*TastySnaps*

Test Plan

TastySnaps\_TestPlan\_v1.0

*03/05/2024*

TABLE OF CONTENTS

[1. VERSION HISTORY………………………………………………………………………………………………….….3](#_heading=h.2et92p0)

[2.REFERENCES…………………………………………………………………………………………….………………..3](#_heading=h.tyjcwt)

3. INTRODUCTION…………………………………………………………………………………………………….…..4

4. SCOPE………………………………………………………………………………………………………….…….….….4

5. TEST ITEMS…………………………………………………………………………………………………..……….....4

6. SOFTWARE RISK ISSUES ……………………………………………………………………………………………5

7. FEATURES TO BE TESTED…………………………………………………………………………………..………5

8. FEATURES NOT TO BE TESTED………………………..…………………………………..….………….….….6

9. PROJECT STRUCTURE……………………………………..…………………………………………………..…….6

10. APPROACH……………………………………………………………………….………………………….…….……7

10.1. BUG LIFE CYCLE.........................................................................................................9

10.2. BUG SEVERITY AND PRIORITY DEFINITION……………………………………………….………...10

11. ITEM PASS/FAIL CRITERIA…………………………………………………………………….…………..…..11

12. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS.....................................12

13. TEST DELIVERABLES ..................................................................................................12

14. REMAINING TEST TASKS............................................................................................13

15. ENVIRONMENTAL NEEDS..........................................................................................13

16. STAFFING AND TRAINING NEEDS ………………………………………………………………………….13

17. RESPONSIBILITIES………………………………………………………………………………………..………..14

18. SCHEDULE……………………………………………………………………………………………………….…….15

19. PLANNING RISKS AND CONTINGENCIES………………………………………………..…………...…16

20. APPROVALS……………………………………………………………………………………………………..……17

21. GLOSSARY……………………………………………………………………………………………………….…….17

# 1. VERSION HISTORY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 1.0 | *Petrenko Victoria* | *03/05/2024* |  |  | Test Plan draft |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 2. REFERENCES

<https://www.testrail.com/> is used for creating test cases and test suits, running test cases and analyzing the results.

<https://victoriapetrenko.atlassian.net/jira/> is used for communicating of team members and bug-reporting.

# 3. INTRODUCTION

This Test Plan document outlines the testing approach and strategies for the *TastySnaps* application. The Test Plan document is created during the Planning Phase of the project.

The Test Plan has been created to facilitate communication within the team members. This document describes approaches and methodologies that will apply to the unit, integration and system testing of the *TastySnaps*.

Its intended audience is the Software Development Team and the Software Testing Team. Some portions of this document may on occasion be shared with the customer/user and other stakeholders whose input/approval into the testing process is needed.

# 4. SCOPE

The scope of the test plan for the *TastySnaps* application includes the various functional and non-functional aspects of the application. *TastySnaps* is a mobile app designed for users to share photos of food, view photos shared by others, and interact through comments and likes. The testing will ensure that the application meets its requirements. The testing efforts will be focused on ensuring that the core functionalities of the TastySnaps application work as intended and provide a positive user experience.

# 5. TEST ITEMS

*TastySnaps* is a mobile application for sharing food photos, intended for iOS and Android devices. The functional and non-functional testing is provided to ensure the quality of the product.

# 6. SOFTWARE RISK ISSUES

Potential risks associated with the software.

*Product risks:*

1. Functionality gaps: certain features may not be implemented as required, leading to functionality gaps.
2. Performance issues: during peak usage time the application may suffer from performance issues, influencing user experience.
3. Security vulnerabilities: a risk of security breaches and other common vulnerabilities.
4. Compatibility problems: the application may not function correctly on all targeted devices and operating systems.

*Team risks:*

1. Skill gaps: the development team may lack specific skills for implementing certain features.
2. Communication failure: problems of communication between team members and stakeholders may result in misunderstanding of requirements, goals and wrong decisions.
3. Resource restrictions: limitations may impact the project timeline and quality.

*Business risks:*

1. Market competition: the market of photo-sharing applications is overfilled with established players, thus attracting a significant amount of users may become a problem.
2. Profit generation: monetization may be challenging, it requires implementation of advertisement and premium features.

