**WebX - Multithreaded Web Proxy Server (C++)**

**🚀 Timeline: 10 Days**

**📌 Project Breakdown**

**WebX is a high-performance multithreaded web proxy server that:  
✅ Intercepts & forwards HTTP requests from clients.  
✅ Fetches web pages from target servers.  
✅ Caches responses for efficiency (LRU cache).  
✅ Handles multiple requests using a thread pool.  
✅ Implements FIFO scheduling to manage request order.**

**By the end, WebX should work as a real proxy:  
✔ Clients (browsers or scripts) send requests through WebX.  
✔ WebX fetches the requested page, caches it, and sends it to the client.  
✔ Subsequent requests for the same page are served from cache, reducing load time.**

**📆 10-Day Implementation Plan**

**Each day is structured for parallel work (for teamwork, if applicable).**

**Day 1: Project Setup & Research**

**✅ Finalize project structure (folders, files, Makefile).  
✅ Set up a basic TCP server that accepts client connections.  
✅ Research proxy behavior, HTTP request handling, and caching techniques.**

**Day 2: Basic Proxy Server Core**

**✅ Implement server.cpp:**

* **Create a socket, bind to a port, listen for connections.**
* **Accept incoming client connections.**

**✅ Implement client.cpp:**

* **Allow clients to send HTTP requests to WebX.**
* **Implement basic request parsing (utils.cpp).**

**✅ Test basic proxy request forwarding (send requests via WebX, forward them to the target server).**

**Day 3: Multithreading & Thread Pool**

**✅ Implement thread\_pool.cpp:**

* **Worker threads wait for tasks.**
* **Use pthread\_create(), pthread\_mutex\_t, pthread\_cond\_t.**

**✅ Modify server.cpp to delegate incoming requests to the thread pool.  
✅ Test with multiple concurrent client connections.**

**Day 4: FIFO Scheduling**

**✅ Implement FIFO scheduling (First-In-First-Out request handling).  
✅ Store pending requests in a queue.  
✅ Ensure requests are served in the correct order.**

**Day 5: Caching (LRU Implementation)**

**✅ Implement cache.cpp:**

* **Store HTTP responses.**
* **Use hash map + doubly linked list for LRU (efficient eviction).**

**✅ Modify server.cpp to serve cached responses if available.  
✅ Test caching performance by requesting the same page multiple times.**

**Day 6: Advanced Features**

**✅ Implement logger.cpp for request logging.  
✅ Implement rate limiting using semaphores (limit max simultaneous connections).  
✅ Add support for large files (handle chunked transfer).**

**Day 7: Testing & Debugging**

**✅ Test WebX with multiple clients (test\_client.cpp).  
✅ Verify cache performance.  
✅ Debug potential synchronization issues (race conditions, deadlocks).**

**Day 8: Performance Optimization**

**✅ Optimize thread synchronization (reduce unnecessary locking).  
✅ Optimize cache replacement policy.  
✅ Profile execution time and improve performance.**

**Day 9: Final Testing & Browser Integration**

**✅ Configure a web browser to use WebX as a proxy.  
✅ Test real website loading through WebX.  
✅ Validate request handling, caching, and concurrency.**

**Day 10: Documentation & Final Debugging**

**✅ Update README.md with setup instructions, features, usage guide.  
✅ Perform final debugging and demonstration recording.  
✅ 🚀 WebX is complete and ready to showcase!**

**🛠 Final Execution Plan**

**How to Compile & Run WebX**

**make**

**./bin/webx 8080 # Run WebX on port 8080**

* **WebX starts listening for incoming requests.**
* **Clients (browsers, curl, test scripts) can send requests through WebX.**

**How to Test WebX**

**✅ Use curl to send a request:**

**curl -x http://localhost:8080 http://example.com**

**✅ Configure a browser to use WebX as a proxy and visit websites.  
✅ Check cache hits/misses in the logs (logger.cpp).**

**🚀 Final Goal**

**WebX should work as a real proxy server:  
✅ Accepts client requests.  
✅ Fetches webpages from the internet.  
✅ Caches frequently accessed pages.  
✅ Serves cached content when possible.  
✅ Handles multiple requests efficiently.**

**This structure remains unchanged and is strictly followed.  
Tomorrow: We start with advanced client implementation! 🔥**

Full plan:  
  
WebX/

│── src/ # Source code files

│ ├── main.cpp # Main entry point

│ ├── server.cpp # Core proxy server implementation

│ ├── client.cpp # Basic client implementation

│ ├── thread\_pool.cpp # Thread pool implementation

│ ├── cache.cpp # LRU caching implementation

│ ├── logger.cpp # Logging functionality

│ ├── utils.cpp # Utility functions

│── include/ # Header files

│ ├── definitions.h # Global definitions and prototypes

│ ├── server.h # Server header file

│ ├── client.h # Client header file

│ ├── thread\_pool.h # Thread pool header file

│ ├── cache.h # LRU cache header file

│ ├── logger.h # Logger header file

│ ├── utils.h # Utility functions header

│── test/ # Testing files

│ ├── test\_client.cpp # Test client to check server functionality

│── include/ # HTML files (if needed)

│── Makefile # Compilation script

│── README.md # Project documentation