**Requirements Specifications**

**P08:agriQual**

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

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Introduction & system actors | 5 | 5 |
| Use case diagram | 20 | 20 |
| Use case descriptions | 20 | 20 |
| Class diagram | 20 | 18 |
| Sequence diagram | 20 | 20 |
| State diagram | 5 | 5 |
| Data Requirements | 20 | 20 |
| Non-functional requirements | 10 | 10 |
| Security requirements | 10 | 10 |
| Use of generative AI | 5 | 5 |
| Who did what | 5 | 5 |
| Review checklist | 5 | 5 |
| Overall formatting/template | 5 | 3 |
| Late submission penalty | -20 | - |
| GitHub Folder structure penalty | -5 | - |
| **Grand Total** | **150** | **146** |
| **General Comments/Individual Grading:** | | |

**Table of Contents**

[1. Introduction 3](#_sx4qedyuncfd)

[2. System Actors 4](#_qq2i8poji1fv)

[3. Use Cases 5](#_m7do6hdqv9y0)

[3.1 Use Case Diagrams 5](#_eqz2zsbyxfba)

[3.2 Description of Use Cases 5](#_bhdvhwmboioh)

[3.2.1](#_ga3x90h3dg8v) Upload Wheat Image for Diagnosis [5](#_ga3x90h3dg8v)

[3.2.2](#_qlp1o8ucard1) Get Weather-Aware Advisory [6](#_qlp1o8ucard1)

[4. Class Diagram 7](#_vmhqy1uj2fmi)

[4.1 Diagram 7](#_nsl0hmy2v0q4)

[4.2 Description 7](#_axtfr95hov6k)

[5. Sequence Diagrams 8](#_ybmxjahara9d)

[5.1 Use case Name e.g.,](#_muidzdsckd0a) Upload Wheat Image for Diagnosis 8

[5.2 Use case Name e.g.,](#_rhhcffcdaj3g) Get Weather-Aware Advisory 8

[6. State Diagrams 9](#_ueu3mgz9tem3)

[6.1 Diagram details 9](#_tgjjq44ajy0f)

[6.2 Diagram 9](#_58uzqzy1i4f7)

[7. Data Requirements 10](#_255syqbn9ytu)

[8. Non-functional Requirements / Quality Attributes 11](#_fyv7us6qiyyi)

[9. Security Requirements 12](#_9p13mqz0rza5)

[10. Security Engineer 13](#_nj4b03lcjn28)

[11. Use of Generative AI 14](#_n90t5bgxkq0h)

[12. Who Did What? 15](#_cgphm3ssy407)

[13. Review checklist 15](#_o8gqwgewa9s6)

# 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.

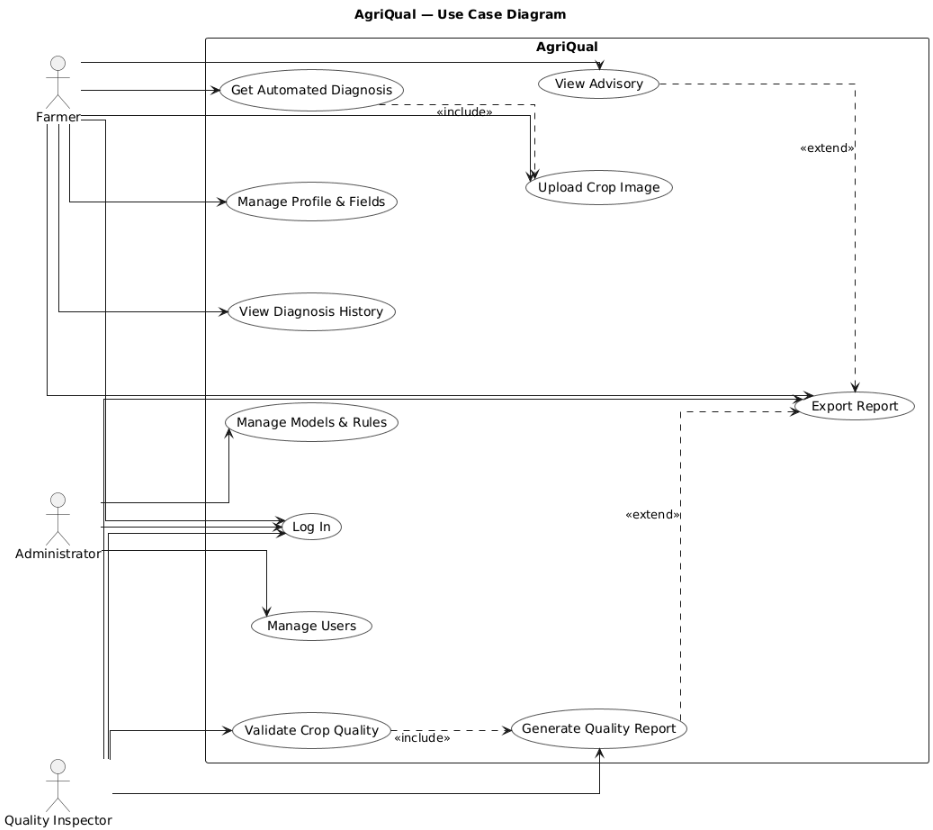
AgriQual also benefits quality inspectors by validating crop assessments to meet agricultural standards and supports system administrators in managing infrastructure, monitoring performance, and maintaining data integrity. By making advanced agricultural knowledge accessible, the system empowers farmers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

# System Actors

|  |  |
| --- | --- |
| **Actor Name** | **Description** |
| Farmer | Primary end-user who uses the system to get personalized advice by providing crop images, soil data and queries. |
| Quality Inspector | Professional who validates crop assessments and ensures accuracy of recommendations. |
| System Administrator | Technical personnel responsible for managing system operations, security and technical issues at the back end. |

# Use Cases

## 3.1: Use Case Diagrams



## 

## 3.2: Description of Use Cases

### 

### 3.2.1 User Registration

|  |  |  |
| --- | --- | --- |
| Identifier | | UC-001 |
| Purpose | | Allow a new farmer or quality inspector to create an account on the system. |
| Pre-conditions | | The user is not logged in and has access to the registration page. |
| Post-conditions | | A verified account is created and stored in the system with the correct role. |
|  | | |
| Step # | Typical Course of Action | |
| 1. | The user selects the “Register” option on the homepage | |
| 2. | The system displays the registration form with role options (Farmer/Inspector). | |
| 3. | The user enters name, email/phone, password, and selects a role. | |
| 4. | The system validates the entered information. | |
| 5. | The system sends an OTP/email for verification. | |
|  |  | |
| 6. | The user enters the OTP/email verification code. | |
| 7. | The system verifies the code successfully. | |
| 8. | The account is created and linked to the chosen role. | |
| 9. | The system redirects the user to their dashboard. | |
| 10. | The use case ends. | |
|  |  | |
|  | | |
| Step # | Alternate Courses of Action | |
| 1. | In step 1, if the user selects 'Sign up with Google/Apple', the system redirects to the provider and fetches verified details. | |
| 2. | In step 5, if the OTP is not received, the user may request a resend and the system issues a new code. | |
| 3. | At step 4, the system flags weak passwords. System shows strength meter and suggestions; user updates password. Continue at step 5. | |
| Step # | Exception Paths | |
| 1. | In step 3, if the email/phone format is invalid, the system shows an error and asks the user to correct it. | |
| 2. | In step 4, if the email/phone is already registered, the system stops registration and offers Login/Password Recovery. | |
| 3. | In step 6, if the OTP entered is incorrect three times, the system locks verification temporarily. | |
| 4. | In step 8, if the database fails to create the account, the system rolls back and shows an error message. | |

### 

### 3.2.2 Secure Login

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-002 |
| **Purpose** | | Allow registered users to securely log in and access the system. |
| **Pre-conditions** | | The user has a registered and verified account. |
| **Post-conditions** | | A secure session is created, and the user is redirected to their dashboard. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user selects the 'Login' option on the homepage. | |
| 2. | The system displays the login form. | |
| 3. | The user enters email/phone and password. | |
| 4. | The system validates the credentials. | |
| 5. | If credentials are correct, the system establishes a secure session. | |
| 6. | The system redirects the user to the appropriate dashboard (Farmer/Inspector). | |
| 7. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 1, if the user chooses 'Login with Google/Apple', the system redirects to the provider and retrieves verified credentials. | |
| 2. | In step 2, if the user selects 'Forgot Password', the system sends a reset link or OTP and guides the user through resetting their password before returning to login. | |
| 3. | In step 5, if there is an existing active session on another device, the system notifies the user and provides the option to continue or terminate the other session. | |
| Step # | **Exception Paths** | |
| 1. | In step 3, if the user enters an invalid email/phone format, the system shows an error and requests correction. | |
| 2. | In step 4, if the credentials are incorrect, the system displays an error and allows limited retries. | |
| 3. | In step 4, if the account is locked after multiple failed attempts, the system denies login and instructs the user to reset their password or contact support. | |

### 3.2.3 Upload Wheat Image for Diagnosis

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-003 |
| **Purpose** | | Farmers upload wheat images to detect diseases or pest issues. |
| **Pre-conditions** | | The farmer is logged in and has access to the upload feature. |
| **Post-conditions** | | The uploaded image is processed, and a diagnostic result is stored. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer selects the 'New Wheat Diagnosis' option. | |
| 2. | The system displays an image upload/camera option. | |
| 3. | The farmer uploads or captures a wheat image (leaf, stem, or ear). | |
| 4. | The system validates the file format and size. | |
| 5. | The system processes the image using the vision agent. | |
| 6. | The advisory agent generates a diagnosis and recommendations. | |
| 7. | The system displays the diagnosis result to the farmer. | |
| 8. | The result is saved to the farmer's history. | |
| 9. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 3, if the farmer cancels the upload midway, the system discards the current request and returns to the dashboard. | |
| 2. | In step 3, if the uploaded image is blurry or low-quality, the system prompts the farmer to retake or enhance the image before proceeding. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 3, if the file format is unsupported or the image size exceeds the limit, the system rejects the upload and informs the farmer of acceptable formats and sizes. | |
| 2. | In step 6, if the advisory agent fails to generate recommendations, the system shows a generic advisory and logs the error for review. | |
| 3. | In step 8, if saving the diagnosis record to the database fails, the system alerts the farmer and allows retry. | |

### 3.2.4 Get Weather-Aware Advisory

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-004 |
| **Purpose** | | Provide wheat farmers with crop care recommendations based on weather forecasts. |
| **Pre-conditions** | | The farmer has at least one registered wheat field or a recent diagnosis. |
| **Post-conditions** | | An advisory is generated, displayed to the farmer, and stored in the system for reference. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer opens the 'Advisory' section from the dashboard. | |
| 2. | The system retrieves the farmer’s registered field location. | |
| 3. | The system fetches the latest weather forecast for that location (temperature, rainfall, humidity, wind). | |
| 4. | The system combines weather data with wheat growth stage and health information. | |
| 5. | The system generates tailored recommendations (e.g., irrigation schedule, spraying window, fertilizer timing). | |
| 6. | The system displays the advisory to the farmer in an easy-to-read format. | |
| 7. | The system saves the advisory to the farmer’s history for future reference. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if no field location is registered, the system prompts the farmer to set a location manually before continuing. | |
| 2. | In step 4, if the farmer selects a different crop stage than the default, the system recalculates the advisory based on the chosen stage. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the system cannot retrieve the farmer’s field location due to missing data, it displays an error and requests manual input. | |
| 2. | In step 3, if the weather API is unavailable, the system uses cached forecast data (if available) and shows a warning. | |
| 3. | In step 3, if the forecast data returned is incomplete or corrupted, the system informs the farmer and defaults to generic advisories. | |
| 4. | In step 4, if crop growth stage data is missing, the system asks the farmer to input the stage manually before proceeding. | |

### 3.2.5 Manage Wheat Fields/Plots

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-005 |
| **Purpose** | | Allow farmers to add, update, and manage their wheat fields. |
| **Pre-conditions** | | The farmer is logged in. |
| **Post-conditions** | | Field information is saved and linked to crop history. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer selects the 'My Fields' option from the dashboard. | |
| 2. | The system displays a list of the farmer’s existing wheat fields (if any). | |
| 3. | The farmer chooses to add a new field, edit an existing one, or delete a field. | |
| 4. | For adding or editing, the farmer enters details (field name, area, location, wheat variety, sowing date). | |
| 5. | The system validates the entered information. | |
| 6. | The system saves the field details to the database. | |
| 7. | The system updates the list of fields displayed to the farmer. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if no fields are registered, the system shows a message 'No fields found' and prompts the farmer to add a new field. | |
| 2. | In step 3, if the farmer chooses 'Copy Field', the system duplicates an existing field entry and allows quick edits before saving. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 4, if required details (e.g., location or variety) are missing, the system highlights missing fields and blocks submission until completed. | |
| 2. | In step 4, if invalid data is entered (e.g., negative area size, invalid date), the system rejects it and prompts correction. | |

### 

### 

### 

### 3.2.6 View Diagnosis History

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-006 |
| **Purpose** | | Allow farmers to view and track past wheat diagnoses. |
| **Pre-conditions** | | The farmer has at least one past diagnosis record in the system. |
| **Post-conditions** | | The farmer can see their diagnosis history, filter records, and export reports if needed. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer selects the 'History' option from the dashboard. | |
| 2. | The system retrieves all past diagnosis records for the farmer. | |
| 3. | The system displays the records in a list with details such as date, field, issue type, and result. | |
| 4. | The farmer applies filters (by field, date range, or issue type) to narrow down results. | |
| 5. | The system updates the list based on the filters applied. | |
| 6. | The farmer clicks on a diagnosis entry to view detailed information. | |
| 7. | The farmer optionally exports the history as a PDF or CSV report. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the farmer has many records, the system paginates results and provides navigation controls. | |
| 2. | In step 3, if the farmer sorts by date or issue type, the system reorders the list accordingly. | |
| 3. | In step 4, if no filters are applied, the system shows the complete unfiltered list by default. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the farmer has no past records, the system shows 'No diagnosis history found' with a link to upload a new diagnosis. | |
| 2. | In step 5, if invalid filter criteria are applied, the system ignores the filter and shows a warning message. | |
| 3. | In step 7, if the report export service fails, the system notifies the farmer and allows retry. | |

