TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE492 - TEST PLAN REPORT

Project Title

ALSApp (Agriculture and Livestock Support Application)

Project Supervisor

Venera ADANOVA

Project Juries

Ayşe Yasemin Seydim

Tansel Dökeroğlu

Project Members

Engin Samet Dede

Sevgi Dilay Demirci

Gizem Ünsal

17/04/2025

Table Of Contents

1.Introduction

1.1 Project Description

The goal of our project is to develop an application that will make tracking easier for farmers in a variety of ways. Furthermore, this platform combines all the necessary resources to improve livestock performance and farming activities. Users will be able to enter their animals' health data and receive relevant information, track their crops digitally for optimal efficiency, and track the growth and development of their crops and animals in real time using an easy-to-use interface that is favourable to both farmers and livestock. By consolidating all agricultural and livestock management needs onto a single platform, this system enables users to manage their operations in a more organized, efficient, and planned manner. It also provides instant updates on industry developments, helping users stay informed and responsive to industry changes. They will be aware of government support, mainly offered by the Ministry of Agriculture and Forestry, and benefit from funding opportunities. In addition, the following process can also be used by the Ministry of Agriculture and Forestry staff to keep track of the reliability and accuracy of support.

1.2 Report Description

This Test Plan Report outlines the strategy, scope, resources, and schedule for testing ALSApp (Agriculture and Livestock Support Application). The objective of this document is to define a comprehensive approach to verify that the system meets its functional and non-functional requirements. Building upon the previously defined High-Level Design, this report details the testing methodologies, test cases, expected outcomes, tools, and responsibilities involved in ensuring the application's quality, reliability, and performance throughout its development lifecycle.

2. Scope

In this section, the scope of testing activities of the ALSApp system are defined. The document provides the key features and the functionality the Quality Assurance (QA) team will verify to validate the reliability, usability, and fulfill the business requirements. The scope mainly encompasses core operational modules that reflect interaction with the end users and processing behind the scenes. Included are the features to be tested:

- **User Authentication and Authorization**

  These will include user registration, login, logout and user role based access control (RBAC). Farmers and Ministry officials will access levels, the proper session managements and security constraints to be tested in both.

- **Livestock and Crop Data Tracking**

  The function testing will verify if the application can add, edit, delete, and get the data of livestock and crop. The system will also be tested using some information that it needs to persist across sessions and devices.

- **Real-Time Weather Data Integration**

  Testing is also performed for its integration with the external weather APIs. Weather alerts will consider the user location, verified by QA, and will verify accuracy, frequency and reliability of such alerts, as well as an error handling of failed API response.

- **Government Support Listings and Applications**

  The scope of testing involves validation of the support programs fetched from the Ministry's database. The functional coverage will include the lists of supports, application for eligibility, status tracking and the rejection / approval workflows.

- **Notification Management System**

  With this feature your users will receive notifications at right time and right place. We will QA the notification generation, delivery, and interface behavior of weather, support update, or account alert notifications.

3. Quality Objectives

The aim of the Quality Assurance (QA) process of ALSApp is not only about getting the application to work properly, but to provide a seamless and also secure, reliable and easy to use experience across all ALSApp supported modules and platforms. This section details the testing strategy both as primary and secondary quality goals that will govern the test strategy tool.

3.1 Primary Objectives

The key task of this phase is to test the functioning part of the application to validate the functional integrity, and find out and fix any issues that prevent a user from interacting with the system or reduce system's reliability. Core of the system's requirements and development specifications, and these goals directly match.

- **Ensure the system fulfills all functional requirements**

  The testing will prove that all the functional requirements defined in the Software Requirements Specification (SRS) and Low-Level Design (LLD) documents are correctly implemented and work as expected at normal and boundary conditions.

- **Maintain usability, stability, and performance across all modules**
  The application should be steady across farmers' and ministry personnel's roles; in market, responsive interface, effective flow of data, and particularly load resilient trades (especially in weather and support modules).

- **Deliver a bug-free experience in all core workflows**

  Before a production release, all major user journeys must be free of blocking or critical defects as they pertain to user authentication, data tracking, government support application and notification handling.

3.2 Secondary Objectives

The project's secondary aims center on making the system durable and enabling both tracing of test results and ensuring stakeholder contentment. Such objectives help projects maintain their quality path by enabling continuous verification and validation procedures.

- **Detecting and reporting defects early**

  Unit and integration testing defects provide the opportunity for early detection during which rework and release time performance is significantly minimized. This objective will use exploratory testing and risk-based prioritization.

- **Maintaining traceability between requirements and tests**

  A Requirements Traceability Matrix (RTM) will chart every test case into one or more requirements. It helps to ensure we cover 100 percent, and it also provides impact analysis if something happens to the scope or the design.

