Test Plan Pangeo by VRtualize

Release 1.0.1

Isaac Egermier, Chezka Gaddi, Garfield Tong, Michael Theesen

Nov 05, 2019

Contents

Revision History

Version	Date	Author	Description of Change
1	October 17, 2019	Chezka Gaddi	First Draft
2	October 25, 2019	Chezka Gaddi	Added branch structure, changed issue management tool

1 Introduction

1.1 Purpose

This test plan contains a description of the testing approach used to create a comprehensive plan for the testing of the Pangeo application which will be provided by the VRtualize team. This document includes:

- Strategy: Test rules and project assumptions including test objectives and certifications; end-to-end description of test set up (start/end conditions, test case creation, task lists, data sources).
- Implementation: how is testing performed, defect management process (reporting, fix planning and execution).
- Management: description of the logistics of testing (e.g. chain of command, risk identification and management, team members).

1.2 Overview

The SDSMT VRtualize Team is working together with L3Harris on project Pangeo. Pangeo is a research project aiming to render the real world in a dynamically loaded virtual reality environment. There are three main goals that contribute to this overall project: imagery retrieval, image caching, and virtual rendering.

1.3 Audience

Pangeo is designed to be an internal research tool for the company L3Harris. In addition, this tool is to be designed to be used by anyone to tour geographical points of interest from the comfort of their home.

Team members mentioned in this document will perform assigned duties and provide additional recommendations. Project Managers associated with the development and oversight of the project who will provide project schedules and reviews. Stakeholder's representative who will potentially take part in testing to see to it that results and business interests remain in sync.

2 Strategy

2.1 Objectives

Test objectives are meant to verify that the Pangeo application meets design specifications.

Testing will include the execution of automated tests, test scripts, and performance tests. Issues will be discussed with the team during the weekly stand-ups and prioritized as High, Medium, or Low severity. High and Medium severity defects must be retested as per the acceptance criteria. Low severity defects will be deferred to a subsequent sprint as discussed. Criteria for the cessation of testing:

- Production ready software (as per the Software Requirements Specification (SRS))
- Automated tests and test scripts suitable for reuse as functional and user-acceptance testing.

2.2 Assumptions

General

- Environment downtime will adversely impact test schedules.
- Test environment will exactly duplicate the production environment.
- Issue reporting includes complete reproduction details (as per Issue Reporting Template).
- Issues are tracked using the ZenHub Issue Tracking System only.
- The team will review, verify, and sign-off on all test cases prior to the start of testing.
- Issues reported fixed after Certification N will include regression tests which will be added to the test plan for Certification N+1.

Key Assumptions

- Certifications are defined as follows:

Certification	Description
Certification 1a	Pulling data from a file and rendering an object that resembles the data.
Certification 1b	Pulling data from a database instead of a file system.
Certification 2a	Pulling from a cache when necessary and pulling from the database ahead of the time
Certification 2b	User experience with movement and menu selections

Todo: Add Certification 2b description

- A release cannot go into production with any severity 1 (Critical), 2 (High) or 3 (Medium) defects.
- Functional testing requires production-like data.
- Alpha tests will be performed by identified alpha testers.

2.3 Test Principles

- Quality and business objectives are the focus of this testing.
- Testing cycles will be controlled by agreed on start and end criteria.
- Testing is divided into two test cycles (Certifications).
- Certification environment must exactly duplicate the production environment to support meaningful test results.

2.4 Testing Scope

2.4.1 Unit

Who: Development Team, the expectation is that the team implement best practices in terms of Test Driven Design (TDD) during initial development.

When: During product development.

Why: Primarily for the purpose of identifying bugs at the unit level as early as possible.

Scope: User stories, separate modular functions, scripts.

How: All major software components will be developed using Unity's built-in Test Runner which utilizes the NUnit framework or the Python module, unittest.

2.4.2 Back-end

Who: Development Team

When: During database setup.

Why: To avoid complications like deadlock, data corruption, and data loss.

Scope: Validating schema, database tables, columns, keys and indexing, stored procedures, triggers, database server validations, validating data duplication.

How: Tests will be developed as SQL queries.

2.4.3 Code Reviews

Who: Development Team

When: Within 4 days of an issue going into Review/QA pipeline.

Why: To ensure that the code upholds the coding standards.

Scope: All units of code.

