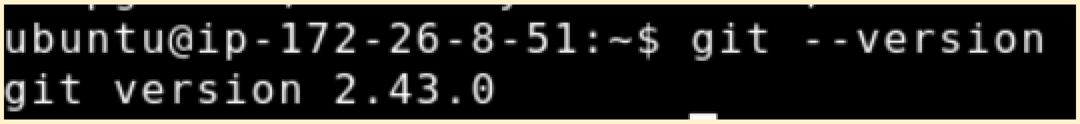
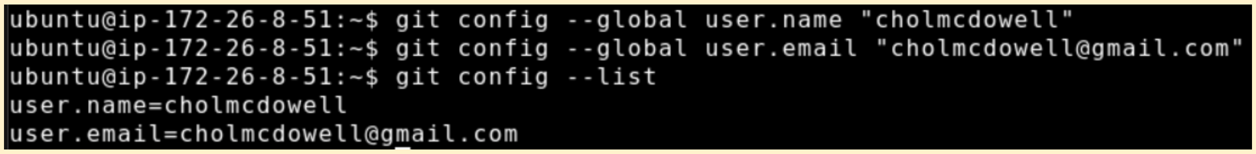
CSCE 412 700 - Project 3 Load Balancer

Zachary McDowell
227007875

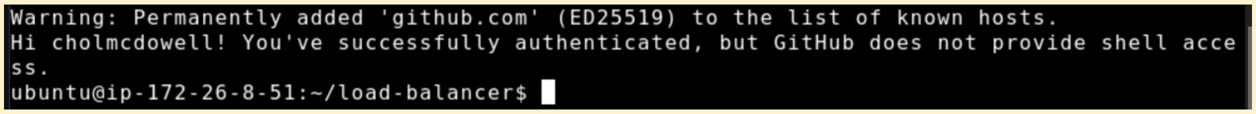
Initialization:



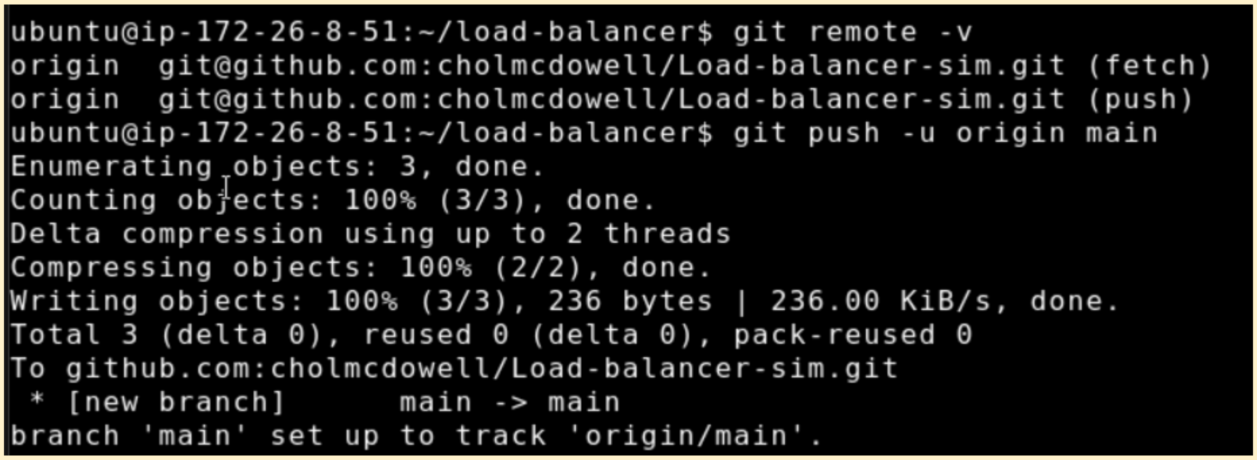
```
ubuntu@ip-172-26-8-51:~$ git --version
git version 2.43.0
```



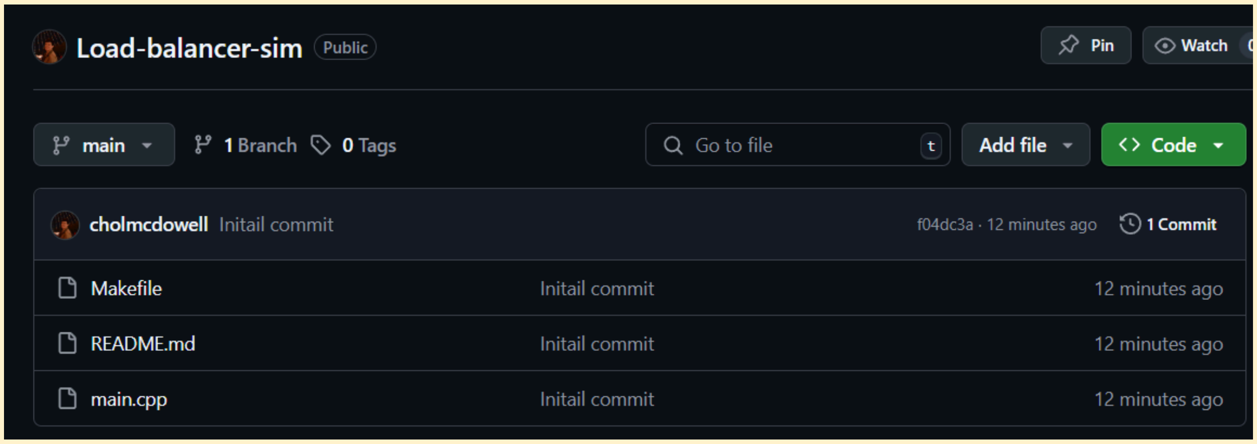
```
ubuntu@ip-172-26-8-51:~$ git config --global user.name "cholmcdowell"
ubuntu@ip-172-26-8-51:~$ git config --global user.email "cholmcdowell@gmail.com"
ubuntu@ip-172-26-8-51:~$ git config --list
user.name=cholmcdowell
user.email=cholmcdowell@gmail.com
```



```
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi cholmcdowell! You've successfully authenticated, but GitHub does not provide shell acce
ss.
ubuntu@ip-172-26-8-51:~/load-balancer$
```



```
ubuntu@ip-172-26-8-51:~/load-balancer$ git remote -v
origin  git@github.com:cholmcdowell/Load-balancer-sim.git (fetch)
origin  git@github.com:cholmcdowell/Load-balancer-sim.git (push)
ubuntu@ip-172-26-8-51:~/load-balancer$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 236 bytes | 236.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cholmcdowell/Load-balancer-sim.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



Load-balancer-sim Public

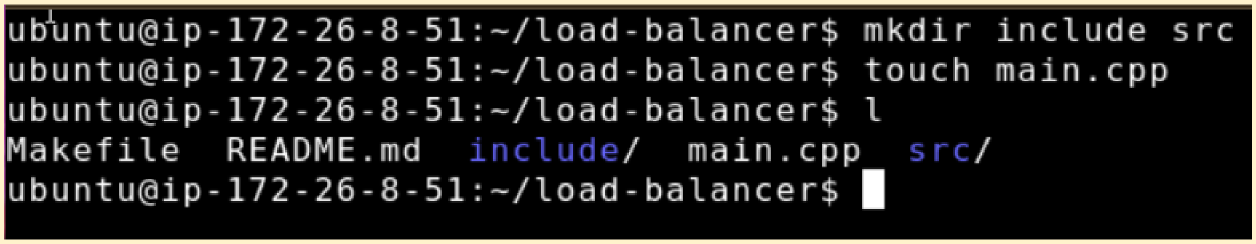
main · 1 Branch · 0 Tags

cholmcdowell Initail commit · f04dc3a · 12 minutes ago · 1 Commit

Makefile	Initail commit	12 minutes ago
README.md	Initail commit	12 minutes ago
main.cpp	Initail commit	12 minutes ago

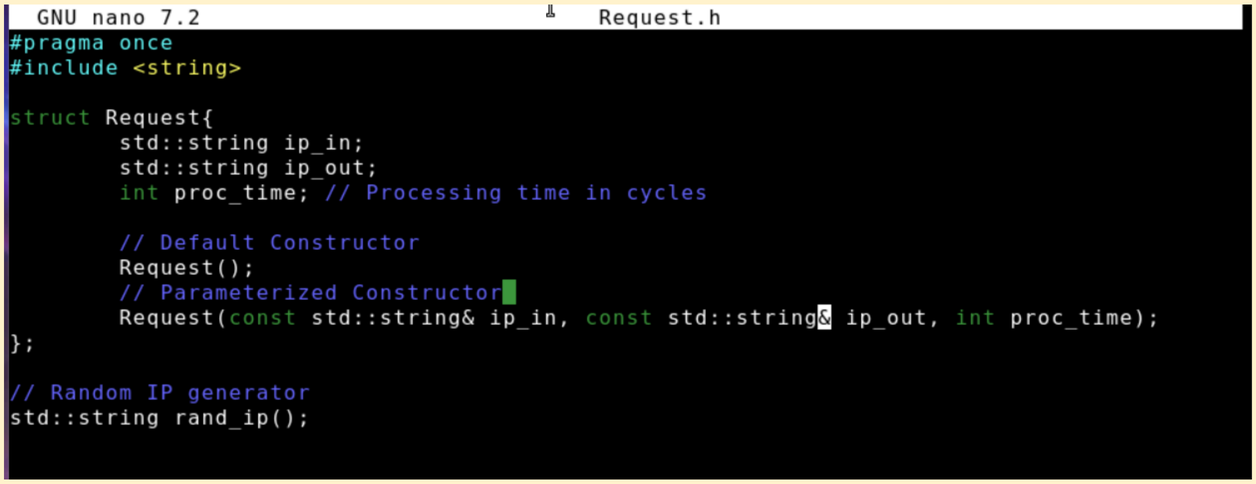
Installed git on my Ubuntu VM instance and configured it to my personal GitHub account. Created three essential files I will be needing for the project, and pushed them to my git repository to ensure connection with the VM.

Classes and Structs

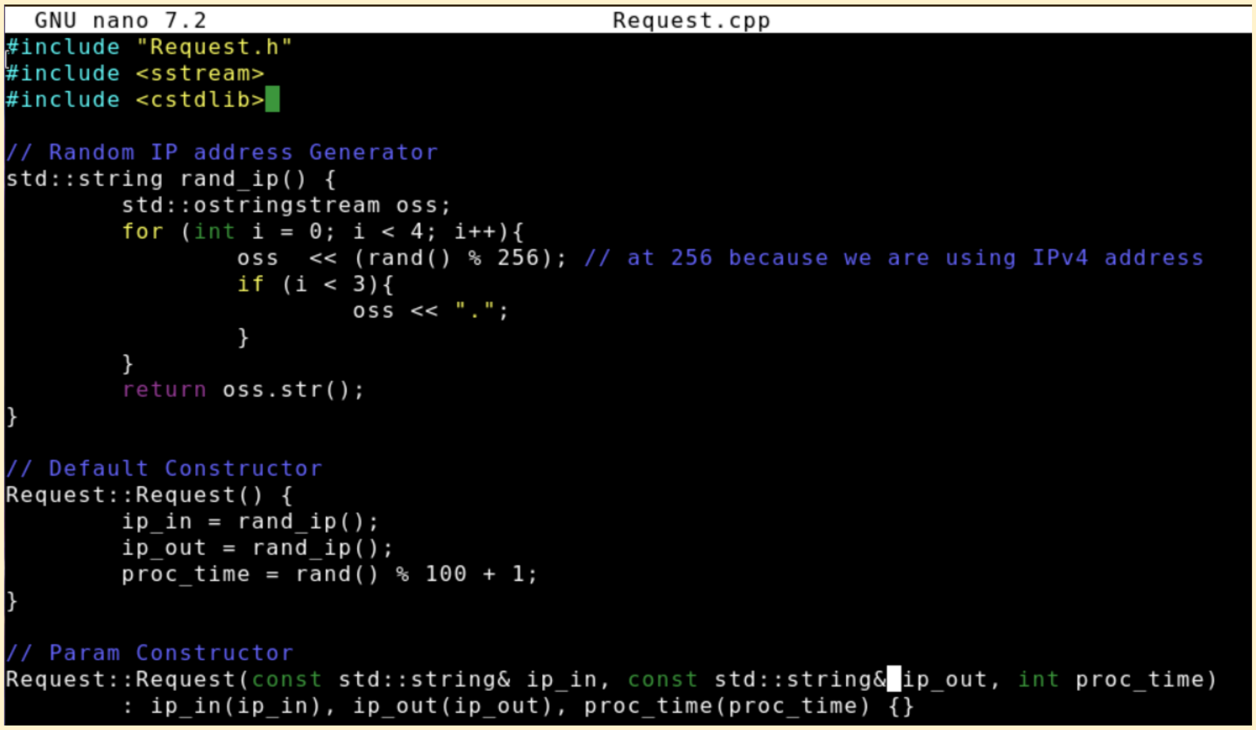


