**Project Specification Document**

IBM

Scott Rotvold

Madeline Gordon

Brandon Ebersohl

Adam Murray

Table of Contents

Section Page

[Table of Contents ii](#_Toc409000255)

[1. Project Vision and Objectives 1](#_Toc409000256)

[1.1 Project Scope and Vision 1](#_Toc409000257)

[1.2 Project Goals and Objectives 1](#_Toc409000258)

[2. Project Planning 2](#_Toc409000259)

[2.1 Project Lifecycle 2](#_Toc409000260)

[2.2 Project Setup 2](#_Toc409000261)

[2.3 Stakeholders 2](#_Toc409000262)

[2.4 Project Resources 3](#_Toc409000263)

[2.5 Assumptions 3](#_Toc409000264)

[2.6 Estimates 3](#_Toc409000265)

[3. Project Tracking 5](#_Toc409000266)

[3.1 Tracking 5](#_Toc409000267)

[3.2 Communication Plan 5](#_Toc409000268)

[Regularly Scheduled Meetings 5](#_Toc409000269)

[Information To Be Shared Within Our Group 5](#_Toc409000270)

[Information To Be Provided To Other Groups 6](#_Toc409000271)

[Information Needed From Other Groups 6](#_Toc409000272)

[3.3 Deliverables 6](#_Toc409000273)

[3.4 Project Metrics 6](#_Toc409000274)

[4. Requirements 8](#_Toc409000275)

[4.1 Overall Description 8](#_Toc409000276)

[4.2 Users and Roles 8](#_Toc409000277)

[4.3 Use Case Diagrams 8](#_Toc409000278)

[4.4 User Stories (Requirements) 8](#_Toc409000279)

[4.5 User Story Acceptance Criteria 9](#_Toc409000280)

[4.6 Constraints and Limitations 10](#_Toc409000281)

[5. Design 12](#_Toc409000282)

[5.1 Introduction 12](#_Toc409000283)

[5.2 Scope 12](#_Toc409000284)

[5.3 High-Level Component Design 12](#_Toc409000285)

[5.4 Activity Diagrams 12](#_Toc409000286)

[5.5 Class Diagram 12](#_Toc409000287)

[5.6 Sequence Diagram 13](#_Toc409000288)

[5.6 Data Architecture 13](#_Toc409000289)

[5.7 Alternative Designs and Design Rationale 13](#_Toc409000290)

[5.8 Low-Level Component Design 13](#_Toc409000291)

[Class 1 13](#_Toc409000292)

[Description 13](#_Toc409000293)

[Variables 13](#_Toc409000294)

[Methods 14](#_Toc409000295)

[Class 2 14](#_Toc409000296)

[Description 14](#_Toc409000297)

[Variables 14](#_Toc409000298)

[Methods 14](#_Toc409000299)

[6. User Interface 16](#_Toc409000300)

[6.1 UI Description 16](#_Toc409000301)

[6.2 UI Mockup 16](#_Toc409000302)

[7. Test Plan 17](#_Toc409000303)

[7.1 Test Plan Description 17](#_Toc409000304)

[7.2 Testing Tools 17](#_Toc409000305)

[7.3 Test Data 17](#_Toc409000306)

[7.4 Test Types and Frequency 17](#_Toc409000307)

[7.5 Test Coverage 17](#_Toc409000308)

[7.6 Test Location 18](#_Toc409000309)

[8. Project Closure 19](#_Toc409000310)

[8.1 Goals / Vision 19](#_Toc409000311)

[8.2 Delivered Solution 19](#_Toc409000312)

[8.3 Remaining Work 19](#_Toc409000313)

[9. Deliverables 20](#_Toc409000314)

[9.1 Study Results 20](#_Toc409000315)

[9.2 Requirements and Design Documents 20](#_Toc409000316)

[9.3 Code 20](#_Toc409000317)

[9.4 Tests and Test Results 20](#_Toc409000318)

[9.5 Build Process Documents 20](#_Toc409000319)

[9.6 Install Process Documents 20](#_Toc409000320)

[9.7 Administrator’s and/or User’s Manual 20](#_Toc409000321)

[9.8 Postmortem Document 20](#_Toc409000322)

[9.9 Final Report 20](#_Toc409000323)

[10. Definitions and Acronyms 21](#_Toc409000324)

1. Project Vision and Objectives

1.1 Project Scope and Vision

The purpose of this project is to create a real-time display of metrics to be viewed in a dashboard via web browser. First, we will research into different technologies needed for the process of collecting and storing metrics. We will store these metrics in a graphing database and gather them using our code written in Python. We hope to create a user-friendly and customizable portal for displaying these metrics that will assist in monitoring and evaluating the health and availability of systems infrastructure. The main issue being addressed involves the large size of IBMs environment and the need for a centralized dashboard to display their metrics.