- **Ensuring successful UAT (User Acceptance Testing) sign-off by stakeholders**

  Key end users and stakeholders will be final validated. With an understanding that ALSApp meets real world needs and that ALSApp is ready for deployment in production environments, sign off in UAT will confirm the project is ready for sign off.

4. Test Strategy

Overall guidelines of how to apply all verification and validation activities during the ALSApp testing lifecycle, the test strategy specifies. Objective, assumptions, methodologies and test levels of the software product to be made sure that the software product has been achieved its purpose as well as reliable and a high quality of end users.

4.1 Test Objectives

Basically, the process of verification is to verify that the implemented features of the system are as per the need under normal as well as exceptional scenarios. Specific objectives include:

- The test observes if the behavior of the system is verified with the documented requirements. To ensure traceability and coverage, the Software Requirements Specification (SRS) and Low-Level Design (LLD) will be derived from all test scenarios.

- Confirm integration between the frontend, backend and third parties. All the ways of communicating between UI components, internal APIs and external systems such as OpenWeatherMap and government support APIs will be thoroughly validated.

- Maintain data's accuracy, security and traceability. Validation tests will prove that data entered or read into ALSApp are processed correctly, securely and traceable in the application workflow.

4.2 Test Assumptions

The testing strategy further assumes that such environmental and development conditions were achieved before actually performing the test. These assumptions will help the QA team continue to work effectively.

- All application modules are considered stable and feature complete. First, the freeze of the functionality is necessary before entering the test cycle so as not to risk scope creep or test invalidation**.**
- The required APIs (e.g: weather and support systems) are expected to be live and available. During testing periods, if external dependencies are not operational and stable to some degree of accuracy, validation of them will not be accurate.

- When there is a defect or QA is unblocked, Developers are available to resolve. It is assumed to be a loop between QA and development teams' handles, which takes care of the bug triage and resolution in a short time.

4.3 Data Approach

To test ALSApp, a synthetic and real-world data combination will be used to simulate realistic user interactions:

- For entities such as livestock records, crop entries, user profiles and support applications, synthetic test data will be generated to make sure that the result is repeatable and safe.

- Real time data, behavior under dynamic conditions will be evaluated in terms of live API responses from OpenWeatherMap.

- Data consistency, integrity and correctness through several modules shall be ensured through manual and automated validations.

4.4 Levels of Testing

In order to assure complete verification at each level of development, ALSApp will be tested in various levels of testing which are explained below:

- **Unit Testing:**

  Tools used by the developers to ensure each component and each method have been conducted properly and can work in isolation.

- **Integration Testing:**

  Integration testing mainly deals with interactions between front-end components, back-end services, and databases, including third-party API calls.

- **System Testing:**

  The System Testing is carried out by testing the entire system's behavior end to end and simulating actual user scenarios and workflows among modules.

- **User Acceptance Testing (UAT):**

  Farmers and from ministry representatives conduct real world validation to ensure that ALSApp meets business needs and can be deployed.

4.5 Functional Testing

Functional testing makes sure that all the features are working as per the defined requirements.

Test Coverage Includes:

- User login and authentication workflows
- User profile creation and management
- Animal and crop registration and editing.
- Retrieval and real-time updates of weather dashboard
- Support from submission to where form support from government, eligibility checks and tracking
- Notification delivery for critical system events

4.6 Regression Testing

Regression testing makes sure that previously existing features got along with the changes made in the system.

Scope:

- Defect resolutions and system upgrades are to be executed after major releases
- No unintended side effects are introduced into any previously tested functionality.
- In the case of high frequent areas such as login and dashboard components, automated regression suites can be used.

4.7 User Acceptance Testing (UAT)

The final validation of readiness of ALSApp is done by UAT involving real stakeholders in a near production environment.

- Participants:
  Specific test scenarios will be given to ministry staff and farmers who are picked for pilot testing to execute test scenarios and report back the feedback.

- Focus Areas:

1. Usability, clarity, and responsiveness of the interface
2. Opportunities for support information and eligibility flows to be more accessible.
3. Accuracy of localized weather alerts
4. Seamless user journey from login to submission and notification

- Goal:

Get stakeholder formal approval and sign off before launch.

5.Roles and Responsibilities

This section describes the specific roles given to each of the team members who are part of the ALSApp quality assurance process. This distributes all the responsibilities to be held accountable and to maintain a smooth testing workflow between the development and QA teams. There is an association between a role and important testing activities such as planning, execution, defect management, and validation of releasing. Below is a table describing the various roles, the members of the team that has been allocated to the function, and responsibilities of the function in the context of the project.

Role	Team Member(s)	Responsibilities
Project Manager	**Sevgi Dilay Demirci**	Coordinating schedules and acts as a primary contact between developer and QA teams**.**
QA Lead	**Gizem Ünsal**	Drives the testing process including planning, organizing, mocks, reviewing, bug triaging and reporting.
QA Engineer	**Engin Samet Dede**	Works on design, prioritizing bugs, retests fixes, and makes sure he is getting to the required boundary.
Developer	**All Team Members**	Bug fixes, assists QA in bug investigation, ensures stable builds are delivered.