# 7. FEATURES TO BE TESTED

1. *Functional testing:*
   1. User registration and authorization.
   2. Photos uploading and management
   3. Slideshow posting
   4. Posting photo descriptions
   5. Commenting function
   6. Like function
   7. Saving photos to Favorites list
   8. Users interactions: following and adding to friends list
   9. User profile functions
   10. Search and hashtags
2. *Non-functional testing:*
   1. Language switch and localization
   2. Performance
   3. Security
   4. UI/UX
   5. Compatibility
   6. Compliance to regulations and standards

# 8. FEATURES NOT TO BE TESTED

* + - 1. Back-end system testing
      2. Third-party integrations

# 9. PROJECT STRUCTURE

1. User authentication
   1. Sign Up
   2. Log in
   3. Forgot password
2. Home page
   1. Feed
   2. Categories
   3. Search bar
3. Recommendations
   1. Categories
   2. Trending posts
   3. Posts you may like
4. User profile
   1. Username
   2. Profile description
   3. Followers/Following/Friends
   4. Settings
   5. Contacts
   6. User`s photo grid
5. Direct messages
   1. New chat
   2. Chats
   3. Drafts
   4. Requests
6. Photo upload
   1. Take a photo
   2. Uploading form
   3. Photo editor
   4. Photo description
   5. Drafts
   6. Slideshow post
7. Photo details
   1. Photo view
   2. Comment section
   3. Like, share, save buttons
   4. Related photos
8. Notifications
   1. Push-Notifications
   2. Notification settings
9. Saved in Favorites
   1. Save photo to Favorites
   2. My Favorites lists
10. Search
    1. Search field
    2. Recently searched

# 10. APPROACH

Testing process is based on the following principles of testing:

*1. Testing shows the presence of defects, not their absence.*

Explanation: the primary goal of testing is to identify defects within a software product. Testing can demonstrate that defects are present, but it cannot prove that there are no defects.

1. *Exhaustive testing is not possible.*

Explanation: the number of possible input combinations (user interactions, system configurations and environmental conditions) is enormous. Testing every possible state and transition is impractical. By accepting this principle, testers can focus on creating efficient, risk-based, and prioritized test cases that provide the greatest value within the constraints of time and resources.

*3. Early Testing.*

Explanation: testing activities should start as early as possible in the software development lifecycle (SDLC). The cost of fixing defects increases significantly as the development process progresses. Early testing ensures that quality is built into the product from the beginning.

By focusing on these principles, testers can ensure that testing efforts are well-directed, efficient, and result in quality and reliability of software.

Agile methodology is based on flexibility, transparency of processes, attention to feedback and the ability to make changes quickly. Also, Agile emphasizes collaboration and the exchange of ideas between team members, which contributes not only to the development of functionality but also helps to create an attractive and user-oriented product.

Within the Agile methodology, we prefer the Scrum methodology because:

1. The workflow is aimed at creating iterations during short, clearly defined time periods of "sprints". This allows quickly implementing new features, testing them, fixing bugs and responding to comments and ideas.

2. During short daily meetings, the team "checks watches", quickly makes adjustments, evaluates the amount of work done and sets a clear plan for the nearest future.

3. Without unnecessary bureaucracy and long meetings, working time is spent more efficiently.

4. Thanks to the visualization board, the process of working on the product is structured, it is easy for participants to track it and coordinate team actions.

5. Scrum is aimed at self-organization and personal motivation of a team with a small number of members who are interested in a quick and high-quality result.

The project is using an Agile Scrum approach, with weekly iterations. At the end of each week the requirements, identified for that iteration, will be delivered to the team and will be tested.

The testing approach involves manual testing techniques to ensure comprehensive coverage of the core application's functionalities, focusing on user interactions. QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.

Test cases are executed on multiple Android and iOS devices to ensure compatibility.

QA conducts manual performance testing to identify bottlenecks and optimize performance.

Developer team will execute automated test scripts that run on the developer environment.

# 10.1. BUG LIFE CYCLE

Bug statuses according to the Bug Life Cycle:

*New*: The bug has been reported but not yet reviewed.

*Assigned*: The bug has been assigned to someone for further review or resolution.

*Opened*: The bug is being reviewed or actively resolved.

*Fixed*: The bug has been fixed in the code.

*Retested*: The bug has been retested after being fixed.

*Verified*: The bug has been confirmed as fixed.

*Closed*: The bug is closed because it has been fixed and tested.

*Re*-*opened*: After a bug has been fixed, it has reappeared in the application during retesting or in use. This means that the problem has returned and needs to be analyzed or fixed again. When a bug is reopened after being closed, it can be reopened for further review and resolution by the developers.

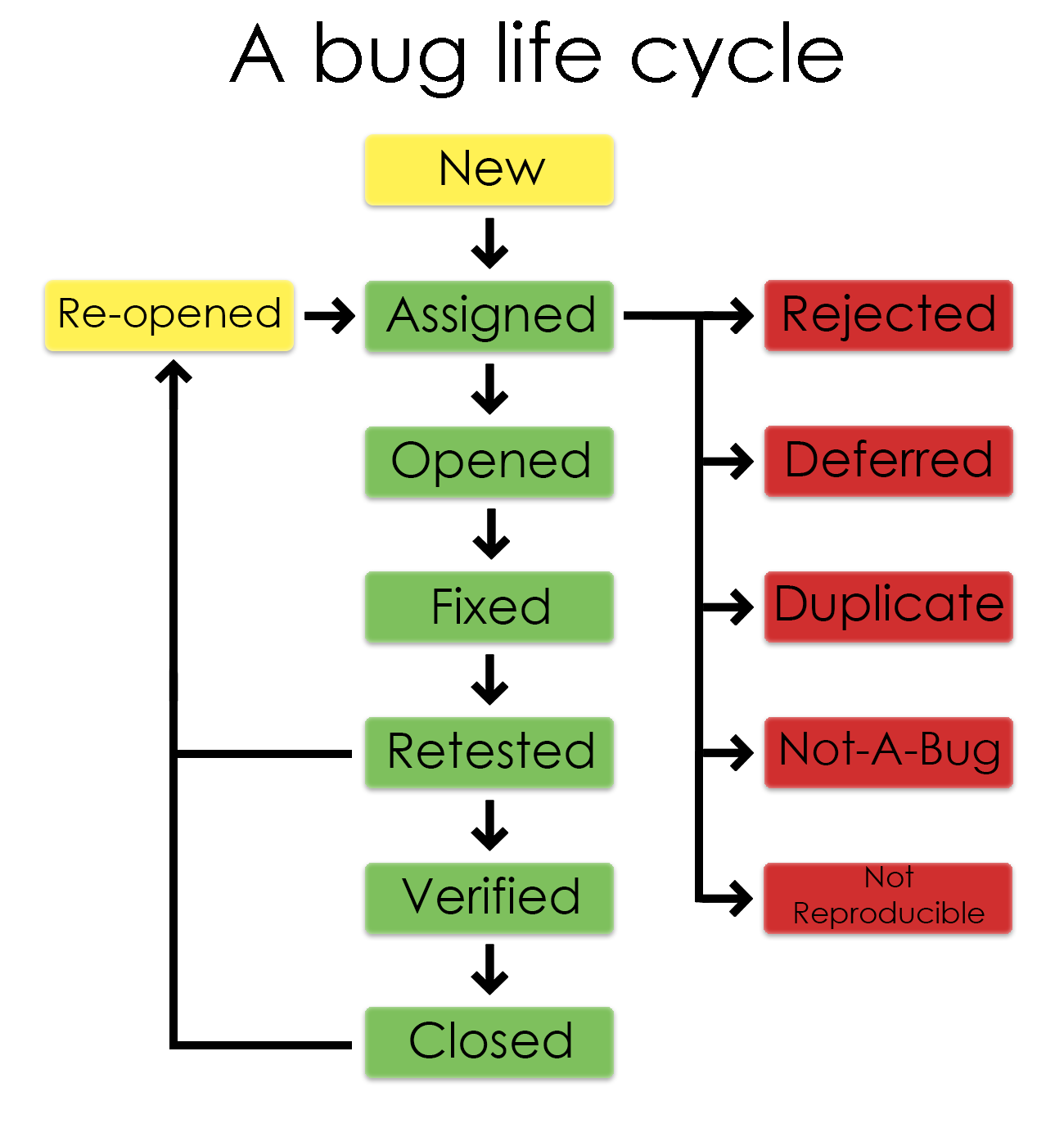
*Rejected*: A bug has been rejected for some specific reason, such as it is not considered a problem, or it needs to be addressed at a later stage of development. The reason for the rejection is usually given.

*Referred*: The bugfix is deferred to the future if the issue is not critical or relevant at the moment. Bugs with a "Deferred" status can remain in the task list for future releases or iterations.

*Duplicate*: A bug is detected as a duplicate of another bug in the system. This can occur when the same issue has been flagged multiple times or is similar to another issue already reported.

*Not* *Reproducible*: The bug cannot be reproduced. This means that it was not possible to reproduce the issue under certain circumstances or on a certain device.

*Not-A-Bug*: After careful analysis, the issue that was originally identified as a bug is actually expected application behavior or the result of user misuse.



# 10.2. BUG SEVERITY AND PRIORITY DEFINITION

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below.

*Severity List*

|  |  |
| --- | --- |
| Critical | The module/product crashes or the bug causes nonrecoverable conditions. System crashes, or  database or file corruption, or potential data loss, program hangs requiring reboot. |
| High | Major system components unusable due to failure or incorrect functionality. Bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact on the user, prevent other areas of the app from being tested, etc. Bugs can have a work around, but the work around is inconvenient or difficult. |
| Medium | Incorrect functionality of component or process. There is a simple work around for the bug. |
| Minor | Documentation errors or signed off. |

*Priority List*

|  |  |
| --- | --- |
| Must Fix | This bug must be fixed immediately; the product cannot ship with this bug. |
| 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. |
| 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. |
| 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 just are out of the current scope. |

# 11. ITEM PASS/FAIL CRITERIA

The item pass/fail criteria define the conditions that each test item must meet to be considered successful or unsuccessful. For *TastySnaps*, the criteria will be based on the application's functional and non-functional requirements.

An entry criterion refers to specific conditions or on-going activities, which must be present before a process can begin.

*Entry Criteria:*

* All test hardware platforms must have been successfully installed, configured, and functioning properly.
* All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behavior.
* All the standard software tools including the testing tools must have been successfully installed and functioning properly.
* Proper test data is available.
* The test environment such as lab, hardware, software, and system administration support should be ready.
* QA resources have completely understood the requirements
* QA resources have sound knowledge of functionality x Reviewed test scenarios, test cases and RTM

An exit criterion refers to the requirements needed to complete a specific task.

*Exit criterion:*

* 100% of requirements coverage has been achieved.
* 100% of requirements are covered by tests.
* 100% of critical bugs and 90% of high priority bugs are fixed.
* All high-risk areas have been fully tested, with only minor risks left outstanding.
* Cost – when the budget has been spent.
* The schedule has been achieved.

# 12. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

*12.1 Suspension criteria*

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

• Significant change in requirements suggested by client

• Software/Hardware problems

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

*12.2 Resumption criteria*

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

# 13. TEST DELIVERABLES

The following are the key deliverables for the *TastySnaps* testing phase:

1. Test Plan Document
2. Test Cases and Test Suits
3. Test Data
4. Test Execution Reports
5. Defect Reports
6. Traceability Matrix
7. Test Summary Report
8. Performance Testing Reports
9. User Acceptance Testing (UAT) Report

These deliverables ensure a thorough and organized approach to testing, providing clear documentation and evidence of the testing process and results.

# 14. REMAINING TEST TASKS

1. Test Case preparation

2. Test environment setup

3. Test data preparation

4. Test execution

5. Defect management

6. Test reporting and documentation

7. User Acceptance Testing (UAT)

8. Test Closure

# 15. ENVIRONMENTAL NEEDS

*Mobile Devices:*

A variety of iOS and Android devices (phones and tablets) to cover different OS versions and screen resolutions.

Minimum: 5 Android devices, 5 iOS devices.

*Network:*

Stable internet connections to simulate real-world usage conditions.

Tools to simulate different network conditions

*Testing Tools:*

Test management tools (Jira, TestRail) for tracking test cases and defects.

Performance testing tools.

Security testing tools.

*Communication Tools:*

Collaboration tools (Slack) for team communication.

Documentation tools (Google Docs) for maintaining test documentation.

# 16. STAFFING AND TRAINING NEEDS

Personnel:

* Scrum master
* 3 developers: back-end, iOS, Android
* QA Test Lead
* QA Testers
* UX/UI Designers
* Database Administrator

Training: sessions for QA team on new tools or testing methodologies.

# 17. RESPONSIBILITIES

**Scrum Master Responsibilities**:

1. Facilitate daily stand-ups, sprint planning, sprint reviews, and retrospectives.
2. Ensure the team adheres to Agile principles and Scrum practices.
3. Remove any impediments that may hinder the team's progress.
4. Act as a communicator between the development team and stakeholders.
5. Track and report on project progress and team performance.

***Developers:***

**Back-End Developer Responsibilities**:

1. Design, develop, and maintain the server-side logic.
2. Implement APIs and integrate with front-end services.
3. Optimize the application for performance and scalability.
4. Collaborate with QA to resolve backend-related defects.

**iOS Developer Responsibilities:**

1. Design, develop, and maintain the *TastySnaps* iOS application.
2. Ensure the application meets performance and quality standards.
3. Implement new features and fix iOS-specific issues.
4. Collaborate with UX/UI designers to ensure a seamless user experience.

**Android Developer Responsibilities:**

1. Design, develop, and maintain the *TastySnaps* Android application.
2. Ensure the application meets performance and quality standards.
3. Implement new features and fix Android-specific issues.
4. Collaborate with UX/UI designers to ensure a seamless user experience.

**QA Test Lead Responsibilities:**

Oversee the overall testing process and strategy.

Coordinate and manage the QA team, ensuring test activities align with project goals.

Review and approve test plans, test cases, and test scripts.

Ensure defects are documented, tracked, and resolved promptly.

Communicate test progress, results, and issues to stakeholders.

**QA Testers Responsibilities:**

Design and execute test cases based on functional and non-functional requirements.

Perform manual testing to identify defects and ensure quality.

Document test results and report defects in the tracking system.

Conduct retesting and regression testing after defect fixes to ensure stability.

Collaborate with developers to reproduce and resolve defects.

**UX/UI Designers Responsibilities:**

Design intuitive and visually appealing user interfaces for the *TastySnaps* application.

Conduct user research and usability testing to gather feedback and improve the design.

Create prototypes, and mockups to communicate design ideas.

Ensure consistency in the design across iOS and Android platforms.

Collaborate with developers to ensure design feasibility and implementation.

**Database Administrator Responsibilities:**

Design, implement, and maintain the database schema and structure.

Ensure data integrity, security, and performance optimization.

Perform regular backups and recovery operations.

Monitor database performance and make necessary adjustments.

Support the development and QA teams with database-related tasks and issues.

# 18. SCHEDULE

The testing schedule for the TastySnaps application outlines the timelines for the various testing activities. This schedule ensures that all testing phases are completed in time, allowing for the identification and fixing of defects before the application is released.

1. Test Plan development

Start Date: Week 1

End Date: Week 2

2. Test environment setup

Start Date: Week 2

End Date: Week 3

3. Test Case development

Start Date: Week 2

End Date: Week 4

4. Test data preparation

Start Date: Week 3

End Date: Week 4

5. Test Execution

Start Date: Week 5

End Date: Week 8

6. User Acceptance Testing (UAT)

Start Date: Week 8

End Date: Week 9

7. Test Reporting and Documentation

Start Date: Week 5

End Date: Week 9

8. Test Closure

Start Date: Week 9

End Date: Week 10

# 19. PLANNING RISKS AND CONTINGENCIES

1. Schedule delays: testing activities may take longer than expected, leading to delays in the project timeline.

Contingency: implement a buffer period in the schedule to cover potential delays. Prioritize critical and high test cases to ensure essential functionality is tested first.

2. Resource availability: personnel may be unavailable due to unforeseen circumstances such as illness, blackouts or personal emergencies.

Contingency: train team members to cover critical roles. Maintain a pool of part-time or contract testers who can be involved if needed.

1. Problems with test environment: may not perform as expected.

Contingency: review and update the test environment to ensure it is adequate to the production setup.

1. Changing Requirements:

Risk: Requirements may change during the testing phase, leading to additional testing efforts.

Contingency: Maintain flexibility in the test plan and update test cases as requirements evolve. Regularly communicate with stakeholders to manage changes.

# 20. APPROVALS

| **Role** | **Name** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Scrum Master | [Scrum Master's Name] |  | [Date] |
| QA Test Lead | [QA Test Lead's Name] |  | [Date] |
| iOS Developer Lead | [iOS Developer's Name] |  | [Date] |
| Android Developer Lead | [Android Developer's Name] |  | [Date] |
| Back-End Developer Lead | [Back-End Developer's Name] |  | [Date] |
| UX/UI Designer Lead | [UX/UI Designer's Name] |  | [Date] |
| Database Administrator | [Database Admin's Name] |  | [Date] |
| Product Owner | [Product Owner's Name] |  | [Date] |
| Stakeholder Representative | [Stakeholder's Name] |  | [Date] |

# 21. GLOSSARY

This section provides definitions for terms and acronyms used in the Test Plan for the *TastySnaps* application. It aims to ensure a common understanding of terminology among all stakeholders.

*Agile*: A methodology for software development that emphasizes iterative development, collaboration, and flexibility to adapt to changing requirements.

*Automated testing*: The use of software tools to execute pre-scripted tests on a software application before it is released into production.

*Back-End Developer*: A developer responsible for server-side web application logic and integration with front-end components.

*Critical Bugs*: Defects that significantly impact the functionality and usability of the application, requiring immediate attention and resolution.

*Defect*: An error or bug in the software that causes it to produce incorrect or unexpected results.

*Exploratory Testing*: A testing approach where testers explore the application without predefined test cases, using their creativity and intuition to find defects.

*Integration Testing*: Testing performed to evaluate the interaction between different modules or components of a software application.

*iOS Developer*: A developer specialized in creating applications for Apple's iOS operating system.

*Iteration*: A time-boxed period during which a set amount of work is completed in Agile methodologies. Also known as a sprint in Scrum.

*Manual Testing*: Testing of software manually without the use of automated tools, where testers manually execute test cases and observe results.

*Non-Functional Testing*: Testing the non-functional aspects of the application, such as performance, security, usability, and compatibility.

*Performance Testing*: Testing conducted to evaluate the performance of the application under certain conditions, including load testing, stress testing, and scalability testing.

*Product Owner*: A key stakeholder responsible for defining and prioritizing the product backlog in Agile projects.

*QA (Quality Assurance)*: A systematic process to ensure that the software meets specified requirements and standards, involving both testing and process improvement.

*Regression Testing*: Testing conducted to ensure that recent code changes have not adversely affected existing functionality.

*Scrum*: An Agile framework for managing work with an emphasis on iterative progress through small, time-boxed sprints.

*Scrum Master*: A facilitator for the Scrum team, responsible for ensuring the team follows Agile principles and practices.

*Security Testing*: Testing to identify vulnerabilities, threats, and risks in the application and ensure that the application is secure from attacks.

*Sprint*: A set period during which specific work has to be completed and made ready for review, typically lasting between one and four weeks.

*Stakeholder*: Any individual or group that has an interest in the outcome of the project, including customers, users, developers, testers, and management.

*Test Case*: A set of conditions or variables under which a tester determines whether an application or software system is working correctly.

*Test Environment*: The combination of hardware, software, and network configurations where the testing will be conducted.

*Test Plan*: A document outlining the scope, approach, resources, and schedule for testing activities.

*Test Script*: A set of instructions executed by an automated testing tool to perform testing on the software application.

*UAT (User Acceptance Testing*): A phase of testing where the end-users test the software to ensure it meets their requirements and is ready for deployment.

*UI/UX Designer*: A professional responsible for designing the user interface and user experience, ensuring the application is both functional and aesthetically pleasing.

*Unit Testing*: Testing individual components or modules of the software to ensure they work as intended.