

**Egerton University**

**Computer Science Department**

**Project Requirements Specification for MkulimaAid**

**(Crop Disease Detection System)**

**Prepared by: Fabius Lihanda Achevi**

**Registration Number: S13/04376/21**

**Date: January 2024**

Table of Contents

[**Chapter 1: Introduction** 3](#_Toc188528292)

[**1.1** **Background Information** 3](#_Toc188528293)

[**1.2 Current Challenges** 3](#_Toc188528294)

[**1.3 Opportunities for Improvement** 3](#_Toc188528295)

[**1.4 Purpose** 4](#_Toc188528296)

[**1.5 Objectives** 4](#_Toc188528297)

[**1.6 Scope** 6](#_Toc188528298)

[**Chapter 2: Requirements** 7](#_Toc188528299)

[**2.1 Functional Requirements** 7](#_Toc188528300)

[**2.2 Non-Functional Requirements** 8](#_Toc188528301)

[**Chapter 3: Architectural Design** 10](#_Toc188528302)

[**3.1 Enhanced Entity-Relationship (EER) Diagram** 10](#_Toc188528303)

[**3.2 Flowchart for the Crop Disease Subsystem** 11](#_Toc188528304)

[**3.3 Use Case Diagram** 12](#_Toc188528305)

[**3.4 Frontend and Backend Tools** 14](#_Toc188528306)

[**3.5 Integration Tools** 15](#_Toc188528307)

[**3.6 Hosting Tools** 15](#_Toc188528308)

[**Chapter 4: Project Timeline and Gantt Chart** 16](#_Toc188528309)

[**Project Phases** 16](#_Toc188528310)

[**Task Dependencies** 16](#_Toc188528311)

[**Timeline** 17](#_Toc188528312)

[**Milestones** 18](#_Toc188528313)

[**Chapter 5: Project Theme** 19](#_Toc188528314)

[**Key Aspects of the Project Theme** 19](#_Toc188528315)

[**Visual Theme and Design** 19](#_Toc188528316)

[**Chapter 6: Budget and Resources** 20](#_Toc188528317)

[**6.1 Budget** 20](#_Toc188528318)

[**Chapter 7: Resources** 22](#_Toc188528319)

[**7.1. Technical Resources** 22](#_Toc188528320)

[**7.2. Human Resources** 22](#_Toc188528321)

[**Chapter 8: Risk Management** 23](#_Toc188528322)

[**8.1. Identified Risks** 23](#_Toc188528323)

[**8.2. Mitigation Strategies** 23](#_Toc188528324)

[**8.3. Monitoring and Updates** 24](#_Toc188528325)

[**8.4. Early Integration Testing** 24](#_Toc188528326)

[**Chapter 9: Quality Assurance** 25](#_Toc188528327)

[**Testing Plan** 25](#_Toc188528328)

[**Reporting and Bug Tracking** 27](#_Toc188528329)

[**Chapter 10: Implementation Plan** 27](#_Toc188528330)

[**10.1. Deployment Plan** 27](#_Toc188528331)

[**10.2. Training Users** 28](#_Toc188528332)

[**10.3. User Manual** 28](#_Toc188528333)

[**Chapter 11: Maintenance and Support** 29](#_Toc188528334)

[**11.1. Maintenance Schedule** 29](#_Toc188528335)

[**11.2. System Monitoring** 30](#_Toc188528336)

[**11.3. Support Strategy** 30](#_Toc188528337)

[**11.4. Maintenance Policy** 31](#_Toc188528338)

[**Appendices** 31](#_Toc188528339)

# **Chapter 1: Introduction**

## **Background Information**

Agriculture is the backbone of many economies, especially in developing regions, where it provides livelihoods for the majority of the population. It plays a critical role in ensuring food security, economic stability, and sustainable development. Despite advancements in agricultural technologies, crop diseases remain one of the most pressing challenges for farmers worldwide. These diseases not only reduce yields but also compromise crop quality, leading to significant financial losses and disruptions in the food supply chain.

Smallholder farmers, who constitute a large proportion of the agricultural workforce, are particularly vulnerable to crop diseases. They often lack access to timely diagnostic tools, expert guidance, and affordable solutions. Traditional methods of disease identification involve physical inspections by agricultural experts, which are not only costly and time-consuming but also geographically and logistically restrictive. The absence of efficient and scalable solutions leaves farmers to rely on guesswork or informal advice, which can lead to ineffective or even harmful interventions.

The rapid advancement of technology presents an opportunity to address these challenges. By leveraging artificial intelligence, machine learning, and digital platforms, farmers can now access innovative tools that provide real-time disease detection, actionable insights, and a platform for community engagement. This integration of technology into agriculture, often referred to as "smart farming," has the potential to revolutionize how farmers manage crop health and productivity.

## **1.2 Current Challenges**

1. **Delayed Disease Identification:** Farmers often detect diseases only after they have spread extensively, leading to significant crop damage and reduced yields.
2. **Limited Access to Expert Advice:** In many rural areas, agricultural extension officers are scarce, making it difficult for farmers to receive timely professional guidance.
3. **High Costs of Diagnosis:** Traditional diagnostic services, such as laboratory tests or expert consultations, are often unaffordable for smallholder farmers.
4. **Lack of Awareness:** Farmers may not be familiar with the symptoms, causes, or prevention methods for various crop diseases, leaving them ill-prepared to address outbreaks.
5. **Inadequate Community Support:** There is limited collaboration among farmers to share experiences, solutions, and best practices for managing crop diseases.
6. **Data Scarcity:** Farmers often lack access to data-driven insights, such as disease trends or effective preventive measures, to inform their decision-making processes.

## **1.3 Opportunities for Improvement**

1. **AI-Powered Diagnostics:** Leveraging machine learning algorithms to analyze crop images can provide accurate and rapid disease detection, empowering farmers to act promptly.
2. **Digital Knowledge Repositories:** Centralized platforms can offer comprehensive resources, such as disease management guides, video tutorials, and expert articles, to educate farmers on sustainable practices.
3. **Community Engagement Platforms:** Interactive forums can foster collaboration between farmers and agricultural experts, enabling the exchange of knowledge and collective problem-solving.
4. **Real-Time Notifications:** Automated alerts about disease trends, weather changes, or preventive measures can keep farmers informed and proactive.
5. **Scalable and Inclusive Solutions:** Technology can bridge the gap between resource-rich and resource-poor farmers by providing affordable and accessible tools for crop health management.
6. **Data-Driven Insights:** Interactive analytics can help farmers identify patterns and make informed decisions to prevent outbreaks and improve productivity.

MkulimaAid aims to address these challenges and capitalize on these opportunities by creating a comprehensive web-based platform tailored to the needs of farmers. By combining AI-powered disease detection, educational resources, community engagement, and administrative tools, the platform seeks to transform agricultural practices, enhance crop health management, and ultimately improve the livelihoods of farmers.

