**SHUTTLE**

A Basic Web Server

Kyle Nickel

Daman Camara

Matthew Shortall

Zachary Sorenson

University of Maryland University College

Dr. Duchon

CMSC-495

Table of Contents

1 Project Plan 2

1.1 Introduction 2

1.1.1 Purpose 2

1.1.2 Scope 2

1.2 Definitions 2

1.3 Skills Required 3

1.4 Environment 3

1.5 Milestones 3

1.5.1 Overview and Assignments 3

1.5.2 Daemon 4

1.5.3 Logger 4

1.5.4 Cacher 4

1.5.5 RequestHandler 5

1.6 Testing 5

1.6.1 Unit Testing 5

1.6.2 Application Testing 5

1.6.3 Test Site 5

# 1 Project Plan

## 1.1 Introduction

### 1.1.1 Purpose

The Shuttle web server project is designed to be a basic web server that responds to HTTP GET requests. It will be able to handle multiple concurrent connections utilizing Java threading. Shuttle will also cache the most recent hits to increase the speed of returning requests.

### 1.1.2 Scope

Shuttle is not meant to be on the same scale as Apache HTTPD, but instead a small basic server. It will provide concurrency for multiple connections and respond to HTTP GET requests with parsing of parameters. Additionally, it will log all connections and served pages.

Shuttle will also implement a basic caching system to quickly return recently requested pages.

This document shall outline the components of the Shuttle web server to include the Daemon, Logger, Cacher and RequestHandler. The document has been written with language intended for use by system architects, designers and developers.

## 1.2 Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| GET | A HTTP request providing parameters in the URL. |
| Caching | Keeping resources in memory for fast access. |
| Unit Testing | Testing on a method level for each class. |
| Git | A source code manager. |
| GitHub | A web service allowing for collaboration with Git. |
| Daemon | A master process that runs without user intervention. |

## 1.3 Skills Required

Java

Socket Programming (Java)

Unit Testing

Application Testing

Caching Algorithms

File I/O

Git

## 1.4 Environment

The main environment used for the development of Shuttle will be Java 7 (currently update 45). This will allow Shuttle to run on a wide variety of platforms while mitigating development architecture conflicts.

The codebase will be stored in a Git repository on GitHub.com at <https://github.com/nickelkr/shuttle>. Each participant should fork this repository and any changes committed back to this master branch.

For testing purposes, the Shuttle project will be using the JUnit test framework. This framework is well documented and widely used for Java unit tests. For validation during application testing, Firefox v25.0 and Internet Explorer v10 will be used. Additional browser versions shall be utilized when appropriate.

## 1.5 Milestones

### 1.5.1 Overview and Assignments

The Shuttle project is made up of the following components:

|  |  |
| --- | --- |
| **Component** | **Assigned To** |
| Daemon | Kyle |
| Logger | Zack |
| Cacher | Matt |
| RequestHandler | Daman |
| Unit Tests | Everyone |
| Sample Web Site | Everyone |
| Application Test | Everyone |

### 1.5.2 Daemon

The Shuttle daemon is the master process. This part of the application provides the user with the ability to start the application and provide options like port number.

It will also be the class handling the incoming connections. As Shuttle receives a connection request, it will be in charge of creating a new thread and handling the request with a RequestHandler instance.

This daemon will also be in charge of managing all sub-processes for RequestHandler, Logger and Caching instances.

When the user kills the Shuttle daemon he or she is stopping the server entirely.

### 1.5.3 Logger

The Logger is in charge of ensuring all major processes are documented and saved to the file system.

Any connections made to Shuttle should be documented with the corresponding IP address. It should also log any pages that are sent out to such IP addresses.

Changes to the state of the Shuttle daemon should also be logged: starting, stopping.

The Logger should provide an API for each component of Shuttle to provide logging information. The Logger itself should handle time-stamping and naming of each log line to differentiate component logging.

### 1.5.4 Cacher

The Cacher is responsible for keeping recently requested pages in memory. As pages are requested by outside systems, the RequestHandler should first look to the Cacher to see if Shuttle currently has the page in memory. If the page is stored in memory, Shuttle can serve the page without performing slower I/O operations. Otherwise, the RequestHandler should retrieve the page and provide the Cacher with the served page.

The Cacher should implement a Least Recently Used algorithm, LRU. It will be responsible for swapping pages in and out of memory, and also responding to page requests from the RequestHandler returning either the page requested or a fault.

### 1.5.5 RequestHandler

The RequestHandler is responsible for handling a single HTTP GET request per instance. The Shuttle daemon will create an instance of the RequestHandler and pass the request in.

The RequestHandler should parse the request, along with any parameters, and then return the requested page from either the cache or the file system. If the page is not found, or another error occurs, the RequestHandler is responsible for conveying the issue with standard HTTP codes.

## 1.6 Testing

### 1.6.1 Unit Testing

Each team member shall write unit tests for each method in the class that they are responsible for developing.

The tests shall be written using the JUnit testing framework. Before committing any work to the production branch, all tests should pass JUnit testing. Additionally, all tests should meet any specific pass/fail criteria specified in separate documentation.

### 1.6.2 Application Testing

A small program is required to test Shuttle as a whole. This program shall use threading to create multiple connections and request many pages in quick succession. This will demonstrate the ability to concurrently handle connections, as well as test the caching system and page retrieval.

### 1.6.3 Test Site

To fully test Shuttle, a test web site will be required. This web site shall consist of approximately 10 web pages. These pages will provide basic information, as well as documentation about the Shuttle web server. This will provide the project with a simple static website along with pages to use while testing.