Crop Guidance and Farmers Friend

Crop Advisory

Software Requirements Specification

INT 221
MVC Programming

Student Names
Raghav Gupta
Aryaman Agrawal
Student Registration Numbers.
12215792
12216481

LOVELY
PROFESSIONAL
UNIVERSITY

Prepared for
Continuous Assessment 3
Spring 2025

Table of Contents

1. Introduction

Crop Advisory and Farmer's Companion: Precision Guidance for Sustainable Agriculture
Crop Guidance and Farmers Friend.
Problem Title: Helping the farmers in terms of Crop suggestion, precautions based on the met department forecast of rain fall / weather, potential pest attacks, weather warnings etc.

1.1 Purpose

The purpose of this project is to provide a web-based solution for farmers and agricultural stakeholders to:

- *Get crop suggestions based on location and season.*
- *Receive pest alerts for specific regions.*
- *Access weather forecasts for agricultural planning.*
- *Allow administrators to manage pests and alerts through a dedicated admin panel.*

This system aims to improve agricultural productivity and decision-making by providing accurate and timely information.

1.2 Scope

The system includes the following features:

- ***Dynamic Crop Suggestions**: Users can select a country, state, district, and season to get crop recommendations.*
- ***Pest Alerts**: Displays pest alerts based on the user's location.*
- ***Weather Forecasts**: Provides weather data for planning agricultural activities.*
- ***Admin Panel**: Allows administrators to manage pests and alerts.*

The system is designed for farmers, agricultural stakeholders, and administrators. It is accessible via a web browser and optimized for both desktop and mobile devices.

1.3 Definitions, Acronyms, and Abbreviations

- ***CRUD**: Create, Read, Update, Delete operations.*
- ***UI**: User Interface.*
- ***DFD**: Data Flow Diagram.*
- ***AJAX**: Asynchronous JavaScript and XML, used for dynamic data fetching.*
- ***CSRF**: Cross-Site Request Forgery, a security feature.*

1.4 References

- *Laravel Documentation: https://laravel.com/docs*
- *Tailwind CSS Documentation: https://tailwindcss.com/docs*
- *MySQL Documentation: https://dev.mysql.com/doc/*

1.5 Overview

This document outlines the requirements, constraints, and design of the Crop Advisory System. It includes:

- *General description of the system.*
- *Functional and non-functional requirements.*
- *External interface requirements.*
- *Analysis models such as Data Flow Diagrams (DFDs).*
- *Deployment and client-related details.*

2. General Description

2.1 Product Perspective

The Crop Advisory System is a web-based application designed to assist farmers and agricultural stakeholders. It integrates with a MySQL database and uses Laravel for backend operations. The system provides a user-friendly interface for accessing crop suggestions, pest alerts, and weather forecasts.

2.2 Product Functions

The system provides the following functionalities:

- ***Dynamic Dropdowns**: Users can select a country, state, district, and season to get crop suggestions.*
- ***Pest Alerts**: Displays pest alerts based on the user's location.*
- ***Weather Forecasts**: Provides weather data for planning agricultural activities.*
- ***Admin Panel**: Allows administrators to manage pests and alerts.*

2.3 User Characteristics

- ***Farmers**: Basic computer literacy and familiarity with dropdown-based interfaces.*
- ***Admins**: Familiarity with CRUD operations and basic database management.*

2.4 General Constraints

- *The system requires an active internet connection.*
- *It is optimized for modern browsers (e.g., Chrome, Firefox, Edge).*
- *The backend is built using Laravel, and the database is MySQL.*

2.5 Assumptions and Dependencies

- *The crop_rules table in the database contains accurate and up-to-date data.*
- *The system depends on Laravel and MySQL for backend operations.*
- *Users have access to a device with a modern web browser.*

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- **Frontend**: *Built using Tailwind CSS for responsiveness and modern design.*
- **Dynamic Dropdowns**: *Cascading dropdowns for country, state, district, and season.*
- **Admin Panel**: *Provides CRUD operations for managing pests and alerts.*

3.1.2 Hardware Interfaces

- *No specific hardware requirements; the system runs on any device with a modern browser.*

3.1.3 Software Interfaces

- **Backend**: *Laravel framework.*
- **Database**: *MySQL for storing crop rules, pests, and alerts.*

3.1.4 Communications Interfaces

- *AJAX is used for dynamic data fetching between the frontend and backend.*

3.2 Functional Requirements

3.2.1 CROP SUGGESTIONS

3.2.1.1 Introduction

This feature allows users to get crop suggestions based on their selected country, state, district, and season. The system dynamically fetches data from the crop_rules table to populate dropdowns and provide recommendations.

