**COMSATS University Islamabad,**

**Abbottabad Campus**

**SOFTWARE REQUIREMENTS SPECIFICATION   
(SRS DOCUMENT)**

**for**

**PLANT DISEASE DIAGNOSE SYSTEM**  
Version 1.0

***By***

**ASAD KHAN FA21-BSE-081**

**FURQAN AHMAD FA21-BSE-089**

**ABDUL MUHAMIN FA21-BSE-061**

***Supervisor*Sir Javed Raza**

***Bachelor of Science in Computer Science (20xx-20xx)***

**Table of Contents**

Contents

[**Introduction** 5](#_Toc195360100)

[**Purpose** 5](#_Toc195360101)

[**Scope** 5](#_Toc195360102)

[**Overall description** 5](#_Toc195360103)

[**Product perspective** 5](#_Toc195360104)

[**Operating environment** 5](#_Toc195360105)

[**Design and implementation constraints** 6](#_Toc195360106)

[**Requirement identifying technique** 6](#_Toc195360107)

[**Use case diagram** 7](#_Toc195360108)

[**Use case description** 7](#_Toc195360109)

[**Functional Requirements** 11](#_Toc195360110)

[**Feature 1: User Authentication** 11](#_Toc195360111)

[**Feature 2: Plant Image Upload** 12](#_Toc195360112)

[**Feature 3: Disease Detection and Diagnosis** 12](#_Toc195360113)

[**Feature 4: Weather Data Integration** 12](#_Toc195360114)

[**Feature 5: Historical Data Management** 12](#_Toc195360115)

[**Feature 6: Notifications and Alerts** 12](#_Toc195360116)

[**Non-Functional Requirements** 12](#_Toc195360117)

[**Usability:** 12](#_Toc195360118)

[**Performance:** 13](#_Toc195360119)

[**Reliability:** 13](#_Toc195360120)

[**Security:** 13](#_Toc195360121)

[**Quality Attributes** 13](#_Toc195360122)

[**Usability** 13](#_Toc195360123)

[**Performance** 13](#_Toc195360124)

[**Reliability** 13](#_Toc195360125)

[**Security** 13](#_Toc195360126)

[**References** 13](#_Toc195360127)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**<Supervisor’s Name>**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

The Plant Disease Prediction System is designed to assist farmers by providing early detection of plant diseases through image analysis. With the integration of machine learning, this system offers accurate disease diagnosis based on images of plant leaves or other parts. The goal is to reduce crop losses by offering timely intervention suggestions. This document outlines the scope and functionality of the system, which includes features like image upload, disease prediction, treatment recommendations, and plant health monitoring. The system will be user-friendly, accessible to a global audience, and support various languages.

**Purpose**

The purpose of this Software Requirements Specification (SRS) is to define the requirements and specifications for the Plant Diagnose System. This system utilizes machine learning to predict plant diseases based on user-uploaded images, integrated with weather data for precise diagnoses, and provides treatment recommendations and preventive measures.

**Scope**

The scope of this project includes developing an mobile application that allows users to upload plant images for disease detection. The system will predict diseases based on the uploaded images, offer treatment recommendations, and suggest preventive measures. The system will also provide historical data tracking, real-time alerts, and a community forum for user interaction. It will be accessible in multiple languages, ensuring global applicability. The weather and environmental data integration will enhance prediction accuracy.

**Overall description**

**Product perspective**

The Plant Diagnose System is a standalone application that combines computer vision and weather data integration to provide accurate plant disease diagnoses. It is a new product, distinct from traditional manual diagnosis methods, aiming to automate and simplify the process using advanced machine learning models.

**Operating environment**

* **Hardware Requirements**: Desktop or laptop with a minimum of 8GB RAM, 2.5GHz processor.
* **Operating Systems**: Windows 10, macOS 10.15 or higher.
* **Software Requirements**: Python 3.12, TensorFlow 2.x, OpenCV.
* **Network Requirements**: Internet connectivity for accessing weather data API and cloud-based model updates.

**Design and implementation constraints**

* system must integrate with a weather data API for real-time analysis.
* Machine learning models must be pre-trained and optimized for local processing to reduce latency.
* Use of Python and TensorFlow as the primary development tools.

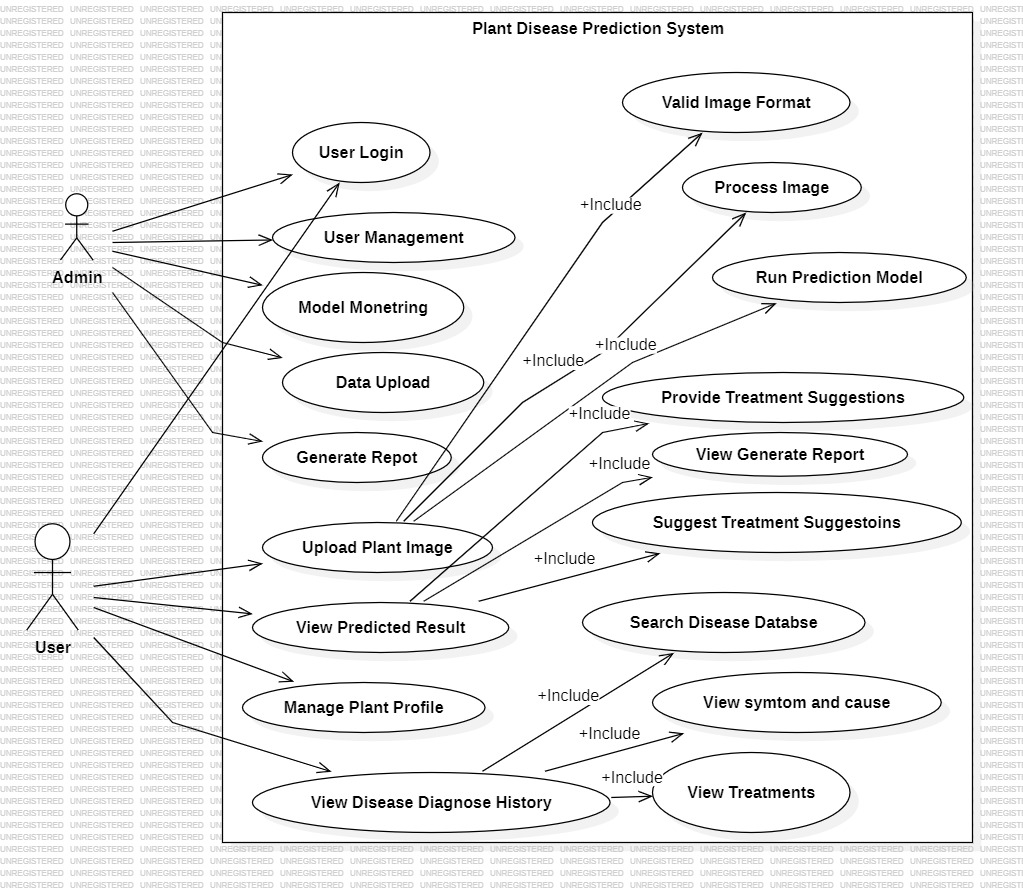
**Requirement identifying technique**

This section describes the requirements identifying technique(s) which further help to derive functional requirements specification. The selection of the technique(s) will depend on the type of project. For instance,

* **Use case** is an effective technique for interactive end-user applications
* **Event- response tables** is for real time system and
* **Story boarding** for graphically intensive applications.

In addition to the above, the projects involving data warehouses, batch processes, hardware devices with embedded control software, and computationally intensive applications required to follow other suitable techniques. Such techniques are described further in Chapter 12, “A picture is worth 1024 words.” For documenting this section let consider identifying requirements through use case as an example.

**Use case diagram**

****

**Use case description**

|  |  |
| --- | --- |
| Use Case ID | UC-1 |
| Use Case Name | User Login / Signup |
| Actors | Admin, Farmer |
| Description | Enables users to register or log in to access system features. |
| Trigger | User selects "Login" or "Sign up" option. |
| Preconditions | User must have a valid internet connection. |
| Postconditions | User is authenticated and redirected to the dashboard. |
| Normal Flow | 1. User enters login credentials (email and password). 2. System validate credential. 3. User is logged in successfully or redirected to the signup form for registration. |
| Alternative Flow | If the credentials are invalid, the system prompts the user to retry or reset their password. |
| Exceptions | Database connection failure or invalid credentials |

|  |  |
| --- | --- |
| Use Case ID | UC-2 |
| Use Case Name | User Management |
| Actors | Admin |
| Description | Enables the admin to manage user accounts, including adding, editing, and deleting users. |
| Trigger | Admin selects the "User Management" option. |
| Preconditions | Admin must have logged into the system. |
| Postconditions | User accounts are updated based on the admin's actions. |
| Normal Flow | 1. Admin selects a user account.  2. Admin performs the desired action (add, edit, delete).  3. System updates the user account. |
| Alternative Flow | If user details are invalid, the system displays an error message. |
| Exceptions | Database connection |

|  |  |
| --- | --- |
| Use Case ID | UC-3 |
| Use Case Name | Model Monitoring |
| Actors | Admin |
| Description | Allows the admin to monitor the performance of the disease prediction model. |
| Trigger | Admin selects the "Model Monitoring" option. |
| Preconditions | The prediction model must be operational. |
| Postconditions | Admin views performance metrics such as accuracy, precision, and efficiency. |
| Normal Flow | 1. Admin accesses the monitoring dashboard.  2. System retrieves and displays model metrics. |
| Exceptions | Model monitoring metrics unavailable due to |

|  |  |
| --- | --- |
| Use Case ID | UC-4 |
| Use Case Name | Data Upload |
| Actors | Admin |
| Description | Provides functionality for the admin to upload new datasets for model training or updates. |
| Trigger | Admin selects the "Data Upload" option. |
| Preconditions | Admin must be logged into the system. |
| Postconditions | The dataset is successfully uploaded and ready for processing. |
| Normal Flow | 1. Admin uploads a dataset file.  2. System validates the file format and content.  3. System stores the dataset. |
| Alternative Flow | If the file format is invalid, the system displays an error message. |
| Exceptions | File upload fails due to connectivity or storage issues |

|  |  |
| --- | --- |
| Use Case ID | UC-5 |
| Use Case Name | Generate Report |
| Actors | Admin, User |
| Description | Allows the admin to generate detailed reports of prediction results. |
| Trigger | Admin selects the "Generate Report" option. |
| Preconditions | Admin must have performed a prediction task. |
| Postconditions | Report is generated and made available for download or viewing. |
| Normal Flow | 1. User selects report type.  2. System compiles report data.  3. System generates the report. |
| Alternative Flow | If no data is available for the report, the system displays a message. |
| Exceptions | Report generation fails due to data retrieval errors |

|  |  |
| --- | --- |
| Use Case ID | UC-6 |
| Use Case Name | Upload Plant Image |
| Actors | User |
| Description | Allows users to upload an image of a plant for diagnosis. |
| Trigger | User clicks the "Upload Image" button. |
| Preconditions | 1. User is logged in. 2. Image file is available on the user's device. |
| Postconditions | Image is processed and ready for analysis. |
| Normal Flow | 1. User selects an image file. 2. System checks for valid image format. 3. Image is uploaded successfully. |
| Alternative Flow | If the image format is invalid, the system prompts the user to upload a supported file type. |
| Exceptions | Image upload fails due to network issues |

|  |  |
| --- | --- |
| Use Case ID | UC-7 |
| Use Case Name | View Prediction Results |
| Actors | User |
| Description | Displays the health diagnosis and prediction results for the uploaded image. |
| Trigger | User completes image upload and waits for analysis results. |
| Preconditions | 1. User has uploaded a valid image. 2. System has successfully processed the image. |
| Postconditions | Results and recommendations are displayed on the dashboard. |
| Normal Flow | 1. System processes the image. 2. Prediction results are displayed, including potential disease, cause, and treatment recommendations. |
| Alternative Flow | If the model cannot make a prediction, the system provides troubleshooting suggestions (e.g., upload a clearer image). |
| Exceptions | Processing fails due to incomplete data or system error. |

|  |  |
| --- | --- |
| Use Case ID | UC-8 |
| Use Case Name | Manage Plant Profiles |
| Actors | User |
| Description | Users can create, edit, or delete plant profiles for monitoring multiple plants. |
| Trigger | User accesses the plant profile section. |
| Preconditions | User must be logged in. |
| Postconditions | Changes to the plant profiles are saved successfully. |
| Normal Flow | 1. User views the plant profiles dashboard. 2. User selects an option to add, edit, or delete a profile. 3. Changes are saved in the database. |
| Alternative Flow | If the user cancels, no changes are made. |
| Exceptions | Changes fail to save due to network issues |

|  |  |
| --- | --- |
| Use Case ID | UC-9 |
| Use Case Name | View Disease Diagnose History |
| Actors | User |
| Description | Allows users to view information about plant diseases history. |
| Trigger | User selects "My plants" from the bottom tab. |
| Preconditions | User is logged in and has a stable internet connection. |
| Postconditions | Disease-related information is displayed. |
| Normal Flow | 1. System retrieves and displays matching records, including symptoms, causes, and treatments. |
| Alternative Flow | If no records are found, the system suggests refining the search criteria. |
| Exceptions | Database connection failure or search timeout. |

**Functional Requirements**

**Feature 1: User Authentication**

* The system shall allow users to register by providing an email address, username, and password.
* The system shall validate user credentials during login.
* The system shall provide a password recovery option using a registered email.

**Feature 2: Plant Image Upload**

* The system shall allow users to upload an image file of a plant.
* The system shall validate the image file format (e.g., JPEG, PNG).
* The system shall notify the user if the upload fails due to an unsupported format or network issue.

**Feature 3: Disease Detection and Diagnosis**

* The system shall analyze the uploaded plant image using a trained machine learning model.
* The system shall display diagnosis results, including the detected disease, causes, and severity level.
* The system shall recommend treatments and preventive measures based on the diagnosis results.

**Feature 4: Weather Data Integration**

* The system shall fetch current weather data from an external API.
* The system shall combine weather data with the diagnosis results to improve accuracy.
* The system shall notify users if weather data retrieval fails.

**Feature 5: Historical Data Management**

* The system shall maintain a record of past plant diagnoses for each user.
* The system shall allow users to view, sort, and filter historical data based on date or plant type.
* The system shall enable users to delete specific records from their history.

**Feature 6: Notifications and Alerts**

* The system shall send real-time notifications to users about critical plant health updates.
* The system shall alert users about upcoming weather conditions that may affect plant health.
* The system shall allow users to enable or disable notifications in their profile settings

**Non-Functional Requirements**

**Usability:**

* The system should have an intuitive interface to accommodate non-technical users, such as farmers.

**Performance:**

* The system must process images and display predictions within 5 seconds.
* It should handle up to 1,000 concurrent users without any latency.

**Reliability:**

* The system must maintain 99% uptime and ensure uninterrupted access.

**Security:**

* All user data, including uploaded images, must be encrypted.
* Role-based access control should be implemented to restrict admin functionalities​

**Quality Attributes**

**Usability**

* The system shall have an intuitive interface to cater to non-technical users such as farmers.

**Performance**

* The system shall process images and display predictions within 5 seconds.
* Handle up to 1000 concurrent users without latency.

**Reliability**

* The system shall maintain 99% uptime and ensure uninterrupted access.

**Security**

* All user data, including uploaded images, must be encrypted.
* Role-based access control to restrict admin functionalities.

**References**

1. TensorFlow Documentation : <https://www.tensorflow.org/learn>
2. OpenCV Documentation: <https://opencv.org>
3. Weather API Documentation: <https://openweathermap.org/api>