```
ubuntu@ip-172-26-8-51:~/load-balancer$ mkdir include src
ubuntu@ip-172-26-8-51:~/load-balancer$ touch main.cpp
ubuntu@ip-172-26-8-51:~/load-balancer$ l
Makefile  README.md  include/  main.cpp  src/
ubuntu@ip-172-26-8-51:~/load-balancer$ 
```

Created a folder for my class and struct to keep organized.

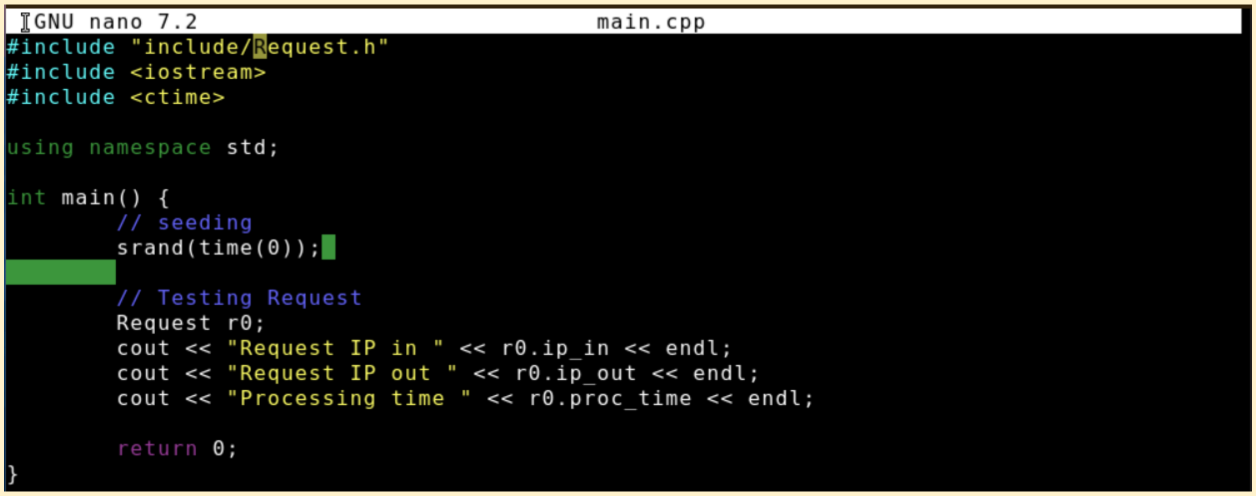


```
  GNU nano 7.2                              Request.h
#pragma once
#include <string>

struct Request{
        std::string ip_in;
        std::string ip_out;
        int proc_time; // Processing time in cycles

        // Default Constructor
        Request();
        // Parameterized Constructor
        Request(const std::string& ip_in, const std::string& ip_out, int proc_time);
};

// Random IP generator
std::string rand_ip();
```

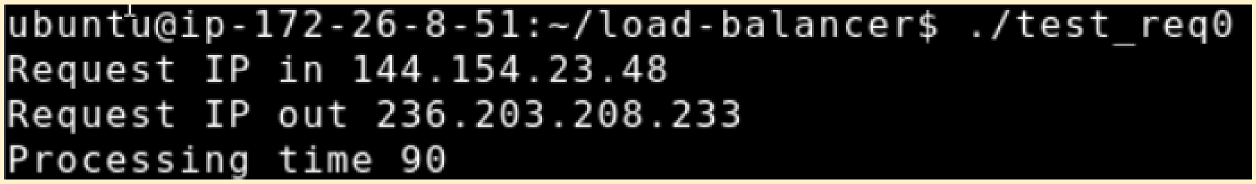


```
  GNU nano 7.2                              Request.cpp
#include "Request.h"
#include <sstream>
#include <cstdlib>

// Random IP address Generator
std::string rand_ip() {
        std::ostringstream oss;
        for (int i = 0; i < 4; i++){
                oss  << (rand() % 256); // at 256 because we are using IPv4 address
                if (i < 3){
                        oss << ".";
                }
        }
        return oss.str();
}

// Default Constructor
Request::Request() {
        ip_in = rand_ip();
        ip_out = rand_ip();
        proc_time = rand() % 100 + 1;
}

