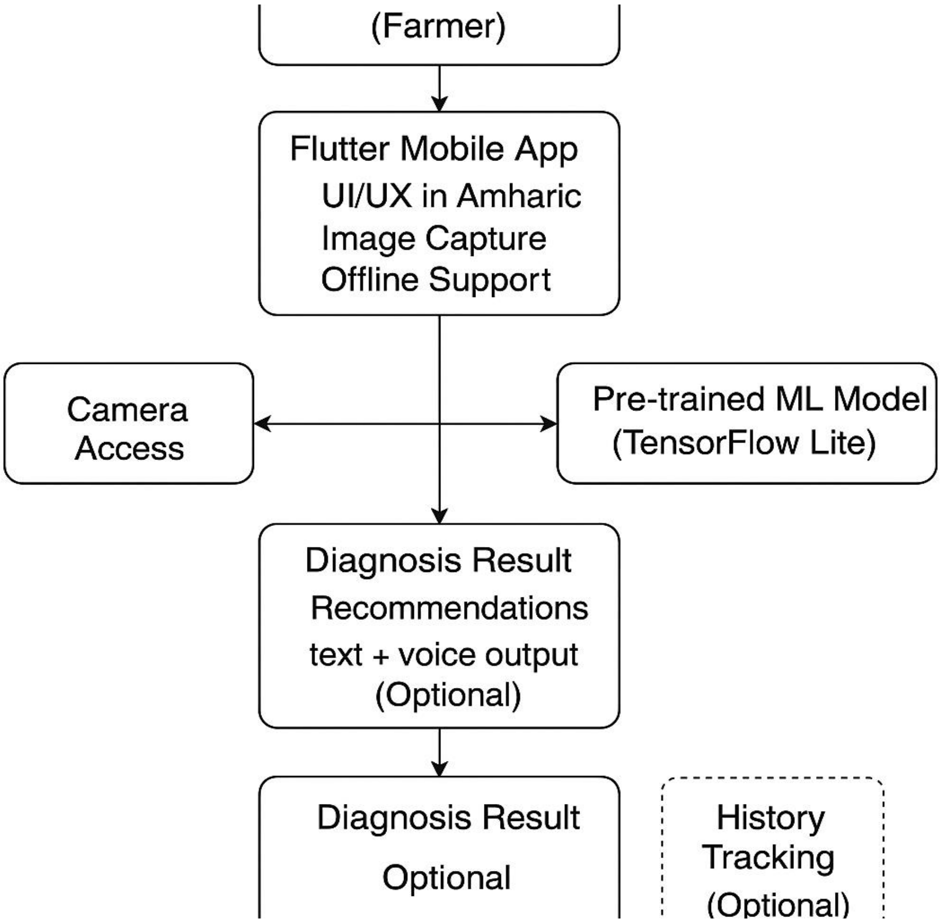
For our coffee leaf disease classification project, we will employ a robust deep learning approach leveraging Convolutional Neural Networks (CNNs), a state-of-the-art technique for image analysis. Our methodology will involve constructing a CNN architecture using the modular and efficient capabilities provided by either the PyTorch or TensorFlow/Keras framework. This will allow us to define a network composed of convolutional layers for automated feature extraction from the coffee leaf images, followed by pooling layers to enhance spatial invariance and reduce dimensionality. Reshape layers will facilitate the transition to fully connected (dense) layers, which will ultimately perform the classification into healthy or disease categories.

Crucially, we will utilize the data loading and preprocessing utilities offered by our chosen framework (e.g., \`torch.utils.data.DataLoader\` and \`torchvision.transforms\` in PyTorch, or \`tf.data.Dataset\` and \`tf.keras.preprocessing.image.ImageDataGenerator\` in TensorFlow/Keras). This will enable us to efficiently load our dataset of upto 65,000 images, perform essential preprocessing steps such as resizing and normalization, and implement effective data augmentation techniques (like random rotations, flips, and zooms) on the training set to improve the model's generalization and robustness.

