Test Plan Pangeo by VRtualize

Release 1.3

Isaac Egermier
Chezka Gaddi
Garfield Tong
Michael Theesen

Dec 10, 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 QA testing, branch structure, issue life-cylce, and project timeline; changed issue management tool.
3	December 09, 2019	Chezka Gaddi	Added effort estimation, modified the project timeline, changed Certification descriptions, added regression test description.

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.
- Implementation: How testing is performed, defect management process (reporting, fix planning, and execution).
- Management: Description of the logistics of testing.

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

The intended audience of the test plan is the Project Manager, Development team, and QA team. Some portions of this document may on occasion be shared with the client/user and other stakeholder whose input/approval into the testing process is needed.

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. 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.
- 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 1	Image Retrieval Retrieval of elevation data and satellite imagery from external sources.
Certification 2	Terrain Rendering Generation of a 3D virtual model of terrain.
Certification 3	User Interface Display and navigation of the user through the application features.
Certification 4	Image Caching Chunk prediction and the management of cached data.
Certification 5	User Experience User movement interactions with the terrain model.

- 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 Testing Scope

2.3.1 Unit

Who: Development Team

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

2.3.2 Back-end

Who: Development Team

When: During database setup

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

Scope: Database

How: Tests will be developed as SQL queries

2.3.3 Code Reviews

Who: Development Team

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

Why: To ensure that the code upholds 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

2.3.4 QA Testing

Who: QA Team

When: After units of code passes code reviews

Why: To ensure units of code pass user acceptance criteria

Scope: All units of code

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

2.3.5 Integration

Who: Development Team

When: Combining individual units of code

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

2.3.6 Regression

Who: Development Team

When: When a defect is discovered

Why: To make ensure the reproduction of the failing case and recognize when that failure is fixed

Scope: Specific to the defect

How: Depending on the type of defect found, the regression test could be a User Acceptance Test or use the same format as a unit test

2.3.7 System and Functional

Who: QA Team

When: Prior to Exploratory Testing

Why: Thorough testing of all application functions

Scope: All required features of the application as described in the SRS

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

2.3.8 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.

2.3.9 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

2.3.10 Exploratory and Alpha

Who: Alpha testers

When: After functional tests

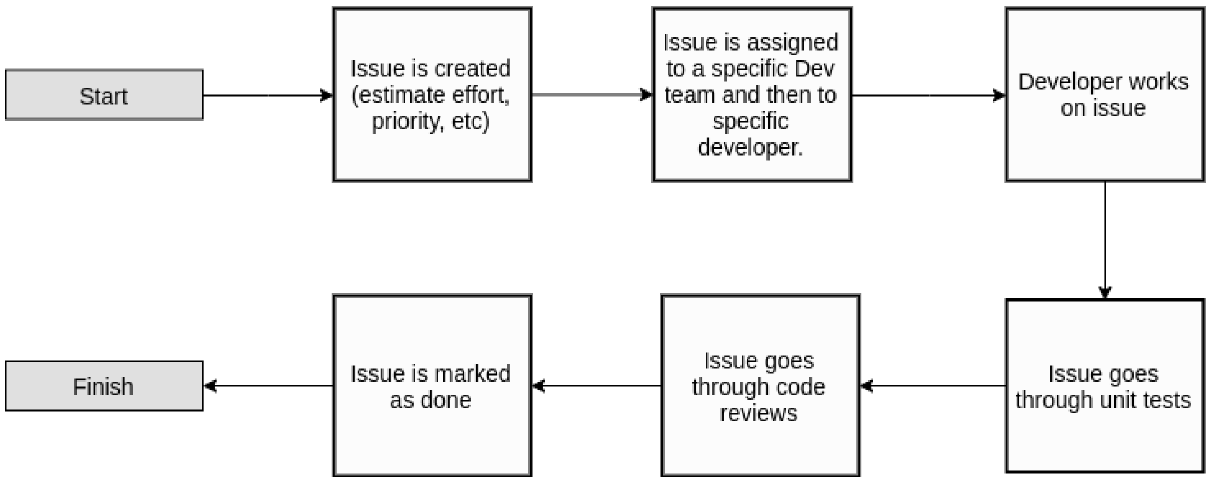
Why: Primarily to familiarize the alpha testers with the features and behavior of the software to set expectations for new features and identify any hiccups

Scope: Production level product

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

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

3 Issue Management

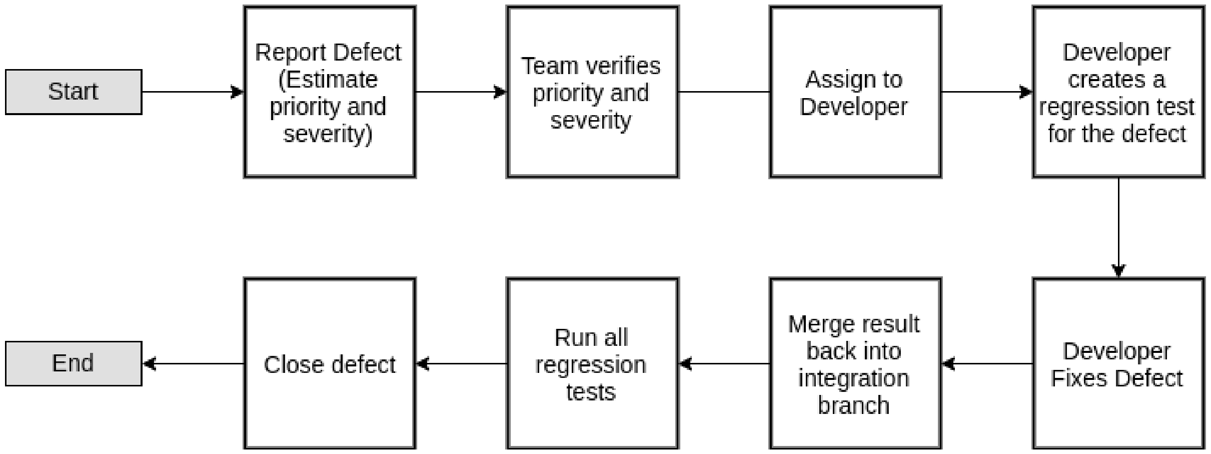

Start
Issue is created (estimate effort, priority, etc)
Issue is assigned to a specific Dev team and then to specific developer.
Developer works on issue
Issue goes through unit tests
Issue goes through code reviews
Issue is marked as done
Finish

3.1 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 pipeline. Team members working on the same Certification subsections 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

3.2 Defect Reporting

3.2.1 Bug Life-cycle


Start
Report Defect (Estimate priority and severity)
Team verifies priority and severity
Assign to Developer
Developer creates a regression test for the defect
Developer Fixes Defect
Merge result back into integration branch
Run all regression tests
Close defect
End

1. Bug or defect is reported and the founder will estimate the priority and severity
2. Defects 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.
3. The bug will be assigned to a developer based on who's code the bug was found in.
4. The developer will develop a regression test for the bug and add it to the test database.
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 the bug was fixed.
8. Defect is closed.

3.2.2 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
4 (Low)	• Minor feature not working as per requirements but functionality is testable using workaround

3.3 Metrics

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

- Weekly Status Report: Includes weekly pass/fail/complete percentages. Identify and troubleshoot any defects in the Critical category which have persisted for over a week.
- Sprint End Report: Compile trajectory graphs for defect lists broken out by status, severity, and age.

3.4 Start and End Criteria

- Start criteria detailed in the *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.
- 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 ZenHub'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 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.
- 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.

4.1 Test Design

- Team member reviews requirement under User Story and prepares tests which verifies requirement is met.
- Test cases are mapped to User Stories and Requirements as part of requirement tracking.
- Test cases are reviewed by Development Team to ensure the test faithfully validates existing requirement(s).
- Development Team will use prototypes, user stories, use cases, and functional specifications to write step by step test cases.
- QA Team will 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.

4.2 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 defect 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.

