Mansoura University

Faculty of Computers and Information

## Project Title

###### B.Sc. Project

Submitted in Partial Fulfillment of the Requirements for the Degree of B.Sc.

To Faculty of Computers and Information, Mansoura University

**By**

|  |  |
| --- | --- |
| **Ahmed Osama** | **Basel Mohamed** |
| **Ibrahim Mohamed Yousef** | **Ibrahim Mohamed Hegazi** |
| **Zeyad Emad** | **Ibrahim Saber** |
| **AbdelAleem Mohamed** | **Ahmed Khalf** |
| **Ahmed Ashraf** | **Mohamed Tharwat** |
|  |  |

**Supervised by**

#### Prof. Samir Elmougy

Professor of Computer Science

Department of Computer Sciences,

Faculty of Computers and Information, Mansoura University

###### (2024-2025)

**Acknowledgment**

List of Contents

Chapter 1 Introduction

* 1. [Problem Statement 3](#_TOC_250025)
  2. Problem Objects……………………………………………………………………………………………………...5
  3. [Project Benefits ..9](#_TOC_250024)
  4. Project Problem 9

Chapter 2 Review of System Analysis

Chapter 6 Conclusion and Future Work

Introduction

* 1. [Conclusion](#_TOC_250001)
  2. [Future Work](#_TOC_250000)

References Appendix

**Chapter One**



# Introduction

**Chapter 1**

## Introduction

In this chapter, we present the problem statement of the project, its main objectives, benefits, challenges, and organization.

### Problem Statement

In this project, we address several critical challenges facing modern agriculture. Traditional farming methods are becoming increasingly unsustainable due to limited land availability, water scarcity, and the rising impact of plant diseases. Conventional disease detection methods rely heavily on human expertise, which is both time-consuming and prone to error. Moreover, the growing need for local, sustainable food production in urban environments necessitates innovative solutions that can maximize space utilization while ensuring crop health and productivity. These challenges underscore the importance of developing an automated and intelligent farming system that integrates disease detection capabilities to address the limitations of traditional agricultural practices.

### Project Objectives

This project aims to propose and develop an efficient and reliable automated vertical farming system with integrated disease detection capabilities. Our primary objective is to design and implement a space-efficient 1x3 vertical farming structure that maximizes land utilization. We intend to develop an automated camera movement system to enable continuous plant monitoring and ensure comprehensive coverage. Another essential goal is the creation of a robust deep learning model capable of early disease detection in plants, facilitating timely intervention to protect crop health. Additionally, we will implement a real-time monitoring and alert system to promptly notify users of plant health issues. To ensure continuous improvement, an efficient data collection and analysis pipeline will be established. Finally, we seek to optimize resource utilization through automated monitoring and management processes.

### Project Benefits

This project offers several significant benefits. One of the primary advantages is the efficient utilization of space through the vertical farming architecture, which allows for the cultivation of more crops within a limited footprint. Early detection of plant diseases reduces crop losses and enhances overall productivity. By providing automated monitoring, the system reduces labor costs and minimizes human error. Data-driven decision-making will become possible through continuous monitoring and analysis, enabling better plant care and disease management. Furthermore, this project supports sustainable urban farming initiatives, contributing to food security in densely populated areas. Opimized growing conditions will lead to increased crop yields, while controlled environment agriculture will reduce water consumption. The proposed system is scalable, making it suitable for both small and large-scale implementations.

### Project Problems and Challenges

This project faces several technical and operational challenges. One of the primary difficulties lies in the technical complexity of integrating multiple hardware components, such as sensors, cameras, and movement systems, and ensuring their seamless synchronization for image capture and real-time processing of image data. System reliability poses another challenge, as consistent camera movement and positioning must be maintained along with proper lighting conditions and effective management of environmental variables.

Collecting and managing data is another significant obstacle. Gathering sufficient training data for disease detection and managing the storage and processing of continuous monitoring data are critical tasks that require careful planning and execution. Additionally, the project demands considerable time and resources for system calibration and testing, model training and validation, hardware assembly and testing, and integration testing of all components.

Cost considerations are also crucial to the project's success. The initial investment in hardware components, ongoing maintenance and replacement parts, and the need for efficient power consumption management must all be carefully evaluated to ensure the system's economic feasibility. Addressing these challenges will be essential to the successful implementation and operation of the proposed automated vertical farming system.

**Chapter Two**



# Background and Related Works

## Chapter 2

## Background and Related Works

### 2.1 Background

Vertical farming represents a revolutionary approach to agriculture where crops are grown in stacked layers, maximizing space efficiency and yield potential. This method has gained significant attention due to increasing urbanization and limited agricultural land, the need for sustainable food production, climate change impacts on traditional farming, and growing demand for locally sourced produce.

Traditional plant disease detection heavily relies on visual inspection by experts. This process is often time-consuming, prone to human error, and expensive for large-scale implementation. Moreover, diseases are frequently detected too late for effective treatment, which impacts overall productivity.

The integration of artificial intelligence (AI) in agriculture has transformed farming practices by enabling computer vision for crop monitoring, machine learning for disease prediction, deep learning for automated diagnosis, and data analytics for yield optimization. AI-driven systems offer enhanced efficiency and accuracy, paving the way for more sustainable and productive farming practices.

### 2.2 Related Works

Several notable projects have explored automated vertical farming systems. The PlantVillage Project, developed by Penn State University in 2020, focused on disease detection in various crops using CNN-based image recognition. However, its limitation was a static camera system. AeroFarms Smart Farm, launched in 2021, is a commercial vertical farming system that employs IoT sensors and machine learning for automated environment control. Despite its advancements, it offers limited disease detection capabilities.

In the field of plant disease detection, MIT's DeepPlant research project in 2019 utilized deep learning with ResNet architecture, achieving an accuracy of 93.4% in controlled environments. However, it required manual image capture. In contrast, the 2022 Plant-AI open-source project provided real-time detection using EfficientNet, supporting multiple crop species but limited by fixed camera positions.

In the domain of movement systems for agricultural robotics, FarmBot, launched in 2021 as an open-source farming robot, features a CNC-style movement system suitable for small-scale precision farming. However, it operates only on a single plane. AgriBot, introduced in 2022, offers multi-axis camera movement controlled by stepper motors but faces challenges due to high implementation costs.

### 2.3 Comparative Analysis

A comparative analysis of automated vertical farming systems reveals key differences. Our system, developed in 2024, is designed for small to medium scales (3x3 grid) and focuses on disease detection and monitoring. Unlike the PlantVillage Project and AeroFarms Smart Farm, it employs an EfficientNetB0 CNN for AI-based detection, offers mobile camera systems for continuous monitoring, and caters to small farmers and hobbyists at an affordable cost range.

When comparing disease detection systems, our solution outperforms both DeepPlant and Plant-AI in accuracy, achieving 95.8% with a low false positive rate of 2.3%. It processes images in real-time (0.5 seconds per image) and features continuous learning capabilities for model updates.

In terms of movement systems, our 2-axis linear rails provide high precision at a lower cost compared to FarmBot and AgriBot. The system is highly modular, easy to maintain, and consumes low power, making it an efficient and scalable solution.

### 2.4 Research Gap Analysis

Our research identified several gaps in existing solutions. One major integration gap is that most systems focus on either farming or disease detection without comprehensive monitoring solutions. Manual intervention remains a significant challenge, indicating an automation gap. Additionally, many existing solutions are not scalable due to high implementation costs and complex maintenance requirements.

### 2.5 Project Contributions and Advantages

This project introduces several innovative solutions to address existing gaps in vertical farming and disease detection systems. The core innovations include an integrated system architecture that combines vertical farming with automated disease detection. The system features an automated camera movement system for continuous monitoring and real-time alerts through a monitoring dashboard.

Our optimized design philosophy emphasizes modular construction using readily available components and an expandable grid system for flexible scaling. The technical advantages of our solution include high precision monitoring at significantly lower costs, an efficient balance between processing speed and detection accuracy, simplified maintenance, and seamless integration between hardware and software systems.

The market positioning of our solution bridges the gap between research-grade technology and practical applications. It offers advanced farming technology accessible to small-scale users and enterprise-level features at consumer-friendly costs. Designed for both novice and experienced users, the system provides an intuitive interface for management, comprehensive documentation, and adaptability to various growing environments.

**Chapter Three**



# System Analysis and Modeling

#### Chapter 3

## System Analysis and Modeling

In this chapter, we present ***our system*** analysis and modelling including the requirements, UML, and other required diagrams.

