**Tomato Care**

|  |
| --- |
| **Riphah** |

**By:**

**Muhammad Salman Afaq**

**24775**

**Usman Afaq**

**24779**

**Rafaqat Ahmad**

**24784**

**Supervised by:**

**Muhammad Usman Karim**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Spring 2023**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Computer Science**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: [15 December 2023]

**Final Approval**

This is to certify that we have read the report submitted by ***Muhammad Salman Afaq (24775),Usman Afaq (24779), Rafaqat Ahmad (24784)*** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Computer Science (BSCS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Computer Science (BSCS).

**Committee:**

|  |  |
| --- | --- |
| **1** | Muhammad Usman Karim  (Supervisor) |
|  |  |
| **2** | Dr. Muhammad Musharraf  (Head of Department) |

**Declaration**

We hereby declare that this document “**Tomato Disease Detection**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **Muhammad Usman Karim**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muhammad Salman Afaq**

**24775**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Usman Afaq**

**24779**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Rafaqat Ahmad**

**24784**

**Dedication**

Our final year project is dedicated to our parents, friends and teachers, whose love and support have been our pillars of strength. To our professors and especially supervisor

"M. Usman Karim", your guidance has shaped our academic journey.

**Acknowledgement**

First of all we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

We extend our heartfelt gratitude to our project supervisor “**Muhammad Usman Karim”**, whose unwavering support, invaluable guidance, and continuous mentorship were indispensable to the successful completion of this project. Their dedication and commitment have been a driving force behind our work.

Furthermore, we want to say a big thank you to our family and friends. They have been our constant source of support and motivation, always encouraging us to do our best and be honest and hardworking.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muhammad Salman Afaq**

**24775**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Usman Afaq**

**24779**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Rafaqat Ahmad**

**24784**

**Table of Contents**

|  |  |
| --- | --- |
| List of Figures | 1 |
| List of Tables | 2 |
| Chapter 1: Introduction | 3 |
| * 1. Opportunity & Stakeholders | 4 |
| * 1. Motivations and Challenges | 5 |
| * 1. Goals and Objectives | 6 |
| * 1. Solution Overview |  |
| * 1. Report Outline |  |
| Chapter 2: Literature / Market Survey |  |
| 2.1 Introduction |  |
| 2.2 Literature Review/Technologies Overview |  |
| 2.3 Summary |  |
| Chapter 3: Requirement Engineering |  |
| 3.1 Introduction |  |
| 3.2 Problem Scenarios |  |
| 3.3 Functional Requirements |  |
| 3.4 Non-Functional Requirements  3.5 SQA activities: Defect Detection  3.5.1 Test Case Design |  |
| Chapter 4: System Design |  |
| 4.1 Introduction |  |
| 4.2 Architectural Design |  |
| 4.3 Detailed Design  4.4 SQA activities: Defect Detection  4.4.1 Test Case Design |  |
| Chapter 5: Implementation |  |
| 5.1 Endeavour (Team + Work + Way of Working) |  |
| 5.2 Flow Control/Pseudo codes |  |
| 5.3 Components, Libraries, Web Services and stubs  5.4 IDE, Tools and Technologies |  |
| 5.5 Best Practices / Coding Standards  5.5.1 Computer Science Practices  5.5.2 Development Practices & Standards |  |
| 5.6 Deployment Environment  5.7 SQA activities: Defect Detection  5.7.1 Test Case Design (White box) |  |
| 5.8 Summary |  |
| Chapter 7: Conclusion and Outlook |  |
| 7.1 Introduction |  |
| 7.2 Achievements and Improvements |  |
| 7.3 Critical Review |  |
| 7.4 Future Recommendations/Outlook |  |
| 7.5 Summary |  |
| References |  |
| Appendices |  |
| Appendix-A: Software Requirements Specifications (SRS) |  |
| Appendix-B: Design Documents |  |
| Appendix-C: Coding Standards/Conventions |  |
| Appendix-D: Test Scenarios |  |
| Appendix-E: Work Breakdown Structure |  |
| Appendix-F: Roles & Responsibility Matrix |  |

**List of Figures**

|  |  |
| --- | --- |
| 1.1 Caption of first figure of first chapter | 6 |
| 1.2 Caption of second figure of first chapter | 7 |
| 2.1 Caption of first figure of second chapter | 14 |
| 2.2 Caption of second figure of second chapter | 22 |
| 2.3 Caption of third figure of second chapter | 26 |
| 5.1 Caption of first figure of fifth chapter | 49 |
| 5.2 Caption of second figure of fifth chapter | 49 |

**List of Tables**

|  |  |
| --- | --- |
| 1.1 label of first table of first chapter | 6 |
| 1.2 label of second table of first chapter | 7 |
| 2.1 label of first table of second chapter | 14 |
| 2.2 label of second table of second chapter | 22 |
| 2.3 label of third table of second chapter | 26 |
| 5.1 label of first table of fifth chapter | 49 |
| 5.2 label of second table of fifth chapter | 49 |

**Abstract**

The project, titled **"Tomato Disease Detection Using Machine Learning"** addresses the critical challenge of safeguarding tomato crops against diseases that threaten global agricultural sustainability. Focused on Early Blight, Late Blight, and Leaf Mold, our objective is to develop an efficient automated system for early disease detection, providing farmers with a tool for timely intervention.

