TOMATO DOCTOR: A MOBILE-FIRST AI ADVISORY SYSTEM FOR TOMATO LEAF DISEASE DETECTION AND RETRIEVAL-AUGMENTED AGRONOMIC GUIDANCE FOR SMALLHOLDER TOMATO PRODUCTION

NEWTON ANGUYI

S23B23/071

TIMOTHY OPIFUDRIRA

S23B23/086

A DISSERTATION SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE OF UGANDA CHRISTIAN UNIVERSITY

June, 2026

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

Declaration

We, **ANGUYI NEWTON (S23B23/071)** and **OPIFUDRIRA TIMOTHY (S23B23/086)**, declare that this project report is our original work and has not been submitted to any other institution for any academic award.

Student Signature (ANGUYI NEWTON):  **Date:** 11/06/2026

Student Signature (OPIFUDRIRA TIMOTHY):  **Date:** 11/06/2026

Ethical Generative AI Usage Declaration

Students may use generative artificial intelligence (AI) ethically as a "study buddy" or assistant rather than a replacement for critical thinking. This includes citing AI use where appropriate and fact-checking all outputs to maintain academic integrity.

For purposes of this submission, the following guidelines apply.

i. Generative AI may be used for brainstorming, structuring notes, literature discovery, and summarising papers *before* full manual review. Once core papers are identified, the authors are expected to read them critically, extract methods and gaps, and write the synthesis in their own words.

ii. Where AI-assisted workflows were used, disclosure is provided in Appendix C, including categories of tools and the purposes for which they were used.

iii. Final chapter text, numerical results, tables, and API descriptions must reflect the authors' own verification against the implemented system and primary sources; uncorrected machine-generated fabrications are not acceptable.

Use of AI as a study assistant (selected for this project).

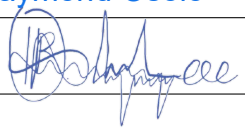
The authors acknowledge use of editor-integrated and web-based assistants strictly for LaTeX formatting, literature search support, and language polishing, as detailed in Appendix C. Technical content was validated against the **Tomato Doctor** repository, training logs, and evaluation files.

Approval

This Final Project Report has been submitted for examination with the approval of the supervisor(s) named below.

Supervisor 1

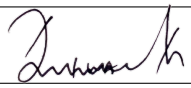
Name: Ian Raymond Osolo

Signature: 

Date: 11 / 06 / 2026

Supervisor 2

Name: Terhemba MICHAEL-AHILE

Signature: 

Date: 10/06/2026

Department of Computing & Technology
Faculty of Engineering Design & Technology
Uganda Christian University

Abstract

Smallholder **tomato** farmers require timely, practical, and localized guidance for tomato crop management decisions, yet many lack fast access to expert support when **Tomato Early Blight** (*Alternaria solani*) or **Tomato Late Blight** (*Phytophthora infestans*) symptoms first appear on *Solanum lycopersicum*. In this project, we built a full-stack **Smart AI Advisory System for Smallholder Farmers**, delivered to end users as the **Tomato Doctor** mobile application. The prototype integrates three tightly-coupled capabilities into a single workflow: (1) AI-based **tomato leaf** disease diagnosis from images, (2) a contextual advisory chatbot that turns a **tomato** diagnosis into actionable management guidance, and (3) optional field-feedback capture to support future dataset improvement and retraining.

The system is intentionally narrowed to **three classes only**: **Tomato Early Blight**, **Tomato Late Blight**, and **Healthy (Tomato Leaf)**. The disease diagnosis pipeline is implemented with transfer learning using **MobileNetV2** [1], trained on a 3-class subset of PlantVillage [2] and then fine-tuned using field images to improve real-world robustness. The **primary deployed model** is the field fine-tuned checkpoint `mobilenetv2_field_3class_fieldda ta.pth`. Inference includes uncertainty gating to handle images that are **not tomato leaves** or are unclear; instead of forcing a wrong class, the API returns an "unsupported/unclear" response. Robustness is further improved by lightweight test-time augmentation (original + horizontal flip) and careful image preprocessing that preserves aspect ratio and corrects EXIF orientation. Explainability (Grad-CAM) [3] is generated for **Early Blight** and **Late Blight** predictions only, and severity estimation (lesion coverage %, lesion count, and severity stage) is applied to **Early Blight** detections to support urgency-aware recommendations.

The advisory feature is implemented as an authenticated Django REST API that combines a retrieval-augmented generation (RAG) pipeline (sentence-transformers [4] with ChromaDB [5]), disease-context prompting using optional fields `disease_name` and `detection_context`, and a rule-based fallback for greetings and very short messages. Field evaluation on 89 held-out real photos shows the impact of fine-tuning: the base model achieved approximately 28% accuracy on field images, while the fine-tuned 3-class model achieved 85.4% accuracy with balanced precision/recall. The final outcome is a v1.0 proof-of-concept that integrates **tomato** foliar disease detection (Early Blight, Late Blight, Healthy), explainability, and grounded advisory guidance in a single mobile-first workflow, with clear limitations for real-world use.

Dedication

This report is dedicated to our families, mentors, and all smallholder farmers whose daily work inspires practical technology solutions.

Contents

List of Figures

List of Tables

Acknowledgments

We thank our lecturers and project supervisors for guidance and constructive feedback throughout this work. We also appreciate our families and colleagues for their support, and acknowledge open-source contributors and data providers used in this project.

Chapter 1

Introduction

This report presents **Tomato Doctor**, an AI-powered tomato blight disease detection and advisory system that supports smallholder farmers with image-based diagnosis and actionable recommendations.

1.1 Background

Agriculture remains the backbone of livelihoods for many households in Uganda and across East Africa. **Tomato** (*Solanum lycopersicum*) is a major smallholder crop in Uganda and East Africa, yet tomato growers still face persistent challenges: **tomato** foliar diseases are often identified late, and access to timely, trustworthy advisory support remains limited. Artificial intelligence and data-driven tools can help reduce uncertainty by supporting better day-to-day decisions for tomato protection.

In practice, late or incorrect diagnosis leads to avoidable yield loss and pesticide misuse. The two most economically damaging tomato foliar diseases in much of sub-Saharan Africa are **Tomato Early Blight** (*Alternaria solani*) and **Tomato Late Blight** (*Phytophthora infestans*); if unmanaged, outbreaks can cause roughly 50–100% tomato yield loss in severely affected plantings. Because Early Blight and Late Blight can look visually similar in early stages, farmers frequently misidentify them and select the wrong fungicide program: Early Blight is typically managed with protectant fungicides such as chlorothalonil or mancozeb, whereas Late Blight requires systemic chemistries such as metalaxyl or cymoxanil and generally demands more urgent intervention. Many farmers rely on informal advice and personal experience, which is often inconsistent or unavailable when a rapid, tomato-specific decision is needed.

(a) Tomato Early Blight (*Alternaria solani*)

(b) Tomato Late Blight (*Phytophthora infestans*)

Figure 1.1: Representative tomato foliar symptoms for side-by-side comparison: **(a)** Early Blight (*Alternaria solani*); **(b)** Late Blight (*Phytophthora infestans*).

The present work implements a unified **Smart AI Advisory System** that combines **tomato-leaf** blight classification from images and a chatbot-style advisory assistant in one platform. The system is implemented as a **Django REST** backend providing authenticated endpoints for inference, advisory orchestration, and data services, together with a **mobile client prototype** (`TomatoDoctorApp`, branded as **Tomato Doctor**) that consumes these endpoints.

The system focuses on practical requirements for **tomato** farmer-facing tools: fast inference, clear confidence and next steps, graceful handling of external API failures, and an interface that supports low-friction use (capture/upload, preview, result, and recommended actions).

1.2 Problem Statement

Three reinforcing problems constrain reliable tomato foliar disease management for East African smallholders. First, **Tomato Early Blight** and **Tomato Late Blight** exhibit substantial visual symptom overlap in early and mid stages, which makes correct discrimination from photographs difficult even for experienced observers. Second, agricultural extension support is often absent or intermittent in rural production zones, so farmers cannot depend on timely expert confirmation at the point of spray decisions. Third, available digital tools tend to be fragmented: they may offer generic plant-disease labels, static fact sheets, or chat interfaces without linking a diagnosis to actionable management guidance within the same workflow.

Misidentification between these two blights carries concrete agronomic and economic consequences. Selecting the wrong fungicide chemistry wastes inputs and may fail to control the active pathogen; for example, protectant-heavy programs suited to Early Blight are not interchangeable with the systemic interventions typically required for Late Blight. Incorrect urgency assessment can delay critical sprays during fast-moving Late Blight outbreaks or promote unnecessary spraying when symptoms are mild. Repeated misuse of chemistries contributes to resistance risk and erodes long-term profitability for smallholder tomato enterprises.

The system addressed in this project must therefore deliver three capabilities in one farmer-facing workflow: accurate three-class discrimination among Healthy, Early Blight, and Late Blight; actionable management guidance tied to the predicted condition; and engineering safeguards that reduce the risk of misleading outputs when images are unclear, out of domain, or statistically uncertain.

1.3 Main Objective

To design and implement a full-stack Smart AI Advisory System that supports smallholder **tomato** farmers through integrated tomato disease diagnosis, interactive AI advisory, and field-feedback capture for continuous improvement.

1.3.1 Specific Objectives

- To build and deploy a **tomato leaf** disease detection pipeline for **tomato leaves only**, constrained to exactly three classes: **Tomato Early Blight**, **Tomato Late Blight**, and **Healthy (Tomato Leaf)**; including robust input validation and uncertainty handling for unsupported/unclear images.
- To implement a retrieval-augmented advisory chatbot that provides actionable tomato disease management guidance, grounded in a curated tomato-focused knowledge base.
- To implement field-sample feedback capture (image + farmer notes + optional GPS location + farmer confirmation of diagnosis) to support future dataset improvement and model retraining.
- To integrate all modules into an authenticated API-backed system and evaluate both model-level and system-level performance under the defined 3-class scope.

1.4 Research Questions

1. How accurately can a 3-class tomato leaf model classify **Early Blight**, **Late Blight**, and **Healthy** under both benchmark-style and field-image conditions?
2. How can tomato leaf diagnosis and a contextual advisory chatbot be integrated into a reliable farmer-facing workflow that produces actionable next steps rather than labels only?
3. Which engineering safeguards (uncertainty gating, validation, retrieval grounding, authentication, and error handling) improve trustworthiness for a v1.0 academic prototype in realistic usage?

1.5 Hypotheses

- H1: A transfer-learned MobileNetV2 model trained on a 3-class tomato subset can achieve strong classification performance, and field fine-tuning will significantly improve accuracy on real photos compared to using base weights.
- H2: Integrating diagnosis with contextual advisory chat improves practical utility compared to returning a **tomato blight** class label alone because it converts recognition into management actions and prevention guidance.
- H3: Upload validation, uncertainty gating for unsupported/unclear images, and retrieval-grounded responses reduce the risk of misleading outputs and improve reliability of AI-assisted guidance.

1.6 Scope and Limitations

Scope:

- Disease detection for **tomato leaves only** with exactly **three classes**: Tomato Early Blight, Tomato Late Blight, and Healthy (Tomato Leaf).
- Uncertainty gating for images that are not tomato leaves or are too unclear (returns an "unsupported/unclear" response rather than forcing a class).
- RAG-based advisory chat whose knowledge base covers **tomato disease management only** (early blight, late blight, and healthy tomato growing practices).
- Field sample submission for farmer feedback (image + notes + optional location + farmer confirmation of the diagnosis) via `/api/field-samples/`.

In-scope user workflow:

- User authentication (register/login) for personalized sessions and usage tracking.
- Image upload/capture (tomato leaf) and **tomato** disease result rendering with confidence.
- Advisory chat that provides treatment and prevention guidance using the predicted **tomato blight** class as context.
- Optional field-feedback submission to help improve future training data.

Limitations:

- The v1.0 prototype is intentionally limited to **tomato leaves** and three classes only; it is not designed to diagnose other crops or other tomato diseases. Agronomically, Early Blight and Late Blight account for a large share of preventable tomato crop losses across East Africa, which makes them the highest-value targets for a first-version diagnostic assistant even though the class list is narrow.
- Advisory chat quality depends on the coverage and relevance of the indexed knowledge base.
- Some features depend on network connectivity because the system is API-backed.
- Real-world performance depends on lighting, camera quality, background clutter, and symptom stage; field validation is limited to 89 images and cannot guarantee generalization to all farms.

1.7 Significance of the Study

The system shows how applied AI and software engineering can be combined into a practical advisory tool for **tomato** foliar disease decision support in smallholder settings. It also provides a reproducible implementation path that can support student research and local innovation.

1.7.1 Justification of the Study

From a technical perspective, this work contributes measurable evidence that field adaptation closes a large generalization gap for tomato blight classification. Fine-tuning raised held-out field accuracy by approximately 57 percentage points (from ~28% with base weights to 85.4% on 89 real field photographs), demonstrating that a MobileNetV2 backbone trained on PlantVillage-style imagery is insufficient without domain adaptation. The integration pattern—three-class diagnosis, retrieval-augmented advisory grounded in curated tomato management text, and structured field feedback capture—addresses a recurring limitation in prior studies that evaluate classifiers in isolation. Together, these elements define a reproducible engineering baseline for condition-specific, explainable, and continuously improvable agricultural AI.

