Software Test Plan

Capstone Project

Version: 0.5
Status: Draft

Table of Contents

Revision history

Version	Date	Revised by	Description	Approved by
0.1	Oct 21st, 2014	Humza Tariq, Dylan Aspden, Kurt DaCosta, Aman Uppal	Set up skeleton template	
0.2	Oct 22nd, 2014	Humza Tariq, Dylan Aspden, Kurt DaCosta, Aman Uppal	Filled out various sections of the document	
0.3	Oct 24th, 2014	Humza Tariq, Dylan Aspden	Filled out additional sections of the document	
0.4	Oct 25th, 2014	Aman Uppal	Ported test plan to new format	
0.5	Oct 25th, 2014	Humza Tariq, Aman Uppal	Finalized content for team review	

1 Purpose of this document

This software test plan covers the testing methodology at a high level. It will be used throughout the rigorous testing of our application.

The purpose of this document is to:

- Define and document the various types of testing which will be conducted
- Define and document the various types of testing strategies which will be employed

2 Stakeholders & Distribution

Stakeholder Name	Role
Dr. Woolhouse	External Client Lead
Nick Rogers	DML Member
Dr. Down	Capstone Supervisor
Dr. Zheng	Course Supervisor
John Ernsthausen	Teaching Assistant

3 Objectives

3.1 Purpose

This document describes the proposed plan for testing the ViMDa (Visualization of Music Data) system. This test plan document supports the following objectives:

- Identify project information and software components that should be tested
- Come up with a high level list of recommended test requirements
- Identify the testing strategies that project will make use of and briefly describe them
- Establish resources required for testing and provide an estimate for its completion

3.2 Scope

The scope of the test plan will cover the testing of both the functional and non-functional requirements against our application.

Out of scope items include extensive user experience testing and all types of testing which are not documented below.

The interfaces between the following subsystems will be tested:

1. Data Pipeline (Machine Learning Worker)
2. Application Server (API Server)
3. Data Layer (DB Server)
4. Job Queue (SOA Messaging)

The external interfaces to the following devices will be tested:

1. Web Application

The most critical performance measures to test are:

1. Response time for database queries (pre-clustering)
2. Completion time for manipulating data (pre-clustering)
3. Completion time for clustering algorithm (time and space complexity)
4. Network transfer time for requesting clustered data sets (post-clustering)
5. Response time for web application visual rendering (how long it takes on different systems)

3.3 Reference Material

1. Business Requirements Document – ViMDa
2. Test Plan Template - LINK [http://sce.uhcl.edu/helm/RUP_course_example/courseregistrationproject/artifacts/test/plans/test_plan_arch.htm]

4 Testing Breakdown

Below you will find the types of testing that will be performed on our application. Any numbers found in brackets denote the requirement number for a requirement, which can be found in the BRD.

4.1 Unit Testing

- Verify all possible paths in a module are successful prior to each deployment

4.2 System Testing

- System testing will be determined in a later revision of the document in the form of multiple test cases once we have a better idea of the expected workflow

4.3 Smoke and Sanity Testing

- Verify that application passes smoke testing checklist (document to be created) after each deployment

4.4 Regression Testing

- Verify if the bug that was fixed does not persist in further revisions
- Verify that all dependent modules of the fix/update are still functioning as expected

4.5 Functional Testing

- Verify client will communicate securely with application server via API requests (BRD, 1.1.1)
- Verify that system will allow graphics acceleration to be performed on client's hardware (BRD, 1.1.2)
- Verify that the system will audit all user activity (BRD, 1.1.3)
- Verify that system will allow the user to view audited user activity (BRD, 1.1.4)
- Verify that system will allow the user to perform a request to visualize certain data (BRD, 1.2.1)
- Verify that system will allow the user to select constraints, features and tables to visualize using the provided interface (BRD, 1.2.2)
- Verify that system will update the user on time remaining for the processing to complete (BRD, 1.2.3)
- Verify that system will display a visual/sound notification on-screen when the processing is complete (BRD, 1.2.4)
- Verify that the analyzed cluster will produce visualizations in 3D and 2D graphs (BRD, 1.3.1)
- The system will allow users to interact with the interface and explore the visualized cluster in a friendly manner (BRD, 1.3.2)