Leveraging Convolutional Neural Networks (CNNs) and transfer learning on a diverse dataset, our model achieved a validation accuracy of 95%. Real-world testing demonstrated its robustness, showcasing high sensitivity and specificity.

This Tomato Disease Detection System stands as a practical solution, emphasizing the potential of machine learning in precision agriculture for sustainable food production.

**Chapter1:Introduction  
Tomato Care** is a complete agricultural solution which is designed to solve the problems of farmers. This advanced platform, which is available for both web and Android, leverages cutting-edge technology to revolutionize the way farmers detect and manage diseases in their tomato crops. With current methods falling short in addressing the **complexities** of tomato crop health, **Tomato Care** aims to provide an efficient and innovative solution to empower farmers. By seamlessly integrating image-based disease detection, treatment recommendations, and real-time updates, **Tomato Care** ensures a holistic approach to crop health management.

This user-friendly platform not only enhances the **accuracy** of disease identification but also encourages a collaborative environment where farmers can make informed decisions for optimal yields. The old ways of finding diseases in tomato plants are not always good enough, and it is hard to find and fix problems quickly. But **Tomato Care** is here to help. It uses advanced technology to make sure we find and understand the diseases in tomato crops better and faster.

**1.1 Opportunity & Stakeholders**

**1.1.1 Opportunity**

* Pressing issue of tomato diseases in agriculture.
* Recognition of the need for a more efficient and accessible disease detection solution.
* Conducted extensive market research.
* Identified a gap in existing solutions.
* Acquired necessary skills in machine learning and data analysis.
* Drew inspiration from pioneering projects and research in plant disease detection.
* Focus on collecting relevant data.
* Development of a robust machine learning model.
* Emphasis not only on technology but also on making a tangible impact in the agricultural sector.
* Commitment to continuous testing and refinement as the cornerstone of the project.

**1.1.2 Stakeholders:**

* Farmers: Detecting diseases early helps farmers grow more tomatoes and spend less on treatments, improving their income.

|  |
| --- |
| * 1. **Motivations and Challenges**   **1.2.1 Motivations**   * Improve tomato crop health and productivity. * Increase yields by preventing diseases. * Support sustainable farming, reduce pesticide reliance. * Safeguard farmers' livelihoods and promote economic stability. * Ensure a stable global tomato supply, mitigating food shortages. * Leverage technology, like machine learning, for innovative solutions. * Empower farmers for proactive disease management. * Safeguard investments, reduce financial losses from diseases.   **1.2.2 Challenges**   * Farmers encounter a critical challenge in timely and accurate disease detection in tomato plants. * Timely and accurate detection directly influences crop yields. * Traditional disease identification methods rely on visual inspection. * Visual inspection is prone to errors and is time-consuming. * Farmers require an accessible and reliable solution. * Quick disease identification enables farmers to take proactive measures.   1. **Goals And Objectives**   **1.3.1 Goals**   * To utilize artificial intelligence and image analysis to process images of tomato plants, training models to recognize visual cues associated with various diseases. * To make the efforts of farmer less and make the system to detect the disease and suggest the treatment.   **1.3.1 Objectives**   * To develop a system using smart technology to analyze pictures of tomato plants and automatically spot signs of diseases. * Create a monitoring setup that can quickly identify early signs of diseases in tomato crops, enabling timely and targeted interventions to prevent the spread of infections and minimize crop damage.   1. **Solution Overview**   The solution for tomato disease detection integrates AI-powered image analysis and sensor technology to enable early and accurate identification of diseases. Precision agriculture practices optimize resource use, directing interventions effectively. The system includes real-time monitoring and automated alerts for timely, targeted actions. Overall, it aims to minimize crop damage, optimize resource utilization, and provide farmers with efficient decision support tools.   * 1. **Report Outline**   **1.5.1 Introduction**  Brief overview of the importance of tomato disease detection  Purpose and scope of the report  **1.5.2 Background**  Current challenges in tomato disease management  Overview of existing detection methods and their limitations  **1.5.3 Methodology**  Description of the AI-powered image analysis and sensor technology  Explanation of data collection and model training processes  **1.5.4 Key Components of the Solution**  AI-powered Image Analysis  Training models on diverse datasets  Recognition of visual patterns related to tomato diseases  Sensor Technology  Monitoring parameters like leaf color and texture  Real-time data collection for enhanced disease detection  **1.5.5 Precision Agriculture Practices**  Optimization of resource utilization  Targeted interventions based on data-driven insights  **1.5.6 Early Intervention Strategies**  Real-time monitoring system  Automated alert mechanisms for timely farmer notifications  **1.5.7 Benefits**  Timely disease detection and intervention  Resource optimization and efficiency gains  Automated decision support for farmers  **1**.**7.8 Challenges and Considerations**  Timely disease detection and intervention  Resource optimization and efficiency gains  Automated decision support for farmers  **1.5.9 Challenges and Considerations**  Potential challenges in implementing the solution  Mitigation strategies and considerations for successful deployment  **1.5.10 Conclusion**  Summary of key findings and insights  **1.5.11 Recommendations**  Suggestions for further research or improvements  Concluding remarks  **1.5.12 References**  Citations for sources and literature referenced in the report.  **1.5.13 Appendix**  Supplementary information, data, or details supporting the main report content. |
|  |
|  |
|  |
|  |
|  |
|  |

**Chapter 2: Literature / Market Survey**

**2.1 Introduction**

Tomato Care is a complete agricultural solution which is designed to solve the problems of farmers. This advanced platform, which is available for both web and Android, leverages cutting-edge technology to revolutionize the way farmers detect and manage diseases in their tomato crops. With current methods falling short in addressing the complexities of tomato crop health, Tomato Care aims to provide an efficient and innovative solution to empower farmers.

By seamlessly integrating image-based disease detection, treatment recommendations, and real-time updates, Tomato Care ensures a holistic approach to crop health management. This user-friendly platform not only enhances the accuracy of disease identification but also encourages a collaborative environment where farmers can make informed decisions for optimal yields.

The Tomato Disease Detection market survey involves a thorough analysis of key players, technology trends, and customer needs in the agricultural technology sector. By assessing market size, growth potential, and competitive dynamics, businesses can gain valuable insights for informed decision-making.

**2.2 Literature Review/Technologies Overview**

The literature review and technologies overview for Tomato Disease Detection look at what researchers and experts have already studied and developed in this area. They've been using fancy technologies like machine learning and computer vision to identify diseases in tomatoes. One cool thing they use is image recognition, especially with something called convolutional neural networks. These technologies help to quickly spot and classify diseases just by looking at pictures of tomato leaves. Also, there are smart sensors and data analytics that help farmers monitor their crops in real-time, catching diseases early. All of this technology together helps in managing diseases in tomatoes better, reducing losses, and making farming more sustainable. As scientists keep studying and improving these technologies, the way we detect and handle tomato diseases keeps getting better.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Paper ID** | **Domain** | **Document** | **Algorithm** | **Accuracy** | **Reference** |
| **1** | Machine Learning | **Tomato Leaf Disease Detection Using Convolutional Neural Networks** | Convolutional Neural Networks (CNNs) | 99.52% | https://ieeexplore.ieee.org/document/9988540/ |
| **2** | Deep Learning | **Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network** | Inception V3, Rainbow concatenation | 98.52% | https://www.mdpi.com/1424-8220/21/23/7987 |
| **3** | Deep Learning | **Tomato Leaf Disease Detection using Convolution Neural Network** | Convolutional Neural Networks (CNNs), Transfer learning with Inception V3 | 91.2% | https://www.sciencedirect.com/science/article/pii/S1877050920306906 |
| **4** | Deep  Learning | **Disease detection on the leaves of the tomato plants by using deep learning** | Convolutional Neural Networks (CNNs) | 99% | https://ieeexplore.ieee.org/document/9397001 |
| **5** | Machine Learning | **Tomato leaf disease detection and classification using image processing and machine learning** | Vector Machines (SVMs), K-Nearest Neighbors (KNN), Random Forests, Convolutional Neural Networks (CNNs) | 97.5% | https://pubmed.ncbi.nlm.nih.gov/27172010/ |

**2.3 Summary**

The research on Tomato Disease Detection shows that scientists are using smart technologies like machine learning and computer vision to find and manage diseases in tomato plants. They look at pictures of tomato leaves to figure out what kind of disease might be there. The market survey tells us that companies are using these technologies a lot, especially in precision farming. They also use smart sensors and data analytics to keep an eye on the plants in real-time, finding diseases early. This helps farmers take care of their crops better, reducing losses, and making tomato farming more sustainable. It's a growing and changing area with lots of opportunities for making tomato plants healthier.

**Chapter 3: Requirement Engineering**

**3.1 Introduction**

In the realm of agricultural technology, the quest for efficient and accurate disease detection in tomato crops has led to the development of advanced solutions through Requirement Engineering. This process involves systematically capturing, analyzing, and defining the needs and specifications for a technology or system tailored to detect diseases in tomatoes.

Understanding the specific requirements is crucial for designing effective solutions that address the challenges faced by farmers in managing crop health. From the types of diseases common in tomato plants to the technological capabilities required for timely detection, Requirement Engineering serves as the foundation for developing robust and user-friendly Tomato Disease Detection systems.

This introduction sets the stage for exploring the detailed requirements essential for creating effective and impactful solutions in the domain of tomato crop health management.

**3.2 Problem Scenarios**

**3.2.1 Context:**

In tomato farming, farmers often face challenges in timely identifying and managing diseases that affect their crops. The lack of an efficient system for disease detection leads to increased crop losses and poses a threat to overall crop health.

**3.2.2 Problem:**

The absence of a reliable and quick disease detection system makes it difficult for the farmer to take prompt action. There is a need for a technology solution that can accurately identify the type of disease affecting the tomato plants, enabling the farmer to implement timely and targeted interventions to mitigate the spread of the disease.

**3.2.3 Goals:**

Develop a Tomato Disease Detection System that can accurately identify various diseases based on visual cues from tomato plant leaves.

Ensure the system is user-friendly and accessible to farmers with varying levels of technological expertise.

Enable timely detection of diseases to facilitate proactive measures, reducing crop losses and promoting overall crop health.