1.2 Project Goals and Objectives

|  |  |
| --- | --- |
| **#** | **Goal or Objective** |
| 1 | Create a Jenkins test environment with sample data |
| 2 | Create a portal (dashboard) that allows users to view the gathered metrics in either graph form or other representations |
| 3 | Create a user-friendly GUI that is simple to learn and understand |
| 4 | Create a dashboard that gives the user control over configuration details |
| 5 | Create a flexible framework that can be deployed within IBM’s Cloud Solution’s development infrastructure |

2. Project Planning

2.1 Project Lifecycle

The team will use an agile approach. Our sponsors have outlined our project with three different phases. Our team will research the different technologies that we could use for this project to create a development plan that will be best for our project. We will work in iterations while continuously communicating with each other and with our sponsors.

2.2 Project Setup

|  |  |
| --- | --- |
| **#** | **Decision Description** |
| 1 | GitHub, Windows & Linux, Jenkins, Graphite v. Grafana, Python |
| 2 | Standards that must be followed: Default capstone coding standards |
| 3 | Access: Open Source, testing with “dummy data” |
| 4 | Server setup: Provided by NDSU |

2.3 Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role** |
| Person A | Sponsor: Jake Morlock |
| Person B | Sponsor: Mathew Odden |
| Person C | Sponsor: Adam Reznechek |
| Person D | Instructor: Dean Knudson |
| Person E | Team member: Scott Rotvold |
| Person F | Team member: Adam Murray |
| Person G | Team member: Madeline Gordon |
| Person H | Team member: Brandon Ebersohl |

2.4 Project Resources

|  |  |  |  |
| --- | --- | --- | --- |
| **Resource** | **Resource Description** | **Quantity** | **Total** |
| NDSU Server | A database server provided by the NDSU with the help of Nate Olson | 1 | $0.00 |
| Capstone Team | Our team of students who will be the primary developers of the project. | 4 | $ 0.00 |
| Jenkins | Open-Source software for continuous integration | 4 | 0.00 |
| Personal Computers | Our team will use our personal computers to access Jenkins and write the code for our project | 4 | $0.00 |

2.5 Assumptions

|  |  |
| --- | --- |
| **#** | **Assumption** |
| A1 | We will have all of the preliminary research and setup completed by February 7th |
| A2 | We will begin writing our code and setting up our test data by February 9th |
| A3 | We will present our work for mid-term presentations on March 10th |
| A4 | We will complete our dashboard model by April 1st and begin testing for bugs |
| A5 | We will set up our project for automated deployment by April 15th |
| A6 | We will have our project ready for a final presentation on May 5th |

3. Project Tracking

3.1 Tracking

|  |  |  |
| --- | --- | --- |
| **Information** | **Description** | **Link** |
| Code Storage | e. g. Project code will be stored in a git-hub account. | https://github.com/the0ldknighte/  IBM\_Capstone\_Project |
| Bug Tracking | e. g. Bug tracking will be done with Trac. | https://csprojects.cs.ndsu.nodak.edu/csci445/  2015/spring/csci445s15ibm/trac |
| Project Schedule | e. g. The project schedule will be stored in the git repository. | https://github.com/the0ldknighte/  IBM\_Capstone\_Project |
| Continuous Integration | e. g. Continuous integration will be done with Jenkins. |  |
| Regression Testing | e. g. Regression testing will use JUnit unit tests and Jenkins. |  |

3.2 Communication Plan

Regularly Scheduled Meetings

|  |  |  |
| --- | --- | --- |
| Meeting Type | Frequency/Schedule | Who Attends |
| Conference Call | Wednesday at 8 a.m. (Weekly) | Project team and mentors |
| Team Meeting | As needed (Weekly) | Project team |
| Short Meeting | Weekly in class | Project team |

Information To Be Shared Within Our Group

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Project team | Task assignments | Weekly in class | Team meetings, listing in MS Project file. |
|  |  |  |  |

Information To Be Provided To Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Sponsor and mentor | Final report | At completion of project | Req./Design docs., code, Power Point presentation |
| Sponsor and mentor | Weekly report | Weekly | Email and Git site access |

Information Needed From Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| Who? | What Information? | When? | How? |
| Sponsor and mentor | Requirement changes | Start of each sprint | Conference call or meeting with sponsor and mentor. |
| Nate Olson | Availability of test server | As Needed | Email or in person |