// Param Constructor
Request::Request(const std::string& ip_in, const std::string&ip_out, int proc_time)
        : ip_in(ip_in), ip_out(ip_out), proc_time(proc_time) {}
```

Created the Request struct and appropriate functionality.

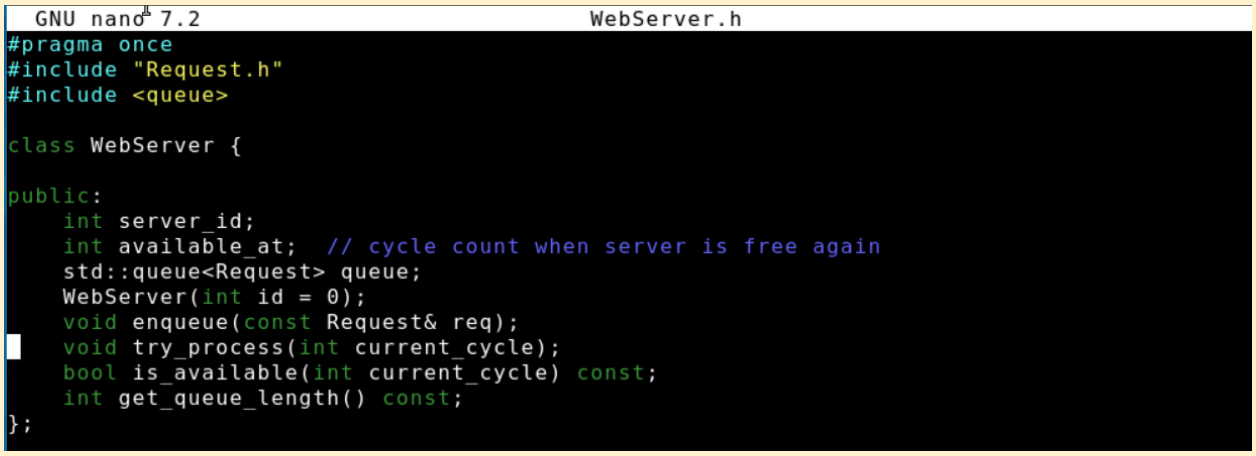


```
GNU nano 7.2                                    main.cpp
#include "include/Request.h"
#include <iostream>
#include <ctime>

using namespace std;

int main() {
        // seeding
        srand(time(0));

        // Testing Request
        Request r0;
        cout << "Request IP in " << r0.ip_in << endl;
        cout << "Request IP out " << r0.ip_out << endl;
        cout << "Processing time " << r0.proc_time << endl;