4.3 Risks and Risk Response

Risk	Likelihood	Effect	Response
Resource availability	Low	Unable to receive information we need to render a specific location. Users will no longer be able to use the application that requires information that we don't currently have for the next 24 hours.	Ensure proper storage of repeatedly used testing resources, and understanding capabilities of API token reuse to minimize new API calls.
Unforeseen Delays	Low	Impossible to tell the impact due to not knowing the issue. It could potentially create a time slip and we will not be able to deliver the product that we have agreed to produce for L3Harris, which may impact future projects with the company.	Work will be scheduled the highest priority in the next Sprint. During Christmas break there will be some time to make up for time slips that may occur to get the project back on schedule for development for Certification 3.

5 Test Environment and Product Requirements

5.1 External Interfaces and Requirements

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

5.2 Hardware

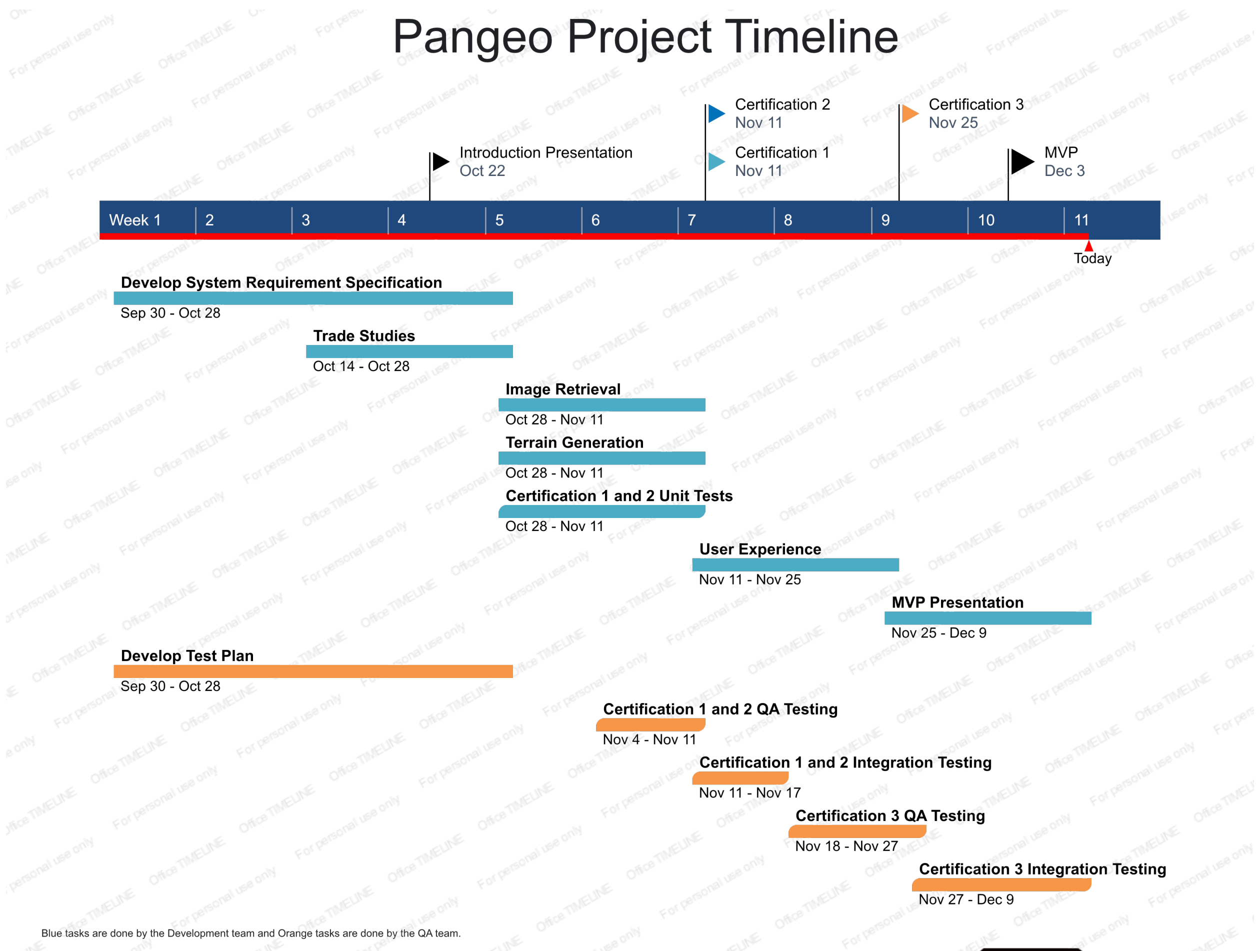
- Quad Core Processor
- 8GB of RAM
- GTX 970/RX 480
- 1 Gbps NIC
- MySQL Database (either hosted on the network or on the local machine)