3.3 Deliverables

|  |  |  |  |
| --- | --- | --- | --- |
| # | Deliverable | Delivery Time | |
|  |  | Interim | Final |
| 1 | Prototype Code |  |  |
| 2 | “Dummy Data” |  |  |
| 3 | Deployable product |  |  |
| 4 | Project Documentation |  |  |

3.4 Project Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Frequency | Location |
| Estimated User Story Points | Per User Story at the start of each sprint | MS Project Plan |
| Estimated User Story Points | Per Sprint at the start of each sprint | MS Project Plan |
| Actual User Story Points | Per Sprint at the end of each sprint | MS Project Plan |

4. Requirements (User Stories)

4.1 Overall Description

<< Provide a more detailed, two to three paragraph description of the project. This description may include more technical details to describe the purpose of the project. >>

The primary focus of this project is the client (IBM) wishes to gather important metrics on how their Jenkins setup is working for their Open Stack project. Jenkins is as system primarily used for continuous integration of updates into a large software project. Since Open Stack is a large and open source project, there are large quantities of source updates submitted every day, which have areas that could be tracked using metrics. These metrics may include such traceable aspects as time to commit, time to merge updates, type of function and many others. Jenkins already has ways to find these metrics, however, IBM wants to track them in real time and store data that is displayed in a user friendly way using open source graphing software such as graphite.

The graphite system provides a detailed and easy to understand layout of those metrics so that the users at IBM can pinpoint where they can improve the continuous integration process. The graphite system works using a design pattern as shown below.



**Image taken from the Graphite website:** [**http://graphite.readthedocs.org/en/latest/faq.html**](http://graphite.readthedocs.org/en/latest/faq.html)

This metric tracking and displaying system will then be hosted on an apache webserver which will display the required metrics as well as archive old metrics to create more details historical records.

There is a possibility and requested addition to make the project self-deployable and a one file install using a system called Chef. Chef will provide the user with a all in one setup package to provide an easy setup for the users. This will make the deployment much less of a hassle and will lower the required level of knowledge for the person administering the metric system to their Jenkins database. This feature is however, an optional feature and will not be implemented until the core metric system has been completed to the customer’s satisfaction.

4.2 Users and Roles

<< Provide a list and description of the different types of users or roles within the system. This may include different classes of users, such as administrator, instructor, student, etc. This list may also include autonomous agents that interact with the system as well. These may include users (or personas) that are used as part of any user stories produced for the project. >>

|  |  |
| --- | --- |
| **User** | **Description** |
| User | A regular user who can view the data and see the graphical displays |
| Admin | A user who also has the ability to edit and update historical data as well as edit the display of data |
|  |  |
|  |  |
|  |  |

4.3 Use Case Diagrams

<< Provide any use-case diagrams that are being used as part of the project. Uniquely label each use case so that if necessary it is easy to reference from other parts of the document. >>

User and Admin Running System Use Case



Install Project Use Case



4.4 User Stories (Requirements)

<< This section lists the user stories for the project, when they were added, and information about which of these user stories the team has committed to complete (C), stories that will be targeted but not guaranteed to be completed (T), and those which the team will not commit to (NC).

It is preferred to have user stories written in the form - *As an [actor] I want [action] so that [achievement].* So, for example: *As a Bison Tracker member, I want to set different privacy levels on my photos, so I can control who sees which of my photos.*

The **ID** column provides a high-level ID for each user story. This is useful for generating acceptance criteria IDs which can be easily associated with a particular user story. The **Added** column lists when the user story was added to the project as it is possible that not all requirements will be elicited or available at the project’s onset. The **Description** column gives a description of the user story. The **Status** column indicates whether these user stories are Committed (i.e. C will be completed by the team), Targeted (i.e. T will be completed if the team has time after all committed requirements are completed), or Not Committed (i.e. NC will not be completed by the team). The **Story Points** indicate how much work is associated with each story.

Story points are assigned at the start of each sprint for those user stories that are assigned to that sprint. The default is to use values of 1, 2, 3, 5, 8, 13, 21, 34, and 45. Anything bigger than 45 should be made into multiple stories.

