**CSCI 415 Assignment 2: Server Proxy Report, due 2/22/16**

**Garret Leatherman, Matthew Leet, Erik Canton, Jamie Thul**

**Overview**

To begin with we based our Multi-threaded Java Proxy Server on a very helpful tutorial by Jmelton: <http://www.jtmelton.com/2007/11/27/a-simple-multi-threaded-java-http-proxy-server/> . The caching functionality was based on a simple cache class by Christian Schenk: <http://www.christianschenk.org/blog/implementing-a-simple-cache-with-java/>. In addition we added request logging.

Our server proxy is a project, implemented in Java, consisting of four major classes:

ProxyServer: Handles the main function and is responsible for maintaining the server socket connection. It does this by multithreading connecting clients so the main server socket remains free to receive additional requests.

ProxyThread: Once the clientSocket is instantiated, the created thread attempts to resolve the client’s request. It also calls the Logit and SimpleCache classes to store information.

LogIt: A logging class for recording all server interactions to a text file.

SimpleCache: A caching class that holds a webpage object that is instantiated in the ProxyServer class. This allows the Server to return a cached version of a requested webpage instead of spending time attempting to resolve it using an Http URL Connection.

**Running the project:**

In its current build, the best way to run the Proxy Server class is to load it into Eclipse and pressing the run button. The server is primed to listen to the localhost at 555, so that is what you specifically need in order to send requests. Once the project is running, open a browser like Firefox or Internet Explorer and navigate the the LAN proxy settings. From there, you can set your connection to localhost with port number 555. At this point, the project should recognize your chosen browser as a connecting client and will be attempting to resolve any requests it sends.

**Difficulties:**

Implementing our own java.net proxy server was the major challenge of this project. Our initial builds allowed us to send client requests to the server, but we had trouble getting the response from the server back to the client. The socket connection proved to be very inconsistent and would close before we wanted it to. We decided to follow more conventional code examples and split the server and client interaction into two separate classes. We solved our problems this way instead of trying to invent a proxy server ourselves. After that started working we needed to integrate the logging and caching classes into the proxy. Creating a custom log file and getting the right information (IP address, byte size) was a bit difficult. Caching was also a bit of a learning process. In the end we learned a lot about setting up client-server connections and the complexities involved.

**High level UML Class Diagram:**

