# 1. Introduction

### 1.1 Purpose

The purpose of this test plan is to ensure the proper functionality of the Proxy Server and Web Server.

### 1.2 Scope

The scope of this test plan covers the following:

Proper communication between the Proxy Server, Web Server, and Client.

Handling of HTTP status codes in both the proxy and web servers.

Successful retrieval of resources by the client from the web server through the proxy server.

# 2. Features to be Tested

### 2.1 Proxy Server

Verify that the proxy server can accept incoming connections from clients.

Confirm that the proxy server can establish connections to the specified web server.

Ensure bidirectional communication between clients and the web server through the proxy server.

Check if the proxy server handles errors and exceptions appropriately.

### 2.2 Web Server

Confirm that the web server can handle incoming requests from the proxy server.

Verify that the web server returns the correct HTTP status codes (e.g., 200, 404, 500) based on the resource availability.

Test the web server's ability to send resource content to the client.

### 2.3 Client

Verify that the client can send requests to the proxy server.

Confirm that the client receives the responses from the proxy server.

Check how the client handles different HTTP status codes.

# 3. Test Environment

## 3.1 Software

Go programming language (version 1.21+)

Visual studio code

### 3.2 Hardware

Ensure that the testing environment has sufficient resources to run the servers and clients simultaneously.

# 4. Test Cases

## 4.1 Proxy Server

* Proxy server starts successfully and listens on the specified port (1234).
* Proxy server establishes a connection to the web server.
* Proxy server handles multiple client connections concurrently.
* Proxy server handles errors gracefully.

## 4.2 Basic Web Server

* Web server starts successfully and listens on the specified port (1235).
* Web server returns the correct HTTP status codes for existing and non-existing resources.
* Web server sends resource content to the client upon request.

## 4.3 Client

* Client sends a valid request to the proxy server.
* Client displays the content received from the proxy server.
* Client handles different HTTP status codes appropriately.

# 5. Test Execution

## 5.1 Pre-conditions

Ensure that all necessary dependencies are installed.

Start the proxy server and web server before executing client tests.

Use go run main.go in each respective folder to run the files, start with server, then proxy and later client.

## 5.2 Execution Steps

### 5.2.1 Test from browser

To test from the browser type 127.0.0.1:1234/testfile.html or 127.0.0.1:1234/testfile.jpg

This test is conclusive as both files are returned and displayed appropriately

### 5.2.2 Test from client

To test from client, change the value of resource URL in the code.

This test is conclusive as testfile.html and testfile.jpg are both returned and printed, though the printed text for the image is jumbled as it is in binary.

### 5.2.3 Test with given URLs

The test with <http://www.google.com/> is successful as text is returned and printed in the console

The test with <ftp://ftp.us.dell.com/readme.txt> fails as our server is handling http requests thus returning an unsupported protocol error.

# 6. Conclusion

In conclusion, the Proxy Server and Web Server have been thoroughly tested, and the results are in alignment with the expected functionality. Key findings include:

The Proxy Server successfully handles incoming connections, establishes connections to the web server, and ensures bidirectional communication.

The Web Server demonstrates successful handling of requests, returning accurate HTTP status codes, and sending resource content to the client.

The Client effectively sends requests to the proxy server, displays received content and handles various HTTP status codes appropriately.

Overall, the Proxy Server and Web Server function as expected, meeting the requirements outlined in the test plan.