|  |
| --- |
|  |
| Java GlusterFS |
| Feasibility Study and Project Plan |
|  |
| **CIS 4911 Senior Capstone Project**  **Section U01**  **Team Members: Maylem Gonzalez** |
| **Ian Herbig**  **Mentor: Louis Zuckerman**  **Instructor: Masoud Sadjadi**  **12/9/2014** |
|  |

|  |
| --- |
|  |

*Abstract*

*This document details a high-level overview of the current state of Java GlusterFS and explores the problem presented by the project and the feasibility of various solutions to that problem. Additionally, this document details the project organization and roles of its associated members.*

Table of Contents

[1: Introduction 3](#_Toc405934933)

[1.1: Problem Definition 3](#_Toc405934934)

[1.2: Background 3](#_Toc405934935)

[1.3: Definitions, Acronyms, and Abbreviations 4](#_Toc405934936)

[1.4: Overview of Document 5](#_Toc405934937)

[2: Feasibility Study 6](#_Toc405934938)

[2.1: Description of Current System 6](#_Toc405934939)

[2.2: Purpose of New System 7](#_Toc405934940)

[2.3: High-level Definition of User Requirements 8](#_Toc405934941)

[2.4: Alternative Solutions 8](#_Toc405934942)

[2.4.1: Description of Alternatives 8](#_Toc405934943)

[2.4.2: Selection Criteria 9](#_Toc405934944)

[2.4.3: Analysis of Alternatives 9](#_Toc405934945)

[2.5: Recommendations 12](#_Toc405934946)

[3: Project Plan 13](#_Toc405934947)

[3.1: Project Organization 13](#_Toc405934948)

[3.1.1: Project Personnel Organization 13](#_Toc405934949)

[3.1.2: Hardware and Software Resources 14](#_Toc405934950)

[3.2: Identification of Tasks, Milestones, and Deliverables 15](#_Toc405934951)

[4: Appendix 16](#_Toc405934952)

[4.1: Appendix A – Project Schedule 17](#_Toc405934953)

[4.3: Appendix C – Cost Matrix 19](#_Toc405934954)

[4.4: Appendix D – Diary of Meetings 20](#_Toc405934955)

[5: References 26](#_Toc405934956)

# 1: Introduction

The Java GlusterFS is a Java implementation of the virtual file system, GlusterFS. The purpose of the project is then to bind Java to the GlusterFS. In this chapter, the GlusterFS will be explained, and a brief overview of the document will be provided.

In section 1.1, we describe the problem that the project attempts to solve. Section 1.2 then provides a background of the project. Section 1.3 provides useful definitions, acronyms and abbreviations that will be used throughout the document. Finally, section 1.4 provides a brief overview of the document as a whole.

## 1.1: Problem Definition

The Java GlusterFS project is, in its current state, an incomplete implementation of Java 7’s NIO.2 file system API backed by GlusterFS via libgfapi-jni. This project seeks to make headway towards a more complete implementation of Java’s FileSystemProvider API.

## 1.2: Background

In 2013, our mentor, Louis Zuckerman, set out to extend the virtualized file system, GlusterFS, into Java. Java7’s NIO.2 File System Provider API provided the tools necessary. Throughout the year, he implemented large portions of the file system ranging from connecting to the GlusterFS volume and performing basic synchronous file I/O to file attributes and permissions.

Zuckerman’s work was heavily backed by testing, boasting an impressive 85% branch coverage; a milestone we hope to match, if not better. The implementation, however, is as yet incomplete, and our project seeks to remedy that to as great an extent as possible in 12 weeks.