6. Execution Strategy

Execution strategy tells how, when the testing begins, what are the conditions by which the testing is marked complete and what to do with the defects during deployment. This makes it a process to run QA operations in a structured and traceable manner from the start of testing to closure.

6.1 Entry Criteria

There are simply some preconditions as before which the QA must fulfill before the testing can start and can perform effective and accurate verification activity. These entry criteria include:

- **Fully Deployed Development Environment:**
- The application should be deployed and is accessible for testing the latest version.

- **Frontend and Backend Integration Completed:**
  The end product (i.e.: the core modules and APIs (e.g.: weather data, government support), must work and be successfully integrated.
- **Test Cases and Test Data Prepared:**
  Then, all test cases are reviewed and completed and synthetic or real data is populated where needed.

- **QA Team Access and Documentation Availability:**

  Credentials and all relevant tech documents (resources, requirements) should be available to the QA team; and the QA team needs to have full access to the actual testing environment.

6.2 Exit Criteria

When the application meets such conditions that show a stable, acceptable version of this product it is the end of the QA process. Exit criteria include:

- **All High-Severity Defects Resolved:**

  Severity 1 or 2 issues must be fixed, retested and verified before passing the test phase.

- **Minimum 95% Test Case Pass Rate:**
  To make core functionality reliable, a high level of test coverage and success is needed**.**

- **User Acceptance Testing Feedback Incorporated:**
  UAT of the mobile application, must also involve feedback from pilot users (farmers and ministry staff) and it must be evaluated and applied where required.

- **Approval of Staging Deployment:**

  The application must be formally approved for deployment to the staging or pre-production environment by the QA Lead and Project Manager.

6.3 Validation and Defect Management

Bugs are reported, tracked, prioritised and resolved actually and this is referred to as Defect Management. There will be a structured defect lifecycle and communication routine followed by the ALSApp QA team to resolve issues and communicate with the defects.

- **Defects Logged in GitHub:**

  All bugs are reported via an agreed defect tracking tool if there is one, or the GitHub Issues. Each issue includes step by step reproduction steps, screenshots (as applicable) and expected vs. actual behavior.

- **Severity and Priority Assignment:**
  All defined severity and priority levels are used to categorize each bug (see Section 6.3.2). It aids in making the decision as to the sequence and needs to be rectified.

- **Retesting After Resolution:**
  When a bug has been marked fixed by developers, the QA team then does focused retesting to ensure a bug has been resolved.

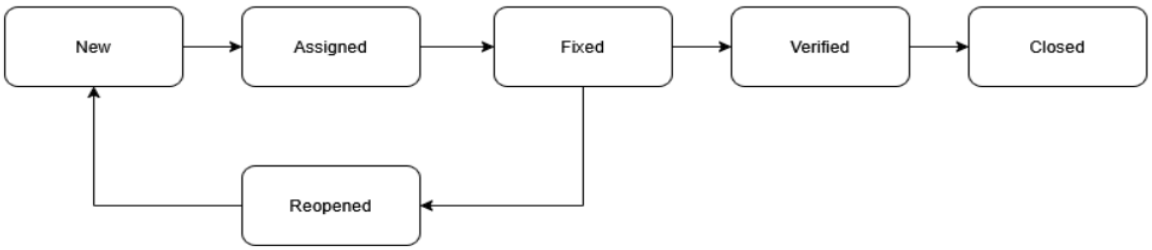
- **Daily Synchronization with Developers:**

  QA teams sync meetings with the development teams are held daily and QA and development teams discuss the bugs status, blockers, and triage decisions during the daily stand-ups or syncing meetings.,

6.3.1 Bug Life Cycle

The term bug life cycle refers to the life which a defect goes through from time of discovery to closure. Transparent bug status tracking and communication is handled using a standard workflow that ALSApp adopted.

Basic Bug Status Flow:



- **New:** QA discovers the bug which goes into their systems for tracking purposes.
- **Assigned:** A developer obtains the task through review of the reported issue.
- **Fixed:** The developer handles the issue and sends it to verification after resolving the problem.
- **Verified:** The system bug test outcome is confirmed by QA technicians.
- **Closed:** The official bug termination occurs at this stage.
- **Reopened:** Such bugs become eligible for reassignment following their fix because they still exist.

The detailed description of the Bug Life Cycle with transition steps and responsibilities appears in Section 8.2 Bug Life Cycle.

6.3.2 Severity and Priority Definitions

The bugs will be ranked based on how severe the problem is and how critical it is to be fixed. These labels guarantee the release of the work on the quality standards required, and also to help the team of development prioritizing their work.