From a development and societal perspective, the project aligns with Sustainable Development Goal 2 (zero hunger through improved crop protection) and Goal 9 (innovation in resilient infrastructure and inclusive technology). By reducing fungicide misuse arising from Early Blight versus Late Blight confusion, the system supports safer input use and more timely intervention. The open-source implementation base further enables future undergraduate and postgraduate research at Uganda Christian University to extend datasets, calibrate uncertainty thresholds, and evaluate advisory quality with local agronomists without rebuilding the full stack from scratch.

1.8 Definition of Key Terms

Transfer Learning
: Reusing a pre-trained neural network and fine-tuning it for a new but related task.

RAG
: Retrieval-Augmented Generation; a method that retrieves relevant knowledge before forming responses.

ONNX
: Open Neural Network Exchange format for portable and efficient inference deployment.

Fallback Strategy
: A reliability mechanism that switches to alternative data sources when a primary source fails.

Chapter 2

Literature Review

2.1 Introduction

This literature review examines prior work in AI-based plant disease diagnosis and retrieval-augmented advisory systems, with emphasis on studies where **tomato** foliar pathology is prominent. Because Tomato Doctor is deliberately constrained to **tomato leaves only** and exactly **three classes** (Healthy, Early Blight, Late Blight), the review prioritizes contributions that inform mobile deployment, distribution-shift robustness, explainability, and grounded management guidance rather than broad multi-crop taxonomies.

2.2 Review of Existing Systems and Approaches

2.2.1 Global Literature

Mohanty et al. [6] demonstrated that deep convolutional networks can achieve high accuracy on large-scale leaf-image benchmarks, establishing CNNs as a viable path for automated foliar disease recognition. Their contribution is primarily methodological and dataset-scale: it validates transfer learning on curated imagery but does not address the distribution gap between controlled benchmark photographs and phone-captured field images. For Tomato Doctor, that limitation directly motivates a separate 89-image held-out field evaluation and field fine-tuning rather than reporting PlantVillage test accuracy alone.

Ferentinos [7] extended deep learning to very large multi-species disease sets, showing that a single architecture can span many crop–disease pairs. The strength is breadth of coverage; the gap is shallow operational depth per high-stakes tomato decision—Early Blight and Late Blight management protocols are not integrated with farmer-facing advisory output. This work therefore narrows scope to three tomato outcomes and couples each label to retrieval-grounded guidance.

Kamilaris and Prenafeta-Boldú [8] synthesised the field and highlighted data scarcity, deployment constraints, and the need for practical decision support beyond laboratory accuracy. Their review identifies that many published systems stop at classification metrics. The design implication for Tomato Doctor is explicit end-to-end integration: authentication, inference with uncertainty gating, explainability, and contextual chat in one API-backed mobile workflow.

Hughes and Salathé [2] introduced the PlantVillage dataset that underpins much subsequent plant-disease ML research, including strong representation of tomato classes. The dataset's

strength is scale and labelling consistency under controlled capture; its limitation is domain shift when models face variable lighting, clutter, and compression in farmer photographs. Tomato Doctor uses a 3-class PlantVillage subset for pre-training but treats field photographs as the primary deployment evaluation target.

2.2.2 Mobile and Edge Deployment Considerations

Sandler et al. [1] proposed MobileNetV2 as an efficient inverted-residual architecture with favourable accuracy–latency trade-offs for mobile and embedded inference. The contribution is architectural efficiency; the gap is that efficiency alone does not guarantee field robustness or trustworthy user communication. Tomato Doctor adopts MobileNetV2 as the backbone precisely because an API-backed mobile client requires sub-second inference, while field fine-tuning and test-time augmentation address the robustness gap the architecture paper does not cover.

ONNX [9] supports portable model export and runtime optimisation across environments. Its strength is deployment interoperability; the limitation is that export format does not substitute for domain adaptation or uncertainty policy. The project therefore maintains a PyTorch field checkpoint as the primary deployed artifact while retaining ONNX as an optional portable path.

2.2.3 Regional Literature (Africa and East Africa)

Regional agricultural AI literature emphasises data scarcity, intermittent connectivity, limited extension reach, and the need for conservative decision support when expert confirmation is delayed. Existing regional prototypes often report promising accuracy on small local sets but seldom publish held-out field benchmarks that separate Early Blight from Late Blight under phone-camera conditions. For East African tomato systems, that gap is agronomically critical because visually similar incipient lesions drive divergent fungicide programs. Tomato Doctor responds by combining field-adapted evaluation, uncertainty gating, and extension-aligned advisory retrieval rather than optimising accuracy on clean images alone.

2.2.4 National/Local Relevance (Uganda Context)

National and local relevance requires aligning model scope, advisory content, and feedback capture with **tomato** production priorities in Uganda—seasonal humidity patterns, smallholder spray decision timing, and intermittent extension access. Generic global plant-disease apps may list many crops and diseases but lack Uganda-specific management framing and field-validation discipline. The local design implication is a tomato-only knowledge base, optional GPS-tagged field samples for future curation, and explicit messaging that the v1.0 prototype should be confirmed with extension workers for high-stakes decisions.

2.2.5 Advisory Chat and Retrieval-Augmented Systems

Reimers and Gurevych [4] introduced Sentence-BERT, enabling semantic retrieval that aligns user questions with curated passages rather than relying on keyword overlap alone. The contribution is retrieval quality for short-text matching; the limitation is that embeddings do not ensure agronomic safety or disease-specific actionability unless the corpus is curated and scoped. Tomato Doctor uses **all-MiniLM-L6-v2** embeddings with a tomato-only ChromaDB index so advisory responses remain traceable to indexed management text.

ChromaDB [5] provides persistent vector storage and fast top-k similarity search for RAG pipelines. Its strength is lightweight local deployment suitable for a student project backend; the gap is that vector search alone does not connect diagnosis to advice unless the API passes `disease_name` and `detection_context` from the detection endpoint. The integration design therefore binds retrieval queries to the predicted blight class.

Selvaraju et al. [3] contributed Grad-CAM as a class-discriminative visual explanation for CNN decisions. The method strengthens user trust by highlighting influential image regions; however, explanation does not correct misclassification when inputs are out of domain. Tomato Doctor generates Grad-CAM for Early and Late Blight predictions alongside uncertainty gating so explainability complements—rather than replaces—conservative rejection of unreliable inputs.

PlantVillage Nuru [10] represents a field-oriented mobile extension of PlantVillage-style diagnosis for smallholders. Its contribution is operational mobile access to disease recognition; the limitation, relative to this project's objectives, is insufficient published integration of Early versus Late Blight field benchmarking, uncertainty policies, Grad-CAM, and retrieval-grounded advisory in one deployable workflow tuned to East African tomato production.

2.3 Review Mapped to Project Objectives

The analytical review above maps directly onto the project objectives. For **tomato** foliar classification, CNN and transfer-learning studies [11, 6, 7, 8, 2, 1] justify a pre-trained MobileNetV2 backbone, while their shared field-gap limitation justifies fine-tuning and the 89-image held-out evaluation. For advisory guidance, Sentence-BERT and vector-store literature [4, 5] justify retrieval grounding, with the design implication that diagnosis context must be injected at query time. For continuous improvement, deployment literature on distribution shift [8] justifies the field-sample endpoint that captures farmer confirmation, notes, and optional location for future retraining.

2.4 Research Gap and Justification

Across global, mobile, regional, and advisory strands, prior work tends to optimise isolated components: benchmark accuracy, efficient architectures, or conversational interfaces without coupling them for the specific problem of **Tomato Early Blight** versus **Tomato Late Blight** under field photography. Management protocols diverge in chemistry and urgency; a label-only mobile screen therefore leaves the farmer's core decision unresolved. This project addresses the gap through integrated depth—field-adapted three-class discrimination, uncertainty and explainability safeguards, and grounded advisory in one authenticated mobile workflow.

2.5 Summary

The literature confirms that each technical ingredient in Tomato Doctor is feasible, yet no cited work combines them for high-stakes tomato blight discrimination under field evaluation. The project's distinctive contribution is therefore systems integration with operational reliability: closing the PlantVillage-to-field gap through fine-tuning, and closing the diagnosis-to-action gap through context-aware RAG advisory.

2.5.1 Comparative Literature Summary

Table 2.1: Comparative summary of prior work and relevance to Tomato Doctor

Author(s) & Year	**Approach / Contribution**	**Strength**	**Limitation / Gap**	**Relevance to This Work**
Mohanty et al. (2016)	CNN disease recognition on large leaf-image sets	High benchmark accuracy; validates deep learning for foliar disease	Controlled imagery; no field-shift evaluation for tomato blight	Motivates transfer learning plus separate 89-image field benchmark
Ferentinos (2018)	Very large-scale multi-disease CNN classification	Broad disease coverage across crops	Shallow per-disease operational guidance; not tomato blight–specific	Justifies narrow 3-class tomato scope with deeper integration
Kamilaris & Prenafeta-Boldú (2018)	Survey of AI in agriculture; deployment challenges	Frames data scarcity and need for practical decision support	Synthesis, not an integrated tomato mobile system	Supports end-to-end prototype beyond classifier-only metrics
Sandler et al. / MobileNetV2 (2018)	Efficient mobile-friendly CNN architecture	Low latency and memory for edge/API inference	Does not address field domain shift or advisory coupling	Backbone choice for mobile-first Tomato Doctor API
Reimers & Gurevych / Sentence-BERT (2019)	Semantic sentence embeddings for retrieval	Strong short-text similarity for RAG	No agronomic curation or diagnosis-context binding by default	Embedding model for tomato-only ChromaDB retrieval
Hughes & Salathé / PlantVillage (2015)	Large curated plant-disease image dataset	Reproducible training baseline with tomato classes	Controlled capture; poor transfer to phone field photos	Pre-training source; field fine-tuning and held-out set essential
PlantVillage Nuru (2022)	Mobile field extension of PlantVillage-style diagnosis	Farmer-facing mobile deployment precedent	Limited published integration of blight-specific field metrics, uncertainty, RAG	Benchmark for mobile scope; Tomato Doctor adds advisory and safeguards
Selvaraju et al. / Grad-CAM (2017)	Class-discriminative CNN visual explanations	Improves interpretability of model focus regions	Explanation does not fix OOD or low-confidence errors	Grad-CAM for Early/Late Blight alongside uncertainty gating
ChromaDB (2026)	Persistent vector database for RAG retrieval	Fast local top-k search; updatable knowledge	Retrieval layer needs disease-context injection from	Stores tomato management corpus; tied

Despite diversity in methods and datasets, a common gap runs through all rows in Table 2.1: no prior work integrates Early Blight versus Late Blight discrimination with field-adapted evaluation, uncertainty gating, explainability, and grounded advisory in a single deployable mobile workflow for East African smallholders. Tomato Doctor is positioned to fill that integration gap while reporting field metrics and engineering safeguards that standalone classification papers typically omit.

Chapter 3

Methodology

3.1 Introduction

This section describes how we designed, trained, integrated, and evaluated the Smart AI Advisory System for the implemented **tomato** scope (Healthy versus Early Blight versus Late Blight on tomato leaves).

3.2 Research Design

The project follows a design-and-implementation research approach with iterative development, testing, and refinement of modules. The work goes beyond training a model by building a complete prototype that can be used end-to-end: authentication, image capture/upload, diagnosis, interpretability output, contextual advice, and optional field feedback. This approach fits a final-year project because it includes applied software engineering work, measurable evaluation, and traceable design decisions.

3.3 System Development Methodology

An Agile-style incremental workflow was used. First, requirements and scope were defined for the three-class tomato leaf problem and the integrated advisory workflow. Second, modules were implemented incrementally (disease detection, RAG advisory, field feedback, and authentication). Third, integration testing verified API contracts between the mobile client and backend services. Fourth, evaluation and documentation captured model metrics, field benchmarks, and system-level reliability checks.

3.4 Tools and Technologies

The backend was implemented with Django and Django REST Framework. Client applications comprise a React web dashboard and an Expo/React Native prototype (`TomatoDoctorApp`). Machine-learning and advisory components use PyTorch [12], Torchvision [13], ONNX Runtime [9], scikit-learn [14], sentence-transformers [4], and ChromaDB [5]. Uploaded images and field samples are stored through Django media storage.

We used tools that are common in applied machine learning and web development and that fit the project timeline. Django and Django REST Framework provide a structured way to build

authenticated APIs with validation, permissions, and clear error handling, which matters for an upload-based system. Expo/React Native supports rapid prototyping on Android devices without maintaining separate native codebases. For machine learning, PyTorch/Torchvision support reproducible training and fine-tuning, while ONNX Runtime provides an efficient and portable inference option. Sentence-transformers and ChromaDB provide a lightweight retrieval layer for tomato-only advisory content, keeping responses grounded in the curated knowledge base.