Each team should benchmark their story points and indicate what was used as the benchmark. For example, everyone in the group agrees that the sample Bison Tracker story above is a 3-point story, so the points for all other stories are allocated in relation to that benchmark.>>

**User Story Points Benchmark** – All team members agree that creating a 3D model of a helicopter for use in this system would be worth 5 points. All team members have had previous experience building 3D models from their gaming class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** |
| 100 | Onset | As a user I need to know and understand the technologies I am using | **C** | **45** |
| 200 | Sprint 1 | As a user I need source control | **C** | **5** |
| 300 | Onset | As a user I want to be able to Rewind/Fast Forward/Pause/Play data so that the movement of nodes can be easily visualized over time | **C** | **8** |
| 400 | Onset | As a user I want to be able to filter out different types of nodes so that the display is not cluttered | **C** | **5** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 600 | Sprint 1 | As a user I want to be able to view data in real time | **NC** | **45** |
| 700 | Sprint 2 | As a user I want 3D models to face the correct direction when moving so that it is intuitive to understand where they are heading | **T** | **3** |
| 800 | Sprint 3 | As a user I want to view “enemy” objects with a halo around them in order to more easily identify them | **C** | **2** |
| 900 | Sprint 3 | As the application owner I want the system to be designed with globalization in mind so that it is easy to modify for other environments | **T** | **34** |

4.5 User Story Acceptance Criteria

<< This section lists acceptance criteria for each of the user stories. Acceptance criteria in this section should define the boundaries of a user story and are used to confirm when a story is completed and working as intended.

The **ID** column specifies the ID for the acceptance criteria with respect to the corresponding user story ID. The **Description** column specifies the acceptance criteria formally. The **Verification** column provides information about how the acceptance criteria will be tested.

Remember that not all user stories are necessarily functional. User stories can also specify the need for security (e.g. who can and cannot use a system), globalization (e.g. descriptions and other strings will not be hard coded so that the software can be translated into other languages more easily), portability (e.g. developing an iPhone application to be compatible with other smart phone platforms like Android or Windows Phone) accessibility (e.g. the program complying with ADA guidelines), availability (e.g. the system will be able to reboot in under 2 minutes in the event of failure.), etc. >>

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Verification** |
| 110 | Nodes shall be represented in three-dimensional space. |  |
| 120 | Users shall be able to manipulate the map to view nodes from different perspectives. | Usability test to determine if map can be manipulated in all three dimensions. |
| 130 | Node positional information shall conform to guidelines established in IEEE 702.34 guidelines. | Create test cases to verify that node positional data is stored in accordance with given standards. |
|  |  |  |
|  |  |  |
| . |  |  |
| . |  |  |
| . |  |  |
| 910 | Strings for field descriptions in the user interface shall not be hard coded. |  |
| 920 | Alternative language packs shall be selectable by the user. | Create stub test data to simulate alternative languages. |

4.6 Constraints and Limitations

<< This section provides a list of constraints and limitations for the project. This provides additional information about any limitations that may exist in the project (e.g. will not work in versions of Internet Explorer prior to 8.0) that are not covered by requirements, but provide important information related to the project. This may include constraints related to security, performance, and other aspects of the system.

The **Constraint** field lists the constraint or limitation for the project. The **ID** field lists the related requirement ID (if any) that provides additional context for the constraint. >>

|  |  |
| --- | --- |
| **Constraint** | **ID** |
| Project will be all open source, no paid services can be used. | 720 |
| Testing will not be done directly upon Jenkins servers as acquired access is difficult externally. | 1100 |
| The group has little python experience so learning will be required. |  |
|  |  |
|  |  |

5. Design

5.1 Introduction

<< Briefly describe the major aspects of the design and, if applicable, how a developer will use it. For example: “Create and post a General Ledger transaction using the glTrx routines. Perform account inquiries with gjJournal routines.” >>

5.2 Scope

<< Provide a brief overview of the scope of this design. Also touch on anything that will not be included in this document. >>

5.3 High-Level Component Design

<< Create a diagram of the high-level components or modules in the program, linking them with arrows to show any dependencies. Also complete the tables to provide a description of each module as well as the table which traces components to their related requirements. >>

|  |  |
| --- | --- |
| **Component** | **Description** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Component** | **Related Requirements** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

5.4 Activity Diagrams

<< Include activity diagrams for important workflows in the program. At least one diagram should be included for the main workflow in the program. Optionally include labels that indicate which component is responsible for that part of the activity. Activity diagrams for components which perform complex tasks should also be included. >>

5.5 Class Diagram

<< Include a class diagram for all classes to be designed. Optionally include major data elements of those classes and important methods and functions that will be used by other classes. >>

5.6 Sequence Diagram

<< Include sequence diagrams for important functionality of the program to indicate control flow. These diagrams should include classes found in the class diagram and use the methods for those classes to show the interaction between them. >>

5.6 Data Architecture

<< Include any information or diagrams that provide details about databases, xml configuration files, or other data structures that are a part of the system. If a very specific format is required, it may be worthwhile to provide a more robust description or a detailed design such as a database schema. >>

5.7 Alternative Designs and Design Rationale

<< Provide information regarding other designs which were considered, but not chosen. Provide rationale for why the chosen design was selected over the alternative candidate designs. >>

5.8 Low-Level Component Design

<< Optionally provide a detailed design for each class or module in the program. Include information about the variables, methods, and other important aspects of the class such as exception handling.