Severity Level Description:

- **Severity 1:** Critical (like the exapmles as app crash, data loss, security breach). Excludes testing or usage totally.
- **Severity 2:** High - Major functional failure with broken fundamental functionality, but the system remains functioning.
- **Severity 3:** Medium (minor logic problems, efficiency delays, usability challenges).
- **Severity 4:** Low - embraces the concepts of Cosmetic/UI concerns, typos, slight visual discrepancies.

Priority Level Description:

- **Priority 1:** Fix a blocker for release or important flow.
- **Priority 2:** Quickly resolve high business impact issues.
- **Priority 3:** Can Fix - Not blocking the working process, but beneficial if addressed.
- **Priority 4:** Low - Optional or enhancement-level component for after release.

7. Roles and Responsibilities

7.1 QA Team

Testers perform multiple tasks, which include developing and running test cases and identifying along with reporting functional issues, followed by retesting solved problems.

- Write and execute test cases
- Report and track bugs
- Perform regression and retesting

7.2 Development Team

The development team handles the task of bug correction, supports testing activities, and provides environmental setup.

- Fix assigned defects
- Support QA with debugging and environment setup

7.3 UAT Participants

Ministry staff along with farmers function as real users to validate products that operate in real-world operational settings.

- Ministry officials and farmers
- Users will test the system as well as give feedback during the UAT phase.

8. Test Strategy

8.1 QA Role in the Test Process

The Quality Assurance (QA) team is responsible for assuring that the ALSApp system fulfills all functional and non-functional requirements both for farmers and staff of ministry. The QA process is tightly related to Agile development iterations and the key responsibilities involved are the following:

1)Requirement Analysis:

The Software Requirements Specification (SRS), Use Cases and Low Level Design documents are reviewed by QA to be sure they are fully understood by them. It gives special attention to modules like crop/livestock tracking, ministry support integration and real time weather data.

2)Test Case Design:

The continuity is all in QA where there is proper preparation of test cases covering all positive, negative and edge scenarios of each module. It uses exploratory testing techniques on which unexpected issues are identified using the domain knowledge.

3)Traceability Matrix (RTM):

Full requirement for the coverage is ensured by creating an RTM. Functional requirements are mapped to each test case.

4)Test Execution:

Core workflows such as user registration, product data entry, weather alert retrieval, support filing, and its end-to-end execution are run manually. Pass / fail determination is based on what is actually recorded against what is expected.

5)Defect Logging and Reporting:

Bugs detected are logged with reproducibility steps, severities and screenshots. Shared documentation is used by the QA team to track issues.

6)Regression and Retesting:

After fixing the defects, QA does a retesting and regression testing of all the modules that are affected only in order to test the system stability.

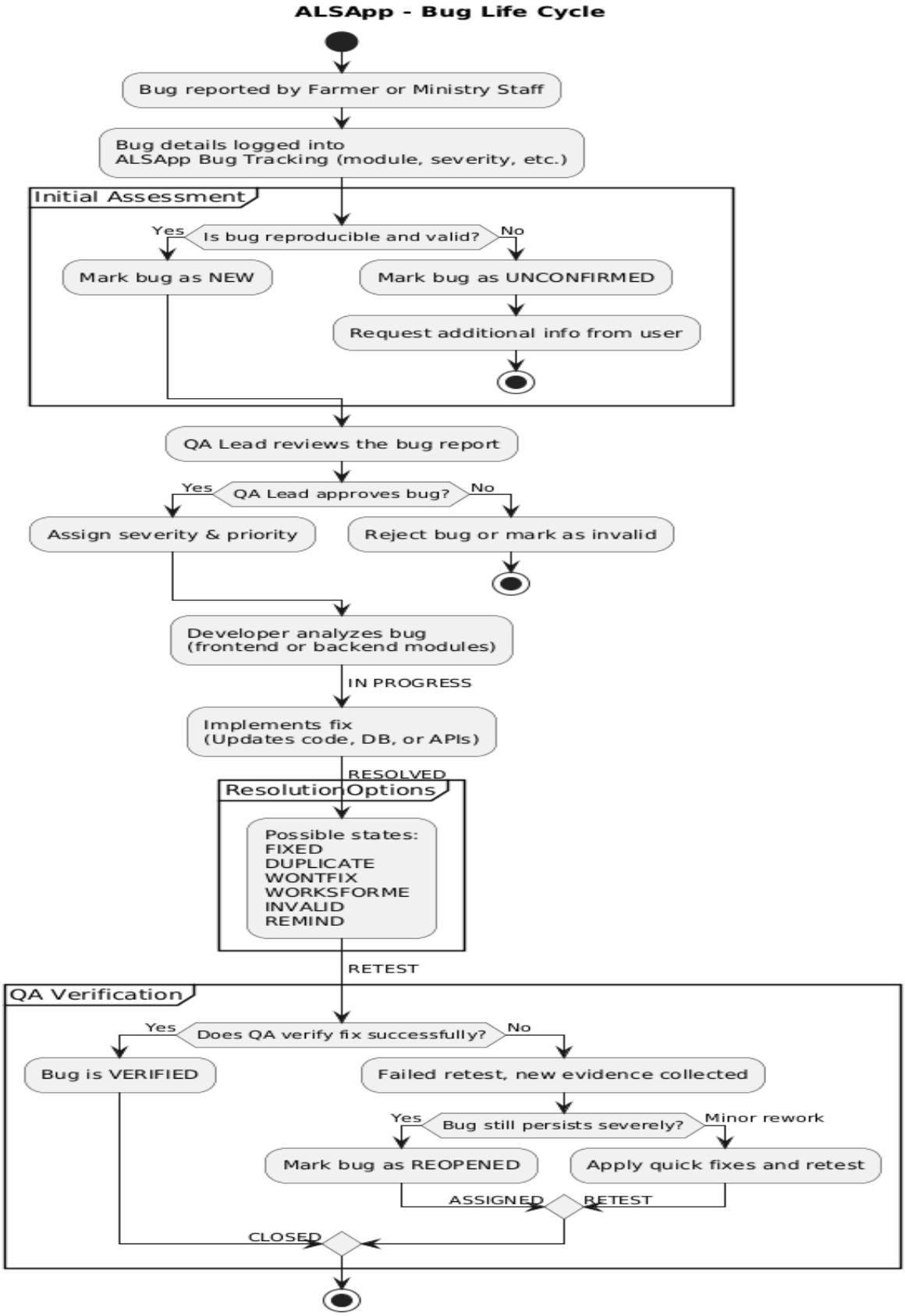
7)Test Data Preparation: There are various data sets generated for different user roles (farmer, ministry employee), product types, locations, etc. and weather conditions.

