****

**COMSATS University Islamabad,**

**Attock Campus, Attock Pakistan**

**Wheat Shield**

***By***

**Kashif Hussain CIIT/FA20-BCS-019/ISB**

**Muhammad Zubair CIIT/FA20-BCS-041/ISB**

***Supervisor*Mr. Waseem Khan**

***Bachelor of Science in Computer Science / Software Engineering (2020-2024)***

****

**COMSATS University Islamabad,**

**Attock Campus, Attock Pakistan**

**Wheat Shield**

**A project presented to**

**COMSATS University, Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelors of Science in Computer Science (2020-2024)***

**By**

**Kashif Hussain CIIT/FA20-BCS-019/ISB**

**Muhammad Zubair CIIT/FA20-BCS-041/ISB**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other, we will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Kashif Hussain Muhammad Zubair

--------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “**Wheat Shield**” was developed by   
**Kashif Hussain (CIIT/FA20-BCS-019)** and **Muhammad Zubair (CIIT/FA20-BCS-041)** under the supervision of “Mr. Waseem Khan” and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use still cameras with a continuous recording facility irrespective of the fact that any event may happen or not. Moreover, they are usually pointing at a specific user defined locations so more than one camera are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable **S**urveillance **C**ontrol **S**ystem **(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Mr. Muhammad Waseem”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are grateful to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us with the values of honesty & hard work.

Kashif Hussain Muhammad Zubair

--------------------------- ---------------------------

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Table of Contents**

[1 Introduction 1](#_Toc165901886)

[1.1 Vision Statement 1](#_Toc165901887)

[1.2 Related System Analysis/Literature Review 1](#_Toc165901888)

[1.3 Project Deliverables 1](#_Toc165901889)

[1.3.1 Disease Detection Module: 2](#_Toc165901890)

[1.3.2 Weather Monitoring Module: 2](#_Toc165901891)

[1.3.3 E-commerce Platform: 2](#_Toc165901892)

[1.3.4 Location-based Services: 2](#_Toc165901893)

[1.3.5 User Authentication and Management: 2](#_Toc165901894)

[1.3.6 Documentation 2](#_Toc165901895)

[1.3.7 Testing and Quality Assurance: 2](#_Toc165901896)

[1.3.8 Deployment Package: 2](#_Toc165901897)

[1.4 System Limitations/Constraints 2](#_Toc165901898)

[1.5 Tools and Technologies 4](#_Toc165901899)

[1.6 Relevance to Course Modules 4](#_Toc165901900)

[1.6.1 Machine Learning and Artificial Intelligence (AI): 4](#_Toc165901901)

[1.6.2 Web Development: 4](#_Toc165901902)

[1.6.3 Database Management Systems (DBMS): 5](#_Toc165901903)

[1.6.4 Software Engineering: 5](#_Toc165901904)

[1.6.5 Data Communication and Computer Networks: 5](#_Toc165901905)

[1.6.6 Data Structures and Algorithms: 5](#_Toc165901906)

[2 Problem Definition 6](#_Toc165901907)

[2.1 Problem Statement 6](#_Toc165901910)

[2.2 Problem Solution 6](#_Toc165901911)

[2.3 Objectives of the Proposed System 7](#_Toc165901912)

[2.4 Scope 8](#_Toc165901913)

[2.5 Modules 9](#_Toc165901914)

[2.5.1 Module 1: Disease Detection 9](#_Toc165901915)

[FE-1: Upload images or provide input data for disease detection. 9](#_Toc165901916)

[FE-2: Display results indicating the detected disease and relevant information. 9](#_Toc165901917)

[2.5.2 Module 2: Weather Monitoring 9](#_Toc165901918)

[FE-1: Display real-time weather updates based on the user's location. 9](#_Toc165901919)

[FE-2: Provide weather forecasts for upcoming days. 9](#_Toc165901920)

[FE-3: Send alerts for weather conditions affecting crop health*.* 9](#_Toc165901921)

[2.5.3 Module 3: Agricultural Commerce 9](#_Toc165901922)

[FE-1: Browse agricultural products available for sale. 9](#_Toc165901923)

[FE-2: Create listings for selling agricultural products. 9](#_Toc165901924)

[2.5.4 Module 4: Location-based Services 9](#_Toc165901925)

[FE-1: Identify nearby medical shops and agricultural supply stores. 9](#_Toc165901926)

[FE-2: Provide directions and contact information for identified locations. 9](#_Toc165901927)

[FE-3: Enable location-based search for specific products or services. 9](#_Toc165901928)

[2.5.5 Module 5: User Management 10](#_Toc165901929)

[FE-1: Register new users and manage user accounts. 10](#_Toc165901930)

[FE-2: Assign roles and permissions to users based on their responsibilities. 10](#_Toc165901931)

[FE-3: Monitor user activity and access logs for security purposes. 10](#_Toc165901932)

[2.5.6 Module 6: Content Management 10](#_Toc165901933)

[FE-1: Manage content related to disease information, weather updates, and agricultural products. 10](#_Toc165901934)

[FE-2: Create and update educational resources for farmers. 10](#_Toc165901935)

[FE-3: Review and moderate user-generated content, such as product listings and reviews. 10](#_Toc165901936)

[2.5.7 Module 7: Analytics and Reporting 10](#_Toc165901937)

[FE-1: Generate reports on disease detection statistics, weather patterns, and sales data. 10](#_Toc165901938)

[FE-2: Analyze trends and patterns to identify areas for improvement and optimization. 10](#_Toc165901939)

[FE-3: Provide insights and recommendations for enhancing the platform's effectiveness. 10](#_Toc165901940)

[2.5.8 Module 8: System Administration 11](#_Toc165901941)

[FE-1: Configure system settings and preferences. 11](#_Toc165901942)

[FE-2: Manage integrations with external APIs and services. 11](#_Toc165901943)

[FE-3: Monitor system performance and troubleshoot technical issues. 11](#_Toc165901944)

[3 Requirement Analysis 12](#_Toc165901945)

[3.1 User classes and characteristics 12](#_Toc165901947)

[3.2 Requirement Identifying Technique 13](#_Toc165901948)

[3.2.1 Use case (use case diagram) 13](#_Toc165901949)

[3.2.2 Detail use case: 14](#_Toc165901950)

[3.2.3 Event- response table : 16](#_Toc165901951)

[3.2.4 Storyboarding: 17](#_Toc165901952)

[3.3 Functional Requirements 19](#_Toc165901953)

[3.3.1 Functional Requirement X 19](#_Toc165901954)

[3.4 Non-Functional Requirements 21](#_Toc165901955)

[3.4.1 Reliability 21](#_Toc165901956)

[3.4.2 Usability 21](#_Toc165901957)

[3.4.3 Performance 22](#_Toc165901958)

[3.4.4 Security 22](#_Toc165901959)

[3.5 External Interface Requirements 22](#_Toc165901960)

[3.5.1 User Interfaces Requirements 22](#_Toc165901961)

[3.5.2 Software interfaces 23](#_Toc165901962)

[3.5.3 Communications interfaces 24](#_Toc165901963)

[4 Design and Architecture 26](#_Toc165901964)

[4.1 Architectural Design 26](#_Toc165901969)

[4.2 Design Models 27](#_Toc165901970)

[4.3 Data Design 28](#_Toc165901971)

[4.3.1 User Data: 28](#_Toc165901972)

[4.4 Human Interface Design 28](#_Toc165901973)

[4.4.1 Screen Objects and Actions 28](#_Toc165901974)

[5 Implementation 31](#_Toc165901975)

[5.1 Algorithm 31](#_Toc165901977)