How: Team members working on the same Certification will be required to participate and approve code reviews with the author of the code. Results in one of three results:

- Pass: an ok to push to merge with next branch
- Revision/Pass: can be pushed after some changes
- Fail: too many problems, will require another code review

2.4.4 QA Testing

Who: QA team (Members working on the other Certification will serve as QA)

When: After units of code go into integration branch.

Why: To ensure tha units of code passes user acceptance criteria.

Scope: All units of code.

How: Using the testing software used to create the unit, back-end, and user acceptance tests.

Todo: Check that this covers the structure we talked about.

2.4.5 Integration

Who: Development Team

When: Combining individual units.

Why: To expose defects in the interfaces and interactions between integrated components or systems.

Scope: Interaction between the database and the application, simulating the key interactions of a user using the application.

How: Tests will be developed using the Integration Test Framework, which is part of the Unity Test Tools package.

2.4.6 System and Functional

Who: QA

When: Prior to Exploratory Testing

Why: Thorough testing of all application functions.

Scope: [Note: Functional test plans are detailed lists of all the features and capabilities expected from the application. The Test Plan should be a fairly high-level document so the details and scope of the functional tests will typically appear in a spreadsheet, appendix, or other content management document].

How: Tests are performed using agreed on scripts, automated processes, and input decks.

Acceptance Criteria:

1. Functional spec and use case documents available before Test Design phase.
2. Test environment available, configured, verified, and ready to use.
3. Unit test results compiled and shared with the Development Team to avoid duplication of effort.

4. Test cases reviewed, validated, and signed off on by the Development Team.

2.4.7 Exploratory and Alpha

Who: Alpha testers

When: After Certification 2.

Why: Primarily to familiarize the alpha testers with the features and behavior of the software to set expectations for new features and identify any hiccups.

Scope: Production level product.

How: Testers are encouraged to try the interface without scripts or documentation.

2.4.8 Soak and Performance

Who: Development Team

When: Any new system update.

Why: To ensure the application does not have any memory leaks and performs to the agreed-upon performance specification.

Scope: Memory management, algorithms, and loading time.

How: Unity Profiler and Unity Performance Testing Extension to internally monitor performance and optimizations of key systems.

2.4.9 Stress

Who: Development Team

When: Before product release.

Why: To determine the acceptable user limitations.

Scope: Algorithms and loading time.

How: Unity Profiler and Unity Performance Testing Extension to internally monitor performance and optimizations of key systems.

2.4.10 Alpha Testing

Who: Identified alpha testers

When: Before release.

Why: Software is tested outside a controlled test environment by users who are unfamiliar with the product or features. These tests validate the requirements gathering and test design phases of the production cycle.

How: Business users/customers will frequently attempt to use the software in unanticipated ways. This can lead to new requirement generation (user expects functionality that was not developed), modification of the user interface (user follows a use path that causes failures or other unwanted behavior in the software), or clarification to user guides, help screens, on screen guidance.

Deliverables: UAT Test Cases written by Development Team and reviewed and signed off on by Development Team and Project Manager.

3 Issue Management

3.1 Issue Tracking Manager

ZenHub is an agile project management application within GitHub that allows organization of issues into task boards and reports.

3.1.1 Installation

Download the browser extension for Firefox or Chrome and accept permissions and sign in with GitHub. ZenHub will be added right into the GitHub interface.

3.1.2 Structure

The ZenHub Workspace is split up between 7 pipelines;

- **New Issues** Issues that have yet to be assigned or evaluated.
- **Icebox** Features that are wanted but that are not pertinent to the MVP.
- **Backlog** Issues that were not completed from the past Sprint. The highest priority.
- **In Progress** Issues that currently being worked on by the development team.
- **Review/QA** Issues that are ready to be code reviewed and QA tested.
- **Done** Issues that are done.
- **Closed** Closed issues.

Todo: Make sure that the pipelines have the correct description. Done and Closed need further editing.

3.1.3 Issue Types

- User Story
- Feature Request

- UX Issue
- Performance
- Bug

Todo: Description of each of the tags needed?

3.1.4 Issue Complexity

Complexity	Difficulty
1	Trivial (Typo, small fix)
2	Busy Work (Easy Function)
3	Light (New Function)
5	Medium Rare (Set of interacting functions)
8	Medium Well (New feature)
13	Heavy (Major fix, multiple features)
21	Perplexing (Researching new techniques)
40	Beastly (Unsure if possible)

3.2 Issue Life-Cycle

1. **Issue Creation** Project Manager assigns initial priority, effort, queue position, etc. and assign to a group based on Certification subsection.
2. **Issue Assignment** During Sprint Planning, issues to be worked on get decided and are assigned to specific members.
3. **Work on Issue** Develop Unit Tests at this time.
4. Issue goes through unit tests
5. **Issue goes through code review** Code Review is scheduled no later than 4 days after the issue is moved into the QA/Review swimlane. Team members working on the same Certification subsection are required to participate and approve each issue.
    - if passes, issue is merged into integration branch (developer marks the issue as done)
    - if fails, repeat process at step 3

Todo: Make this section flow a little bit better, also, determine how to handle issues that go through entire testing suite.

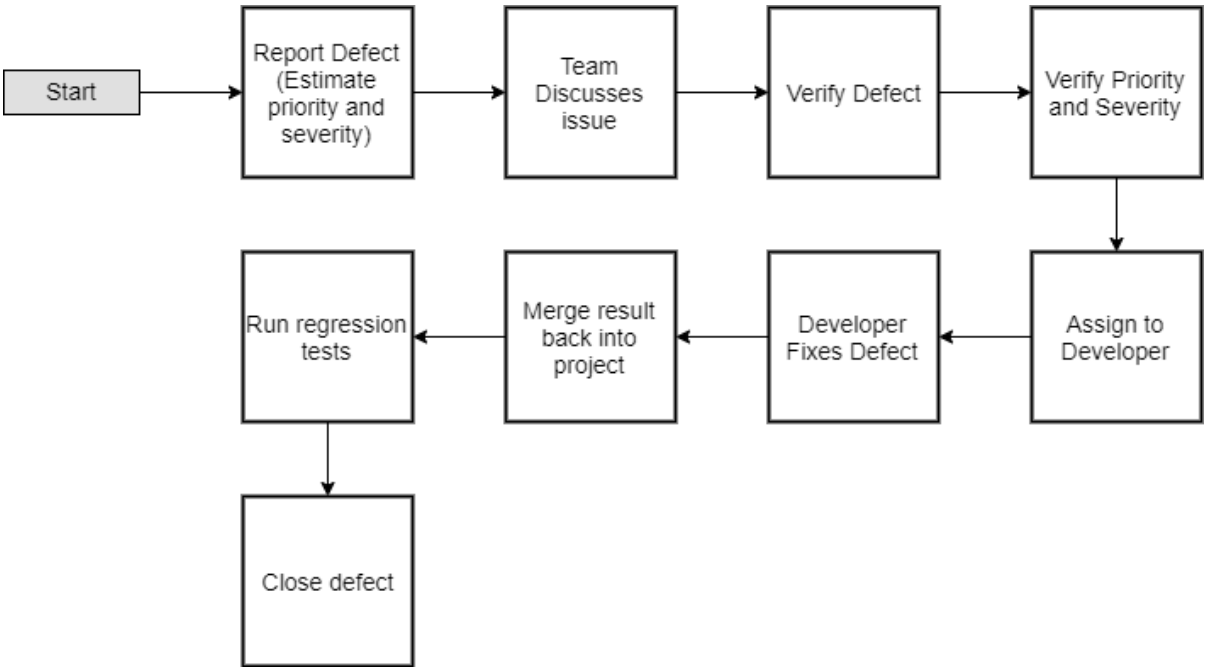
6. **Issue goes through QA testing** QA tests are developed by members not working on the same Certification subsection. This is to ensure that functions behave as expected.
    - if passes, continues to integration tests
    - if fails, issue is reopened and is reverted back to the feature branches
7. Integration Tests
    - if passes, stays in integration branch until Certification 2

- if fails, a new issue is created (go to step 1)

Todo: Add a diagram for the life-cycle of an issue?

3.3 Defect Reporting

3.3.1 Bug Life-cycle



1. Bug or defect is reported using ZenHub's Issue Tracking System and the founder will estimate the priority and severity
2. The bug will be verified by the team
3. The priority and severity of the bug will be verified
4. The bug will be assigned to a developer based on who's code the bug was found in.
5. The developer fixes the bug.
6. The fix will be merged back to the integration branch of the project.
7. Regression tests will be run to ensure it bug was fixed.
8. Defect is closed.

3.3.2 Severity Levels

Severity	Risks
1 (Critical)	• Defect causes the application to crash or hang • Corrupts application or system data • Consumes system resources to the point that other system processes are adversely affected
2 (High)	• Missing major application functionality without a workaround
3 (Medium)	• Missing minor application functionality without a workaround • Missing major application functionality with a workaround • Defect causes other features to be unavailable for review or testing

3.4 Metrics

Tracking progress and success of the tests for each test cycle. Delivered to Project Manager and Development Team.

- Weekly Status Report: Provided to Project Manager and Development Team. Includes weekly pass/fail/complete percentages. Identify and troubleshoot any defects in the Showstopper or Critical category which have persisted for over a week. Report ongoing risks and mitigation plans.
- Sprint End Report: Compile trajectory graphs for defect lists broken out by status, severity, and age.

3.5 Start and End Criteria

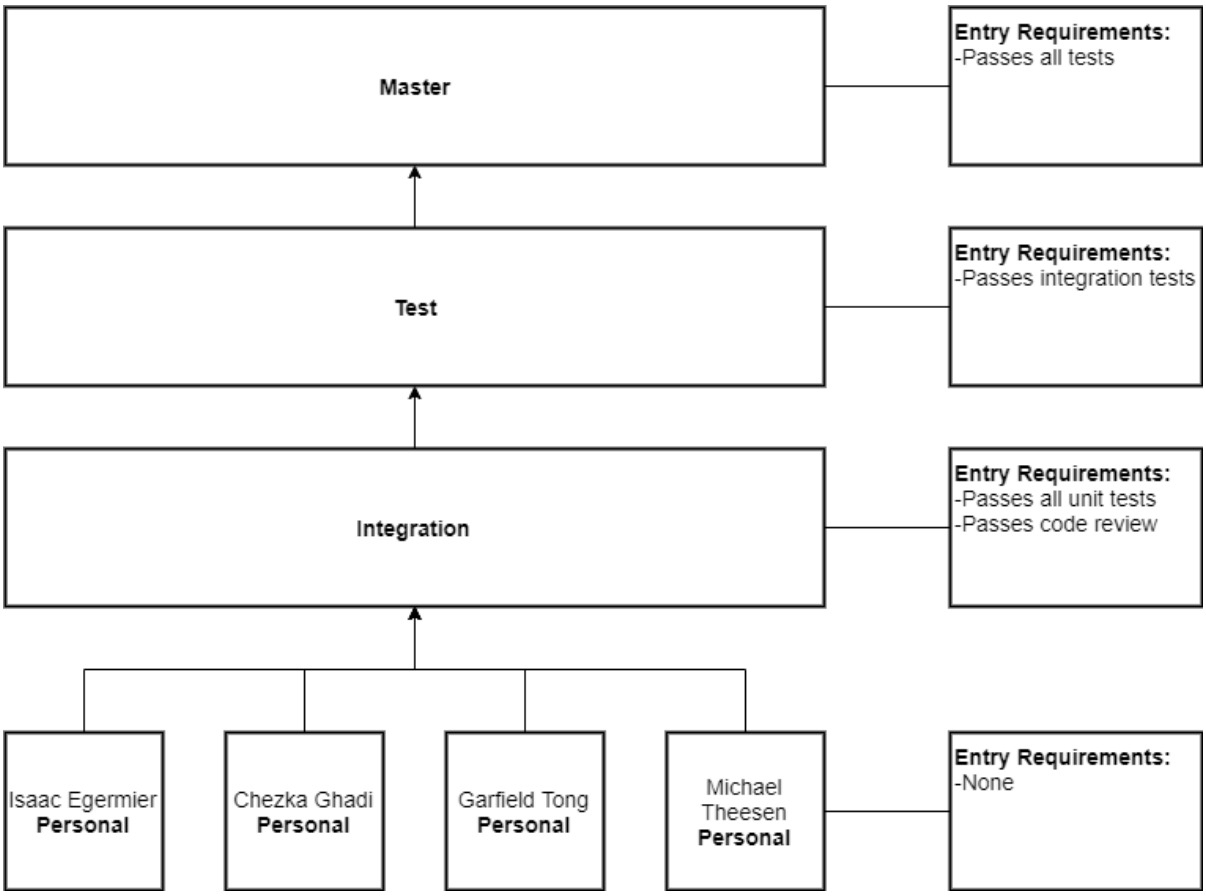
- Start criteria detailed in the Test Planning Strategy section.
- Start criteria refer to the desirable and necessary conditions which need to be in place before test execution can be started.
- Start and end criteria are flexible since, especially during Sprint 2, it is understood that environments, accounts, data, and documentation may still be in an immature state. Start criteria will be evaluated by Product Manager for a go no-go determination at the start of a sprint.
- Start criteria expectation is that the activities listed in the Test Execution section of the schedule are all in place at the start of each sprint.
- **End criteria**
  - Test Script execution Owner: Dev Team
  - 95% pass rate on Test Scripts Owner: Dev Team
  - Zero severity 1 or 2 level defects Owner: Dev Team
  - 95% severity 3 level defects closed Owner: Dev Team
  - Remaining defects converted to Change Requests or Deferred Owner: Dev Team
  - 100% Coverage of requirements captured by expected and actual test script execution. Owner: Dev Team

- 100% Test strategy metrics collected Owner: Dev Team
- 100% of defects logged in GitLab's Issue Tracker System Owner: Dev Team
- Final Test report reviewed, verified, and signed off on by Product Manager and Dev Team
- Test environment check pointed, tagged, and backed up Owner: Dev Team

4 Repository Management

The repository is owned by Michael Theesen and is stored in the GitHub server under the name VRtualize-Pangeo. All members of the team have Developer permissions.

4.1 Branch Structure



4.1.1 Feature Branches

Each issue with have a feature branch for the purpose of developement specifically for that issue named using the following standard:

developer-name/issue#-name-of-issue

Unit tests will be developed, run, and passed in this branch before a code review can be scheduled.

4.1.2 Integration Branch

Cost of Entry: Passes unit tests and code review.

The relationships between units of code is tested within this branch.

Advancement to upper branches will not occur until the code base of Certification 2 is in the integration branch.

4.1.3 Testing Branch

Cost of Entry: Passes integration tests (Certification 2)

The application as a whole is tested within this branch.

4.1.4 Master

Cost of Entry: Passes all tests.

Product-ready code is found in this branch.

Todo: Would the diagram be sufficient, should we go more in depth of what occurs in each of the branches?

---

5 Test Management

Test Management is accomplished using a variety of tools. All testing artifacts, documents, issues, test cases, and results are stored, verified, and updated using the ZenHub Issue Tracking System.

- Developer technical communications including technical presentations, meeting minutes, and communications with the sponsor will be placed into Google Drive..
- Prior to Sprint 2, Dev Team, Product Team, and QA Team have Read/Write access to test scripts, expected results, and automated tests. After start of Sprint 2, test scripts, expected results, and automated tests can only be modified by Development Team with sign off by Project Manager.
- During test design, tests will be placed under revision control to ensure logging of change history.
- Development Team members have access to individual test results and issue documentation.

Todo: The second bullet it untrue, should we edit it to change when it happens, or get rid of it completely. Currently we're doing a write the expectations as we go kind of deal.

5.1 Test Design

- Team member reviews requirement under test and prepares a test which verifies requirement is met.
- Test cases are mapped to User Stories and Requirements as part of the requirement tracking.
- Test cases are reviewed by Development Team and Project Manager to ensure the test faithfully validates existing requirement(s).
- As required, tests and test cases are reworked, Project Manager signs off, and test is entered into test database.
- Development Team will use prototype, user stories, use cases, and functional specifications to write step by step test cases.
- QA Team to maintain test and issue tracking information to be shared periodically with Project Manager. Change requests or requirement clarifications can cause test cases to be modified, added, or removed as necessary.
- Change requests must be reviewed and accepted by Development Team and Project Manager.

5.2 Executing the Test Plan

- QA Team performs testing tasks as per test plan. Defects are logged using the ZenHub Issue Tracking System. Developer to report the issue is responsible for initial assignment of severity but final determination made by the entire Development Team.
- Product issues related to defects that prevent execution on test plan are reported, logged, and escalated as necessary to the Development Team. e.g. defects causing product features to be unavailable for testing.
- Any defects marked as fixed in a previous test cycle are verified as fixed using test scripts and regression tests.