## 1.3: Definitions, Acronyms, and Abbreviations

* **API**: Application programming interface
* **File system**: A system used to control how data is stored and retrieved. Focuses on logical units of storage rather than physically contiguous units when applied to a computer’s method of data storage.
* **FOSS**: Free, open-source software
* **GlusterFS**: A scale-out, networked-attached storage platform
* **Horizontal scaling**: A form of scaling a system which involves adding more nodes to a system. Contrast with vertical scaling, where the size of each node in a system is larger.
* **I/O**: Input-output
* **Java Native Interface (JNI)**: A programming framework which allows Java code to call and be called by native applications and libraries written in other languages
* **libgfapi**: A library for accessing data in GlusterFS
* **libgfapi-jni**: Java Native Interface bindings for the libgfapi
* **Native applications**: Programs specific to hardware and operating system platforms
* **Network-attached storage**: file-level computer data storage connected to a computer network providing data access to a heterogeneous group of clients.
* **NIO.2**: Java 7’s enhanced file package which provides file system APIs
* **NIO**: Non-blocking I/O; also called asynchronous I/O
* **Scale-out**: see “horizontal scaling”
* **Synchronous I/O**: Input and output which blocks the progress of a program until its completion; also called blocking I/O

## 1.4: Overview of Document

This document is broken up into 5 main chapters, and each chapter is further broken up into sections and subsections. This section marks the end of the first chapter. Chapter 2 consists of a feasibility evaluation of the Java GlusterFS project. Section 2.1 describes the state of the current Java GlusterFS system, along with its known limitations. Section 2.2 details the new functionalities to be added in the proposed extension to the system. Section 2.3 focuses on the functional and non-functional requirements of the project along with a brief analysis of any privacy and security concerns inherent to the implementation of the project.

Section 2.4 concerns itself with alternative solutions to the extension of the Java GlusterFS project. Subsection 2.4.1 details these alternatives, and subsection 2.4.2 describes the criteria used in selecting one alternative over another. Subsection 2.4.3 contains an analysis of the alternatives. Section 2.5 concludes chapter 2 by recommending which option to take, after the proposed extension to the system and its alternatives have been analyzed.

Chapter 3 describes the plan for the Java GlusterFS project. Section 3.1 introduces the project organization. Subsection 3.1.1 details the organization of the project members. Subsection 3.1.2 lists the hardware and software resources needed in completing this project. Section 3.2 concludes chapter 3 with a list of the tasks, milestones and deliverables to be completed in the duration of this project.

Chapters 4 and 5 are the metachapters in that they concern themselves with the content in the previous chapters. Chapter 4 contains the appendices and Chapter 5 contains any references we made throughout the document.

# 2: Feasibility Study

A feasibility study consists of an analysis and an evaluation of the potential of a proposed project. Within this chapter, we analyze the current system allowing us to elaborate on the reasoning for expanding its functionality. Additionally we provide high-level definitions for the user requirements which allow us to discuss alternative solutions to the problem.

## 2.1: Description of Current System

The state of the Java GlusterFS binding can be described in three forms: its functionality, the limitations on that functionality, and the system’s constraints. They are as follows:

**Functionality**:

* Able to connect to a GlusterFS volume using the NIO.2 API
* Able to perform basic synchronous file I/O
  + Reading the contents of a file all at once
  + Writing a chunk of bytes to a file all at once
* Able to handle file attributes
  + See owner/group id (as a number), size, permissions, and last modified timestamp on files and directories
  + Set file permissions
* Able to view file system/volume statistics
  + See the total, free, and usable bytes in a volume
* Directory listing (with filtering) functionality
* Able to move/rename files
* Able to watch files for changes
* Able to publish test coverage and code quality reports to SonarQube

**Limitations**:

* Unable to perform asynchronous I/O
  + Cannot seek and read/write portions of a file
* Basic error reporting and handling
* Watch system is slow and blocking, utilizing polling
* Attribute support limited
  + Owner/group identification limited to numbers, no name support
  + Cannot change owner/group

**Constraints**:

* System is written in Java
* System requires Internet access due to GlusterFS

## 2.2: Purpose of New System

The project this document describes seeks to rectify some of the limitations as well as introduce new functionality that is intended, but has not yet been implemented. These improvements are as follows:

* Able to delete files on the GlusterFS volume
* Able to copy files to/from/within the GlusterFS volume
* Improve upon existing attribute support
  + Enable modification of owner/group
  + Provide support for owner/group names
* Improve synchronous file I/O

## 2.3: High-level Definition of User Requirements

* The user must be able to connect to a GlusterFS volume.
* The user must be able to perform standard file operations on the volume up to and including the creation, modification, copying, and deleting of files in the volume.
* The user must be able to navigate the volume as they would a standard directory from within a Java application, including listing and filtering those lists.
* The user must be able to perform standard file attribute operations on files in the volume up to and including viewing and modifying owner/group id, name, size, permissions, and last modified timestamps.
* The user must be able to see statistics related to the volume as a whole including the total, free, and usable size of the volume.
* The user must be able to watch a directory or file for changes.