[5.2 External APIs/SDKs 31](#_Toc165901978)

[5.3 User Interface 32](#_Toc165901979)

[5.3.1 Home Screen 32](#_Toc165901980)

[5.3.2 Login Screen 32](#_Toc165901981)

[5.3.3 Image Upload 33](#_Toc165901982)

[5.3.4 Recommendations 33](#_Toc165901983)

[5.3.5 Weather 34](#_Toc165901984)

[5.3.6 Shop 34](#_Toc165901985)

[6 Testing and Evaluation 35](#_Toc165901986)

[6.1 Unit Testing 35](#_Toc165901989)

[6.2 Functional Testing 35](#_Toc165901990)

[6.3 Rules Testing 37](#_Toc165901991)

[6.4 Integration Testing 38](#_Toc165901992)

[7 Conclusion and Future Work 39](#_Toc165901993)

[7.1 Conclusion 39](#_Toc165901995)

[7.2 Future Work 39](#_Toc165901996)

[8. References 40](#_Toc165901997)

**List of Tables**

[Table 1: System analysis 3](#_Toc165848164)

[Table 2: Tools and Technologies for Proposed Project 5](#_Toc165848165)

[Table 3: user classes 14](#_Toc165848166)

[Table 4: Use Case 16](#_Toc165848167)

[Table 5: Event response 17](#_Toc165848168)

[Table 6: FR-1 20](#_Toc165848169)

[Table 7: FR-2 20](#_Toc165848170)

[Table 8: FR-3 21](#_Toc165848171)

[Table 9: FR-4 21](#_Toc165848172)

[Table 10: FR-5 21](#_Toc165848173)

[Table 11: Details of APIs used in the project 32](#_Toc165848174)

[Table 12: Unit testing 36](#_Toc165848175)

[Table 13: Functional testing 37](#_Toc165848176)

[Table 14: Ruling testing 37](#_Toc165848177)

[Table 15: Integration testing 38](#_Toc165848178)

**List of Figures**

[Figure 1: Use Case 15](#_Toc165848990)

[Figure 2: Activity Diagram 29](#_Toc165848991)

[Figure 3: Sequence Diagram 29](#_Toc165848992)

[Figure 4: Home Screen 33](#_Toc165848993)

[Figure 5: Login Screen 33](#_Toc165848994)

[Figure 6: Uploading Screen 34](#_Toc165848995)

[Figure 7: Recommendation Screen 34](#_Toc165848996)

[Figure 8: Weather Screen 35](#_Toc165848997)

[Figure 9: E-commerce shop 35](#_Toc165848998)

# Introduction

## Vision Statement

For farmers seeking efficient disease detection, weather monitoring, and agricultural commerce solutions, Wheat Shield is a web-based application that provides accurate disease identification, real-time weather updates, and a seamless e-commerce platform. Unlike traditional methods, Wheat Shield empowers farmers with timely insights and resources, enhancing crop management practices and fostering sustainable agricultural development.

## Related System Analysis/Literature Review

Our Wheat Shield project has some related systems that have field in common and here below is complete literature review of those apps :

Table : System analysis

|  |  |  |
| --- | --- | --- |
| **Application Name** | **Weakness** | **Proposed Project Solution** |
| Plantix | No e-commerce functionality | Wheat Shield offers a comprehensive e-commerce platform. |
| AgriApp | Lack of integration with weather data. | integrates real-time weather monitoring to enhance decision-making for farmers. |
| Climate FieldView | Proprietary platform with subscription | Wheat Shield provides a free usage of the platform for detecting and monitoring disease. |

## Project Deliverables

Wheat Shield project provides these deliverables:

### Disease Detection Module:

Algorithm for detecting disease and User interface for uploading images Output displaying data.

### Weather Monitoring Module:

Integration with weather API and User interface for displaying weather.

### E-commerce Platform:

User interface for browsing agricultural products and Functionality for users to create accounts, list products, like products and make purchases.

### Location-based Services:

Integration with mapping APIs to provide location-based services and displaying nearby medical shops and agricultural supply stores.

### User Authentication and Management:

User registration and Role-based access control for administrators and users.

### Documentation

Comprehensive documentation outlining the project's objectives, methodologies, and technical details. User manuals and guides for using each module of the web application.

### Testing and Quality Assurance:

Test cases and scripts for functional, integration, and usability testing and Quality assurance reports ensuring the reliability and stability of the application.

### Deployment Package:

Packaged application ready for deployment on web servers and deployment scripts for automating the deployment process.

## System Limitations/Constraints

There are some limitations and constraints for the proposed project "Wheat Shield":

LI-1: Accuracy of Disease Detection: Despite efforts to be precise, there might still be mistakes in identifying diseases due to differences in picture quality or surroundings.

LI-2: Weather Data Availability: The reliability of weather updates depends on getting good data from outside sources. Problems with internet connection or wrong data from these sources might make the weather module less reliable.

LI-3: Geographic Coverage: The location-based services, such as identifying nearby medical shops and agricultural supply stores, may be limited to certain geographic regions where relevant data is available. Users in areas with limited data coverage may experience reduced functionality in this aspect.

LI-4: E-commerce Security: While efforts will be made to ensure the security of transactions on the e-commerce platform, there may still be risks associated with online payments, such as data breaches or fraudulent activities, which are inherent to conducting business transactions online.

LI-5: Device and Browser Compatibility: Compatibility with different devices (desktops, tablets, smartphones) and web browsers (Chrome, Firefox, Safari, etc.) may vary, leading to potential user experience inconsistencies or limitations in functionality across different platforms.

LI-6: Resource Constraints: Limited resources, such as server capacity, bandwidth, and development time, may impose constraints on the scalability and performance of the web application, particularly during periods of high usage or rapid expansion of user base.Tools and Technologies

Here are all the hardware/software tools and technologies with version number which will be used in implementation of the project.

Table : Tools and Technologies for Proposed Project

|  |  |  |  |
| --- | --- | --- | --- |
| **Tools**  **And**  **Technologies** | **Tools** | **Version** | **Rationale** |
| Visual Studio Code | 2024 | Code editor |
| **Technology** | **Version** | **Rationale** |
| React JS | latest | Front-end Development |
| JavaScript | ECMAScript 6 | Programming language for client-side scripting |
| Redux | latest | State management library for ReactJS applications |
| MongoDB | latest | NoSQL database for storing application data |
| Bootstrap | latest | Back-end Development |
| Python | latest | Back-end Development |

## Relevance to Course Modules

### Machine Learning and Artificial Intelligence (AI):

The disease detection module of "Wheat Shield" utilizes machine learning algorithms to identify septoria and stripe rust in wheat plants. Students who have studied machine learning and AI will find practical application of concepts such as classification algorithms and image recognition.

### Web Development:

"Wheat Shield" is a web application developed using ReactJS for the front-end and Node.js for the back-end. Students who have studied web development will find relevance in concepts such as HTML, CSS, JavaScript, server-side scripting, and client-server communication.

### Database Management Systems (DBMS):

MongoDB is used as the database management system for storing application data in "Wheat Shield." Students who have studied DBMS will understand database design principles, querying techniques, and database integration within web applications.

### Software Engineering:

Concepts of software engineering, such as requirements gathering, system design, testing, and project management, are applied throughout the development lifecycle of "Wheat Shield." Students will recognize the importance of following a structured approach to software development to ensure the project's success.

### Data Communication and Computer Networks:

Understanding of computer networks is essential for integrating location-based services, weather APIs, and payment gateways within "Wheat Shield." Students will appreciate the significance of network protocols, API communication, and security considerations in web application development.