5.3 Software

- OS: Windows 10
- MySQL Database

6

Project Timeline



Task Dates

~ **September 30 - October 28** Develop System Requirement Specification

~ **September 30 - October 28** Develop Test Plan

~ **October 14 - October 28** Trade Studies

~ **October 22** Introduction Presentation

~ **October 28 - November 11** Image Retrieval

~ **October 28 - November 11** Terrain Generation

~ **October 28 - November 11** Certification 1 and 2 Unit Tests

~ **November 4 - November 11** Certification 1 and 2 QA Testing

~ **November 11** Certification 1

~ **November 11** Certification 2

~ **November 11 - November 17** Certification 1 and 2 Integration Testing

~ **November 11 - November 25** User Experience

~ **November 18 - November 27** Certification 3 QA Testing

~ **November 25** Certification 3

~ **November 25 - December 9** MVP Presentation

~ **November 27 - December 9** Certification 1,2, and 3 Integration Testing

~ **December 3** MVP Release

7 Effort Estimation

The effort in work hours for the entirety of the Pangeo project is estimated below. This estimate is calculated by breaking down the tasks needed to be accomplished for the project.

Fall 2019

Documentation SRS Test Plan	80
Technology Research	60
Image Retrieval Database Creation API Calls Database Data Retrieval	80
Terrain Rendering Conversion of Map Data to 1D array Creation of Mesh Stitching Multiple Meshes Generation of 3D terrain in Unity	80
User Interface Creation of Menu Binding Actions to Controller Navigation to Different Functionalities	20
	320 work hours

Spring 2020

Documentation Technical Hand-off	80
Technology Research	20
Data Caching Chunk Prediction Cache Maintenance Map Requests	80
Terrain Rendering Satellite Imagery Overlay Varying Map Chunk Resolution	40
User Interface Movement of User Around the Application	40
Testing Alpha and Beta Testing Performance Testing	140
	400 work hours

The VRtualize team is able to put in 10 hours per week for a total of 19 weeks which comes out to be 760 work hours, which is sufficient time to finish the project by March 27th, 2020.

8 Roles and Responsibilities

8.1 Project Management

Point of Contact: Jocelyne Freemyer

- Liason for stakeholders
- Assist in Sprint Planning
- Facilitate team building activities
- Review, verify, and confirm:
  - User Stories
  - Test Plan
  - Test Strategy
  - Test Estimates

8.2 QA Team

Point of Contact: Chezka Gaddi

- 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)
- Generate burndown charts and cumulative flow charts after each sprint
- Identify, record, and report defects

8.3 Development Team

Point of Contact: Isaac Egermier

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

9 Team Members and Contact Info

Position	Name	Contact Info
Project Manager	Jocelyne Freemyer	Jocelyne.Freemyer@mines.sdsmt.edu
Development Point of Contact	Isaac Egermier	Isaac.Egermier@mines.sdsmt.edu
Quality Assurance Point of Contact	Chezka Gaddi	Chezka.Gaddi@mines.sdsmt.edu
Repo Manager	Michael Theesen	Michael.Theesen@mines.sdsmt.edu
Architecture Point of Contact	Garfield Tong	Garfield.Tong@mines.sdsmt.edu