- Verify communication of application server via API layer (BRD, 2.1.1)
- Verify that system will accept requests for analyzed data sets (BRD, 2.1.2)
- Verify that system will accept requests for submitting machine learning jobs (BRD, 2.1.3)
- Verify that system will validate the requests for machine learning jobs (BRD, 2.1.4)
- Verify that system will submit machine learning jobs to job queue (BRD, 2.1.5)
- Verify that system will serve static web assets and web application source code to web clients (BRD, 2.1.6)
- Verify that the system will accept jobs via the API layer (BRD, 3.1.1)
- Verify that system will distribute jobs between Machine Learning Workers (BRD, 3.1.2)
- Verify that system will notify the application server of the job status (BRD, 3.1.3)
- Verify that the system will provide mechanisms for retry attempts (BRD, 3.1.4)
- Verify that system will accept connections from application server (BRD, 4.1.1)
- Verify that system will accept connections from workers (BRD, 4.1.2)
- Verify that system will store analyzed data (BRD, 4.1.3)
- Verify that system will accept machine learning jobs from job queue (BRD, 5.1.1)
- Verify that system will implement data pipeline for all stages of machine learning (BRD, 5.1.2)
- Verify that Data pipeline will be structured as a Directed Acyclic Graph (DAG) (BRD, 5.1.3)
- Verify that system will parse job metadata to create the correct path in the data pipeline (BRD, 5.1.4)
- Verify that the system will submit jobs into data pipeline (BRD, 5.1.5)
- Verify that system will pre-analyze data manipulation (data pipeline stage) (BRD, 5.1.6)
- Verify that system will perform machine learning algorithms (data pipeline stage) (BRD, 5.1.7)
- Verify that system will precompile data compression (data pipeline stage) (BRD, 5.1.8)
- Verify that the system will store analyzed data and job metadata on database (BRD, 5.1.9)
- Verify that system will notify job queue of job progress (BRD, 5.1.10)
- Verify that system will notify job queue on job completion (BRD, 5.1.11)
- Verify that system will notify job queue on job failure (BRD, 5.1.12)

4.6 Non-functional Testing

- Verify target resolution of 1920x1080 works with ViMDa (BRD, 6.1.2)
- Verify ViMDa's front end works with Chrome (BRD, 6.1.3)
- Verify SQL statement validation occurs (BRD, 10.1.1)
- Verify deployment of ViMDa has no impact on the master DB (BRD, 12.1.1)
- Verify deployment will halt any work being done on ViMDa (BRD, 12.1.5)

4.7 Performance Testing

- Verify response time to access external Data Pipeline
- Verify response time to access external Application Server
- Verify response time to access external Data Layer
- Verify response time to access external Job Queue
- Verify response time, if not specified otherwise is up to 5 seconds 95% of the time (BRD, 7.1.1)
- Verify client side rendering time is up to 3 seconds (BRD, 7.1.2)
- Verify cluster formation time is up to 5 minutes for 80% of the clusters to be formed (BRD, 7.1.3)
- Verify query execution time is up to 1 minute 95% of the time (BRD, 7.1.4)

4.8 Load Testing

- Verify data pipeline completion time for 1 running jobs
- Verify job queue response time for 1 concurrent requests
- Verify job queue response time for 10 concurrent requests
- Verify application server response time for 1 request of clustered data set
- Verify application server response time for 2 request of clustered data set
- Verify application server response time for 5 request of clustered data set

5 Testing Types & Strategy

5.1 Unit Testing

Unit testing ensures all possible paths that can be exercised within all modules (independent of one another) are successful. Unit testing and its success is important because without both the integrity of the application would be in question.

Test Objective	Ensure that all possible paths which can be exercised within a module are successful.
Technique	Exercise each possible path which can be taken in each individual module (independent of the rest of the system as possible).
Completion Criteria	• All unit tests are successful or; • Any potential defects were logged in our bug tracking system
Special	TBD.

Considerations	

5.2 System Testing

Test Objective	TBD - Refer to section 2.2 above.
Technique	See above.
Completion Criteria	See above.
Special Considerations	See above.

5.3 Smoke and Sanity Testing

Smoke and sanity testing ensures that the core functionality of the application is still working as expected after a new build has been created. Smoke and sanity testing is conducted to test the basic functionality of the current build. If the smoke and sanity test fails, then the entire build is rejected and is looked into to determine why the test has failed.

Test Objective	Ensure that all core functionality of our application passes the smoke and sanity test.
Technique	Execute each test case which is part of the smoke and sanity testing checklist.
Completion Criteria	• All test cases which are part of the smoke and sanity testing checklist passed successfully or; • Any potential defects were logged in our bug tracking system
Special Considerations	TBD.

5.4 Regression Testing

Regression testing ensures that changed functionality does not affect unchanged functionality. In addition, if any bug was fixed, ensure that the previously failed test case(s) pass the second time around.

Test Objective	Ensure that all unchanged functionality does not get effected by changed functionality in our application.
Technique	Test all functionality within a module which has not been changed; in addition, test all functionality within dependent modules to ensure behaviour is still as expected.
Completion Criteria	• All unchanged functionality still function as they did previously before the revisions were made or; • Any potential defects were logged in our bug tracking system
Special Considerations	TBD.

5.5 Functional Testing

Functional testing ensures that all functional aspects that can be derived from the BRD are met as documented.

As an example, if a functional requirement in the BRD is: The user should be able to select and copy text to the clipboard.

Through functional testing we will ensure that:

- The user can select text
- The user can access the copy option
- The user can select the copy option
- The text is copied to the clipboard

Test Objective	Ensure that all functional aspects that can be derived from the BRD for our application are met as documented.
Technique	Execute all test cases for our functional requirements (to be drafted out).
Completion Criteria	• All functional test cases passed or; • Any potential defects were logged in our bug tracking system
Special Considerations	TBD.

5.6 Non-Functional Testing

Non-functional testing ensures that all non-functional aspects that can be derived from the BRD are met as documented.

As an example, if a non-functional requirement in the BRD is: The application will support up to two monitors for the display of the application.

Through non-functional testing we will ensure that:

- The application is properly displayed with the use of only one monitor
- The application is properly displayed with the use of two monitors

Test Objective	Ensure that all non-functional aspects that can be derived from the BRD for our application are met as documented.
Technique	Execute all test cases for our non-functional requirements (to be drafted out).
Completion Criteria	• All non-functional test cases passed or; • Any potential defects were logged in our bug tracking system
Special Considerations	TBD.

5.7 Performance Testing

Performance testing ensures that all anticipated performance targets are met successfully by the application.

Test Objective	Ensure that all anticipated performance targets are met.
Technique	Time how long it takes to complete various activities that we have performance targets set for while executing them.
Completion Criteria	• Anticipated performance targets are met or; • Any potential defects were logged in our bug tracking system
Special Considerations	TBD.

5.8 Load Testing

Load testing ensures that the application can perform at a defined acceptable level while handling various types and magnitudes of load.

As an example:

- An application can be tested with inputs: 500 of type x and 100 of type y, and the behaviour of the application will be monitored throughout to see if it is behaving as expected under this load

- An application can be tested with inputs: 5000 of type x and 1000 of type y, and the behaviour of the application will be monitored throughout to see if it is behaving as expected under this load

Test Objective	Ensure that our application functions as expected while experiencing various types of load.
Technique	Monitor our application behaviour, stability and responsiveness while dealing with various loads put onto it.
Completion Criteria	• Our application performs at a defined acceptable level under various forms of load or; • Any potential defects are logged in our bug tracking system.
Special Considerations	TBD.

6 Tools & Testing Resources

The tools we will be using for our testing and the management and tracking of testing are listed below:

Category	Tool	Version
Test Management	Microsoft Word	2011 (Mac)
Test Design	Microsoft Word	2011 (Mac)
Defect Tracking	JIRA or Bugzilla (TBD)	TBD
Project Management	Microsoft Word Microsoft Excel Microsoft Project	2011 (Mac) 2011 (Mac) 2010 (Windows)
DBMS Tools	TBD	TBD

All capstone members will play varying roles in the testing of our application which are listed below:

Role	Capstone Member(s)	Specific Responsibilities
Test Manager	Humza Tariq	- Oversee testing of the application - Provide guidance as required

Test Designer	Capstone Team	- Humza will design all black box test cases with the help of Aman - Rest of the team will design all white box test cases
Black Box Tester	Capstone Team	- Everyone will test the application to various degrees from a black box point of view
White Box Tester	Dylan Aspden, Kurt DaCosta, Aman Uppal	- Test the modules they have individual ownership of to ensure it is functioning as expected

7 Project Milestones

A project plan will be created which will incorporate a WBS (Work Breakdown Structure) in Microsoft Project. A subset of those tasks pertain to the testing of ViMDa, which can be found below:

Milestone Task	Effort (in days)	Start Date	End Date
Test Plan Revision 0	6	Oct 21st, 2014	Oct 26th, 2014
Test Report Revision 0	TBD	Q1 2015	March 6th, 2014
Test Plan (Final)	TBD	Q1 2015	April 10th, 2014
Test Report (Final)	TBD	Q1 2015	April 10th, 2014