### Data Structures and Algorithms:

The disease detection module of "Wheat Shield" involves processing and analyzing large datasets using algorithms for image recognition and pattern matching. Students will recognize the importance of efficient algorithms and data structures in solving real-world problems.

# Problem Definition



## Problem Statement

Farmers face challenges in timely disease and effect crop management, impacting crop yield and agricultural sustainability. Traditional methods of disease identification are often time-consuming and rely on manual observation, leading to delyed response and potential crop damage. Additionally, farmers may lack access to real-time weather updates and agricultural resources, hindering informed decision-making. "Wheat Shield" aims to address these challenges by providing a comprehensive web application for disease detection, weather monitoring, and agricultural commerce. By leveraging machine learning algorithms, farmers can accurately identify diseases such as septoria and stripe rust in wheat plants, enabling prompt intervention and minimizing crop losses. Integration with weather APIs ensures farmers have access to up-to-date weather forecasts, empowering them to make informed decisions regarding crop management practices. Furthermore, the e-commerce platform facilitates seamless transactions for buying and selling agricultural products, enhancing accessibility and efficiency in the agricultural marketplace. "Wheat Shield" ultimately aims to optimize crop management processes, improve crop health, and contribute to sustainable agricultural practices.

## Problem Solution

"Wheat Shield" addresses the pressing challenges faced by farmers in timely disease detection and effective crop management. Traditional methods reliant on manual observation often lead to delayed responses and potential crop damage, impacting crop yield and sustainability. By leveraging advanced machine learning algorithms, "Wheat Shield" offers accurate identification of wheat diseases like septoria and stripe rust, enabling prompt intervention to minimize losses. Integration with real-time weather APIs provides farmers with crucial insights, empowering informed decision-making in crop management practices. Additionally, the e-commerce platform facilitates seamless transactions for buying and selling agricultural products, enhancing accessibility and efficiency in the agricultural marketplace. Through educational resources and community collaboration features, "Wheat Shield" fosters knowledge exchange and peer support among farmers, further optimizing crop management processes. With robust data security measures and scalability, "Wheat Shield" is poised to revolutionize agricultural practices, contributing to improved crop health and sustainable agricultural development.

## Objectives of the Proposed System

The objectives of the proposed "Wheat Shield" system are:

BO-1: Improve Disease Detection Accuracy: Achieve a minimum accuracy rate of 90% in identifying Septoria and stripe rust in wheat plants to facilitate timely intervention and minimize crop losses.

BO-2: Enhance Crop Management Efficiency: Provide farmers with actionable insights and recommendations based on disease detection results and weather forecasts to optimize crop management strategies and maximize yield.

BO-3: Increase Access to Agricultural Resources: Facilitate convenient access to essential agricultural resources, such as medical shops and agricultural supply stores, by providing location-based services within the application.

BO-4: Foster Agricultural Commerce: Create a vibrant marketplace for buying and selling agricultural products, promoting accessibility and efficiency in agricultural transactions.

BO-5: Ensure User Satisfaction and Engagement: Maintain high levels of user satisfaction and engagement by delivering a user-friendly and reliable application experience, supported by comprehensive documentation and responsive customer support.

BO-6: Ensure Data Security and Compliance: Implement robust security measures to safeguard user data and transactions, ensuring compliance with relevant regulations and standards.

BO-7: Foster Sustainable Agricultural Practices: Encourage adoption of sustainable agricultural practices by providing farmers with tools and insights to optimize resource utilization and minimize environmental impact.

BO-8: Support Scalability and Growth: Design the system architecture to be scalable and adaptable, capable of accommodating future growth in user base and functionality enhancements.

BO-9: Promote Continuous Improvement: Commit to ongoing updates and enhancements based on user feedback and technological advancements, continuously improving the functionality and usability of the "Wheat Shield" system.

## Scope

The scope of the "Wheat Shield" project encompasses a holistic solution aimed at revolutionizing various aspects of crop management for farmers. The main functionalities include disease detection, weather monitoring, agricultural commerce, and location-based services. The disease detection module utilizes advanced machine learning algorithms to accurately identify common wheat diseases such as Septoria and stripe rust, aiding farmers in timely intervention and minimizing crop losses. The weather monitoring module integrates real-time weather data from external APIs, providing farmers with up-to-date forecasts and alerts to optimize crop management decisions in response to changing weather conditions. The agricultural commerce platform serves as a user-friendly marketplace for buying and selling agricultural products, enhancing accessibility and efficiency in agricultural transactions. Location-based services enable farmers to conveniently access nearby resources such as medical shops and agricultural supply stores, facilitating seamless integration into their daily operations. User authentication and management functionalities ensure secure access to the application and personalized user experiences. Comprehensive documentation, compliance with regulations, scalability, reliability, and a commitment to continuous improvement are integral parts of the project scope. By encompassing these functionalities, "Wheat Shield" aims to empower farmers with advanced tools and insights, foster sustainable agricultural practices, and contribute to the efficiency and resilience of agricultural operations.

## Modules

### Module 1: Disease Detection

### FE-1: Upload images or provide input data for disease detection.

### FE-2: Display results indicating the detected disease and relevant information.

### Module 2: Weather Monitoring

### FE-1: Display real-time weather updates based on the user's location.

### FE-2: Provide weather forecasts for upcoming days.

### FE-3: Send alerts for weather conditions affecting crop health*.*

### Module 3: Agricultural Commerce

### FE-1: Browse agricultural products available for sale.

### FE-2: Create listings for selling agricultural products.

### Module 4: Location-based Services

### FE-1: Identify nearby medical shops and agricultural supply stores.

### FE-2: Provide directions and contact information for identified locations.

### FE-3: Enable location-based search for specific products or services.

### Module 5: User Management

### FE-1: Register new users and manage user accounts.

### FE-2: Assign roles and permissions to users based on their responsibilities.

### FE-3: Monitor user activity and access logs for security purposes.

### Module 6: Content Management

### FE-1: Manage content related to disease information, weather updates, and agricultural products.

### FE-2: Create and update educational resources for farmers.

### FE-3: Review and moderate user-generated content, such as product listings and reviews.

### Module 7: Analytics and Reporting

### FE-1: Generate reports on disease detection statistics, weather patterns, and sales data.

### FE-2: Analyze trends and patterns to identify areas for improvement and optimization.

### FE-3: Provide insights and recommendations for enhancing the platform's effectiveness.

### Module 8: System Administration

### FE-1: Configure system settings and preferences.

### FE-2: Manage integrations with external APIs and services.

### FE-3: Monitor system performance and troubleshoot technical issues.

# Requirement Analysis

|  |
| --- |
|  |



## User classes and characteristics

Here are the user and their characteristics mentioned below in form of table.

Table : user classes

|  |  |
| --- | --- |
| **User** | **Description** |
| Farmer | Farmers are the primary users of the "Wheat Shield" system, responsible for managing their wheat crops and utilizing the application for disease detection, weather monitoring, and agricultural commerce. They may vary in experience level, from novice to experienced farmers. |
| Agricultural Expert | Agricultural experts may include agronomists, researchers, or consultants who provide expertise and guidance to farmers. They may use the system to analyze disease patterns, provide recommendations, and offer educational resources to farmers. |
| E-commerce user | E-commerce users are individuals or businesses interested in buying or selling agricultural products through the platform. They may include farmers looking to sell their crops, agricultural suppliers, or consumers seeking agricultural products. |
| System Administrator | System administrators are responsible for managing and maintaining the "Wheat Shield" platform. They oversee user accounts, system configurations, integrations with external services, and ensure the system's security and performance. |

## Requirement Identifying Technique

### Use case (use case diagram)

A visual representation illustrating interactions between system actors and functionalities, depicting various scenarios of user-system interaction

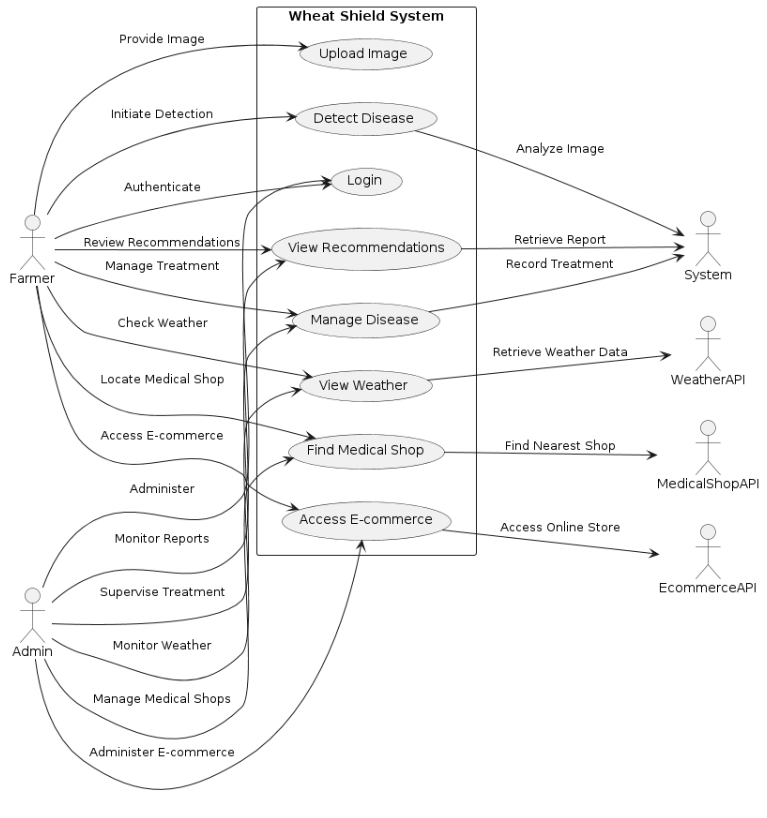


Figure : Use Case

### Detail use case:

An in-depth description of specific interactions between system actors and functionalities, providing a comprehensive understanding of the steps involved in achieving a particular system goal or task.

Table : Use Case

|  |  |
| --- | --- |
| **Use Case ID** | UC-1 |
| **Use Case Name** | Detect and Manage Wheat Diseases |
| **Actors** | Primary Actor: Farmer  Secondary Actor: System |
| **Description** | A Farmer accesses the Wheat Shield system, uploads an image of a wheat plant showing signs of disease, and requests disease detection. The System analyzes the image for disease detection and provides the farmer with a detailed report including the detected disease, its effects on the wheat plant and recommended treatments. |
| **Trigger** | The Farmer indicates the need to detect and manage diseases affecting their wheat crops. |
| **Preconditions:** | PRE-1: The Farmer is logged into the Wheat Shield system.  PRE-2: The Farmer has uploaded a clear image of a wheat plant showing symptoms of disease. |
| **Postconditions** | POST-1: The disease detection report is generated and stored in the Wheat Shield system.  POST-2: The Farmer receives a detailed report including the detected disease, its effects, and recommended treatments. |
| **Normal Flow** | 1. Farmer uploads an image of a wheat plant showing symptoms of disease.  2. System analyzes the uploaded image for disease detection. 3. System generates a disease detection report.  4. System displays the disease detection to the Farmer. |
| **Business Rules** | BR-1: Disease detection accuracy is based on the quality and clarity of the uploaded image.  BR-2: Recommended treatments are based on expert agricultural knowledge and research. |
| **Assumptions** | 1. Farmers have basic knowledge of using the Wheat Shield system.  2. The Wheat Shield system has access to a comprehensive database of wheat diseases and treatments. |

### Event- response table.

A structured representation detailing system responses corresponding to specific events or triggers, facilitating clear comprehension of system behavior under varying conditions.

Table : Event response

|  |  |  |
| --- | --- | --- |
| **Event** | **System State** | **Response** |
| Image uploaded | Idle | Start image processing to detect diseases.  Analyze the uploaded image for signs of septoria and stripe rust.  Generate a disease report based on the analysis results. |
| Disease detected | Image Processing | Display the detected disease to the user.  Provide information on the effects of the disease on wheat plants.  Recommend suitable pesticides or medicines for treatment. |
| Weather information updated | Idle | Retrieve the latest weather information for the current location.  Update the weather display with the new information. |
| Product browsing | Idle | Allow users to browse through the list of available products.  Display product details including name, description, and price.  Provide options to view more details or purchase the product. |
| Product purchase | Idle | Process the purchase transaction.  Update the inventory to reflect the purchased product. |
| Crop selling | Idle | Allow farmers to list their crops for sale.  Provide a platform for buyers to browse and purchase listed crops.  Facilitate the transaction process between sellers and buyers. |

### Storyboarding:



Figure Access web app

#### *Login:*



Figure login

#### *Uploading picture*



Figure Uploading picture

#### *Recommendations Generated:*

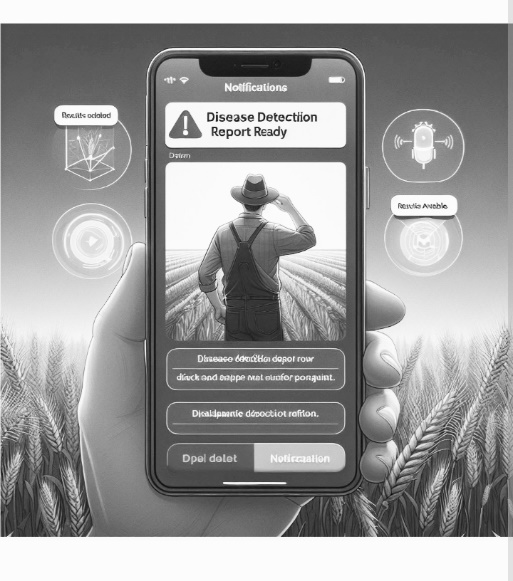


Figure Recommendations

#### *Treatment:*



Figure Treatment

## Functional Requirements

Descriptions of specific functionalities and features that the system must possess to satisfy user needs and achieve project objectives.

### Functional Requirement X

A specific capability or behavior that the system must exhibit to fulfill user needs or meet project objectives, as outlined in the project specifications

Table : FR-1

|  |  |
| --- | --- |
| **Identifier** | FR-1 |
| **Title** | Image Upload |
| **Requirement** | The user shall be able to upload images of wheat plants for disease detection. |
| **Source** | User requirement |
| **Rationale** | Users need a convenient way to input data for disease detection, and image upload is a common and intuitive method. |
| **Priority** | High |

Table : FR-2

|  |  |
| --- | --- |
| **Identifier** | FR-2 |
| **Title** | Input Data Submission |
| **Requirement** | The user shall be able to submit input data (e.g., numerical values or textual descriptions) for disease detection if image upload is not feasible. |
| **Source** | User requirement |
| **Rationale** | Provides flexibility for users who may not have access to image upload capabilities or prefer alternative input methods |
| **Priority** | Medium |

Table : FR-3

|  |  |
| --- | --- |
| **Identifier** | FR-3 |
| **Title** | Disease Identification |
| **Requirement** | The system shall analyze the uploaded images or input data to identify common wheat diseases, including Septoria and stripe rust. |
| **Source** | System requirement |
| **Rationale** | Core functionality of the disease detection module, enabling users to diagnose plant diseases accurately. |
| **Priority** | High |

Table : FR-4

|  |  |
| --- | --- |
| **Identifier** | FR-4 |
| **Title** | Error Handling: Invalid Image Format |
| **Requirement** | If the uploaded image format is invalid (e.g., not supported or corrupted), the system shall display an error message prompting the user to upload a valid image format (e.g., JPEG, PNG). |
| **Source** | System requirement |
| **Rationale** | Ensures user awareness and prompt action in case of invalid inputs, improving user experience. |
| **Priority** | Medium |

Table : FR-5

|  |  |
| --- | --- |
| **Identifier** | FR-5 |
| **Title** | Error Handling: Unavailable Disease Database |
| **Requirement** | If the system's disease database is unavailable or outdated, the system shall notify the user and prompt them to try again later or contact support for assistance. |
| **Source** | System requirement |
| **Rationale** | Provides transparency and guidance to users in case of technical issues with the disease detection functionality. |
| **Priority** | Low |

## Non-Functional Requirements

### Reliability

REL-1: The "Wheat Shield" system shall have a mean time between failures (MTBF) of at least 500 hours, where a failure is defined as any event that disrupts the core functionalities of disease detection, weather monitoring, or agricultural commerce.

REL-2: In the event of a software failure, the system shall automatically generate an error report and notify system administrators via email within 5 minutes of the occurrence, detailing the nature of the failure and any potential impact on system performance.

REL-3: To protect against system failures, the "Wheat Shield" system shall implement regular data backups scheduled to occur daily during off-peak usage hours, with backups stored securely in an off-site location.

REL-4: Error detection mechanisms shall be embedded within critical system components to monitor for abnormal behavior or unexpected events, with alerts triggered for immediate investigation and resolution by system administrators.

### Usability

USE-1: The "Wheat Shield" user interface shall adhere to industry-standard design principles and guidelines, ensuring ease of learning and intuitive navigation for users with varying levels of technical expertise.

USE-2: The system shall provide clear and concise error messages in case of user input errors, guiding users toward corrective actions and facilitating error recovery without undue frustration or confusion.

USE-3: Accessibility features such as keyboard shortcuts and screen reader compatibility shall be implemented to ensure inclusivity and accommodate users with disabilities.

### Performance

PER-1: 95% of webpage requests made to the "Wheat Shield" system shall complete within 3 seconds from the time the user initiates the request, as measured over a typical internet connection with a bandwidth of 10 Mbps or higher.

PER-2: The system shall support concurrent user sessions for up to 1000 users without experiencing degradation in response time or system performance, as verified through load testing simulations conducted quarterly.

### Security

SEC-1: The "Wheat Shield" system shall implement encryption mechanisms using industry-standard protocols to secure data transmission between the client and server components, mitigating the risk of unauthorized interception or tampering.

SEC-2: User authentication and authorization processes shall be enforced through robust security measures, including password hashing, multi-factor authentication, and session management, to prevent unauthorized access to sensitive data and functionalities.

SEC-3: Access to system resources and administrative privileges shall be restricted based on role-based access control (RBAC) principles, with granular permissions assigned to users based on their designated roles and responsibilities within the system.

## External Interface Requirements

### User Interfaces Requirements

Specifications outlining the design elements, layout, and interactive features necessary for intuitive and efficient user interaction with the system

* **GUI Standards and Style Guides:**
  + Implemented industry-standard GUI design principles such as Material Design for web applications or Human Interface Guidelines for mobile apps.
* **Fonts, Icons, and Images:**
  + Used a readable and accessible font such as Roboto and utilized standard icon sets like Material Icons and optimized images for web to ensure fast loading times.
* **Color Schemes:**
  + Choosed color palette with high contrast for readability and accessibility and used color combinations that comply with WCAG guidelines to ensure accessibility for all users, including those with color vision deficiencies.
* **Screen Layout and Resolution Constraints:**
  + Designed layouts using a responsive grid system such as Bootstrap or CSS Grid to ensure compatibility with various screen sizes.
* **Standard Buttons:**
  + Used clear and consistent labels for buttons and ensured that buttons are easily distinguishable and clickable.
* **Message Display Conventions:**
  + Use a consistent style for displaying messages, such as using a colored box for error messages and a neutral color for informational messages.

### Software interfaces

Specifications detailing the communication protocols, data formats, and interactions required for seamless integration between the system and external software components or services.

**SI-1: Disease Detection Algorithm**

* **Connection:** The "Wheat Shield" system shall integrate with the Disease Detection Algorithm version 2.0.
* **Description:** The system shall utilize the Disease Detection Algorithm to analyze images of wheat plants and identify common diseases such as septoria and stripe rust.
* **Interface:** The system shall transmit images or input data to the Disease Detection Algorithm through an API endpoint.
* **Dependency:** The proper functioning of the disease detection module relies on the accuracy and performance of the Disease Detection Algorithm.

**SI-2: Weather API**

* **Connection:** The "Wheat Shield" system shall utilize the Weather API version 3.5 for real-time weather updates.
* **Description:** The system shall retrieve weather data, including temperature, humidity, and precipitation forecasts, based on the user's location.
* **Interface:** The system shall make HTTP requests to the Weather API endpoint to fetch weather information.
* **Dependency:** Timely and accurate weather updates are essential for providing relevant recommendations and alerts to farmers.

**SI-3: Crop Data Database**

* **Connection:** The "Wheat Shield" system shall connect to the Crop Data Database version 1.2.
* **Description:** The system shall retrieve crop-related data, including planting schedules, growth stages, and yield predictions, from the Crop Data Database.
* **Interface:** The system shall establish a database connection using SQL queries to retrieve and store crop data.
* **Dependency:** Access to comprehensive and up-to-date crop data is crucial for providing tailored recommendations and insights to farmers.

**SI-4: Google Maps API**

* **Connection:** The "Wheat Shield" system shall integrate with the Google Maps API version 4.7.
* **Description:** The system shall utilize the Google Maps API for location-based services, including mapping medical shops, agricultural supply stores, and weather stations.
* **Interface:** The system shall make API calls to the Google Maps API to fetch location data and display maps within the application.
* **Dependency:** Accurate mapping and location services enhance the user experience and facilitate navigation for users.

**SI-5: React JS Framework**

* **Connection:** The "Wheat Shield" web application shall be built using the React JS framework version 17.
* **Description:** React JS shall be used for frontend development to create dynamic and interactive user interfaces.
* **Dependency:** The proper functioning of the web application relies on the stability and performance of the React JS framework for rendering UI components and managing state.

### Communications interfaces

**CI-1: Web Browser Interface**

* **Requirement:** The "Wheat Shield" system shall provide a web browser interface for users to access the application.
* **Description:** Users shall interact with the system through a web browser interface, accessing features such as disease detection, weather monitoring, agricultural resources, and e-commerce functionalities.
* **Supported Browsers**: The web browser interface shall be compatible with modern web browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
* **Cross-Platform Compatibility**: The interface shall be responsive and compatible with various operating systems, including Windows, macOS, iOS, and Android.
* **User Authentication**: The system shall implement secure user authentication mechanisms to verify user identities and protect sensitive data.

**CI-2: Network Protocols**

* **Requirement:** The "Wheat Shield" system shall utilize standard network protocols for data transmission between client and server components.
* **Description:** The system shall communicate over the internet using protocols such as Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) to ensure secure and reliable data exchange.
* **Data Encryption:** Communication between client and server components shall be encrypted using Transport Layer Security (TLS) to prevent unauthorized interception or tampering.

# Design and Architecture



## Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system’s modules collaborate with each other in order to achieve the desired functionality.

Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections.

* In initial design stage create Box and Line Diagram for simpler representation of the systems
* After finalizing architecture style/pattern diagram (MVC, Client-Server, Layered, Multi-tiered) create a detailed mapping modules/components to each part of the architecture

To view example of box and line diagram and architecture styles, see Appendix B.

## Design Models

***Design Models for Object Oriented Development Approach***

Architectural blueprints and representations utilized during the development phase, emphasizing object-oriented principles such as encapsulation, inheritance, and polymorphism to design robust and scalable software systems

* + 1. **Activity Diagram**

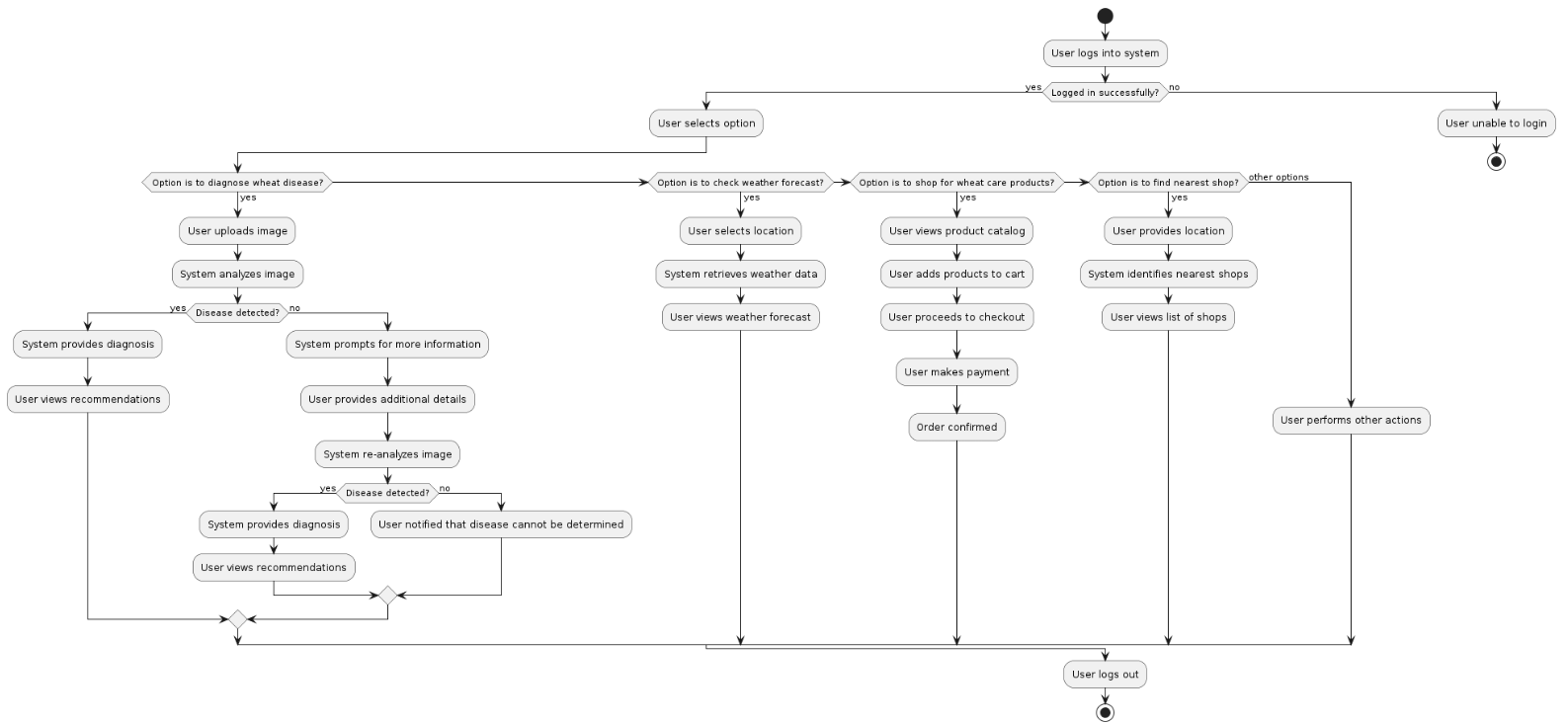


Figure : Activity Diagram

* + 1. **Sequence Diagram**

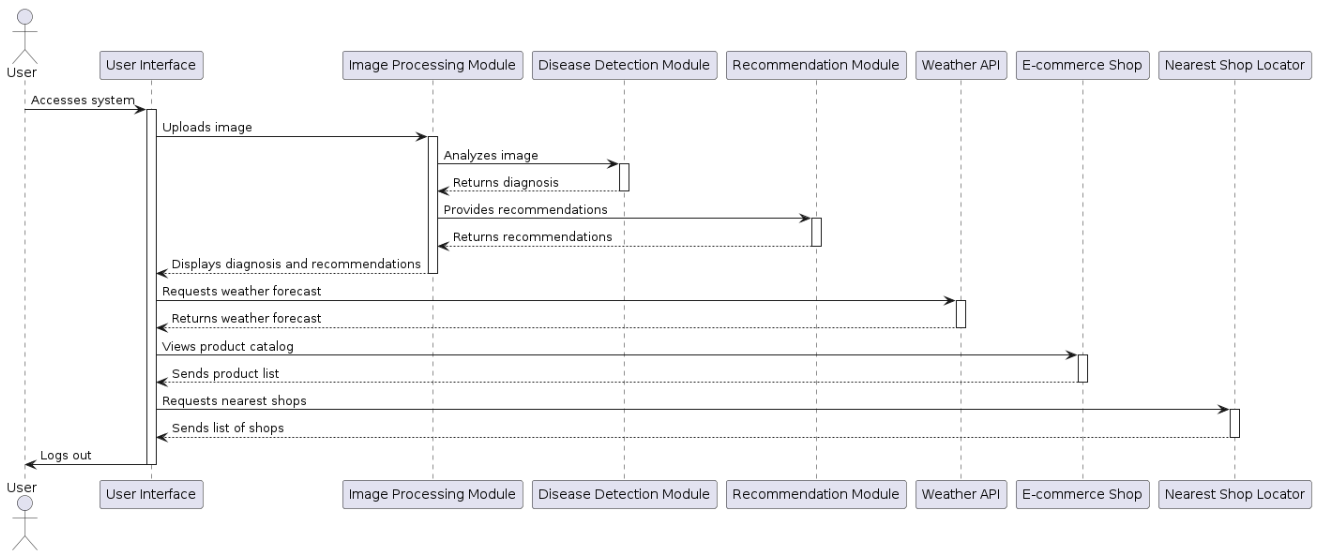


Figure : Sequence Diagram

## Data Design

### User Data:

This includes information about users such as their profiles, preferences, and login credentials. User data may be stored in a database table with fields such as user ID, username, password (hashed for security), email, role, and any additional attributes relevant to the system.

* + 1. **Disease Data:**

Disease data consists of information related to different diseases, their symptoms, causes, and diagnostic criteria. This data may be organized into a database table with fields such as disease ID, disease name, symptoms, causes, treatments, and any related references or links.

* + 1. **Product Data:**

Product data pertains to the items available for purchase on the e-commerce platform. This data may include details such as product ID, name, description, price, availability, and images. Product data can be stored in a database table with appropriate fields to represent each attribute.

* + 1. **Weather Data:**

Weather data comprises information about current weather conditions, forecasts, and historical weather patterns. This data may be obtained from external APIs or services and stored in the system's database or retrieved in real-time when needed.

## Human Interface Design

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

### Screen Objects and Actions

#### Login Screen:

* *Screen Objects*: Username field, Password field, Login button, Forgot password link.
* *Actions*: Users enter their credentials (username and password) and click on the Login button to authenticate. The Forgot password link allows users to reset their password if they forget it.

#### Home Screen:

Screen Objects: Navigation menu, Search bar, service, Monitor, Test, location and shop widget.

Actions: Users can navigate to different sections of the application using the navigation menu. They can search for products using the search bar. The featured products section displays highlighted items. The weather widget provides real-time weather updates.

#### Shop Screen:

Screen Objects: Product image, Product name, Description, Price, Add to Cart button, Quantity selector.

Actions: Users can view detailed information about a product, including its image, description, and price. They can adjust the quantity of the product and add it to their shopping cart using the Add to Cart button.

#### Shopping Cart Screen:

Screen Objects: List of items in the cart, Subtotal, Proceed to Checkout button, Remove item button.

Actions: Users can view the items added to their shopping cart, adjust quantities, and remove items if needed. The Subtotal indicates the total cost of items in the cart. Users can proceed to checkout to complete their purchase.

#### Location Screen:

Screen Objects: input location.

Actions: Users enter the current location so in response to which the application will respond and will give the result based on the input.

#### Disease Detection Screen:

Screen Objects: Upload image button, Result display area, Diagnosis report button.

Actions: Users can upload an image of a plant affected by a disease. The system processes the image and displays the detection result along with recommendations. Users can download or print the diagnosis report.

# Implementation



## Algorithm

## External APIs/SDKs

Application Programming Interfaces (APIs) or Software Development Kits (SDKs) provided by third-party vendors or platforms, allowing developers to integrate external functionalities, services, or data sources into their own software applications.

Table : Details of APIs used in the project

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API and version** | **Description of API** | **Purpose of usage** | **List down the API endpoint/function/class in which it is used** |
| FastAPI | Web framework for building APIs with Python | Handling model prediction requests | /ping, /predict |
| Express.js | Web framework for Node.js | Handling database CRUD operations and user authentication | /search, /like-product, /add-product, /get-products, /signup, /login, etc. |
| TensorFlow | Machine learning library for building and training models | Making predictions using pre-trained model | Model prediction function/class |
| Weather API | Third-party API for fetching weather information | Fetching weather data for weather-based features | Weather API endpoints for retrieving weather information |