Enhance the efficiency of disease management practices through the integration of advanced technologies, such as machine learning and computer vision.

* 1. **Functional Requirements**

**3.3.1 User / Farmer**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| **1.1** | User shall be able to Sign Up. |
| **1.2** | User shall be able to login to their account. |
| **1.3** | User shall be able to view and edit their profile. |
| **1.4** | User can upload images for disease detection. |
| **1.6** | User can access a personalized calendar for plant care. |
| **1.7** | User can give feedback. |
| **1.8** | User can also interact with the chatbot related to the content. The chatbot will provide relevant answers. |
| **1.9** | User can also view the reports. |

* + 1. **Admin**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| **2.1** | Admin shall be able to login into the system. |
| **2.2** | Admin shall be able to Add, edit or remove diseases. |
| **2.3** | Admin shall be able to manage treatments. |
| **2.4** | Admin shall be able to view and manage user’s profiles. |
| **2.5** | Admin can also manage localized alerts. |
| **2.6** | Admin can review the reports. |
| **2.7** | Admin shall be able to control the chatbot functions. |

**3.4 Non-Functional Requirements**

**3.4.1 Performance:**

The system should provide real-time or near-real-time processing for disease detection to ensure prompt results for farmers.

Response time for uploading images and receiving results should be within an acceptable range (e.g., less than 5 seconds).

**3.4.2 Security:**

User authentication and authorization mechanisms should be implemented to ensure that only authorized users (farmers and admins) can access the system.

Data privacy and integrity must be maintained, especially for sensitive information related to the farmer's crops.

**3.5 SQA activities: Defect Detection**

**3.5.1 Test Case Design**

**TD: Test Data**

|  |  |
| --- | --- |
| **Test Data** | TD-1 |
| **Form** | Login |
| **Stakeholder** | User |
| **Field** | Phone Number |
| **Technique** | Equivalence Partitioning |
| **Valid** | * Correct Length * Include Numeric Characters * Country Code * No Space or Special Character |
| **Invalid** | * Not contains ‘@’ * Include characters * Special characters * Not End with ‘.’ |

|  |  |
| --- | --- |
| **Test Data** | TD-2 |
| **Form** | Login |
| **Stakeholder** | User |
| **Field** | Password |
| **Technique** | Equivalence Partitioning |
| **Valid** | * Password Lenth should be >=9 * Includes two special characters * Includes uppercase and lowercase character * Includes one numeric character |
| **Invalid** | * Password length < 9 * No uppercase and lowercase characters * No special character * No numeric character |

|  |
| --- |
| **Chapter 4: System Design** |
| **4.1 Introduction**  Tomato disease detection plays a crucial role in precision agriculture, enabling early identification and intervention to minimize crop losses and ensure food security. Conventional methods for tomato disease detection rely on visual inspection by experts, which is labor-intensive, time-consuming, and prone to human error. To address these limitations, researchers have developed automated tomato disease detection systems using image processing and machine learning techniques.  **4.1.1 Data Collection:**   * **Image Acquisition:** Capture high-quality images of tomato leaves under controlled lighting conditions using cameras or mobile devices. * **Image Preprocessing:** Enhance the quality of the acquired images by adjusting brightness, contrast, and color balance, and remove noise and irrelevant background elements. * **Image Segmentation:** Isolate the leaf region from the background, segmenting individual leaves if necessary, to focus the analysis on the areas of interest. * **Feature Extraction:** Extract relevant features from the segmented leaf images, such as texture, color, and shape characteristics, which represent the visual patterns associated with specific tomato diseases.   **4.1.2 Model Training:**   * **Dataset Preparation:** Create a comprehensive dataset of labeled tomato leaf images, representing various diseases and healthy leaves. * **Model Selection:** Choose appropriate machine learning algorithms, such as convolutional neural networks (CNNs), support vector machines (SVMs), or random forests, based on the nature of the extracted features and the desired accuracy. * **Model Training:** Train the selected machine learning models on the labeled dataset, allowing them to learn the relationship between extracted features and disease labels.   **4.1.3 Disease Detection:**   * **Image Classification:** Apply the trained machine learning models to classify new, unseen tomato leaf images into healthy or diseased categories. * **Disease Identification:** For classified diseased images, identify the specific disease type based on the model's output and associated disease classes. * **Visualization and Interpretation:** Present the classification and identification results in a clear and understandable manner, visualizing the detected diseases and providing explanations for the model's decisions.   **4.1.4 Real-world Deployment:**   * **Integration with Agricultural Systems:** Integrate the tomato disease detection system into existing agricultural management systems to facilitate seamless data acquisition, analysis, and decision-making. * **Mobile Application Development:** Develop mobile applications that enable farmers and agricultural workers to capture leaf images and receive real-time disease detection results using their smartphones or tablets. * **Continuous Monitoring and Feedback:** Continuously monitor the performance of the deployed system, collecting feedback from users, and refining the models as needed to maintain high accuracy and generalizability. |
| * 1. **Architectural Design** |
| **4.3 Detailed Design** |
| **4.3.1 Use Cases**   * **Farmer**      * **Admin**      * **Sign Up:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | Sign Up | | | | **Actors** | | Farmer | | | | **Summary** | | The farmer registers a new account on the platform. | | | | **Pre-Conditions** | | The farmer is not logged in to the platform. | | | | **Post-Conditions** | | The farmer creates an account with basic information like name, email, phone number, and password. | | | | **Special Requirements** | | The email address should be unique and valid. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The farmer opens the Sign Up page. | | 2 | The system displays the Sign Up form. | | 3 | The farmer fills out the form with required information. | | 4 | The system validates the information and creates an account upon successful validation. | | **Alternative Flow** | | | | | | 1 | If the entered email is already registered, the system displays an error message and prompts the farmer to use a different email. | | 2 | If the password doesn't meet the complexity requirements, the system prompts the farmer to choose a stronger password. |  * **Login:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | Login | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The farmer or admin accesses their account by providing login credentials. | | | | **Pre-Conditions** | | The user has a registered account. | | | | **Post-Conditions** | | The user is logged into the platform and has access to their account features. | | | | **Special Requirements** | | The password should be entered correctly. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The user opens the Login page. | | 2 | The system displays the Login form. | | 3 | The user enters their email address and password. | | 4 | The system validates the credentials. | | **Alternative Flow** | | | | | | 1 | If the entered credentials are incorrect, the system displays an error message and allows the farmer to attempt login again. | | 2 | If the farmer forgets the password, there is a "Forgot Password" option that allows them to reset the password |  * **View Profile:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **View Profile** | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The user views their profile information. | | | | **Pre-Conditions** | | The user is logged into the platform. | | | | **Post-Conditions** | | The user can see their profile details like name, email address, phone number, and other relevant information. | | | | **Special Requirements** | | None | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The user clicks on the "Profile" link or icon. | | 2 | The system displays the user's profile page with their information. | | **Alternative Flow** | | | | 1 | None | |  * **Edit Profile:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Edit Profile** | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The user updates their profile information. | | | | **Pre-Conditions** | | The user is logged into the platform. | | | | **Post-Conditions** | | The user's profile information is updated with the changes. | | | | **Special Requirements** | | None | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The user clicks on the "Edit Profile" button. | | 2 | The system displays the editable profile form with the user's current information. | | 3 | The user modifies the desired information on the form. | | 4 | The system validates the input and updates the user's profile with the changes. | | **Alternative Flow** | | | | 1 | If the farmer attempts to submit invalid information, the system displays an error message and prompts the farmer to correct the information. | |  * **Upload Images:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Upload Images** | | | | **Actors** | | Farmer | | | | **Summary** | | The farmer uploads images of their plants to identify potential diseases. | | | | **Pre-Conditions** | | The farmer is logged into the platform and has images of their plants. | | | | **Post-Conditions** | | The system analyzes the images and provides the farmer with potential disease diagnoses and information. | | | | **Special Requirements** | | The image format should be supported by the system. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The farmer clicks on the "Disease Detection" feature. | | 2 | The system displays the image upload interface. | | 3 | \The farmer selects and uploads the images of their plants. | | 4 | The system analyzes the images using image recognition technology. | | **Alternative Flow** | | | | | | 1 | If the uploaded images are of an unsupported format or size, the system prompts the farmer to upload valid images. | | 2 | If there is an issue with the image processing, the system notifies the farmer and recommends re-uploading the images. |  * **Access Calendar:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Access Calendar** | | | | **Actors** | | Farmer | | | | **Summary** | | The farmer receives a personalized calendar with recommended tasks and schedules for their specific crops based on their location and climate. | | | | **Pre-Conditions** | | The farmer has logged in and entered information about their crops, location, and planting season. | | | | **Post-Conditions** | | The farmer has a customized calendar with reminders for essential plant care activities like: **Watering, Fertilization, Pest and disease control, Harvesting** | | | | **Special Requirements** | | Plant care calendar section. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The system automatically generates a personalized calendar after the farmer provides required information. | | 2 | The farmer can filter and prioritize tasks based on urgency or category. | | 3 | The calendar displays tasks with dates, deadlines, and instructions for each activity. | | 4 | The farmer can set reminders for upcoming tasks. | | **Alternative Flow** | | | | 1 | If there are no personalized recommendations for the farmer, the system displays a message suggesting that the farmer has no specific activities scheduled and encourages them to check back later. | |  * **Provide Feedback:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Provide Feedback** | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The user provides feedback on the platform's features and functionalities. | | | | **Pre-Conditions** | | The user is logged into the platform and has encountered an issue or has suggestions for improvement. | | | | **Post-Conditions** | | The feedback is communicated to the admin for review and consideration. | | | | **Special Requirements** | | Feedback form. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The user opens the "Feedback" section. | | 2 | **Describe the issue or suggestion in detail. Attach screenshots or relevant media for clarity.** The system sends the feedback to the admin for review. | | 3 | Submits the feedback. | | 4 | System acknowledges the feedback. | | **Alternative Flow** | | | | 1 | If the feedback form submission fails due to technical issues, the system displays a message informing the farmer about the problem and advises them to try again later. | |  * **Chatbot Interaction:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Interact with Chatbot** | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The user interacts with the chatbot to ask questions, receive assistance, and obtain information. | | | | **Pre-Conditions** | | The user is logged into the platform and has access to the chatbot feature. | | | | **Post-Conditions** | | The user receives answers to their questions and assistance | | | | **Special Requirements** | | Chatbot functionality. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | Farmer accesses the chatbot interface. | | 2 | Engages in a conversation with the chatbot. | | **Alternative Flow** | | | | 1 | If the chatbot encounters a question it cannot answer, it offers to escalate the query to a human support team and provides relevant contact details. | |  * **View Reports:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **View Reports** | | | | **Actors** | | Farmer, Admin | | | | **Summary** | | The user accesses and reviews reports generated by the platform to gain insights into their farming practices and crop performance. | | | | **Pre-Conditions** | | The user is logged into the platform and has access to the reporting feature. | | | | **Post-Conditions** | | The user gains valuable insights into their farming practices and crop performance based on the generated reports. | | | | **Special Requirements** | | Reports should be generated regularly and updated with new data. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The user accesses the reporting section of the platform. | | 2 | The system displays available reports, categorized by type, such as **Crop Performance Reports, Resource Consumption Reports, Soil Health Reports, Environmental Impact Reports** | | 3 | The user selects a specific report to view. | | 4 | The system displays the report in an interactive format, allowing the user to  **Filter and analyze data based on specific criteria.**  **Generate charts, graphs, and visualizations for better comprehension.**  **Compare data over time to identify trends and patterns.** | | **Alternative Flow** | | | | | | 1 | The system prompts the user to provide additional information or filter parameters to refine the report. | | 2 | The user provides the requested information or modifies filter settings. | | 3 | The system updates the report with the refined data. | |  * **Login:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Login** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator accesses the platform's administration panel to manage various functionalities. | | | | **Pre-Conditions** | | The administrator has a registered account and valid login credentials. | | | | **Post-Conditions** | | The administrator is logged in and has access to all administrative features. | | | | **Special Requirements** | | Multi-factor authentication and secure session management. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator opens the Login page. | | 2 | The system displays the Login form. | | 3 | The administrator enters their email address and password. | | 4 | The system validates the credentials. | | 5 | **If credentials are valid** | | 6 | The system logs the administrator into the platform. | | 7 | **If credentials are invalid** | | 8 | The system displays an error message. | | **Alternative Flow** | | | | | | 1 | The system displays an error message and prompts the user to re-enter their credentials. | | 2 | The system informs the user that their account is locked and provides instructions for unlocking it. | | 3 | The system sends a verification code to the administrator's registered phone number or email address for additional security. | |  * **Manage Diseases:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Manage Diseases** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator adds, edits, and updates information about various diseases affecting crops within the platform. | | | | **Pre-Conditions** | | The administrator is logged into the platform. | | | | **Post-Conditions** | | The administrator manages the disease database, ensuring accurate and up-to-date information for users. | | | | **Special Requirements** | | Integration with a dynamic disease database, and support for classification. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator opens the Disease Management section. | | 2 | The system displays a list of existing diseases. | | 3 | The administrator can:  **Add a new disease, Edit an existing disease, Delete a disease:** | | 4 | The system updates the disease database accordingly. | | **Alternative Flow** | | | | | | 1 | The system prompts the administrator to complete all required fields before submitting the new disease information. | | 2 | The system warns the administrator that the disease already exists and suggests potential actions, such as merging information or creating a new entry with different details. | | 3 | The system validates the format and accuracy of the imported data before adding it to the platform. | |  * **Manage Treatments:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Manage Treatments:** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator adds, edits, and updates information about various treatment options for different diseases. | | | | **Pre-Conditions** | | The administrator is logged into the platform. | | | | **Post-Conditions** | | The administrator maintains a comprehensive list of treatment options for users to access and utilize. | | | | **Special Requirements** | | Integration with a treatment database, update mechanism, and metrics for effectiveness. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator opens the Treatment Management section. | | 2 | The system displays a list of existing treatment options. | | 3 | The administrator can:  **Add a new treatment, Edit an existing treatment, Delete a treatment** | | 4 | The system updates the treatment database accordingly. | | **Alternative Flow** | | | | | | 1 | The system requires the administrator to provide clear warnings and disclaimers about potential risks associated with the treatment. | | 2 | The system allows the administrator to edit and update the treatment information to ensure accuracy and currency. | | 3 | The system prompts the administrator to provide more reliable references and evidence to support the treatment information. | |  * **Manage Users:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Manage Users** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator reviews, approves, and manages user accounts within the platform. | | | | **Pre-Conditions** | | The administrator is logged into the platform. | | | | **Post-Conditions** | | The administrator ensures a safe and productive environment for platform users. | | | | **Special Requirements** | | Role-based access control, activity logs, and customizable profiles. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator opens the User Management section. | | 2 | The system displays a list of all registered users. | | 3 | The administrator can:  **Review user profiles, Approve or reject user registrations, Suspend or ban users, Assign user roles and permissions** | | 4 | The system updates user accounts and access accordingly. | | **Alternative Flow** | | | | | | 1 | The system prompts the user to complete all required information before submitting the registration request. | | 2 | The system may request additional documentation or proof of identity to verify the user's information. | | 3 | The system ensures that user privacy is protected and sensitive information is not publicly accessible. | |  * **Manage Localized Alerts:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Manage Localized Alerts** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator creates and sends localized alerts to users based on specific criteria. | | | | **Pre-Conditions** | | The administrator is logged into the platform and has access to relevant data. | | | | **Post-Conditions** | | Users receive timely and relevant notifications about potential threats or important information related to their location. | | | | **Special Requirements** | | Geolocation services, weather API integration, and user preferences. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator opens the Localized Alert Management section. | | 2 | The system displays tools for creating and sending alerts. | | 3 | The administrator can:  **Define alert criteria, Create alert messages, Schedule alert delivery.** | | 4 | The system delivers alerts to relevant users via push notifications, email, or other channels. | | **Alternative Flow** | | | | | | 1 | The system informs the administrator that they need more data to define the alert criteria effectively. | | 2 | The system allows users to control their notification preferences and manage the types of alerts they receive. | | 3 | The system alerts the administrator of any technical issues affecting alert delivery and provides troubleshooting steps. | |  * **Review Reports:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Review Reports** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator analyzes platform usage data and user feedback to gain insights and improve the platform. | | | | **Pre-Conditions** | | The administrator is logged into the platform and has access to reporting dashboards. | | | | **Post-Conditions** | | The administrator identifies trends, patterns, and areas for improvement based on data analysis. | | | | **Special Requirements** | | Comprehensive reporting tools and integration with analytics. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator selects specific reports from a dashboard. | | 2 | The system displays visualizations and data analysis reports on various aspects:  **User activity, Content engagement, Disease management, Chatbot interactions.** | | 3 | The administrator analyzes the data to identify trends, patterns, and areas for improvement. | | 4 | The system logs the administrator's actions and updates the platform accordingly. | | **Alternative Flow** | | | | | | 1 | The administrator investigates further to identify the underlying causes and potential implications. | | 2 | The administrator prioritizes addressing these issues and implementing improvements. | | 3 | The administrator takes immediate action to address the identified risks and protect user data. | |  * **Control Chatbot Functions:**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | | **Control Chatbot Functions** | | | | **Actors** | | Admin | | | | **Summary** | | The administrator manages the chatbot's behavior and capabilities to ensure it provides accurate and helpful responses to users. | | | | **Pre-Conditions** | | The administrator is logged into the platform and has access to chatbot management tools. | | | | **Post-Conditions** | | The administrator optimizes the chatbot's performance and enhances its ability to assist users effectively. | | | | **Special Requirements** | | Customizable chatbot, natural language processing integration, and continuous improvement. | | | | **Basic Flow** | | | | | | **Actor Action** | | | **System Response** | | | 1 | The administrator reviews chatbot logs and user feedback. | | 2 | The system displays details of chatbot interactions, including questions asked, responses provided, and user satisfaction ratings. | | 3 | The administrator analyzes the data to identify areas for improvement. | | 4 | The system updates the chatbot's settings and algorithms based on the administrator's actions. | | **Alternative Flow** | | | | | | 1 | The administrator investigates the issue and resolves it promptly to ensure smooth user experience. | | 2 | The administrator identifies the source of the error and updates the chatbot's knowledge base accordingly. | | 3 | The administrator intervenes and provides the user with the necessary support. | |   **4.3.2 Database Schema Diagram** |
|  |
|  |

|  |
| --- |
| **Chapter 5: Implementation** |
| **5.1 Endeavour (Team + Work + Way of Working)**  **5.1.1 Tomato Care:**  Nurturing Crops with Innovative Farming Solutions  **5.1.2 Team Members**  Rafaqat Ahmad (ID: 24784)  Usman Afaq (ID: 24779)  Muhammad Salman Afaq (ID: 24775)  **5.1.3 Work Breakdown Structure (WBS)**  **5.1.3.1 Phase 1:** Smart Detection System  **Task 1:** Application Development  Subtask: Implementing Real-time Picture Capture  Subtask: Developing Disease Detection Model  Subtask: Integration with Treatment Recommendations  **Task 2:** Model Training and Dataset Management  Subtask: Training Model with Datasets  Subtask: Storing Images in MongoDB Database  **5.1.3.2 Phase 2:** Deployment and Interaction  **Task 3:** Integration and Further Enhancement  **Subtask:** Google Colab Integration for Model Enhancement  **Subtask:** Continuous Model Improvement  **Task 4:** Chatbot Integration  **Subtask:** Developing Farmer-Friendly Chatbot  **Subtask:** Integration with Disease Database for Answers  **5.1.3 Project Schedule**  **5.1.3.1 Milestone 1:** Application Development and Model Training  **Start Date:** [Date]  **End Date:** [Date]  **5.1.3.2 Milestone 2:** Deployment and Optimization  **Start Date:** [Date]  **End Date:** [Date]  **5.1.4 Way of Working**  **5.1.4.1 Agile Methodology**  Overview of Agile Principles Applied  Iterative Development Cycles  Regular Team Reviews and Adjustments  Sprint Durations, Planning Meetings, and Retrospective Sessions  **5.1.4.2 Collaboration Tools**  **Used Tools:** GitHub for Document and Data Sharing, Google Drive for Collaborative Document Editing  **5.1.5 Risk Management**  **5.1.5.1 Risk 1:** Technical Challenges  **Mitigation Plan:** Regular Team Training Sessions  **Contingency Measures:** External Expertise Consultation’  **5.1.5.2 Risk 2:** Resource Constraints  **Mitigation Plan:** Cross-training Team Members  **Contingency Measures:** Resource Reallocation  **Additional Features:**  **Offline Alerts:** Implementing offline alerts for immediate notifications to farmers.  **Stakeholders:** The primary focus is on farmers, ensuring user-friendly features and effective solutions. |
| **5.2 Flow Control/Pseudo codes**  **5.2.1 Sign up Page**  initialize controllers: username, email, phone, password  displaySignUpPage():  show Screen with Logo, Username, Email, Phone, Password fields, and Sign Up button  onSignUpButtonPress():  if validateInputs(username, email, phone, password):  executeSignUp(username, email, phone, password)  else:  displayErrorMessage("Invalid inputs")  validateInputs(username, email, phone, password):  return true if inputs are valid, else false  executeSignUp(username, email, phone, password):  createNewUser locally  sendSignUpRequestToServer  navigateToHomePage  displayErrorMessage(message):  show Error Message  createNewUser(username, email, phone, password):  // Local user creation logic  sendSignUpRequestToServer(username, email, phone, password):  // Send user details to the server  navigateToHomePage():  show Home Page  **5.2.2 Login Page**  # Initialize user input controllers  initialize controllers: username, password  # Display the login page  displayLoginPage():  show Screen with:  - Logo  - Username input field  - Password input field (masked)  - Login button  # Handle Login button press  onLoginButtonPress():  if validateInputs(username, password):  executeLogin(username, password)  else:  displayErrorMessage("Invalid credentials. Please check and try again.")  # Validate user inputs  validateInputs(username, password):  if any input is empty or invalid:  return false  else:  return true  # Execute login logic  executeLogin(username, password):  if authenticateUser(username, password):  navigateToHomePage()  else:  displayErrorMessage("Authentication failed. Please try again.")  # Authenticate user  authenticateUser(username, password):  # Logic to check username and password against stored credentials or server  # Display error message  displayErrorMessage(message):  show Error Message on Screen  # Navigate to the home page after successful login  navigateToHomePage():  show Home Page |
| **5.3 Components, Libraries, Web Services, and Stubs**  **5.3.1 Components**  **5.3.1.1 CNN Model**  Description: Core for disease detection.  Functionality: Implements CNN for image classification.  Implementation: TensorFlow and Python.  **5.3.1.2 Flask Web Application**  Description: Web interface for CNN interaction.  Functionality: Processes image data.  Implementation: Flask in Python.  **5.3.1.3 Flutter Application**  Description: Mobile app for real-time monitoring.  Functionality: Integrates with Flask for interaction.  Implementation: Flutter SDK.  **5.3.2 Libraries**  **5.3.2.1 TensorFlow**  Description: Deep learning library.  Integration: Constructs and trains CNN.  **5.3.2.2 OpenCV**  Description: Image processing library.  Integration: Used for image pre-processing.  **5.3.2.3 Flutter SDK**  Description: Cross-platform app development kit.  Integration: Utilized for Flutter app.  **5.3.3 Web Services**  **5.3.3.1 Flask API**  Description: Interface for CNN-Flutter communication.  Implementation: Created using Flask.  **5.3.3.2 MongoDB Database**  Description: Stores disease data.  Integration: Utilizes MongoDB.  **5.3.4 Stubs**  **5.3.4.1 Testing Stubs**  Description: Simulates external components for testing.  Usage: Validates component functionality.  **5.4 IDE, Tools and Technologies**  **5.4.1 Optimized Technology Stack for Tomato Care Application**  **5.4.2 Integrated Development Environment (IDE):**  Backend: Python (Specifically for Backend Development)  Flutter: Visual Studio Code  Collaborative Model Training: Google Colab  **5.4.2.1 Tools:**  Version Control and Document Sharing: GitHub  Containerization: Not Applicable (Omitted Docker for simplicity)  **5.4.2.2 Technologies:**  Backend: Python (Specifically for Backend Development)  Framework: Flask (Backend Web Framework)  Machine Learning: TensorFlow, OpenCV  Database: MongoDB  Flutter: Dart  **5.5 Best Practices and Coding Standards**  **5.5.1 Software Engineering Practices**  Version Control: Git and GitHub  Continuous Integration and Deployment Practices  **5.5.2 Development Practices and Standards**  Python Coding Standards: Adherence to PEP 8  Flutter: Followed Flutter's Coding Standards  **5.6 Deployment Environment for Tomato Care Application**  **5.6.1 Deployment Environment:**  **5.6.1.1 Operating System:** Not Specified (Omitted AWS EC2  instances for simplicity)  **5.6.1.2 Server Configuration:**  Not Specified (Omitted specific server configuration for simplicity)  **5.6.1.3 Database:**  MongoDB |
|  |
|  |
| **5.8 Summary**  In this chapter, we outlined the composition of our project team, the detailed work breakdown structure for the implementation phase, and the methodologies and tools adopted for efficient collaboration. We addressed potential risks and mitigation strategies, presented flow control or pseudo codes, listed key components and technologies, and detailed the deployment environment. Our team adhered to best practices and coding standards throughout the implementation, ensuring a robust and effective solution. |