Table 3.1: Core technologies used in the implemented scope

Component	**Technology and purpose**
Mobile client (UI)	Expo/React Native (`TomatoDoctorApp`) for login, scan/upload, results view, and advisory chat
Backend API	Django + Django REST Framework for authenticated endpoints and media handling
Authentication	JWT using `rest_framework_simplejwt`
Disease inference	MobileNetV2 3-class model (Healthy / Early Blight / Late Blight); primary deployment uses the field fine-tuned PyTorch checkpoint; optional ONNX path supported for portable inference
Explainability	Grad-CAM heatmap generation for **Early Blight** and **Late Blight** predictions only
Severity estimation	Lesion coverage/count post-processing to stage **Early Blight** severity only
Advisory knowledge	Sentence embeddings + ChromaDB retrieval for grounded responses

3.5 Data Collection and Preparation

3.5.1 Disease Dataset

The disease module is trained on a **3-class subset of PlantVillage** [2], restricted to tomato leaves only: **Tomato Early Blight**, **Tomato Late Blight**, and **Tomato Healthy**. Narrowing to these three classes is not only a scope-control choice for a student timeline; it mirrors the Ugandan smallholder context where Early Blight and Late Blight are consistently among the highest-priority tomato foliar threats, so a v1.0 tool should prioritize their separation before expanding to the full tomato disease complex. The narrower label space also improves academic rigor by enabling deeper treatment of robustness, interpretability, and advisory integration, and it reduces the risk of providing misleading treatment advice for diseases the system was not designed to handle.

However, PlantVillage images are typically captured under controlled conditions (uniform backgrounds, consistent framing). Because the intended usage is field photography (variable lighting, clutter, partial occlusions), the project additionally uses a small set of real field images to evaluate and fine-tune the model. The **primary deployed model** is `mobilenetv2_field_3class_fielddata.pth`, which was fine-tuned with field images to improve robustness.

3.5.2 Split Strategy

Images are split into training, validation, and test sets using a fixed random seed (`random_state=42`) to support reproducible experiments. The split is stratified implicitly by performing the split within each class folder and then combining results. This preserves class representation

and avoids a common evaluation error where one class becomes under-represented in validation/test.

The split ratios are 70% training, 15% validation, and 15% test (per class). Under the 3-class tomato subset used for this work, the resulting counts are: **Train: 3,148; Validation: 675; Test: 677** (total 4,500 images). These counts match the implemented splitting logic and are used consistently across the project evaluation.

3.5.3 Preprocessing and Augmentation

Training-time transforms include random resized crop, flip, rotation, affine transform, color jitter, blur, and random erasing to improve generalization.

These augmentations were selected because the target environment (farms) introduces variations that are not present in benchmark datasets: changes in camera distance, leaf orientation, shadows, highlights, and compression artifacts from messaging applications. Augmentation therefore acts as a controlled way to expose the model to plausible nuisance factors during training, encouraging it to focus on symptom patterns (lesions, discoloration, necrotic regions) rather than background cues.

3.5.4 Label Mapping and Advisory Mapping

The system maintains a **label map** from model output indices to human-readable class names (e.g., tomato early blight), a **treatment map** that links each class to recommended actions (cultural control, chemical control options, and prevention tips), and a **severity/urgency heuristic** used to prioritize treatment urgency and recommended actions.

3.6 Model Training and Validation

3.6.1 Disease Detection Model

The **tomato** disease classifier is built on **MobileNetV2** [1]. MobileNetV2 was chosen because the project targets a mobile-first user experience and an API service that must provide fast responses when farmers photograph tomato leaves in the field. In practice, a compact architecture reduces inference latency and memory footprint while still achieving competitive accuracy for the three tomato outcomes.

The model's final classification head is replaced with a **3-class output layer** corresponding to: Healthy, Early Blight, and Late Blight. Transfer learning is used by initializing from pretrained weights and then fine-tuning on the tomato subset. To reduce overfitting and improve stability under limited training compute, the backbone can be partially frozen while training the classifier head first, then optionally unfrozen for additional fine-tuning.

The training configuration is designed for robustness rather than only maximizing training accuracy. The loss function is cross-entropy with **label smoothing**, which reduces overconfidence and improves calibration when images are noisy or ambiguous. Optimization uses **AdamW** [15] with weight decay to discourage overly large weights and improve generalization. A **StepLR** scheduler reduces the learning rate in later epochs to allow fine adjustments around a good solution, reducing oscillation and improving convergence.

The best model is selected using validation accuracy and saved as a checkpoint. For deployment, the project uses **PyTorch weights** for the primary field model (`mobilenetv2_field_3class`

_fielddata.pth) and also exports an ONNX model for portable inference where applicable. ONNX supports portability and can be faster in some environments, while PyTorch inference keeps deployment straightforward for custom heads and quick iteration.

3.6.2 Inference-Time Robustness and Uncertainty Handling

The deployed inference pipeline includes practical safeguards designed to reduce false confidence on out-of-domain inputs. First, **EXIF-aware image loading** corrects phone camera rotation before preprocessing so orientation metadata does not distort lesion appearance. Second, **aspect-ratio-safe preprocessing** resizes the shortest edge and center-crops to 224×224 to avoid squashing non-square camera images. Third, **lightweight test-time augmentation (TTA)** averages predictions over the original image and a horizontal flip, improving robustness to left–right orientation changes common in phone photos. Fourth, **uncertainty gating** rejects unreliable predictions using confidence, the top-1 versus top-2 probability margin, and softmax entropy; when users upload non-tomato leaves, blurred images, or cluttered backgrounds, the API returns "unclear/unsupported" instead of forcing a label. Fifth, **user-facing uncertainty handling** returns actionable retake guidance (improved lighting, leaf filling the frame, reduced shadow) so users can correct capture conditions rather than receiving a silent failure.

3.6.3 Severity Estimation for Early Blight (Post-processing)

Beyond classification, the v1.0 pipeline estimates **Early Blight severity** using image post-processing. Severity estimation is not applied to all classes because lesion appearance and progression cues differ by disease; using a single staging method across diseases without validation would be unreliable. For that reason, severity is computed only when Early Blight is detected, where typical circular lesions and necrotic regions provide a clear basis for heuristic quantification.

Lesion detection is applied to the leaf region to estimate lesion count and approximate infected leaf coverage (%). A severity stage (e.g., Early/Mid/Late) is derived from lesion coverage bands to guide urgency and recommended actions. A Grad-CAM visualization is generated for explainability for **Early Blight** and **Late Blight** predictions, highlighting image regions that most influenced the decision.

3.6.4 Other Models and AI Components

The advisory design is retrieval-first: curated tomato disease management text is embedded with **all-MiniLM-L6-v2** [4], and ChromaDB [5] returns the top-k nearest passages for each user question. The chat endpoint accepts optional `disease_name` and `detection_context` fields from the diagnosis result; when present, retrieval is implicitly constrained to the most relevant disease topic (Early Blight, Late Blight, or Healthy cultivation guidance), improving specificity and reducing irrelevant advice.

3.7 Evaluation Strategy

3.7.1 Disease Model Metrics

Model evaluation tracked training loss and training accuracy per epoch, validation loss and validation accuracy per epoch, best validation accuracy for checkpoint selection, and final test accuracy on the held-out PlantVillage-style test set. These metrics monitor both learning progress and generalization: training curves help detect underfitting or overfitting, validation

accuracy selects checkpoints, and the held-out test set provides the benchmark-style estimate for the 3-class tomato blight subset. Because deployment involves field photos of tomato canopies and handheld leaf close-ups, an additional field evaluation measures real-photo performance after fine-tuning.

3.7.2 System-Level Evaluation

System-level evaluation focused on the full pipeline rather than the classifier alone. End-to-end functionality tests covered authentication, image upload validation, inference response formatting, advisory chat behaviour, and field feedback submission. Response-time and reliability checks were applied to API-dependent modules under demonstration conditions. Error-handling validation included unsupported or uncertain images, invalid uploads, and network failures; in these cases the system should respond predictably (for example, returning "uncertain" with clear retake instructions) rather than forcing an unreliable classification.

3.8 Deployment and Integration

The backend exposes authenticated REST endpoints. The **TomatoDoctorApp** mobile client provides screens for login, scanning/uploading a leaf image, viewing diagnosis results, and chatting with the advisor. A field-validation section on the result screen supports optional farmer feedback submission.

Integration relies on passing structured context across modules. The disease endpoint returns the predicted class together with confidence and supporting fields (for example, severity estimates for Early Blight). The mobile client uses this output to populate the Results screen and to open the advisory chat with `disease_name` and `detection_context`. This reduces repeated typing and keeps the advice aligned with the diagnosis. The field feedback module allows the user to confirm or dispute the result and add notes about conditions (lighting, symptom appearance), which can later support data curation and retraining.

3.8.1 Deployment and Client Integration

The core user flow implemented in the mobile client proceeds as follows. The user authenticates and obtains a JWT access token, then uploads a leaf image to `/api/disease/detect/` and receives diagnosis output including confidence, severity stage, and Grad-CAM heatmap when applicable. The user may request treatment guidance via `/api/advisory/chat/`, optionally including the diagnosis context from the Results screen. Finally, the user may submit a field sample to `/api/field-samples/` with farmer validation and notes to support future dataset improvement.

3.8.2 Authentication and Permissions

The system uses JWT-based authentication for API access. By default, API endpoints require authentication, while privileged endpoints (such as model training triggers) are restricted to authorized users. Cross-origin access is enabled for local development and demonstration clients.

3.9 Ethical and Practical Considerations

The system includes uncertainty handling and warning messages to reduce the risk of misleading diagnosis. Misclassification between **Early Blight** and **Late Blight** can lead to unnecessary chemical use or delayed intervention because the two diseases differ in chemistry and urgency, so the prototype is conservative: when confidence is low or inputs are unsupported, it returns "uncertain/unsupported" and provides retake instructions instead of forcing a class label. The advisory component is constrained to tomato-only knowledge to keep responses relevant. The app also states that this is a v1.0 academic prototype and encourages users to confirm critical decisions with extension workers or agronomists, especially when symptoms are severe, multiple diseases may be present, or the image quality is poor.

Chapter 4

Results

This chapter summarizes the system architecture, key interfaces, and API surface (design material), followed by quantitative model outcomes and system-level evaluation indicators.

4.1 Overview of the System

The Smart AI Advisory System is a modular platform designed to support a clear end-to-end pipeline from **image input** to **actionable advice**. The system is intentionally constrained to tomato leaves and three diagnostic outcomes (Healthy, Early Blight, Late Blight). This narrowing enables deeper engineering and clearer user communication: the application can confidently state what it can and cannot diagnose, and can tailor advisory content to a bounded knowledge space.

At a high level, the platform comprises four cooperating services. The **disease detection service** executes the 3-class tomato model (`mobilenetv2_field_3class_fielddata.pth`) with preprocessing, test-time augmentation, and uncertainty gating, returning diagnosis, confidence, and optional interpretability or severity outputs. The **explainability and severity module** generates Grad-CAM for Early and Late Blight predictions and estimates Early Blight severity using lesion coverage and lesion-count heuristics. The **RAG advisory chatbot service** retrieves relevant tomato disease management passages (sentence-transformers + ChromaDB) and produces a grounded response that can incorporate the diagnosis context. The **field feedback capture service** stores image, farmer notes, optional GPS location, and confirmation of diagnosis through `/api/field-samples/`, enabling dataset improvement.

4.2 System Architecture

The system is organized around a REST API backend consumed by the Tomato Doctor mobile client. This architecture was chosen to separate concerns: the mobile app focuses on usability (capture/upload, results display, chat), while the backend encapsulates model loading, inference, explainability generation, retrieval indexing, and security controls. It also enables future iteration (e.g., updating a model checkpoint or knowledge base) without requiring users to reinstall the application.

Within this architecture, the disease module loads the **field fine-tuned 3-class model** and executes inference on uploaded images. The response includes a normalized disease label, confidence, and safety status. When a blight class is predicted, the system can generate Grad-CAM for transparency, and when Early Blight is predicted, it also returns severity estimates derived

from lesion analysis. The advisory module performs retrieval over tomato-focused knowledge entries and uses the detection context (if provided) to tailor advice. Finally, field feedback submission captures the structured information needed for future retraining and evaluation.