## 2.4: Alternative Solutions

In consideration of the user requirements described in section 2.3, we must propose a series of alternatives and reduce our technical debt by analyzing them systematically for cost and feasibility concerns. The description of alternatives follows.

### 2.4.1: Description of Alternatives

**Alternative 1**: Start over and prioritize missing functionality

We could scrap the work that has been done and begin anew. In this way, we can prioritize the missing functionality from the start. This presupposes that the currently missing functionality is in some way more valuable or important than the currently present functionality, and would only make sense if the current functionality in some way impedes progress towards the missing functionality.

**Alternative 2**: Extend existing functionality

We could keep the current system as is and simply build the new set of functionality into it. This would limit our need to cover old ground, but may run into issues if existing functionality will in some way impede progress.

**Alternative 3**: Start over and change language

We could scrap the work that has been done and begin anew in a different language. If another language has more advanced tools to enable the integration of GlusterFS into the language, then it may streamline the process. However, it risks unknown complications that may arise. Moreover, it defeats the purpose of the entire project.

Next we must analyze each alternative for its operational, technical, schedule, and economic feasibility to enable us to recommend an appropriate solution to the problem.

### 2.4.2: Selection Criteria

* **Operational feasibility**: Operational feasibility is a measure of how well a proposed system solves the problem presented by the client.
* **Resource feasibility**: Resource feasibility is a measure of the resources which can be allocated to the system and how much of those resources the project would require.
* **Financial feasibility**: Financial feasibility is a comparison of the estimated cost of the project compared to the budget allocated to that project.

### 2.4.3: Analysis of Alternatives

This subsection consists of an analysis of each of the alternatives using the selection criteria discussed in the previous subsection. These alternatives will receive a score for each of the categories in the selection criteria. We have reflected the final score for each of the alternatives below out a maximum score of 30. For a more detailed breakdown of the scores, please refer to Appendix B – Feasibility Matrix.

**Alternative 1**: Start over and prioritize missing functionality **[19]**

* **Operational feasibility**: Under this alternative, the problem would still hopefully be solved, albeit in a longer time frame. However, the problem will have been shifted from implementing the missing functionality of the current system, to a different, larger set of missing functionalities (the new prioritized functionalities along with others from the current system that would still need to be implemented).
* **Resource feasibility**: A considerable amount of time would need to be spent to restart the project. The current system was built over the course of over a year, so we expect that a similar amount of time would be spent building a new system to the current state. Moreover, the current system was built with the aid of other open source community members, which suggests that we would most likely need more than just the current number of personnel working on the project. Finally, a significant amount of time would be spent testing, installing, and researching software that would be compatible with the project, as our mentor has done.
* **Financial feasibility**: This alternative would be financially feasible, since the personnel used would not be paid, and since the software installed for this open source project would be free. As such, the cost of this alternative is estimated to be $0.

**Alternative 2**: Extend existing functionality **[30]**

* **Operational feasibility**: This alternative solves the problem presented by our client, precisely as it was originally framed. A list of missing functionalities would be made and addressed, while the existing functionalities would remain. As such, the user requirements would be more completely met.
* **Resource feasibility**: Much less time would be expended on meeting the user requirements under this alternative, since several of these requirements have already been implemented under the current system. This would allow us to more quickly meet the full list of user requirements. Since we already know what development environment should be used, we would also not need to expend resources in finding and testing alternative software.
* **Financial feasibility**: This alternative would also be financially feasible, since the software currently being used in the project is free, and those working on the current system are not being paid. The cost of this alternative is estimated to be $0.