8)Acceptance Testing: Qa provides test scripts and observations for user acceptance testing (UAT). Mock users representative of real farmers and ministry officials, will be used in UAT planned to be done with.

8.2 Bug Life Cycle

Effective bug management is very important in software quality assurance for stable and reliable applications. Bug Life Cycle is a sequence of steps a software defect passes through the time it gets identified to the time it is finally closed after making it through all the stages. The stages are set up to monitor the status of the defect, enhance the communication line between the QA and development teams, and make sure that the defect is rectified in an order and timely manner. In the following table and diagram below, the status flow and who is responsible for each step has been depicted in the route to ALSApp bug resolution.

Status	Description
New	QA team discovers and reports a new bug.
Assigned	QA Lead assigns the bug to the concerned developer.
In Progress	The developer starts working on the bug fix.
Fixed	The developer labels the problem as fixed and informs QA.
Retest	QA retests the fix in the test environment to confirm it's fixed.
Closed	QA has validated the problem, and it is now officially closed.
Reopened	If the problem persists, the bug is reopened and assigned to a developer.


ALSApp - Bug Life Cycle
Bug reported by Farmer or Ministry Staff
Bug details logged into
ALSApp Bug Tracking (module, severity, etc.)
Initial Assessment
Yes
Is bug reproducible and valid?
No
Mark bug as NEW
Mark bug as UNCONFIRMED
Request additional info from user
QA Lead reviews the bug report
Yes
QA Lead approves bug?
No
Assign severity & priority
Reject bug or mark as invalid
Developer analyzes bug
(frontend or backend modules)
IN PROGRESS
Implements fix
(Updates code, DB, or APIs)
RESOLVED
ResolutionOptions
Possible states:
FIXED
DUPLICATE
WONTFIX
WORKSFORME
INVALID
REMIND
RETEST
QA Verification
Yes
Does QA verify fix successfully?
No
Bug is VERIFIED
Failed retest, new evidence collected
Yes
Bug still persists severely?
Minor rework
Mark bug as REOPENED
Apply quick fixes and retest
ASSIGNED
RETEST
CLOSED

Overall Process

1. **Bug Report**
    - Start: The process starts off when a bug is reported by a user, which in case of ALSApp, can be a Farmer or a Ministry Employee.
    - Details Logged: An entry has been made in ALSApp Bug Tracking System regarding the details logged (eg. Module affected, Severity, Description).

Partition 1: Initial Assessment

- Reproducibility & Validity Check:

    - If the bug is supported by a sufficient amount of descriptive information, the system (ie the QA team) will check if it is reproducible and if it is valid.

    - In that case, the bug is marked in NEW status, meaning this is a genuine issue that can be reproduced.

    - The bug is labeled as UNCONFIRMED, and more details are requested from the user. If no: The bug is marked as UNCONFIRMED. This process stops here until further clarification is given.