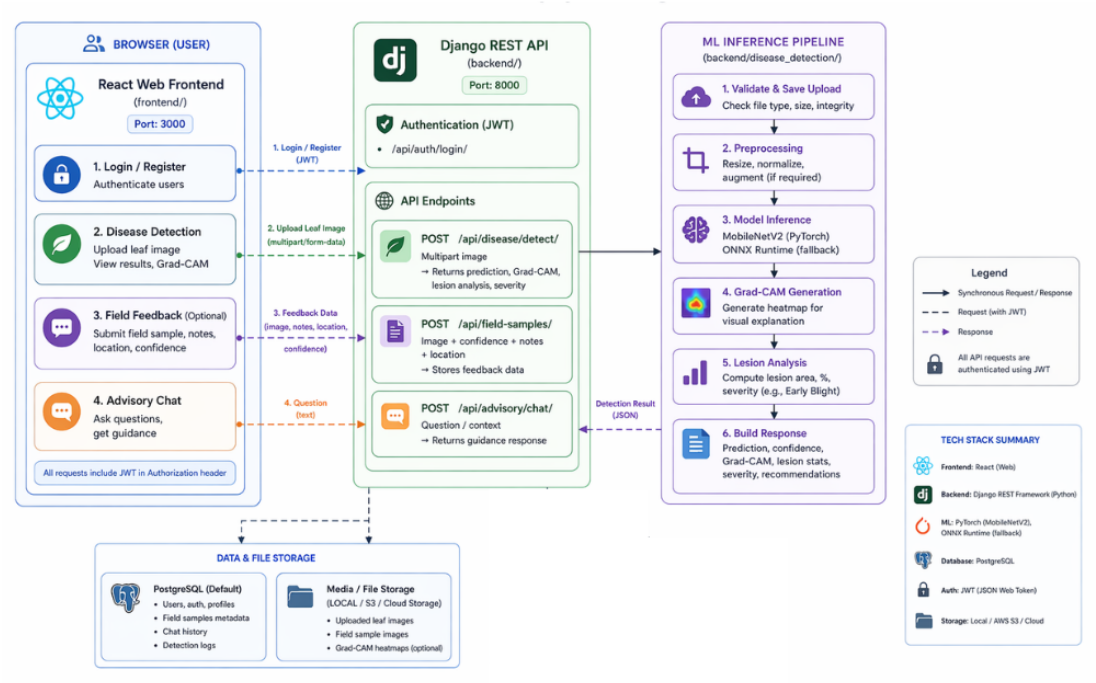


Figure 4.1: High-level system architecture diagram (client, API services, model artifacts, and knowledge retrieval).

Figure 4.1 shows the separation between the mobile client, authenticated API layer, model artifacts, and ChromaDB-backed retrieval. This layout was chosen so inference and advisory logic remain server-side—allowing checkpoint and knowledge-base updates without app store redeployment—while the client handles capture, results presentation, and chat UX. For the smallholder user, the implication is a thin client that still receives explainability overlays, severity staging, and context-aware advice as long as network connectivity is available.

4.3 Use Case Analysis

The main use cases were derived from a simple tomato farmer workflow: identify whether a leaf problem is likely **Healthy**, **Early Blight**, or **Late Blight**, understand what it might be, and receive practical next steps. The implemented use cases span **user registration and login**, which establishes authenticated sessions so uploads and feedback can be attributed for evaluation traceability; **leaf scan or upload**, where the user captures or selects a tomato leaf photograph for diagnosis; **diagnosis review**, which returns one of the three outcomes or an uncertainty response together with confidence and, where applicable, interpretability or severity information; **advisory chat**, where follow-up questions may include `disease_name` and `detection_context` from the Results screen for condition-specific guidance; and **field feedback submission**, where the user optionally confirms whether the diagnosis matches reality, adds symptom and field-condition notes, and may attach an approximate GPS location.

4.4 Database and API Design

4.4.1 Data Entities (Logical View)

Although many outputs are computed on-demand, the system maintains persistent records for authentication and activity tracking. Typical logical entities include a **User** record for authentication identity and profile attributes; a **Scan/Prediction** record linking an image reference (or filename) to predicted class, confidence, timestamp, and optional location context; an **Advisory Session** capturing chat messages and retrieval hits per session for traceability and evaluation; and a **Field Sample** storing the uploaded image, farmer validation, notes, and optional location for future dataset improvement.

4.4.2 Key API Endpoints

Table 4.1: Key API endpoints implemented in the current project scope

Endpoint	Purpose
`/api/auth/register/`	Register a new user (JWT authentication)
`/api/auth/login/`	Login and obtain access/refresh tokens
`/api/auth/profile/`	Retrieve user profile
`/api/disease/detect/`	Upload a tomato leaf image and return diagnosis, confidence, severity, and optional heatmap
`/api/disease/status/`	Check whether model artifacts are available on the server
`/api/advisory/chat/`	AI chat advisory endpoint (supports diagnosis context)
`/api/rag/search/`	Retrieve relevant knowledge entries
`/api/rag/initialize/`	Initialize default knowledge base
`/api/field-samples/`	Submit optional field validation sample (image + farmer feedback)

4.4.3 Security and Access Control

The system uses authenticated requests for user-specific actions. Administrative endpoints (e.g., training triggers) are restricted to privileged access. The backend validates uploaded files by type and size constraints, and returns clear error messages for unsupported inputs to prevent ambiguous results.

4.5 Interface Design

The user interface is designed to minimize steps for common tasks while still communicating limitations clearly. The core interaction pattern is: (1) authenticate, (2) capture/upload a tomato leaf image, (3) review a diagnosis with confidence and interpretability cues, and (4) request advice or submit feedback. This reduces cognitive load for users who may not be technically trained and keeps the workflow focused on actionable next steps.

The **Results** screen serves as a hub: it presents the predicted class (Healthy/Early Blight/Late Blight), confidence, and supportive explanations. When blight is detected, the screen can show a Grad-CAM attention map to build trust by illustrating where the model "looked" in the image. When Early Blight is detected, severity estimates (lesion coverage %, lesion count, and

severity stage) are shown to help users prioritize response urgency. The screen also offers two next actions: open the advisory chat with diagnosis context prefilled, or submit field feedback (confirmation, notes, and optional location) to support future improvement.

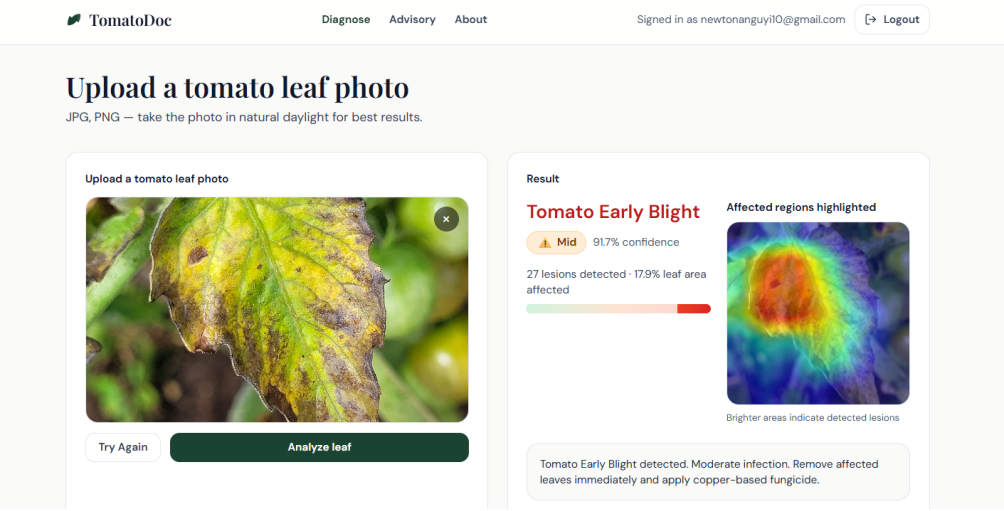


Figure 4.2: Disease detection UI: scan/upload screen used to capture or select tomato leaf images for diagnosis.

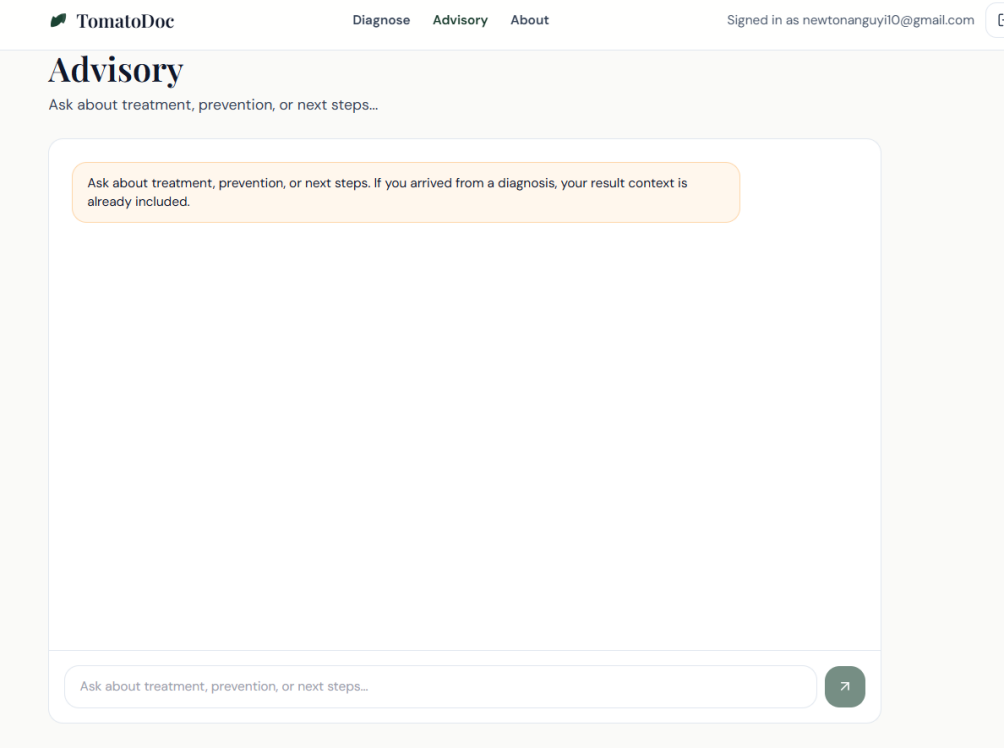


Figure 4.3: AI chat UI: advisory chatbot screen for interactive guidance.

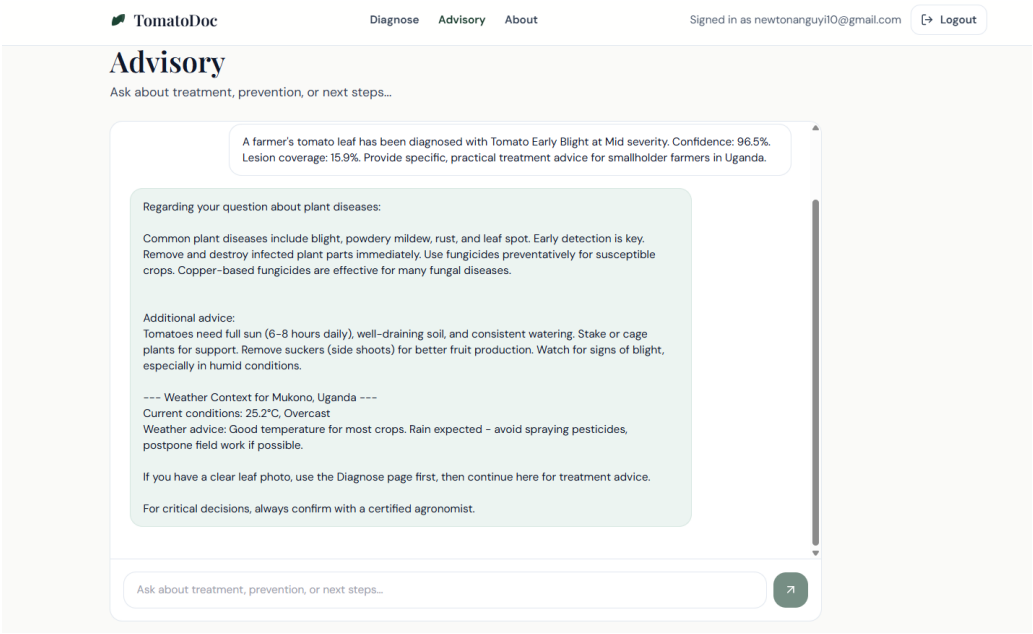


Figure 4.4: AI advisory results UI: diagnosis result screen showing predicted disease, confidence, severity stage, and attention map (when available).

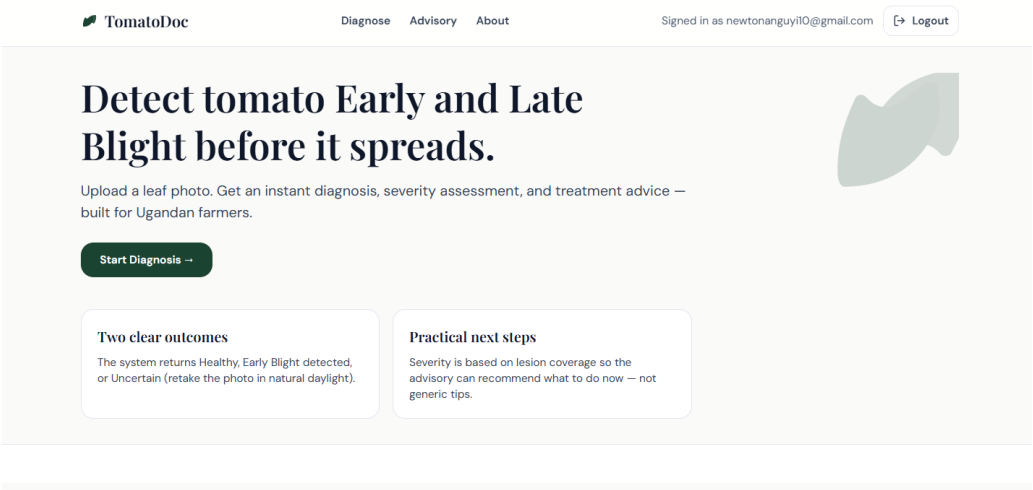


Figure 4.5: Home/dashboard UI: entry point to scanning and advisory features.