### 3.2.7 Farmer Updates Wheat Field Information

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-007 |
| **Purpose** | | Allow farmers to update details of their registered wheat fields. |
| **Pre-conditions** | | The farmer has at least one registered wheat field in the system. |
| **Post-conditions** | | The updated field details are saved and reflected in the farmer’s profile. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer selects the 'My Wheat Fields' option from the dashboard. | |
| 2. | The system displays a list of registered wheat fields. | |
| 3. | The farmer selects a specific field to edit. | |
| 4. | The system shows the current details of the selected field (variety, area, sowing date, etc.). | |
| 5. | The farmer modifies one or more field details. | |
| 6. | The system validates the updated details. | |
| 7. | The system saves the updated field information to the database. | |
| 8. | The updated field list is shown to the farmer. | |
| 9. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the farmer has many fields, the system provides a search bar or filter to quickly locate a specific field. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 3, if the selected field does not exist or was deleted by mistake, the system shows 'Field not found' and prevents editing. | |
| 2. | In step 5, if the farmer enters invalid data (e.g., negative area size, incorrect date format), the system highlights errors and requests correction. | |
| 3. | In step 7, if the database update fails, the system displays an error and allows the farmer to retry. | |

### 3.2.8 Wheat Crop Quality Assessment

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-008 |
| **Purpose** | | Inspectors perform quality assessment on harvested wheat crops (single lot). |
| **Pre-conditions** | | The inspector is logged in |
| **Post-conditions** | | The wheat crop assessment is saved in the system with a grade. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The inspector selects the 'New Wheat Crop Assessment' option. | |
| 2. | The system prompts the inspector to enter crop details (lot ID, quantity, wheat variety, intended market/export standard). | |
| 3. | The inspector uploads sample images or enters physical measurements (e.g., grain size, color, moisture content). | |
| 4. | The system validates the uploaded data and images. | |
| 5. | The system analyzes wheat crop quality using AI (color, grain uniformity, defect detection). | |
| 6. | The system compares results against the chosen market/export standard. | |
| 7. | The system displays the assessment with a quality grade (e.g., A, B, C) and detailed findings. | |
| 8. | The inspector confirms and saves the crop assessment. | |
| 9. | The system stores the assessment in the database with a timestamp and unique assessment ID. | |
| 10. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 5, if AI detects borderline results, the inspector can override or adjust the grading with justification notes. | |
| 2. | In step 6, if the inspector changes the selected market/export standard, the system recalculates the grade using new thresholds. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if required crop details (lot ID, quantity, variety) are missing, the system highlights the missing fields and prevents continuation. | |
| 2. | In step 3, if uploaded images are corrupt or in unsupported formats, the system rejects them and prompts for re-upload. | |
| 3. | In step 6, if the chosen export standard is not in the system, it prompts the inspector to pick another or contact an administrator. | |

### 3.2.9 Generate Compliance Certificate

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-009 |
| **Purpose** | | Generate a compliance certificate for wheat crops that meet quality standards. |
| **Pre-conditions** | | A wheat crop quality assessment exists with a passing grade. |
| **Post-conditions** | | A compliance certificate is generated, stored, and available for download or sharing. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The inspector selects a completed wheat crop assessment with a passing grade. | |
| 2. | The system compiles assessment data (lot ID, variety, grade, metrics, date). | |
| 3. | The system generates a draft compliance certificate with all necessary details. | |
| 4. | The inspector reviews the draft certificate for accuracy. | |
| 5. | The inspector digitally signs and approves the certificate. | |
| 6. | The system assigns a unique certificate ID and generates a QR code for verification. | |
| 7. | The system finalizes and saves the compliance certificate in the database. | |
| 8. | The system provides options for the inspector to download or share the certificate. | |
| 9. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the inspector selects multiple crop assessments, the system allows generating certificates in bulk. | |
| 2. | In step 3, if the inspector chooses a specific template (e.g., export vs. domestic), the system generates the certificate using that template format. | |
| 3. | In step 4, if the inspector detects errors in the draft, they can return to the assessment to make corrections before certificate generation. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 1, if no completed passing assessment is available, the system prevents certificate generation and shows an error. | |
| 2. | In step 3, if the certificate generation service fails, the system retries and alerts the inspector if unsuccessful. | |
| 3. | In step 7, if saving the certificate to the database fails, the system prevents issuance and logs the error for review. | |

### 3.2.10 Contact Support

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-010 |
| **Purpose** | | Allow farmers or inspectors to contact the support team for help or to report issues |
| **Pre-conditions** | | The user is logged in and the support option is available |
| **Post-conditions** | | The support request is recorded and confirmation is provided to the user. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user selects the 'Contact Support' option from the dashboard menu. | |
| 2. | The system displays a support form with fields such as Name, Email, Subject, and Message. | |
| 3. | The user fills out the form with their query or issue. | |
| 4. | The system validates the inputs (mandatory fields, correct email format). | |
| 5. | The user submits the form. | |
| 6. | The system records the support request in the database with a unique ticket ID. | |
| 7. | The system sends a confirmation message or email to the user acknowledging receipt. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the system detects the user’s account information, it auto-fills Name  and Email fields to save time. | |
| 2. | In step 3, if the user attaches a file (e.g., screenshot), the system stores it with the support request. | |
| 3. | In step 5, if the user cancels before submitting, the system discards the draft support request. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 3, if the message body is empty, the system highlights the missing field and prevents submission | |
| 2. | In step 4, if the email format is invalid, the system displays an error message and blocks submission until corrected. | |
| 3. | In step 5, if the attachment exceeds the size limit, the system rejects the file and prompts the user to upload a smaller one. | |
| 4 | In step 6, if the database is unavailable, the system logs the error and informs the user to retry later. | |

### 3.2.11 Receive Weather Alerts

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-011 |
| **Purpose** | | Automatically notify farmers about severe weather conditions that could impact crops health |
| **Pre-conditions** | | The farmer has registered fields with location data in the system and enabled weather notifications. |
| **Post-conditions** | | Weather alerts are sent to the farmer |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The system continuously monitors weather data for all registered field locations. | |
| 2. | The system detects severe weather conditions (storms, frost, drought, high winds) approaching a farmer's area. | |
| 3. | The system analyzes the weather threat against current crop types and growth stages. | |
| 4. | The system sends an immediate alert notification via the farmer's preferred method (SMS, email, push notification). | |
| 5. | The farmer receives the alert with weather details, impact assessment, and recommended actions | |
| 6. | The farmer acknowledges the alert or requests additional information. | |
| 7. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 5, if the primary notification method fails, the system attempts backup communication channels | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if weather data feed is interrupted, the system uses backup weather sources to predict forecast | |
| 2. | In step 3, if crop data is missing, the system sends generic weather warnings rather than crop-specific advice. | |

### 3.2.12 Generate Inspection Statistics

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-012 |
| **Purpose** | | Allow quality inspectors to create statistical reports showing assessment trends, and quality improvements over time. |
| **Pre-conditions** | | The inspector has completed multiple quality assessments over the reporting period |
| **Post-conditions** | | A comprehensive statistical report is generated showing inspection trends and patterns |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The inspector selects 'Generate Statistics Report' from the reports section. | |
| 2. | The system prompts for report parameters (date range, crop types, assessment categories). | |
| 3. | The inspector specifies the reporting period and filters (monthly, quarterly, by region, by crop). | |
| 4. | The system retrieves all relevant assessment data from the specified timeframe | |
| 5. | The system calculates statistics including grade distributions, common issues, and trend analysis. | |
| 6. | The system generates charts and graphs showing assessment patterns and quality trends.. | |
| 7. | The inspector reviews the statistical summary and adds interpretive comments. | |
| 8. | The system compiles the final report with statistics, visualizations, and inspector observations. | |
| 9. | The report is saved and made available for download | |
| 10 | The use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 3, if the inspector wants to compare with previous periods, the system includes comparative analysis. | |
| 2 | In step 8, if multiple inspectors contributed data, the system provides inspector-specific breakdowns. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 4, if insufficient data exists for the selected period, the system suggests expanding the date range. | |
| 2. | In step 5, if data inconsistencies are detected, the system flags problematic records for review. | |

### 3.2.13 Farmer Views Wheat Crop Tips

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-013 |
| **Purpose** | | Allow farmers to view general wheat cultivation guidelines and tips. |
| **Pre-conditions** | | The farmer is logged in. |
| **Post-conditions** | | The farmer is able to access and read wheat crop cultivation tips. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer selects the 'Crop Tips' option from the dashboard. | |
| 2. | The system retrieves a list of general crop cultivation tips (e.g., soil preparation, irrigation, fertilization, harvesting). | |
| 3. | The system displays the tips categorized by topic for easy browsing. | |
| 4. | The farmer scrolls through and reads the tips. | |
| 5. | The farmer optionally searches or filters tips by category or keyword. | |
| 6. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 4, if the farmer bookmarks a tip, the system saves it to their personal favorites list for future access. | |
| 2. | In step 5, if the farmer requests, the system generates a downloadable PDF version of the selected tips. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the tips database is empty, the system shows 'No crop tips available' and suggests contacting support. | |
| 2. | In step 5, if the search query returns no results, the system displays 'No matching tips found' and suggests broader keywords. | |

### 

### 3.2.14 Notifications & Reminders

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-014 |
| **Purpose** | | Notify farmers and inspectors about important tasks, risks, or advisories. |
| **Pre-conditions** | | The user has notifications enabled in their account settings. |
| **Post-conditions** | | Notifications are delivered to the user and interaction is logged. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The system detects a relevant event (e.g., diagnosis ready, weather risk, scheduled advisory). | |
| 2. | The system composes a notification message with event details. | |
| 3. | The system determines the user’s preferred notification channel (SMS, email, or app push). | |
| 4. | The system delivers the notification to the user. | |
| 5. | The user views the notification and optionally takes action (e.g., opens advisory, marks complete). | |
| 6. | The system logs delivery status and user interaction. | |
| 7. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 3, if the user changes their preference, the system updates the delivery channel accordingly. | |
| 2. | In step 4, if the user has enabled 'Digest Mode', the system accumulates notifications and sends them as a daily/weekly summary instead of immediate delivery. | |
| 3. | User can also disable notification from the website | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the notification message cannot be composed due to missing data, the system generates a fallback generic message. | |
| 2. | In step 6, if logging fails due to a database error, the system temporarily stores delivery details in cache and retries later. | |

### 3.2.15 Manage Knowledge Base

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-015 |
| **Purpose** | | Allow administrators to create, update, and manage wheat crop and disease knowledge base content. |
| **Pre-conditions** | | The administrator is logged in with proper access rights. |
| **Post-conditions** | | Knowledge base entries are updated and versioned for use by farmers and inspectors. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The administrator selects the 'Knowledge Base' option from the dashboard. | |
| 2. | The system displays the current list of knowledge base entries. | |
| 3. | The administrator chooses to create a new entry or edit an existing one. | |
| 4. | The system displays a form for entering or updating the content, including title, category, and body text. | |
| 5. | The administrator fills in or updates the knowledge base details. | |
| 6. | The system validates the entered data. | |
| 7. | The administrator submits the entry for publishing. | |
| 8. | The system saves the entry, assigns a version number, and publishes it to the knowledge base. | |
| 9. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 3, if the administrator chooses 'Delete Entry', the system archives the entry instead of permanently deleting it. | |
| 2. | In step 4, if the administrator prefers, they can attach images or documents to the knowledge base entry for richer content. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the system fails to load the current entries due to a server issue, it displays an error and prompts retry later. | |
| 2. | In step 4, if required fields (title, body) are missing, the system prevents saving until all required fields are filled. | |
| 3. | In step 5, if attached files exceed size limits or use unsupported formats, the system rejects them and informs the administrator. | |
| 4. | In step 8, if the database fails during saving, the system logs the error and notifies the administrator of failure to publish. | |

### 

### 

### 

### 3.2.16 Privacy & Consent Management

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-016 |
| **Purpose** | | Allow users to manage their data privacy settings and consents. |
| **Pre-conditions** | | The user is logged in and has access to account settings. |
| **Post-conditions** | | User preferences are updated and enforced by the system. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user navigates to 'Privacy & Consent Settings' from their account dashboard. | |
| 2. | The system displays current privacy options and consents (e.g., data sharing, notifications, analytics). | |
| 3. | The user reviews and modifies their preferences (grant/revoke consents). | |
| 4. | The system validates the changes for completeness and compliance. | |
| 5. | The user confirms and submits their updated privacy settings. | |
| 6. | The system saves the updated consents to the database and updates processing rules accordingly. | |
| 7. | The system displays a confirmation message to the user. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 3, if the user requests account deletion, the system initiates the deletion workflow with confirmation steps. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if privacy settings cannot be retrieved due to a system error, the system shows an error message and prevents edits. | |
| 2. | In step 3, if the user submits invalid input (e.g., unchecked mandatory consent required by law), the system blocks submission until corrected. | |
| 3. | In step 6, if saving to the database fails, the system logs the error and prevents changes from taking effect. | |
| 4. | In step 6, if the system fails to propagate changes to enforcement modules, it shows a warning and requests retry. | |

### 

### 3.2.17 Performance Guardrail (SLA Monitoring)

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-017 |
| **Purpose** | | Ensure that wheat image diagnosis requests are processed within the defined SLA (e.g., 15 seconds). |
| **Pre-conditions** | | A diagnosis request is made and SLA monitoring is enabled. |
| **Post-conditions** | | SLA compliance is tracked, and alerts are generated in case of breaches. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The system timestamps the start of the image diagnosis request. | |
| 2. | The system monitors processing time as the vision agent analyzes the wheat image. | |
| 3. | The system continuously checks the elapsed time against the SLA threshold (15 seconds). | |
| 4. | If the request completes within SLA, the system logs success and returns results to the farmer. | |
| 5. | If the request is nearing the SLA threshold, the system triggers auto-scaling or optimization to speed up processing. | |
| 6. | If the request exceeds the SLA, the system logs a breach, triggers alerts, and provides fallback recommendations to the farmer. | |
| 7. | The system compiles SLA compliance statistics for monitoring dashboards. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 4, if the request completes faster than expected, the system records the performance gain for optimization insights. | |
| 2. | In step 5, if auto-scaling is unavailable, the system prioritizes critical requests over low-priority ones to maintain SLA compliance. | |
| 3. | In step 7, SLA metrics may be exported for external audit or compliance reporting. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if monitoring fails due to missing timestamps, the system marks the request as unverifiable and logs an error. | |
| 2. | In step 3, if system clocks are not synchronized across servers, the SLA calculation may be inaccurate; the system falls back to server-side timing. | |
| 3. | In step 5, if auto-scaling fails due to cloud resource limits, the system logs the issue and alerts administrators. | |
| 4. | In step 7, if the metrics database is unavailable, the system caches SLA compliance data locally until the database is restored. | |