**Alternative 3**: Start over and change language **[13]**

* **Operational feasibility**: This alternative would also solve the problem, but would take much longer to do so since it would require us to start from scratch, as well as additionally research alternative languages for the solution. The solved problem would have shifted to a slightly different problem that would now include also finding a suitable language other than Java for the task.
* **Resource feasibility**: The time requirement for this project would be greater than if we were to continue implementing the current system. We would have a significant research overhead due to the extra time that would need to be spent finding an alternative language and compatible software. Moreover, all of the currently existing functionalities would have to be implemented in this new language. This would also require us to most likely seek the aid of other open source community members, but, given the change in language, this may require that we seek a new set of contacts in the community.
* **Financial feasibility**: This alternative would also be financially feasible, since the project would still be an open source project, and the software sought and used would still be free. The cost of this alternative is estimated to be $0.

## 2.5: Recommendations

Based on the alternatives analyzed in the previous section, we recommend that Alternative 2 (extending the existing functionality of the current system) be selected, as this alternative produced the best total feasibility score in our feasibility matrix (see Appendix B) and also matched the other alternatives in monetary cost (see Appendix C).

The following is a list of the major functionalities and concerns that have yet to be handled in the Java GlusterFS project:

* **Delete files**
* **Copy files**
* Move files
* Create directories
* Update watch service to use libgfchangelog (instead of polling)
* Finish attribute support
  + **Owner/group names & ability to change**
  + More ways to set permissions
* **Advanced synchronous file I/O**
  + Seeking & reading/writing a portion of a file
* Asychronous file I/O
* Better error reporting & handling
* Miscellaneous concerns
  + Align project versions with glusterfs (this project & libgfapi-jni)
  + Publish test coverage report to Coveralls.io

Since this list consists of some features that would each take several months to implement, we further recommend that the functionalities bolded above be prioritized in the following four months. These recommended functionalities would require the least amount of external aid from community members, and also represent some of the most important features that any user would expect to be implemented when using Java GlusterFS.

# 3: Project Plan

This chapter focuses on the details of project management. We specifically focus on the project organization and the identification of tasks, milestones, and deliverables. Within, we present an initial timetable for the project and allot time to each aspect of the project.

## 3.1: Project Organization

The organization of the Java GlusterFS project for the remainder of this semester is a loose organization with a fair amount of redundancy in roles between the two project members. These roles will be elaborated on in some minor detail. In addition, this section will detail the various hardware and software resources that will be used throughout the project’s lifespan.

### 3.1.1: Project Personnel Organization

The two individuals working as a team within this project share a measure of redundancy in their roles due to the relatively small size of the team and the nature of the project itself. However, there remains a measure of specialization, so the roles will be described as follows:

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Description** |
| **Ian Herbig** | Project Manager/Software Engineer/Tester/Document Editor | Serves as the project manager for the other team member. Implements functional requirements. Performs unit and integration tests. Manages the documentation. |
| **Maylem Gonzalez** | Project Manager/Software Engineer/Tester/Minute-Taker | Serves as the project manager for the other team member. Implements functional requirements. Performs unit and integration tests. Takes minutes during meetings. |

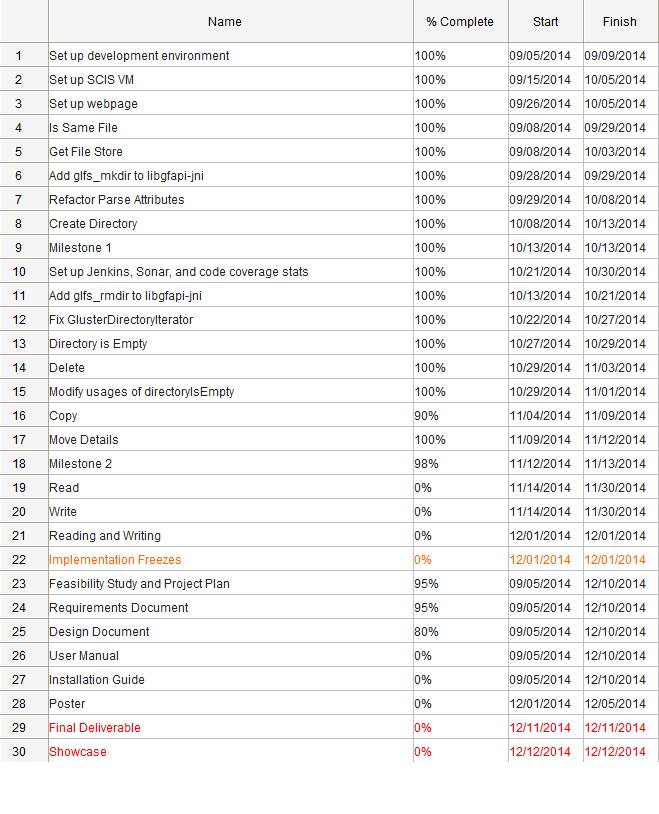
### 3.1.2: Hardware and Software Resources

This section details the various hardware and software needs of the project.

|  |  |
| --- | --- |
| ***Hardware*** | ***Software*** |
| Minimum 1GB RAM (recommended 2GB) | OpenJDK 7 |
| Internet access | Oracle JDK 7 |
| Minimum 20GB storage | Apache Maven v3.0 |
| Minimum 1GHz processor (recommended 1.6GHz) | IntelliJ IDEA |
| 64-bit processor | Lombok plugin for IntelliJ IDEA |
| Portability recommended | Vagrant |
| Minimum 1024x768 display | Software contained in the glusterfs-common package |
|  | Git |
|  | JUnit |
|  | A 64-bit Linux operating system |
|  | Gantter.com |
|  | Trello |
|  | SonarQube |
|  | Jenkins |
|  | Atlassian Clover |

## 3.2: Identification of Tasks, Milestones, and Deliverables

The following table is an incomplete schedule subject to revision.

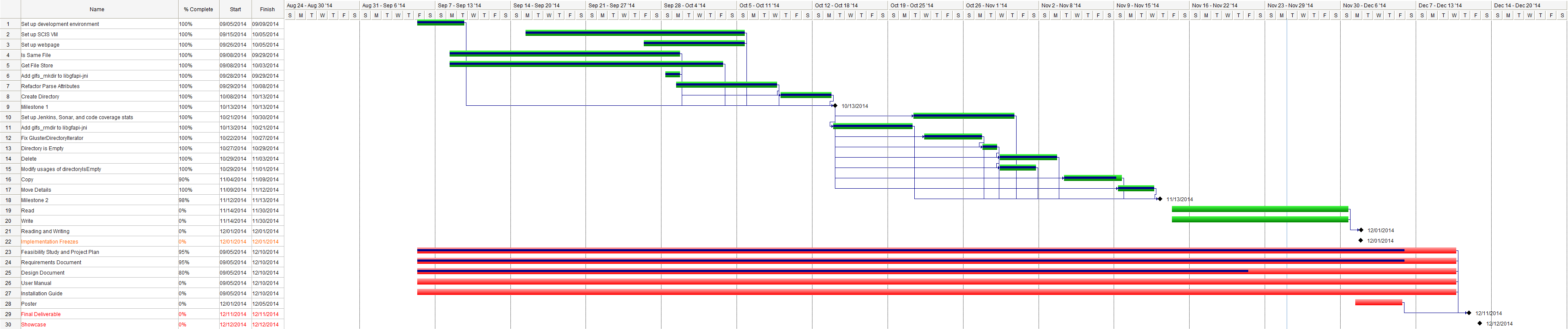


# 4: Appendix

This chapter consists of the following appendices:

* Appendix A – Project Schedule
* Appendix B – Feasibility Matrix
* Appendix C – Cost Matrix
* Appendix D –Diary of Meetings

## 4.1: Appendix A – Project Schedule



Legend:

Orange: Important date

Green: Task

Diamond: Milestone

Red: Deliverable4.2: Appendix B – Feasibility Matrix

This appendix consists of a feasibility matrix with the alternatives discussed in chapter 2. The scaled used below is from 0 to 10, where 0 is completely infeasible and 10 is very feasible.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Alternative 1: Start over and prioritize missing functionality** | **Alternative 2: Extend existing functionality** | **Alternative 3: Start over and change language** |
| **Operational feasibility** | 5 | 10 | 0 |
| **Resource feasibility** | 4 | 10 | 3 |
| **Financial feasibility** | 10 | 10 | 10 |
| **Total** | 19 | 30 | 13 |

