**Software Design Document (SDD)**

**CSC 440 Team #1**

**Version 1.0**

**Date: 11/22/2015**

**Table of Contents**

**1. Introduction**

**1.1 Date of Issue/Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Revision** | **Editor** |
| 11/10/2015 | Created the document | 0 | Kyle Marcum |
| 11/14/2015 | Added more to the document | 1 | Kyle Marcum |
| 11/21/2015 | Added the diagrams and corrections | 2 | Kyle Marcum  John Pratt |

**1.2 Context** – This Software Design Document was formed to provide the documentation for the design of the Web Server File Generator Software**.** Within the Software Design Document are narrative and graphical documentation of the software design for the project including use system design concerns, architectural style, sequence diagrams, deployment diagrams, class diagrams, design patterns, and other supporting requirement information.

**1.3 Scope –** This document describes the implementation details of a server/client file transfer software. The software will consist of four major functions. First, a server that will be able to create and transfer requested files to a client. Second, the server will document the requests in a log file. Third, the server will store the most commonly requested files in cache for quicker access. Last, a client that will be able to initiate multiple simultaneous requests.

**1.4 Authorship –** Created by Team #1

**1.5 Change history -** This section is described in the Date of Issue/Revision History table above.

**1.6 Summary -** The Software Design Document is divided into 5 sections with various subsections. The sections of the Software Design Document are:

1. Introduction

1.1 Date of Issue/Revision history

1.2 Context

1.3 Scope

1.4 Authorship

1.5 Change History

1.6 Summary

2. Software Architecture

2.1 Overview

2.2 System Design Concerns

2.3 Architectural Style

2.4 Sequence Diagrams

2.5 Deployment Diagrams

3. Detailed Design

3.1 Overview

3.2 Client Side Diagram

3.2.1 Class Diagrams

3.2.2 Design Patterns

3.3 Server Side Diagram

3.3.1 Class Diagrams

3.3.2 Design Patterns

4. Glossary

5. References

**2. Software Architecture**

**2.1 Overview –** This section of the design document describes the architectural design of the Web Server File Generator Software. The components and their interactions, system design concerns, and architectural style. It contains sequence diagrams of the system and deployment diagrams of the software.

This project according to the Software Requirement Specification will be composed by three (3) separate parts.

A Client that is designed to act like a traditional browser but limited to create TCP connections to the Server, to submit HTTP requests then when receives a response from the Server displays it.

A file generator that is a separate module helping the Server to serve Web files when Client requests them. This file generator uses a text file containing the file types the server can provide to create randomly these files and filling them with random alphabetic characters. The files generator also respects the requirement stating the limited sizes of the web files created.

A Server designed to receive HTTP requests over a TCP connection, identify the resource requested, get them from the file generator, then generate and send back the HTTP response to the request over the TCP connection to the Client. According to the Requirements, this server can also handle concurrent HTTP requests, create a file cache to make future responses more efficient and finally log the different requests in a log file.

**2.3 System Design Concerns –** This section discusses design concerns for the software. The non-functional requirements, data management, and boundary conditions of the software.

**2.3.1 Non-functional requirements**

**2.3.1.1 Scalability –** For our project, scalability is the ability for the software to be able to handle many more clients and more file requests. One problem that we may encounter is the inability to handle massive amounts of requests without affecting performance. It will probably slow the system down tremendously. We would have to use multiple servers to handle massive requests. Another problem that we may have is the ability to serve large amount of clients. Again, to fix that issue we would need multiple servers, each using its own resources, handling large amounts of clients.

**2.3.1.2 Availability -** Application downtime occurs due to inadequate testing, change management, and lack of ongoing failure monitoring. Downtime also derives from operations errors caused by lack of rigorous procedures and backup/restoration errors. Hardware reliability has improved so much over the past few years that typically less than 10% of the downtime is due to hardware problems. A small percentage of downtime is caused by environmental and other miscellaneous problems. As a general idea, availability is a measure of how often the application is available for use. More specifically, availability is a percentage calculation based on how often the application is actually available to handle service requests when compared to the total, planned, available runtime.

One concern our software may have is internet traffic on the NKU wireless network. If heavy traffic is across the network, then it may slow down our communication between server and client. Another concern is that the Server may get overloaded with requests and stop taking new requests until it catches up.