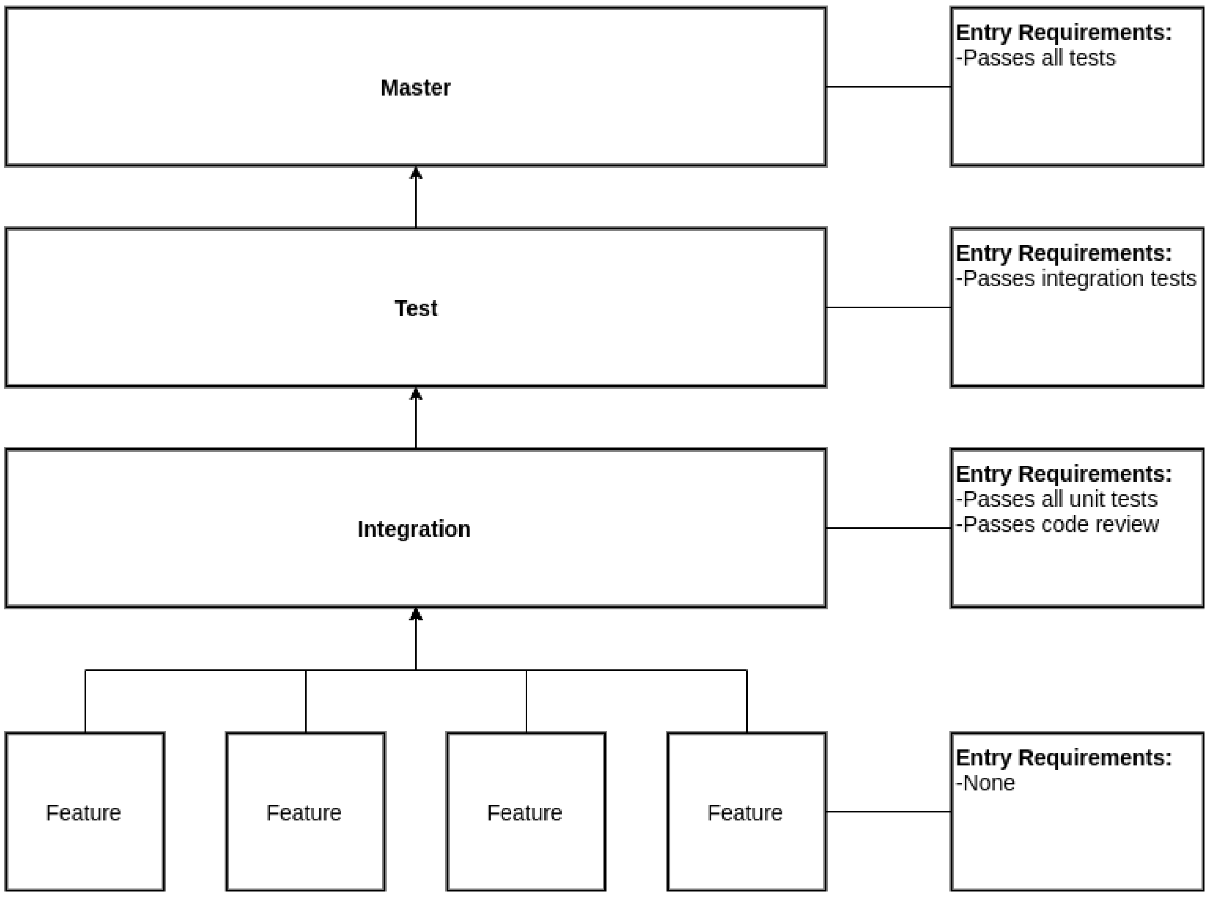
10 Sign Off

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

A 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.

A.0.1 Branch Structure



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.

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.

Testing Branch

Cost of Entry: Passes integration tests (Certification 2)

The application as a whole is tested within this branch.

Master

Cost of Entry: Passes all tests.

Product-ready code is found in this branch.

B Issue Tracking Manager

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

B.0.1 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.

B.0.2 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 desired but not required.
- **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 have passed review and QA tests and can be queued for next release.
- **Closed** Issues that will no longer be worked on (Done, won't fix, done with development, already implemented, duplicate, etc.).

B.0.3 Issue Types

- User Story
- Feature Request
- UX Issue
- Performance
- Bug

B.0.4 Issue Complexities

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