        return 0;
}
```

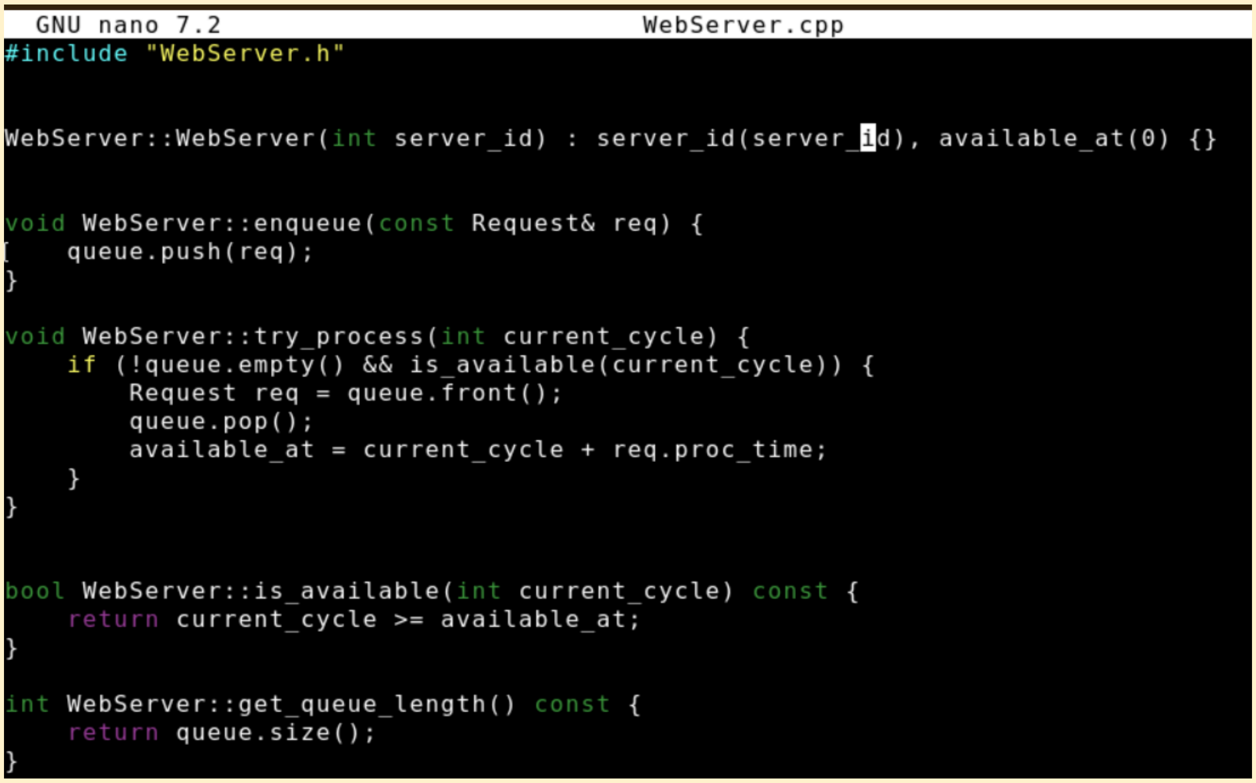


```
ubuntu@ip-172-26-8-51:~/load-balancer$ ./test_req0
Request IP in 144.154.23.48
Request IP out 236.203.208.233
Processing time 90
```

Testing Request struct in main.cpp.

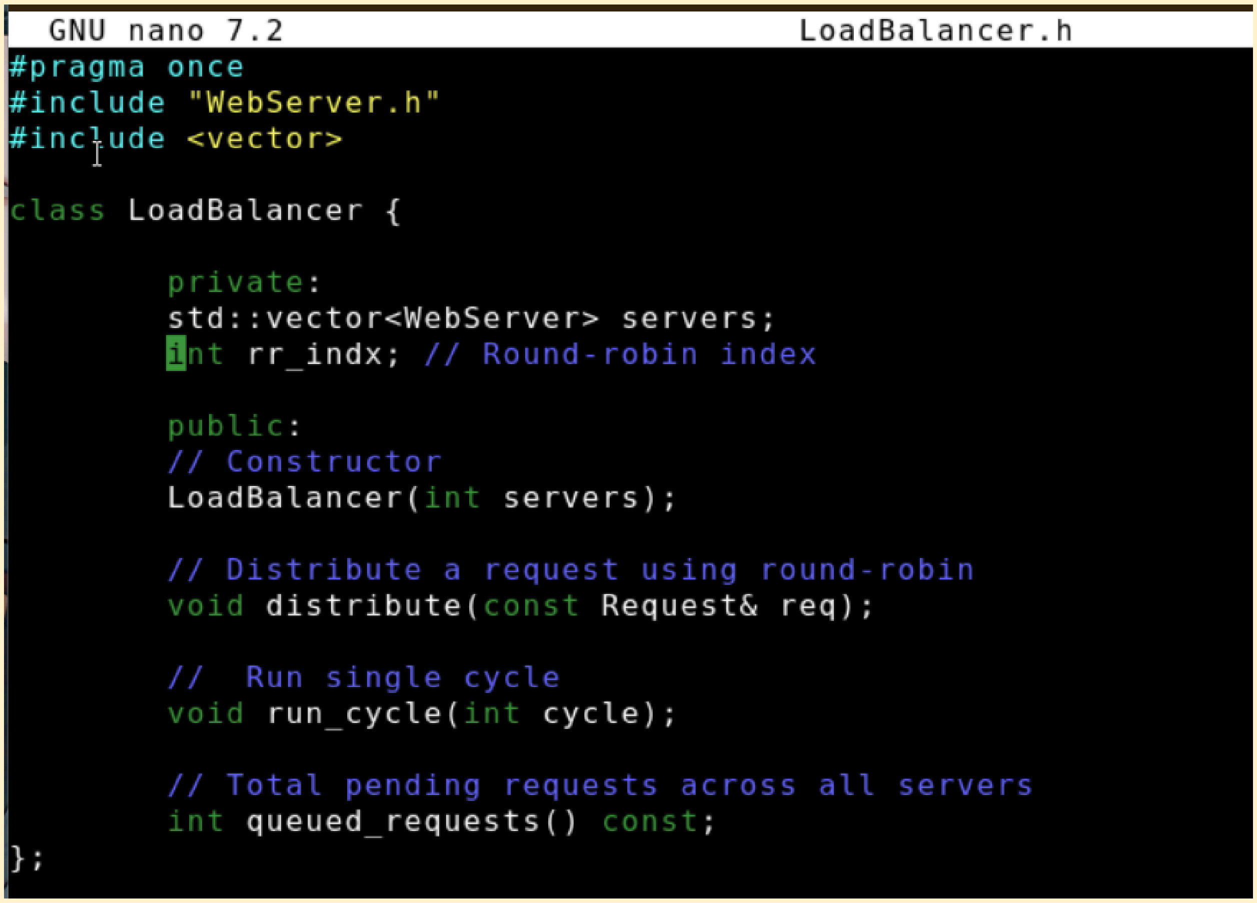


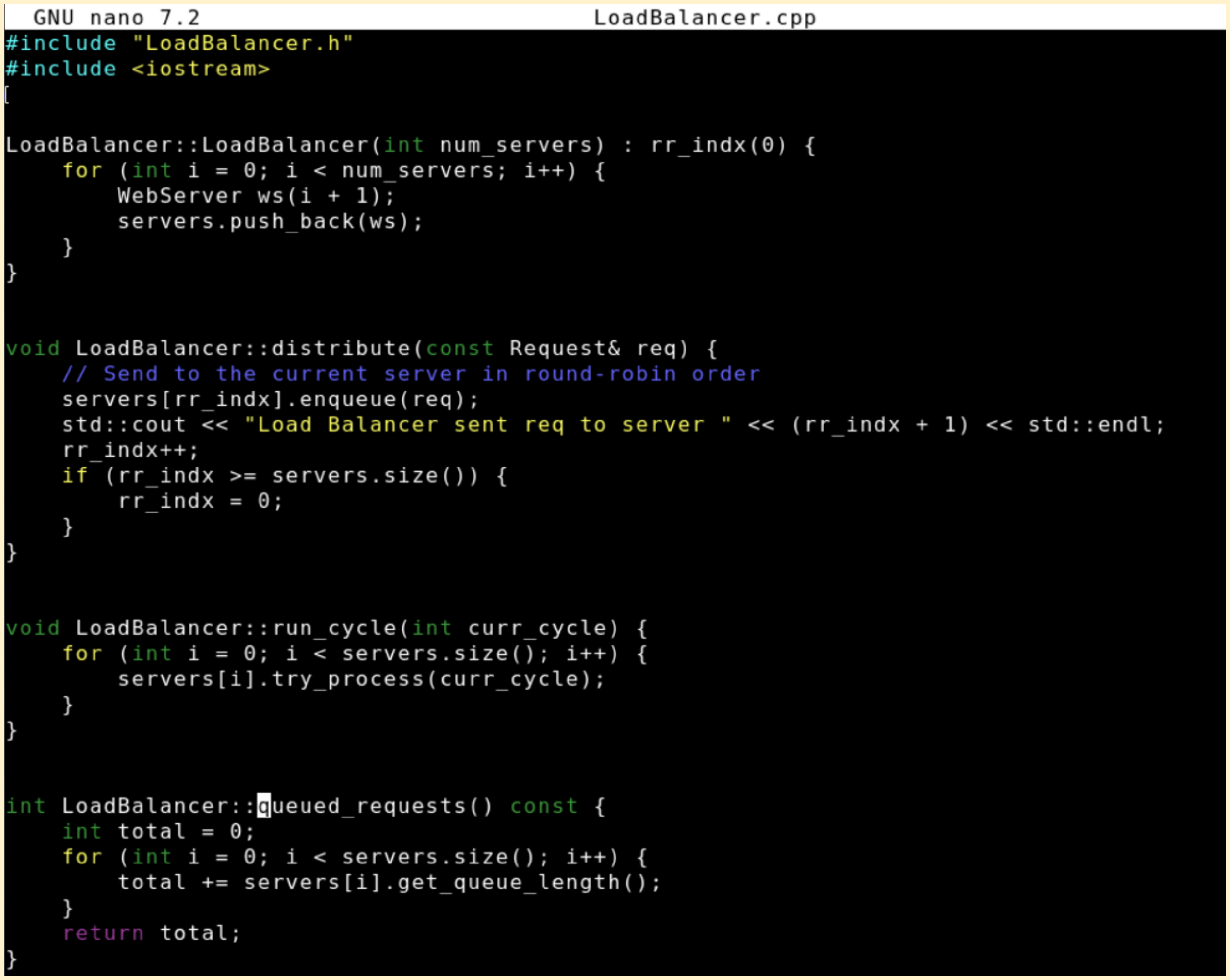
```
GNU nano 7.2                                    WebServer.h
#pragma once
#include "Request.h"
#include <queue>

class WebServer {

public:
    int server_id;
    int available_at;  // cycle count when server is free again
    std::queue<Request> queue;
    WebServer(int id = 0);
    void enqueue(const Request& req);
    void try_process(int current_cycle);
    bool is_available(int current_cycle) const;
    int get_queue_length() const;
};
```



```
  GNU nano 7.2                                   WebServer.cpp
#include "WebServer.h"


WebServer::WebServer(int server_id) : server_id(server_id), available_at(0) {}


void WebServer::enqueue(const Request& req) {
    queue.push(req);
}

void WebServer::try_process(int current_cycle) {
    if (!queue.empty() && is_available(current_cycle)) {
        Request req = queue.front();
        queue.pop();
        available_at = current_cycle + req.proc_time;
    }
}


bool WebServer::is_available(int current_cycle) const {
    return current_cycle >= available_at;
}

int WebServer::get_queue_length() const {
    return queue.size();
}
```

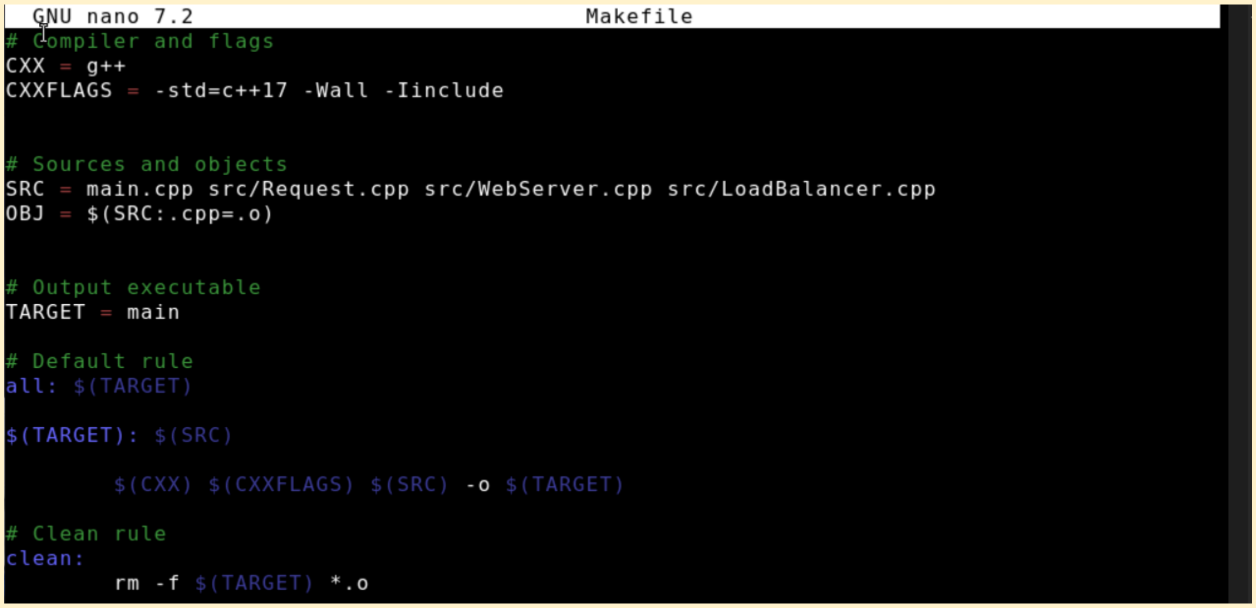
Created the WebServer class and it’s functionality.


