**Project Report  
COMP 2322 Computer Networking  
Multi-threaded Web Server**

**Name: Wen Siqi  
Student ID: 23102039d**

**Design and Implementation Summary**

1. **Myhttp library**

**http\_request and http\_response:**

This is the basic function library that supports the HTTP protocol, implemented by me. Inside this library, there are 7 classes, of which the http\_request and http\_response classes are the two major classes that process HTTP requests and responses separately. They both have built-in methods to parse and generate requests and responses according to their instance variables. Most methods here are used to set or get the variables of requests and responses. If they find that any config is lacking in request or response, they will raise an exception, and the server will send 400 BadRequest to the client if necessary.

**File\_handler，FileTypeManager:**

These two classes are the two major IO and file handler classes. They get information from a request or response, sometimes the built-in method may ask for parameters whose type is http\_request or http\_response. FileTypeManager is the class that manages file types and maps Content-Type with file extension. This server project is extension-sensitive. The FileTypeManager class will check whether the extension is supported and whether the extension and Content-Type are matched.

If not supported, the class will raise an UnsupportedMediaType exception, and the server will send a 415 code and a matched HTML body if necessary. And if not matched, the class will raise a BadRequest exception and send a 400 code with a BadRequest HTML body.

File\_Handler is the major IO class. It can check whether the file requested exists, what is the last modification time and get the file content based on the file type. There is also a built-in method check(). This method will check the permission, file exists, and the type supported in order. Here we check the permission by using the absolute path and check the root prefix to determine whether the requested file is allowed to access, if not, the class will raise a PermissionError and send 403 forbidden to the client.

**Cache\_Table\_Manager**

This class is used to manage the cache and the cache table. The cache table is stored as a JSON file on the disk of the client. This will serialize the JSON string to a dictionary type instance variable. This class manage the update and check of the cache.

1. Server Architecture

The server is implemented in Python using socket programming and threading to handle multiple client requests concurrently. Key components include:

* Multi-threading: Each client connection is handled by a separate thread, allowing simultaneous processing of requests.
* HTTP Protocol Compliance: Supports GET and HEAD methods, handles headers like Connection, If-Modified-Since, and returns appropriate status codes (200, 304, 400, 403, 404, 415, 500).
* File Handling: Retrieves files from the server’s file system, distinguishes between text (e.g., HTML, CSS) and image (e.g., PNG, JPEG) types, and enforces MIME type validation.
* Caching: Implements a cache manager to store and validate files using Last-Modified timestamps.
* Error Handling: Catches exceptions such as file not found, permission errors, and unsupported media types, returning corresponding HTTP statuses.

2. Client Implementation

The client sends HTTP requests based on user input and processes server responses:

* CLI Interface: Accepts commands like GET /path file\_type(b/t) [connection] and constructs HTTP requests.
* Cache Management: Checks cached files using a JSON-based cache table and updates it based on server responses.
* Response Handling: Parses server responses, saves files to the cache folder, and handles 304 Not Modified scenarios.

3. Key Features

* Persistent Connections: Supports keep-alive and close via the Connection header.
* Header Fields: Validates If-Modified-Since to avoid redundant transfers and uses Last-Modified for cache freshness.
* Logging: Records client IPs, timestamps, requested files, and response statuses in a daily log file (e.g., server\_YYYYMMDD.log).

Demonstration of Execution

*(Screenshots and results are omitted as per instructions.)*

Test Cases

1. GET Request for Text File:
   * Command: GET /sample.txt t
   * Server returns 200 OK with the file content.
2. HEAD Request for Image:
   * Command: HEAD /image.png b
   * Server returns headers without the body.
3. Error Handling:
   * 404 Not Found: Requesting a non-existent file.
   * 415 Unsupported Media Type: Requesting an unsupported file format.
   * 304 Not Modified: Cached file matches server’s Last-Modified timestamp.
4. Connection Management:
   * Using Connection: close terminates the session after one request.

Log File

*(Sample log entries are omitted. The log file server\_YYYYMMDD.log includes:)*

* UTC and local timestamps.
* Client IP addresses.
* Requested resources.
* Response status codes (e.g., 200 OK, 404 Not Found).