## User Interface

### Home Screen

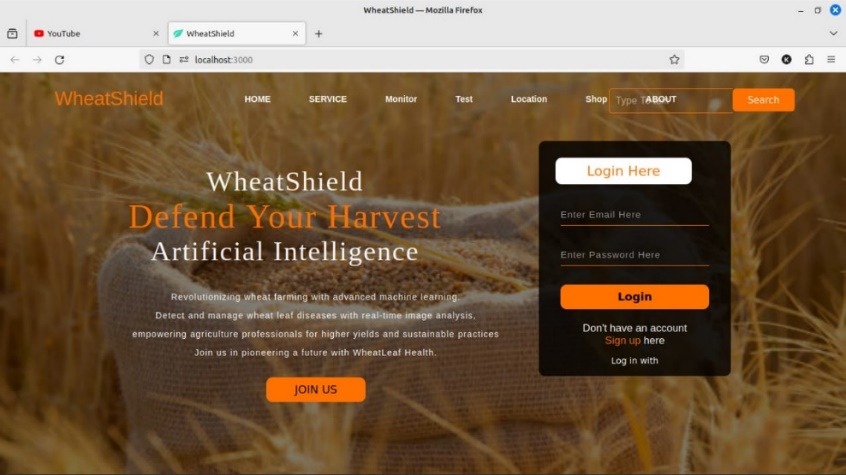


Figure : Home Screen

### Login Screen

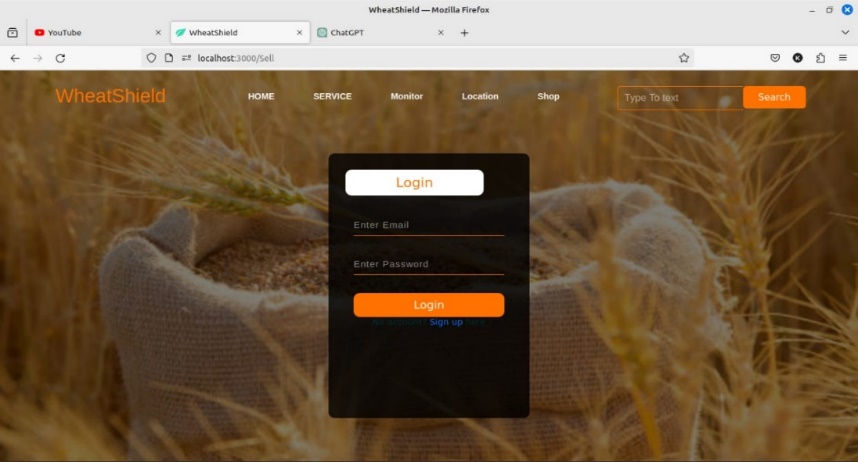
en

Figure : Login Screen

### Image Upload

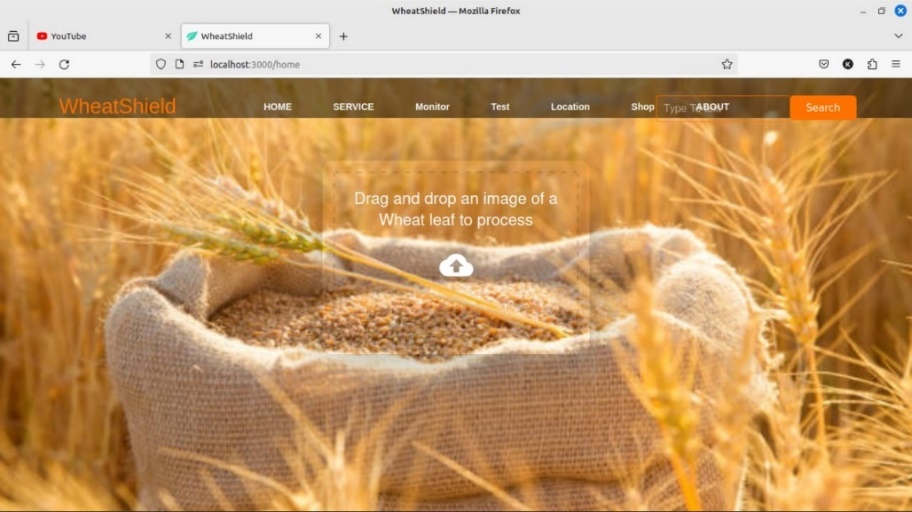


Figure : Uploading Screen

### Recommendations

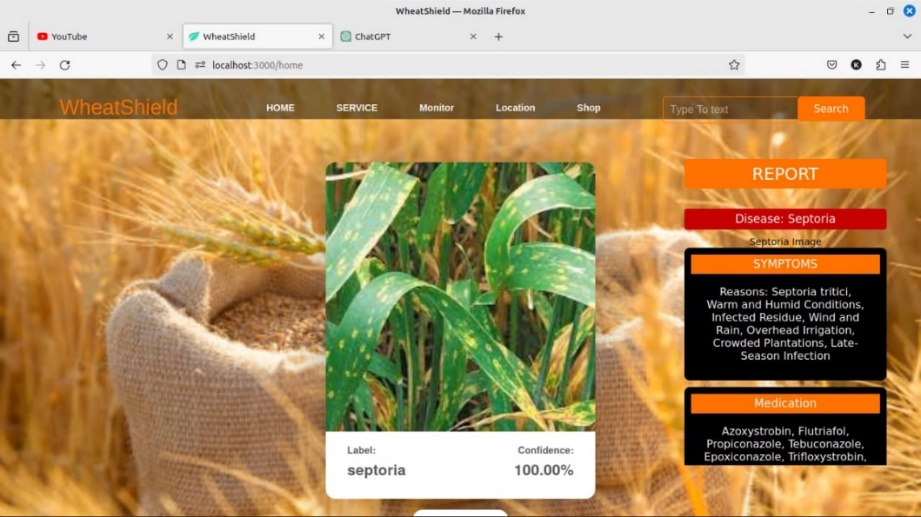


Figure : Recommendation Screen

### Weather



Figure : Weather Screen

### Shop

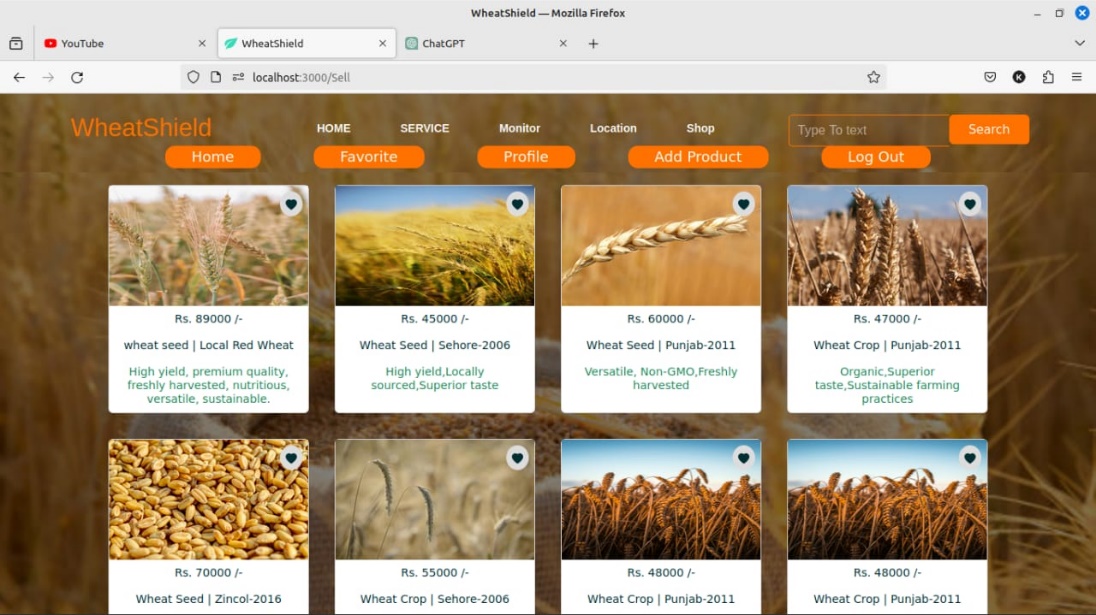


Figure : E-commerce shop

# Testing and Evaluation



## Unit Testing

* **Unit Testing 1:** Login as Patient with valid and invalid credentials
* **Testing Objective:** To ensure the login form is working correctly with valid and invalid credentials/inputs.

Table : Unit testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1 | Check the email field of login to validate that it takes proper email | Email: [muhammadzubair16.k@gmail.com](mailto:muhammadzubair16.k@gmail.com) | Validates email address and moves cursor to next textbox | Pass |
| 2 | Check the email field of login to validate that it displays error message. | Email: [kashifhussain.jk@gmail.com](mailto:kashifhussain.jk@gmail.com) | Highlights field and displays error message | Pass |

## 

## Functional Testing

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the module is tested. This is to ensure that the system produced meets the specifications and requirements.

**Functional Testing 1:** Login with different roles (Management, Patient, Doctor)

**Objective**: To ensure that the correct page with the correct navigation bar is loaded.

Table : Functional testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Login as a ‘Management’ member. | Username : zubair  Password: @1234 | Main page for the Management is loaded with the Management navigation bar. | Logged in and redirected to management main page. | Pass |
| 2. | Login as a ‘Farmer’ member. | Username: Ali  Password: 1234 | Main page for the Farmer is loaded with the Farmer navigation bar. | Logged in and redirected to farmer main page. | Pass |

## 

## Rules Testing

Table : Ruling testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Condition 1: Weather Conditions** | **Condition 2: Soil Moisture Level** | **Condition 3:**  **Plant Symptoms** | **Condition 4: Location** | **Action: Disease Detected** |
| Rainy | High | Yellow spots on leaves | Rural Area | Septoria |
| Sunny | Low | Red spots on leaves | Urban Area | Stripe Rust |
| Overcast | Medium | Black spots on leaves | Suburban Area | Septoria |
| Sunny | High | White powdery spots on leaves | Hillside region | Stripe Rust |
| Rainy | Low | No visible symptoms | Plains region | No disease detected |
| Sunny | High | Wilting of leaves | Remote area | Septoria |
| Overcast | Medium | Stunted growth | Forested area | No disease detected |

## Integration Testing

Table : Integration testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Weather Data Integration | Request Weather Data for Prediction | Successfully retrieved | Successfully retrieves weather data for the specified location and date. | Pass |
| 2. | Weather Data Integration | Generate Disease Prediction Based on Weather | Successfully generated | Successfully generates disease prediction based on the provided weather data. | Pass |
| 3. | Disease Prediction and Notification | Generate Disease Prediction for Notification | Successful retrieval | Successfully generates disease prediction for the specified location and date. | Pass |
| 4. | User Authentication and Data Retrieval | Authenticate User | Success and access granted | Successfully authenticates the user and provides access to data retrieval functions. | User authenticated successfully and granted access to data retrieval module. |

# Conclusion and Future Work



## Conclusion

In conclusion, the development and testing of the Wheat Shield project have been completed successfully. The system has been designed and implemented to provide farmers with valuable insights into wheat disease prediction based on weather data. Through the integration of various modules, including weather data retrieval, disease prediction, user authentication, and notification, Wheat Shield offers a comprehensive solution to help farmers make informed decisions and protect their crops from potential diseases.

The testing phase, including unit testing, functional testing, integration testing, and business rules testing, has ensured the reliability and functionality of the system. Each component has been thoroughly examined to ensure seamless operation and accurate results.

Overall, Wheat Shield represents a significant advancement in agricultural technology, offering farmers a powerful tool to monitor and mitigate the impact of diseases on their wheat crops. With further refinement and optimization, Wheat Shield has the potential to become an indispensable asset for farmers worldwide.

## Future Work

In the future, several enhancements and extensions can be considered to further improve the Wheat Shield system:

Enhanced Prediction Models: Develop more sophisticated disease prediction models using advanced machine learning techniques to improve the accuracy of predictions.

Real-time Data Updates: Implement real-time data updates for weather and disease information to provide farmers with the most up-to-date insights and recommendations.

Mobile Application: Develop a mobile application version of Wheat Shield to provide farmers with convenient access to the system from their smartphones or tablets.

Integration with IoT Devices: Integrate Wheat Shield with IoT devices and sensors installed in the field to collect additional data points for better analysis and prediction.

Multi-language Support: Add support for multiple languages to make Wheat Shield accessible to farmers from different regions and linguistic backgrounds.

# References