```
  GNU nano 7.2                          LoadBalancer.h
#pragma once
#include "WebServer.h"
#include <vector>

class LoadBalancer {

        private:
        std::vector<WebServer> servers;
        int rr_indx; // Round-robin index

        public:
        // Constructor
        LoadBalancer(int servers);

        // Distribute a request using round-robin
        void distribute(const Request& req);

        //  Run single cycle
        void run_cycle(int cycle);

        // Total pending requests across all servers
        int queued_requests() const;
};
```

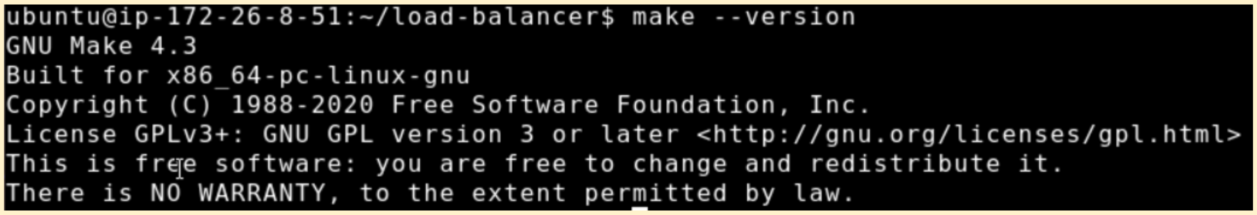


```
  GNU nano 7.2                                        LoadBalancer.cpp
#include "LoadBalancer.h"
#include <iostream>
[

LoadBalancer::LoadBalancer(int num_servers) : rr_indx(0) {
    for (int i = 0; i < num_servers; i++) {
        WebServer ws(i + 1);
        servers.push_back(ws);
    }
}


void LoadBalancer::distribute(const Request& req) {
    // Send to the current server in round-robin order
    servers[rr_indx].enqueue(req);
    std::cout << "Load Balancer sent req to server " << (rr_indx + 1) << std::endl;
    rr_indx++;
    if (rr_indx >= servers.size()) {
        rr_indx = 0;
    }
}


void LoadBalancer::run_cycle(int curr_cycle) {
    for (int i = 0; i < servers.size(); i++) {
        servers[i].try_process(curr_cycle);
    }
}


int LoadBalancer::queued_requests() const {
    int total = 0;
    for (int i = 0; i < servers.size(); i++) {
        total += servers[i].get_queue_length();
    }
    return total;
}
```

Created LoadBalancer class and functionality.

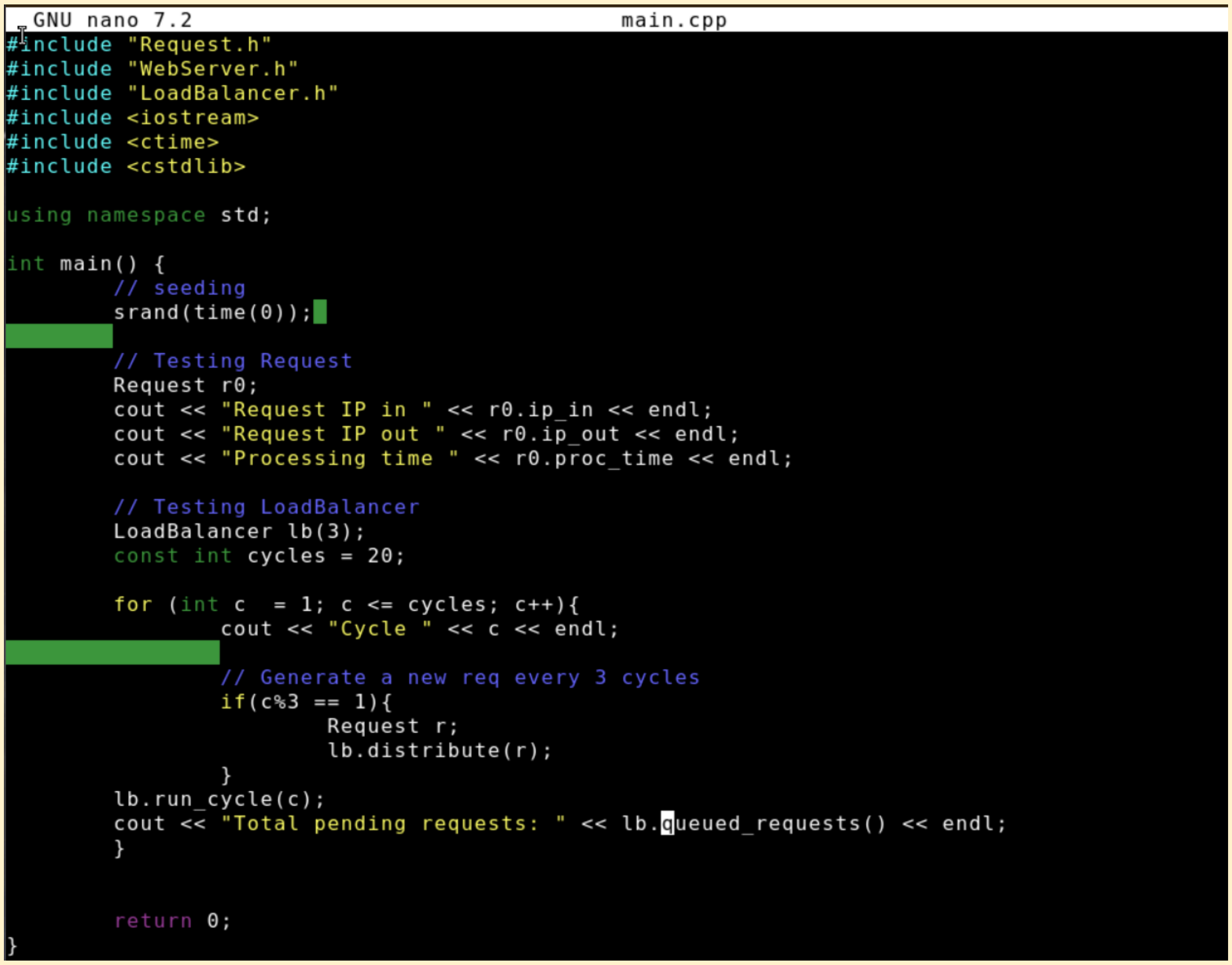


```
  GNU nano 7.2                                    Makefile
# Compiler and flags
CXX = g++
CXXFLAGS = -std=c++17 -Wall -Iinclude


# Sources and objects
SRC = main.cpp src/Request.cpp src/WebServer.cpp src/LoadBalancer.cpp
OBJ = $(SRC:.cpp=.o)


# Output executable
TARGET = main

# Default rule
all: $(TARGET)

$(TARGET): $(SRC)

        $(CXX) $(CXXFLAGS) $(SRC) -o $(TARGET)

# Clean rule
clean:
        rm -f $(TARGET) *.o
```

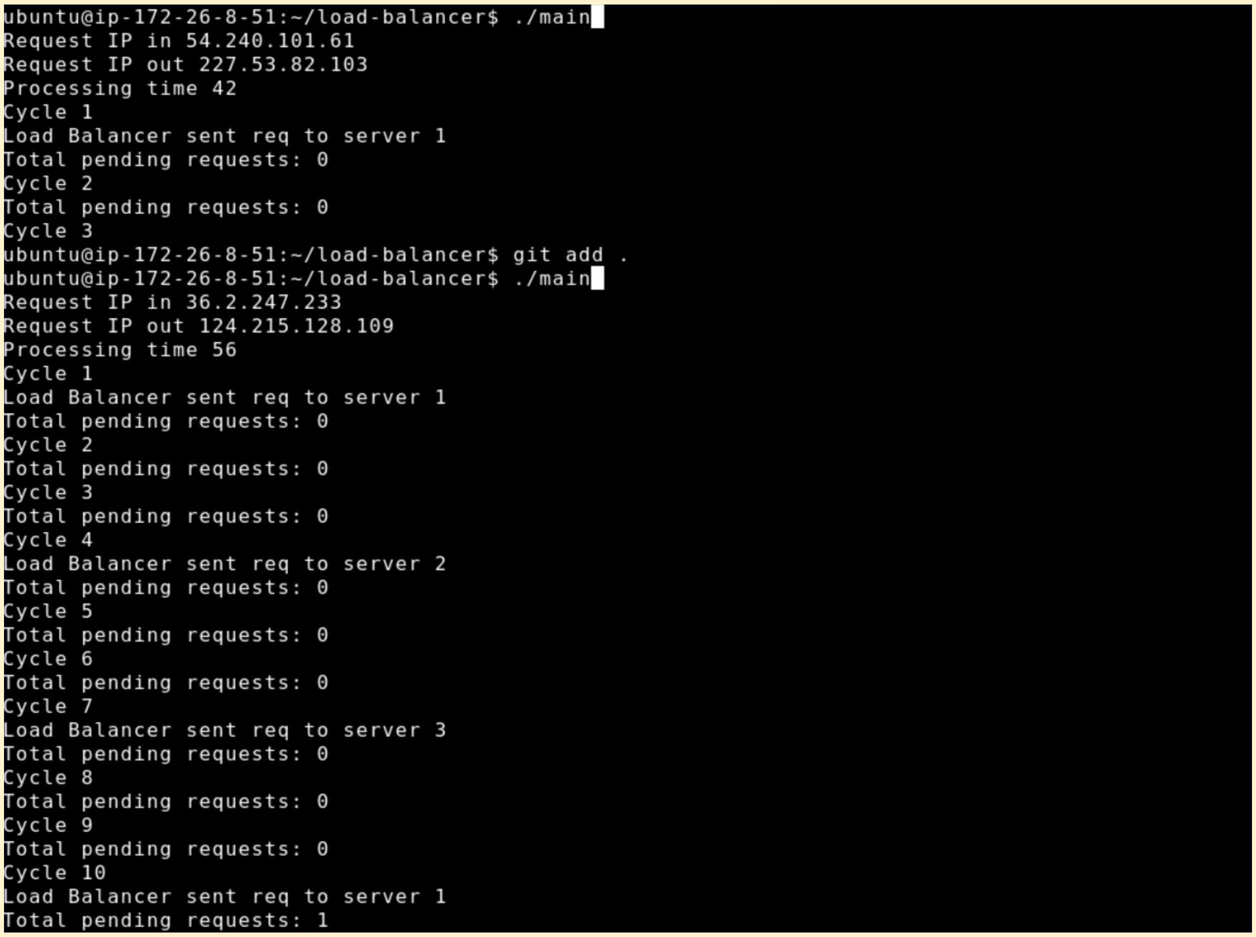
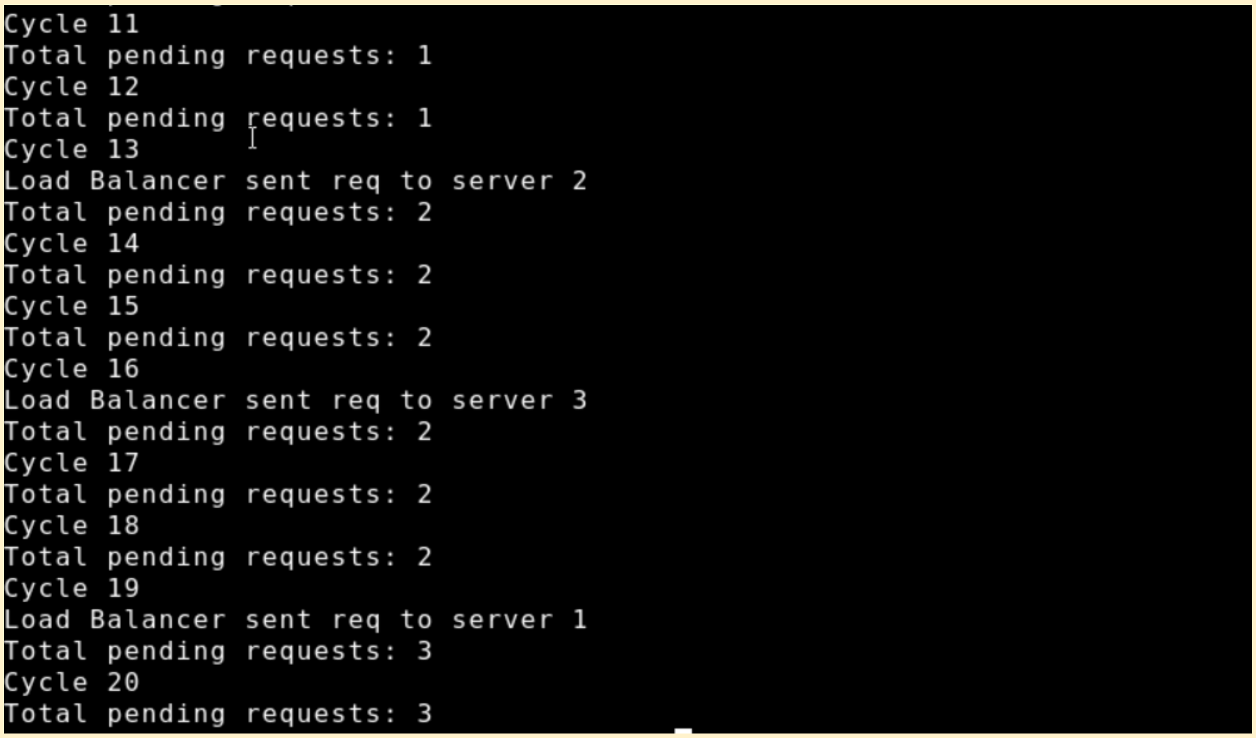


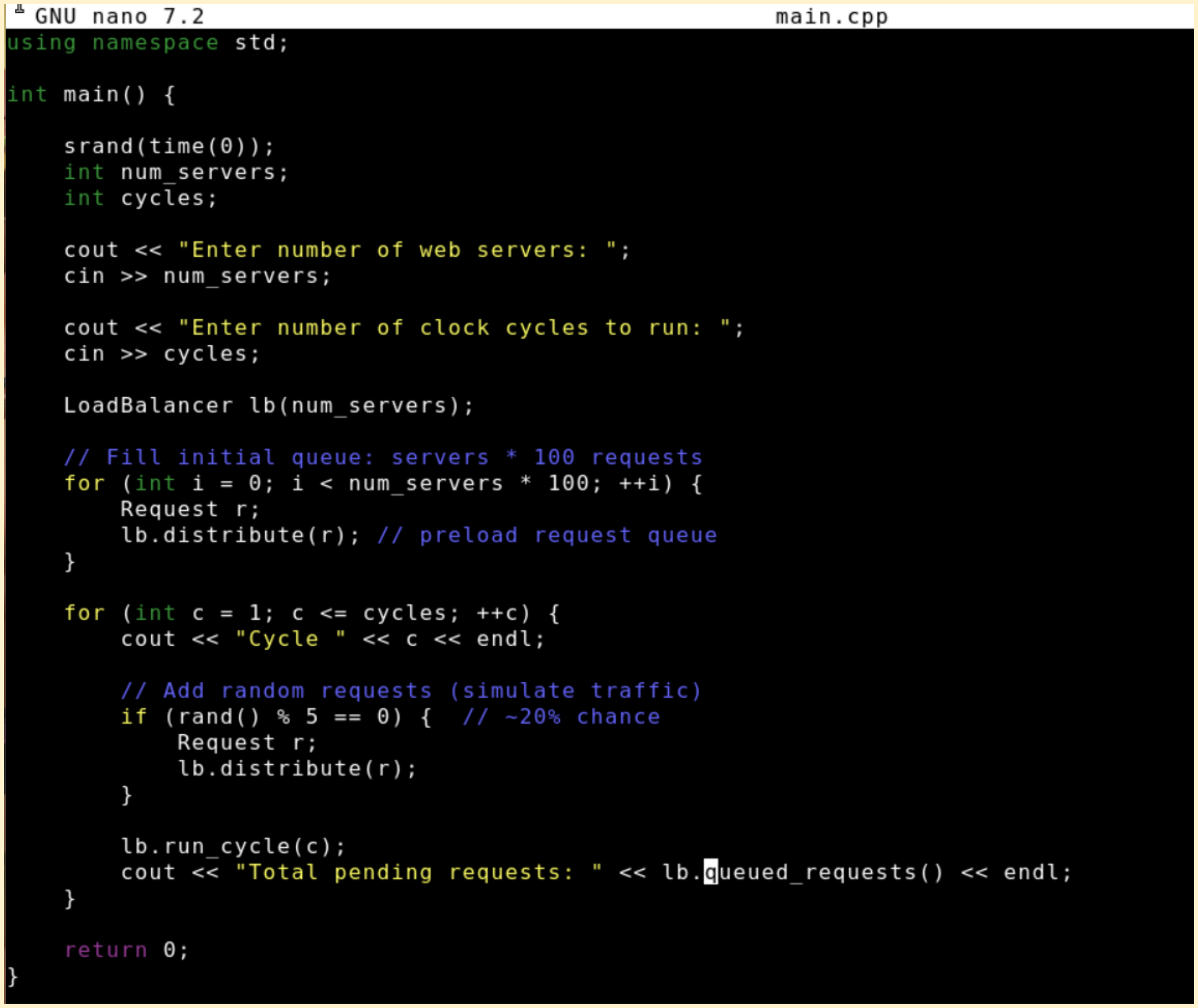
```
ubuntu@ip-172-26-8-51:~/load-balancer$ make --version
GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Creating Makefile.



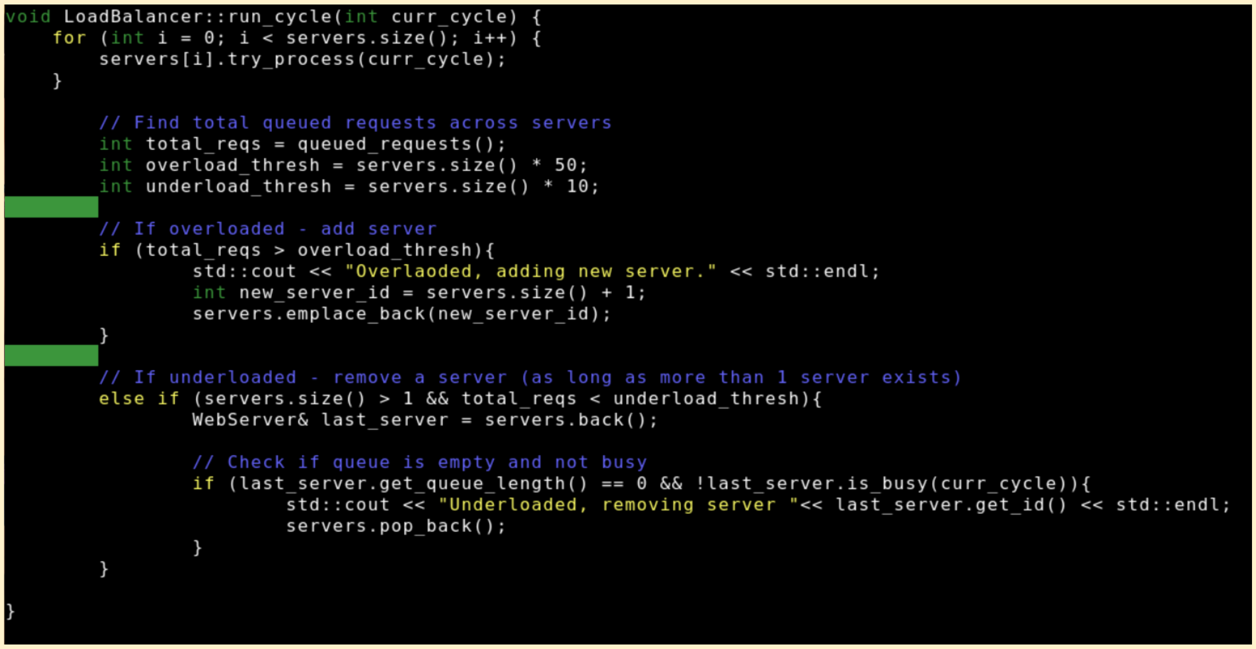
```
  GNU nano 7.2                                   main.cpp
#include "Request.h"
#include "WebServer.h"
#include "LoadBalancer.h"
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

int main() {
        // seeding
        srand(time(0));

        // Testing Request
        Request r0;
        cout << "Request IP in " << r0.ip_in << endl;
        cout << "Request IP out " << r0.ip_out << endl;
        cout << "Processing time " << r0.proc_time << endl;

        // Testing LoadBalancer
        LoadBalancer lb(3);
        const int cycles = 20;

        for (int c  = 1; c <= cycles; c++){
                cout << "Cycle " << c << endl;

                // Generate a new req every 3 cycles
                if(c%3 == 1){
                        Request r;
                        lb.distribute(r);
                }
        lb.run_cycle(c);
        cout << "Total pending requests: " << lb.queued_requests() << endl;
        }


        return 0;
}
```


```
ubuntu@ip-172-26-8-51:~/load-balancer$ ./main
Request IP in 54.240.101.61
Request IP out 227.53.82.103
Processing time 42
Cycle 1
Load Balancer sent req to server 1
Total pending requests: 0
Cycle 2
Total pending requests: 0
Cycle 3
ubuntu@ip-172-26-8-51:~/load-balancer$ git add .
ubuntu@ip-172-26-8-51:~/load-balancer$ ./main
Request IP in 36.2.247.233
Request IP out 124.215.128.109
Processing time 56
Cycle 1
Load Balancer sent req to server 1
Total pending requests: 0
Cycle 2
Total pending requests: 0
Cycle 3
Total pending requests: 0
Cycle 4
Load Balancer sent req to server 2
Total pending requests: 0
Cycle 5
Total pending requests: 0
Cycle 6
Total pending requests: 0
Cycle 7
Load Balancer sent req to server 3
Total pending requests: 0
Cycle 8
Total pending requests: 0
Cycle 9
Total pending requests: 0
Cycle 10
Load Balancer sent req to server 1
Total pending requests: 1
```

```
Cycle 11
Total pending requests: 1
Cycle 12
Total pending requests: 1
Cycle 13
Load Balancer sent req to server 2
Total pending requests: 2
Cycle 14
Total pending requests: 2
Cycle 15
Total pending requests: 2
Cycle 16
Load Balancer sent req to server 3
Total pending requests: 2
Cycle 17
Total pending requests: 2
Cycle 18
Total pending requests: 2
Cycle 19
Load Balancer sent req to server 1
Total pending requests: 3
Cycle 20
Total pending requests: 3
```
Testing LoadBalancer and utilizing the Makefile.

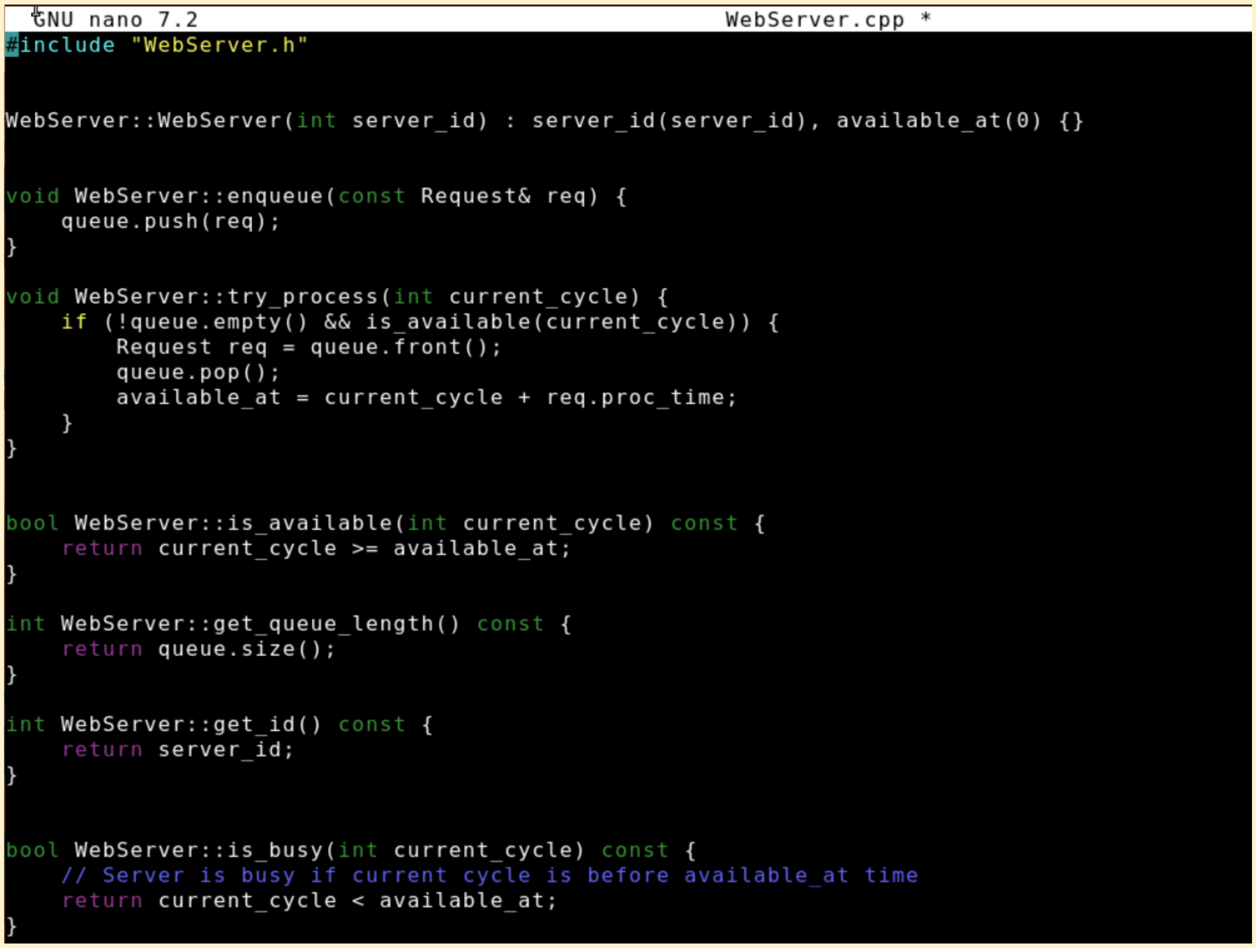


```
GNU nano 7.2                                      main.cpp
using namespace std;

int main() {

    srand(time(0));
    int num_servers;
    int cycles;

    cout << "Enter number of web servers: ";
    cin >> num_servers;

    cout << "Enter number of clock cycles to run: ";
    cin >> cycles;

    LoadBalancer lb(num_servers);

    // Fill initial queue: servers * 100 requests
    for (int i = 0; i < num_servers * 100; ++i) {
        Request r;
        lb.distribute(r); // preload request queue
    }

    for (int c = 1; c <= cycles; ++c) {
        cout << "Cycle " << c << endl;

        // Add random requests (simulate traffic)
        if (rand() % 5 == 0) {  // ~20% chance
            Request r;
            lb.distribute(r);
        }

        lb.run_cycle(c);
        cout << "Total pending requests: " << lb.queued_requests() << endl;
    }

    return 0;
}
```

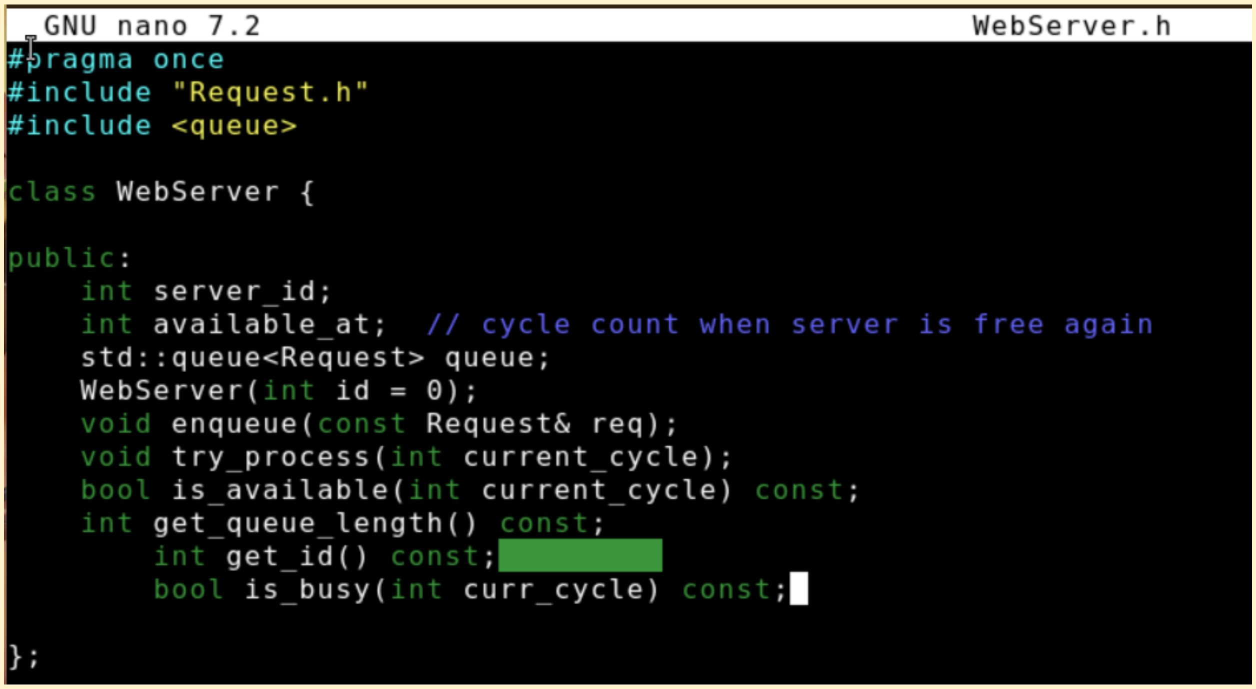
Update main.cpp to better configure for user input.



```
void LoadBalancer::run_cycle(int curr_cycle) {
    for (int i = 0; i < servers.size(); i++) {
        servers[i].try_process(curr_cycle);
    }

        // Find total queued requests across servers
        int total_reqs = queued_requests();
        int overload_thresh = servers.size() * 50;
        int underload_thresh = servers.size() * 10;

        // If overloaded - add server
        if (total_reqs > overload_thresh){
                std::cout << "Overlaoded, adding new server." << std::endl;
                int new_server_id = servers.size() + 1;
                servers.emplace_back(new_server_id);
        }

        // If underloaded - remove a server (as long as more than 1 server exists)
        else if (servers.size() > 1 && total_reqs < underload_thresh){
                WebServer& last_server = servers.back();

                // Check if queue is empty and not busy
                if (last_server.get_queue_length() == 0 && !last_server.is_busy(curr_cycle)){
                        std::cout << "Underloaded, removing server "<< last_server.get_id() << std::endl;
                        servers.pop_back();
                }
        }

}
```

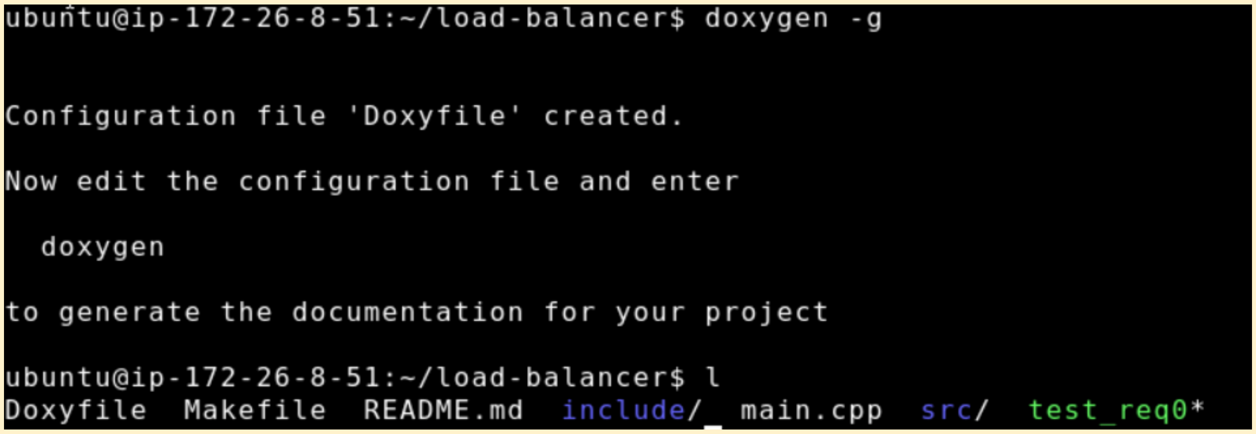


```
  GNU nano 7.2                                    WebServer.cpp *
#include "WebServer.h"


WebServer::WebServer(int server_id) : server_id(server_id), available_at(0) {}


void WebServer::enqueue(const Request& req) {
    queue.push(req);
}

void WebServer::try_process(int current_cycle) {
    if (!queue.empty() && is_available(current_cycle)) {
        Request req = queue.front();
        queue.pop();
        available_at = current_cycle + req.proc_time;
    }
}


bool WebServer::is_available(int current_cycle) const {
    return current_cycle >= available_at;
}

int WebServer::get_queue_length() const {
    return queue.size();
}

int WebServer::get_id() const {
    return server_id;
}


bool WebServer::is_busy(int current_cycle) const {
    // Server is busy if current cycle is before available_at time
    return current_cycle < available_at;
}
```

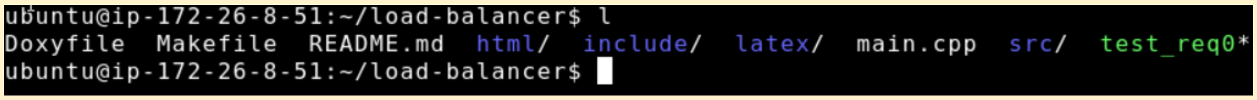


```
 GNU nano 7.2                                              WebServer.h
#pragma once
#include "Request.h"
#include <queue>

class WebServer {

public:
    int server_id;
    int available_at;  // cycle count when server is free again
    std::queue<Request> queue;
    WebServer(int id = 0);
    void enqueue(const Request& req);
    void try_process(int current_cycle);
    bool is_available(int current_cycle) const;
    int get_queue_length() const;
        int get_id() const;
        bool is_busy(int curr_cycle) const;

};
```

Update LoadBalancer and WebServer classes to better handle dynamic allocation and deallocation.



```
ubuntu@ip-172-26-8-51:~/load-balancer$ doxygen -g


Configuration file 'Doxyfile' created.

Now edit the configuration file and enter

  doxygen

to generate the documentation for your project

ubuntu@ip-172-26-8-51:~/load-balancer$ l
Doxyfile  Makefile  README.md  include/  main.cpp  src/  test_req0*
```



```
ubuntu@ip-172-26-8-51:~/load-balancer$ l
Doxyfile  Makefile  README.md  html/  include/  latex/  main.cpp  src/  test_req0*
ubuntu@ip-172-26-8-51:~/load-balancer$ 
```

I then used ChatGPT to add the Doxygen comments for all my header, source, and main files. Then created a Doxyfile for the project code.

Executing Code

~ make
~./main
    ~ Enter desired servers and then clock cycles
~ make clean