Partition 2: QA Lead Review

- **QA Review:**
    - In case the issue is genuine, the QA lead examines the bug report.
    - Approval:
        - If the bug is approved, the QA Lead assigns a priority and severity to it.
        - The bug is then handed off to the appropriate developer in the bug's transition from the BUGS stage to the ASSIGNED one.
    - Rejection:
        - If the bug does not fulfill the criteria, it is rejected or marked as invalid, and the process ends.

Partition 3: Developer Processing

- **Bug Analysis and Fix**
    - When assigned by the developer, the developer analyzes the bug in the light of ALSApp's prime modules, which covers crop/livestock tracking, weather data or integration with ministry support.
    - In Progress:
        - The developer starts fixing by updating code and APIs or making appropriate changes in the database.
    - After solving the issue, the status becomes RESOLVED.
    - The fix is submitted for review at this point.

Partition 4: Resolution Options

- **Resolution Options:**

    - Right after the bug is marked as RESOLVED, a dedicated partition shows the various possible outcomes or resolution types:

        - FIXED: The issue is fully fixed.

        - DUPLICATE: The bug is a repeat of an already reported issue.

        - WONTFIX: It's decided that the bug will not be fixed (perhaps because it is deemed acceptable or lower priority).

        - WORKSFORME: The system or developer reports that the problem cannot be replicated on their end.

        - The project policy may allow for the addition of additional alternatives (such REMIND or INVALID).

    - Before proceeding to retesting, this division shows potential states and stresses the decision-making component of bug resolution.

Partition 5: QA Verification (Retest)

- **Retesting the Fix:**
    - Once the resolution option is selected and the updated build is deployed in the test environment, QA performs RETEST.
- **Verification Outcome:**
    - Successful Retest:
        - If QA verifies that the fix works correctly within the respective module (be it crop/livestock tracking, weather, or ministry support), the bug is moved to the VERIFIED state and then CLOSED.
    - Failed Retest:
        - If the bug still exists, QA documents the failure.
        - Reopen Option:
            - If the bug persists significantly, it is marked as REOPENED and reassigned to the developer for further work.
        - Minor Issues:
            - In cases where only minor corrections are needed, small adjustments might be applied and the process loops back to retesting.

Final Outcome

- Closed:
    - When the repair is properly confirmed, the bug is tagged as CLOSED, and the procedure is complete.
- Reopen and Repeat:
    - If the defect remains unresolved or fresh evidence suggests that it continues, it will be REOPENED and returned to the ASSIGNED phase for another cycle.

8.3 Testing Types

Three types of testing will be executed for ALSApp:

1)Black Box Testing:

It examines module behavior while keeping the system code internals unknown to testers.

2)GUI Testing:

Ensures alignment with UI mockups, form validations, and user flow consistency. System integration testing focuses on the verification of component communications specifically targeting the weather API and ministry support API and database services. The system gets tested for its compliance with requirements through Functional Testing. System testing allows evaluation of full application compliance after completing its integration with the entire system.

3)Performance Testing:

Includes response time testing (e.g., under 2 seconds per weather request), and simultaneous user access scenarios. UAT (User Acceptance Testing) functions as the termination step for inspecting actual operational usability along with user contentment.

8.4 Bug Severity and Priority Definition

They will be divided between severity (impact) and priority (fix urgency):

8.4.1. Severity Levels

Critical (1): App crash, data loss, or blocked core functionalities like login or data entry. Major features broken (e.g., weather not updating, but workarounds are possible).

Medium (3): Minor feature malfunction (e.g., incorrect alert text).

Low (4): Cosmetic UI issues, typos, or layout inconsistencies.

8.4.2. Priority Levels

Must (2): Must be done before being released.

Fix (2): Should be fixed as soon as possible.

Have Time (3): Allow time, does not block release.

Enhancement (4): Priority may be low, but this functionality will be nice-to-have for future iterations.

Weekly status and prioritization discussions will occur in bug triage meetings.

9. Resource and Environment Needs

This section describes the tools and environments needed in addition to the system configuration for the ALSApp system to be fully tested. The resources are picked to handle client server architecture through mobile and web platforms with the capability for manual and automated testing.

9.1 Testing Tools

This section outlines the tools and platforms that are used in the entire process of testing the ALSApp. Each tool is picked depending on its capability to be used for defined particular testing task from test case concerns to its execution to defect tracking and reporting. Efficiency, traceability and accurate documentation in the test cycle is a number of manual and automatic solutions that are provided.

Purpose	**Tool/Platform**
Test Case Design & Tracking	Microsoft Excel, Google Sheets
Test Case Execution	Manual Testing, Selenium (for automation)
Defect Logging & Tracking	Microsoft Word, Trello or GitHub Issues
Test Reporting	PDF Reports (generated manually)
API Testing	Postman
Code Version Control	Git (via GitHub)
Mind Mapping & Test Planning	XMind, Draw.io

9.2 Configuration Management

Thus, in order to enforce version control and consistency across dev and QA teams,

Tracked in Google Drive for Documents Versioning or in the SVN.