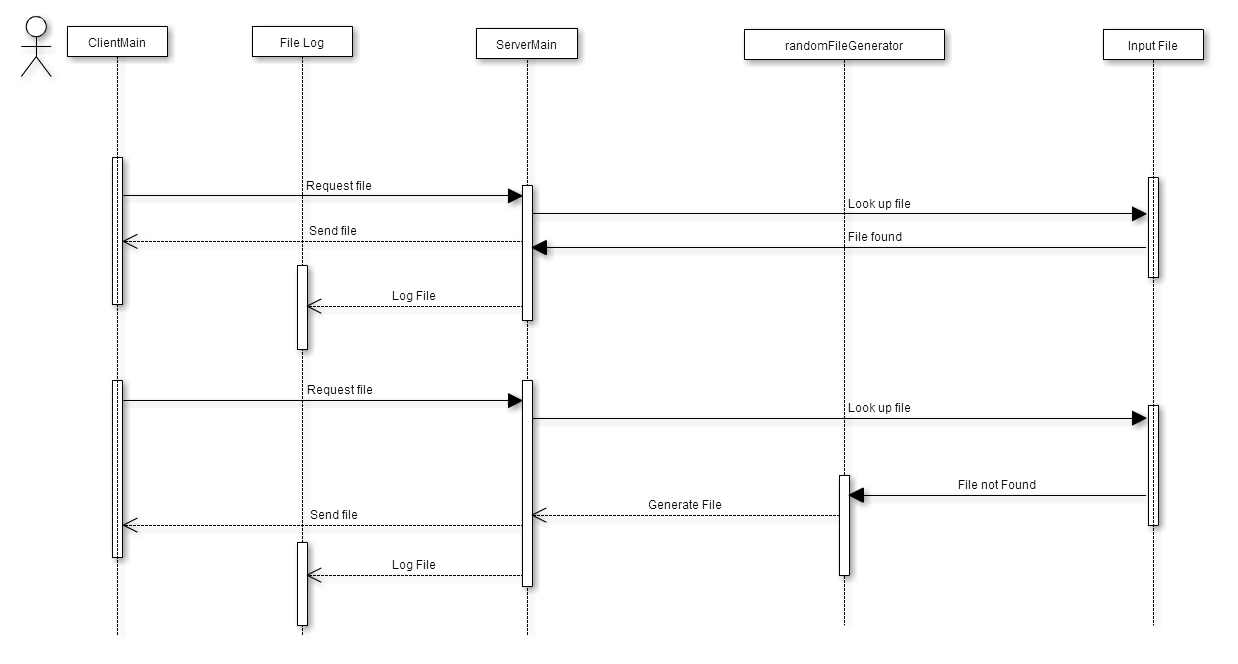
**2.3.1.3 Performance -** The performance concern is primarily one of implementation. The computers we are using are far in excess of what is required for this project. However, sloppy coding could easily result in inefficiencies and process hangs.

**2.3.2 Data Management -** Data Management shouldn’t be a really big problem because in this project we are not managing data and won’t need a database in the early versions of the solution. The only data kept are the logs.