3.2.1.2 Inputs

- *Country: Selected by the user from the first dropdown.*
- *State: Selected by the user from the second dropdown, based on the selected country.*
- *District: Selected by the user from the third dropdown, based on the selected state.*
- *Season: Selected by the user from the fourth dropdown, based on the selected district.*

3.2.1.3 Processing

1. *The system fetches unique states from the crop_rules table based on the selected country.*
2. *The system fetches unique districts based on the selected state.*
3. *The system fetches unique seasons based on the selected district.*
4. *The system fetches crop recommendations based on the selected country, state, district, and season.*

3.2.1.4 Outputs

- *A list of recommended crops is displayed to the user.*

3.2.1.5 Error Handling

- *If no data is available for the selected filters, the system displays a message: "No crops found for the selected filters."*
- *If the user does not select a required field, the system disables the subsequent dropdowns and the "Get Suggestions" button.*

3.2.2 PEST ALERTS

3.2.2.1 Introduction

This feature provides pest alerts for specific regions based on the user's location. The data is fetched from the pests table.

3.2.2.2 Inputs

- *Region: The user's location (e.g., country, state, or district).*

3.2.2.3 Processing

1. *The system queries the pests table to fetch pest alerts for the specified region.*
2. *The data is filtered to include only active pest alerts.*

3.2.2.4 Outputs

- *A list of pest alerts is displayed to the user.*

3.2.2.5 Error Handling

- *If no pest alerts are available for the selected region, the system displays a message: "No pest alerts found for your region."*

3.2.3 WEATHER FORECAST

3.2.3.1 Introduction

This feature provides weather forecasts for specific regions to help users plan agricultural activities.

3.2.3.2 Inputs

- *Region: The user's location (e.g., country, state, or district).*

3.2.3.3 Processing

1. *The system fetches weather data from an external API or database.*
2. *The data is filtered to include only the relevant region and time period.*

3.2.3.4 Outputs

- *A weather forecast is displayed, including temperature, precipitation, and other relevant details.*

3.2.3.5 Error Handling

- *If the weather data cannot be fetched, the system displays a message: "Unable to fetch weather data. Please try again later."*

3.2.4 ADMIN PANEL

3.2.4.1 Introduction

The admin panel allows administrators to manage pests and alerts through CRUD operations.

3.2.4.2 Inputs

- *Pest Data: Includes pest name, region, and description.*
- *Alert Data: Includes alert title, description, and region.*

3.2.4.3 Processing

1. *The system allows admins to create, read, update, and delete pest and alert records.*
2. *Data is validated before being saved to the database.*

3.2.4.4 Outputs

- *Updated pest and alert records are displayed in the admin panel.*

3.2.4.5 Error Handling

- *If validation fails, the system displays error messages (e.g., "Pest name is required").*
- *If a database operation fails, the system logs the error and displays a message: "Unable to save changes. Please try again."*

3.5 Non-Functional Requirements

3.5.1 Performance

- *Dropdowns should populate within 2 seconds.*

3.5.2 Reliability

- *The system should have 99.9% uptime.*

3.5.3 Availability

- *Accessible 24/7.*

3.5.4 Security

- *CSRF protection for all forms.*
- *Secure database queries to prevent SQL injection*.

3.5.5 Maintainability

- *Modular code structure using Laravel.*

3.5.6 Portability

- *Compatible with all modern browsers.*

3.7 Design Constraints

- *The system must use Laravel and MySQL.*
- *The frontend must use Tailwind CSS for styling.*

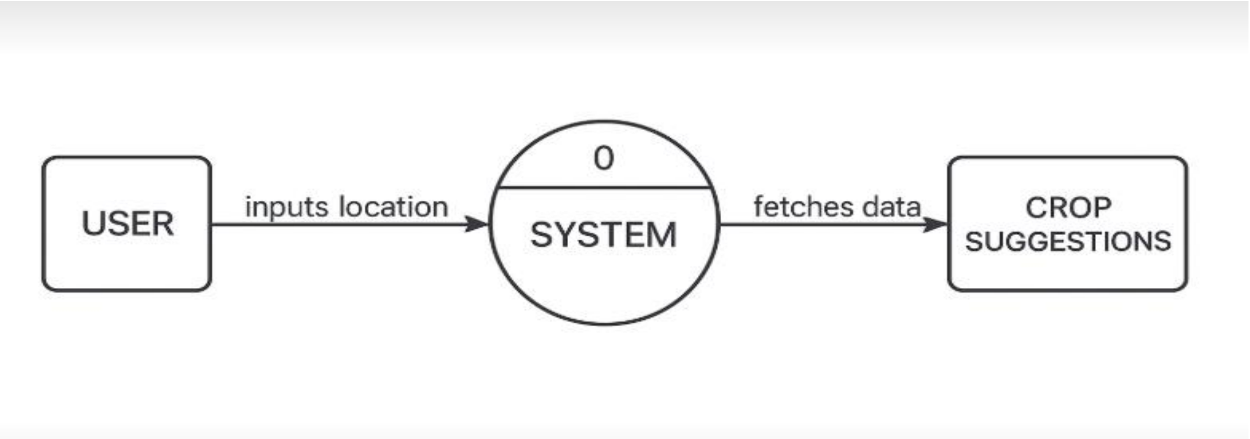
3.9 Other Requirements

- *None at this time.*

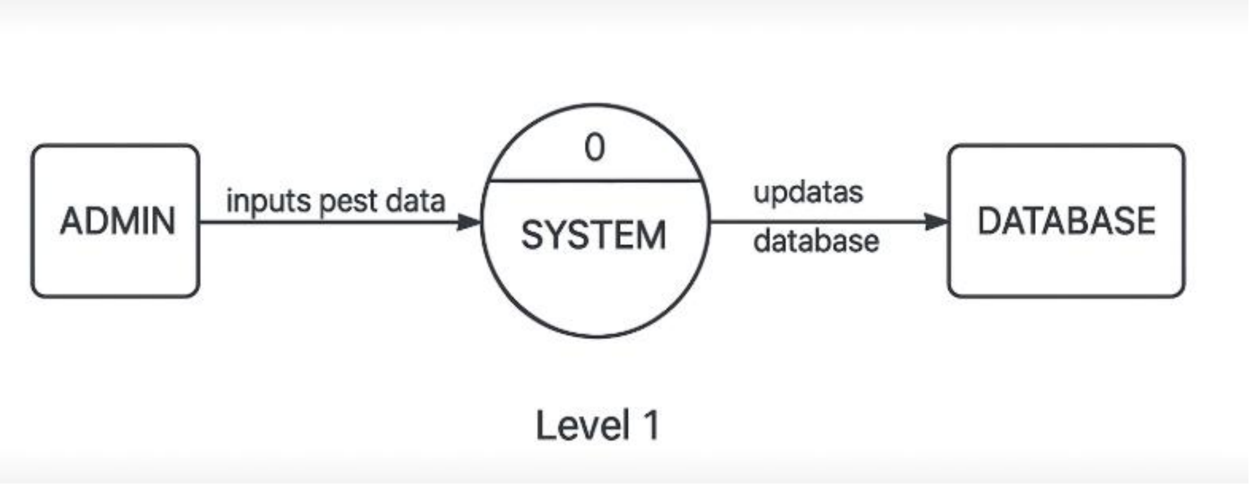
4. Analysis Models

4.1 Data Flow Diagrams (DFD)

- **Level 0**: User inputs location → System fetches data → Displays crop suggestions.



- **Level 1**: Admin inputs pest data → System updates database → Displays pest alerts.

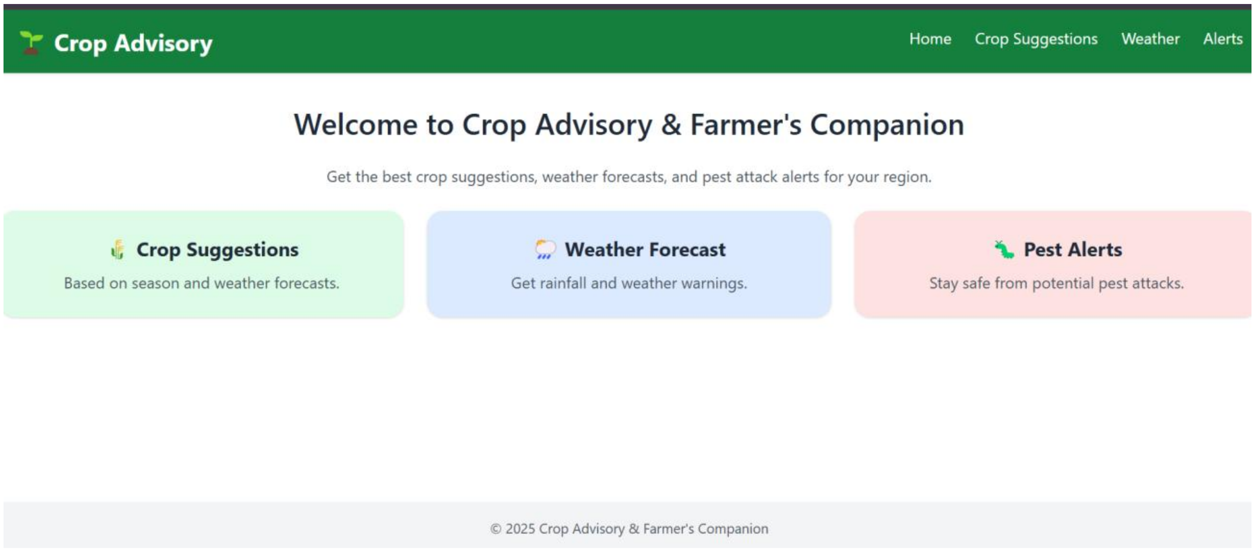


5. GITHUB LINK

https://github.com/Raghavgupta2003/Crop-Guidance-and-Farmers-Friend

A. Appendices

A.1 Appendix 1 – USER DASHBOARD



Welcome to Crop Advisory & Farmer's Companion

Get the best crop suggestions, weather forecasts, and pest attack alerts for your region.

Crop Suggestions

Based on season and weather forecasts.

Weather Forecast

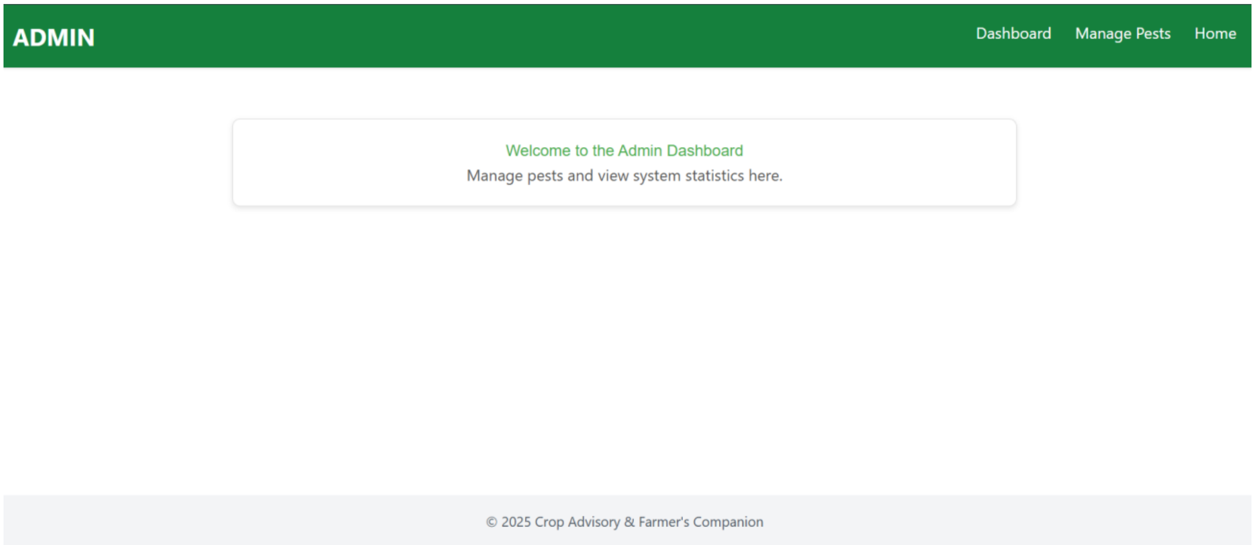
Get rainfall and weather warnings.

Pest Alerts

Stay safe from potential pest attacks.

© 2025 Crop Advisory & Farmer's Companion

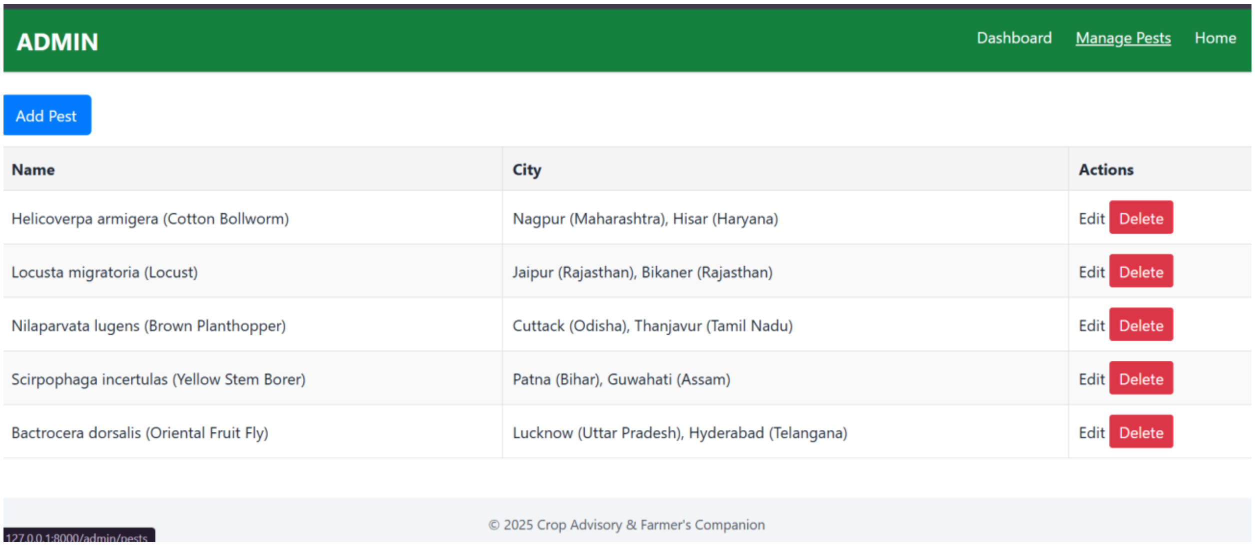
A.2 Appendix 2 – ADMIN DASHBOARD



Welcome to the Admin Dashboard

Manage pests and view system statistics here.

© 2025 Crop Advisory & Farmer's Companion



Name	City	Actions
Helicoverpa armigera (Cotton Bollworm)	Nagpur (Maharashtra), Hisar (Haryana)	Edit Delete
Locusta migratoria (Locust)	Jaipur (Rajasthan), Bikaner (Rajasthan)	Edit Delete
Nilaparvata lugens (Brown Planthopper)	Cuttack (Odisha), Thanjavur (Tamil Nadu)	Edit Delete
Scirpophaga incertulas (Yellow Stem Borer)	Patna (Bihar), Guwahati (Assam)	Edit Delete
Bactrocera dorsalis (Oriental Fruit Fly)	Lucknow (Uttar Pradesh), Hyderabad (Telangana)	Edit Delete

© 2025 Crop Advisory & Farmer's Companion