Table of Contents

"Crazy Cats"

Date	Version	Description	Author	Reviewer	Approver
22.09	0.1	Creating a test plan	Prorochenko J		

1. INTRODUCTION

Welcome to the testing plan for the "Crazy Cats" mobile application! As the primary stakeholder, I am pleased to present the testing strategies for this application designed to ensure its effectiveness and reliability. "Crazy Cats" is a mobile application for iOS and Android, created to provide a community for cat enthusiasts. Our app allows users to share photos of their favorite animals, create communities, and comment on pictures, fostering opportunities for social interaction and support. Our goal is to provide you with a pleasant and satisfying experience using "Crazy Cats" and to meet your needs as cat lovers.

2. SCOPE

The target audience for the "Crazy Cats" mobile application consists of users with iOS and Android devices who adore cats and want to share photos of their favorites and communicate with other cat enthusiasts. The main functions to be tested include adding cat photos, viewing photos from other users, creating communities, and commenting on photos.

The application should be compatible with both platforms, meet the requirements of the Lean Software Development methodology, and ensure productivity and speed. Business goals include supporting social interaction among users and improving their mood through viewing funny cat photos.

2.1 Functions to be tested.

Adding cat photos by users: Verification of users' ability to add their cat photos to the application.

Viewing cat photos uploaded by other users: Testing the functionality of viewing photos uploaded by other users.

Creating communities for sharing cat photos: Identification and verification of users' ability to create communities for sharing cat photos.

Commenting on cat photos: Verification of users' ability to leave comments under cat photos.

2.2 Functions not to be tested

Server load testing: Testing how the program performs under heavy user loads will not be conducted.

Server scaling: Testing how the program performs with an increased number of servers will not be carried out.

Data backup: Automatic data backup functionality will not be checked.

Sorting or search algorithm efficiency: The speed of sorting or search algorithms will not be tested.

Administrative panel for administrators: If there is no separate control panel, its functionality will also not be verified.

3 QUALITY OBJECTIVES

Reliability: Ensure that the "Crazy Cats" application operates consistently and smoothly for all users without known instances of crashes or failures.

Performance: Ensure a fast and efficient response of the application to user actions, including quick loading of photos and comments.

Intuitive experience: Ensure that the application interface is intuitive and user-friendly for all users, as well as aesthetically pleasing and attractive.

Security: Ensure the security of users' personal data, including maintaining the confidentiality of photos and other inputted information.

Compatibility: Ensure compatibility of the application with various versions of iOS and Android operating systems, as well as with different models of mobile devices.

4 TEST APPROACH

Testing on various devices: Checking compatibility with different models and versions.

Testing from different perspectives: Evaluating functionality, reliability, and performance.

Lean Software Development: Utilizing Lean approach for efficiency and continuous improvements.

Quality control: Rigorous quality control to detect and rectify defects.

5 ROLES AND RESPONSIBILITIES

Role	Staff Member	Responsibilities
Chief Stakeholder		1) Providing requirements and usage scenarios. 2) Making decisions regarding functionality and design.
Back-end Developer		1) Developing the server-side of the application. 2) Ensuring security and efficiency of the server architecture. 3) Integrating the server and client sides of the application.
iOS Developer		1) Developing the client-side of the application for iOS devices. 2) Ensuring compatibility with various versions of iOS and device architectures. 3) Testing and optimizing the application's performance on iOS.
Android Developer		1) Developing the client-side of the application for Android-based devices. 2) Ensuring compatibility with various versions of Android and device models. 3) Testing and optimizing the application's performance on Android.
Tester		1) Performing various types of testing, including functional, integration, and regression testing. 2) Recording and analyzing test results.

		3) Identifying and documenting application bugs and deficiencies.

6 ENTRY AND EXIT CRITERIA
6.1 Entry

Development of the User Interface (UI) for both platforms (iOS and Android) should be completed.

The application's functionality, including adding photos, viewing other users' photos, creating communities, and commenting on photos, should be implemented.

Integration between the client and server parts of the application should be successfully executed.

All major functional requirements and usage scenarios provided by the chief stakeholder should be implemented.

Execution of key testing stages, such as unit, integration, and system testing, should be completed.

All critical errors found during previous testing stages should be identified and rectified.

Passing the security requirements compliance check and ensuring data privacy.

Obtaining confirmation from developers and testers about the readiness of the application for the next development stage or release.

6.2 Exit Criteria

All requirements from the test plan have been successfully tested and confirmed.

All critical errors and issues have been identified and rectified.

Full agreement among developers, testers, and stakeholders regarding the quality and readiness of the application for release.

Successful completion of all test scenarios and test cases, including positive and negative cases.

Absence of any open major or critical errors.

Signing of appropriate documentation and testing reports.

Receipt of official approval from stakeholders for the release of the application.

7. Suspension Criteria and Resumption Requirements

7.1 Suspension Criteria

• The build contains many serious defects which seriously impede or limit testing progress.

• Significant change in requirements suggested by the client.

• Software/Hardware problems that prevent testing activities from continuing.

• Assigned resources are not available when needed by the test team.

7.2 Resumption Criteria

Resumption will only occur when the problem(s) that caused the suspension have been resolved:

• All critical defects that caused the suspension have been fixed and verified.

• Any significant requirement changes have been fully clarified, documented, and integrated into the test plan.

• Software or hardware problems have been resolved, and the test environment is operational.

• All necessary resources, including personnel and tools, are available and ready for testing.

8 TEST STRATEGY

8.1 QA role in test process

Planning and Strategy: Defining testing goals and scope, considering requirements and participating in approach determination.

Test Cases: Creating and executing test cases, both manually and using automation.

Defect Management: Identifying, registering, and tracking defects in collaboration with developers.

Regression Testing: Checking new features to prevent regressions.

Performance and Security Testing: Evaluating application performance and ensuring security.

Cross-Platform Testing: Verifying compatibility on various devices and platforms.

8.2 Bug life cycle:

All the issues found while testing will be logged into a Word document. The bug life cycle will follow these steps:

New: when a tester identifies a new defect, it is logged as a new issue in the Word document, providing detailed information such as steps to reproduce, screenshots, severity, and any other relevant data.

Assigned: the newly logged defect is reviewed by the Test Lead or Project Manager, and then assigned to a developer for resolution.

Open: the assigned developer starts working on fixing the defect. The status is updated to "Open" to indicate that it is currently being addressed.

Fixed: once the developer has resolved the defect, they change the status to "Fixed" and include notes on the resolution.

Retest: the tester retests the defect to ensure that it has been successfully fixed. If the defect is not fixed, it is reassigned back to the developer with additional notes.

Closed: if the retest confirms that the defect is resolved, the status is changed to "Closed" to indicate that the issue is fully resolved.

Reopened: if a closed defect reoccurs or is found not to be completely fixed, it is reopened and reassigned to the developer for further investigation.

Deferred: if the defect is deemed low priority or outside the scope of the current release, it may be deferred for future consideration.

Duplicate: if the defect is found to be a duplicate of an already logged issue, its status is changed to "Duplicate" and it is linked to the original issue.

Rejected: if the defect is not considered valid (e.g., due to incorrect testing or misunderstanding of requirements), it is rejected with an explanation provided by the reviewer.

8.3 Testing types

Functional Testing: Verifying the functionality of the program, such as adding pictures of cats, viewing images from other users, creating communities, and commenting on photos.

UI Testing: Checking the conformity of interface elements display to the application's design and its ease of use.

Interaction Testing: Verifying the correctness of the application's interaction with the server for uploading and saving photos, as well as data exchange between users.

Performance Testing: Measuring the speed of photo loading, application response, and its responsiveness to user requests.

Security Testing: Checking the presence of security measures to protect user data and prevent abuse.

Compatibility Testing: Verifying the compatibility of the application with different operating system versions and devices.

Resilience Testing: Verifying the program's resilience to internal and external failures, such as network connection loss or server failure.

8.4 Bug Severity and Priority Definition

Bug Severity and Priority fields are both crucial for categorizing bugs and prioritizing when and how the bugs will be fixed in the "Crazy Cats" project. The severity and priority levels for bugs will be defined as outlined below:

Severity List

Severity ID	**Severity**	**Severity Description**
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Severity 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Severity 2 bugs cause serious problems such as a lack of functionality, insufficient or unclear error messages that can have a major impact on the user, preventing other areas of the app from being tested, etc. Severity 2 bugs can have a workaround, but

		the workaround is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple workaround for the bug if it is Severity 3.
4	Minor	Documentation errors or signed-off Severity 3 bugs.

Priority List

Priority ID	**Priority Level**	**Priority Description**
1	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.
2	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay the shipping date, then fix it.
4	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed. Enhancements/Good-to-have features incorporated are just out of the current scope.

9 RESOURCE AND ENVIRONMENT NEEDS

9.1 Testing Tools

Process	Tool
Test Case Creation	TestRail
Test Case Tracking	TestRail Jira
Test Case Execution	TestRail Selenium
Test Case Management	TestRail Jira Git
Defect Management	TestRail Jira Git

9.2 Configuration Management

Documents CM: SVN
Code CM: Git

9.3 Test Environment

Operating Systems: iOS: mobile devices such as iPhone or iPad running on various versions of iOS. Android: mobile devices from different manufacturers running on various versions of Android.

Development and Testing Tools: Xcode: integrated development environment for creating and testing iOS applications. Android Studio: integrated development environment for creating and testing Android applications.

Emulators and Simulators: iOS Simulator: built-in tool in Xcode for testing iOS applications without a real device. Android Emulator: tool for emulating various Android devices for application testing.

10 TEST SCHEDULE

Task Name	**Start**	**Finish**	**Effort**	**Comments**
Preparing test requirements	date	date	1 week	
Test case development	date	date	2 weeks	
Test case and test scenario execution	date	date	3 weeks	
Detection and documentation of errors	date	date	2 weeks	
Verification of fixes	date	date	1 week	
Release readiness approval	date	date	1 week	
Preparation and release of an update	date	date	1 week	

APPROVALS:

	Project Manager	QA Lead
Name		
Signature		