## **1.4 Purpose**

The purpose of this document is to outline the technical and functional requirements for developing the MkulimaAid platform. This ensures all stakeholders, including developers, administrators, and end-users, have a clear understanding of the project's scope and expectations. By providing a structured approach to the system's design and implementation, this document serves as a blueprint to:

1. Align project objectives with the needs of smallholder farmers and agricultural experts.
2. Establish clear and measurable goals for each component of the platform, including disease detection, community engagement, and educational resources.
3. Define the technical architecture, functional capabilities, and system constraints to streamline development and ensure compatibility across devices and platforms.
4. Highlight current challenges in agriculture, such as limited disease identification tools and lack of timely advice, and demonstrate how MkulimaAid will address these gaps.
5. Promote collaboration among stakeholders by providing a shared understanding of the project timeline, deliverables, and success metrics.
6. Set a foundation for future enhancements, such as multilingual support, offline capabilities, and expanded disease coverage.
7. Provide transparency into the project’s direction, ensuring stakeholders are informed of progress and aligned with its vision.

## **1.5 Objectives**

#### **1.5.1 Main Objective**

To design and implement a web-based platform, MkulimaAid, that leverages artificial intelligence for accurate crop disease identification and management, empowering farmers to improve agricultural productivity and sustainability.

#### **1.5.2 Specific Objectives**

1. **Real-Time Disease Detection**

Develop an AI-powered system that identifies crop diseases from uploaded images with high accuracy, providing immediate and actionable results.

1. **Educational Resource Integration**

Create a repository of detailed resources on crop diseases, including symptoms, causes, prevention, and control measures, tailored for farmers.

1. **Community Engagement**

Establish an interactive Q&A forum for farmers and agricultural experts to exchange knowledge, experiences, and solutions to crop-related challenges.

1. **Administrator Tools**

Develop dashboards for administrators to manage content, track disease trends, oversee community activity, and generate system reports.

1. **Accessibility and Inclusivity**

Ensure the platform is user-friendly, supporting farmers with varying levels of digital literacy and providing multilingual support in future phases.

1. **Data-Driven Insights**

Incorporate interactive visualizations and analytics to help farmers make informed decisions about disease prevention and farm management.

1. **Trending Disease Insights**

Highlight emerging crop diseases based on regional data, user activity, and seasonality, enabling proactive disease management.

1. **Notifications and Alerts**

Implement a robust notification system to keep users informed about disease trends, platform updates, and answered forum questions.

1. **Scalability and Future Enhancements**

Design the system architecture to support future additions such as offline functionality, integration of weather and soil data, and global scalability.

1. **Sustainability Focus**

Promote sustainable farming practices by providing resources and tools that encourage environmentally friendly disease control methods.

1. **Mobile Responsiveness**

Ensure the platform is optimized for mobile devices, enabling farmers to access tools and resources on-the-go.

1. **Security and Privacy**

Implement robust security measures to protect user data, ensuring compliance with data protection standards and building user trust.

These objectives collectively define the scope of the MkulimaAid platform, ensuring it addresses the critical challenges faced by farmers while providing a foundation for continuous improvement and innovation.

## **1.6 Scope**

The scope of the MkulimaAid project defines its boundaries, limitations, and intended functionalities. This ensures a focused approach while laying the groundwork for future improvements and expansions.

**System Boundaries**

1. **Disease Identification**

The platform will utilize AI-powered image recognition to identify crop diseases based on user-uploaded images.

Disease identification is limited to crops and diseases covered in the PlantVillage dataset.

The system will not support detection of unknown or untrained diseases in its initial phase.

1. **Geographic Coverage**

The platform targets smallholder farmers in developing regions, where crop diseases significantly impact food security.

Expansion to other geographic regions will be considered in future phases based on adoption and feedback.

1. **Language Support**

Initially, the platform will support English as the primary language.

Localization for additional languages will be explored in subsequent phases to improve accessibility for diverse users.

1. **Device and Connectivity Requirements**

The platform requires an internet connection and access to a smartphone or desktop with a web browser.

Offline capabilities, such as saving data for later uploads, are not included in the initial release but will be explored in future updates.

**Functional Scope**

1. **Core Features**

* **AI-Powered Disease Detection:** High-accuracy disease identification with a confidence score and recommended solutions.
* **Trending Insights:** Display of trending crop diseases and preventive measures based on regional and seasonal data.
* **Educational Resources:** Articles, tutorials, and videos covering crop disease prevention, soil management, and sustainable farming practices.
* **Community Q&A Forum:** Real-time discussions where farmers and experts can exchange knowledge and resolve queries.
* **Administrator Tools:** Dashboards for content management, user tracking, and platform maintenance.

1. **User Engagement**

* Real-time notifications for trending diseases, forum updates, and educational content.
* A feedback mechanism for users to share their experiences and suggest improvements.

**Limitations**

1. **Data Input Dependency**

The accuracy of disease detection depends on the quality and clarity of user-uploaded images.

External data, such as weather conditions or soil quality, will not be factored into disease diagnosis.

1. **Scalability Constraints**

Initial deployment focuses on a limited dataset and regions. Scalability to support a larger user base and diverse crops will require additional resources and funding.

1. **User Accessibility**

Digital literacy and access to modern technology may limit adoption among certain farmer demographics.

By defining the scope, the MkulimaAid project maintains a clear focus on its goals, ensuring efficient resource allocation while providing a strong foundation for growth and enhancements.

# **Chapter 2: Requirements**

# **2.1 Functional Requirements**

The functional requirements describe the specific functionalities and behaviors of the MkulimaAid platform to ensure it fulfills its purpose.

**User Authentication and Authorization**

* Users must create accounts to access the platform.
* Role-based access control: Farmers, admins, and experts will have different permissions.
* Secure login with password strength validation and account recovery options.

**AI-Powered Disease Detection**

* Enable users to upload clear images of affected crops for diagnosis.
* Analyze the images using a trained machine learning model to detect diseases with a confidence score.
* Provide actionable insights, including:
* Disease name.
* Symptoms and causes.
* Treatment and preventive measures (organic and chemical).

**Educational Content Management**

* Admins can create, edit, and delete articles, tutorials, and videos.
* Content should be categorized by topics such as sustainable farming, pest control, and soil management.

**Trending Diseases Section**

* Display trending crop diseases based on user activity, regional data, and seasonality.
* Allow admins to update or highlight trending diseases.

**Community Q&A Forum**

* Users can post questions, share experiences, and provide answers.
* Questions and answers should include user gravatars, usernames, and timestamps.
* Admins can moderate content to ensure quality and relevance.

**Admin Dashboard**

* Provide tools for managing users, educational resources, and trending insights.
* Include analytics and visualizations:
* Number of users registered per week/month.
* Most identified diseases.
* Generate system activity reports.

**Notifications**

* Notify users of disease trends, answered forum questions, and newly uploaded content.
* Enable admins to send targeted alerts to specific user groups or regions.

**Profile Management**

* Allow users to update personal details, change passwords, and upload/remove profile pictures.
* Display user profiles with gravatars and activity summaries.

**Feedback Mechanism**

* Provide a form for users to submit feedback or report issues.
* Admins can view and address feedback via the dashboard.

## **2.2 Non-Functional Requirements**

The non-functional requirements outline the quality attributes and performance benchmarks for the platform.

**Performance**

* Ensure the system can handle concurrent uploads and diagnoses for up to 1,000 users per minute.
* Provide responses for disease identification within 5 seconds of image upload.

**Scalability**

* Design the platform to support future expansion, including larger datasets and a global user base.

**Reliability and Availability**

* Maintain 99.9% uptime for the platform to ensure consistent availability.
* Implement automated backups to prevent data loss.

**Security**

* Use HTTPS for secure data transmission.
* Encrypt user passwords and sensitive information.
* Implement protections against common vulnerabilities such as SQL injection and cross-site scripting (XSS).

**Usability**

* Ensure the platform is intuitive and easy to navigate for users with varying levels of digital literacy.
* Support responsive design for seamless use across devices (desktop, tablet, and mobile).

**Localization**

* Design the system with future support for multiple languages.

**Accessibility**

* Follow web accessibility standards (e.g., WCAG 2.1) to accommodate users with disabilities.

**Maintainability**

* Use modular and well-documented code for ease of updates and debugging.
* Enable admins to perform basic maintenance tasks via the dashboard.

**Compliance**

* Adhere to data protection regulations, such as GDPR or local equivalents, to ensure user data privacy.

**Interoperability**

* Allow integration with third-party tools such as agricultural databases or government farming initiatives in future phases.

# **Chapter 3: Architectural Design**

The architectural design of MkulimaAid encompasses various elements that collectively define the system's functionality, database structure, and workflow. This section provides a detailed overview of the system's design through the following diagrams:

1. Enhanced Entity-Relationship (EER) Diagram – Represents the database schema.
2. Flowchart – Depicts the process flow of the crop disease recognition subsystem.
3. Use Case Diagram – Highlights user interactions with the system.

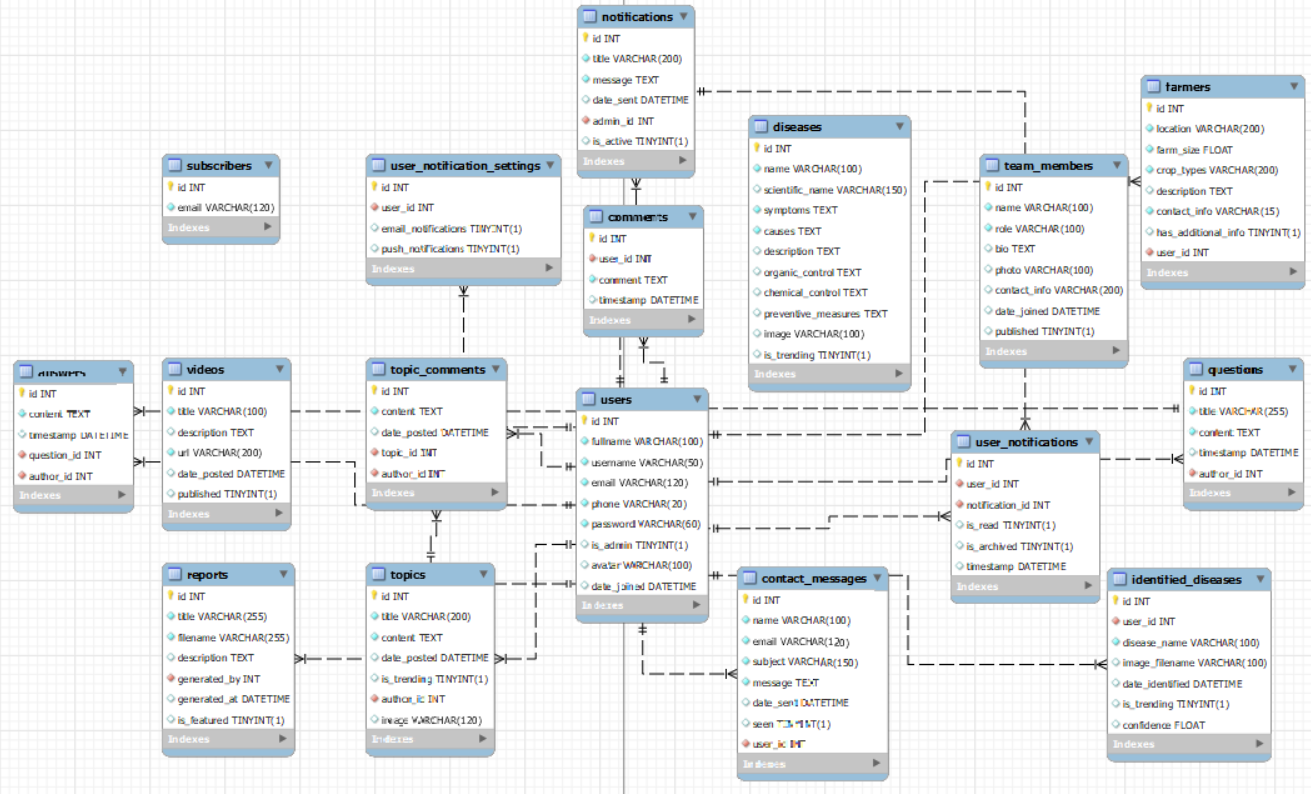
## **3.1 Enhanced Entity-Relationship (EER) Diagram**

The Enhanced Entity-Relationship (EER) Diagram illustrates the database structure of MkulimaAid, showing how data is stored, linked, and managed across the system. Key entities include:

* **Users:** Stores user details, such as usernames, roles, and profile information.
* **Diseases:** Contains information about trending crop diseases, including symptoms, control measures, and images.
* **IdentifiedDiseases:** Logs details of crop diseases identified by the system, with fields for user reference, prediction confidence, and dates.
* **Reports:** Stores both user- and system-generated reports for administrative review.
* **Comments:** Captures user feedback on their experiences with the platform.
* **Notifications:** Records notifications sent to users about important updates or trending diseases.

**Key Features**

* Relationships between entities ensure data consistency.
* Use of foreign keys to maintain referential integrity.
* Efficient indexing for optimal query performance.



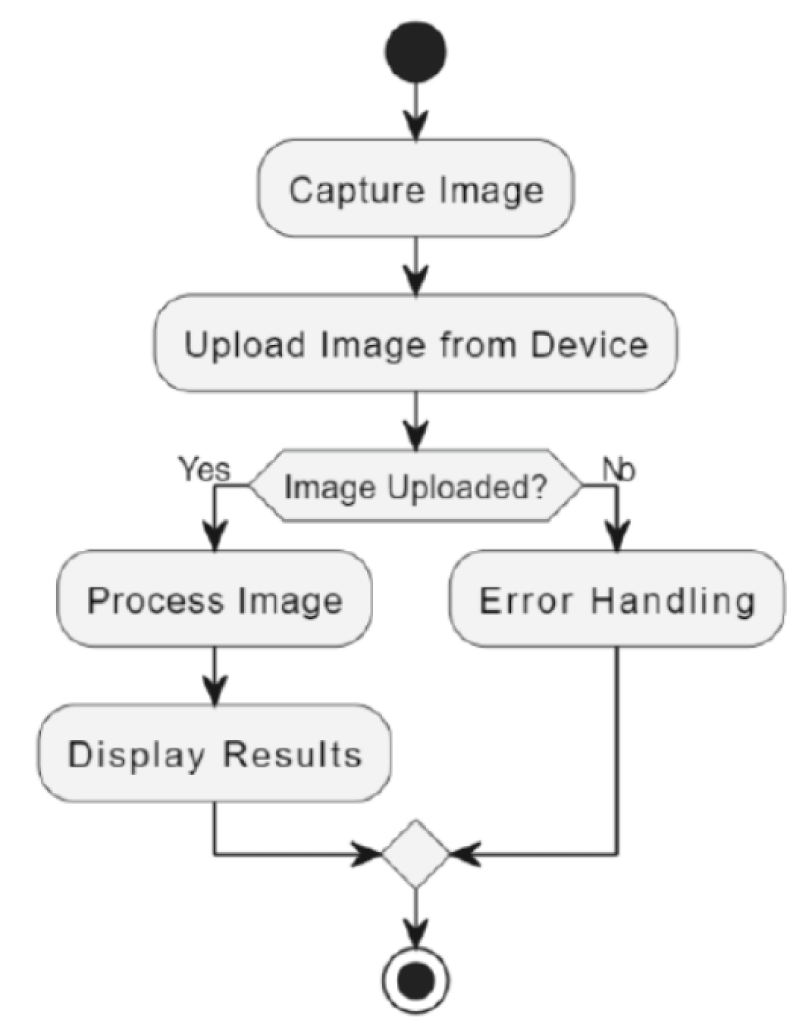
## **3.2 Flowchart for the Crop Disease Subsystem**

The flowchart illustrates the step-by-step workflow of the crop disease recognition subsystem. The process includes:

1. **Image Upload**: Users upload images of affected crops.
2. **Image Processing**: The system preprocesses the image for analysis.
3. **Model Prediction**: A pre-trained model analyzes the image and predicts the disease.
4. **Result Display**: The system presents the identified disease, confidence score, and relevant solutions.
5. **Data Logging:** Identified diseases are saved to the database for tracking and insights.

**Significance**

1. Ensures clarity in the disease identification process.
2. Highlights system interactions, including data processing and storage.
3. Identifies user touchpoints and system responses.



## **3.3 Use Case Diagram**

The use case diagram highlights the interactions between various user roles and the system functionalities. Key actors include:

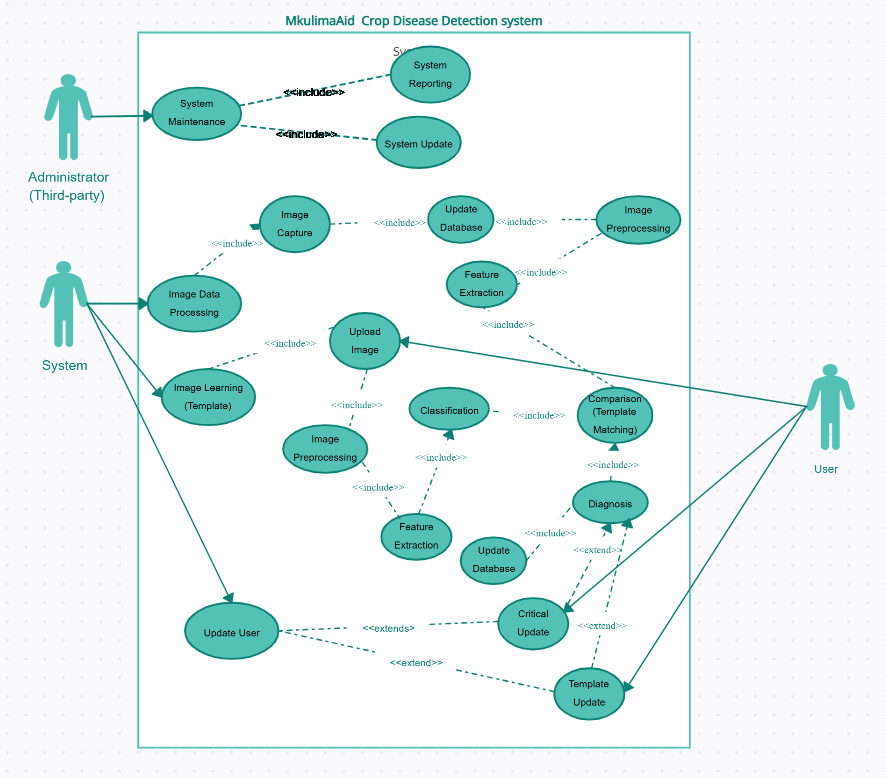
* **Farmers(users):** Use the platform for disease identification, notifications, and educational content.
* **Admins:** Manage content, review reports, send notifications, and oversee system activities.

**Use Cases**

1. **Identify Crop Diseases:** Farmers upload crop images to identify diseases and receive solutions.
2. **Manage Topics:** Admins add and update trending topics for user engagement.
3. **Generate Reports:** Users and the system generate reports for reference and tracking.
4. **Send Notifications:** Admins notify users about critical updates and trends.

**Significance**

1. Captures user-system interactions effectively.
2. Highlights core system functionalities.

These diagrams collectively demonstrate the robust architecture of the MkulimaAid system, detailing how the components work together to deliver a seamless experience. The EER diagram ensures proper database management, the flowchart outlines the subsystem's workflow, and the use case diagram defines user interactions with the system. Together, they provide a comprehensive understanding of MkulimaAid's design and functionality.

## **3.4 Frontend and Backend Tools**

The architecture of MkulimaAid incorporates a combination of frontend and backend tools to deliver an efficient, user-friendly, and scalable platform. This section provides an overview of the tools and technologies used.

#### **3.4.1 Frontend Tools**

The frontend is responsible for the user interface and experience. It ensures that users can seamlessly interact with the platform.

1. **HTML5**

* Structures the content and layout of web pages.
* Provides semantic elements for better accessibility and SEO.

1. **CSS3**

* Enhances the visual design of the platform.
* Utilizes modern features like flexbox and grid for responsive layouts.
* Includes green-themed styles for consistency with the crop-related theme.

1. **Bootstrap**

* Aids in creating a responsive and mobile-first design.
* Provides pre-styled components such as buttons, forms, and modals.

1. **JavaScript**

* Adds interactivity to the platform, such as form validation, dynamic content updates, and animations.

1. **AJAX**

* Enables asynchronous data fetching, providing a smooth user experience by reducing page reloads.

#### **3.4.2 Backend Tools**

The backend handles the logic, database interactions, and server-side functionalities of the platform.

1. **Flask (Python)**

* Lightweight web framework used to build the core application logic.
* Handles routing, user authentication, file uploads, and database interactions.

1. **SQLAlchemy**

* ORM (Object-Relational Mapping) tool for database operations.
* Simplifies query building and ensures database portability.

1. **MySQL**

* Relational database system for storing data.
* Provides robust and scalable data management for user records, disease data, notifications, and more.

1. **Pre-Trained Machine Learning Model**

* A trained model (e.g., based on TensorFlow or PyTorch) is integrated to analyze crop images and predict diseases.

1. **WTForms**

* Manages form creation, validation, and submission.
* Ensures secure and user-friendly interactions.

1. **Jinja2 Templating Engine**

* Renders dynamic content in HTML pages.
* Enables the seamless integration of backend data into the frontend.

## **3.5 Integration Tools**

1. **Flask-Mail**

* Manages email notifications for password resets and other system alerts.

1. **Flask-Session**

* Provides secure session management, including timeouts for inactive users.

1. **Flask-Login**

* Facilitates user authentication and session handling.

1. **APIs and Webhooks**

* Potentially used for integrating external services, such as weather data or third-party farming tools.

**Why These Tools Were Chosen**

1. **Efficiency:** Tools like Flask and SQLAlchemy are lightweight yet powerful, ideal for small- to medium-scale applications.
2. **Scalability:** The modular nature of Flask allows the system to scale as requirements grow.
3. **User Experience:** Modern frontend tools like Bootstrap and AJAX ensure a responsive, interactive platform.
4. **Security:** WTForms, Flask-Login, and Flask-Session provide robust mechanisms for secure user interactions.
5. **Ease of Development:** Python's simplicity and Flask's flexibility speed up development without sacrificing functionality.

The selection of these frontend and backend tools ensures that MkulimaAid is robust, user-friendly, and scalable. Together, they enable the platform to deliver core functionalities effectively, from crop disease detection to user interaction and data management.

## **3.6 Hosting Tools**

Heroku is the primary hosting platform for MkulimaAid. It simplifies deployment and ensures scalability and reliability.

1. **Platform-as-a-Service (PaaS)**

* Heroku offers an intuitive deployment workflow, integrating seamlessly with Git.
* Supports scaling resources to match traffic demands.

1. **Static File Hosting**

* Static assets such as images, CSS, and JavaScript are served efficiently using Heroku’s static file serving capabilities.

1. **Database Hosting**

* Heroku provides add-ons like Heroku ClearDB MySQL or Heroku Postgres for managing the relational database.

1. **Domain Management**

* Heroku’s custom domain feature allows easy integration of the project’s domain name (mkulimaaid.com).

1. **SSL/TLS Encryption**

* Heroku automatically manages SSL certificates to ensure secure HTTPS connections.

1. **Monitoring and Analytics**

* Built-in Heroku monitoring tools provide insights into application performance, scaling requirements, and potential errors.

**Why Heroku Was Chosen**

1. **Ease of Use:** Simplifies the deployment process, making it ideal for rapid development.
2. **Cost-Effective:** Free and affordable plans are suitable for the project's current stage.
3. **Scalability:** Can accommodate increasing user traffic with minimal effort.
4. **Security:** Offers built-in SSL/TLS support for secure communication.
5. **Integration:** Works seamlessly with Flask, Python, and the MkulimaAid tech stack.

# **Chapter 4: Project Timeline and Gantt Chart**

The MkulimaAid project timeline provides a structured plan for the development and deployment of the system. A Gantt chart has been used to visually represent the key phases, tasks, and milestones, along with their respective durations and dependencies.

The chart outlines the following:

## **Project Phases**

* Initiation
* Requirement Analysis
* System Design
* Development (Frontend and Backend)
* Testing
* Deployment
* Documentation
* Project Finalization

## **Task Dependencies**

Each task is interconnected, with subsequent tasks dependent on the completion of prior activities. This ensures a logical flow and timely completion of the project.

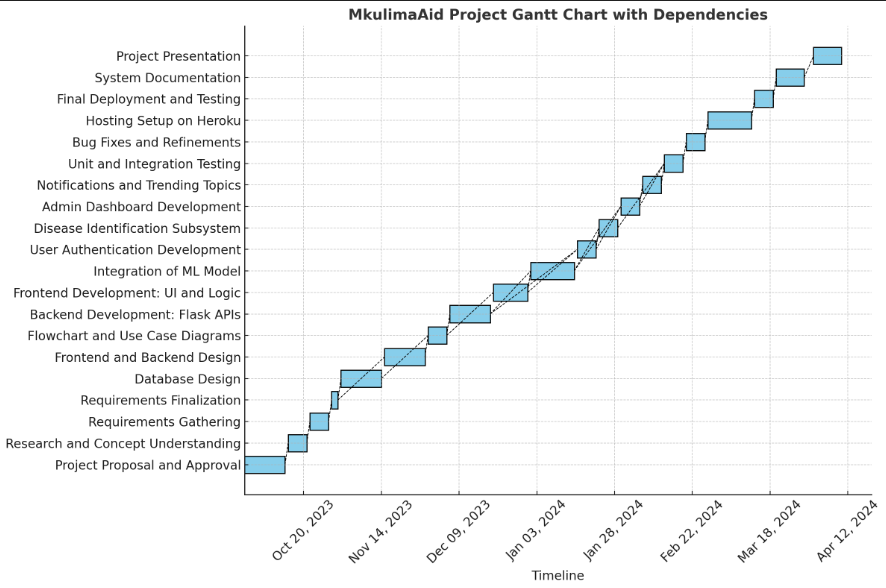
## **Timeline**

The project spans from October of the previous year to April of the current year, covering all necessary activities to deliver a fully functional and user-friendly MkulimaAid system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task ID | Task Name | Start Date | End Date | Duration | Dependencies |
| 1 | Project Proposal and Approval | 1 Oct (Year-1) | 14 Oct (Year-1) | 2 weeks | - |
| 2 | Research and Concept Understanding | 15 Oct (Year-1) | 21 Oct (Year-1) | 1 week | 1 |
| 3 | Requirements Gathering | 22 Oct (Year-1) | 28 Oct (Year-1) | 1 week | 2 |
| 4 | Requirements Finalization | 29 Oct (Year-1) | 31 Oct (Year-1) | 3 days | 3 |
| 5 | Database Design | 1 Nov (Year-1) | 14 Nov (Year-1) | 2 weeks | 4 |
| 6 | Frontend and Backend Design | 15 Nov (Year-1) | 28 Nov (Year-1) | 2 weeks | 4 |
| 7 | Flowchart and Use Case Diagrams | 29 Nov (Year-1) | 5 Dec (Year-1) | 1 week | 5, 6 |
| 8 | Backend Development: Flask APIs | 6 Dec (Year-1) | 19 Dec (Year-1) | 2 weeks | 7 |
| 9 | Frontend Development: UI and Logic | 20 Dec (Year-1) | 31 Dec (Year-1) | 2 weeks | 7 |
|  |  |  |  |  |  |
| 10 | Integration of ML Model | 1 Jan (Year) | 15 Jan (Year) | 2 weeks | 8, 9 |
| 11 | User Authentication Development | 16 Jan (Year) | 22 Jan (Year) | 1 week | 8, 9 |
| 12 | Disease Identification Subsystem | 23 Jan (Year) | 29 Jan (Year) | 1 week | 10, 11 |
| 13 | Admin Dashboard Development | 30 Jan (Year) | 5 Feb (Year) | 1 week | 10, 11 |
| 14 | Notifications and Trending Topics | 6 Feb (Year) | 12 Feb (Year) | 1 week | 13 |
| 15 | Unit and Integration Testing | 13 Feb (Year) | 19 Feb (Year) | 1 week | 12, 13, 14 |
| 16 | Bug Fixes and Refinements | 20 Feb (Year) | 26 Feb (Year) | 1 week | 15 |
| 17 | Hosting Setup on Heroku | 27 Feb (Year) | 12 Mar (Year) | 2 weeks | 16 |
| 18 | Final Deployment and Testing | 13 Mar (Year) | 19 Mar (Year) | 1 week | 17 |
| 19 | System Documentation | 20 Mar (Year) | 29 Mar (Year) | 2 weeks | 18 |
| 20 | Project Presentation | 1 Apr (Year) | 10 Apr (Year) | 2 weeks | 19 |

## **Milestones**

* M1 (14 Oct Year-1): Project Approval Complete
* M2 (31 Oct Year-1): Requirements Finalized
* M3 (28 Nov Year-1): System Design Complete
* M4 (31 Dec Year-1): Core System Components Developed
* M5 (12 Mar Year): Deployment Ready for Testing
* M6 (10 Apr Year): Final Project Submission



# **Chapter 5: Project Theme**

The theme of MkulimaAid is centered around empowering farmers through innovative technology to enhance crop health and agricultural productivity. The system integrates cutting-edge tools, user-friendly interfaces, and real-time insights to support farmers in diagnosing crop diseases and accessing valuable agricultural resources.

## **Key Aspects of the Project Theme**

1. **Sustainability**

MkulimaAid promotes sustainable farming practices by providing detailed insights into organic and chemical control methods, along with preventive measures tailored to specific crop diseases.

1. **Accessibility**

The platform is designed with simplicity and usability in mind, ensuring that farmers, regardless of their technical expertise, can easily access and navigate the system to meet their needs.

1. **Community Engagement**

MkulimaAid fosters a collaborative environment by including features like a Q&A forum, interactive topics, and notifications about trending diseases, allowing farmers to share experiences and learn from one another.

1. **Innovation**

By leveraging artificial intelligence for crop disease identification, MkulimaAid showcases how modern technology can address traditional challenges in agriculture.

1. **Scalability**

The platform is built to accommodate growth, with functionalities that can expand to include more crops, advanced analytics, and integration with other agricultural tools.

## **Visual Theme and Design**

The system’s visual design reflects its agricultural focus, incorporating:

* **Color Palette:** Green tones to symbolize growth, nature, and agriculture.
* **Typography:** Clear and modern fonts to ensure readability across devices.
* **Imagery:** Use of crop images, disease visualizations, and engaging graphics to make the platform visually appealing.

MkulimaAid is a blend of technology, community, and agriculture, aiming to transform the way farmers manage their crops and ensure food security for all.

# **Chapter 6: Budget and Resources**

This section details the financial and technical resources required to execute the MkulimaAid project successfully, including cost estimates, hardware specifications, software tools, and human resource allocation.

## **6.1 Budget**

The budget outlines the projected costs for software, hardware, human resources, and other essential aspects of the MkulimaAid project.

#### **6.1.1. Software Costs**

|  |  |  |
| --- | --- | --- |
| **Item** | **Details** | **Cost (KES)** |
| Development Tools | Python, Flask, TensorFlow/Keras (free, open-source) | 0 |
| Design Tools | Figma (Professional plan for team collaboration) | 20,880 KES |
| Version Control | GitHub (Pro plan for private repositories) | 6,960 KES |
| Testing Tools | Selenium, Pytest, and Postman (free tools) | 0 |
| Database Management | SQLite (development) and PostgreSQL/MySQL (production) | 14,500 KES |
| Domain and SSL Certificate | Custom domain and security certificates | 7,250 KES |
| Cloud Hosting Services | Heroku | 43,500 KES |

**Total Software Cost:** 92,090 KES/year

#### **6.1.2. Hardware Costs**

|  |  |  |
| --- | --- | --- |
| **Item** | **Details** | **Cost (KES)** |
| Development Machines | Laptops/Desktops (2–3 developers) | 435,000 KES |
| Testing Devices | Smartphones, tablets, and PCs for cross-platform testing | 217,500 KES |
| Server Hardware (if local) | Alternative to cloud hosting (8-core CPU, 16GB RAM) | 174,000 KES |

**Total Hardware Cost:** 652,500 to 826,500 KES

#### **6.1.3. Human Resource Costs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **Hours/Week** | **Hourly Rate (USD)** | **Duration** | **Cost (KES)** |
| Project Manager | 10 | $30 | 12 weeks | 522,000 KES |
| Backend Developer | 20 | $25 | 12 weeks | 870,000 KES |
| Frontend Developer | 20 | $20 | 12 weeks | 696,000 KES |
| QA Tester | 15 | $20 | 8 weeks | 348,000 KES |
| ML Engineer | 15 | $35 | 8 weeks | 609,000 KES |

**Total Human Resource Cost:** 3,045,000 KES

#### **6.1.4. Miscellaneous Costs**

|  |  |  |
| --- | --- | --- |
| **Item** | **Details** | **Cost (KES)** |
| Training and Workshops | Training team members and end-users | 72,500 KES |
| Marketing and Promotion | Outreach campaigns to farmers | 145,000 KES |
| Contingency Fund | Reserve for unexpected expenses | 217,500 KES |

**Total Miscellaneous Cost:** 435,000 KES

#### **Grand Total**

|  |  |
| --- | --- |
| **Category** | **Estimated Cost (KES)** |
| Software Costs | 92,090 KES/year |
| Hardware Costs | 652,500 to 826,500 KES |
| Human Resource Costs | 3,045,000 KES |
| Miscellaneous Costs | 435,000 KES |

**Overall Project Cost:** 4,224,590 to 4,399,590 KES

# **Chapter 7: Resources**

## **7.1. Technical Resources**

**Development Environment**

Laptops/Desktops with at least:

* **Processor:** Intel Core i5 or equivalent
* **RAM:** 8GB minimum (16GB recommended for machine learning tasks)
* **Storage:** 500GB SSD or higher

**Software requirements**

* Python 3.9+
* Flask framework
* TensorFlow/Keras for machine learning
* SQLite/MySQL/PostgreSQL for database management

**Deployment Platform**

* Cloud Services: Heroku (preferred). Minimum specs: 4 vCPUs, 8GB RAM, 100GB storage.
* Domain and SSL Certificates: For secure access.

**Testing Environment**

* Virtual machines or staging servers to simulate real-world conditions.
* Devices for cross-platform testing: desktops, tablets, and smartphones.

## **7.2. Human Resources**

**Team Composition**

* **Project Manager:** Oversees development and timeline adherence.
* **Backend Developer:** Implements server-side functionalities.
* **Frontend Developer:** Designs and builds user interfaces.
* **ML Engineer:** Optimizes the crop disease detection model.
* **QA Testers:** Ensure system functionality and reliability.

**Stakeholders**

* **End-users:** Farmers and agricultural experts for feedback.
* **Domain Experts:** Validate the ML model's accuracy.

# **Chapter 8: Risk Management**

Risk management is a critical part of ensuring the successful execution of the MkulimaAid project. This section identifies potential risks, outlines mitigation strategies, and emphasizes the importance of regular monitoring and updates throughout the project lifecycle.

## **8.1. Identified Risks**

|  |  |  |
| --- | --- | --- |
| **Risk Category** | **Description** | **Potential Impact** |
| Technical Risks | Bugs in crop disease detection algorithm or integration issues with frontend and backend. | Delays in deployment, reduced accuracy. |
| Resource Risks | Insufficient computational power or failure of hardware resources during critical phases. | Workflow interruptions, missed deadlines. |
| Security Risks | Data breaches or unauthorized access to sensitive user information. | User trust loss, legal repercussions. |
| Human Resource Risks | Unavailability of key team members due to unforeseen circumstances or lack of technical skills. | Slower progress, compromised quality. |
| Financial Risks | Budget overruns due to unforeseen expenses, such as additional hosting costs. | Project scope reduction, delays. |
| Operational Risks | Difficulty in gathering feedback from farmers or stakeholders during testing phases. | Limited validation, reduced system effectiveness. |
| Time Risks | Inadequate time for testing and quality assurance due to delayed milestones. | Deployment of an unreliable system. |

## **8.2. Mitigation Strategies**

|  |  |
| --- | --- |
| **Risk** | **Mitigation Strategy** |
| Technical Risks | Perform early integrations and unit testing to identify and resolve issues incrementally. |
| Resource Risks | Use cloud hosting for scalability; maintain backup hardware for development and testing. |
| Security Risks | Implement robust security measures, including encryption, regular audits, and secure coding practices. |
| Human Resource Risks | Cross-train team members; maintain a buffer time for potential delays caused by team changes. |
| Financial Risks | Allocate a contingency fund; prioritize essential features to ensure core functionality. |
| Operational Risks | Engage stakeholders and farmers early through workshops and regular feedback loops. |
| Time Risks | Set realistic deadlines |

## **8.3. Monitoring and Updates**

* **Regular Monitoring:** Schedule weekly reviews to assess progress, address emerging risks, and implement corrective actions.
* **Issue Tracking:** Use tools like Jira or Trello for tracking technical issues and ensuring timely resolution.
* **Stakeholder Involvement:** Maintain open communication with stakeholders, including farmers, domain experts, and project sponsors, to gather feedback regularly.
* **Performance Metrics:** Define and monitor KPIs (Key Performance Indicators) for system performance, user engagement, and timeline adherence.

## **8.4. Early Integration Testing**

Early and frequent integration testing is essential to detect potential problems before they escalate. The team will adopt an iterative development approach, performing the following:

* **Unit Testing:** Test individual components of the system to ensure they function as expected.
* **System Integration Testing:** Check the interaction between the backend, frontend, and crop disease detection module to identify issues early.
* **User Acceptance Testing (UAT):** Involve end-users in testing to validate the system's usability and functionality.

# **Chapter 9: Quality Assurance**

Quality assurance ensures that the MkulimaAid system meets the required functionality, performance, and reliability standards. A structured testing plan is critical for detecting and resolving issues at various stages of development to deliver a robust and user-friendly solution.

## **Testing Plan**

The testing plan for MkulimaAid comprises various levels of testing to ensure every component and integration of the system works as intended. The testing plan includes:

#### **9.1. Unit Testing**

**Objective:** Verify that individual components or modules of the system function correctly in isolation.

**Scope:**

* Testing the backend API endpoints for correct data handling.
* Verifying the accuracy of the crop disease detection model predictions.
* Ensuring form validations (e.g., login, registration, comment submissions).

**Tools Used:** Pytest, unittest.

#### **9.2. Integration Testing**

**Objective:** Ensure that various modules and components of the system work together seamlessly.

**Scope:**

* Testing the integration between the frontend and backend, such as form submissions and API data rendering.
* Verifying database operations, including CRUD functionality for users, comments, diseases, and reports.
* Testing the crop disease detection flow from image upload to prediction results.

**Tools Used:** Postman, Selenium.

#### **9.3. System Testing**

**Objective:** Test the entire MkulimaAid system as a whole to validate compliance with requirements.

**Scope:**

* Ensure the system supports multiple concurrent users.
* Verify the performance of the crop disease detection model under load.
* Test all features across different browsers and devices for compatibility.

**Tools Used:** BrowserStack, Apache JMeter.

#### **9.4. User Acceptance Testing (UAT)**

**Objective:** Validate the system's functionality and usability from an end-user perspective.

**Scope:**

* Engage farmers and agricultural experts to provide feedback on system usability.
* Test the user interface for intuitiveness and ease of navigation.
* Verify that the results of the crop disease detection align with domain expert feedback.

**Participants:** Farmers, domain experts, and stakeholders.

#### **9.5. Regression Testing**

**Objective:** Ensure that new updates or bug fixes do not introduce new issues.

**Scope:**

* Test core functionalities after integrating new features or making system changes.
* Validate the existing unit and integration tests.

**Tools Used:** Pytest, automated test scripts.

#### **9.6. Performance and Load Testing**

**Objective:** Evaluate how the system performs under various load conditions.

**Scope:**

* Test the system's response time and stability under high user traffic.
* Ensure scalability for future growth in user base and dataset size.

**Tools Used:** Apache JMeter, Locust.

#### **9.7. Security Testing**

**Objective:** Identify and address vulnerabilities in the system to ensure data protection.

**Scope:**

* Test for SQL injection, XSS (Cross-Site Scripting), and CSRF (Cross-Site Request Forgery) vulnerabilities.
* Validate encryption mechanisms for sensitive data like passwords and user information.
* Perform authentication and authorization checks for restricted pages and actions.

**Tools Used:** OWASP ZAP, Burp Suite.

#### **9.8. Testing Schedule**

|  |  |  |
| --- | --- | --- |
| **Testing Phase** | **Duration** | **Responsibility** |
| Unit Testing | Weeks 1–3 | Backend and Frontend Developers |
| Integration Testing | Weeks 3–4 | Development Team |
| System Testing | Weeks 5–6 | QA Testers |
| User Acceptance Testing | Week 7 | QA Team, Stakeholders |
| Regression Testing | Ongoing | Development and QA Team |
| Performance and Load Testing | Weeks 6–7 | QA Testers |
| Security Testing | Weeks 5–7 | Security Specialists |

## **Reporting and Bug Tracking**

* Bugs and issues discovered during testing phases will be documented using tools like Jira or Trello.
* Reports will include details about the bug, its severity, steps to reproduce, and the resolution process.

# **Chapter 10: Implementation Plan**

The implementation plan outlines how the MkulimaAid system will be rolled out and made accessible to users. It includes steps for deployment, user training, and the provision of a user manual to ensure seamless adoption by stakeholders.

## **10.1. Deployment Plan**

The deployment of MkulimaAid will follow these steps:

**a. Environment Setup**

Prepare the production environment on Heroku, including configuring the necessary tools and services such as:

* Database setup (PostgreSQL for production).
* Deployment of the trained crop disease detection model.
* Integration of domain name and SSL certificate for secure access.

**b. Deployment Stages**

* Staging Environment
* Deploy the system to a staging environment to simulate real-world usage.
* Perform final testing, including cross-browser compatibility, load testing, and system stability checks.
* Production Environment
* Once the system passes all tests, deploy it to the production environment.
* Ensure monitoring tools (e.g., performance analytics, error tracking) are in place.

**c. System Monitoring**

* Use tools like Heroku metrics, application logs, and monitoring services to:
* Track system performance.
* Identify and address issues in real time.

## **10.2. Training Users**

To ensure users can utilize the MkulimaAid platform effectively, training sessions will be conducted:

**a. Target Users**

* Farmers and agricultural experts.
* System administrators and stakeholders.

**b. Training Format**

* Workshops and Seminars
* Organize hands-on sessions in agricultural centers or through online platforms.
* Focus on teaching farmers how to upload crop images, view results, and interact with system features like the Q&A forum and reports.
* Step-by-Step Tutorials
* Provide visual guides and video tutorials for using the system.
* Tutorials will include how to log in, upload images, view disease predictions, and access reports.

**c. Administrator Training**

* Train system administrators on how to manage content, such as uploading trending diseases, managing team members, and handling notifications.

## **10.3. User Manual**

A comprehensive user manual will be created to assist users in navigating the system.

**a. Contents of the User Manual**

**Introduction**

* Overview of MkulimaAid, its purpose, and benefits.

**Getting Started**

* Instructions for accessing the platform.
* How to register, log in, and manage accounts.

**Using Key Features**

* Uploading crop images for disease detection.
* Viewing disease prediction results.
* Accessing trending topics and reports.
* Using the Q&A forum to ask or answer questions.

**Admin Features**

* How to add, edit, and manage content (e.g., diseases, team members, notifications).
* Monitoring user activity and system performance.

**Troubleshooting**

* FAQs and common issues.
* Contact details for technical support.

**Glossary**

* Definitions of key terms used in the system (e.g., "crop disease detection," "Q&A forum").

**b. Manual Formats**

* PDF Version: Available for download on the platform.
* Printed Version: Distributed during training sessions and outreach events.

# **Chapter 11: Maintenance and Support**

The maintenance and support plan ensures the MkulimaAid system operates efficiently and remains up-to-date. This section details the maintenance schedule, system monitoring, and user support strategies.

## **11.1. Maintenance Schedule**

A regular maintenance schedule will be implemented to ensure the system's reliability, security, and performance.

|  |  |  |
| --- | --- | --- |
| **Maintenance Task** | **Frequency** | **Description** |
| System Backup | Weekly | Back up the database and system files to prevent data loss. |
| Security Updates | Monthly or as needed | Apply updates to secure against vulnerabilities (e.g., SSL certificate renewal, dependency upgrades). |
| Performance Monitoring | Weekly | Monitor system performance metrics and address any identified bottlenecks. |
| Bug Fixes | As required | Resolve bugs or issues reported by users or identified during monitoring. |
| Database Optimization | Monthly | Clean and optimize the database to improve query performance and system speed. |
| Feature Updates | Quarterly | Add new features or improve existing ones based on user feedback or system requirements. |
| Hardware Checks | Quarterly | Inspect and upgrade hardware (if local servers are used) to maintain operational efficiency. |

## **11.2. System Monitoring**

Monitoring tools will be used to identify and address issues before they affect users:

**a. Monitoring Tools**

* **Application Monitoring:** Use tools like New Relic or Sentry to track errors, crashes, and performance issues.
* **Server Monitoring:** Utilize Heroku's built-in monitoring features to analyze server metrics such as CPU usage, memory, and disk space.

**b. Alerts and Notifications**

* Set up automated alerts for critical issues (e.g., server downtime, database errors).
* Notify the technical team immediately to ensure quick resolution.

## **11.3. Support Strategy**

A robust support system will be established to assist users in resolving issues and provide guidance.

**a. Support Channels**

**Email Support**

* Provide an official email address for users to submit queries and report issues.
* Ensure responses within 24–48 hours.

**Help Desk System**

* Implement a ticketing system where users can log their issues and track their resolution status.

**Knowledge Base**

* Create a self-service portal with FAQs, troubleshooting guides, and video tutorials.

**b. Support Team**

* **Support Staff:** Dedicated team members will handle user inquiries and technical issues.
* **Escalation Process:** Establish a hierarchy for resolving complex issues, escalating them to senior developers or system administrators.

**c. Feedback Mechanism**

* Provide a feature for users to submit feedback directly through the platform.
* Regularly review feedback to identify common issues and areas for improvement.

## **11.4. Maintenance Policy**

The maintenance policy will outline procedures for handling system downtimes and updates:

* **Scheduled Downtime:** Notify users at least 48 hours in advance about planned maintenance.
* **Emergency Maintenance:** Address critical issues immediately and inform users of progress.
* **Service Level Agreement (SLA**): Guarantee a minimum uptime of 99%, ensuring high availability.

# **Appendices**

**Appendix 1: PlantVillage Dataset Overview**

The PlantVillage dataset is a comprehensive collection of over 50,000 expertly curated images depicting healthy and diseased leaves of various crop plants. This open-access repository was developed to facilitate the creation of mobile disease diagnostics through machine learning and crowdsourcing efforts.

Key Features:

* **Diversity:** The dataset encompasses a wide range of crops and associated diseases, providing a robust foundation for training machine learning models in plant disease identification.
* **Quality:** Each image in the dataset has been carefully curated to ensure clarity and relevance, enhancing the effectiveness of AI-driven diagnostic tools.
* **Accessibility:** As an open-access resource, the PlantVillage dataset is available to researchers and developers worldwide, promoting innovation in agricultural technology.

The MkulimaAid platform leverages this dataset to train its AI-powered disease detection system, enabling accurate and rapid identification of crop diseases from user-uploaded images. By utilizing the PlantVillage dataset, MkulimaAid ensures that farmers receive reliable diagnostics, empowering them to take timely and effective action in managing crop health.

**Appendix 2: Sample Farmer Survey Questionnaire on Crop Disease Management**

To better understand the needs and challenges faced by farmers regarding crop disease management, the following survey questionnaire can be utilized:

**General Information**

1. Name of Farmer:
2. Location (Village/Town, County):
3. Contact Information (Phone/Email):

**Farm Details**

1. Total land area under cultivation (in acres or hectares):
2. Types of crops grown:
3. Average yield per crop (per season):

**Crop Disease Management**

1. Have you experienced crop diseases on your farm in the past year? (Yes/No)

* If yes, please specify the diseases and affected crops:

1. How do you currently identify crop diseases?

* Self-inspection
* Consultation with local experts
* Agricultural extension services
* Other (please specify):

1. What methods do you use to manage or treat crop diseases?

* Chemical pesticides
* Organic treatments
* Traditional practices
* Other (please specify):

1. Have you received any training on crop disease management? (Yes/No)

* If yes, please specify the type of training and provider:

**Access to Information and Technology**

1. Do you own a smartphone or have access to one? (Yes/No)
2. How often do you use the internet?

* Daily
* Weekly
* Rarely
* Never

1. Are you a member of any agricultural forums or groups online? (Yes/No)

* If yes, please specify:

**Feedback on MkulimaAid Platform**

1. Would you be interested in using a web-based platform for crop disease identification and management? (Yes/No)
2. What features would you find most beneficial?

* AI-powered disease detection
* Access to educational resources
* Community forums for farmer interaction
* Notifications about disease outbreaks
* Other (please specify):

1. What challenges do you anticipate in using such a platform?

* Limited internet access
* Language barriers
* Digital literacy
* Other (please specify):

This questionnaire aims to gather valuable insights to tailor the MkulimaAid platform to the specific needs and circumstances of farmers. It draws inspiration from existing agricultural surveys, such as the FDA Farm Investigation Questionnaire.