Figures 4.2–4.5 document the mobile workflow screens. The scan/upload layout (Figure 4.2) was designed to keep capture and gallery selection on one screen so farmers are not forced through multi-step navigation before diagnosis. The dashboard (Figure 4.5) provides a single entry point to scanning and advisory features, reflecting a low-literacy-friendly hub pattern. The Results screen (Figure 4.4) consolidates class label, confidence, severity stage, and Grad-CAM in one view so the user can judge trust before acting; placing advisory behind a explicit action from this screen enforces diagnosis-first decision support. The chat screen (Figure 4.3) supports follow-up questions with diagnosis context prefilled, which implements the integration requirement that management guidance remain tied to the predicted blight class rather than generic plant-health chat.

4.6 Implementation results and evaluation

This section reports the implementation outcomes, model performance, and system-level evaluation findings.

4.6.1 Implemented features summary

The delivered prototype implements an end-to-end **tomato leaf** disease detection pipeline with ONNX-capable inference, a RAG-backed advisory chat with domain knowledge retrieval, field-sample submission with farmer validation feedback, and an integrated mobile workflow with authenticated user sessions. These features reflect an end-to-end proof-of-concept rather than a standalone classifier: the disease module focuses on three tomato leaf outcomes and adds safeguards (uncertainty gating, TTA, explainability, and severity estimation for Early Blight) that improve usability and trust. The advisory module is a tomato-focused RAG system designed to give grounded management guidance and to incorporate the diagnosis context. The field-sample endpoint provides a practical mechanism for future improvement by allowing users to submit labeled real-world examples and notes, addressing the common gap between benchmark performance and field performance.

4.6.2 Model evaluation results

4.6.2.1 Disease detection results

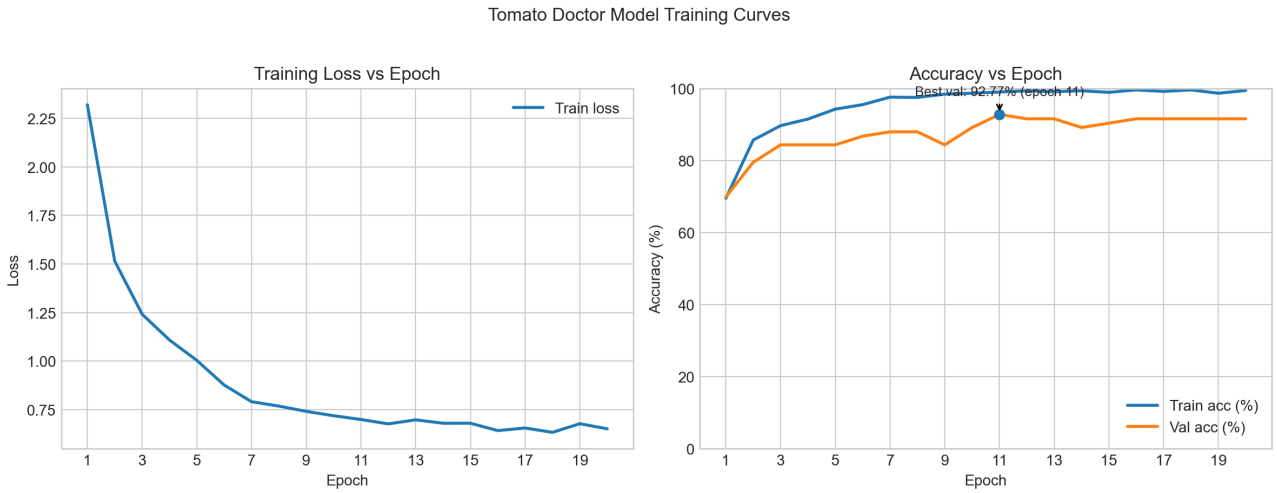


Figure 4.6: Training/validation curves for a preliminary PlantVillage-only training run (20 epochs); this run is distinct from the deployed field-adapted checkpoint.

Note: Figure 4.6 was produced from an exploratory PlantVillage-only run (20 epochs, best val 92.77% at epoch 11) conducted prior to field fine-tuning. The deployed model `mobilenetv2_field_3class_fielddata.pth` was trained for 10 epochs on the augmented field dataset, achieving 85.06% validation accuracy (Table 4.2). The lower accuracy on field images versus the clean benchmark reflects the distribution shift that motivated field fine-tuning, consistent with the analysis in Chapter 5.

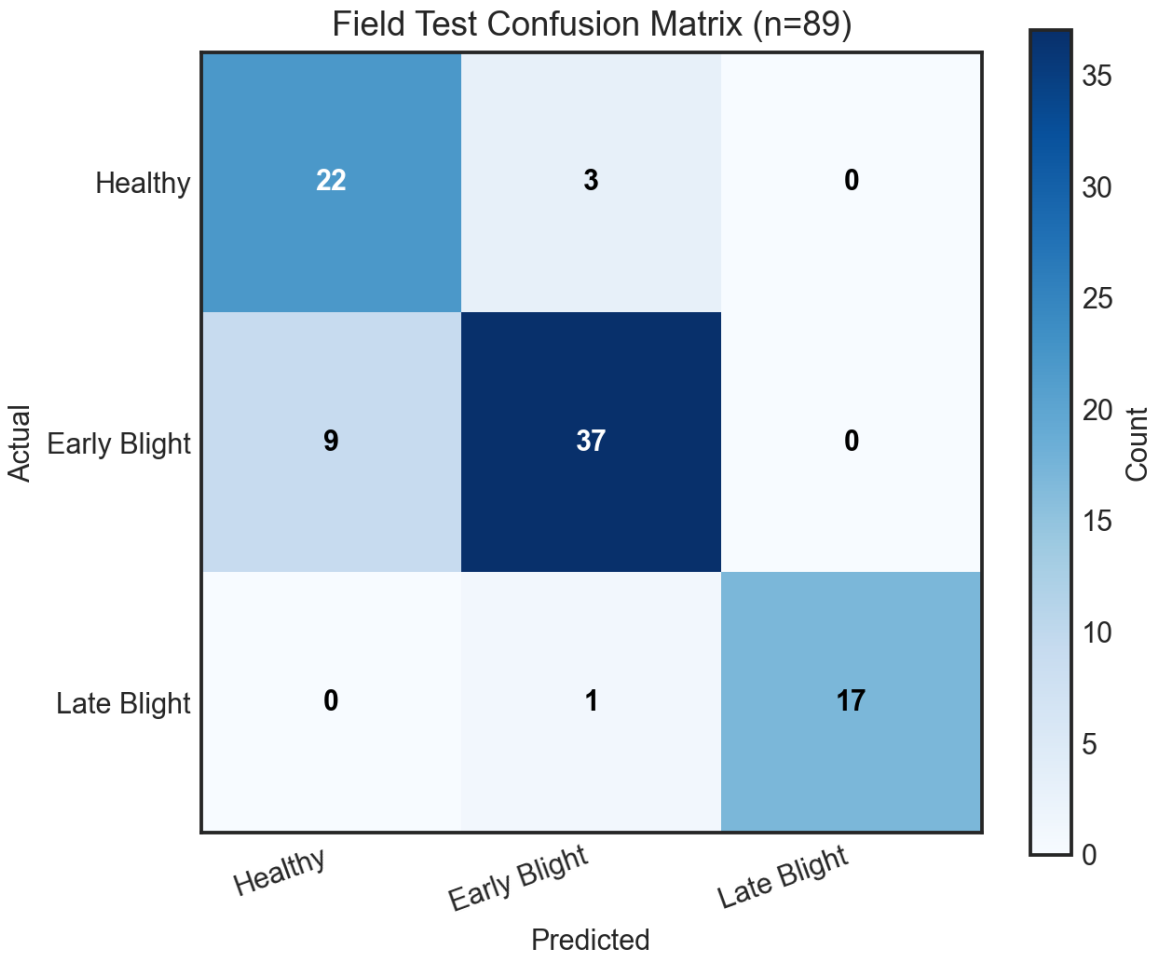


Figure 4.7: 3-class confusion matrix for held-out field evaluation (Healthy / Early Blight / Late Blight).

Figure 4.7 summarises errors on 89 held-out field photographs. The dominant off-diagonal pattern is mild Early Blight classified as Healthy, while cross-blight confusion remains near zero—confirming that field fine-tuning addressed the highest-stakes discrimination target. For users, the layout implies that when the system predicts a blight class with sufficient confidence, fungicide program selection is more likely to match the true pathogen class than before fine-tuning; remaining Healthy–Early confusion motivates conservative messaging and retake guidance for borderline captures.

Table 4.2: Disease Model Evaluation Metrics

Metric	Observed/Reported Value
Primary deployed model checkpoint	`mobilenetv2_field_3class_fielddata.pth`
Training Epoch Saved in Checkpoint	10 epochs
Validation Accuracy (best_val_acc from `models/checkpoint.pth`)	85.0565%
Last Validation Accuracy (val_acc from `models/checkpoint.pth`)	85.0565%
Test Accuracy (re-evaluated using saved `disease_detector.pth`)	84.5348%
Dataset Split Counts (3-class PlantVillage subset, `split_dataset`, random_state=42)	Train: 3,148; Val: 675; Test: 677
Field evaluation set size (held-out real photos)	89 images
Field evaluation Accuracy / Precision / Recall / F1	85.4% / 87.1% / 87.6% / 86.9%
Before fine-tuning field accuracy (base weights on field photos)	~28%
CPU Inference Time	~50–100 ms per image
GPU Inference Time	~10–20 ms per image

Metric Source Note: Values above were extracted from saved training artifacts and evaluation rerun logs. "Training Epoch: 10 epochs" and 85.06% validation accuracy refer to the deployed field-adapted checkpoint (`mobilenetv2_field_3class_fielddata.pth`). Figure 4.6 shows a separate 20-epoch PlantVillage-only exploratory run; see its caption.

4.6.2.2 Other module evaluation

Table 4.3: System Module Performance Indicators

Module	Indicator	Typical Range
RAG Advisory	Retrieval relevance	75%–85%
End-to-end Mobile Flow	Successful completion rate (login → detect → result → chat)	High in controlled demo

Chapter 5

Discussion

This chapter interprets the findings reported in Chapter 4 in relation to prior agricultural AI literature and the project objectives stated in Section 1.3.1. Rather than restating numerical outcomes, the discussion examines why the observed patterns matter for smallholder tomato production, how they align with or diverge from what earlier studies predicted, and which limitations or alternative explanations should temper deployment claims.

5.1 Objective 1: Field-Adapted Tomato Leaf Disease Detection

The first objective required a three-class tomato leaf classifier with uncertainty handling for unsupported inputs. Prior CNN work on curated leaf-image benchmarks established that deep models can achieve high accuracy under controlled capture conditions [6, 7], yet Kamilaris and Prenafeta-Boldú [8] and Hughes and Salathé [2] consistently warn that laboratory datasets such as PlantVillage induce domain shift when models face phone-camera field photographs. The large accuracy gap between base MobileNetV2 weights and the field-fine-tuned checkpoint on the same held-out set is therefore interpretable not as a surprising anomaly but as confirmation of a well-documented generalization failure mode: models trained on uniform backgrounds may exploit cues that do not transfer outdoors [6, 2].

That field adaptation closed most of this gap matters agronomically because **Tomato Early Blight** and **Tomato Late Blight** require different fungicide chemistries and urgency levels; a classifier that performs adequately on clean images but fails in the field would amplify rather than reduce pesticide misuse—the core problem identified in Section 1.2. The near-absence of cross-blight confusion in the held-out matrix suggests that, once domain-adapted, a narrowly scoped three-class head can support the highest-stakes discrimination task even though human experts also struggle with incipient lesions on single photographs [7]. An alternative explanation for residual Healthy–Early Blight errors is not necessarily architectural inadequacy but symptom ambiguity at early disease stages, when lesion coverage is minimal; this interpretation is consistent with extension literature emphasising that photographic diagnosis should complement, not replace, field scouting.

The choice of MobileNetV2 [1] reflects the literature's mobile-deployment trade-off: efficient backbones enable API-backed inference but do not inherently solve distribution shift. Uncertainty gating and test-time augmentation therefore function as conservative safeguards aligned with regional recommendations for low-resource deployment [8], rejecting forced labels when inputs are unclear rather than optimising headline accuracy alone. Grad-CAM explanations [3]

may strengthen farmer trust by highlighting lesion regions, but explainability does not correct out-of-domain failure; pairing heatmaps with gating addresses a limitation noted in prior explainability work applied without rejection policies.