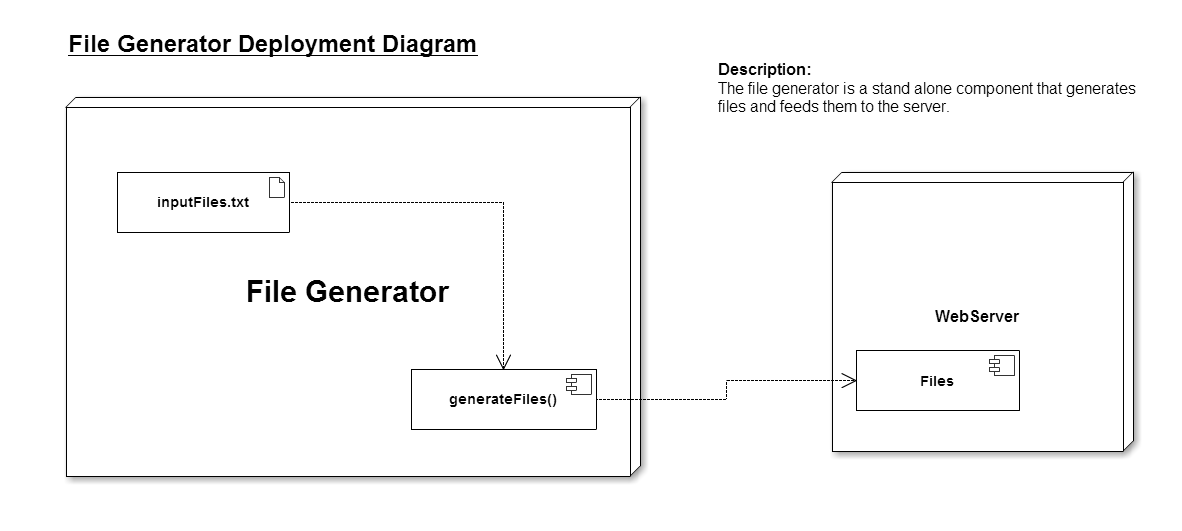
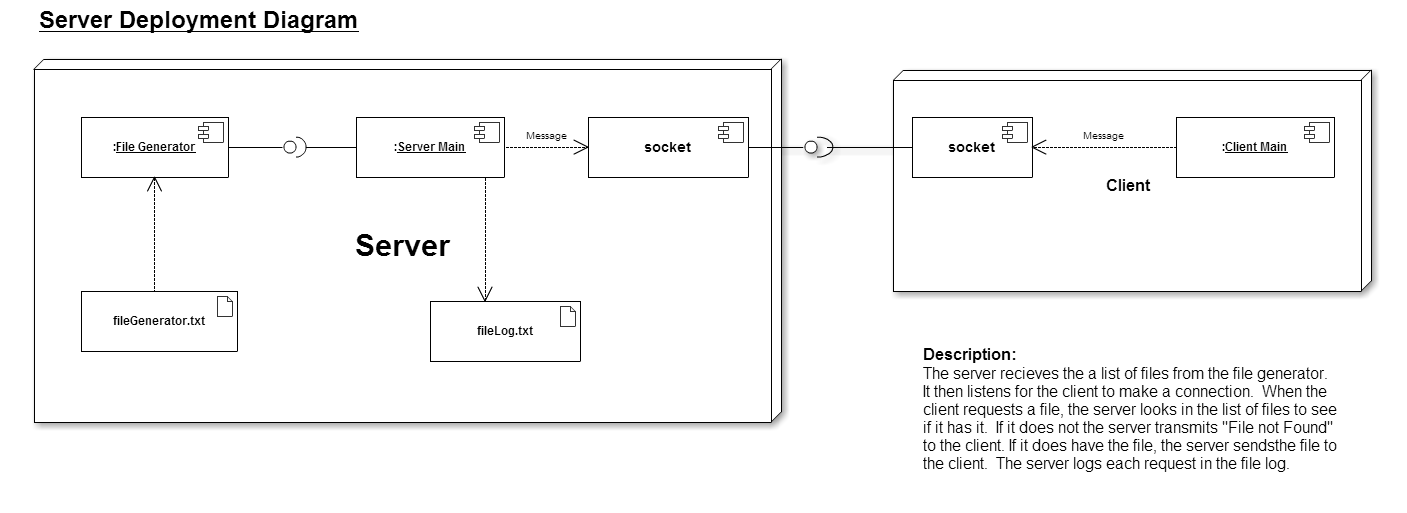
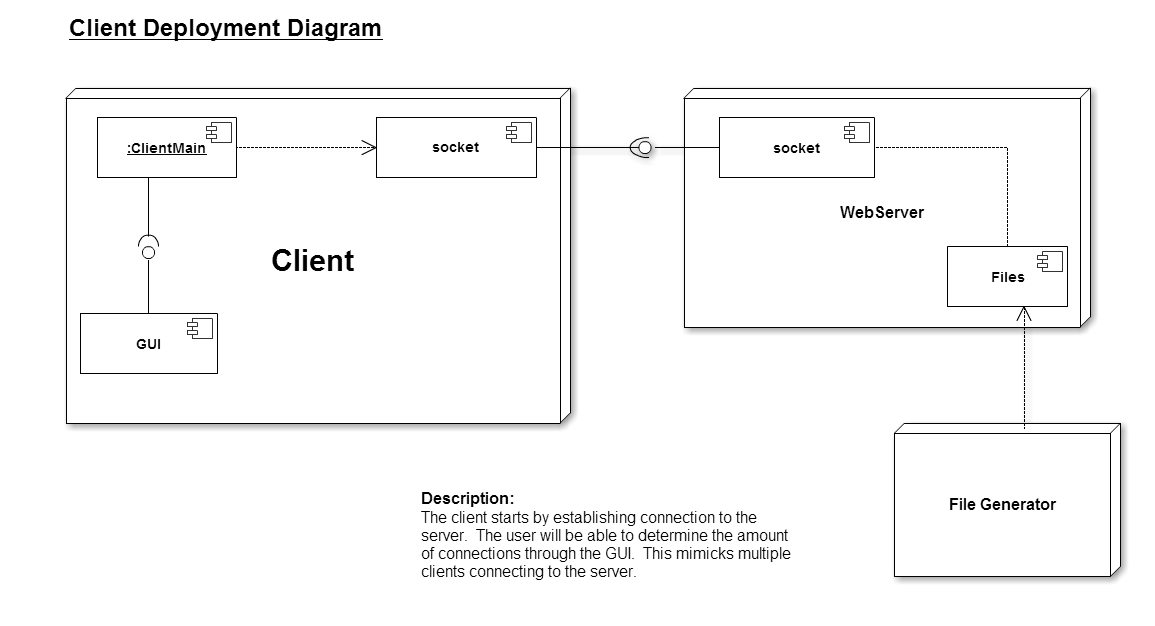
**2.3.3.1 Boundary Conditions (Initialization/Termination/Failure) –** This project includes some controls to prevent basic security issues like denial of service by limiting the number of concurrent requests and handles the Termination/Failure case gracefully.

**2.4 Architectural Style -** Specifies the system-wide structural properties of an application, and have an impact on the architecture of its subsystems. The Web Server File Generator Software is a combination of Data Flow and Interactive styles. The server uses a facade pattern, as a large part of its function is handled by separate classes, and the singleton pattern for thread and logic safety.

**2.5 Sequence Diagram**



**2.6 Deployment Diagrams**

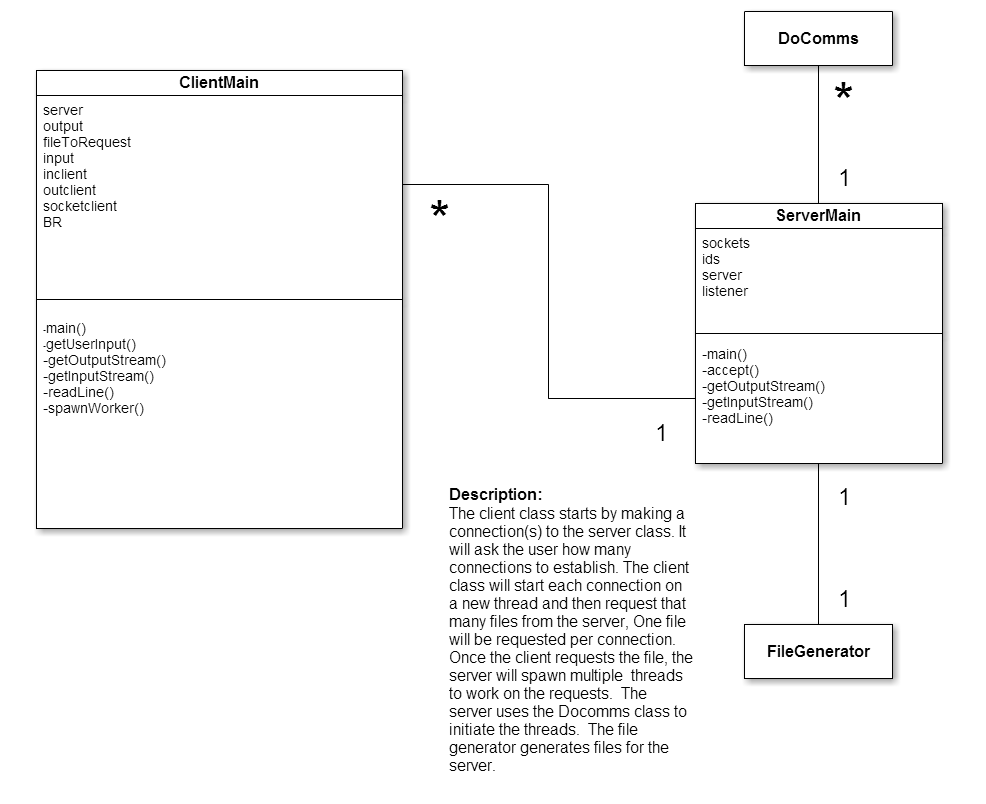


**3. Detailed Design**

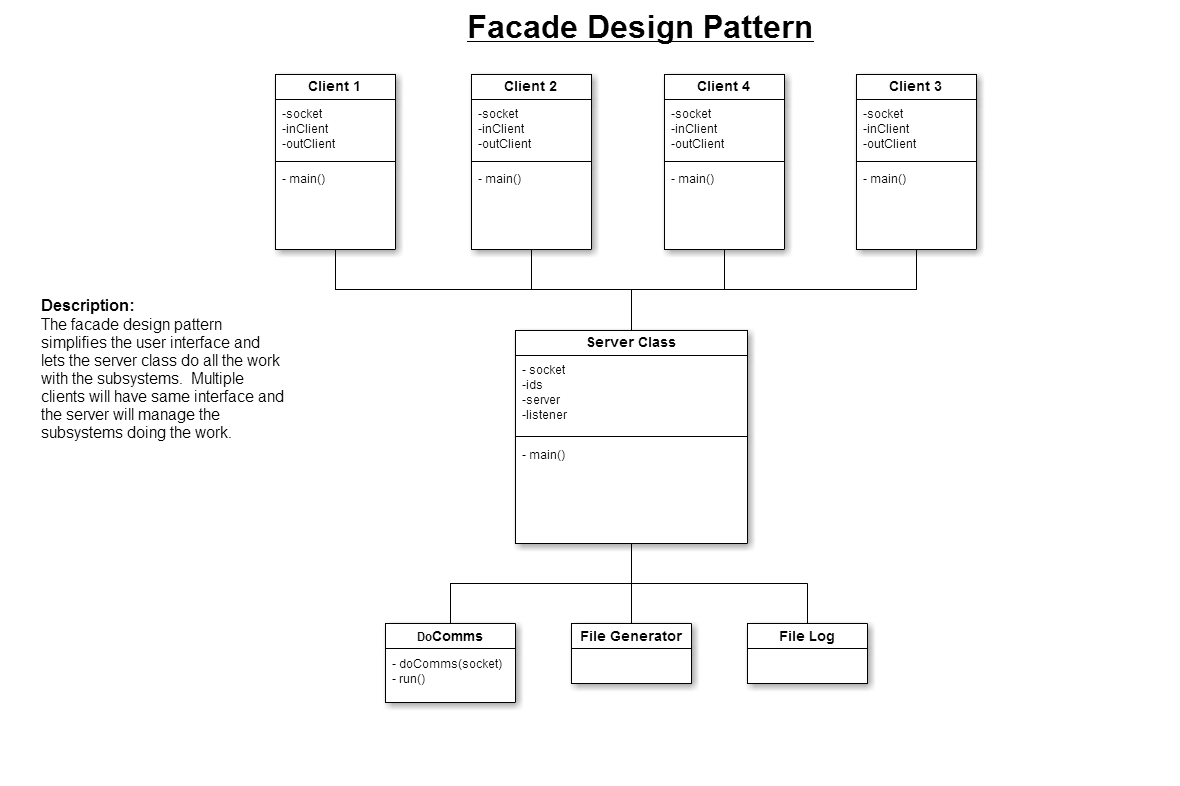
**3.1 Overview -** The client initiates multiple requests to the server. The server generates a set of number of files via the FileGenerator class. When the client initiates the request(s), the server starts an instance of doComms for each request from the client, to handle the request. In this section, diagrams are provided to show the class interaction and design patterns used. There are both class and pattern design diagrams for the server and client side.

**3.2 Client Side Design**

**3.2.1 Class Diagram**

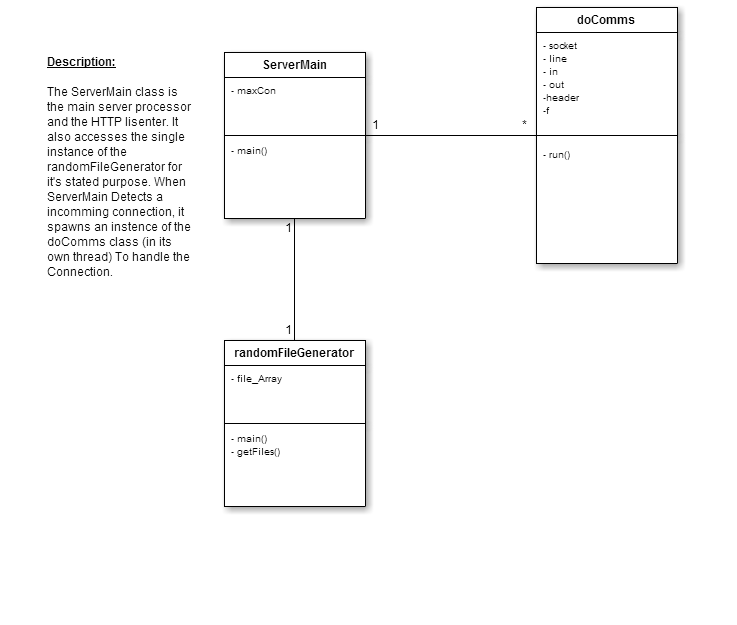


**3.2.2 Design Pattern**

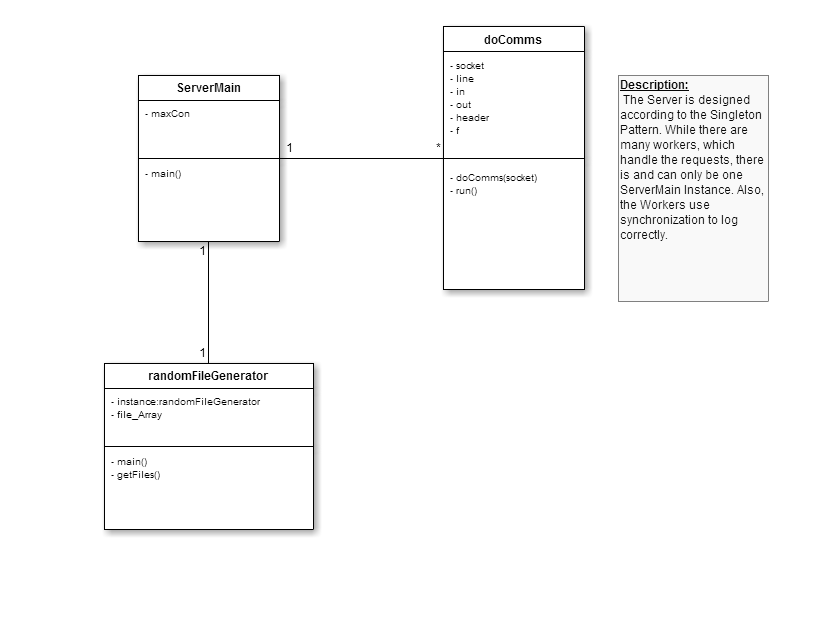


**3.3 Server Side Design**

**3.3.1 Class Diagram**

****

**3.3.2 Design Pattern**



**4. Glossary**

|  |  |
| --- | --- |
| **Term** | **Description** |
| Architectural Design | The overall structure of the software system |
| Sequence Diagram | An interaction diagram that shows how processes interact with one another and in what order |
| SRS | Software Requirements Specification |
| Data Flow Architecture style | Systems oriented around the transport and transformation of a stream of data |
| Interactive Architectural style | Systems that serve users or user-centric systems |

**5. References**

* **Example of SDS document in software engineering**

[**http://www.slideshare.net/RaviYasas/example-for-sds-document-in-software-engineering**](http://www.slideshare.net/RaviYasas/example-for-sds-document-in-software-engineering)

**2015**

* **Software Design Document**

**By: Thera Wii**

**April 7, 2009**

[**https://www.cs.drexel.edu/~dpn52/Therawii/design.pdf**](https://www.cs.drexel.edu/~dpn52/Therawii/design.pdf)

* **CSC 440 class slides**

**Professor: Wei Hao**

**Winter term 2015**