The training process will be driven by the backpropagation algorithm, facilitated by the automatic differentiation capabilities of our chosen framework. We will select an appropriate loss function (e.g., categorical cross-entropy) to quantify the discrepancy between the model's predictions and the ground truth labels. Optimization of the network's parameters (weights and biases) will be achieved using a variant of gradient descent, such as Adam or SGD, provided by the framework's optimizers. We will monitor the model's performance on a separate validation set during training to tune hyperparameters, detect overfitting, and ensure optimal generalization. Finally, the trained model's performance will be evaluated on a held-out test set to obtain an unbiased estimate of its ability to classify unseen coffee leaf images. This comprehensive approach, utilizing established deep learning frameworks and best practices in data handling and training, will enable us to build a highly effective coffee leaf disease classification system.

5. Architecture Design Diagram

High-level overview of the architecture of our project



Component	Description
Flutter Mobile App	Built with Flutter for Android/iOS; manages UI, image capture, and shows diagnosis.
Camera Access	Uses mobile device's camera to scan leaf images.
Pre-trained ML Model (TFLite)	TensorFlow Lite version of a CNN that identifies disease class.
CNN Classifier	Deep learning model trained on ~65K labeled leaf images.
Diagnosis Result Display	Shows text and voice output in Amharic/local languages with treatment tips.
Local Database & Sync	Stores history of scanned leaves; syncs with server when online.

`

6. Data Sources

For this project, our primary data sources will be the publicly available datasets referenced in our preliminary research: the JMuBEN and JMuBEN2 datasets from Kenya, the Ethiopian Coffee Leaf Dataset, and the Coffee-Leaf-Diseases dataset from Kaggle. These datasets collectively offer a diverse collection of images relevant to our target coffee leaf diseases (Cercospora, Rust, Phoma, Miner) and healthy leaves, totaling approximately 65,000 images. Notably, a subset of our working data, specifically 58,555 images, consists of cropped views emphasizing the damaged portions of the leaves, while the remaining images present potentially whole or less tightly cropped leaves. Preprocessing steps will include standardizing image dimensions to ensure consistent input for our CNN model, normalizing pixel values to facilitate efficient training, and applying data augmentation techniques such as rotations and flips, particularly to the training set, to enhance the model's robustness and generalization across different image variations and viewpoints. We will also address potential class imbalances identified in our initial data analysis through techniques like class weighting or oversampling.

7. Literature Review

Existing literature supports the use of convolutional neural networks (CNNs) for accurate plant disease detection, with Taye & Goel (2024) achieving 99.9% accuracy using ResNet50 on Ethiopian coffee leaves. Dinku (2022) confirmed CNN feasibility with local datasets, while Abdalla et al. (2023) highlighted the dominance of deep learning in agricultural applications. Rahman et al. (2020) demonstrated the potential for mobile deployment, and FAO (2021) emphasized the need for multilingual tools to enhance rural adoption. However, current solutions lack real-time diagnosis, local language support, and actionable guidance. Building on these findings, our project integrates CNN-based disease detection with a mobile app offering real-time results, multilingual accessibility, and post-diagnosis recommendation addressing both technical and user-experience gaps in existing research.

Implementation Plan

1. Technology Stack

Category	Technology/Tool
Programming Languages	Dart (Flutter), Python (Model Training)
Mobile Framework	Flutter
Deep Learning	TensorFlow, Keras, CNN
Deployment	TensorFlow Lite for mobile
IDE	Visual Studio Code
Dataset Tools	Kaggle, Mendeley, XML Annotators
Communication	WhatsApp (Team Coordination)

Preprocessing	OpenCV, PIL (Python Image Processing)

For the implementation of our coffee leaf disease classification project, we plan to utilize the following technologies and tools:

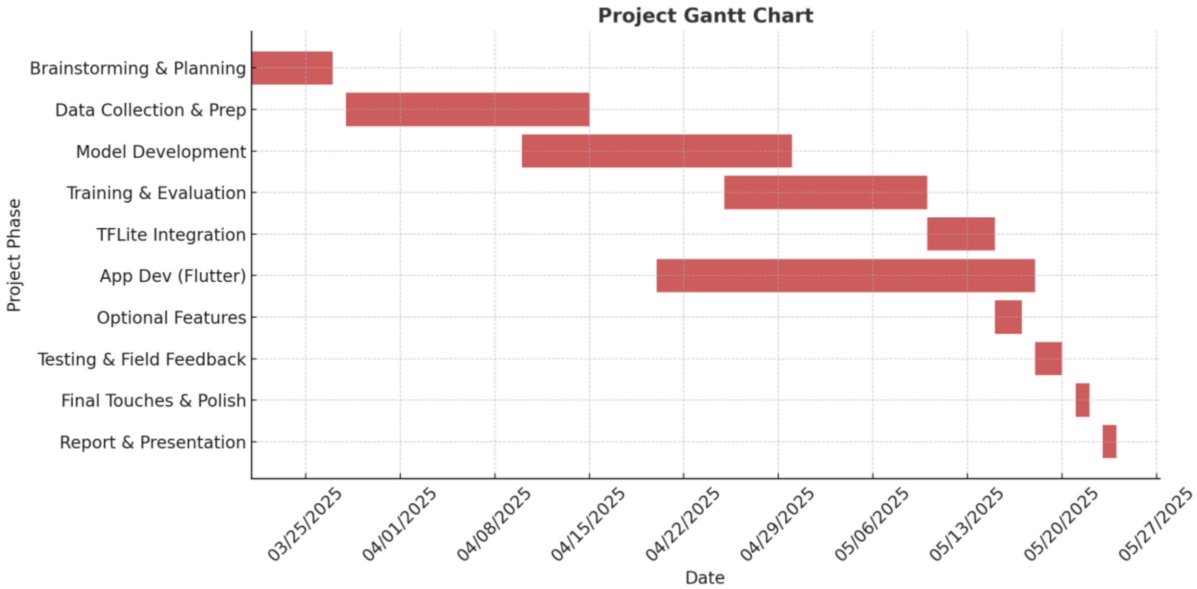
- Programming Language: Python will be our primary programming language due to its extensive ecosystem of scientific computing and deep learning libraries, strong community support, and ease of use.
- Deep Learning Framework: We will leverage either TensorFlow (with Keras API) or PyTorch. Both are powerful and widely adopted deep learning frameworks well-suited for developing and training convolutional neural networks. The final choice will be based on experimentation and team familiarity to optimize development efficiency within the Google Colab environment.
- Core Libraries:
    - NumPy: For efficient numerical computations and array manipulation.
    - Pandas: For data manipulation and analysis, particularly useful during the data exploration and preprocessing phases.
    - Matplotlib and Seaborn: For data visualization within the Colab notebooks, enabling us to understand dataset characteristics and model performance.
    - Scikit-learn (sklearn): For various machine learning utilities, including data splitting (train_test_split with stratification), preprocessing (e.g., standardization), and evaluation metrics.
    - OpenCV (cv2) or Pillow (PIL): For image loading and manipulation within the Colab environment.
- Data Handling Libraries (within chosen framework):
    - TensorFlow: tf.data for efficient data pipelines and tf.keras.preprocessing.image.ImageDataGenerator for image loading and augmentation, fully compatible with Google Colab.
    - PyTorch: torch.utils.data.DataLoader for creating efficient data loaders and torchvision.transforms for image preprocessing and augmentation, readily usable within Google Colab.
- Hardware Components:
    - Google Colaboratory (Colab): We will exclusively utilize Google Colab for model training. Colab provides free access to cloud-based computational resources, including GPUs and TPUs, which are essential for accelerating the training of our convolutional neural network on our relatively large dataset.
- Development Environment: Google Colab notebooks will serve as our primary development environment, offering an interactive and collaborative platform for coding, experimentation, and visualization directly within the browser.
- Version Control: Git for managing code changes and collaborating effectively as a team, with a platform like GitHub or GitLab for repository hosting, which integrates well with Google Colab.
- Experiment Tracking (Optional but Recommended): We may explore using tools like TensorBoard (integrated with TensorFlow and easily used in Colab) or other

lightweight experiment tracking solutions compatible with Colab to monitor metrics and compare different model configurations.

- **Task Distribution Matrix**

Task	**Abdulkerim**	**Luel**	**Rabiya**
Project Planning & Research	✓	✓	✓
Data Collection & Preprocessing	✓		✓
CNN Model Design & Training	✓ **(Lead)**		
Machine Learning Integration	✓ **(Lead)**		
Flutter App UI/Frontend		✓ **(Lead)**	✓ **(Lead)**
TFLite Integration	✓	✓	✓
Voice Output (Optional)		✓	✓
History Tracker (Optional)		✓	✓
Testing & Debugging	✓	✓	✓
Paper/Report Preparation	✓	✓	✓
Presentation & Slide Prep	✓	✓	✓

2. Timeline



3. Milestones

Milestone	**Target Date**
Brainstorming & Requirements Finalized	Mar 27, 2025
Dataset Ready (Preprocessed + Annotated)	Apr 29, 2025
CNN Model Achieves >90% Accuracy	May 15, 2025
TFLite Model Integrated into Mobile App	May 15, 2025
Voice Output & History Tracker Implemented (Opt.)	May 17, 2025
Flutter App MVP Completed	May 18, 2025
Field Testing Complete with Feedback	May 20, 2025
Final Version Ready	May 22, 2025
Report & Presentation Submitted	May 24, 2025

4. Challenges and Mitigations

We anticipate several potential challenges during this project, along with corresponding mitigation strategies:

Data Quality: The collected datasets may exhibit variability in image quality and potential inconsistencies in labeling. The prevalence of cropped images focusing on damaged areas could limit the model's generalization to broader leaf views. Variations in background noise and image formats across sources also pose a challenge. To mitigate this, we will implement robust preprocessing pipelines, including standardization of image sizes and pixel values, noise reduction, and careful visual inspection for labeling errors. While we recognize the value of building models from fundamental principles, our primary approach to addressing the cropped image issue and achieve optimal performance will involve initial experimenting with well-established CNN architecture known for their effectiveness in image classification. If these models, trained on a dataset incorporating both cropped and full leaf images, do not yield satisfactory results, we may explore more targeted strategies or architectural adaptations.

Model Performance: Achieving high accuracy and good generalization across the different coffee leaf diseases can be challenging due to subtle visual differences, varying disease stages, and the risk of overfitting. To address this, we will experiment with various established convolutional neural network architectures (e.g., ResNet, EfficientNet) as our starting point, leveraging their proven capabilities. Hyperparameter tuning, guided by validation set performance, will be crucial. We will employ data augmentation, dropout, and early stopping to prevent overfitting. Performance will be evaluated using a range of metrics, and if specific diseases pose greater difficulty, we will consider techniques like class weighting or targeted data collection. While building a model from scratch offers valuable learning, our strategy for this project prioritizes leveraging the strengths of established architectures and adapting them as needed based on the specific characteristics of our coffee leaf data to achieve the best possible classification accuracy and generalization.

Availability of Actual Leaves with Diseases to scan: There is a possibility that we might not find institutions that would grant us access to their facilities to test our application. For this reason, we plan to use high quality color print of leaves of plants with diseases. In addition, our application will include a fallback feature, which is supporting image uploading. This way we can test whether our app is working in case we do not get access to the leaves.

5. Ethical Considerations

- **Use of Pretrained Models and Bias**
  Using a pretrained CNN model introduces the risk of bias, as these models are often trained on datasets that may not represent the specific characteristics of Ethiopian coffee leaves. This can lead to inaccurate predictions or misclassifications when applied to local plant diseases. To address this, it's essential to fine-tune the model using region-specific data to ensure it performs reliably in the target environment and does not disadvantage local farmers through biased outputs.
- **Data Privacy**
  While the project primarily uses plant images, any data collected from farmers or local organizations must be handled ethically. This includes obtaining informed consent,

ensuring proper anonymization, and respecting data ownership. Clear communication and transparency about how the data will be used are key to maintaining trust and collaboration with the farming community.

6. References

Abdalla, A., Khan, S., & Malik, A. (2023). An advanced deep learning models-based plant disease detection: A review of recent research. *Frontiers in Plant Science*, 14, 1158933. https://doi.org/10.3389/fpls.2023.1158933

Dinku, A. T. (2022). Ethiopian coffee leaf disease detection using deep learning. [Master's Thesis, St. Mary’s University]. http://repository.smuc.edu.et/handle/123456789/7873

Food and Agriculture Organization (FAO). (2021). Multilingual voice interfaces for inclusive digital agriculture. https://www.fao.org/documents/card/en/c/CB4957EN