Code Changes: Code changes are managed through Git and GitHub, in support of saving changes to individual files, maintaining various branches for development and testing, etc.

Tag and document build versions in order to be able to trace them and follow them through each testing phase.

9.3 Test Environment

To test the cross-platform compatibility of ALSApp, the App will be tested on various platforms and device configurations.

Supported Mobile Devices:

Android: Samsung Galaxy series, Google Pixel, Xiaomi devices

iOS: iPhone 7 and newer, iPad 5th Gen and newer

Test Server Configuration:

This would also be deployed on Firebase/Node.js or Spring Boot (based on what the final hosting is).

Uses Cloud Firestore or MySQL as Database (according to what is described in LLD specs). Weather API and mock Ministry Support API are connected to API Services.

Other than that, every test environment has to be refreshed before each test iteration so that test runs are clean and results are accurate.

10. Test Schedule

The following schedule presents the timeline of all scheduled testing activities throughout the ALSApp project. Each task has a well-defined start date, end date, effort and notes, which are notes to let you know highlights of what to review on that task. It follows an iterative testing schedule to progressively develop its verification while still being synchronous to the delivery milestones.

It also covers test planning, design, testing, and validation, functional as well as non-functional testing, fully covering the complete timeline of them. Environment setup, user acceptance testing and performance testing are strategically planned to fit build readiness and stakeholder feedback. A test cycle is done to ensure all major defects are identified and fixed early and the product becomes stable and deployable by the end of the test cycle.

Task Name	Start Date	End Date	Effort	Notes
Test Planning & Strategy Finalization	Apr 15, 2025	Apr 17, 2025	3 days	Review documents, finalize test objectives
Requirement & Use Case Review	Apr 15, 2025	Apr 18, 2025	4 days	Review Analysis & LLD Reports
Test Case Design	Apr 18, 2025	Apr 21, 2025	4 days	Write manual test cases
RTM Creation & Peer Review	Apr 20, 2025	Apr 22, 2025	3 days	Map test cases to requirements
Test Environment Setup	Apr 19, 2025	Apr 20, 2025	2 days	Deploy app to QA env
Functional Testing – Iteration 1	Apr 22, 2025	Apr 25, 2025	4 days	Test login, registration, product entry
Functional Testing – Iteration 2	Apr 26, 2025	Apr 29, 2025	4 days	Test weather, support, notifications
Bug Fixing & Retesting	Apr 30, 2025	May 1, 2025	2 days	Retest after bug fixes
Integration & System Testing	May 2, 2025	May 3, 2025	2 days	End-to-end validation
Regression Testing	May 4, 2025	May 5, 2025	2 days	Verify no new issues
User Acceptance Testing	May 6, 2025	May 7, 2025	2 days	Test with simulated real users
Performance Testing	May 8, 2025	May 8, 2025	1 day	Load testing
Final Bug Fixes & Validation	May 9, 2025	May 10, 2025	2 days	Last validation cycle
Release to Staging	May 11, 2025	May 11, 2025	1 day	Deploy to final test env
Report Submission & Sign-off	May 12, 2025	May 12, 2025	1 day	Delivery of the QA report

11. Glossary

ALSApp: Agriculture and Livestock Support Application, a digital platform for farmers and livestock breeders.

Application: A software program designed to perform specific user tasks, often used on mobile devices or computers.

Livestock: Domesticated animals raised in agriculture for food, fiber, or labor. Examples include cattle, sheep, and poultry.

Real-Time: Data or information that is updated and available immediately as events occur.

Interface: The point of interaction between the user and the application, often designed to be user-friendly and intuitive.

Crop: Cultivated plants grown on a large scale for food, fiber, or other commercial purposes. Examples include wheat, corn, and rice.

Crop Tracking: The process of monitoring and managing crop growth, health, and yield using digital tools.

Sequence: The specific order in which tasks, processes, or events occur within a system or workflow.

Stimulus: An external factor or input that triggers a response in the system, such as a user action or environmental change.

Flagging: The act of marking or identifying data, issues, or conditions that require attention or action, such as potential errors or anomalies. API (Application Programming Interface): A set of rules that allows different software components to communicate. ALSApp uses APIs for weather data and government support services.

Bug Lifecycle: The process of software defect follows from discovery to closure, including statuses such as New, Assigned, Fixed, Verified, Closed, or Reopened.

Defect Severity: Indicates the impact a bug has on the functionality of the software (e.g., Critical, High, Medium, Low).

Defect Priority: Defines how soon a defect should be fixed, regardless of its severity. Priorities range from P1 (urgent) to P4 (low).

Functional Testing: A type of testing that checks if the application meets the defined requirements by testing features like login, data tracking, and notifications.

