Pikup
Test Plan

Introduction

Objective of Document

The objective of this document is to detail the approach to be employed for the testing of the Pikup project. The Test Approach determines what aspects of the application are being tested and by what resources.

The Test Approach defines test phases, resources, schedule, metrics, configuration management, and a description of the test environments and test tools that will be used in the effort.

Sign-Off

The sign-off of this document indicates that the signing reviewers agree with the adopted and stated Test Approach in relation to the Pikup project. Minimally, the Project Manager, User Representative and the Project Leaders of the application teams impacted should sign-off on this document. Sign-off may be in the form of a physical signature or via email.

Project Description

The goal of this design is to address the needs for Quality Engineering to ensure the Pikup project can be launched without any defects.

Management Summary

Standard testing phases will be conducted in an Agile approach. The Unit, System/System Integration, End-to-End, Performance and Stress phases will be utilized for this effort. The Unit, Performance and Stress test phases will be the responsibility of the Software Quality Engineer Team.

Common Elements across All Test Phases

Testing Control

Risk and Issue Management

The Software Quality Engineer will track risks and issues using GitHub and will house project risks for risks to testing activities.

Defect Management

All defects will be entered into GitHub. The Software Quality Engineer will discuss open defects with members of the App Development Team. These defects will then be classified as defects to be fixed and assigned to the App Development Team. Once defects have been fixed by App Dev, they will be made available for Crew Software Quality Engineer testing in the next version of the application code. Software Quality Engineer, after testing the defect fix, will update the Resolution with successful or failed status. If

successful, the defect will be closed. If unsuccessful, the defect will be transferred back to the App Dev for further work.

Key Dates and Deliverables

This section includes all key dates and deliverables for the test execution of each of the test phases. These tasks and dates along with the resources responsible are provided to the project manager to incorporate into the overall project plan. Indicate the base-lined dates here and then they are maintained in the overall project plan.

Deliverable / Task	Start Date	End Date	Responsibility
Build	01/15/2022	4/18/2021	App Dev
Unit Testing	01/15/2022	4/18/2021	App Dev / Software Quality Engineer
Script Writing	01/15/2022	4/18/2021	Software Quality Engineer
Software Quality Engineer Testing	01/15/2022	4/18/2022	Software Quality Engineer
Deployment	04/18/2021	4/23/2021	App Dev / Software Quality Engineer
Script Clean-Up	N/A	N/A	Software Quality Engineer

Testing Metrics/Measures

Test Metrics for project testing will be tracked at the test plan level. The Software Quality Engineer will track the status of the test plan (Passed, Failed, Not Complete, Blocked, and No Run) by test cycle using GitHub. These statistics along with defect statistics (Number of Open, Number of Closed, and Numbers by Severity) will be published on a weekly basis.

Unit Test Approach

3.1 Test Overview

Test Phase Objective

The objective of this phase is to execute tests that will ensure that each work product:

1. Satisfies the design, standards (documentation and code) and technical requirements.

2. Tests correctly against agreed upon test scenarios.
3. Is ready to be included in the next test phase.

Test Activities

Unit Testing will be conducted by the development team prior to Software Quality Engineer testing.

Testing Requirements

- Code must be unit tested by developer. This consists of a complete test and not just the positive scenarios.
- Developer must regression test other impacted modules when present.

In Scope

The Development Team will create and execute test scripts to test the application code developed as part of this project.

Test Cycle Strategy & Success Criteria

The testing cycle will focus on components and functionality directly impacted by code changes included in the release. The number of test cycles will depend on the number and severity of defects identified during testing. Critical and high severity defects will have to be assessed and resolved as part of the integration process and could drive additional iterations of testing.

Based on the severity definitions below, success criteria will target zero Critical and zero High defects coming out of integration testing.

Severity Definitions

1	Critical System Inoperable	• The system is unusable. A major, critical function/process is broken. No workaround can be identified. • The entire application component or function will not work. There is no bypass available, and testing cannot continue due to the error stopping the business process. It is either not possible to progress to downstream functions, or the error is passed to the downstream function not allowing the process to continue.

2	High Impacts Most Customers	• A major function or process is significantly impacted. The system is usable, but severely restricted. An acceptable workaround can be identified to continue until a fix is available. • In testing, the entire application component or function will not work. There is a bypass available, and some testing can continue without extending the error to the downstream function. • A non-critical feature, function, or process is not working properly, but impacts are minor; the workaround is unacceptable or difficult to do.
3	Medium Impacts Some Customers	• A non-critical feature, function, or process is not working properly, but impacts are minor. Usually, a workaround is easily identified. • In testing, the function tested will not perform as expected but testing can continue.
4	Low Impacts Few Customers	• A problem is identified which is minor and may be corrected by a documentation change or a text revision. No functionality or process is impacted. Work can continue.

System/System Integration Test Approach

Phase Containment

This section specifies who reviews each test deliverable for the test phase. It is critical to define specific entry and exit criteria for the test phase and to communicate these criteria to the impacting upstream development phases (e.g., design) and to preceding test tasks. This formalizes the inputs and outputs of the test phase, identifying dependencies, and ensures that the entire test process results in a quality solution.

Entry Criteria

A Test Readiness Review will be conducted to ensure everything is in place prior to the start of the System Integration Test phase. The Test Environment will be reviewed for appropriate configuration. All schedules and necessary data gathering have been completed for the test environment. Once the review is complete, the project will officially enter the Test Phase of the development cycle. GitHub defects created from this point forward will be logged as found in Test Phase.

Exit Criteria

Upon completion of the System Integration Test Phase, a test closure will be provided of Test Plans, Test Completion, and Defects. If the exit criteria are not met, the project team will determine what course of action to take at that time.

Resourcing

Roles and Responsibilities

This section is a general overview of all the test phase resourcing requirements.

- To successfully complete work, the following resources will be needed:

System/System Integration and End-to-End Testing:

- Software Quality Engineer Coordinator
- Software Quality Engineer Testing Resources

Performance and Stress Testing:

- Application teams
- Application teams for all other testing

These individuals must have a working knowledge of the application along with an understanding of all dependent systems.

Title	Role	Represented By
Test/ Software Quality Engineer Coordinator	1. Plan and facilitate project schedule, status meetings, and testing activities for the project. 2. Provide a weekly status to management for all testing phases. 3. Conduct a System Test Readiness Review with app dev. team and project managers prior to the beginning of New Functionality testing. 4. Ensure all testing is executed and results are captured in GitHub. 5. Ensure all defects are documented in GitHub 6. Facilitate a Test Closure Review of results.	Nathan Preslar
Test/ Software Quality Engineer Team	1. Review regression testing plan with team. 2. Participate in the Functional and Regression test phases. 3. Create and execute regression Test Scripts in GitHub. 4. Document defects associated with assigned test scripts in GitHub. 5. Attend defect meetings as necessary.	Nathan Preslar
App Dev	1. Ensure new release is installed and ready for start of testing. 2. Ensure correct configurations are set on the servers.	Ben Szabo Cristobal Short Jie Li Abu Abukar

Project Manager	1. Responsible for the overall success of project 2. Obtain approval for changes to rules, processes and policies	Cristobal Short
Defect Coordinator	1. Forward defects over to App Dev	Nathan Preslar

Test Tools

The Software Quality Engineer group will be utilizing GitHub.

Appendix A - Test Activities

Functional Test

Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. Functional Testing tests the features and functions of a system (software, hardware, etc.) based on its functional requirements to ensure requirements and specifications are met. It ensures that the program physically works the way it was intended.

Regression Test

Selective re-testing of existing functionality that was not intended to change with new requirements to detect faults introduced during modification of the system or the system component and to confirm that modifications have not caused unintended adverse effects. Regression testing could compare the “new” version of a system (named group of applications that provide function to the business) to a current production version.

Stress \ Performance Test

Stress Test: Tests conducted to evaluate a system or component at or beyond the limits of the specified requirements. The system resource (CPU, RAM, file handles, threads, sockets, etc.) are reduced below required capacity. The test involves operating outside normal parameters, often to a breaking point, in order to observe the results.

Performance Test: Tests conducted to evaluate the compliance of a system or component with specified performance requirements. The tests measure throughput, response time, and resource utilization, over a range of transaction volume and resource capacity. This test should involve expected operating volumes. Tests to evaluate a system at or beyond the limits of the specified requirements, often to a breaking point.

Control

Document Location

This document is stored in GitHub.

Document Owner and Contributors

Owning Team	**Author**
Software Quality Engineer	Nathan Preslar
Contributing Team	**Author**

Document History

Author	**Date**	**Version**	**Comment**
Nathan Preslar	02/15/2022	1.0	Original Document

Document Review

Name	**Position**	**Action**
Abu Abukar	Full Stack App Dev	
Ben Szabo	Back End App Dev	
Jie Li	Front End App Dev	
Cristobal Short	Full Stack App Dev	