## 4.3: Appendix C – Cost Matrix

This appendix consists of a cost matrix for the proposed project (Alternative 2).

|  |  |
| --- | --- |
| **Project Component** | **Monetary Cost** |
| **Hardware** | $0 |
| **Software** | $0 |
| **Personnel** | $0 |
| **Training** | $0 |
| **Testing** | $0 |
| **Total** | $0 |

## 4.4: Appendix D – Diary of Meetings

**Begin meeting**: 9/3/14 7:15PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Personal introductions
* Overview of GlusterFS
* Overview of development environment
* Current state of glusterfs-java-filesystem
* Demos detailing functionality and unit tests
* Discuss extent of project (resulting in timeline)

**End meeting**: 9:00PM

**Next meeting date**: 9/7/14 3PM

**Next meeting agenda**: Set up development environment, sort out development logistics (physical and management perspectives)

**Start meeting**: 9/7/14 1:55PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Forked GlusterFS-Java-Filesystem repository into FIU SCIS organization
* Overview of Maven commands
* Set up development environment
  + Made shell script to run IntelliJ with correct Java 7 JDK
  + Built libgfapi-jni with Maven
  + Built glusterfs-java-filesystem with Maven
  + Ran glusterfs-java-filesystem-example with Maven
  + Ran/Debugged glusterfs-java-filesystem tests with IDEA
  + Ran/Debugged glusterfs-java-filesystem-example with IDEA
* Broke up milestones into tasks
* Discussed class diagrams and use cases

**End meeting**: 5:10PM

**Next meeting date**: 9/14/14 3PM

**Next meeting agenda**: Resolve issues with boot testing VM with Vagrant, discuss progress and position in timeline, and identify further tasks.

**Start meeting**: 9/14/14 3:40PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed shortcuts to make the development process more efficient in IntelliJ IDEA
* Concluded that toFile() should not be implemented in the project
* Discussed how relativize and resolve work in the library
* Overview of Mockito, Lombok plugin, and PowerMock syntax for JUnit testing
* Discussed the acceptance process to be used in the project

**End meeting**: 6:10PM

**Next meeting date**: 9/21/14 4PM

**Next meeting agenda**: Identify further tasks, discuss any issues with JUnit testing, and discuss the acceptance of any implemented and tested features.

**Start meeting**: 9/17/14 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Addressed questions concerning the creation of JUnit tests
* Overview of GlusterFS and how it works
* Decided on the structure of the git repository
* Discussed whether the SCIS virtual machine would suit our needs
* Overview of Jenkins and continuous integration

**End meeting**: 8:30PM

**Next meeting date**: 9/21/14 4PM

**Next meeting agenda**: Discuss the acceptance of the isSameFile and getFileStore features along with any questions concerning JUnit testing.

**Start meeting**: 9/21/14 3:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the acceptance of the two completed features (issamefile and getfilestore)
* Overview of pull requests
* Discussed changes needed to be implemented in libgfapi-jni project for the upcoming createDirectory and directoryIsEmpty features
* Discussed integration testing in the libgfapi-jni project
* Overview of file attributes
* Discussed preferred programming practices for this project
* Overview of HawtJNI

**End meeting**: 5:30PM

**Next meeting date**: 9/28/14 4PM

**Next meeting agenda**: Discuss the next round of features set for acceptance along with any questions concerning testing in the libgfapi-jni project.

**Start meeting**: 9/28/14 7:00PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to parse file attributes and how to refactor our initial implementation to best accomplish this
* Solved the issue of glfs\_mkdir not being recognized in GlusterFileSystemProvider
* Discussed a procedure to follow for the next time that we encounter an issue with the snapshot and missing dependencies
* Discussed how to best demo our work and agreed to use Jenkins and Sonar
* Discussed how to fix issues encountered in getting our development environment to run in FIU's SCIS virtual machine

**End meeting**: 8:15PM

**Next meeting date**: 10/5/14 4PM

**Next meeting agenda**: Discuss the next round of features set for acceptance, and determine how to tie Jenkins and Sonar to our project.