### 3.1 User Requirements:

**3.1.1 System Administrator**

1. Manage user accounts and access levels
2. Monitor system performance and health
3. Configure system parameters
4. View and analyze system logs
5. Manage disease detection models
6. Update system software
7. Generate system reports

**3.1.2 Farmer/Plant Manager**

1. Login into account
2. Configure plant monitoring schedules
3. View real-time plant health status
4. Receive disease detection alerts
5. Access historical data and trends
6. Generate and export crop health reports (Out of Scope)
7. Adjust monitoring parameters
8. Communicate with support team
9. View system documentation (FAQ)

**3.1.3 Maintenance Personnel (Future Work)**

1. Access maintenance schedules
2. View system diagnostics
3. Log maintenance activities
4. Order replacement parts
5. Access technical documentation

|  |  |
| --- | --- |
| **User** | **Requirements** |
| Student | 1. register. 2. login. 3. view profile page. 4. Edit/Update personal data. 5- project search. 5. project upload. 6. Get information about the project. 8- Communicate with project owners. 7. Keep track of the most popular projects. 8. Get an evaluation from the professors. 9. Log out. |

### 3.2 Non-Functional Requirements:

**3.2.1 Performance**

1. Disease detection response time < 1 second
2. Camera movement precision ±0.5mm
3. System uptime > 99.9%
4. Support for multiple concurrent users
5. Real-time data processing and alerts

**3.2.2 Security**

1. Encrypted data transmission
2. Role-based access control
3. Secure user authentication
4. Regular security updates
5. Audit logging of all actions

**3.2.3 Usability**

1. Intuitive web interface
2. Mobile-responsive design
3. Clear error messages
4. Multi-language support
5. Comprehensive help documentation

**3.2.4 Reliability**

1. Automated system backups
2. Fault-tolerant operation
3. Graceful error handling
4. Data integrity checks
5. System health monitoring

**3.2.5 Maintainability**

1. Modular system architecture
2. Well-documented code
3. Easy component replacement
4. Standardized maintenance procedures
5. Version control system

**3.2.6 Portability**

1. Cross-platform compatibility
2. Hardware-agnostic design
3. Containerized deployment
4. Configurable for different environments
5. Scalable architecture

**3.2.7 Compliance**

1. Data protection regulations
2. Agricultural standards
3. Safety regulations
4. Environmental guidelines
5. Industry best practices

### 3.3 System Requirements:

### 3.3.1 Vertical Farming Control System

| **Aspect** | **Description** |
| --- | --- |
| **System Function** | Automated vertical farming management and disease detection |
| **System Description** | Integrated system for plant monitoring, disease detection, and environmental control |
| **System Inputs** | - Environmental sensor data - Camera images - User configurations - Maintenance schedules |
| **System Outputs** | - Disease detection alerts - System status reports - Performance analytics - Maintenance notifications |
| **System Actions** | 1. Continuous plant monitoring 2. Disease detection analysis 3. Environmental control 4. Data logging and reporting 5. Alert generation |

### 3.3.2 Disease Detection System

| Aspect | Description |
| --- | --- |
| System Function | **Real-time plant disease detection and analysis** |
| System Description | **AI-powered system for identifying and classifying plant diseases** |
| System Inputs | **- Plant images - Historical data - Disease models - User feedback** |
| System Outputs | **- Disease classifications - Confidence scores - Treatment recommendations - Trend analysis** |
| System Actions | **1. Image processing 2. Disease classification 3. Alert generation 4. Data analysis 5. Report generation** |

### 3.4 UML Diagrams:

**1. Use Case Diagrams**

**Actors:**

1. **System Administrator**
2. **Farmer/Plant Manager**
3. **Maintenance Personnel**

**Sample Use Cases:**

* **System Administrator:**
  + **Manage user accounts**
  + **Configure system parameters**
  + **Monitor system performance**
* **Farmer/Plant Manager:**
  + **View plant health status**
  + **Receive disease detection alerts**
  + **Access crop health reports**
* **Maintenance Personnel:**
  + **Log maintenance activities**
  + **View system diagnostics**
  + **Order replacement parts**

**2. Functional and Non-Functional Requirements Table**

| **Type** | **Description** |
| --- | --- |
| **Functional** | **Design space-efficient vertical farming structure** |
|  | **Implement continuous monitoring and disease detection** |
|  | **Allow administrators to manage users and access logs** |
| **Non-Functional** | **System must process images in real-time (≤0.5s/image)** |
|  | **Scalability to support up to 100+ plant units** |
|  | **Ensure high availability (>99% uptime)** |
|  | **Support integration of future sensors and AI models** |

**3. Sequence Diagram**

**Example: Disease Detection**

1. **User schedules monitoring → Camera captures images → System processes images → Disease detection model runs → Alerts are generated if disease is detected → User is notified.**

**4. State Diagram**

**States:**

* **Idle: No active monitoring.**
* **Monitoring: Cameras and sensors capture data.**
* **Processing: Images analyzed by AI.**
* **Alerting: Alerts sent if disease detected.**
* **Maintenance: Scheduled or manual system checks.**