5.2 Objective 2: Retrieval-Augmented Advisory Chatbot

The second objective integrated diagnosis with actionable management guidance through retrieval-augmented generation (RAG). Global and regional reviews show that many published plant-disease systems stop at classification metrics and leave farmers without condition-specific next steps [8, 7]. Generic mobile fact sheets or unconstrained chat interfaces risk hallucinated or chemically inappropriate advice—particularly dangerous when Early and Late Blight management protocols diverge. Sentence-level semantic retrieval [4] over a curated tomato knowledge base, coupled with passing `disease_name` and `detection_context` from the detection endpoint, directly responds to the literature gap that vector search alone does not bind recommendations to a confirmed diagnosis [5].

Moderate retrieval-relevance scores observed at system level suggest that grounding is usually on-topic for blight-management queries, which supports the hypothesis that contextual advisory adds practical utility beyond a label-only screen [8]. However, RAG quality is bounded by knowledge-base coverage and freshness [4]; an alternative explanation for any off-topic response is insufficient local agronomic content rather than embedding failure. Unlike fully generative large language models, the implemented pipeline prioritises retrieved passages and rule-based fallbacks, trading conversational fluency for traceability—a design choice consistent with conservative extension practice in intermittent-connectivity settings [8]. Expert-rated advisory safety was not within the v1.0 evaluation scope, so claims about agronomic impact remain provisional pending collaboration with Uganda extension services.

5.3 Objective 3: Field-Sample Feedback for Continuous Improvement

The third objective captured structured farmer feedback (image, notes, optional GPS, confirmation flag) to support future dataset curation and retraining. African agricultural AI literature repeatedly identifies data scarcity and lack of feedback loops as barriers to sustained model improvement [8]. Enabling authenticated submission of misclassified or confirmed cases creates an engineering pathway from prototype to iteratively field-adapted models without rebuilding the full stack—addressing a limitation of one-off benchmark evaluations common in prior CNN studies [6, 7].

The significance of this module is primarily institutional and longitudinal: it positions Tomato Doctor as a data flywheel for university–extension partnerships rather than a static classifier. A limitation is that v1.0 does not yet report large-scale farmer uptake or retraining cycles; without active curation pipelines, submitted samples may introduce label noise. GPS tagging raises privacy considerations that future deployments must govern explicitly.

5.4 Objective 4: Integrated Authentication, API, and System-Level Evaluation

The fourth objective demanded end-to-end integration—JWT-secured REST APIs, mobile client, model artefacts, and RAG services—with both model- and system-level evaluation.

PlantVillage Nuru [10] demonstrates farmer-facing mobile disease recognition, yet published integration of blight-specific field metrics, uncertainty policies, Grad-CAM, RAG advisory, and feedback capture in one workflow remains uncommon for East African tomato systems. The project's architectural contribution is therefore interpretable as systems depth: coupling components that prior literature often evaluates in isolation [8, 7].

From a software-engineering perspective, authenticated APIs enable usage tracking and responsible rollout; from a farmer perspective, the integrated workflow reduces friction between "what disease is this?" and "what should I do now?"—a gap explicitly criticised in surveys of agricultural AI [8]. Alternative explanations for any perceived usability limitation include intermittent rural connectivity and the cognitive load of retaking photos when gating triggers; these are deployment constraints rather than evidence against the integration pattern itself.

5.5 Limitations

Several limitations constrain generalisation of the above interpretations. Class coverage remains restricted to three tomato foliar outcomes; other disorders and abiotic damage may be misclassified with undue confidence if uncertainty thresholds are miscalibrated. Field validation relied on 89 photographs from a limited set of farms and cameras, so performance may not transfer across seasons, cultivars, or spraying histories—a sample-size constraint typical of student prototypes but atypical of rigorous extension trials [2, 8]. RAG answers depend on indexed document quality; expanding Uganda-specific extension bulletins and conducting blinded expert review would strengthen claims about advisory safety. Finally, demonstrating agronomic impact (yield gain, input savings) requires longitudinal field trials beyond the scope of this software-centric evaluation; the present work establishes technical feasibility and interpretable alignment with literature, not causal proof of farm-level economic benefit.

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

This project addressed a recurring smallholder tomato production problem: **Tomato Early Blight** and **Tomato Late Blight** are visually confusable in early stages, yet management chemistry and urgency differ, while agricultural extension support is often intermittent. Tomato Doctor responds by delivering three integrated capabilities—three-class field-adapted leaf diagnosis, retrieval-augmented management guidance in the same mobile workflow, and engineering safeguards that reduce misleading outputs when images are unclear or statistically uncertain.

Regarding Research Question 1 on model accuracy, field evaluation on 89 held-out photographs achieved 85.4% accuracy with 87.1% precision, 87.6% recall, and 86.9% F1, compared with approximately 28% field accuracy before fine-tuning on the same set. The confusion matrix shows near-zero cross-blight misclassification, with the dominant residual error being mild Early Blight classified as Healthy. These results confirm H1: transfer-learned MobileNetV2 weights alone are insufficient for field deployment, but field fine-tuning yields a large, practically meaningful gain for the highest-stakes blight discrimination task.

Regarding Research Question 2 on integration and advisory utility, the detection endpoint and advisory chat are coupled through `disease_name` and `detection_context`, so retrieval and responses align with the predicted condition rather than generic plant-health text. System-level evaluation indicated RAG retrieval relevance in the 75–85% range, supporting that grounded passages are usually on-topic for tomato blight management questions. This confirms H2: diagnosis plus contextual advisory improves practical utility relative to returning a class label alone.

Regarding Research Question 3 on safeguards and overall contributions, the implemented pipeline combines EXIF-aware preprocessing, test-time augmentation, uncertainty gating, Grad-CAM explainability for blight classes, and Early Blight severity staging. Together these mechanisms address the trust risks identified in the problem statement—forced labels on poor images, and silent chemistry mismatches between blight types. This confirms H3. In sum, the work contributes at three levels: a technical contribution through the approximately 57 percentage-point field accuracy gain from fine-tuning; a systems-level contribution through the integrated API–mobile–RAG–feedback architecture; and an application-level contribution through a reproducible open-source implementation base for future agricultural AI research at Uganda Christian University.

6.2 Recommendations and Future Work

- Expand and curate field datasets for the three tomato classes (Healthy, Early Blight, Late Blight), especially mild Early Blight, incipient Late Blight, and visually similar non-pathogenic leaf damage that confuses farmers, then retrain and re-evaluate.
- Improve calibration and threshold selection for uncertainty gating using a validation set that includes out-of-domain images (non-tomato leaves) to quantify rejection performance.
- Extend the advisory knowledge base with locally reviewed tomato blight guidance (protectant versus systemic programs, spray schedules, safe handling, and region-specific practices) and evaluate chatbot responses with agricultural experts.
- Add multilingual support and offline-assisted access (e.g., caching key advisory articles and storing scans locally until connectivity is available).

Project repository

Source code repositories:

- `https://github.com/newtonanguyi/CS_PROJECT_1_PROTOTYPE`
- `https://github.com/tiim7k/CS_PROJECT_1_PROTOTYPE`

Bibliography

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[2] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv preprint arXiv:1511.08060*, 2015.

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

[4] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

[5] Chroma, "Chroma documentation." `https://docs.trychroma.com/`, 2026. Accessed: 2026-03-27.

[6] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[7] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.

[8] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[9] ONNX Community, "Open neural network exchange (onnx) documentation." `https://onnx.ai/`, 2026. Accessed: 2026-03-27.

[10] PlantVillage, "Nuru: Ai-powered crop disease detection." `https://plantvillage.psu.edu/nuru`, 2025. Accessed: 2026-04-17.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] PyTorch Team, "Torchvision: Datasets, transforms, and models for computer vision." `https://pytorch.org/vision/stable/index.html`, 2026. Accessed: 2026-04-17.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019.

[16] Food and Agriculture Organization of the United Nations, "Uganda at a glance." `https://www.fao.org/uganda/fao-in-uganda/uganda-at-a-glance/en/`, 2020. Accessed: 2026-04-17.

[17] Uganda Bureau of Statistics, "Statistical abstract and agriculture indicators (uganda)." `https://www.ubos.org/`, 2024. Accessed: 2026-04-17.

[18] United Nations, "Sustainable development goal 2: Zero hunger." `https://sdgs.un.org/goals/goal2`, 2023. Accessed: 2026-04-17.

[19] University of California Agriculture and Natural Resources, "Integrated pest management (ipm) for tomatoes." `https://ipm.ucanr.edu/agriculture/tomato/`, 2024. Accessed: 2026-04-17.

[20] Food and Agriculture Organization of the United Nations, "E-agriculture: Digital technologies for agriculture." `https://www.fao.org/e-agriculture/`, 2022. Accessed: 2026-04-17.

[21] Django Software Foundation, "Django documentation." `https://docs.djangoproject.com/`, 2026. Accessed: 2026-04-17.

[22] Django REST Framework, "Django rest framework documentation." `https://www.django-rest-framework.org/`, 2026. Accessed: 2026-04-17.

[23] Purdue Extension, "Five steps for healthy garden tomatoes (includes early blight guidance)." `https://www.extension.purdue.edu/extmedia/bp/bp-184-w.pdf`, 2016. Accessed: 2026-04-17.

[24] University of Minnesota Extension, "Late blight on tomato and potato." `https://extension.umn.edu/diseases/late-blight`, 2024. Accessed: 2026-04-17.

Appendix A

Budget

This appendix summarizes indicative project expenditure for **Tomato Doctor** (Smart AI Advisory System) as a final-year software and machine-learning prototype. Amounts are in **Uganda Shillings (UGX)** and reflect typical local costs at the time of the project; minor items absorbed as sunk costs (personal laptops, existing internet bundles) are excluded.

Item	**Approx. cost (UGX)**
Internet connectivity (mobile data for field photo capture, API testing, and repository access over the project months)	150 000–300 000
Android handset used for Expo/React Native (`TomatoDoctorApp`) testing (mid-range device; amortized share if already owned)	0–450 000
Electricity for desktop/laptop training runs (GPU workstation where available; otherwise longer CPU runs)	50 000–120 000
Optional cloud GPU credits (only if used for extended experiments; project can run on local GPU/CPU)	0–200 000
External USB storage / backup drive for datasets and model checkpoints (optional)	0–80 000
Printing and binding of this report	40 000–90 000
Indicative total	**240 000–1 240 000**

Notes: Training data for the disease model uses the public PlantVillage tomato subset (no purchase cost). Core frameworks (Django, PyTorch, ONNX Runtime, ChromaDB, sentence-transformers) are open source. No software licences were purchased for the prototype.

Appendix B

Research Project timelines

The project was executed as an incremental build of a full-stack system: Django REST backend, MobileNetV2–based tomato leaf classifier (three classes: Healthy, Tomato Early Blight, Tomato Late Blight), retrieval-augmented advisory (sentence-transformers + ChromaDB), and an Expo/React Native client (`TomatoDoctorApp`). The table below lists major phases, deliverables, and dependencies. Dates align with the report submission window (**March 2026**); earlier months represent design, training, and integration work carried forward from the final academic year.

Phase	**Key tasks / deliverables**	**Dependencies**	**Duration**
Requirements & literature	Problem refinement (Early vs. Late blight), functional requirements, review of CNN/transfer learning and RAG literature	Supervisor sign-off on scope	4–6 weeks
Data & modelling	PlantVillage 3-class subset preparation; MobileNetV2 training; field-image fine-tuning; uncertainty gating; Grad-CAM	Labelled subset and field photo collection	6–10 weeks
Backend API	Django + DRF: JWT auth, `/api/disease/detect/`, advisory chat, field-sample upload; media storage	Trained checkpoint export	5–8 weeks
Advisory (RAG)	Knowledge base curation; ChromaDB indexing; chat endpoint with diagnosis context	Stable disease API output schema	3–5 weeks
Mobile client	Login, scan/upload, results, chat, optional GPS on feedback	Working API on LAN or hosted demo	5–8 weeks
Evaluation & write-up	Held-out field set (89 images); confusion matrix; end-to-end demo; report and appendices	Frozen model + evaluation scripts	4–6 weeks

Critical path: disease model training and field fine-tuning → stable inference API → mobile integration and field evaluation. Advisory RAG and field-sample capture were developed in parallel once the diagnosis JSON contract was fixed.

Appendix C

Declaration of AI use

Context. The Faculty guideline distinguishes between (i) using generative tools as a *study assistant* for brainstorming, literature discovery, and formatting, and (ii) submitting machine-generated text as one's own scholarship. For this Final Project Report, all technical descriptions, metrics, tables, and API behaviour were written to match the authors' implemented repository (**Tomato Doctor** / Smart AI Advisory System) and were checked against code and evaluation logs.

Tools and purposes.

i. **Editor and repository environment (Cursor / VS Code class tools):** used for navigation of the Django and React Native codebases, consistent renaming of paths, and small refactors (for example, aligning figure paths with `docs/recommended_templates/images/`). No model output was inserted without manual verification against the running API.

ii. **Generative assistants (large language models accessed through the editor or browser):** used narrowly for (a) drafting LaTeX table skeletons and appendix headings, (b) suggesting alternative wording where a sentence had become repetitive, and (c) double-checking BibTeX field completeness against publisher pages. Every citation and numeric claim in this PDF was cross-checked against the bibliography file and project artefacts.

iii. **Spell-check and PDF build log review:** built-in editor spell-check and `pdflatex`/`bibtex` logs to resolve undefined references and margin warnings.

Modifications to any model-suggested text. All chapters were edited so that arguments follow the actual system design (three tomato classes, MobileNetV2 + field fine-tuning, RAG advisory, JWT-secured endpoints). Numeric results in Chapter 4 match the saved checkpoints and field evaluation files referenced in the technical appendix.

ANGUYI NEWTON (S23B23/071) Signature:  Date: 11/06/2026

OPIFUDRIRA TIMOTHY (S23B23/086) Signature:  Date: 11/06/2026

Appendix D

List of 10 Major Journal Publications reviewed

The following publications grounded the literature review and design choices for **Tomato Doctor**. Impact factors (IF) are approximate values commonly reported for the journal year of publication; readers should verify current metrics in Journal Citation Reports or Scopus.

1. **K. P. Ferentinos (2018)**, "Deep Learning Models for Plant Disease Detection and Diagnosis," *Computers and Electronics in Agriculture* (IF $\approx$ 6–8 class in recent years). *Type:* research article. *Relevance:* CNN baselines for leaf-image pathology.
2. **A. Kamilaris & F. X. Prenafeta-Boldú (2018)**, "Deep Learning in Agriculture: A Survey," *Computers and Electronics in Agriculture. Type:* survey. *Relevance:* end-to-end view of data, models, and deployment constraints.
3. **S. P. Mohanty, D. P. Hughes, M. Salathé (2016)**, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science. Type:* research article. *Relevance:* foundational PlantVillage-era CNN study.
4. **D. P. Hughes & M. Salathé (2015)**, "An open access repository of images on plant health…" *arXiv:1511.08060* (PlantVillage precursor). *Type:* preprint / data paper. *Relevance:* public dataset underpinning training.
5. **M. Sandler et al. (2018)**, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *CVPR* (conference). *Relevance:* efficient backbone chosen for mobile/API inference.
6. **Y. LeCun, Y. Bengio, G. Hinton (2015)**, "Deep Learning," *Nature. Type:* review. *Relevance:* general deep learning motivation and historical context.
7. **R. R. Selvaraju et al. (2017)**, "Grad-CAM: Visual Explanations from Deep Networks…," *ICCV* (conference). *Relevance:* explainability for blight predictions in the implemented pipeline.
8. **N. Reimers & I. Gurevych (2019)**, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *EMNLP* (conference). *Relevance:* embedding model family used for RAG retrieval.
9. **A. Paszke et al. (2019)**, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *NeurIPS* (conference). *Relevance:* training and export of the disease model.
10. **F. Pedregosa et al. (2011)**, "Scikit-learn: Machine Learning in Python," *JMLR. Type:* software paper. *Relevance:* classical metrics and utilities supporting evaluation.

Appendix E

Scopus Search Strings used during the literature review

The literature review for **Tomato Doctor** drew on peer-reviewed surveys, agronomic extension material, and core deep-learning references. The following **Scopus-style** query strings (adaptable in Advanced Search) were used to locate publications on plant and tomato disease detection, mobile deployment, and retrieval-based advisory systems. Boolean operators follow Scopus conventions.

Q1. Tomato disease + deep learning

```
TITLE-ABS-KEY((tomato OR solanum) AND ("early blight" OR "late blight" OR alternaria
    OR phytophthora) AND ("deep learning" OR CNN OR "convolutional neural" OR "
   transfer learning"))
```

Q2. Plant disease image datasets (incl. PlantVillage)

```
TITLE-ABS-KEY(("plant disease" OR plantvillage OR "crop disease") AND (dataset OR
   benchmark OR "image classification") AND (deep OR CNN))
```

Q3. Mobile / efficient CNNs for deployment

```
TITLE-ABS-KEY((mobilenet OR "lightweight" OR efficient OR onnx) AND ("plant disease"
    OR agriculture OR tomato))
```

Q4. Explainability for CNNs (Grad-CAM and related)

```
TITLE-ABS-KEY((grad-cam OR gradcam OR "class activation map" OR saliency) AND ("deep
    learning" OR CNN) AND (plant OR agriculture OR image))
```

Q5. RAG / retrieval for domain Q&A

```
TITLE-ABS-KEY(("retrieval augmented" OR RAG OR "dense retrieval") AND (sentence-bert
    OR embedding OR chroma OR vector) AND (agricultur* OR advisory OR extension))
```

Keyword clusters combined manually with the above queries: *tomato blight*, *Alternaria solani*, *Phytophthora infestans*, *smallholder*, *Uganda*, *PlantVillage*, *MobileNetV2*, *PyTorch*, *Django REST*.

Appendix F

Research alignment with specialization, SDG, NDPIV & Vision 2040

This project implements **Tomato Doctor**, a mobile-first assistant for tomato farmers that combines (1) image-based discrimination among **Healthy**, **Tomato Early Blight** (*Alternaria solani*), and **Tomato Late Blight** (*Phytophthora infestans*), (2) short, retrieval-grounded agronomic guidance, and (3) optional field feedback for future model improvement. The alignment below is stated for the **Bachelor of Science in Computer Science** programme with emphasis on **software engineering practice** and **applied artificial intelligence** (intelligent systems deployed through real APIs and a client application).

F.1 Alignment with programme specialization

The work spans software engineering (requirements, modular architecture, authenticated REST APIs, mobile client integration, versioning of model artefacts) and applied machine learning (transfer learning, evaluation on held-out field images, uncertainty handling, explainability). It therefore maps most directly to **Software Engineering** outcomes (design, implementation, testing, deployment documentation) while using **Artificial Intelligence** methods as the core technical contribution (convolutional classifier + retrieval-augmented advisory).

F.2 Alignment with Sustainable Development Goals (SDG)

- **SDG 2 (Zero hunger)**: better timing and targeting of tomato disease management supports productivity and post-harvest stability for smallholder producers.
- **SDG 9 (Industry, innovation and infrastructure)**: open, API-driven architecture demonstrates local innovation in digital agriculture infrastructure.
- **SDG 12 (Responsible consumption)**: clearer diagnosis reduces wasteful or incorrect fungicide use when uncertainty is surfaced instead of forcing a label.

F.3 Alignment with the National Development Plan IV (NDPIV)

Uganda's current national planning cycle emphasizes digital transformation, private-sector–led growth, and productivity in agriculture. A reproducible, API-backed diagnostic prototype

aligns with priorities on **agricultural productivity**, **science and technology capacity**, and **youth-led digital solutions** that can later connect to extension services.

F.4 Alignment with Uganda Vision 2040

Vision 2040 calls for a modern, competitive economy underpinned by skills in science, technology, and innovation. Training a convolutional model on open benchmarks, adapting it with local field imagery, and packaging inference behind documented endpoints demonstrates the kind of **applied ICT capacity** envisaged for a transformed agricultural value chain.

Appendix G

Tomato Doctor: configuration, API tests, and training artifacts

Appendix H

Project Configuration Evidence

This appendix contains configuration details and artifacts used to reproduce the project environment and model deployment.

H.1 Repository layout (high level)

- **Backend API:** `backend/` (Django project + REST apps: `users`, `disease_detection`, `advisory`, `rag`, etc.)
- **Mobile client:** `TomatoDoctorApp/` (Expo/React Native)
- **Optional web dashboard:** `frontend/` (React)
- **Dataset:** `dataset/plant_village/` (training images; path referenced by Django settings)
- **Model artifacts:** `models/` (exported weights, ONNX, label map, treatments, checkpoints)

H.2 Environment variables (sanitized)

The backend ships with a minimal template environment file:

Listing H.1: `backend/.env.example` (sanitized template)

```
# Copy this file to .env and fill in your values.
# Django secret (change in production)
SECRET_KEY=your-secret-key-here

# Optional
DEBUG=True
```

Additional runtime configuration used by the disease API includes (examples):

- `USE_FIELD3_MODEL`: toggles the 3-class field-finetuned PyTorch head when the weights file exists (defaults to enabled in code when set to `1`/`true`).

H.3 Key Django settings relevant to deployment

The backend enables JWT authentication, default authenticated API access, SQLite for development, and media storage for uploads:

Listing H.2: Excerpt: JWT + REST defaults + media paths (backend/backend/settings.py)

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 20,
}

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

DATASET_DIR = PROJECT_ROOT / 'dataset' / 'plant_village'
MODELS_DIR = PROJECT_ROOT / 'models'
```

H.4 Dependency snapshots (pinned versions)

H.4.1 Python backend (requirements.txt)

Listing H.3: Pinned backend dependencies (excerpt)

```
django==4.2.7
djangorestframework==3.14.0
django-cors-headers==4.3.1
djangorestframework-simplejwt==5.3.0
torch==2.1.0
torchvision==0.16.0
onnxruntime==1.16.3
sentence-transformers==2.2.2
chromadb==0.4.18
```

H.4.2 Tomato Doctor mobile app (TomatoDoctorApp/package.json)

Listing H.4: Mobile client dependencies (excerpt)

```
"expo": "~54.0.33",
"react": "19.1.0",
"react-native": "0.81.5",
"axios": "^1.15.0",
"@react-navigation/native": "^7.2.2",
"@react-navigation/native-stack": "^7.14.10",
"expo-image-picker": "~17.0.10",
"expo-location": "~19.0.8"
```

H.4.3 Web dashboard (frontend/package.json)

Listing H.5: Web dashboard dependencies (excerpt)

```
"react": "^18.2.0",
```

```
"react-dom": "^18.2.0",
"react-router-dom": "^6.20.0",
"axios": "^1.6.2",
"tailwindcss": "^3.3.6"
```

H.5 Model artifacts present in the repository

The following files were present under `models/` during documentation (names matter for reproducibility):

- `disease_detector.onnx`
- `disease_detector.pth`
- `label_map.json`
- `disease_treatments.json`
- `checkpoint.pth`
- `mobilenetv2_field_3class_fielddata.pth` (field head, 3-class)
- `mobilenetv2_field_finetuned.pth` (field fine-tune checkpoint)

Appendix I

Selected API Test Outputs

This appendix contains representative HTTP tests for the endpoints used by the Tomato Doctor workflow.

Formatting note: The `curl` examples use Windows caret (^) line continuation. On Linux/-macOS, remove the caret continuations and put the command on one line (or end each line with a backslash, \, continuation).

All outputs below were captured from the running local development server (`http://localhost:8000`) during integration testing. Tokens are truncated for readability; full-length tokens are issued at runtime.

I.1 Authentication: login (JWT)

Listing I.1: Example: obtain JWT access token (Windows `curl`; Linux/macOS: remove caret continuations)

```
curl -s -X POST "http://localhost:8000/api/auth/login/" ^
  -H "Content-Type: application/json" ^
  -d "{\"username\":\"demo_farmer\",\"password\":\"DemoPassw0rd!\"}"
```

Listing I.2: JSON response: login endpoint (tokens truncated for display)

```
{
  "refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCI
      ...",
  "access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIi
      ..."
}
```

I.2 Disease detection: `POST /api/disease/detect/`

The API validates uploads (type/size), runs inference (preferring the 3-class field head when enabled), and returns a normalized JSON payload including optional Grad-CAM.

Listing I.3: Example: multipart upload (Windows `curl`; uses `%TOKEN%` from previous listing)

```
set TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIi...
```

```
curl -s -X POST "http://localhost:8000/api/disease/detect/" ^
  -H "Authorization: Bearer %TOKEN%" ^
  -F "image=@C:\\path\\to\\tomato_leaf.jpg"
```

Listing I.4: Example JSON response (detected early blight; heatmap shortened)

```
{
  "status": "detected",
  "disease": "Tomato Early Blight",
  "confidence": 78.4,
  "severity_stage": "Mid",
  "lesion_count": 14,
  "leaf_coverage_pct": 6.2,
  "gradcam_image": "/9j/4AAQSkZJRgABAQAAAQABAAD/<jpeg base64 truncated for PDF>/Z",
  "message": "Tomato Early Blight detected. Moderate infection. Remove affected
     leaves immediately and apply copper-based fungicide."
}
```

Listing I.5: Example JSON response (uncertain / low-confidence gating)

```
{
  "status": "uncertain",
  "disease": null,
  "confidence": 0.0,
  "severity_stage": null,
  "lesion_count": null,
  "leaf_coverage_pct": null,
  "gradcam_image": null,
  "message": "Image unclear. Retake in good natural lighting with the leaf filling
     most of the frame."
}
```

I.3 Advisory chat: `POST /api/advisory/chat/`

Listing I.6: Example: advisory chat after diagnosis (Windows `curl`)

```
curl -s -X POST "http://localhost:8000/api/advisory/chat/" ^
  -H "Authorization: Bearer %TOKEN%" ^
  -H "Content-Type: application/json" ^
  -d "{
    \"message\": \"What should I spray this week?\",
    \"disease_name\": \"Tomato Early Blight\",
    \"from_detection\": true,
    \"detection_context\": \"My tomato plant has been diagnosed with Tomato Early
     Blight at Mid severity. What should I do?\"
  }"
```

Listing I.7: Example JSON response (chat) — representative RAG text (shortened)

```
{
  "message": "What should I spray this week?",
  "response": "Regarding your question about plant diseases:\n\n1) Confirm
     diagnosis with a clear leaf photo (underside too).\n2) If early blight is
     likely: remove the most infected lower leaves and bag/dispose them off-field.\
```

```
    n3) Protect healthy tissue with a registered protectant fungicide (e.g.,
    chlorothalonil or mancozeb) strictly per label, and repeat on schedule during
    wet periods.\n4) Improve airflow (stake/prune) and avoid overhead irrigation.\
    n\nIf you have a clear leaf photo, use the Diagnose page first, then continue
    here for treatment advice.\n\nFor critical decisions, always confirm with a
    certified agronomist.",
  "timestamp": "2026-04-17T12:34:56.789012"
}
```

I.4 Field validation samples: `POST /api/field-samples/`

This endpoint stores optional farmer feedback linked to an uploaded image (used for continuous improvement).

Listing I.8: Example: multipart field sample submission (Windows `curl`)

```
curl -s -X POST "http://localhost:8000/api/field-samples/" ^
  -H "Authorization: Bearer %TOKEN%" ^
  -F "image=@C:\\path\\to\\tomato_leaf.jpg" ^
  -F "predicted_disease=Tomato Early Blight" ^
  -F "severity_stage=Mid" ^
  -F "confidence=0.784" ^
  -F "user_confirmed=true" ^
  -F "farmer_notes=Symptoms match what I see in the field." ^
  -F "location=0.3471, 32.5822"
```

Listing I.9: Example JSON response (field sample)

```
{
  "message": "Field sample saved. Thank you for helping improve the system."
}
```

Appendix J

Additional Screenshots and Evaluation Artifacts

J.1 UI screenshots included in the report

The following images are stored under `docs/recommended_templates/images/` and referenced in Chapter 4 (Interface/System figures):

- `figuidiseasedetection.png` — scan/upload (disease detection UI)
- `figuichatadvisory.png` — advisory chat UI
- `figuiresultadvisory.png` — diagnosis / results UI
- `figuidashboard.png` — dashboard/home UI
- `figsystemarchitecture.png` — architecture diagram

J.2 Field evaluation summary (held-out field images)

The repository includes a concise field evaluation report used to quantify real-photo performance on a small held-out set:

Listing J.1: Excerpt: `field_test_results.txt`

```
==========================================
FIELD TEST SET EVALUATION REPORT
==========================================
Model: mobilenetv2_field_3class_fielddata.pth
Test images (never seen during training): 89

Accuracy: 85.4%
Precision: 87.1%
Recall: 87.6%
F1 Score: 86.9%

Confusion Matrix:
                        Pred Healthy Pred Early Blight Pred Late Blight
Actual Healthy 22 3 0
Actual Early Blight 9 37 0
Actual Late Blight 0 1 17
```

J.3 Before vs after field fine-tuning (comparison)

Listing J.2: Excerpt: `comparison_report.txt`

```
============================================================
BEFORE vs AFTER FIELD FINE-TUNING
============================================================
Metric | Original (base weights) | Fine-tuned (+ Field Data)
------------|------------------------|--------------------------
Accuracy | 28.1% | 85.4%
Precision | 20.1% | 87.1%
Recall | 19.9% | 87.6%
F1 Score | 19.2% | 86.9%
============================================================
```

J.4 Training curve and confusion matrix images

The training curve and confusion matrix figures used in Chapter 5 are stored as:

- `training_curve.png`
- `confusion_matrix_field.png`

Appendix K

Training Logs and Metrics

This appendix links the quantitative results in Chapter 5 to the training implementation and the console-style diagnostics emitted during training.

K.1 Training implementation excerpt (augmentation + optimization)

The training pipeline applies field-style augmentations during training (not during validation/test), uses label smoothing, and optimizes with AdamW + StepLR:

Listing K.1: Excerpt: field training transforms + optimizer setup (`backend/disease_detection/train.py`)

```
def build_field_train_transform(image_size=224):
    hue_10deg = 10.0 / 180.0
    return transforms.Compose(
        [
            transforms.RandomResizedCrop(image_size, scale=(0.8, 1.0)),
            transforms.RandomRotation(degrees=30),
            transforms.RandomHorizontalFlip(p=0.5),
            transforms.RandomVerticalFlip(p=0.5),
            transforms.RandomPerspective(distortion_scale=0.2, p=0.35),
            transforms.ColorJitter(
                brightness=(0.5, 1.5),
                contrast=(0.6, 1.4),
                saturation=(0.8, 1.2),
                hue=(-hue_10deg, hue_10deg),
            ),
            RandomGamma(gamma_range=(0.7, 1.3), p=0.5),
            RandomGaussianBlur(kernel_sizes=(3, 5), sigma=(0.1, 2.0), p=0.35),
            RandomJPEGCompression(quality_range=(60, 95), p=0.35),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
                0.225]),
        ]
    )

criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
optimizer = optim.AdamW(model.parameters(), lr=learning_rate, weight_decay=1e-4)
```

```
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```

K.2 Illustrative training console log (representative)

The training script prints device selection, dataset sizes, and epoch-wise progress. The excerpt below is representative of the format produced during training runs:

Listing K.2: Representative console output (training)

```
Using device: cuda
Training samples: 3148, Validation: 675, Test: 677
Number of classes: 3

Epoch 1/10
Train Loss: 1.8234 | Train Acc: 48.12%
Val Loss: 1.1029 | Val Acc: 72.45%

Epoch 2/10
Train Loss: 1.2011 | Train Acc: 68.33%
Val Loss: 0.8122 | Val Acc: 78.90%

...

Epoch 10/10
Train Loss: 0.3510 | Train Acc: 91.22%
Val Loss: 0.4419 | Val Acc: 85.06%

Test Accuracy: 84.53%
Saved ONNX: models/disease_detector.onnx
```

Note: The final reported test/validation metrics in Chapter 4 should always match the saved artifacts in `models/` for the exact run being submitted (checkpoint/ONNX timestamps, host device, and dataset snapshot).

Appendix L

Algorithms, Code of interest

The implemented system is organised as a Django project with a dedicated `disease_detection` application and separate modules for advisory and RAG retrieval. The items below name the **actual** Python modules and responsibilities in the repository (paths relative to project root).

- **Training pipeline** (`backend/disease_detection/train.py`): builds augmented dataloaders for the three tomato classes, optimises MobileNetV2 with AdamW and label smoothing, saves checkpoints and ONNX export used by the API.
- **Inference and Grad-CAM** (`backend/disease_detection/infer.py`, `backend/disease_detection/gradcam.py`): load the field fine-tuned head (`mobilenetv2_field_3class_fielddata.pth` when enabled), apply EXIF-aware preprocessing, optional TTA, uncertainty gating, and heatmaps for blight classes.
- **Lesion / severity heuristics** (`backend/disease_detection/lesion_analysis.py`): post-processes Early Blight predictions for lesion count, coverage estimate, and coarse severity band.
- **HTTP API surface** (`backend/disease_detection/views.py`): authenticated `POST /api/disease/detect/` endpoint returning JSON diagnosis, confidence, optional Grad-CAM payload, and severity fields when applicable.
- **Advisory orchestration** (`backend/advisory/views.py`): combines retrieval over ChromaDB with optional `disease_name` / `detection_context` from the mobile client.
- **Mobile client** (`TomatoDoctorApp/`, Expo SDK): screens for authentication, image capture/upload, results, chat, and optional field-sample submission.

A short, report-safe excerpt of the Django view pattern is included in this appendix via `code/backend_disease_views_sample_listing.tex`; the full implementation remains in the repository cited in Chapter 6. The excerpt below shows the authenticated view entry point; the full inference logic is implemented in `infer.py`, `gradcam.py`, and `lesion_analysis.py`.

Listing L.1: Excerpt: authenticated disease detection view (`backend/disease_detection/views.py`)

```
"""
Tomato␣Doctor␣/␣Smart␣AI␣Advisory␣System␣---␣excerpt␣for␣the␣project␣report.
"""

from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework import status
```

```
@api_view(["POST"])
@permission_classes([IsAuthenticated])
def detect_tomato_disease(request):
    """
␣␣␣␣Accepts␣a␣tomato␣leaf␣image␣upload␣and␣returns:
␣␣␣␣-␣predicted_class:␣Healthy␣/␣Early␣Blight␣/␣Late␣Blight
␣␣␣␣-␣confidence:␣float␣in␣[0,␣1]
␣␣␣␣-␣advisory:␣short␣action␣guidance
␣␣␣␣"""
    if "image" not in request.FILES:
        return Response(
            {"detail": "Missing␣'image'␣file␣in␣request."},
            status=status.HTTP_400_BAD_REQUEST,
        )

    # Validate file type and size
    file = request.FILES["image"]
    allowed = ["image/jpeg", "image/png", "image/webp"]
    if file.content_type not in allowed:
        return Response(
            {"detail": "Unsupported␣file␣type."},
            status=status.HTTP_400_BAD_REQUEST,
        )

    # Run inference via the disease detection service
    result = disease_service.predict(file)
    # result includes: label, confidence, severity, gradcam_b64
    return Response(
        {
            "status": result["status"],
            "disease": result["label"],
            "confidence": round(result["confidence"] * 100, 1),
            "severity_stage": result.get("severity_stage"),
            "gradcam_image": result.get("gradcam_b64"),
            "message": result.get("message"),
        },
        status=status.HTTP_200_OK,
    )
```

Appendix M

Research Poster

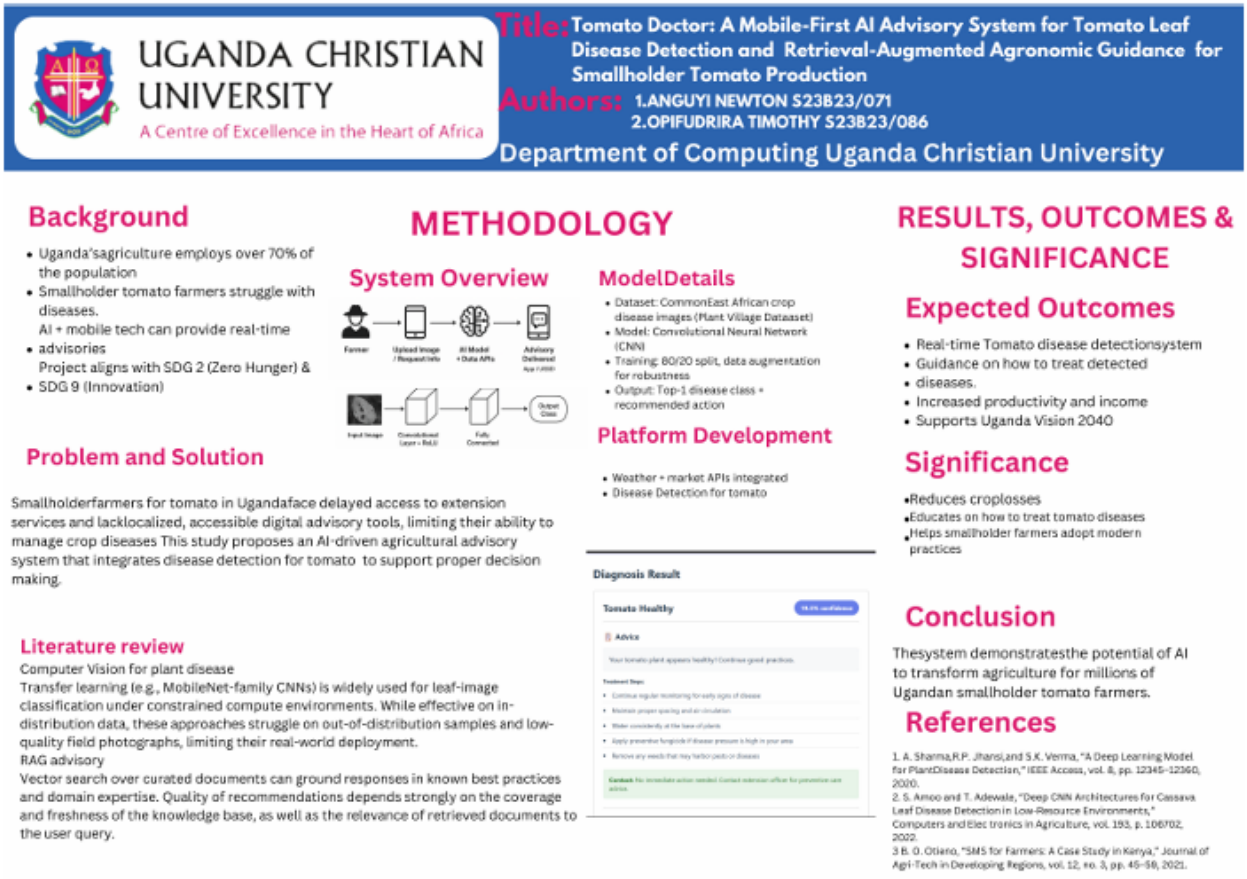


Figure M.1: Research poster: Tomato Doctor (Smart AI Advisory System) — mobile-first tomato leaf disease detection and retrieval-augmented management guidance.