**Start meeting**: 10/13/14 6:50PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to use and set up Jenkins with the Java GlusterFS project
* Discussed issue where createDirectory was resulting in an error on some machines
* Discussed how Java handles the default permissions for a new file when no file attributes are provided
* Discussed the acceptance of the createDirectory feature

**End meeting**: 8:50PM

**Next meeting date**: 10/20/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents.

**Start meeting**: 10/20/14 7:10PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed a potential issue with the current GlusterDirectoryIterator implementation where the special entries "." and ".." were being returned (contrary to the behavior of the default provider)
* Determined the best fix for the GlusterDirectoryIterator issue
* Discussed the current implementation of isDirectoryEmpty
* Discussed a potential issue with calls to verifyStatic() in the JUnit tests
* Discussed how to set up Jenkins on the FIU VM

**End meeting**: 8:30PM

**Next meeting date**: 10/26/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents. Additionally, possible modifications to the Jenkins and Sonar profiles will be addressed.

**Start meeting**: 10/29/14 7:10PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed how to set up the Atlassian Clover code coverage tool with Jenkins and Sonar
* Discussed possible tools and locations for filming our videos
* Discussed the current state of the documentation, including class diagrams and sequence diagrams
* Discussed the extent to which security and file permissions will be implemented and enforced

**End meeting**: 8:10PM

**Next meeting date**: 11/3/14 7PM

**Next meeting agenda**: Discuss the next round of features set for acceptance as well as the current status of the project documents and videos. Additionally, brainstorm real-world scenarios where our Java library can be used.

**Start meeting**: 11/12/14 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the pull request for dirIsEmpty branch and any necessary changes to the current implementation or tests. Decided to modify if statements in advanceHelper method to switch cases.
* Discussed the move feature (specifically, how its implementation differs from spec) and decided that it is only missing a throws clause to be complete.
* Discussed the copy feature and how file attributes would be copied over (using chmod, chown).
* Discussed utilizing the platform-specific umask value in the logic of create directory. Decided that it would be best to not do this.
* Changed the priority of (1) user and group name and (2) reading and writing features. Reading/writing has a higher priority now, and user and group name feature will be done only if time permits.
* Discussed what aspects of the reading and writing features will need to be prioritized.
* Discussed how the user and group names feature would be implemented using a Linux UserPrincipleLookupService.

**End meeting**: 8:25PM

**Next meeting date**: 11/17/14 7PM

**Next meeting agenda**: Discuss the implementations and tests of delete, copy and move, as well as discuss the current status of the project documents and videos.

**Start meeting**: 6:45PM

**Location**: Picture Marketing office in Doral

**In attendance**: Louis Zuckerman, Maylem Gonzalez, Ian Herbig

**Summary of events**:

* Discussed the architectural and design patterns found in the Java-GlusterFS project. Determined that the following six patterns apply: Service Provider Interface, Abstract Factory, Object Pool, Singleton, Iterator, Fluent Interface, and Prototype.
* Determined that time should currently not be spent on trying to get Cobertura to work with our project.
* Discussed the use of isSameFile() in the move feature and how it results in a difference from specs. Decided that a note should be added to detail this change in behavior for future consideration.
* Discussed the use of ByteBuffer in reading/writing bytes in the project.
* Discussed the implementation of write(ByteBuffer src) and determined that the copy feature needs to be put on hold until the write implementation is modified to spec and bugs with the current implementation are fixed.
* Decided that UtilJNI should be made public instead of package protected.

**End meeting**: 8:15PM

5: References

|  |  |
| --- | --- |
| [1] | L. Zuckerman, "GitHub Repository for Java GlusterFS," [Online]. Available: https://github.com/semiosis/glusterfs-java-filesystem. [Accessed 5 9 2014]. |
| [2] | J. Darcy, "GlusterFS Internals and Directions," in *Red Hat Summit*, Boston, 2013. |
| [3] | "File I/O (Featuring NIO.2)," Oracle, [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/fileio.html. [Accessed 5 9 2014]. |
| [4] | "Java SE 7 Java Native Interface-related APIs and Developer Guides," Oracle, [Online]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/jni/. [Accessed 5 9 2014]. |