### 

### 3.2.18 Farmer Deletes Old Wheat Diagnosis

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-018 |
| **Purpose** | | Allow farmers to delete outdated or irrelevant wheat diagnosis records. |
| **Pre-conditions** | | The farmer is logged in and has at least one diagnosis record available. |
| **Post-conditions** | | The selected diagnosis record(s) are deleted or archived from the farmer’s history. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer navigates to the 'Diagnosis History' section from the dashboard. | |
| 2. | The system retrieves and displays a list of past wheat diagnoses. | |
| 3. | The farmer selects one or more diagnosis records to delete. | |
| 4. | The system prompts the farmer with a confirmation message to prevent accidental deletion. | |
| 5. | The farmer confirms the deletion action. | |
| 6. | The system deletes or archives the selected records from the database. | |
| 7. | The system refreshes the list of diagnoses to reflect the deletion. | |
| 8. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if there are many records, the system provides search and filter options to help locate the diagnosis quickly. | |
| 2. | In step 3, the farmer can choose 'Delete All' for bulk deletion of older records within a selected date range. | |
| 3. | In step 4, if the farmer cancels at the confirmation prompt, the deletion is aborted and no records are removed. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 2, if the diagnosis history cannot be retrieved due to a server error, the system shows an error message and retry option. | |
| 2. | In step 3, if the selected record ID does not exist or was already deleted, the system shows 'Record not found'. | |
| 3. | In step 6, if the database fails during deletion, the system logs the error and notifies the farmer that the deletion was unsuccessful. | |
| 4. | In step 7, if the updated list cannot be refreshed, the system still confirms deletion but shows a fallback message. | |

### 3.2.19 Role-Based Access Control (RBAC) Enforcement

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-019 |
| **Purpose** | | Ensure that only authorized roles (farmer, inspector, admin) can access certain features or actions. |
| **Pre-conditions** | | The user is authenticated and requests access to a protected resource or action. |
| **Post-conditions** | | Access is granted or denied based on role permissions, and the decision is logged. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user attempts to access a protected feature or action in the system. | |
| 2. | The system identifies the user’s role (e.g., Farmer, Inspector, Admin). | |
| 3. | The system checks the requested action against the role’s access permissions. | |
| 4. | If the user’s role is authorized, the system grants access and executes the action. | |
| 5. | If the user’s role is not authorized, the system denies access and displays an error message. | |
| 6. | The system logs the access decision (granted/denied) with user ID, role, and timestamp. | |
| 7. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the user has multiple roles (e.g., Admin + Inspector), the system prompts them to choose which role to use for the session. | |
| 2. | In step 3, if the requested action requires elevated privileges, the system provides an option to request temporary access from an admin. | |
| 3. | In step 5, if access is denied, the system may suggest alternative features that the current role can access instead. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 1, if the user is not authenticated, the system redirects them to the login page. | |
| 2. | In step 2, if the system fails to identify the user’s role due to corrupted data, it denies access and logs the issue. | |
| 3. | In step 3, if the role permissions table is unavailable due to a database error, the system defaults to 'deny all' for security. | |
| 4. | In step 4, if access is incorrectly granted due to misconfigured permissions, the system logs the incident for review. | |

### 3.2.20 Farmer Changes Password

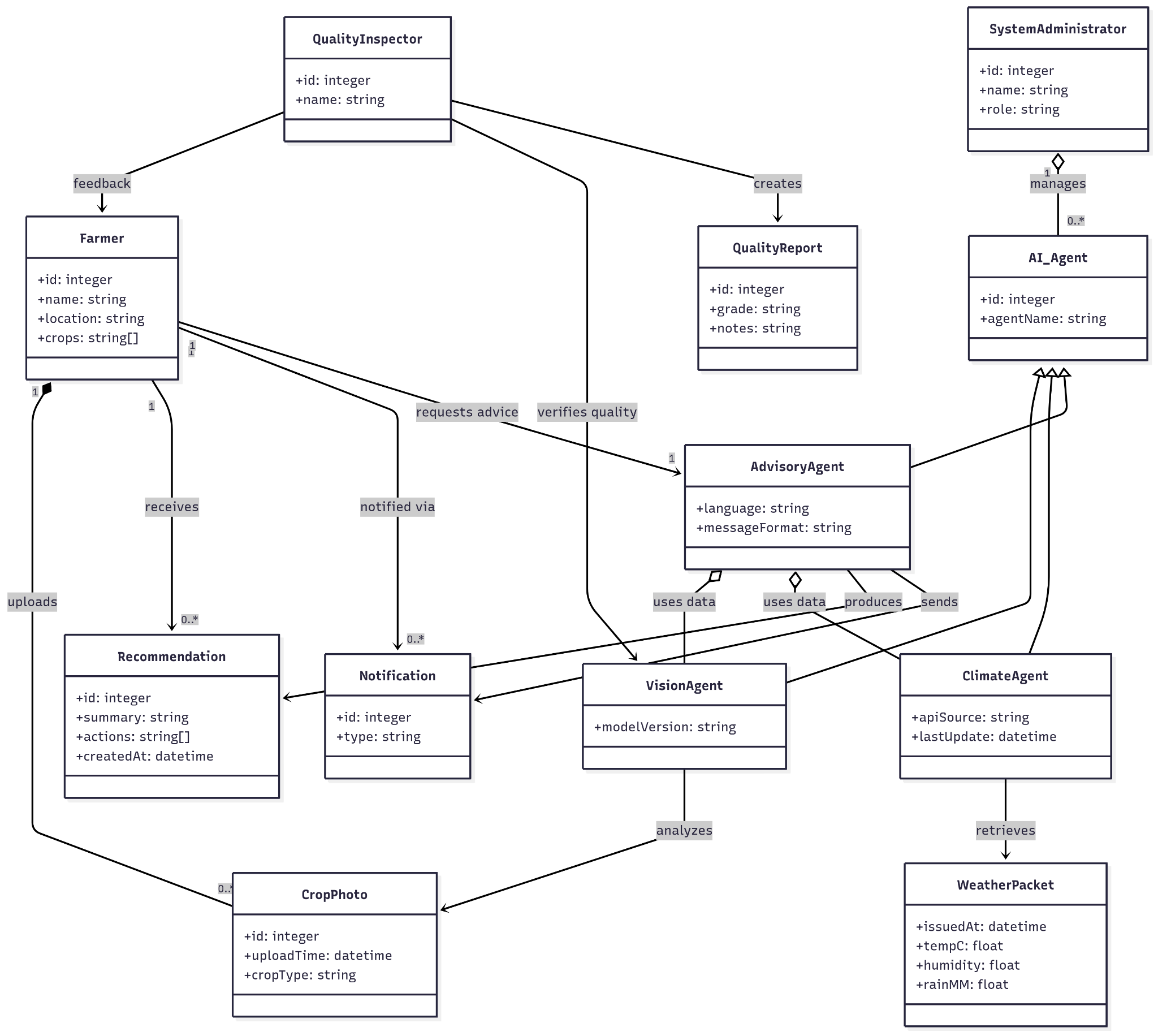
### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-020 |
| **Purpose** | | Allow farmers to securely change their account password. |
| **Pre-conditions** | | The farmer is logged in and has access to account settings. |
| **Post-conditions** | | The farmer’s password is successfully updated in the system. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The farmer navigates to 'Account Settings' and selects the 'Change Password' option. | |
| 2. | The system displays a password change form with fields for current password, new password, and confirm password. | |
| 3. | The farmer enters their current password and the desired new password twice for confirmation. | |
| 4. | The system validates the current password against stored credentials. | |
| 5. | The system checks the new password for strength and compliance with security policies. | |
| 6. | The farmer submits the form. | |
| 7. | The system updates the password in the database. | |
| 8. | The system notifies the farmer of a successful password change and logs out all active sessions except the current one. | |
| 9. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In step 2, if the farmer has forgotten their current password, they can use the 'Forgot Password' workflow to reset it via OTP or email verification. | |
| 2. | In step 3, if the farmer enables the 'Show Password' option, the system displays the entered password characters for easier input validation. | |
| 3. | In step 5, if the new password is weak, the system suggests stronger alternatives with a password strength meter. | |
| 4. | In step 6, if the farmer enables 'Log out from all devices', the system ensures all previous sessions are invalidated after the password change. | |
| **Step #** | **Exception Paths** | |
| 1. | In step 3, if the new password and confirmation password do not match, the system rejects the update and prompts correction. | |
| 2. | In step 4, if the current password is incorrect, the system denies the request and displays an error message. | |
| 3. | In step 5, if the new password does not meet security requirements, the system prevents submission until corrected. | |
| 4. | In step 7, if the database update fails, the system logs the error and notifies the farmer that the password change was unsuccessful. | |

# Class Diagram

Remove the words (creates, manages etc) from edges

## Diagram



## Description

## Farmer

**Purpose:** Primary user of the system who receives advice and submits crop photos.  
 **Attributes**

* **id: integer** – Unique farmer identifier.
* **name: string** – Farmer’s full name.
* **location: string** – Village/tehsil/GPS area used for weather & market context.
* **crops: string[]** – List of crops currently grown (e.g., ["wheat","cotton"]).

## CropPhoto

**Purpose:** Image submitted by a farmer for diagnosis.  
 **Attributes**

* **id: integer** – Unique photo id.
* **uploadTime: datetime** – Server time when the photo was received.
* **cropType: string** – Crop depicted (e.g., *wheat*).

## AI\_Agent

**Purpose:** Base class for all agents; unifies interface and enables polymorphism.  
 **Attributes**

* **id: integer** – Unique agent id.
* **agentName: string** – Human-readable name (e.g., *VisionAgent*).

## VisionAgent : AI\_Agent

**Purpose:** Computer-vision analysis of crop photos (diseases, pests, deficiencies).  
 **Attributes**

* **modelVersion: string** – Model build/tag used for inference.

## ClimateAgent : AI\_Agent

**Purpose:** Retrieves and summarizes weather conditions/forecasts.  
 **Attributes**

* **apiSource: string** – Provider identifier (e.g., *OpenWeather*).
* **lastUpdate: datetime** – Most recent successful weather sync timestamp.

## AdvisoryAgent : AI\_Agent

**Purpose:** Aggregates outputs from all agents and generates farmer-facing advice.  
 **Attributes**

* **language: string** – Language of the generated message.
* **messageFormat: string** – Delivery style/format (e.g., *concise*, *step-by-step*).

## Recommendation

**Purpose:** A single actionable advisory delivered to a farmer.  
 **Attributes**

* **id: integer** – Recommendation id.
* **summary: string** – Plain-language overview (one–two sentences).
* **actions: string[]** – Concrete steps (e.g., “Irrigate 15 mm tomorrow morning”).
* **createdAt: datetime** – Time the advice was produced.

## Notification

**Purpose:** Transport wrapper for sending advice/alerts to the farmer.  
 **Attributes**

* **id: integer** – Notification id.
* **type: string** – Channel, e.g., *Email*, App notification.

## QualityInspector

**Purpose:** Human/expert role to assess harvest quality and give feedback.  
 **Attributes**

* **id: integer** – Inspector id.
* **name: string** – Inspector’s name.

## QualityReport

**Purpose:** Result of a quality inspection.  
 **Attributes**

* **id: integer** – Report id.
* **grade: string** – Outcome/grade (e.g., *A, B, C* or *Meets Standard*).
* **notes: string** – Observations and improvement suggestions.

## SystemAdministrator

**Purpose:** Operates and maintains the platform and models.  
 **Attributes**

* **id: integer** – Admin id.
* **name: string** – Admin name.
* **role: string** – Role or scope (e.g., *Ops*, *ML Engineer*).

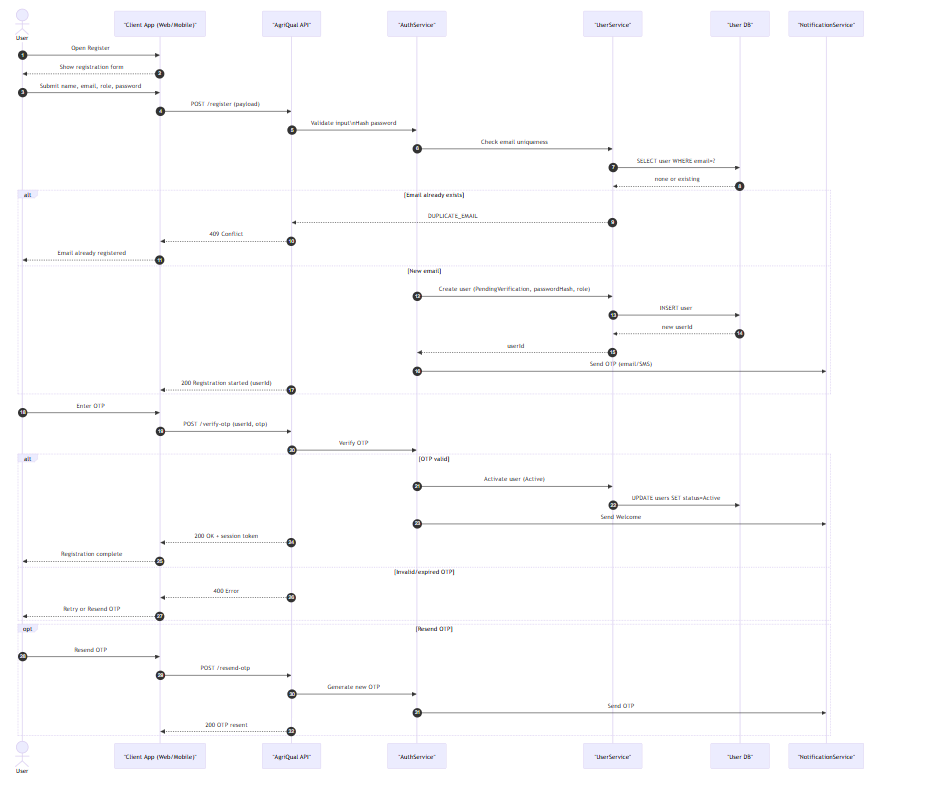
## WeatherPacket

**Purpose:** Structured weather snapshot/forecast values consumed by ClimateAgent & advisors.  
 **Attributes**

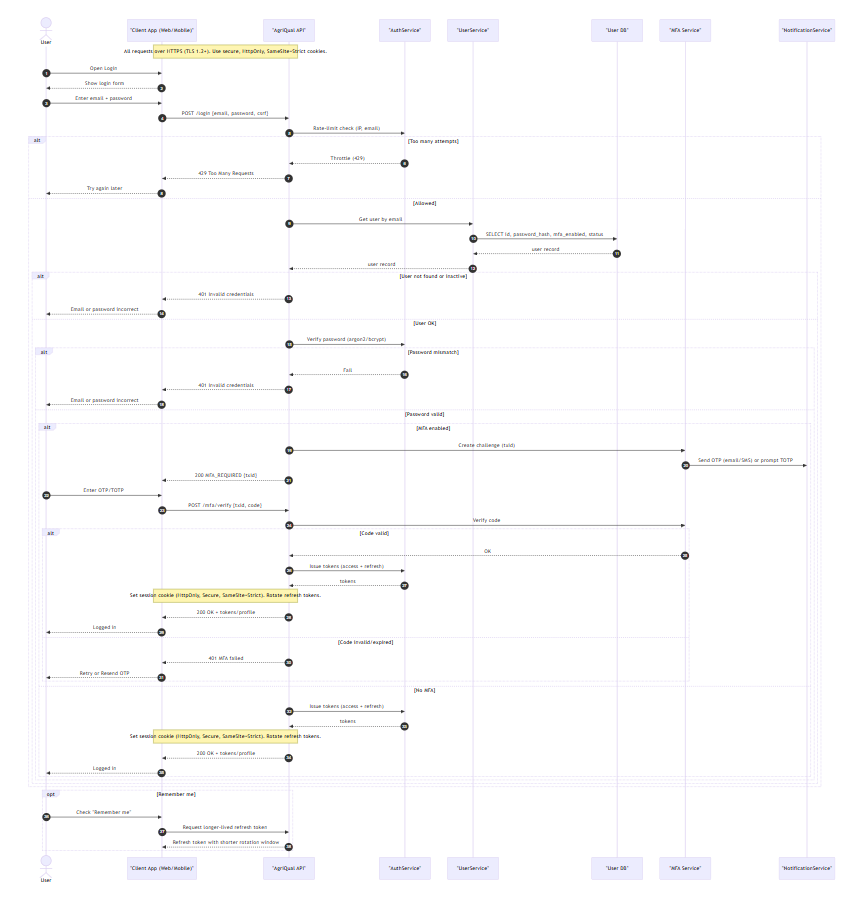
* **issuedAt: datetime** – Forecast issuance time.
* **tempC: float** – Air temperature in °C.
* **humidity: float** – Relative humidity percentage (0–100).
* **rainMM: float** – Predicted/observed rainfall in millimeters for the period.

# Sequence Diagrams

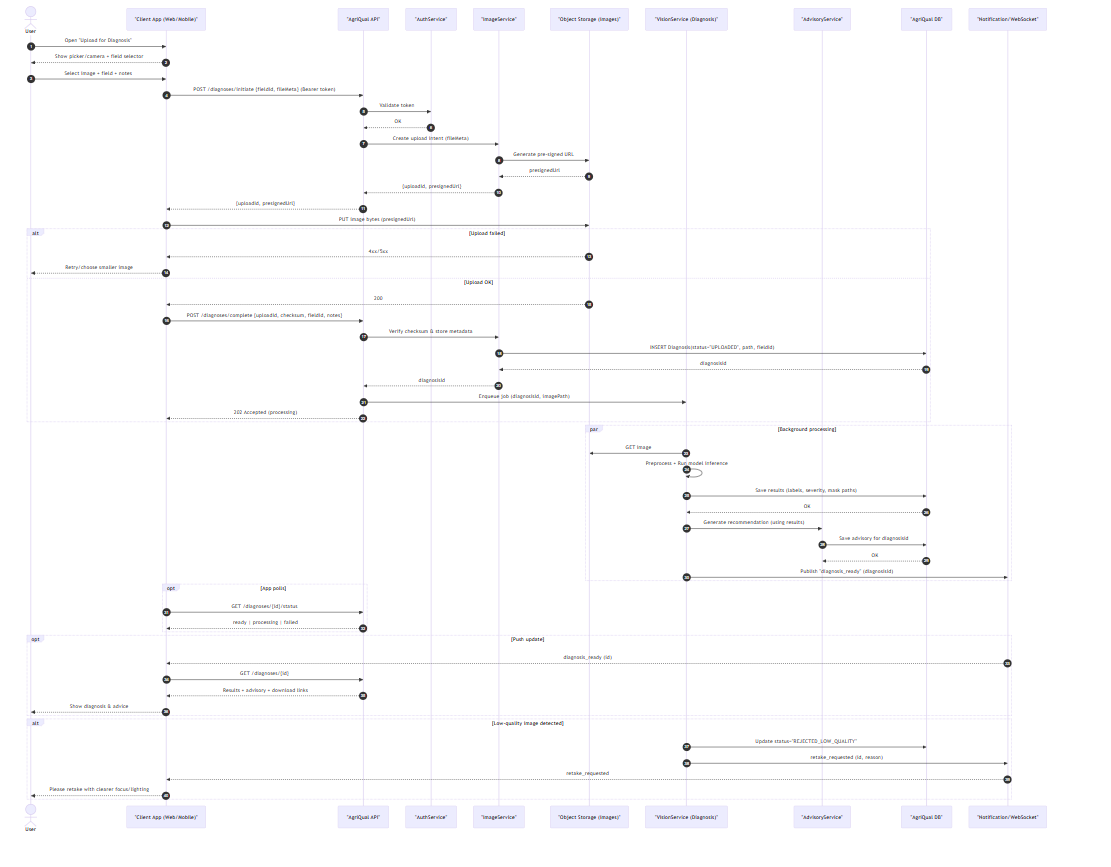
1. User Registration



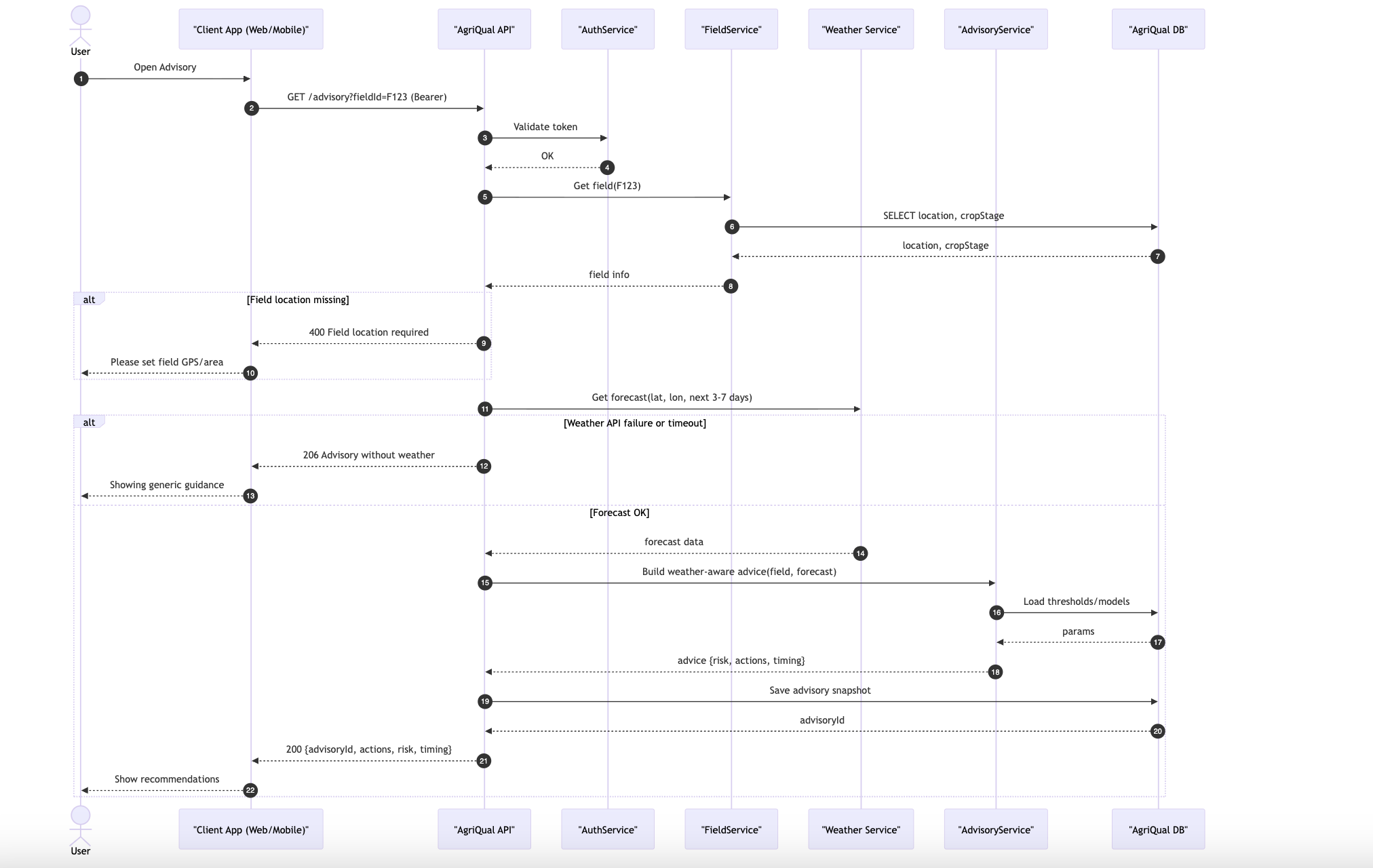
1. Secure Login



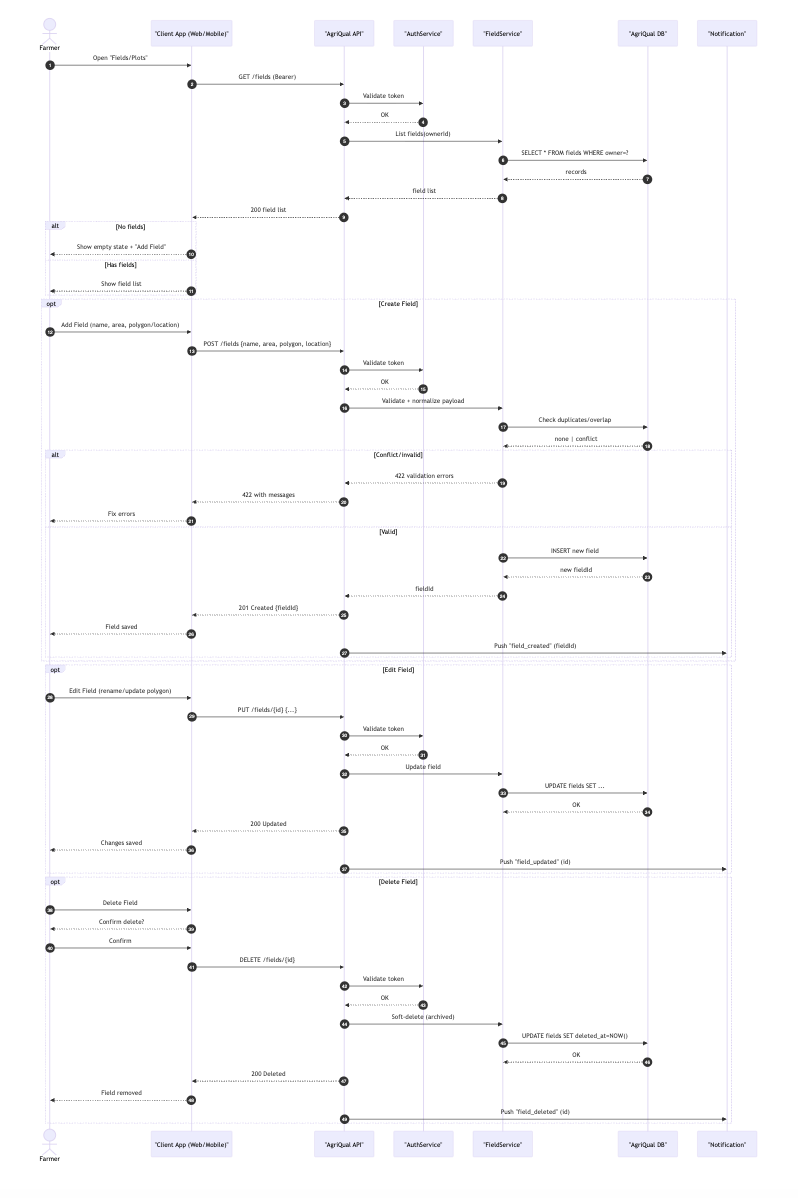
### Upload Wheat Image for Diagnosis



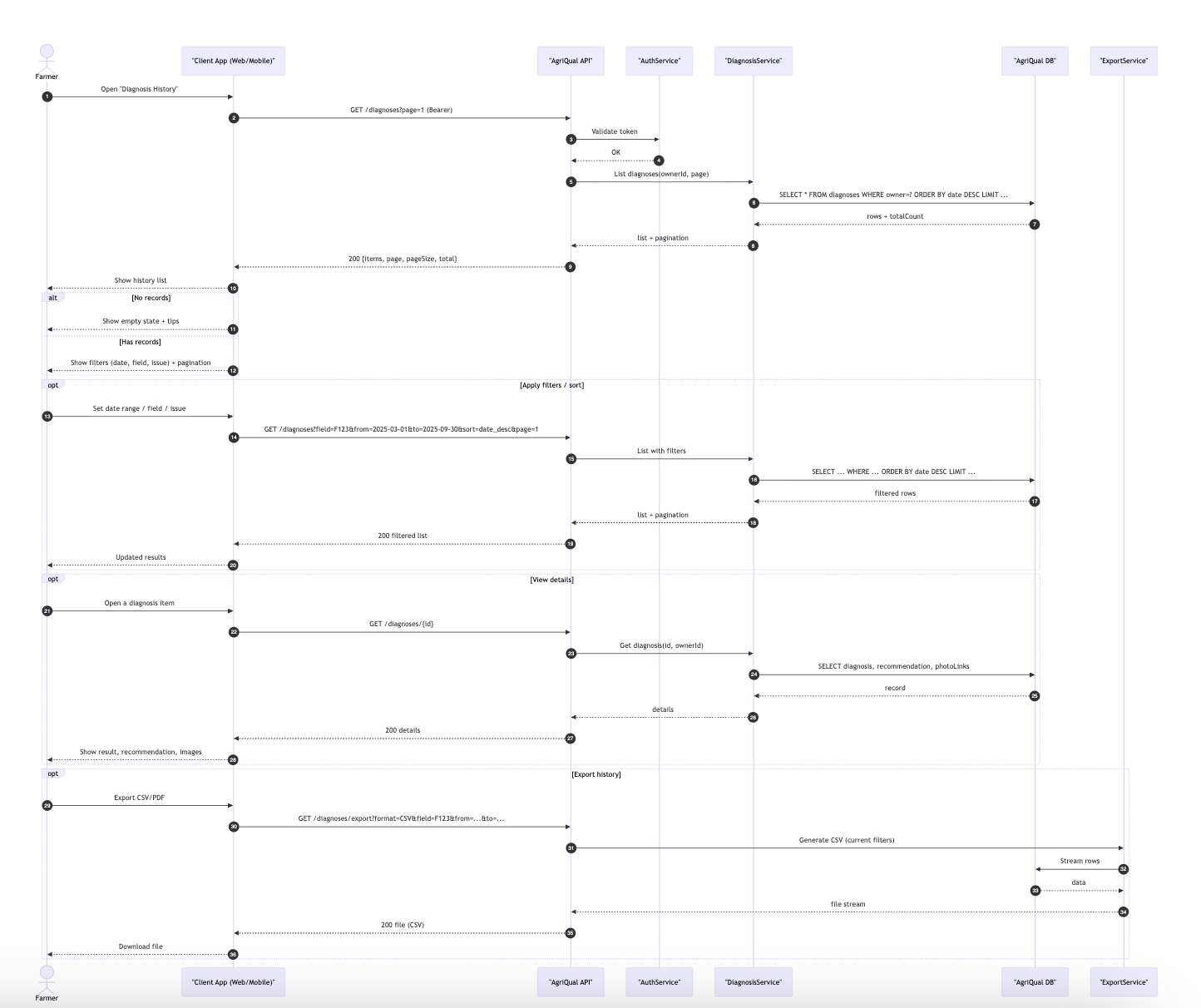
1. Get Weather Aware Advisor



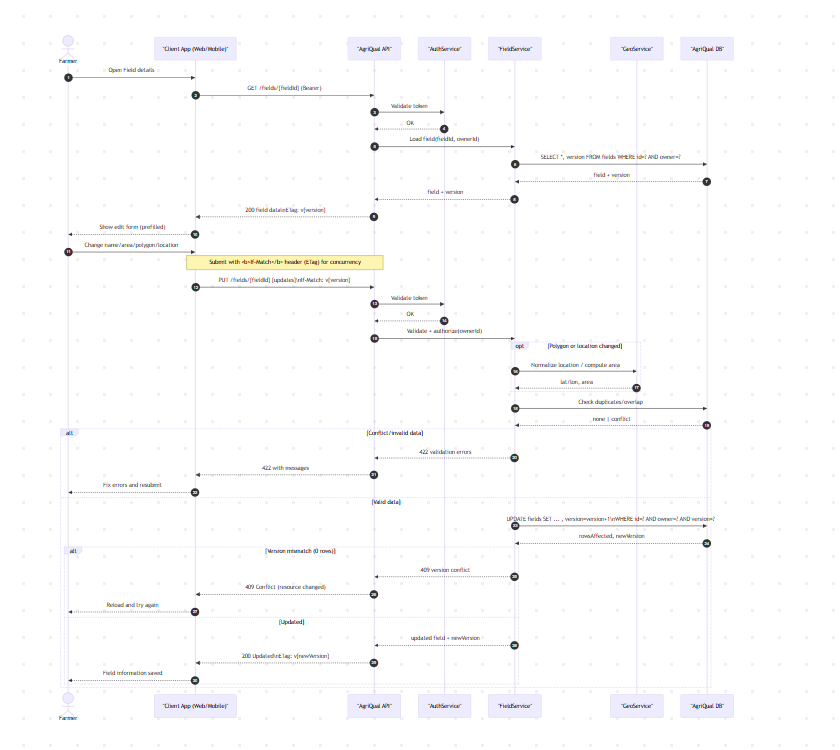
### Manage Wheat Fields/Plots



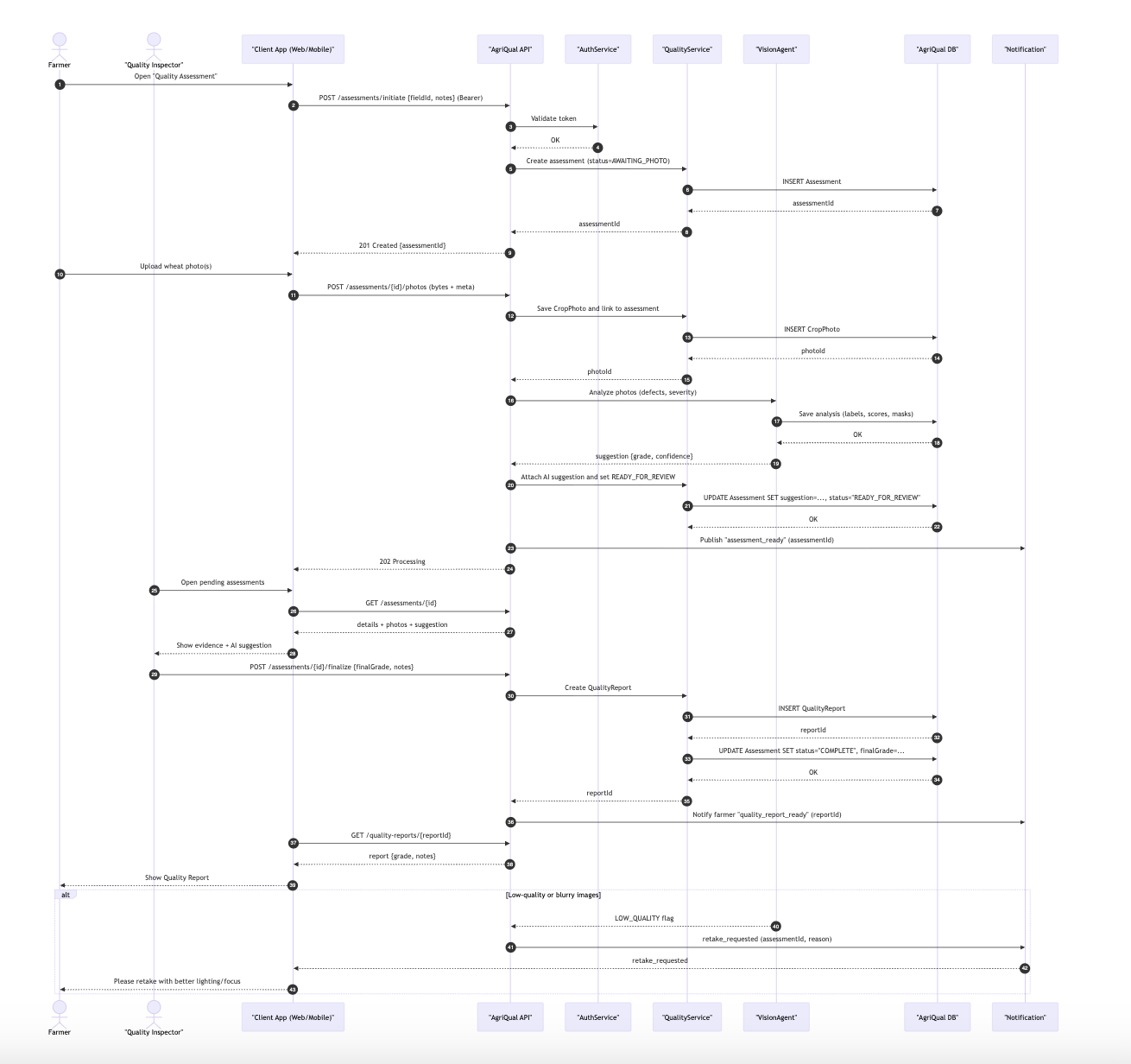
1. View Diagnosis History



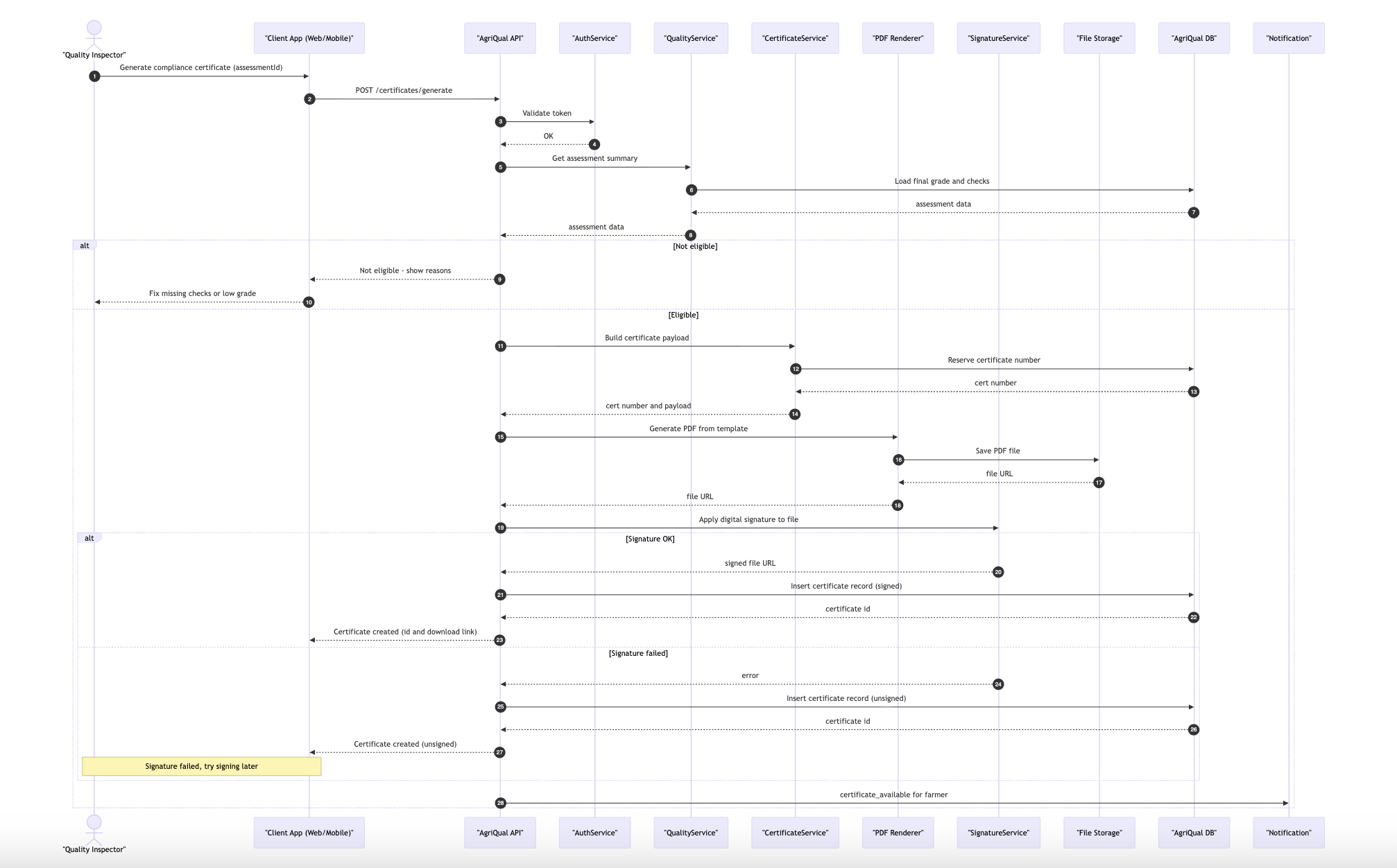
### Farmer Updates Wheat Field Information



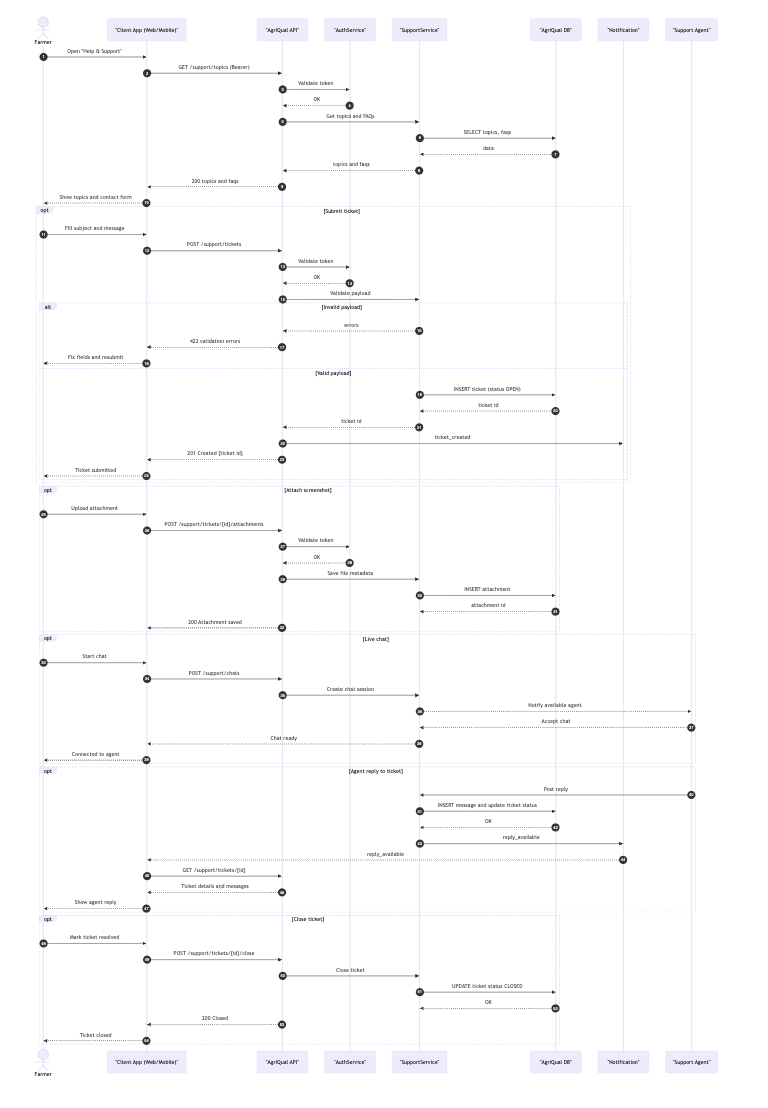
### Wheat Crop Quality Assessment



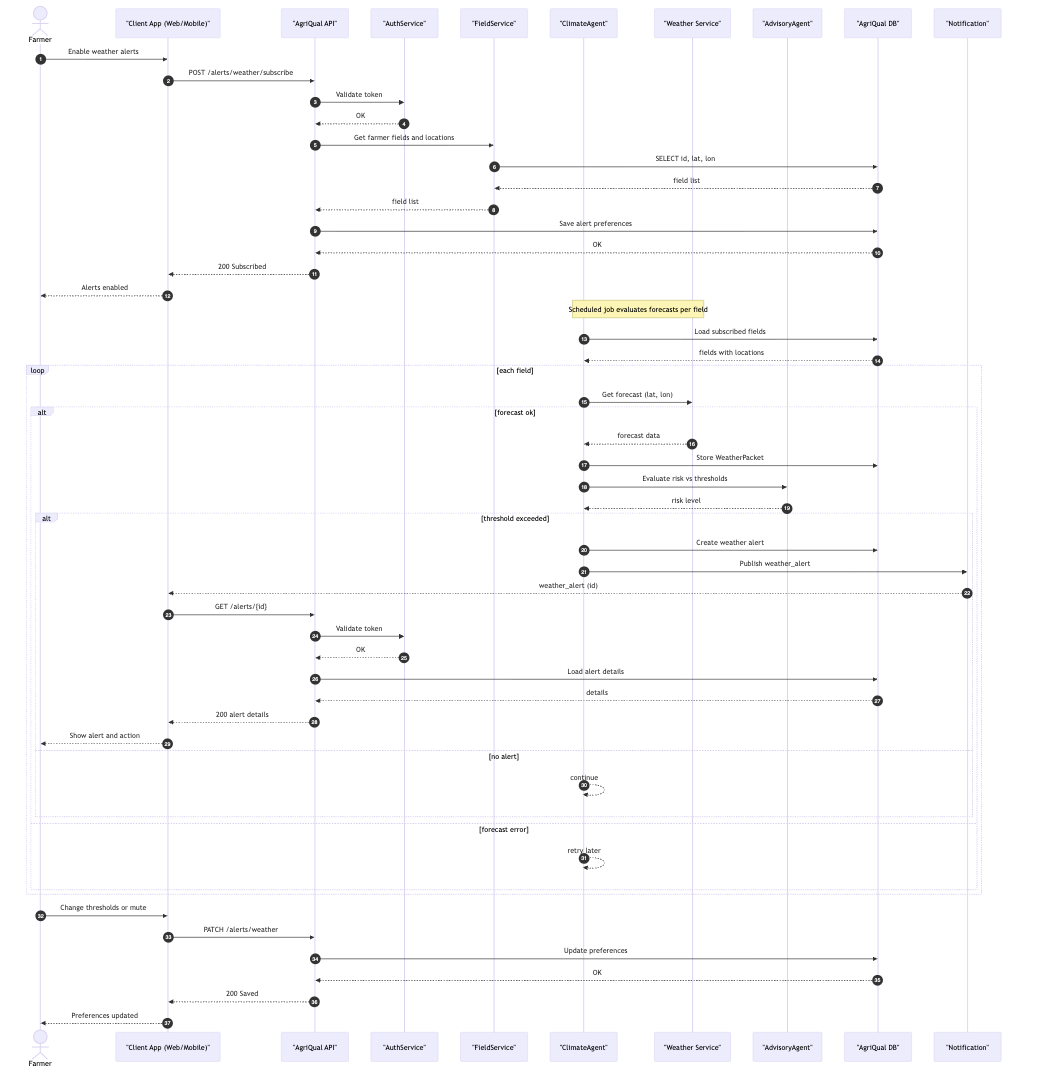
### Generate Compliance Certificate



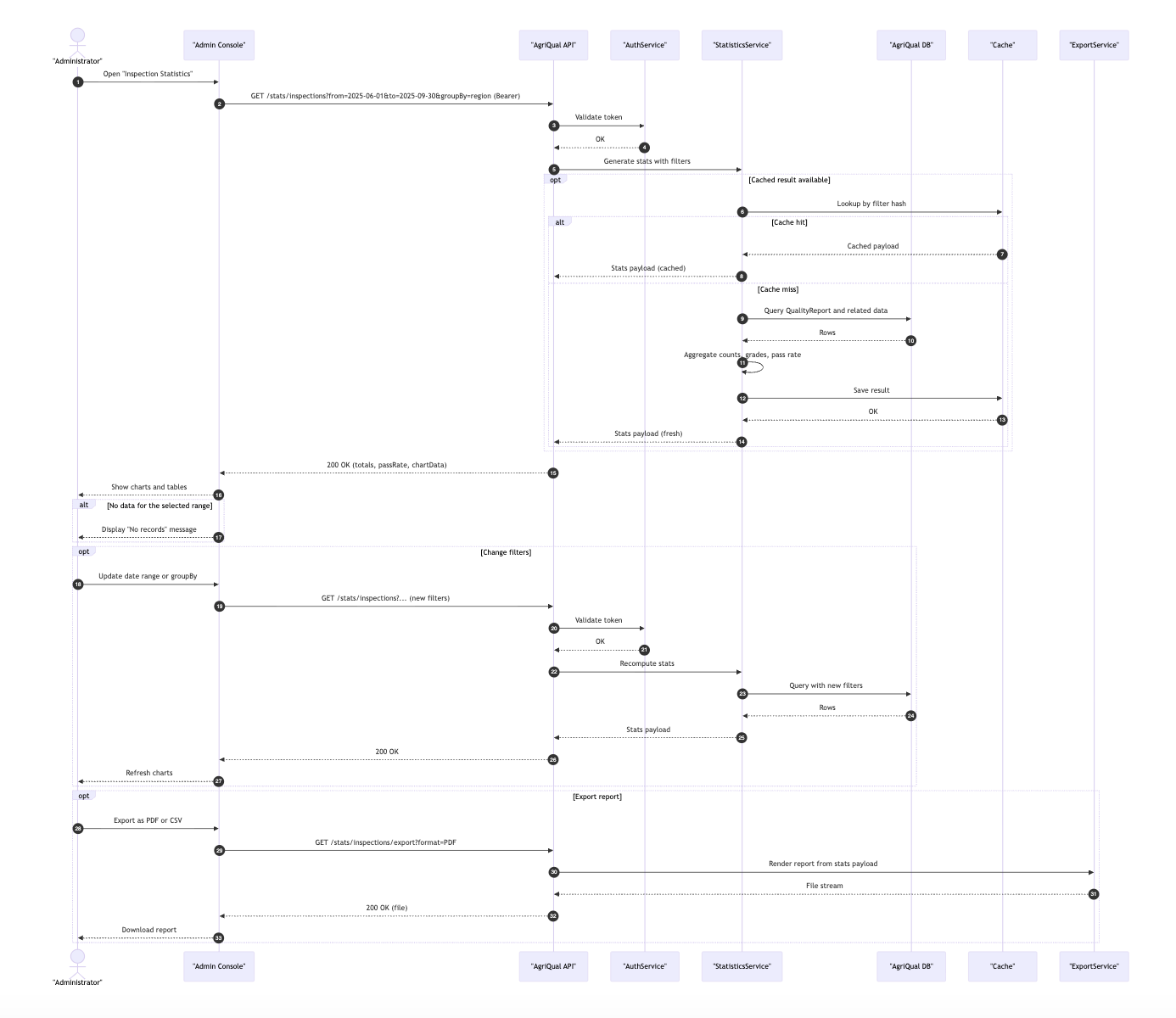
1. Contact Support



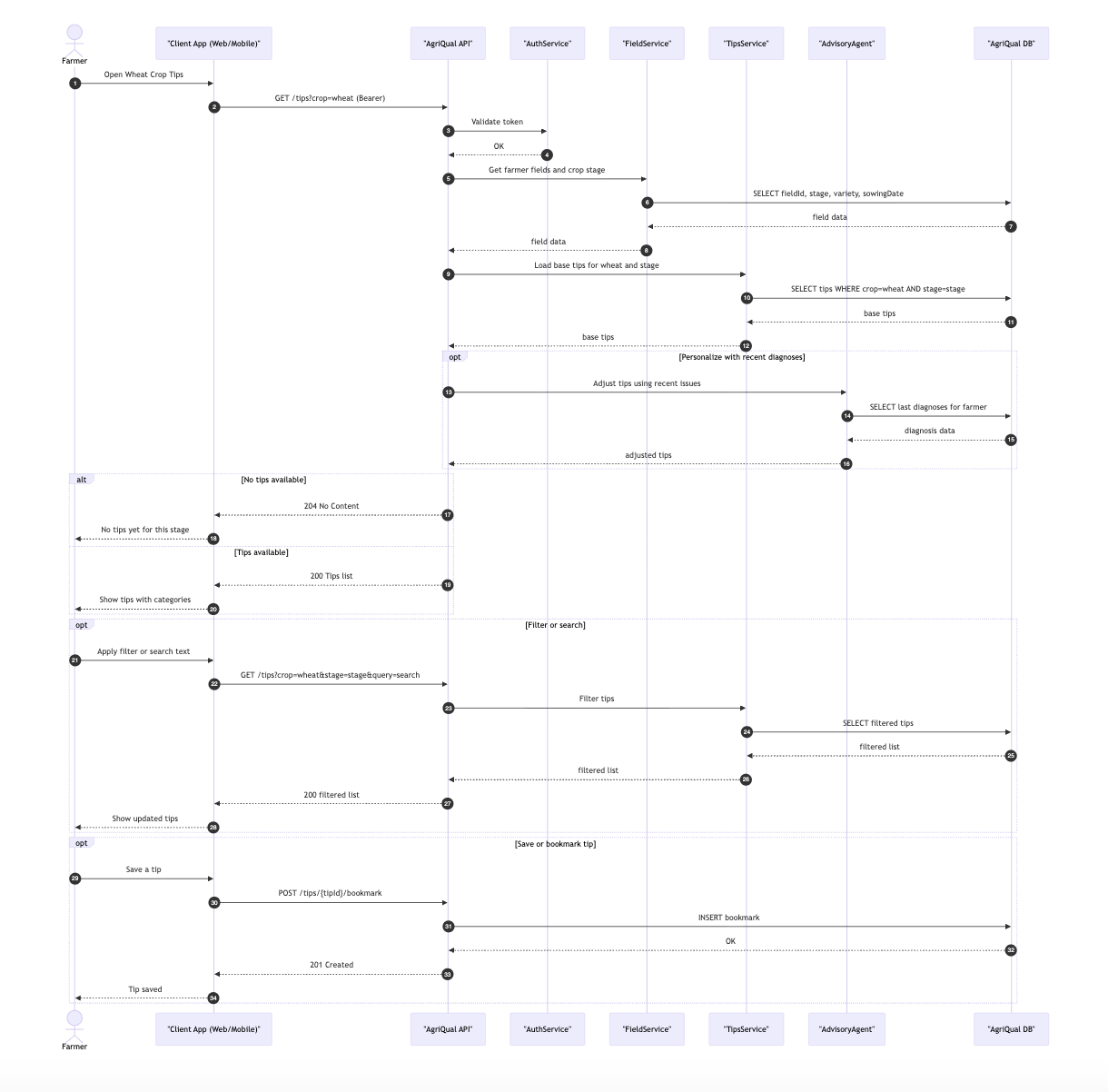
### Receive Weather Alerts



### Generate Inspection Statistics



### Farmer Views Wheat Crop Tips



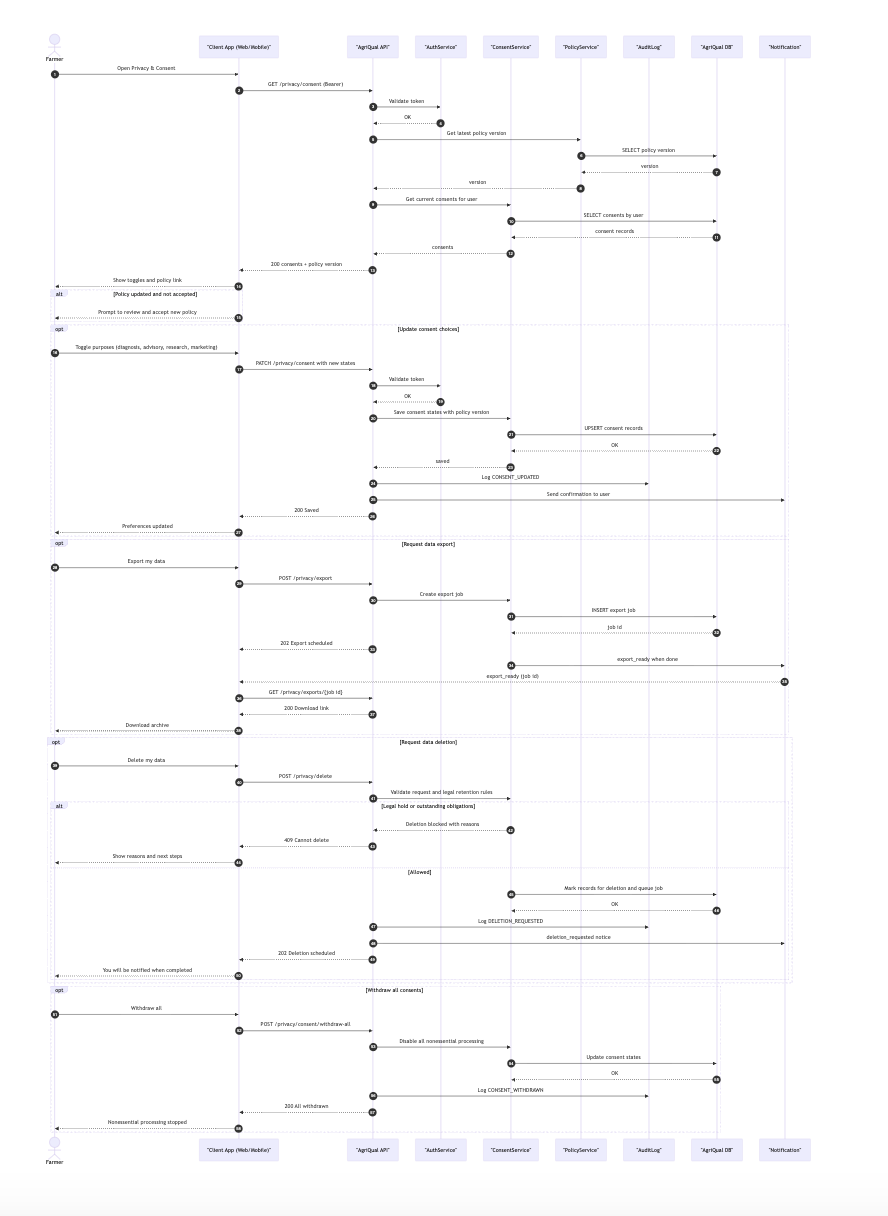
### Notifications & Reminders



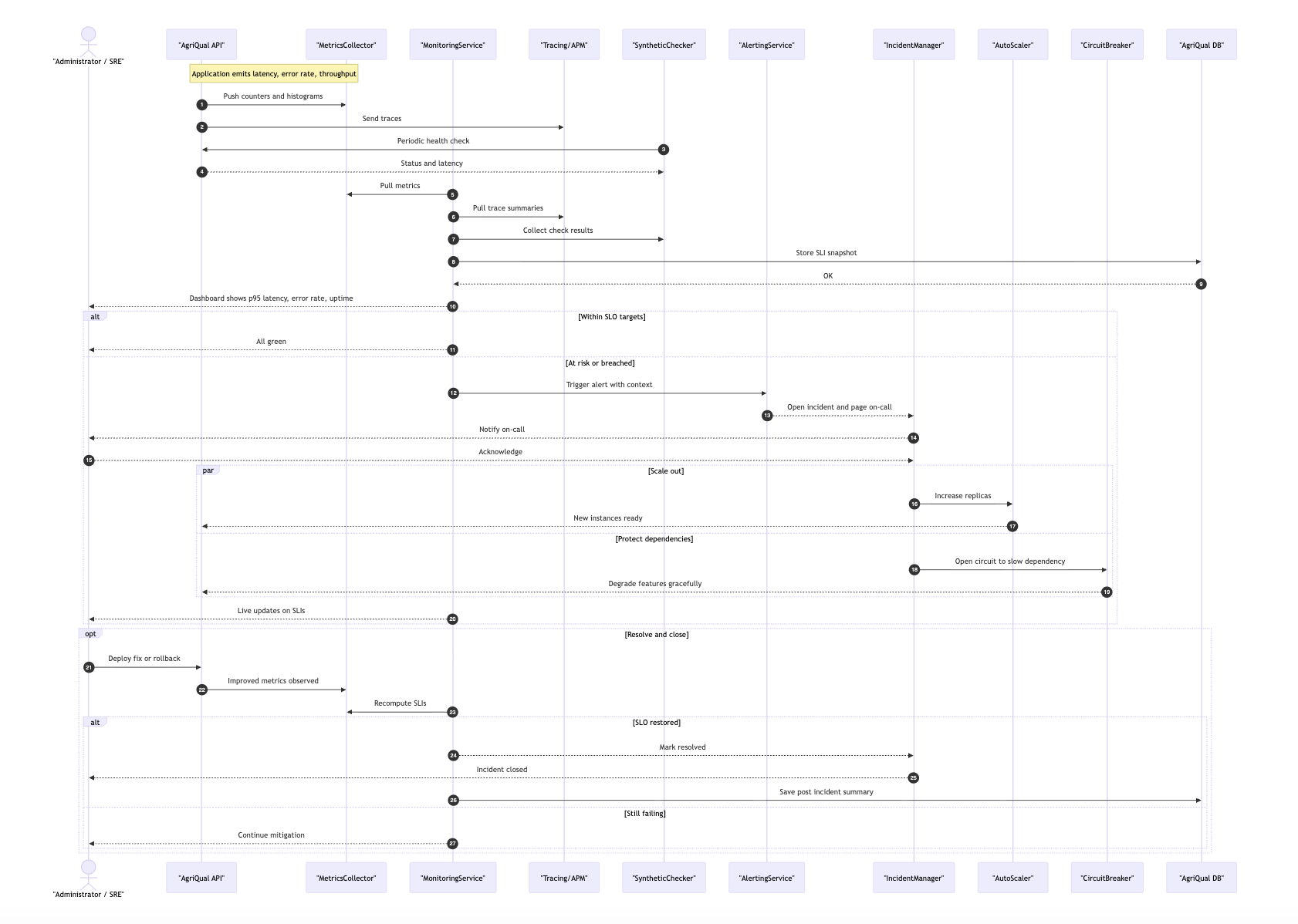
### Manage Knowledge Base

# 

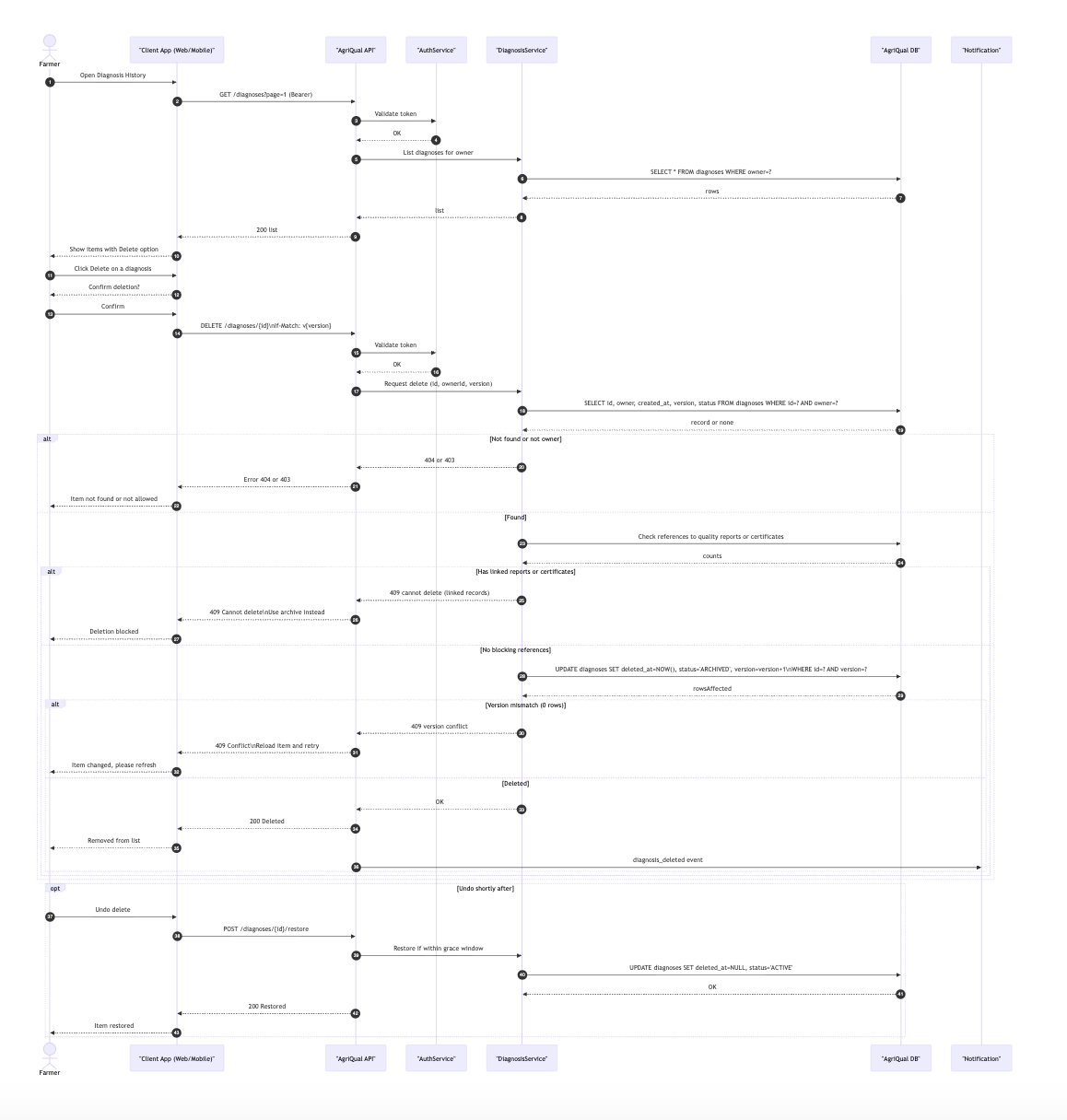
### Privacy & Consent Management



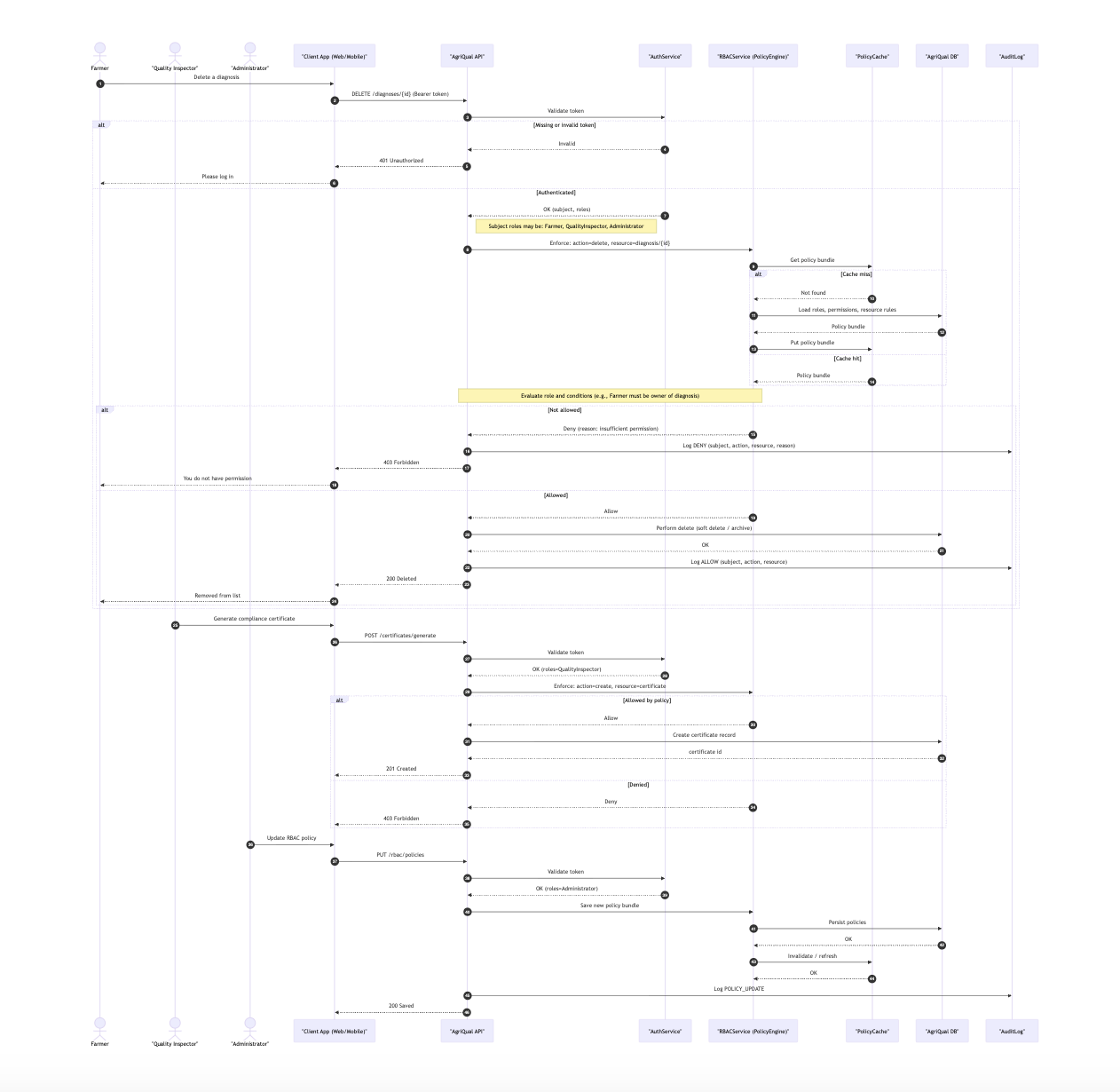
### Performance Guardrail (SLA Monitoring)



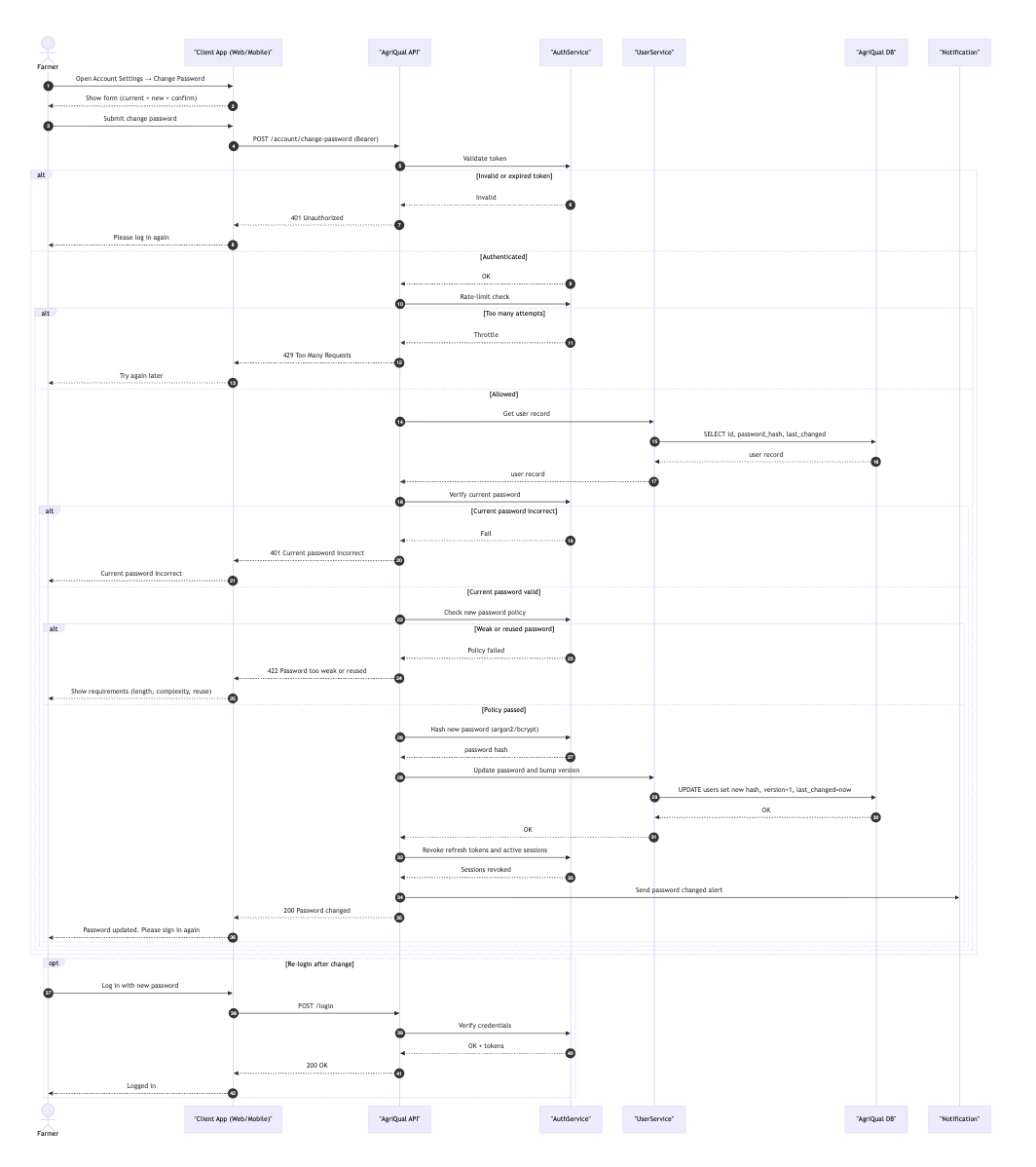
### Farmer Deletes Old Wheat Diagnosis



### Role Based Access Control (RBAC) Enforcement



### Farmer Changes Password



# 

# 

# State Diagrams

**Image Processing AI Agent**

## 

## 6.1 Diagram details

**Idle State** – Waiting for an image upload. Transition occurs when an image is uploaded.

**Image Preprocessing** – Resizes and normalizes the image. Moves to Image Analysis if successful; Error State if preprocessing fails.

**Image Analysis** – Detects wheat heads and extracts features. If heads are detected, move to Classification; if none, moves to No Detection.

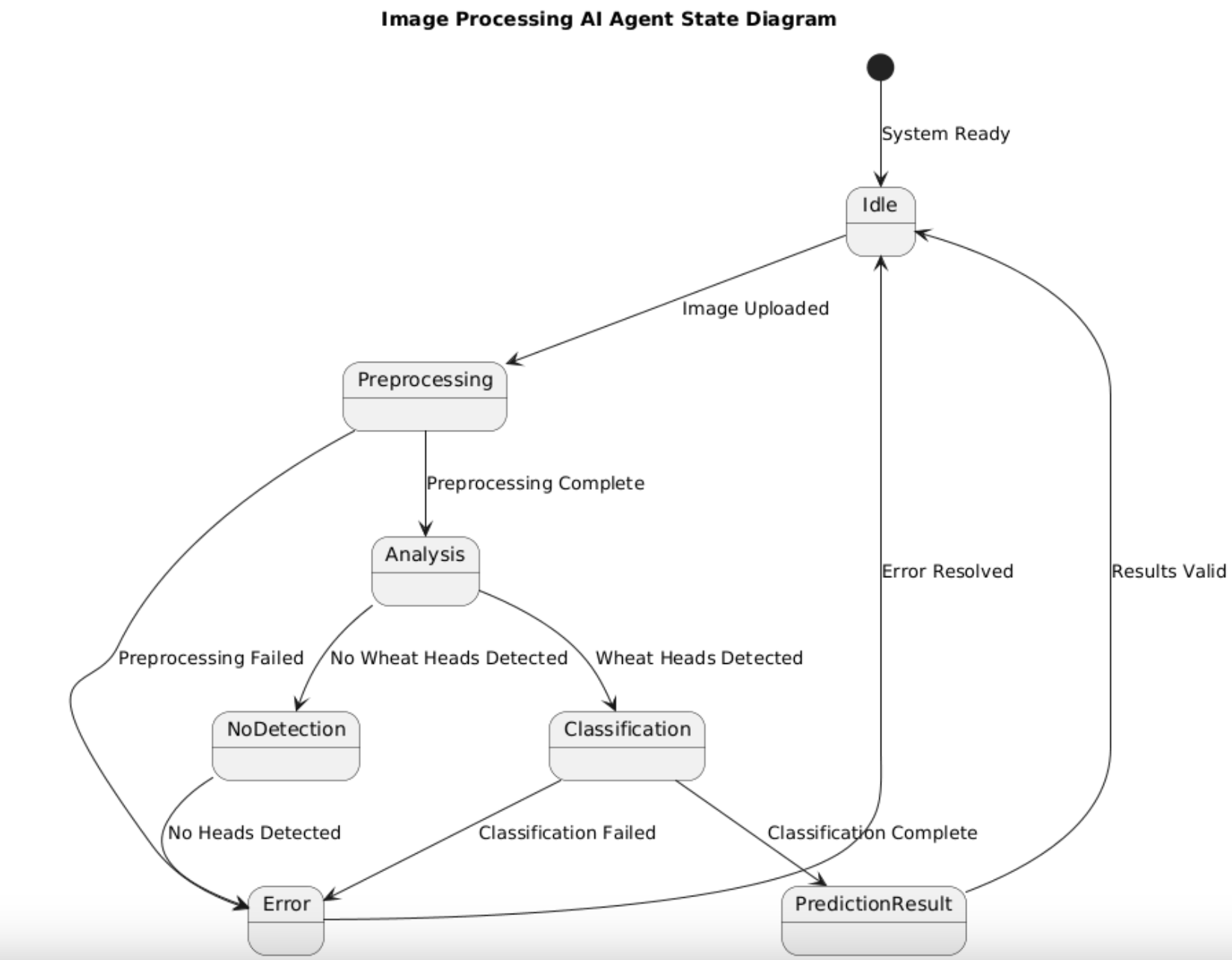
**Classification** – Classifies detected wheat heads (healthy, diseased, pest-infested). Moves to Prediction Result if successful; Error State if classification fails.

**Prediction Result** – Generates the final analysis results. Returns to Idle State if successful; Error State if results cannot be generated.

**No Detection** – Notifies the user that no wheat heads were detected, then returns to Idle State.

**Error State** – Handles failures during preprocessing, analysis, or classification. Returns to Idle State if resolved.

## 6.2 Diagram



**Climate Agent**

## 6.1 Diagram details

**Idle State** – Waiting for a weather data request from the Advisory Agent. Transition occurs when a request is received.

**Data Retrieval** – Fetches weather data from source. Moves to Data Validation if successful; Error State if retrieval fails.

**Data Validation** – Checks completeness and accuracy of data. Moves to Data Processing if valid; Error State if invalid.

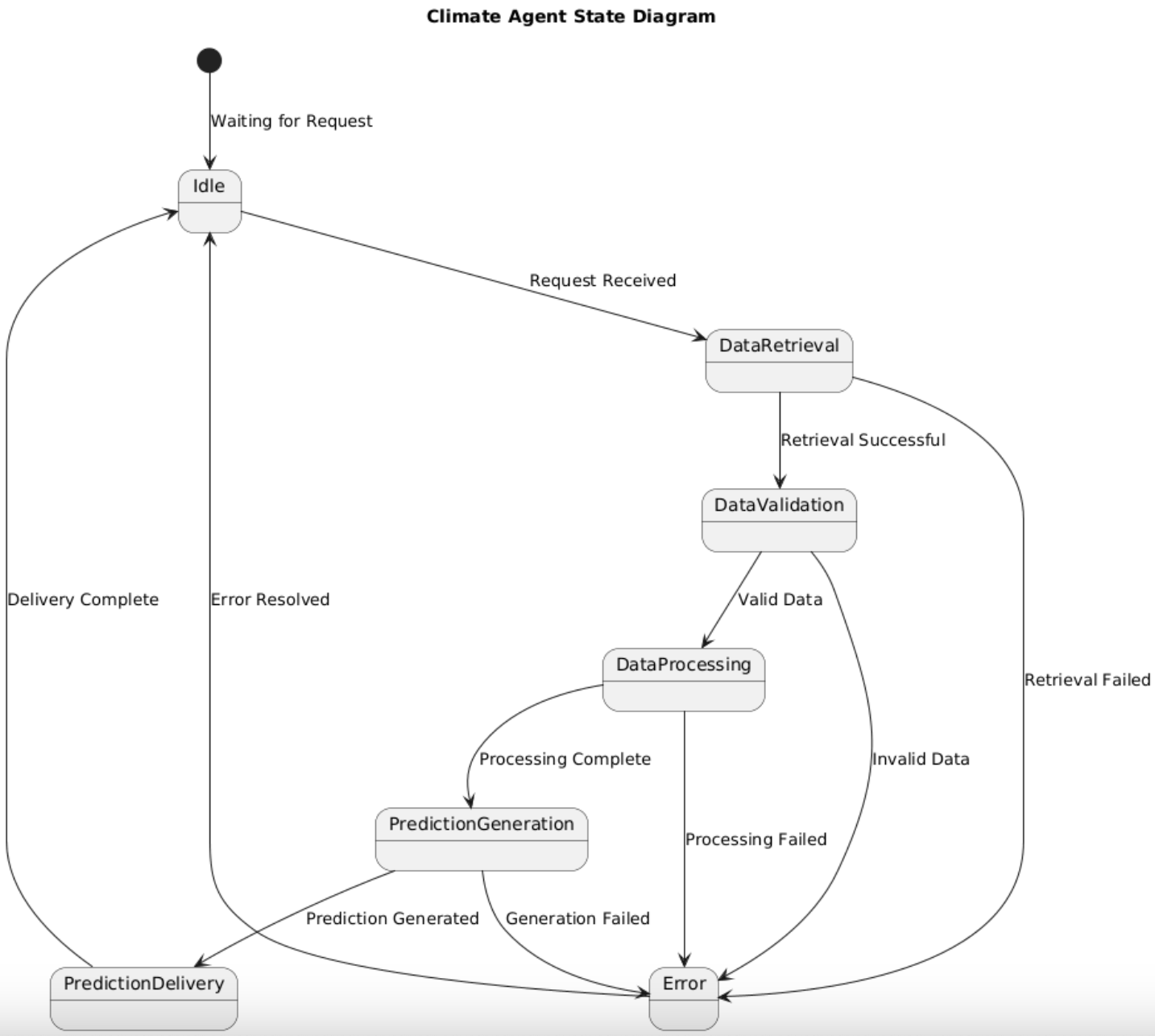
**Data Processing** – Aggregates and analyzes weather parameters. Moves to Prediction Generation if successful; Error State if processing fails.

**Prediction Generation** – Generates forecasts and climate risk assessments. Moves to Prediction Delivery if successful; Error State if generation fails.

**Prediction Delivery** – Sends forecasts and alerts to the Advisory Agent. Returns to Idle State once delivery is complete.

**Error State** – Handles failures during retrieval, validation, processing, or prediction. Returns to Idle State if resolved.

## 6.2 Diagram



# Data Requirements

|  |  |
| --- | --- |
|  |  |
| **Data Sources** | * https://zenodo.org/records/5092309 * https://github.com/hellodfan/GrainSet/blob/master/datasets/datalist.zip |
| **Data Requirements** | * The dataset consists of over 6000 images at 1024x1024 pixel resolution * more than 300,000 wheat heads * Training Datase**t**: 4000 images from Europe and Canada covering 7 measurement sessions * Test Dataset: 2000 images from North America (except Canada), Asia, Oceania, and Africa covering 17 measurement sessions. * Validation Dataset: The “public test” set from the 2020 Global Wheat Head Challenge is included for intermediate validation. * Bounding boxes must be complete; images without wheat heads should be clearly labeled as no\_box * Pre-processing Requirements: Images should be normalized to 1024x1024 pixels. Data augmentation (rotation, flipping, brightness adjustment) may be applied to increase model robustness. |
| **Model Requirements** | * Training Strategy: Convolutional Neural Network (CNN) models for object detection * Model should be validated on the provided validation set to monitor accuracy * The detection model should aim for at least 80% precision and recall on the validation dataset. * Evaluation Metrics:Precision, Recall, and F1 Score for wheat head detection. |

# Non-functional Requirements / Quality Attributes

|  |  |
| --- | --- |
| **Sr#** | **Requirements** |
| 1 | The system should process and classify an uploaded image in less than 15sec |
| 2 | The system should support at least 100 concurrent users |
| 3 | The system should provide a confidence score for each prediction |
| 4 | The system should have a simple and intuitive interface where most user tasks are completed in 2-3 clicks. |
| 5 | The system should handle an increase in dataset size(adding new crops) |
| 6 | The system should provide clear error logs for troubleshooting |
| 7 | The system should retrieve weather data within 30 seconds of a request from the advisory agent to ensure timely recommendations for farming activities |
| 8 | Certificate generation should complete within 5 seconds for single quality assessments |
| 9 | Statistical reports should process 1000 assessment records within 10 seconds. |
| 10 | The system should support crop image uploads up to 10MB. |
| 11 | Notifications should be delivered within 10-20 seconds of system events. |
| 12 | Weather alerts must reach users within 5-10 minutes for severe weather warnings. |

# Security Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr#** | **Security Risks** | **Potential Losses** | **Controls** |
| 1 | Broken Access Control (e.g., unauthorized user accessing another farmer’s or inspector’s data) | Unauthorized data exposure, privacy violations, reputational damage, possible legal or regulatory penalties. | Apply strict role-based access control (RBAC), verify data ownership before showing or modifying records, and maintain audit logs of access decisions. |
| 2 | Weak or Broken Authentication | Account takeover, fraudulent use of the system, loss of trust if accounts are misused. | Enforce strong password rules, store passwords using hashing (e.g., bcrypt), enable multi-factor authentication (MFA), use session timeouts, and apply login rate-limiting. |
| 3 | Injection Attacks (SQL injection, malicious file input, command injection) | Database compromise, loss of sensitive data, system downtime. | Use parameterized queries, validate all user inputs (IDs, file names, etc.), limit file formats and sizes, and sanitize data before processing. |
| 4 | Data Poisoning (malicious data fed into the diagnosis or training pipeline) | Corrupted crop diagnosis results, wrong recommendations to farmers, potential crop damage, business credibility loss. | Use strict validation of uploaded images, keep a clean and verified training dataset, apply anomaly detection, and log unusual data submissions for review. |
| 5 | Abuse of External Requests (e.g., Weather API misuse, Server Side Request Forgery) | Unauthorized access to internal systems, service disruption, leakage of sensitive information. | Allow only pre-approved external domains, block internal IP ranges, enforce network isolation for API calls, and log all outbound requests. |

# Security Engineer

|  |  |
| --- | --- |
| **Name of the Security Engineer** | Syeda Umaima Hasan |

# Use of Generative AI

Generative AI was used to rephrase the content provided so that it was in an easy to understand and simplified text. It was also used to cross check definitions of state, class and sequence diagrams to ensure proper and correct diagrams were made and submitted.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Zarak Qadir Khan | Use cases and Data Requirements |
| Syeda Umaima Hasan | Sequence Diagrams and Security Requirements |
| Mishaal Usman | State Diagrams and Non Functional Requirements |
| Muhammad Walid Khan | Introduction and Class Diagrams |

# 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 | Umaima |
| 1. System Actors | Umaima |
| 1. Use Cases | Mishaal,Umaima,Walid |
| 1. Class Diagram | Zarak |
| 1. Sequence Diagrams | Walid |
| 1. State Diagrams | Zarak |
| 1. Data Requirements | Umaima |
| 1. Non Functional Requirements | Walid |
| 1. Security Requirements | Mishaal |
| 11. Use of Generative AI | Zarak |