6 Test Environment and Product Requirements

6.1 External Interfaces and Requirements

- Virtual Reality Headset (Desktop Based)
- 1 Gbps or better network connection

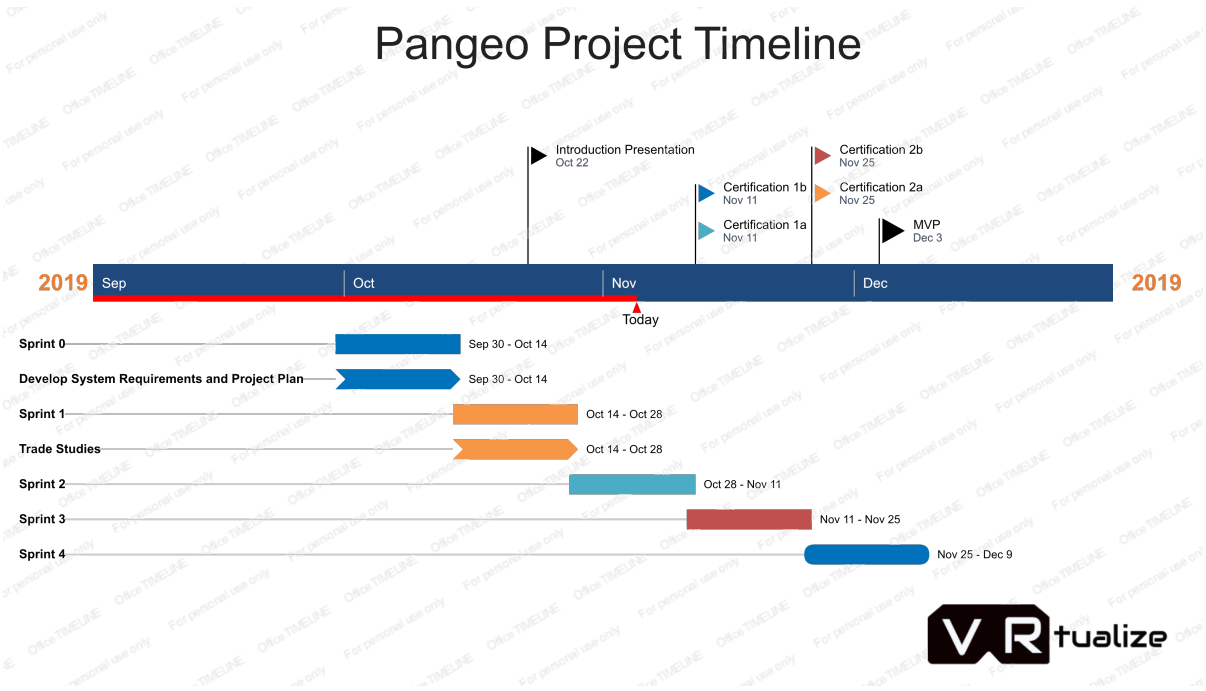
6.2 Hardware

- Quad Core Processor
- 8GB of RAM
- GTX 970/RX 480
- 1 Gbps NIC
- MySQL Database (either hosted on the network or on the local machine)

6.3 Software

- OS: Windows 10
- MySQL Database

7 Project Timeline



Todo: Update to entire year?

8 Roles and Responsibilities

8.1 Project Management

Point of Contact: Jocelyne Freemyer

- **Review, verify, and confirm:**
  - Test Plan
  - Test Strategy
  - Test Estimates

Todo: Add more responsibilities to Project Manager

8.2 QA Team

Point of Contact:

- Initial draft of test plan.
- Process for identifying, recording, and communicating defect reporting.
- Initial draft of issue reporting document (for review by Product Team and Dev Team).
- Acknowledge and communicate test progress and completion for each test cycle.
- Give go-ahead for next test cycle at the completion of each cycle.
- Perform exploratory testing and report, develop, and communicate observed inconsistencies, gaps, or ambiguous requirements.
- Execute tests (test scripts, automated tests, and regression tests as needed)
- Identify, record, and report defects. Provide initial severity rank.
- Produce testing metrics as outlined above.

8.3 Development Team

Point of Contact:

- Review test plan, burndown charts, hotspot charts, test scripts, exploratory findings, automated tests, etc. Sign-off or facilitate modifications as appropriate.
- Deliver agreed on product components as per scheduled dates.
- Communicate barriers to the schedule or product features in a timely manner.
- Implement fixes to defects discovered.

Todo: Assign a point of contact for each of the teams

---

9 Team Members and Contact Info

Position	Name	Contact Info
Project Manager	Jocelyne Freemyer	Jocelyne.Freemyer@mines.sdsmt.edu
Developer	Isaac Egermier	Isaac.Egermier@mines.sdsmt.edu
Developer	Chezka Gaddi	Chezka.Gaddi@mines.sdsmt.edu
Developer	Michael Theesen	Michael.Theesen@mines.sdsmt.edu
Developer	Garfield Tong	Garfield.Tong@mines.sdsmt.edu

10

Sign Off

Name:	Isaac Egermier	Jocelyne Freemyer	Chezka Gaddi	Michael Theesen	Garfield Tong
Signature:					
Date:					