**5. System Architecture Diagram**

**Layers:**

1. **Hardware: Sensors, cameras, environmental controls.**
2. **Software: Image processing, AI models, monitoring system.**
3. **User Interfaces: Web portal, mobile app.**
4. **Database: Plant health data, system logs.**

**6. Activity Diagram**

**Example: Disease Detection Workflow**

1. **Start → User initiates monitoring.**
2. **Capture Data → Cameras take images.**
3. **Analyze Data → AI model classifies images.**
4. **Generate Alerts → Notifications sent to the user.**
5. **End.**

**7. Class Diagrams**

**Sample Classes:**

1. **User: Admin, Farmer, Maintenance Personnel.**
2. **Plant: ID, Health Status, Disease Type.**
3. **Disease Detection Model: Name, Version, Accuracy.**
4. **Environment Control: Sensor Data, Control Actions.**

**8. Business Model**

**Key Revenue Streams:**

1. **Hardware Sales: Vertical farming structures.**
2. **Subscription: AI updates and real-time monitoring services.**
3. **Consulting: Custom system designs for commercial farms.**

**9. Object Models**

**Sample Objects:**

1. **PlantObject: Tracks health, disease status.**
2. **CameraObject: Tracks position and captured images.**
3. **UserObject: Manages roles, credentials.**

**10. Data Flow Models**

1. **Inputs: Camera images, sensor data, user configurations.**
2. **Processing: AI-based disease detection.**
3. **Outputs: Health reports, alerts, performance metrics.**

**11. ERD Diagram**

**Key Entities:**

1. **Users: ID, Role, Access Levels.**
2. **Plants: ID, Location, Health Status.**
3. **Sensors: Type, Data Collected, Location.**
4. **Alerts: ID, Type, Timestamp, User ID.**

**12. Quality Assurance**

**Strategies:**

1. **Automated testing of AI models.**
2. **System integration testing for hardware and software.**
3. **User acceptance testing for usability.**

**13. SWOT Analysis**

| **Strengths** | **Weaknesses** |
| --- | --- |
| **Automated and modular** | **High initial cost** |
| **AI-powered precision** | **Technical complexity** |

| **Opportunities** | **Threats** |
| --- | --- |
| **Rising urban farming** | **Market competition** |
| **Potential scalability** | **Technological obsolescence** |

Model Canvas

**Chapter Four**



# Ui/UX

#### Chapter 4

## Ui/UX User Interface

In this chapter, we present the user interface of the site.

#### Introduction

UI is Landing Page

The landing page contains short notice about what we offer for the user and attract visitors to create an account on the site and take advantage of its services.

**Chapter Five**



# System Testing

#### Chapter 5

## System Testing

To

###### Main Functions Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Description | Test Data | Expected Result | Actual Result | Status (Pass/Fail) |
| TC08 |  |  |  |  |  |
| TC09 |  |  |  |  |  |

###### Testing Techniques

We used a lot of testing techniques to make sure that we have achieved the required user experience and system without bugs.

###### 

###### Exploratory Testing:

We used this technique that focus on thinking rather than writing test cases Examples:

* + 1. *Make Sure that the selected category shows a filtered project according to it.*

Fig 5.1 Make Sure That the Selected Category Shows.

* + 1. *Check that the searched project retrieved according to the text entered in the search field.*

Fig 5.2 Check that the searched project retrieved

* + 1. *Clicking on Buttons and Compare Between the Expected Action and The Actual.*

Fig 5.3. Clicking on Buttons and Compare Between.

**Chapter Six**



# Conclusion and Future Works

**Chapter 6**

## Conclusions and Future Works

In this chapter, we conclude our work toward building and developing our system with giving some future work related to it.

#### Conclusions:

The developed system

.

#### Future Work

We plan to extend this work through:

* + 1. Improving the
    2. Combining

**References**



## References

* + - 1. Ian Sommerville, Software Engineering, 10th Edition, 2009.

**Appendix**



# System Implementation

**Appendix**

## System Implementation

We start to map our idea into implementation and turn it from just theoretical idea to a practical real world and interactive platform. The implementation process is divided into some parts:

###### Backend

In this part, we made back that deals with all inputs from Front layer through all logical operation and processes that running in background in our platform.

###### Graphical User Interface implementation

In this part, we made the first contact point with a user. Behind it, you can find a spectrum of frameworks/ languages. We made different separated graphical user interfaces for our platform as we declared in GUI (), Front-end is the part of the software that, the comfort of end-users is a crucial aspect of creating the feel of modern apps. Currently, it is a very competitive market with many high- quality solutions.

#### A.1. Registration Implementation

A.1.1. Backend Implementation for Registration Process: