

**Egerton University**

**Computer Science Department**

**System Design Documentation for MkulimaAid**

**(Crop Disease Detection and Support System)**

**Prepared by: Fabius Lihanda Achevi**

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

**Date: February 2024**

**Table of Contents**

[**Chapter 1: System Overview** 5](#_Toc190096965)

[**1.1** **Project Name** 5](#_Toc190096966)

[**1.2** **Objectives** 5](#_Toc190096967)

[**1.3 Stakeholders** 5](#_Toc190096968)

[**1.4 Scope** 6](#_Toc190096969)

[**In-Scope Features:** 6](#_Toc190096970)

[**Out of Scope:** 6](#_Toc190096971)

[**Chapter 2: System Architecture Design** 7](#_Toc190096972)

[**2.1 System Architecture Overview** 7](#_Toc190096973)

[**Technology Stack:** 7](#_Toc190096974)

[**2.2 System Components** 8](#_Toc190096975)

[**2.3 System Architecture Diagram** 9](#_Toc190096976)

[**2.4 Sequence Diagram: Crop Disease Detection System** 10](#_Toc190096977)

[**Chapter 3: Data Flow Diagrams (DFDs)** 12](#_Toc190096978)

[**3.1 Levels of Data Flow Diagrams** 12](#_Toc190096979)

[**3.2 Data Flow Diagram – Level 0 (Context Diagram)** 12](#_Toc190096980)

[**3.3 Data Flow Diagram – Level 1** 13](#_Toc190096981)

[**3.4 Data Flow Diagram – Level 2** 14](#_Toc190096982)

[**3.5 Generalized Data Flow Diagram** 15](#_Toc190096983)

[**3.6 Importance of Data Flow Diagrams in System Design** 16](#_Toc190096984)

[**Chapter 4: User Interface Design** 16](#_Toc190096985)

[**4.1 Role-Based Access Control** 17](#_Toc190096986)

[**4.2 Navigation and Quick Access** 17](#_Toc190096987)

[**4.3 User Experience (UX) Considerations** 18](#_Toc190096988)

[**4.4 Report Generation** 18](#_Toc190096989)

[**4.5 Notifications System** 19](#_Toc190096990)

[**4.6 UI Wireframes & Mockups** 19](#_Toc190096991)

[**1.** **Questions and Answers forum** 20](#_Toc190096992)

[**2. Farmers dashboard** 21](#_Toc190096993)

[**3. Admin Dashboard** 22](#_Toc190096994)

[**Chapter 5: Database Design** 23](#_Toc190096995)

[**5.1 Entity-Relationship Diagram (EER)** 23](#_Toc190096996)

[**Chapter 6: Functional Module Design** 24](#_Toc190096997)

[**6.1 Key Functional Modules** 24](#_Toc190096998)

[**6.2 Functional Requirements** 25](#_Toc190096999)

[**Chapter 7: Security Design** 26](#_Toc190097000)

[**7.1 Authentication & Authorization** 26](#_Toc190097001)

[**7.2 Data Validation & Input Sanitization** 26](#_Toc190097002)

[**7.3 Data Integrity & Adherence to ACID Properties** 27](#_Toc190097003)

[**7.4 Secure Data Storage & Encryption** 27](#_Toc190097004)

[**7.5 Preventing Unauthorized Access & Misuse** 27](#_Toc190097005)

[**Chapter 8: Hardware and Software Requirements** 28](#_Toc190097006)

[**8.1 Hardware Specifications** 28](#_Toc190097007)

[**8.2 Software Specifications** 28](#_Toc190097008)

[**8.3 Security Software & Measures** 29](#_Toc190097009)

[**8.4 Backup & Recovery Tools** 29](#_Toc190097010)

[**8.5 Network Requirements** 29](#_Toc190097011)

[**8.6 Technologies Used** 29](#_Toc190097012)

[**Chapter 9: System Integration and Testing Plan** 30](#_Toc190097013)

[**9.1 System Integration Approach** 30](#_Toc190097014)

[**9.2 System Testing Phases** 31](#_Toc190097015)

[**Performance Testing** 31](#_Toc190097016)

[**User Acceptance Testing (UAT)** 31](#_Toc190097017)

[**Test Schedule & Timeline** 31](#_Toc190097018)

[**Environment Testing** 32](#_Toc190097019)

[**Testing Plan Summary** 32](#_Toc190097020)

[**Chapter 10: Deployment and Maintenance Plan** 33](#_Toc190097021)

[**10.1 Deployment Plan** 33](#_Toc190097022)

[**Deployment Strategy** 33](#_Toc190097023)

[**Hosting & Server Configuration** 33](#_Toc190097024)

[**Deployment Steps** 33](#_Toc190097025)

[**10.2 Maintenance Plan** 34](#_Toc190097026)

[**Regular Updates & Bug Fixes** 34](#_Toc190097027)

[**Backup & Disaster Recovery** 34](#_Toc190097028)

[**System Monitoring** 34](#_Toc190097029)

[**10.3 Scalability & Future Enhancements** 34](#_Toc190097030)

[**Scaling Strategy** 34](#_Toc190097031)

[**Future Enhancements** 34](#_Toc190097032)

# **Chapter 1: System Overview**

## **Project Name**

**MkulimaAid:** **A Crop Disease Detection and Farmer Support System**

## **Objectives**

MkulimaAid is designed to assist farmers in identifying crop diseases through image classification, providing accurate disease information, and offering solutions for effective management. The system aims to bridge the gap between farmers and agricultural experts by providing a centralized platform for crop disease diagnosis, knowledge sharing, and community support.

The key objectives of MkulimaAid include:

1. **Crop Disease Identification** – Utilize machine learning to classify and diagnose crop diseases from uploaded images.
2. **Farmer Support and Awareness** – Provide reliable information on disease symptoms, causes, preventive measures, and treatments.
3. **Interactive Knowledge Sharing** – Enable farmers to engage in discussions, ask questions, and receive expert guidance through a Q&A forum.
4. **Trending Disease Alerts** – Notify farmers about prevalent diseases in their regions to help in early detection and prevention.
5. **User-Friendly Dashboard** – Offer an intuitive interface for farmers and administrators to manage disease data, reports, and user interactions.
6. **Secure User Authentication** – Ensure a protected environment with account-based access control for different user roles.
7. **Data Analytics and Reports** – Provide insights on frequently identified diseases, user engagement, and platform usage trends.

## **1.3 Stakeholders**

MkulimaAid is designed to serve a variety of stakeholders, each playing a crucial role in ensuring the system's success:

1. **Farmers** – The primary users who utilize the platform for disease detection, discussions, and learning about disease management.
2. **Agricultural Experts** – Specialists who provide expert advice, answer queries, and contribute to disease awareness content.
3. **Administrators** – Responsible for managing the system, verifying content, moderating discussions, and ensuring smooth operations.
4. **Researchers and Agronomists** – Utilize data collected from the platform for agricultural research and improvements in disease control methodologies.
5. **Government and Agricultural Organizations** – May use the platform to disseminate important agricultural policies, best practices, and support programs.
6. **Developers and IT Support Team** – Maintain the system, perform updates, and ensure optimal performance and security.
7. **Agribusinesses and Suppliers** – May leverage insights from disease trends to provide appropriate farming solutions, such as fertilizers and pesticides.

## **1.4 Scope**

The scope of MkulimaAid is derived from its objectives, defining the functionalities and constraints of the system:

### **In-Scope Features:**

1. **User Authentication and Role-Based Access** – Secure login system with differentiated access for farmers, admins, and experts.
2. **Crop Disease Detection System** – Image classification model to predict crop diseases and suggest treatment options.
3. **Disease Information Database** – A repository of detailed information about common crop diseases, symptoms, and solutions.
4. **Community Q&A Forum** – A platform where users can ask and answer questions related to crop diseases and farming techniques.
5. **Admin Dashboard** – An interface for administrators to manage disease records, user interactions, and system content.
6. **Notifications System** – Alerts and messages for users regarding trending diseases, system updates, and announcements.
7. **Reports and Analytics** – Insights on frequently identified diseases, user engagement, and system usage statistics.
8. **Help Center and Knowledge Base** – A self-service section containing FAQs, troubleshooting guides, and educational materials.
9. **Comment and Feedback System** – Users can provide feedback on the system and share their experiences.

### **Out of Scope:**

1. **Offline Functionality** – The system requires an internet connection and does not support offline operations.
2. **Hardware-Specific Solutions** – MkulimaAid does not include IoT-based hardware for real-time farm monitoring.
3. **Automated Disease Treatment** – The system provides guidance but does not physically treat or control crop diseases. It only offers incentives such as the preventive and control measures.
4. **Integration with Government Subsidy Programs** – The platform does not handle financial transactions or government subsidies for farmers.
5. **AI-Powered Chatbot Support** – While the system provides static information and a Q&A forum, it does not include an AI-driven chatbot for automated responses.

By defining these boundaries, MkulimaAid ensures a focused, scalable, and effective solution for agricultural disease management and farmer support.

# **Chapter 2: System Architecture Design**

## **2.1 System Architecture Overview**

MkulimaAid is built on a client-server architecture using a three-tier structure consisting of the presentation layer (frontend), application layer (backend), and data layer (database). This design ensures modularity, scalability, and maintainability of the system.

The architecture follows the Model-View-Controller (MVC) design pattern, separating data management, user interface, and business logic.

* **Frontend (Presentation Layer):** Provides the user interface (UI) for farmers, experts, and administrators to interact with the system.
* **Backend (Application Layer):** Handles business logic, processes user requests, and communicates between the frontend and the database.
* **Database (Data Layer):** Stores user data, disease records, forum discussions, notifications, and system logs.

### **Technology Stack:**

1. **Frontend:** HTML, CSS, JavaScript (Bootstrap for styling), and AJAX for dynamic interactions.
2. **Backend:** Flask (Python) for handling requests, API endpoints, and processing image classification.
3. **Database:** MySQL for structured storage of user accounts, disease data, reports, and forum discussions.
4. **Machine Learning Model:** Pre-trained image classification model for identifying crop diseases.
5. **Cloud Storage:** Used for storing images and backups.
6. **Hosting & Deployment:** Hosted on Heroku or a cloud-based service.

## **2.2 System Components**

The MkulimaAid system consists of several core components that work together to deliver its functionalities.

**1. User Management System**

* Provides user authentication and role-based access control (Farmers, Experts, Admins).
* Users can manage their profiles, update account settings, and reset passwords.
* Security measures include password hashing, session timeouts, and CSRF protection.

**2. Crop Disease Identification Module**

* Uses a pre-trained machine learning model to analyze uploaded crop images and classify diseases.
* Provides disease details, symptoms, causes, prevention methods, and treatment recommendations.
* Stores user-uploaded images and diagnosis results in the database.

**3. Disease Information Database**

* A structured repository of crop diseases, symptoms, causes, and treatment options.
* Allows administrators to add, edit, and remove disease records from the admin panel.

**4. Q&A Forum (Community Discussion)**

* Enables farmers to ask and answer questions related to crop diseases and farming best practices.
* Displays user avatars, timestamps, and expert responses.
* Admins and experts can moderate discussions and highlight verified answers.

**5. Trending Disease Alerts & Notifications System**

* Sends alerts about newly identified crop diseases affecting specific regions.
* Supports push notifications, email alerts, and dashboard updates.
* Admins can manage and broadcast critical announcements.

**6. Reports & Analytics System**

* Provides data visualization of identified crop diseases and user activity.
* Displays reports on top identified diseases, weekly/monthly user growth, and forum engagement trends.
* Admins can export reports for further analysis.

**7. Help Center & Knowledge Base**

* Provides FAQs, troubleshooting guides, and video tutorials for farmers.
* Includes search functionality to help users find relevant support topics.

**8. Feedback & Support System**

* Users can submit feedback and feature requests through the platform.
* Implements a ticketing system where users report issues and track resolutions.

**9. Admin Dashboard**

* Provides full system control to administrators, including:
* Managing users and permissions.
* Reviewing and approving disease reports.
* Monitoring system logs and analytics.
* Managing forum discussions and expert responses.

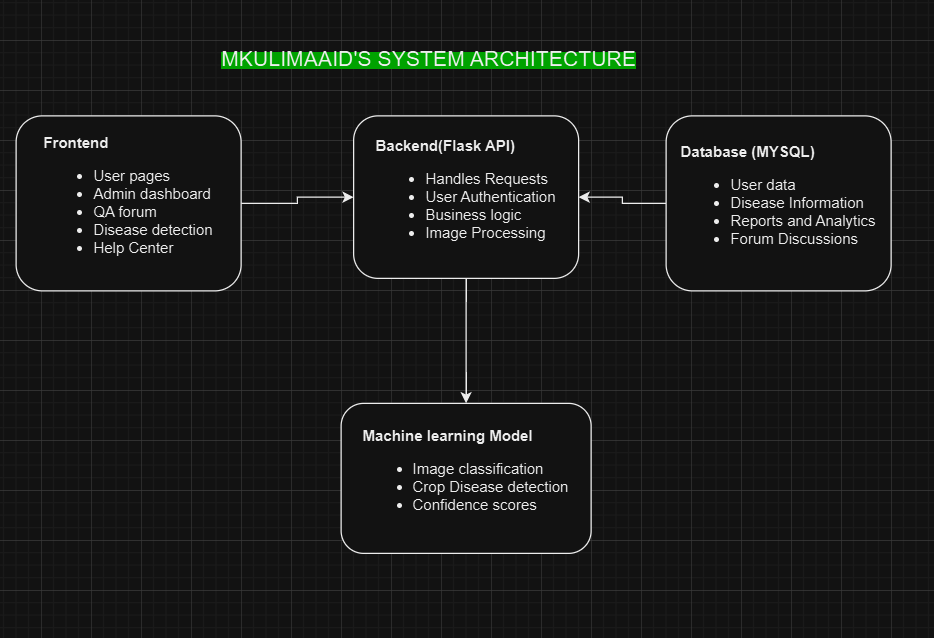
**10. Security & Compliance Measures**

* Implements HTTPS (SSL) encryption for secure data transmission.
* Uses role-based access control (RBAC) to protect sensitive data.
* Regular database backups to prevent data loss.
* Implement Google’s Recaptcha to prevent bot attacks.

## **2.3 System Architecture Diagram**

MkulimaAid follows a modular architecture consisting of four major components:

* **Frontend** – The user interface that allows farmers and administrators to interact with the system.
* **Backend** – The core logic of the system, managing authentication, requests, and interactions between the frontend, database, and ML model.
* **Database** – Stores user data, identified diseases, forum discussions, reports, and system logs.
* **Machine Learning Model** – The CNN-based image classification model used for identifying crop diseases from uploaded images.



**Workflow**

1. A user (farmer/admin) interacts with the system via the frontend, which is built using HTML, CSS, and JavaScript.
2. The backend (Flask server) handles requests, processes user inputs, and communicates with both the database and the ML model.
3. When a farmer uploads an image for disease detection, the backend forwards the image to the ML model.
4. The ML model (CNN-based classifier) processes the image and predicts the crop disease. The response is then sent back to the backend.
5. The backend stores the results in the database and returns the disease information to the user.
6. The user can also access disease records, ask questions in the forum, receive notifications, and interact with the system's knowledge base.

## **2.4 Sequence Diagram: Crop Disease Detection System**

The Crop Disease Detection System is a core module of MkulimaAid, enabling farmers to detect diseases through image uploads. It consists of three main actors:

1. **User (Farmer/Admin)**

* Uploads an image for disease detection.
* Views the predicted disease result.

1. **Python Server (Backend Processing)**

* Handles the image upload request.
* Preprocesses the image before sending it to the ML model.
* Retrieves and displays the predicted results.

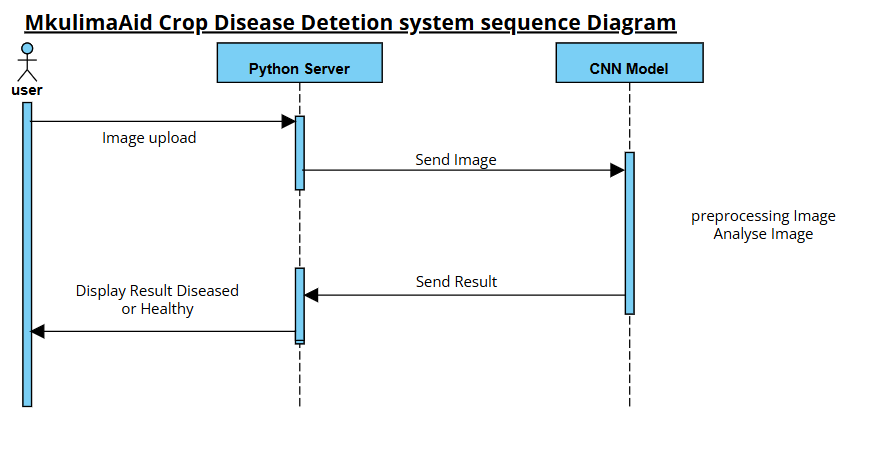
1. **CNN Model (Deep Learning Model)**

* Processes the uploaded image.
* Classifies the disease using trained deep learning techniques.
* Returns the disease prediction with confidence scores.

**Workflow**

1. The user uploads an image via the system.
2. The backend processes the image and sends it to the CNN model.
3. The CNN model analyzes the image and predicts the crop disease.
4. The backend receives the prediction and stores relevant data in the database.
5. The user receives the result.

This process ensures quick and accurate disease detection, allowing farmers to take timely action in managing crop health.



# **Chapter 3: Data Flow Diagrams (DFDs)**

**Data Flow Diagrams (DFDs)** are essential tools in system design that visualize how data moves through a system, showcasing the flow between input, processing, and output stages. DFDs help developers and stakeholders understand the processes within the MkulimaAid system, including data input, processing, storage, and output.

In this chapter, we present different levels of DFDs that illustrate how the MkulimaAid system processes images of plant leaves to detect diseases and provide insights to users.

## **3.1 Levels of Data Flow Diagrams**

DFDs are designed using a hierarchical approach, where each level provides a more detailed view of the system's functionality.

1. **Level 0 (Context Diagram):** Represents the high-level overview of the system, showing how data flows between users and the system.
2. **Level 1:** Provides more details on major processes, including image processing, model predictions, and user interactions.
3. **Level 2:** Expands on Level 1, detailing sub-processes and the data interactions within individual components.

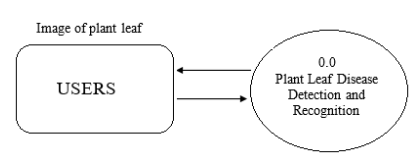
Each level is described in detail below.

### **3.2 Data Flow Diagram – Level 0 (Context Diagram)**

The Level 0 DFD, also known as the context diagram, represents the entire system as a single process. It highlights how data enters the system and the final output without internal complexities.

Components:

* **User (Farmer):** Inputs an image of an infected plant leaf.
* **MkulimaAid System:** Processes the image and identifies the disease.
* **Output:** The system displays the detected plant disease



**Description**

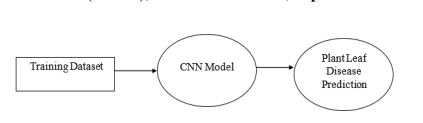
1. The user uploads an image of the affected plant leaf.
2. The system receives the image and processes it using a pre-trained deep learning model.
3. The system returns the disease classification result to the user.

### **3.3 Data Flow Diagram – Level 1**

The Level 1 DFD expands upon Level 0 by breaking down the system into multiple sub-processes, showing how data moves between different components.

Components:

* **User (Farmer):** Uploads the leaf image.
* **Preprocessing Unit:** Enhances and standardizes the image before classification.
* **CNN Model:** A trained Convolutional Neural Network (CNN) analyzes the image and predicts the plant disease.
* **Output System:** Displays results to the user.



**Description**

1. The user uploads an image to the system.
2. The preprocessing unit ensures the image is properly formatted for analysis.
3. The CNN model processes the image, extracts features, and predicts the disease.
4. The output system presents the results to the user.

### **3.4 Data Flow Diagram – Level 2**

The Level 2 DFD further expands on Level 1, detailing specific processes within each component of the system.

Components:

1. **Image Upload Module**: Handles file input from the user.

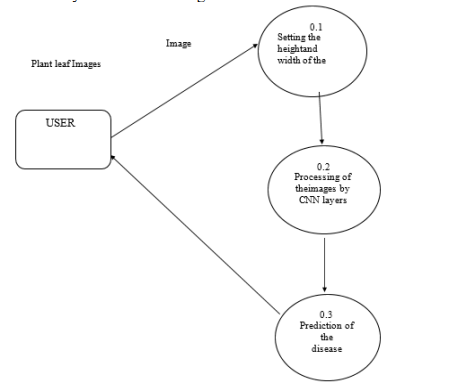
* Image Preprocessing Unit:
* Resizes and normalizes the image.
* Converts the image into a format suitable for deep learning models.

1. **Feature Extraction Unit (CNN):**

* Extracts key patterns from the image.
* Classifies the disease based on learned features.

1. **Result Display Module:**

* Displays prediction results.

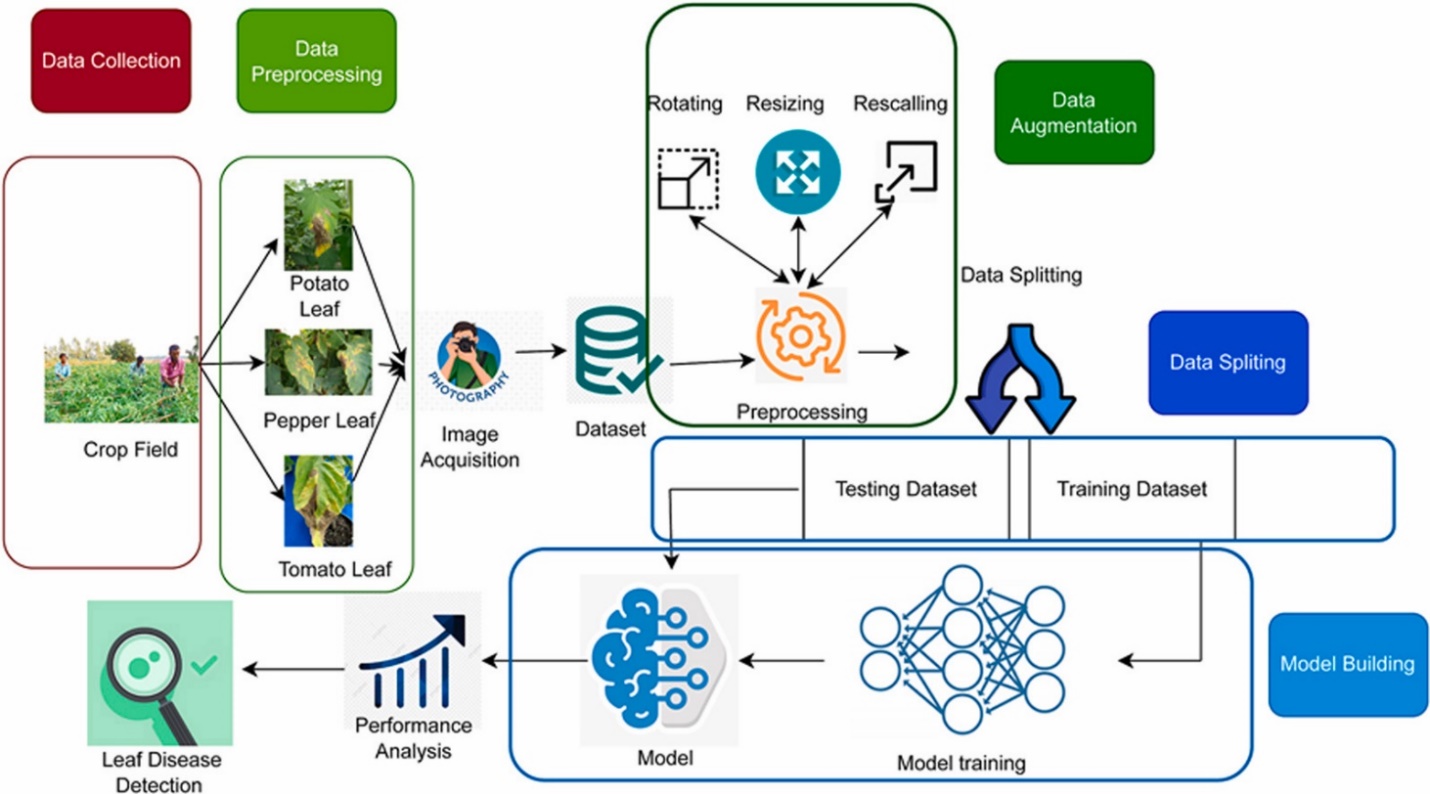


**Description**

1. The uploaded image is first processed by the image preprocessing unit, where necessary transformations are applied.
2. The feature extraction unit (CNN model) analyzes the image and classifies it into a specific disease category.
3. The result display module presents findings to the user.

### **3.5 Generalized Data Flow Diagram**

Beyond the hierarchical breakdown, a generalized DFD represents the system holistically, combining the essential components of Levels 0, 1, and 2. This diagram serves as a summarized visual representation of how data flows across all major system components.



**Why It Matters**

1. Helps in understanding end-to-end system workflow.
2. Highlights interactions between subsystems.
3. Supports troubleshooting and optimization of processes.

### **3.6 Importance of Data Flow Diagrams in System Design**

Data Flow Diagrams (DFDs) play a crucial role in system design and documentation for the MkulimaAid platform.

Benefits of Using DFDs:

1. **Clear Visualization:** DFDs provide a structured way to visualize how data moves through the system.
2. **System Understanding:** Helps developers and stakeholders comprehend how different modules interact.
3. **Process Optimization:** Identifies bottlenecks in the workflow, allowing for system improvements.
4. **Error Detection:** Helps in troubleshooting potential design flaws by mapping out each process.
5. **Documentation & Maintenance:** Serves as an essential reference for future system modifications and updates.

# **Chapter 4: User Interface Design**

The User Interface (UI) Design of MkulimaAid aims to create an intuitive, user-friendly experience for farmers, agricultural experts, and administrators. The design focuses on accessibility, ease of navigation, and seamless interaction with the system's features.

**The UI ensures**

* A responsive and visually appealing layout with a green theme, aligning with the agricultural context.
* Role-based access to restrict functionalities based on user types (farmers, experts, admins).
* Quick access to important features such as disease detection, reports, and community forums.
* Interactive and engaging elements to enhance the overall user experience.
* Notifications and alerts to keep users informed about trending diseases and system updates.

## **4.1 Role-Based Access Control**

MkulimaAid incorporates role-based access control (RBAC) to manage system functionalities efficiently. The key roles include:

**1. Farmers**

* Upload images for disease detection.
* View disease diagnosis results and suggested treatments.
* Participate in the Q&A forum to ask and answer farming-related questions.
* Receive notifications on trending crop diseases.
* Access educational content in the Help Center.
* Manage their user profile (update details, change passwords, upload avatars).
* Submit feedback on system performance.

**2. Agricultural Experts**

* Review and verify disease identification results.
* Answer farmer queries in the Q&A forum.
* Contribute to knowledge base articles and disease information.
* Receive alerts on trending diseases and contribute expert insights.

**3. Administrators**

* Manage users (approve, deactivate, or assign roles).
* Oversee the forum (moderate discussions, remove inappropriate content).
* Add, edit, and delete disease information in the system.
* Publish notifications and announcements for farmers.
* Generate reports and analytics for system monitoring.
* Manage team members' profiles on the team page.

## **4.2 Navigation and Quick Access**

To enhance usability, the system provides quick access options, ensuring seamless navigation.

1. **Navigation Menu**

* **Farmers’ Dashboard:** Home, Upload Image, Q&A Forum, Disease Info, Reports, Notifications, Help Center, Profile.
* **Experts’ Dashboard:** Home, Review Cases, Q&A Forum, Notifications, Knowledge Base, Profile.
* **Admin Dashboard:** Home, User Management, Disease Management, Reports & Analytics, Forum Moderation, Team Management, Notifications, Settings.

2. **Quick Actions & Shortcuts**

* Upload Image button for fast disease identification.
* Search bar for quick lookup of diseases and forum topics.
* Profile dropdown menu with account settings and logout option.
* Recent Activity Panel showing recent forum discussions and system alerts.
* Notification Bell for real-time alerts on trending diseases.

## **4.3 User Experience (UX) Considerations**

MkulimaAid's UI is designed with the following UX principles in mind:

**1. Simplicity & Intuitive Design**

* A clean and uncluttered interface for better readability.
* Minimal clicks required to access key features.
* Logical workflow and structured menus to guide users efficiently.

**2. Mobile Responsiveness**

* The system is fully responsive, allowing farmers to access it from mobile devices.
* Optimized touch-friendly elements for easy interaction on smartphones.

**3. Accessibility**

* Color contrast and large fonts for readability.
* Alternative text for screen reader support.
* Language support for different local dialects in the future.

**4. Feedback Mechanisms**

* Loading indicators to show progress during image uploads.
* Error messages and tooltips to guide users when submitting forms.
* Success notifications after actions like uploading an image or posting a question.

## **4.4 Report Generation**

MkulimaAid includes an interactive report generation feature, allowing admins to generate data-driven insights.

**1. Report Types**

* **Identified Diseases Report:** Summary of crop diseases detected by the system, including frequency and affected regions.
* **User Activity Report:** Tracks engagement levels in the forum and disease identification module.
* **Trending Disease Report:** Highlights the most frequently occurring crop diseases within a specified period.
* **System Usage Report**: Overview of registered users, active sessions, and interactions.

**2. Report Features**

* Filter options (timeframe selection: weekly, monthly, yearly).
* Export capability (PDF, Excel, CSV).
* Graphical representation using charts and graphs.

## **4.5 Notifications System**

MkulimaAid features a comprehensive notification system to keep users informed in real-time.

**1. Notification Types**

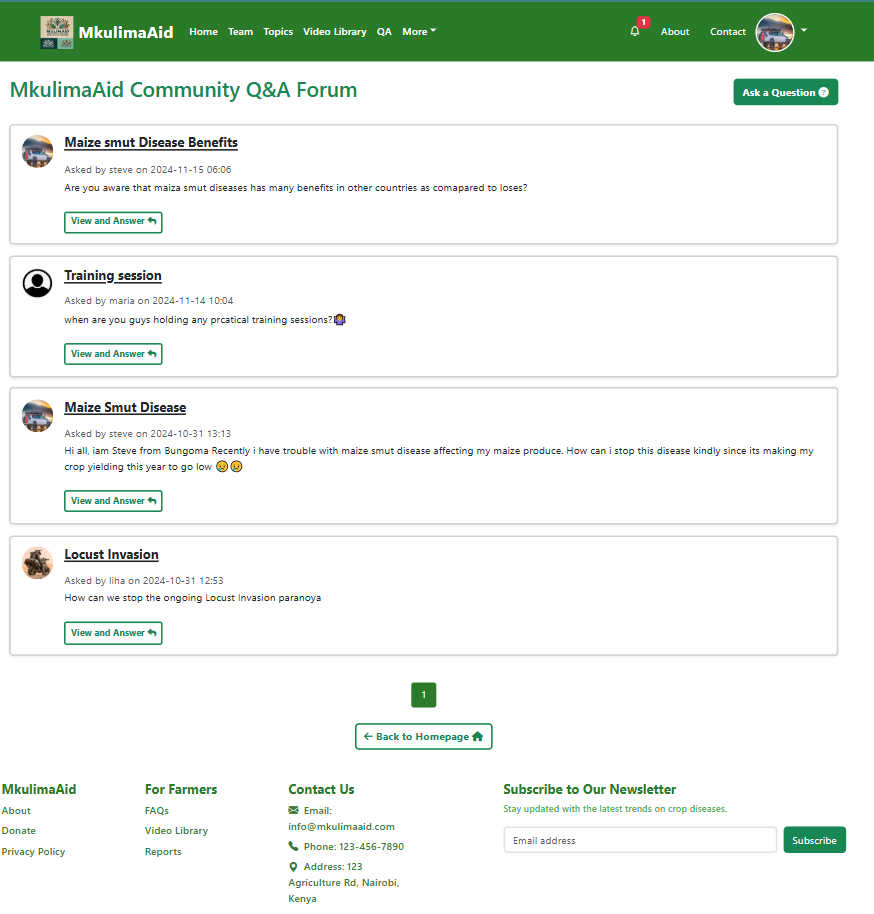
* **Trending Disease Alerts:** Updates on newly detected crop diseases.
* **Admin Announcements:** Important messages from system administrators.
* **Q&A Forum Updates:** Notifications for replies to user questions.
* **System Reminders:** Account security alerts and upcoming maintenance notifications.

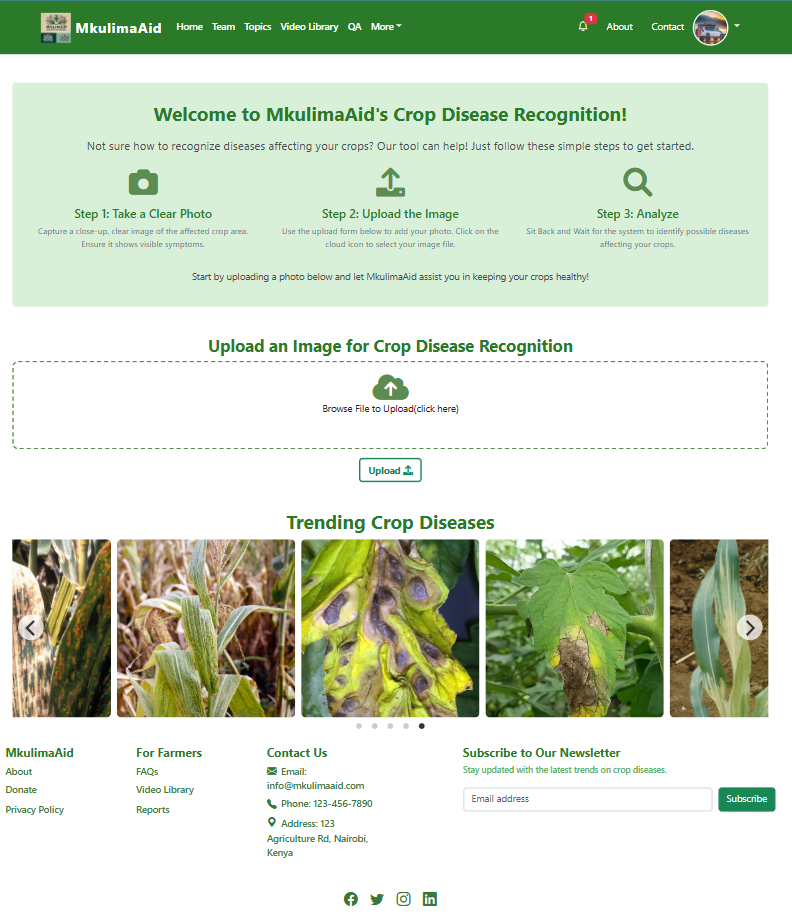
**2. Notification Delivery**

* **In-App Notifications:** Displayed via a notification icon on the dashboard.
* **Email Alerts:** Sent for critical updates (optional subscription).
* **SMS Notifications (Future Scope):** For farmers in rural areas with limited internet access.

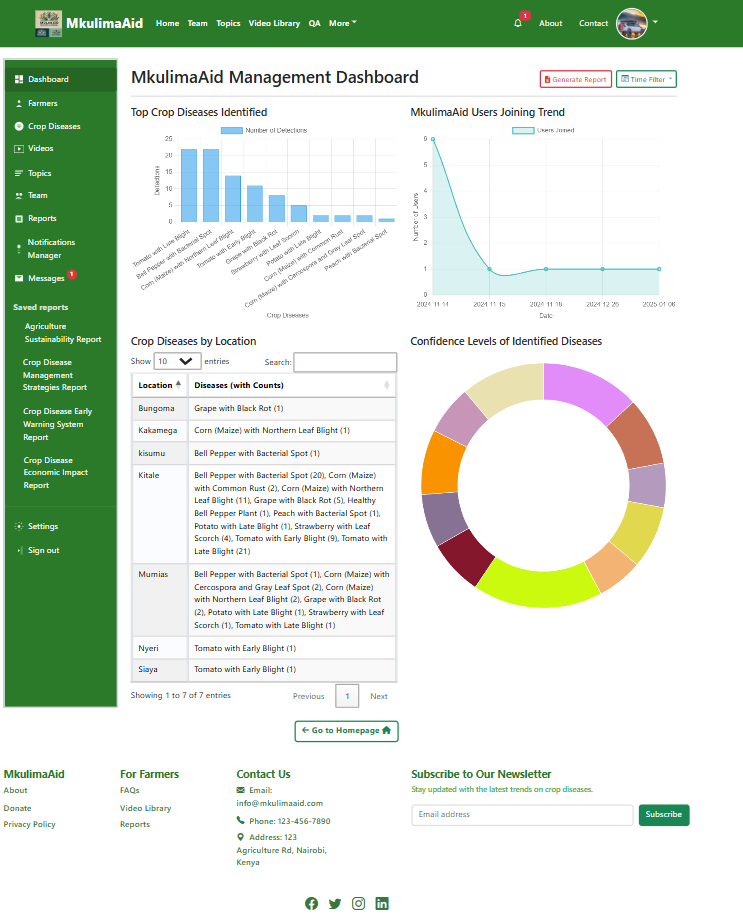
## **4.6 UI Wireframes & Mockups**

### **Questions and Answers forum**



**2. Farmers dashboard**  


### **3. Admin Dashboard**

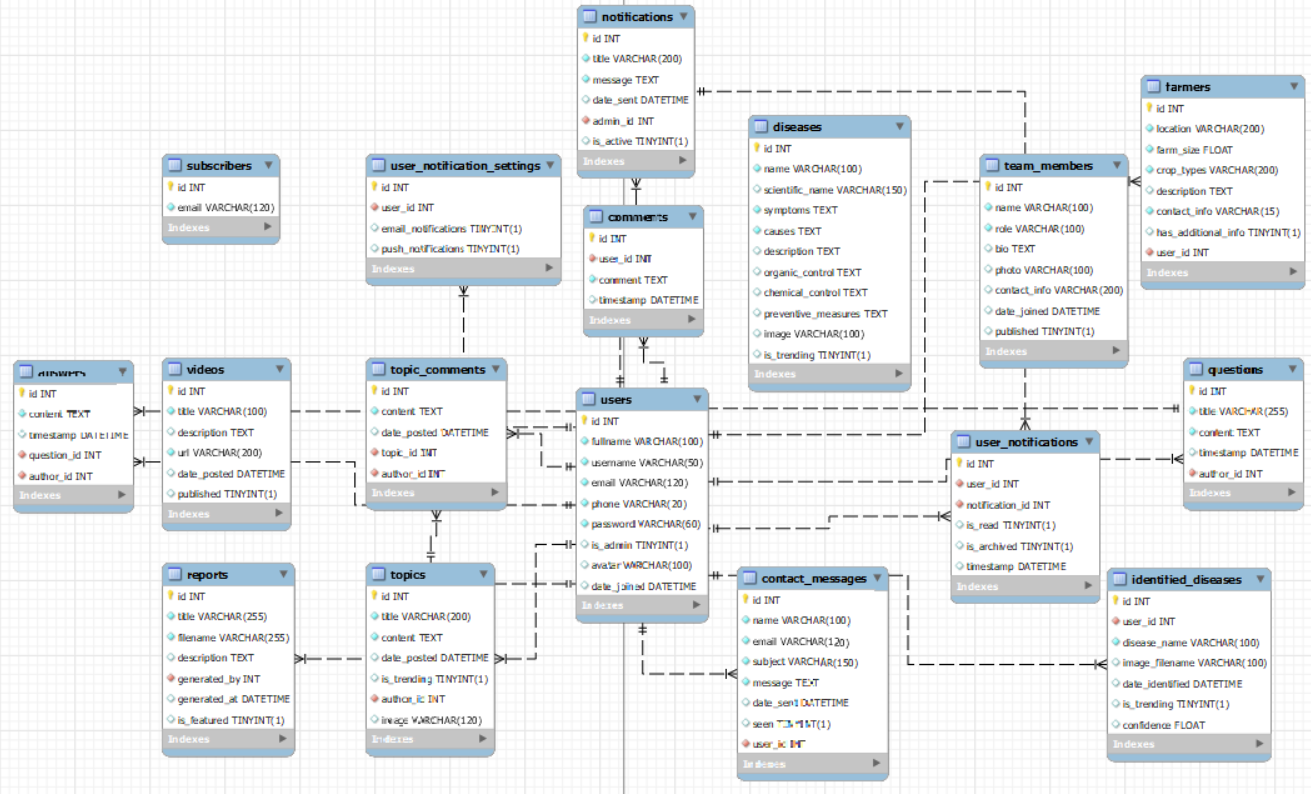
The User Interface Design chapter outlines the key interactions within MkulimaAid, ensuring a smooth and efficient experience for users. By incorporating role-based access control, quick navigation, an intuitive UI, and robust notification and reporting features, the system enhances user engagement and accessibility.

# **Chapter 5: Database Design**

The MkulimaAid system utilizes a relational database to manage user interactions, crop disease identification, and administrative functionalities. The database is structured using MySQL and follows a well-defined Entity-Relationship Diagram (EER) to ensure data integrity, scalability, and efficient retrieval.

## **5.1 Entity-Relationship Diagram (EER)**

The EER diagram below illustrates the core tables and their relationships:



**Key Tables**

* **Users** – Stores user details, including authentication credentials and profile information.
* **Diseases** – Contains crop disease information, including symptoms, causes, control methods, and images.
* **IdentifiedDiseases** – Stores disease identification records from users, including prediction confidence and timestamps.
* **Topics** – Manages discussion topics related to crop disease management.
* **Comments** – Records user feedback and discussions on topics.
* **Reports** – Stores system-generated and user-generated reports, with an option to feature important reports.
* **Notifications** – Used by admins to send important alerts to users.
* **Team** – Holds details of the system’s team members, including their roles and visibility status.

The relationships between these tables ensure seamless data flow and enable functionalities such as user authentication, disease identification tracking, topic discussions, notifications, and team management.

# **Chapter 6: Functional Module Design**

The MkulimaAid system consists of multiple functional modules that work together to provide crop disease recognition, user interaction, and administrative management. This chapter outlines the key components of the system and their functional requirements.

## **6.1 Key Functional Modules**

**1. User Management Module**

* Allows user registration, login, and authentication.
* Supports profile management, including uploading profile pictures.
* Enables users to change passwords and update personal details.

**2. Crop Disease Detection Module**

* Users can upload images of affected crops for disease identification.
* The system utilizes a pre-trained deep learning model for classification.
* Identified diseases are stored with confidence scores and timestamps.

**3. Trending Crop Diseases Module**

* Admins can add, edit, and remove trending diseases from the dashboard.
* Displays disease details, symptoms, causes, and control methods to users.

**4. Discussion Forum Module**

* Users can ask and answer questions related to crop diseases.
* Supports commenting and discussion threads on disease management.
* Displays user avatars, usernames, and timestamps for engagement.

**5. Reports & Analytics Module**

* Admins can generate and manage system reports.
* Provides visual analytics on user activity and identified diseases.
* Features filter options for time-based analysis (weekly, monthly, All time).

**6. Notifications & Messaging Module**

* Admins can send system-wide notifications about trending diseases.
* Users receive real-time updates and unread message counts.
* Implements a messaging-like notification system.

**7. Team Management Module**

* Allows admins to add, edit, and remove team members.
* Includes a publish/unpublish feature for team member profiles.

**8. Help Center Module**

* Provides a searchable knowledge base for user support.
* Categorizes common issues and FAQs for easy navigation.
* Includes a contact support form for user inquiries.

## **6.2 Functional Requirements**

1. **User Authentication**

* Users must log in to access system features.
* Passwords must meet security requirements.

1. **Crop Disease Detection**

* The system should classify uploaded crop images with high accuracy.
* Identified diseases must be stored for future reference.

1. **Content Management (Admin)**

* Admins should manage trending diseases, notifications, and reports.
* Admins should approve or remove user-generated content when necessary.

1. **User Engagement**

* Users can ask questions and participate in discussions.
* Users should receive real-time notifications.

1. **Data Storage & Security**

* All uploaded images and user data must be securely stored.
* System access should be role-based (admin vs. regular users).

# **Chapter 7: Security Design**

The MkulimaAid system implements robust security measures to protect against unauthorized access, misuse, and data breaches. This chapter outlines the key security aspects, including authentication, authorization, data validation, and adherence to ACID properties to ensure system integrity and reliability.

## **7.1 Authentication & Authorization**

**User Authentication**

* **Secure Login System:** Users must authenticate using email and password.
* **Strong Password Enforcement:** Passwords must contain at least 8 characters, including uppercase, lowercase, numbers, and special characters.
* **Session Management:** Implements session timeouts to prevent unauthorized access due to inactivity.
* **Hashed Passwords:** Uses bcrypt hashing to store passwords securely.

**Role-Based Authorization**

* **Admins vs. Regular Users:**
* Admins have full access to manage trending diseases, reports, notifications, and users.
* Regular users can only access crop disease detection, discussions, and help center.
* **Access Control:** Restricts access to sensitive routes using Flask-Login and Flask-Principal.
* **CSRF Protection:** Implements Cross-Site Request Forgery (CSRF) protection on all forms using Flask-WTF.

## **7.2 Data Validation & Input Sanitization**

* **Form Validation:** Ensures all user inputs meet expected formats using Flask-WTF validators.
* **File Upload Security:**
* Allows only image files for disease detection (e.g., .jpg, .png).
* Uses secure\_filename() to prevent malicious file uploads.
* **SQL Injection Protection:** Uses parameterized queries in SQLAlchemy to prevent injection attacks.
* **XSS Protection:** Escapes user-generated content in the forum and comments section to prevent Cross-Site Scripting (XSS) attacks.

## **7.3 Data Integrity & Adherence to ACID Properties**

The MkulimaAid system ensures database reliability through ACID (Atomicity, Consistency, Isolation, Durability) compliance:

1. **Atomicity**

* Transactions are all-or-nothing—if a database operation fails, it is rolled back.
* Example: If a user submits a disease report and an error occurs, the system ensures no partial data is stored.

1. **Consistency**

* Ensures the database remains in a valid state before and after transactions.
* Example: Disease entries must always contain valid scientific names and images.

1. **Isolation**

* Prevents concurrent transactions from interfering with each other.
* Example: Two admins updating the same disease report won't cause data corruption.

1. **Durability**

* All committed transactions are permanently stored, even in case of system failures.
* Uses database backups and transaction logs for recovery.

## **7.4 Secure Data Storage & Encryption**

* User passwords are encrypted using bcrypt hashing.
* Sensitive user data (e.g., emails) is stored securely with proper access controls.
* Database backups are performed regularly to prevent data loss.

## **7.5 Preventing Unauthorized Access & Misuse**

* Rate Limiting: Implements Flask-Limiter to prevent brute-force attacks on login.
* Logging & Monitoring: Tracks failed login attempts, admin actions, and suspicious activities.
* Auto Logout: Inactive users are automatically logged out for security.
* Implement Google’s Recaptcha to prevent botnet login attacks.

The MkulimaAid system follows best security practices to ensure user data protection, secure authentication, and system integrity. By enforcing access controls, secure data handling, and ACID compliance, the system remains robust, reliable, and secure against unauthorized access and misuse.

# **Chapter 8: Hardware and Software Requirements**

The MkulimaAid system requires a well-defined set of hardware and software specifications to ensure optimal performance, security, and reliability. This section outlines the technical specifications necessary for the smooth operation of the system, including hardware, software, security tools, database, programming languages, and network requirements.

## **8.1 Hardware Specifications**

**For Server Deployment**

* **Processor:** Intel Core i5 (or higher) / AMD Ryzen 5 (or higher)
* **RAM:** Minimum 8GB (Recommended: 16GB for high performance)
* **Storage**: Minimum 256GB SSD (Recommended: 512GB SSD or more)
* **Network:** High-speed internet connection (Minimum 10 Mbps for stable connectivity)
* **Backup Storage:** External HDD/SSD (1TB or more) for database and file backups

**For User Devices (Farmers & Admins)**

* **Processor:** Intel Core i3 (or higher) / Equivalent ARM processors for mobile devices
* **RAM:** Minimum 4GB
* **Storage:** At least 32GB available space
* **Display:** Minimum 720p resolution for proper UI rendering
* **Connectivity:** Internet access for online functionalities

## **8.2 Software Specifications**

**Operating System (OS)**

* **Server OS:** Ubuntu 20.04 LTS / CentOS / Windows Server 2019+
* **Client OS:** Windows 10/11, macOS, Linux, Android, iOS

**Database Management System (DBMS)**

* MySQL (Recommended: MySQL 8.0 for performance and security)
* SQLAlchemy ORM for database interaction

**Programming Languages**

* **Backend:** Python (Flask Framework)
* **Frontend:** HTML5, CSS3, JavaScript (Bootstrap, AJAX, jQuery)

**Development Tools & Libraries**

* Flask (for backend development)
* OpenCV & TensorFlow (for image classification of crop diseases)
* Flask-WTF (for secure form handling)
* Jinja2 (for templating)
* Chart.js / Recharts (for visualizations)

## **8.3 Security Software & Measures**

* **Firewall:** UFW (for Linux) / Windows Defender Firewall
* **Antivirus:** ClamAV (Linux) / Windows Defender
* **Encryption:** bcrypt (for password hashing)
* **SSL Certificate:** Let's Encrypt (for HTTPS encryption)
* **Access Control:** Flask-Login & Role-based Access Control (RBAC)

## **8.4 Backup & Recovery Tools**

* **Database Backup**
* Automated MySQL dumps (Daily scheduled backups)
* Cloud Backup Solutions (Google Drive, AWS S3)
* **File Storage Backup**
* Version control using Git/GitHub
* Local external storage backup (HDD/SSD)
* **Disaster Recovery**
* **Data Replication & Redundancy**: Ensures minimal downtime
* **Point-in-time recovery (PITR):** Allows restoration of data to a previous state

## **8.5 Network Requirements**

* **Server Bandwidth:** Minimum 20 Mbps for handling user requests efficiently
* **Client Connectivity:** Minimum 4G / WiFi for smooth access
* **Hosting Options:**
* Cloud Hosting (AWS / DigitalOcean / Linode)
* On-premise server (for self-hosted deployments)

## **8.6 Technologies Used**

**Frontend Technologies**

* HTML, CSS, JavaScript for responsive UI
* Bootstrap & jQuery for better user experience
* AJAX for asynchronous data loading

**Backend Technologies**

* Flask (Python-based framework) for API & business logic
* REST API for seamless communication between frontend & backend
* Session & Token-based authentication (Flask-Login & JWT)

**Machine Learning & Image Processing**

* TensorFlow & OpenCV for crop disease recognition

The MkulimaAid system relies on a robust combination of hardware, software, security, and networking technologies to ensure smooth performance, secure access, and reliable data storage. By utilizing cloud-based backups, security firewalls, and a scalable backend, the system is designed to handle high traffic loads, accurate disease identification, and secure user interactions efficiently.

# **Chapter 9: System Integration and Testing Plan**

The MkulimaAid system undergoes rigorous testing to ensure that all modules function together seamlessly, delivering optimal performance, security, and usability. This chapter outlines the integration process, testing methodologies, test schedules, performance benchmarks, and environment testing to validate system functionality and ensure it meets user expectations.

## **9.1 System Integration Approach**

The system integration process ensures that various independent modules of MkulimaAid—such as user authentication, disease detection, forum, notifications, and admin dashboard—work together as a cohesive system. The approach follows:

1. **Incremental Integration:** Each module is integrated one at a time, ensuring smooth communication and identifying errors early.
2. **Continuous Testing:** Automated and manual tests are conducted to validate integration points.
3. **Performance Benchmarking:** System behavior is tested under varying loads to ensure scalability.
4. **Error Handling & Debugging:** Integration bugs and conflicts are fixed before system deployment.

## **9.2 System Testing Phases**

### **Performance Testing**

Performance testing is conducted to measure system response time, database queries execution time, and server load handling.

* When to Perform Performance Testing?
* Before system deployment (pre-launch).
* After major updates or feature additions.
* During peak usage scenarios (simulating multiple users logging in).

Key Performance Metrics

|  |  |
| --- | --- |
| **Test Case** | **Expected Outcome** |
| User login time | Less than 3 seconds |
| Disease detection speed | Image processing in under 5 seconds |
| Page load time | Less than 2 seconds per page |
| Database query execution | Queries should run within 500ms |

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

User Acceptance Testing ensures that the system meets real-world requirements before deployment.

**UAT Process**

1. Select a group of farmers, admins, and testers.
2. Allow users to interact with all major system features (e.g., disease recognition, forum, and dashboard).
3. Gather feedback on usability, errors, and performance.
4. Fix reported issues and re-test.

### **Test Schedule & Timeline**

The system testing follows a structured timeline to ensure all aspects are covered.

|  |  |  |
| --- | --- | --- |
| **Testing Phase** | **Duration** | **Description** |
| Unit Testing | 2 weeks | Test individual modules separately. |
| Integration Testing | 2 weeks | Test interactions between modules. |
| Performance Testing | 1 week | Load testing, response times. |
| Security Testing | 1 week | Penetration testing & data validation. |
| User Acceptance Testing | 1 week | Farmers & admins test real-world usage. |
| Final Bug Fixes & Review | 1 week | Address feedback & ensure readiness. |

### **Environment Testing**

MkulimaAid is tested across different operating systems, devices, and browsers to ensure a consistent experience.

|  |  |  |
| --- | --- | --- |
| **Environment** | **Tested Components** | **Outcome** |
| Windows 10/11 | Full system test | ✅ Works fine |
| Ubuntu 20.04+ | Flask backend & database | ✅ Works fine |
| macOS | Frontend, authentication | ✅ Works fine |
| Android/iOS | Mobile responsiveness | ✅ Works fine |
| Browsers (Chrome, Edge, Firefox) | UI compatibility | ✅ Works fine |

### **Testing Plan Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Type** | **Objective** | **Tools Used** | **Outcome** |
| Unit Testing | Verify individual functions | PyTest, Unittest | Functions execute correctly |
| Integration Testing | Ensure modules work together | Postman (API), Manual Testing | Seamless interaction |
| Performance Testing | Measure response times & load handling | JMeter, Locust | System performs efficiently |
| Security Testing | Identify vulnerabilities | OWASP ZAP, SQLMap | Secure against threats |
| User Acceptance Testing | Validate usability & real-world functionality | Live user testing | Meets user requirements |

The MkulimaAid system undergoes extensive testing to ensure it meets performance, security, and user experience standards. By integrating modules incrementally, conducting performance tests, ensuring cross-platform compatibility, and gathering real user feedback, the system is validated for seamless operation before deployment.

# **Chapter 10: Deployment and Maintenance Plan**

The MkulimaAid system deployment ensures that the platform is properly launched, configured, and made available to users. This chapter outlines the deployment strategy, maintenance approach, and future scalability plans to ensure the system remains efficient, secure, and up-to-date.

## **10.1 Deployment Plan**

### **Deployment Strategy**

The system follows a phased deployment approach to minimize risks and ensure smooth adoption.

**Deployment Stages**

|  |  |
| --- | --- |
| **Stage** | **Description** |
| Development Phase | Code and test system functionalities locally. |
| Testing Environment | Deploy on a test server to identify bugs before launch. |
| Staging Phase | Conduct final validation and user acceptance testing (UAT). |
| Production Deployment | Deploy the system live for real users. |
| Post-Deployment Monitoring | Continuously monitor performance and fix any issues. |

### **Hosting & Server Configuration**

* **Hosting Provider:** DigitalOcean / AWS / Google Cloud
* **Web Server:** Nginx / Apache
* **Application Server:** Flask running on Gunicorn
* **Database:** MySQL / PostgreSQL
* **Domain Setup:** Configure a custom domain (e.g., www.mkulimaaid.com)
* **SSL Certificate:** Install SSL for secure HTTPS access
* **Email Setup:** Use SMTP for sending notifications (e.g., Gmail SMTP, SendGrid)

## **Deployment Steps**

* **Set up the server:** Install required dependencies, database, and Flask environment.
* **Configure environment variables:** Set API keys, database credentials, and security configurations.
* **Migrate the database:** Apply database migrations using Flask-Migrate.
* **Deploy the application:** Use Gunicorn and Nginx for running the Flask app.
* **Enable monitoring:** Set up logging and performance monitoring (e.g., New Relic, Prometheus).
* **Test the deployment:** Conduct final system checks before full launch.
* **Go live:** Make the system available to users.

## **10.2 Maintenance Plan**

### **Regular Updates & Bug Fixes**

* **Security Patches:** Apply updates to fix vulnerabilities.
* **Feature Enhancements:** Add new functionalities based on user feedback.
* **Performance Optimization:** Improve database queries, caching, and server response time.

### **Backup & Disaster Recovery**

To prevent data loss, automated backups are scheduled.

|  |  |  |
| --- | --- | --- |
| **Backup Type** | **Frequency** | **Storage Location** |
| Database Backup | Daily | Cloud Storage / External Drive |
| Codebase Backup | Weekly | GitHub / GitLab / Bitbucket |
| Server Snapshot | Monthly | Cloud Provider |

### **System Monitoring**

* **Logging & Alerts:** Monitor errors using Flask logging, Sentry, or Datadog.
* **Uptime Monitoring:** Use Pingdom or UptimeRobot to detect downtime.
* **Performance Monitoring:** Track load time, API performance using New Relic or Prometheus.

## **10.3 Scalability & Future Enhancements**

### **Scaling Strategy**

* **Database Scaling:** Use replication and indexing for high performance.
* **Load Balancing:** Distribute traffic across multiple servers.
* **Microservices Architecture:** Separate disease detection, notifications, and user management into independent services.

### **Future Enhancements**

* Mobile App Development for better accessibility.
* AI Model Improvements for better crop disease detection.
* Multilingual Support to expand user reach.

The MkulimaAid deployment ensures a stable and secure system launch, while the maintenance plan ensures ongoing reliability, performance, and security. By implementing regular updates, backups, and monitoring, the system remains efficient and scalable for future growth.