Integration Testing: Testing performed to verify the interaction between different modules or systems (e.g., frontend with backend and APIs).

Low-Level Design (LLD): A detailed design document that specifies how each component of the software system should function.

Regression Testing: A type of testing to confirm that a recent code change has not adversely affected existing functionalities.

Requirements Traceability Matrix (RTM): A document that maps each test case to its corresponding requirement, ensuring full test coverage.

Severity vs Priority: Severity relates to how badly a defect affects the software. Priority relates to how soon it should be fixed.

System Testing: End-to-end testing of the entire application to ensure it meets requirements and behaves as expected in real-world scenarios.

Test Case: A set of conditions or variables used to determine if a system functions correctly.

User Acceptance Testing (UAT): The final phase of testing where real users validate the system before it goes live.

Verification vs Validation: Verification checks if the product is built correctly; validation checks if the right product is built (meets user needs).

Traceability: Ensuring each requirement is tested and accounted for in the testing process using RTM.

12. References

[1] BrowserStack, "Entry and Exit Criteria in Software Testing," *BrowserStack*, [Online]. Available: https://www.browserstack.com/guide/entry-and-exit-criteria-in-software-testing. [Accessed: Apr. 2, 2025].

[2] BugBug, "Bug Life Cycle in Software Testing," *BugBug.io*, [Online]. Available: https://bugbug.io/blog/software-testing/bug-life-cycle/. [Accessed: Apr. 2, 2025].

[3] Agile Alliance, "Agile 101," *AgileAlliance.org*, [Online]. Available: https://www.agilealliance.org/agile101. [Accessed: Apr. 5, 2025].

[4] IBM, "What is an API?," *IBM Cloud Learn*, [Online]. Available: https://www.ibm.com/cloud/learn/api. [Accessed: Apr. 6, 2025].

[5] BugBug, "Bug Life Cycle in Software Testing," *BugBug.io*, [Online]. Available: https://bugbug.io/blog/software-testing/bug-life-cycle/. [Accessed: Apr. 6, 2025].

[6] BrowserStack, "Entry and Exit Criteria in Software Testing," *BrowserStack*, [Online]. Available: https://www.browserstack.com/guide/entry-and-exit-criteria-in-software-testing. [Accessed: Apr. 7, 2025].

[7] Guru99, "Defect Severity and Priority in Testing," *Guru99.com*, [Online]. Available: https://www.guru99.com/defect-severity-priority.html. [Accessed: Apr. 8, 2025].

[8] Software Testing Help, "Functional Testing," *SoftwareTestingHelp.com*, [Online]. Available: https://www.softwaretestinghelp.com/functional-testing/. [Accessed: Apr. 8, 2025].

[9] GeeksforGeeks, "Low Level Design (LLD) in Software Engineering," *GeeksforGeeks.org*, [Online]. Available: https://www.geeksforgeeks.org/low-level-design-llg-in-software-engineering/. [Accessed: Apr. 8, 2025].

[10] Software Testing Fundamentals, "Requirements Traceability Matrix (RTM)," *SoftwareTestingFundamentals.com*, [Online]. Available: https://softwaretestingfundamentals.com/requirements-traceability-matrix/. [Accessed: Apr. 8, 2025].

[11] ISTQB, "ISTQB Glossary," *ISTQB Glossary*, [Online]. Available: https://glossary.istqb.org/en. [Accessed: Apr. 9, 2025].

[12] Requirements.com, "What is Traceability in Software Testing?," *Requirements.com*, [Online]. Available: https://www.requirements.com/blog/what-is-traceability-in-software-testing/. [Accessed: Apr. 9, 2025].

[13] E. S. Dede, S. D. Demirci, and G. Ünsal, "ALSApp: Agriculture and Livestock Support Application – Analysis Report," Department of Computer Engineering, TED University, Ankara, Turkey, 2024.

[14] E. S. Dede, S. D. Demirci, and G. Ünsal, "ALSApp: Agriculture and Livestock Support Application – High Level Design Report," Department of Computer Engineering, TED University, Ankara, Turkey, 2024.

[15] E. S. Dede, S. D. Demirci, and G. Ünsal, "ALSApp: Agriculture and Livestock Support Application – Low Level Design Report," Department of Computer Engineering, TED University, Ankara, Turkey, 2025.

[16] California State University, Sacramento, "Use the Test Plan document to describe the testing approach and overall framework that will drive the testing of the project. Template Instructions," [Online]. Available: https://www.csus.edu/_internal/_documents. [Accessed: Apr. 14, 2025].

[17] StrongQA, "Test Plan Template 05," *StrongQA*, [Online]. Available: https://strongqa-production-assets.s3.amazonaws.com/uploads/document/doc/61/test-plan-template-05.pdf. [Accessed: Apr. 14, 2025].