**Prototype Requirements**

**P08:AgriQual**

| **Student ID** | **Name** |
| --- | --- |
| **26100370** | **Zarak qadir khan** |
| **26100219** | **syeda umaima hasan** |
| **26100247** | **mishaal usman** |
| **26100259** | **muhammad walid khalid** |

**Table of Contents**

[1.](#_ihgb16v2rpl5) Introduction 3

[2.](#_9z3x847xmf5t) Instructions 4

[3.](#_1lyh5ef6mxyp) List of Requirements for Prototype 5

[4.](#_c44xyqi9q3h9) Review checklist 6

# Introduction

AgriQual is an AI-driven system designed to support farmers by providing personalized, real-time agricultural advice. It uses a multi-agent system, combining Vision, Climate, and Advisory agents to deliver comprehensive solutions aimed at optimizing crop yields, reducing losses, and improving farming efficiency. By leveraging advanced machine learning, farmers gain access to expert-level insights that enhance productivity, sustainability, and profitability.

The Vision Agent analyzes crop images using computer vision techniques to detect diseases, pests, and other indicators of crop health and quality. This allows farmers to monitor crops at different growth stages, enabling early interventions and precise management decisions.

The Climate Agent processes meteorological data from sources to provide accurate weather forecasts and climate risk assessments. This information helps farmers plan irrigation, fertilization, and other activities, while also preparing for long-term environmental risks such as droughts, floods, or frost.

The Advisory Agent integrates insights from the Vision and Climate Agents to generate actionable, personalized recommendations. These recommendations guide farmers on optimal irrigation schedules, pest and nutrient management, and harvest timing, ensuring holistic support for effective decision-making.

AgriQual also benefits quality inspectors by validating crop assessments to meet agricultural standards and supports system administrators in managing infrastructure, monitoring performance, and maintaining data integrity. By making advanced agricultural knowledge accessible, the system empowers farmers to improve efficiency and adopt sustainable practices. Ultimately, AgriQual aims to transform farming into a more data-driven, productive, and profitable activity.

# Instructions

* The prototype will be developed in two phases. Select a subset of system requirements and implement them. The result of each phase of prototype development must be a working system with the selected set of requirements implemented completely. Mere mock-up screens will NOT be accepted.
* While you may choose to implement Login/Logout functionality for prototype, you must also implement some core/business use cases of the system.
* Select the set of requirements keeping in mind that you have a total of 4 weeks for prototype development. You may be asked to add more requirements if more can be achieved in the given duration.
* The prototype must be built using the tools and technologies that you have selected for your system development.
* Follow standard coding practices.
* By the end of the prototype development phase,
  1. You should have learnt development tools and technologies.
  2. You should have a clear idea of detailed technical architecture of your system. After the prototype phase, you will be required to define detailed technical architecture of your system.
* **Prototype Submission**
* **Prototype Phase—1**
  1. Deploy the properly tested **working prototype** on an online hosting platform.
  2. Upload the **Code (zip file)** with proper comments in “prototype” folder of your project’s Github repository.
  3. Fill the provided template for **Readme-Prototype.txt** file and upload in “prototype” folder of your project’s Github repository.
  4. Fill the provided **Code Review Checklist** and upload in “prototype” folder of your project’s Github repository.
* **Prototype Phase—2**
  1. Deploy the properly tested **working prototype** on an online hosting platform.
  2. Upload the **Code (zip file)** with proper comments in “prototype” folder of your project’s Github repository.
  3. Prepare a **3-4 minutes video** that explains the functionality of your system prototype—to be uploaded in “prototype” folder of your project’s Github repository.
  4. Fill the provided template for **Readme-Prototype.txt** file and upload in “prototype” folder of your project’s Github repository.
  5. Fill the provided **Code Review Checklist** and upload in “prototype” folder of your project’s Github repository.

# List of Requirements for Prototype

<List down the requirements selected for prototype development separately for each part.>

* **Prototype Use Cases: Phase—1**

| **Requirements** | |
| --- | --- |
| **Sr#** | User Registration/Login, Get Weather-Aware Advisory |
| 1 | Registration form (name, email/phone, password, role: Farmer/Inspector) |
| 2 | Server-side validation (email/phone format, strong password, unique user) |
| 3 | OTP/email verification; resend flow; lock after 3 failed OTPs |
| 4 | Create account + role mapping; redirect to respective dashboard |
| 5 | Login form (email/phone + password) |
| 6 | Role-based routing (Farmer → Farmer Dashboard, Inspector → Inspector Dashboard) |
| 7 | “Forgot password” flow stub (request reset link/OTP) |
| 8 | Fetch farmer’s field location (profile or prompt if missing) |
| 9 | Call Weather API (e.g., OpenWeather) for forecast: temp, humidity, rainfall, wind |
| 10 | Combine weather + crop to generate **advisory** (e.g., irrigation, spray window, fertilizer timing) |

* **Prototype Use Cases: Phase—2**

| **Requirements** | |
| --- | --- |
| **Sr#** | Upload Wheat Image for Diagnosis,Manage Wheat Fields/Plots,View Diagnosis History |
| 1 | Upload JPEG/PNG (≤ 5MB) tied to a selected field. |
| 2 | Server stores image |
| 3 | Save diagnosis record with image URL, fieldId, label, confidence, timestamp |
| 4 | Add/edit/delete field with: name, area, location, variety, sowing date |
| 5 | Validate inputs (required fields, valid date, positive area) |
| 6 | List fields for the logged-in farmer |
| 7 | List diagnoses for the logged-in farmer (reverse chronological) |
| 8 | Click to view details (image, label, confidence, field link) |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| Introduction | mishaal |
| List of Requirements | Mishaal, Umaima, Zarak,Walid |