Note that the templates provided here are designed for Java and that some languages will not contain >>

Class 1

Description

|  |  |  |
| --- | --- | --- |
|  | Access | Description |
|  |  |  |
| **Extends** |  | |
| **Implements** |  | |
| **Throws** |  | |

Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Access | Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access | Description |
|  |  |  |  |
| **Parameter** | **Type** | **Description** | |
|  |  |  | |
|  |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access | Description |
|  |  |  |  |
| **Parameter** | **Type** | **Description** | |
|  |  |  | |
|  |  |  | |

Class 2

Description

|  |  |  |
| --- | --- | --- |
|  | Access | Description |
|  |  |  |
| **Extends** |  | |
| **Implements** |  | |
| **Throws** |  | |

Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Access | Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access | Description |
|  |  |  |  |
| **Parameter** | **Type** | **Description** | |
|  |  |  | |
|  |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access | Description |
|  |  |  |  |
| **Parameter** | **Type** | **Description** | |
|  |  |  | |
|  |  |  | |

6. User Interface

6.1 UI Description

<< Provide a brief description of the UI that will be used in this program and how users will interact with the program. >>

6.2 UI Mockup

<< Create a mockup of the user interface. This can be a simple drawing that demonstrates key parts of the user interface or a screenshot of a prototype created within an IDE. >>

7. Test Plan

7.1 Test Plan Description

<< Provide a brief description of how testing will be conducted for this project. >>

7.2 Testing Tools

<< Provide a brief description of any testing tools, suites, etc. that will be used to assist with testing the project’s code. >>

|  |  |
| --- | --- |
| **Tool** | **Description / Function** |
| Jenkins | Continuous integration and regression testing. |
|  |  |
|  |  |

7.3 Test Data

<< Identify any data that will be necessary as part of the test process. >>

|  |  |  |
| --- | --- | --- |
| **Data** | **Description** | **Link** |
| Sample Data | Sample data of known good and bad examples construct unit tests. | [Link](http://examplelink.com) |
| User Database | Access to company’s Active Directory to test authentication for users. | [Link](http://examplelink.com) |
| Order Database | Stub data for unit tests. | [Link](http://examplelink.com) |

7.4 Test Types and Frequency

<< Provide a list of all the different types of tests that will be performed as part of this project, a brief description of each, and the frequency with which each test will be conducted. >>

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Frequency** |
| Integration Test | Integration test of existing code | Weekly starting with Sprint 3 |
| Regression Test | Run unit tests against all code | Daily (2:00 AM) |
| Usability Test | Perform manual usability test of UI. | Every Sprint |

7.5 Test Coverage

<< Identify components or other aspects of the project that will not be tested and provide rationale for that reasoning. >>

|  |  |
| --- | --- |
| **Component** | **Rationale** |
| Req. 720 | Do not have sufficient time to generate stub data. |
|  |  |
|  |  |

7.6 Test Location

<< Indicate where unit tests, test cases, etc. are being stored >>

8. Project Closure

8.1 Goals / Vision

**<<** Provide an update to the vision statement that was originally stated in the Project Initiation document. >>

8.2 Delivered Solution

<< Provide a high-level description of what was planned and what is being delivered. >>

8.3 Remaining Work

<< Provide a short summary of what should be done next, ways of further improving the project, or any additional recommendations. >>

9. Deliverables

<< This section contains information on the location of any deliverables for the project. If there are none for a particular category, indicate that it is not applicable. Otherwise provide the name of any files, etc. and where they can be found. >>

9.1 Study Results

<< The location of the results of any studies performed as part of this project. >>

9.2 Requirements and Design Documents

<< The location of any requirements and design documents. >>

9.3 Code

<< The location of any code written for this project. >>

9.4 Tests and Test Results

<< The location of any tests (unit, regression, etc.), test results, or other testing documents. >>

9.5 Build Process Documents

<< The location of any documents detailing build processes. >>

9.6 Install Process Documents

<< The location of any documents describing installation processes. >>

9.7 Administrator’s and/or User’s Manual

<< The location of any manuals, or help documentation. >>

9.8 Postmortem Document

<< The location of the Postmortem document. >>

9.9 Final Report

<< The location of the Final Report document. >>

10. Definitions and Acronyms

<< This section provides a definition for terms or acronyms used in this document which may not be familiar for all users. >>

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |