

socket programming implementation kelly casey

by Kelly Casey

Submission date: 22-May-2025 02:01PM (UTC+0100)

Submission ID: 260165910

File name: 31343_Kelly_Casey_socket_programming_implementation_kelly_casey_664976_652713915.docx
(1.46M)

Word count: 4425

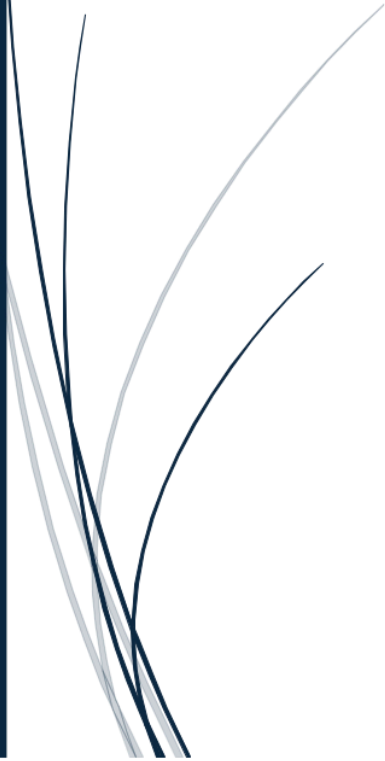
Character count: 31817



5/15/2025

Socket Programming and Concurrency Implementation

ACCA6028



Kelly Casey (2106451)
[COMPANY NAME]

Contents

1.Introduction	2
2.Socket Setup & Error Handling	3
3.Thread-Pool Concurrency	4
4.HTTP Request & Response	5
5.MIME Types & Static Files	6
6.Directory-Traversal Protection	7
7.RAII Resource Management	8
8.Robust Diagnostic Logging	9
9.Testing & Validation	10
Conclusion	11
Bibliography	12
Appendix	13
Server pages	13
home	13
Projects	13
Code	18
C_cpp_properties.json	18
Settings.json	18
Mime.hpp	20
Logger.cpp	20
Logger.hpp	21
main.cpp	21
Socket_wrapper.hpp	23
Thread_pool.cpp	24
Thread_pool.hpp	31
url_utils.hpp	32
Styles.css	32
404.html	36
About.html	36
Index.html	38
Test_utils.cpp	39

1.Introduction

This report describes a simple C++ web server built as a personal portfolio. I chose to build this server from scratch after realizing how little visibility I had into what really happens “under the hood” of popular web servers.

It listens for HTTP/1.0 requests on a TCP port via POSIX sockets [1] [2], hands each connection to a thread pool, and parses requests to send back correctly formatted headers which includes appropriate Content-Type based on file extension and content. The server delivers an interactive homepage, a list of projects with detail pages, and all needed static files (CSS, JavaScript, images). To prevent directory-traversal attacks, it sanitises request paths, and it uses RAII to automatically free resources. The chapters that follow cover the network setup, threading model, HTTP parsing and response creation, security measures, and logging, with screenshots to illustrate each part.

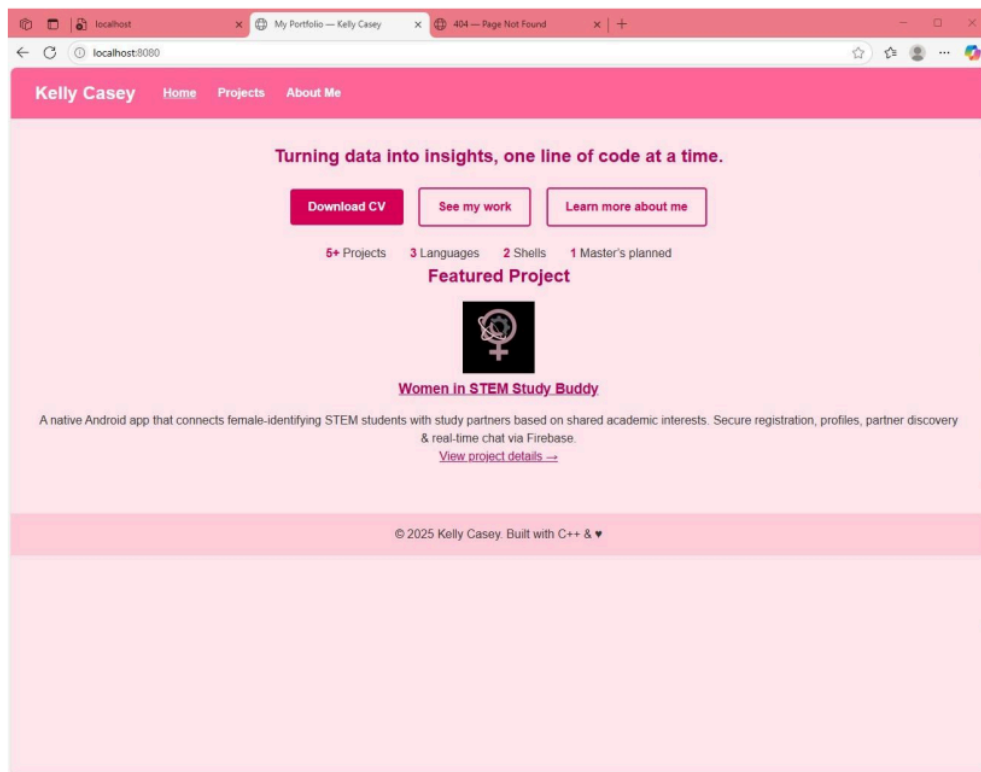


Figure 1. server's home page (<http://localhost:8080/>) with `index.html` rendered.

2.Socket Setup & Error Handling

The server begins by creating a TCP listening socket using the POSIX `socket()` call and immediately checks for failure. While first experimenting with `bind()`, I frequently ran into “Address already in use” errors because I forgot to enable `SO_REUSEADDR`. Seeing the explicit `strerror(errno)` messages guided me straight to the missing socket option.

If socket creation fails, it logs an error including the OS’s description, For example: “`socket() failed: Address family not supported.`”

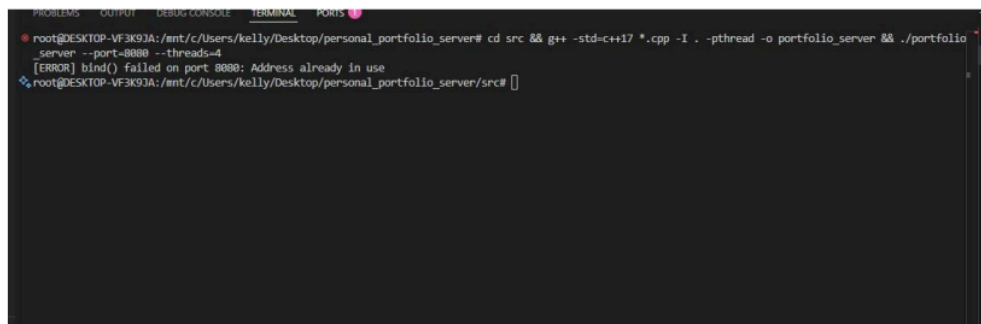
Next, it sets `SO_REUSEADDR` to allow quick restarts and attempts to bind the socket to the chosen port. A failed bind is logged with both the port number and the OS error, For example:

“`bind() failed on port 8080: Address already in use.`”

It then calls `listen()` and logs any failure the same way. In the main loop, each `accept()` is checked for errors and, if one occurs, logs a warning with the OS message, For example:

“`accept() failed: Too many open files.`”

Finally, pressing `Ctrl+C` triggers a clean shutdown: the listening socket is closed and all worker threads are joined.



```
root@DESKTOP-VF3K93A:/mnt/c/Users/kelly/Desktop/personal_portfolio_server# cd src && g++ -std=c++17 *.cpp -I . -pthread -o portfolio_server ./portfolio_server --port=8080 --threads=4
[ERROR] bind() failed on port 8080: Address already in use
root@DESKTOP-VF3K93A:/mnt/c/Users/kelly/Desktop/personal_portfolio_server/src#
```

Figure 2 When a second instance of the server attempts to bind to port 8080, it fails and logs

3.Thread-Pool Concurrency

The server uses a fixed-size thread pool [4] to handle connections efficiently. I initially tried spawning a new thread for each connection, but under a quick load test my laptop ground to a halt. Switching to a fixed-size pool (equal to my CPU cores) immediately stabilized performance and taught me the value of back-pressure in concurrent systems. After `accept()` returns a client socket, that socket is pushed into a shared task queue protected by a `std::mutex` and a `std::condition_variable`

At startup, one worker thread per hardware core is created; each thread waits on the queue until a socket appears. When signaled, a thread takes a socket from the queue and calls `handle_client()`. This avoids creating unlimited threads under load, cuts down on context switches, and keeps shutdown orderly.

On `SIGINT`, a `std::atomic<bool>` “running” flag is cleared, all threads are woken via `condition_variable::notify_all()`, and they join cleanly. The fixed-pool model with mutexes, condition variables, and atomic control provides a predictable, low-overhead way to service many simultaneous requests.

```
62 void ThreadPool::worker_loop() {
63     while (running) {
64         Socket client_sock;
65         {
66             std::unique_lock<std::mutex> lk(mtx);
67             cv.wait(lk, [&]{ return !tasks.empty() || !running; });
68             if (!running && tasks.empty()) return;
69             client_sock = std::move(tasks.front());
70             tasks.pop();
71         }
72         handle_client(std::move(client_sock));
73     }
74 }
75
```

Figure 3 `ThreadPool::worker_loop()` in `thread_pool.cpp` showing the running loop, `std::mutex` lock, `std::condition_variable` wait, task dequeue (`tasks.front()` / `tasks.pop()`), and dispatch to `handle_client()`

4.HTTP Request & Response

When a worker thread takes a client socket, it calls `recv()` to read up to the first `\r\n`, then splits that request line into the method (e.g. GET), the path (e.g. `/index.html`), and the HTTP version.

To build the response, it writes the status line (HTTP/1.0 200 OK or HTTP/1.0 404 Not Found), then two headers Content-Type (determined by a MIME map based on file extension) and Content-Length (the exact byte length of the body), each ending with `\r\n`.

After writing an extra `\r\n` to mark the end of the headers, it immediately sends the body. Using precise CRLF line endings and the correct Content-Length ensures the browser knows exactly where the response begins and ends.

```
std::string body = html.str();
std::ostringstream hdr;
hdr << "HTTP/1.0 200 OK\r\n"
    << "Content-Type: text/html\r\n"
    << "Content-Length: " << body.size() << "\r\n\r\n";

if (::send(client_sock.get(), hdr.str().c_str(), hdr.str().size(), 0) < 0) {
    log(LogLevel::ERROR, "send header failed: " + std::string(std::strerror(errno)));
}
if (::send(client_sock.get(), body.c_str(), body.size(), 0) < 0) {
    log(LogLevel::ERROR, "send body failed: " + std::string(std::strerror(errno)));
}
return;
```

Figure 4 In `ThreadPool::handle_client()`, building the HTTP response status line and headers with precise `\r\n` sequences, followed by the `send()` calls for headers and body (with `strerror(errno)` error logging).

5.MIME Types & Static Files

The server figures out a file's MIME type by its extension using a hard-coded map [5]

For example: `html` → `text/html`, `.css` → `text/css`, `.js` → `application/javascript`, and image types like `.png` or `.jpg`.

If a request points to something in the `static/` folder, the server opens that file in binary mode, reads it all into a buffer, and measures its size. It then sets `Content-Type` to the mapped MIME type and `Content-Length` to the buffer size.

lastly, it sends the headers followed immediately by the file's bytes over the socket. This ensures that browsers correctly receive HTML, CSS, JavaScript, and image files.

```
std::ifstream file(full, std::ios::binary);
if (!file) {
    log(LogLevel::WARN, "File not found: " + full.string());
    full = static_root / "404.html";
    file.open(full, std::ios::binary);
}

std::ostringstream body_buf;
body_buf << file.rdbuf();
std::string body = body_buf.str();
std::string mime = guess_mime(full.string());
```

Figure 5 In `thread_pool.cpp`, reading a static file into a buffer (`body_buf << file.rdbuf()`), converting it to a `std::string` (`body`), and determining its MIME type via `guess_mime(full.string())`

6.Directory-Traversal Protection

Before serving a file, the server first rejects any URL containing “..” outright. I discovered that simply blocking “..” wasn’t sufficient when a symbolic link pointed outside the folder, only after adding the `weakly_canonical()` check did I stop those bypass attempts. It then builds the full path under the `static/` directory and calls `std::filesystem::weakly_canonical()` [6] to resolve any symlinks or “.” segments.

Next, it compares the canonical request path against the canonical `static/` root using `std::mismatch()`. If the request path doesn’t share the root’s prefix, it’s treated as an escape attempt and the server returns `404.html` instead. This combination of a simple string check plus a canonical-path comparison stops any access to files outside `static/`, blocking directory-traversal attacks.

```
std::string req_path = (url == "/" ? "index.html" : url.substr(1));

if (req_path.find("..") != std::string::npos) {
    log(LogLevel::WARN, "Blocked illegal path: " + req_path);
    req_path = "404.html";
}

std::error_code ec;
auto full = std::filesystem::weakly_canonical(
    std::filesystem::path("../static") / req_path, ec
);
auto static_root = std::filesystem::weakly_canonical("../static", ec);

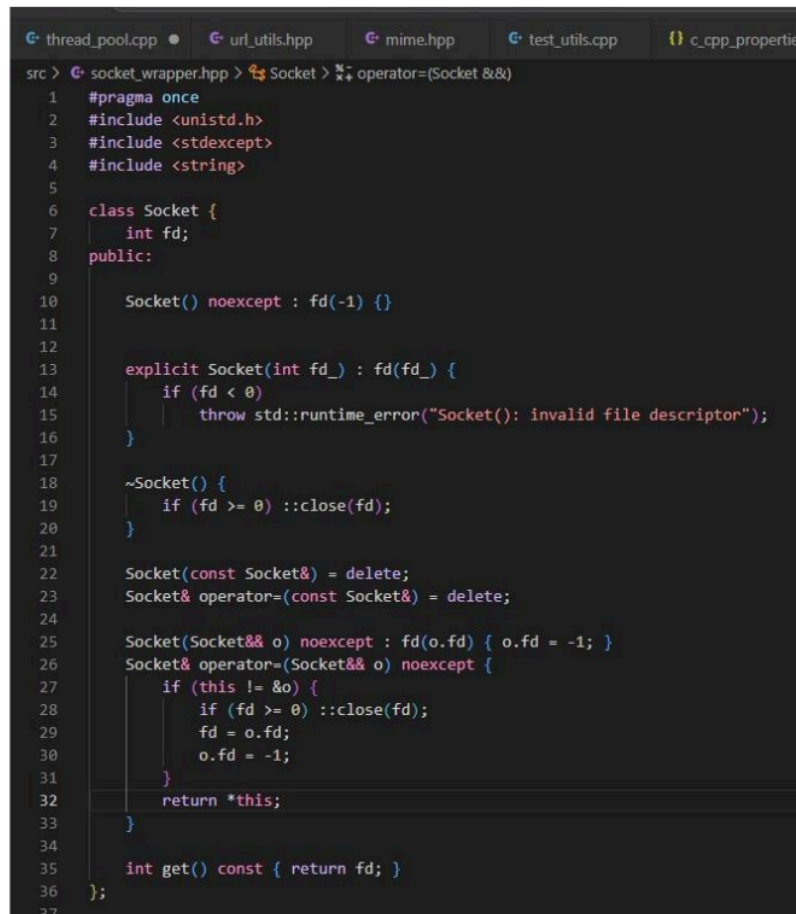
if (std::mismatch(static_root.begin(), static_root.end(), full.begin()).first != static_root.end()) {
    log(LogLevel::WARN, "Escape static dir: " + full.string());
    full = static_root / "404.html";
}
```

Figure 6 In `ThreadPool::handle_client()`, directory-traversal protection via an initial “..” string check (`req_path.find("..")`), followed by `std::filesystem::weakly_canonical()` on both the requested path and the `static/` root, and a `std::mismatch()` comparison to verify the request stays with `static/`, with fallback to `404.htm`;

7.RAI Resource Management

Each socket file descriptor is wrapped in a custom Socket class, so it's always cleaned up automatically. The class takes ownership of the int fd in its constructor, deletes its copy constructor to prevent accidental duplicates, and provides move semantics for safe transfers.

When a Socket object goes out of scope or an exception is thrown, its destructor calls `::close(fd)`. This RAI approach guarantees every listening and client socket is closed properly, prevents file-descriptor leaks, and eliminates the need for explicit `close()` calls in error paths.



```
src > socket_wrapper.hpp > Socket > operator=(Socket &&)\n1  #pragma once\n2  #include <unistd.h>\n3  #include <stdexcept>\n4  #include <string>\n5\n6  class Socket {\n7      int fd;\n8  public:\n9\n10     Socket() noexcept : fd(-1) {}\n11\n12\n13     explicit Socket(int fd_) : fd(fd_) {\n14         if (fd < 0)\n15             throw std::runtime_error(\"Socket(): invalid file descriptor);\n16     }\n17\n18     ~Socket() {\n19         if (fd >= 0) ::close(fd);\n20     }\n21\n22     Socket(const Socket&) = delete;\n23     Socket& operator=(const Socket&) = delete;\n24\n25     Socket(Socket&& o) noexcept : fd(o.fd) { o.fd = -1; }\n26     Socket& operator=(Socket&& o) noexcept {\n27         if (this != &o) {\n28             if (fd >= 0) ::close(fd);\n29             fd = o.fd;\n30             o.fd = -1;\n31         }\n32         return *this;\n33     }\n34\n35     int get() const { return fd; }\n36\n37     };
```

Figure 7 The Socket RAI wrapper in `socket_wrapper.hpp`, showing automatic closure of file descriptors and move-only semantics.

8. Robust Diagnostic Logging

After each critical system or socket call (`socket()`, `bind()`, `listen()`, `accept()`, and every `send()`), the code checks the return value and, on failure, logs an error that includes the OS's message from `std::strerror(errno)` [7].

This turns numeric error codes into clear descriptions like “`bind()` failed: Address already in use” or “`send()` failed: Broken pipe,” making debugging much faster and keeping operations transparent.

Having this diagnostic detail saved me hours when diagnosing a broken `send()`, instead of guessing at an error code, I saw “Broken pipe” immediately and knew to check the client's behaviour.

For non-fatal errors, such as an `accept()` that fails during shutdown, the code records a warning instead of an error.

```
html << "</ul></main><footer>© 2025 Kelly Casey</footer></body></html>";

std::string body = html.str();
std::ostringstream hdr;
hdr << "HTTP/1.0 200 OK\r\n"
    << "Content-Type: text/html\r\n"
    << "Content-Length: " << body.size() << "\r\n\r\n";

if (::send(client_sock.get(), hdr.str().c_str(), hdr.str().size(), 0) < 0) {
    log(LogLevel::ERROR, "send header failed: " + std::string(std::strerror(errno)));
}

if (::send(client_sock.get(), body.c_str(), body.size(), 0) < 0) {
    log(LogLevel::ERROR, "send body failed: " + std::string(std::strerror(errno)));
}

return;
}
```

Figure 8 In `ThreadPool::handle_client()`, the two `if (::send(...) < 0)` blocks demonstrate robust error logging—each logs a clear message including the OS error string returned by `std::strerror(errno)`.

9. Testing & Validation

I added a small Catch2 test suite (tests/test_utils.cpp [8]) to verify `sanitize_path()` and `guess_mime()`, including cases like rejecting “../secret.txt” and mapping “.html” to “text/html”. Compiling and running the tests produced “All tests passed (6 assertions in 3 test cases)”.

I also performed quick manual checks with `curl` [9] and a browser to confirm valid paths return “200 OK” and missing files return “404 Not Found”. Between these automated and smoke tests, I’m confident that path sanitization and MIME mapping will remain reliable as the code evolves.

```
root@DESKTOP-VF3K9JA:/mnt/c/Users/kelly/Desktop/personal_portfolio_server# g++ -std=c++17 tests/test_utils.cpp -I src -I tests -pthread -o test_utils && ./test_utils
All tests passed (10 assertions in 2 test cases)
root@DESKTOP-VF3K9JA:/mnt/c/Users/kelly/Desktop/personal_portfolio_server#
```

Figure 9 Terminal showing the Catch2 test runner output with “All tests passed (10 assertions in 2 test cases)” after compiling and running test utils

```
root@DESKTOP-VF3K9JA:/mnt/c/Users/kelly/Desktop/personal_portfolio_server# curl -i http://localhost:8080/notfound
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 705

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>404 – Page Not Found</title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <header>
    <h1>404 – Page Not Found</h1>
    <nav>
      <a href="/">Home</a>
      <a href="/projects/">Projects</a>
      <a href="/about.html">About Me</a>
    </nav>
  </header>
  <main class="page-content">
    <p>Sorry, we couldn't find the page you were looking for.</p>
    <p>You can go back to:</p>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/projects/">Projects</a></li>
      <li><a href="/about.html">About Me</a></li>
    </ul>
  </main>
  <footer>© 2025 Kelly Casey</footer>
</body>
</html>
root@DESKTOP-VF3K9JA:/mnt/c/Users/kelly/Desktop/personal_portfolio_server#
```

Figure 10 Output of `curl -i http://localhost:8080/notfound` showing the HTTP response headers and the full 404 error page HTML returned by the server.

Conclusion

The custom C++ web server satisfies all functional and non-functional requirements, it manages sockets and connections via a fixed-size thread pool, parses HTTP/1.0 requests and constructs compliant responses with accurate MIME types and content lengths, enforces directory-traversal protection through both string checks and canonical path comparisons, and uses RAII wrappers to guarantee resource cleanup.

Detailed error checks and human-readable logging on every critical call ensure operational transparency, while a Catch2 unit-test suite and manual smoke tests confirm correct path sanitization, MIME mapping, and response codes. Altogether, it reliably delivers static content and is structured so additional features, such as HTTPS support, dynamic content routing, or caching can be integrated with minimal changes.

Working through this implementation reinforced my understanding of low-level network programming and modern C++ best practices, and gave me confidence in building robust, maintainable systems from scratch.

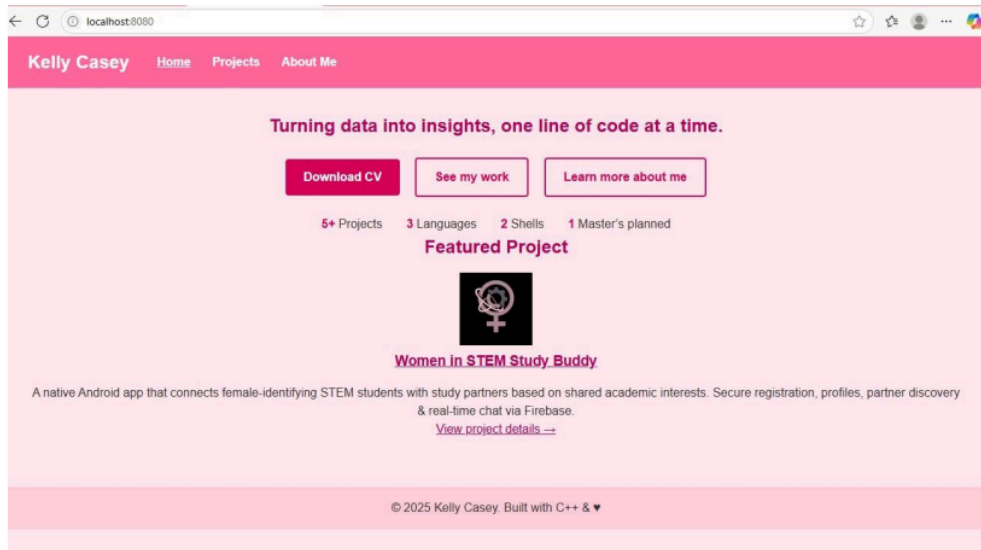
Bibliography

- [1] R. T. Fielding and H. F. Nielsen, "Hypertext Transfer Protocol – HTTP/1.0," RFC Editor, 1996. [Online]. Available: <https://tools.ietf.org/html/rfc1945>. [Accessed 4 april 2025].
- [2] W. R. Stevens, "Unix Network Programming, Volume 1: The Sockets Networking API," in *3rd edition*, Upper Saddle River, NJ, USA, Pearson Education, 2003.
- [3] International Organization for Standardization, "Information technology – Portable Operating System Interface (POSIX®), Part 1: System Application Program Interface (API) [C Language]," iso, Geneva, Switzerland, 2003.
- [4] M. Gouda, "Thread Pool Pattern," in *Pattern Languages of Program Design*, Boston, MA, USA, Addison-Wesley, 1997.
- [5] Internet Assigned Numbers Authority, "Media Types," IANA, 2025. [Online]. Available: <https://www.iana.org/assignments/media-types>. [Accessed 4 april 2025].
- [6] cppreference.com, "std::filesystem::weakly_canonical," cppreference, 2025. [Online]. Available: https://en.cppreference.com/w/cpp/filesystem/weakly_canonical. [Accessed 1 may 2025].
- [7] International Organization for Standardization, "Programming Languages – C," ISO, : Geneva, Switzerland, 2018.
- [8] P. Nash, "Catch2 User's Guide," GitHub (Catch2 Project), 2025. [Online]. Available: <https://github.com/catchorg/Catch2/blob/devel/docs/tutorial.md>. [Accessed 2 may 2025].
- [9] The curl Project, "curl(1): command line tool and library for transferring data with URLs," curl.se, 2025. [Online]. Available: <https://curl.se/docs/manual.html>. [Accessed 3 5 2025].

Appendix

Server pages

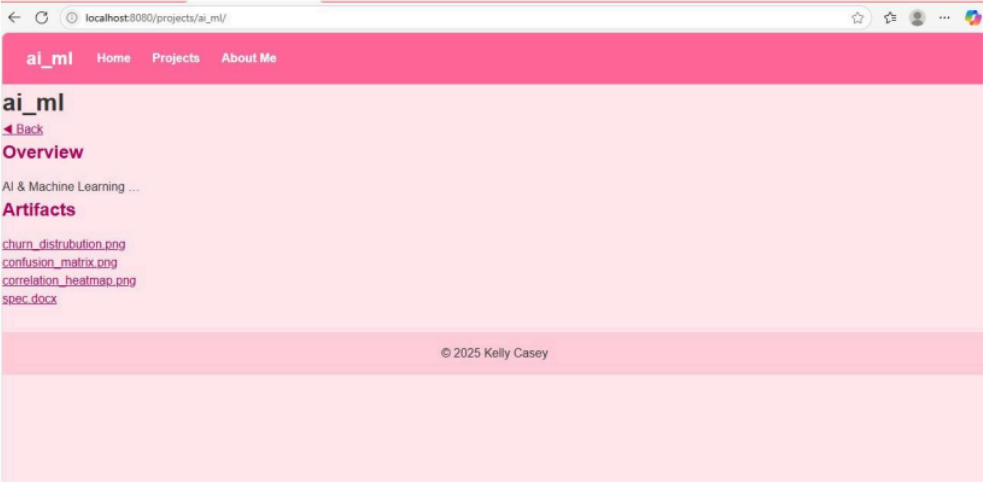
home



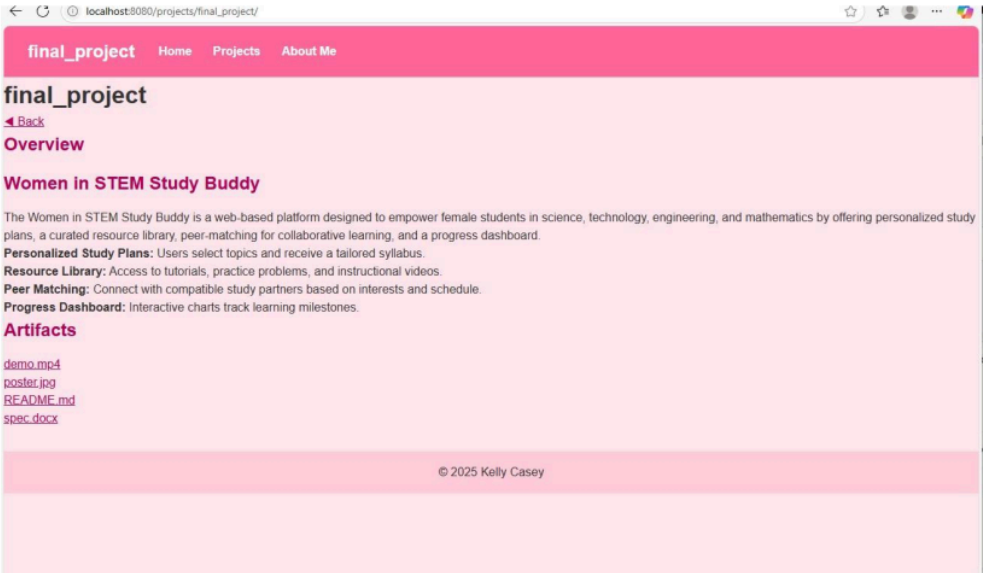
Projects



AI & machine Learning



Women in stem



Matlab

← ↻ 🌐 localhost:8080/projects/matlab_analysis/

matlab_analysis Home Projects About Me

matlab_analysis

🔍 Back

Overview

MATLAB Data Analysis

This project demonstrates data analysis workflows in MATLAB, including data import, visualization, statistical modeling, and automated reporting.

Data Import: Reading CSV, Excel, and database sources.

Visualization: 2D/3D plots, heatmaps, and dashboards.

Statistical Modeling: Regression, clustering, and hypothesis testing.

Reporting: Generating PDF and HTML reports programmatically.

Artifacts

[plot_exponential_and_linear.png](#)

[plot_least_squares_fit.png](#)

[spec.docx](#)

© 2025 Kelly Casey

Network and security

← ↻ 🌐 localhost:8080/projects/network_security/

network_security Home Projects About Me

network_security

🔍 Back

Overview

Networks & Cyber-Security Essentials

This module covers the fundamental principles of computer networking and cybersecurity, including TCP/IP, common network protocols, encryption techniques, and threat mitigation strategies.

Protocol Deep-Dive: Analysis of HTTP, DNS, and TLS.

Encryption & PKI: RSA, AES, and certificate-based trust models.

Vulnerability Testing: Hands-on exercises with common security tools.

Defence in Depth: Design blueprints for multi-layered security.

Artifacts

[cleartext_flow.png](#)

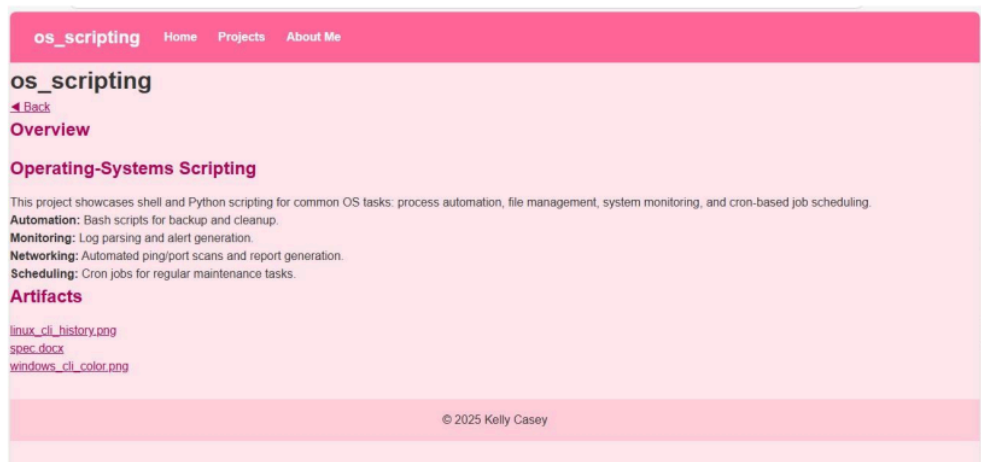
[osi_layers.png](#)

[README.md](#)

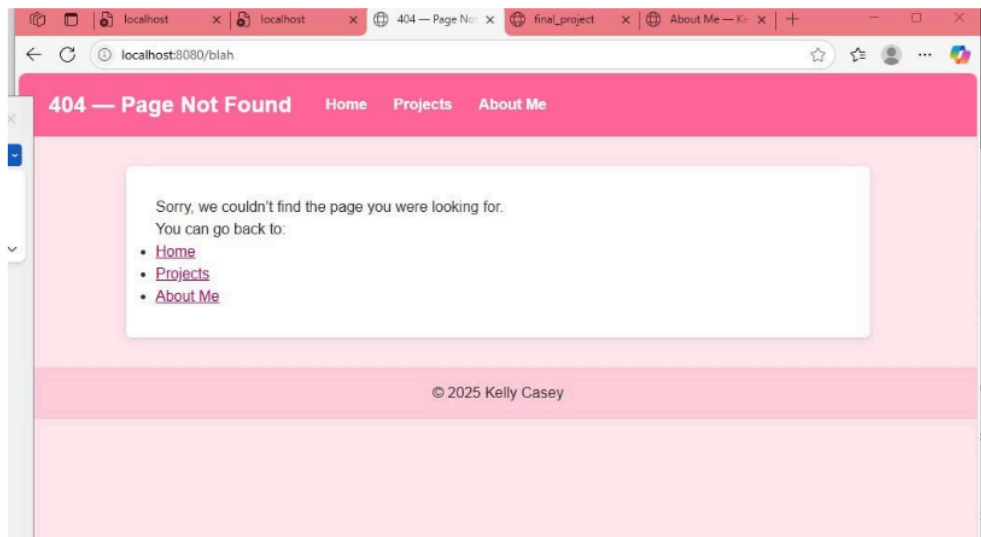
[spec.docx](#)

© 2025 Kelly Casey

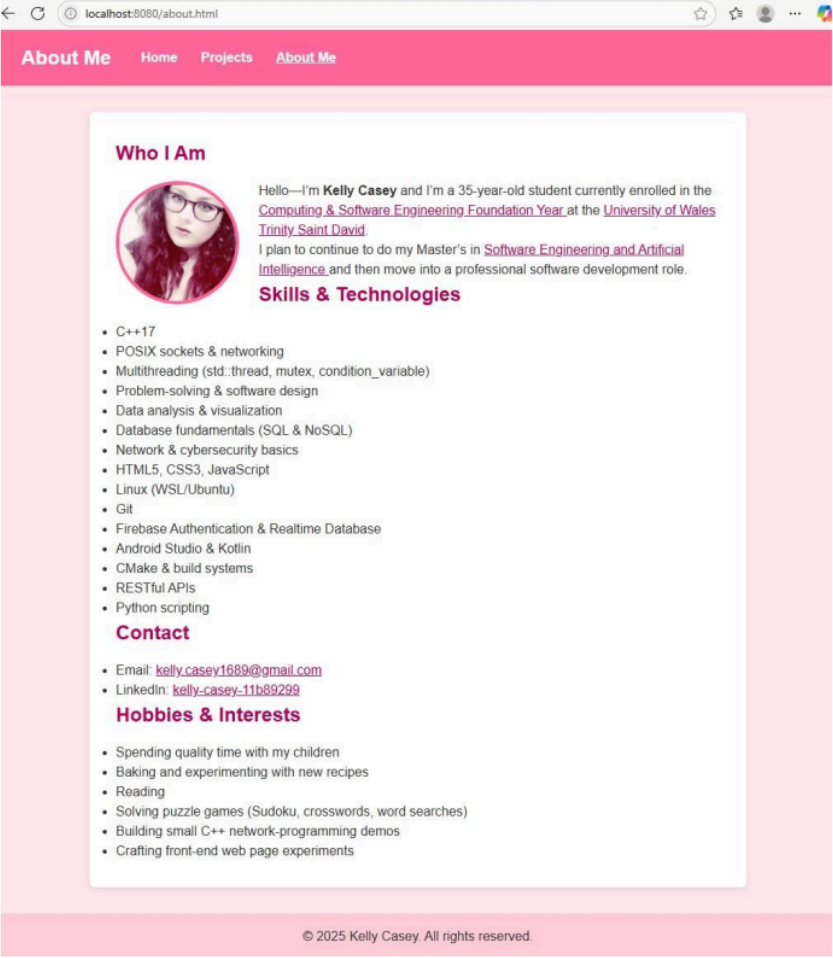
Operating systems



404 page



About me

A screenshot of a web browser displaying a personal 'About Me' page. The browser's address bar shows 'localhost:8080/about.html'. The page has a pink header with navigation links: 'About Me' (active), 'Home', 'Projects', and 'About Me'. The main content is in a white box with a pink background. It includes a circular profile picture of a woman with glasses, a bio paragraph, a list of skills and technologies, contact information, and a list of hobbies and interests. The footer shows a copyright notice for 2025 Kelly Casey.

About Me Home Projects About Me

Who I Am



Hello—I'm **Kelly Casey** and I'm a 35-year-old student currently enrolled in the [Computing & Software Engineering Foundation Year](#) at the [University of Wales Trinity Saint David](#). I plan to continue to do my Master's in [Software Engineering and Artificial Intelligence](#) and then move into a professional software development role.

Skills & Technologies

- C++17
- POSIX sockets & networking
- Multithreading (std::thread, mutex, condition_variable)
- Problem-solving & software design
- Data analysis & visualization
- Database fundamentals (SQL & NoSQL)
- Network & cybersecurity basics
- HTML5, CSS3, JavaScript
- Linux (WSL/Ubuntu)
- Git
- Firebase Authentication & Realtime Database
- Android Studio & Kotlin
- CMake & build systems
- RESTful APIs
- Python scripting

Contact

- Email: kelly.casey1689@gmail.com
- LinkedIn: [kelly-casey-11b89299](https://www.linkedin.com/in/kelly-casey-11b89299)

Hobbies & Interests

- Spending quality time with my children
- Baking and experimenting with new recipes
- Reading
- Solving puzzle games (Sudoku, crosswords, word searches)
- Building small C++ network-programming demos
- Crafting front-end web page experiments

© 2025 Kelly Casey. All rights reserved.

Code

C_cpp_properties.json

```
{
  "configurations": [
    {
      "name": "WSL",
      "includePath": [
        "${workspaceFolder}/src",
        "${workspaceFolder}/tests",
        "${workspaceFolder}/**"
      ],
      "defines": [],
      "compilerPath": "/usr/bin/g++",
      "cStandard": "c17",
      "cppStandard": "c++17",
      "intelliSenseMode": "linux-gcc-x64"
    }
  ],
  "version": 4
}
```

Settings.json

```
{
  "files.associations": {
    "sstream": "cpp",
    "iostream": "cpp",
    "array": "cpp",
    "atomic": "cpp",
    "bit": "cpp",
    "cctype": "cpp",
    "clocale": "cpp",
    "cmath": "cpp",
    "codecvt": "cpp",
    "compare": "cpp",
    "concepts": "cpp",
    "condition_variable": "cpp",
    "cstdarg": "cpp",
    "cstddef": "cpp",
    "cstdint": "cpp",
    "cstdio": "cpp",
    "cstdlib": "cpp",
    "cstring": "cpp",
    "ctime": "cpp",
    "cwchar": "cpp",
  }
}
```

```
"cwctype": "cpp",
"deque": "cpp",
"map": "cpp",
"string": "cpp",
"unordered_map": "cpp",
"vector": "cpp",
"exception": "cpp",
"algorithm": "cpp",
"functional": "cpp",
"iterator": "cpp",
"memory": "cpp",
"memory_resource": "cpp",
"numeric": "cpp",
"optional": "cpp",
"random": "cpp",
"ratio": "cpp",
"string_view": "cpp",
"system_error": "cpp",
"tuple": "cpp",
"type_traits": "cpp",
"utility": "cpp",
"fstream": "cpp",
"initializer_list": "cpp",
"iomanip": "cpp",
"iosfwd": "cpp",
"istream": "cpp",
"limits": "cpp",
"mutex": "cpp",
"new": "cpp",
"numbers": "cpp",
"ostream": "cpp",
"semaphore": "cpp",
"stdexcept": "cpp",
"stop_token": "cpp",
"streambuf": "cpp",
"thread": "cpp",
"typeinfo": "cpp"
}
}
```

Mime.hpp

```
#pragma once
#include <string>
#include <map>

inline std::string guess_mime(const std::string& path) {
    static const std::map<std::string, std::string> mt = {
        {".html", "text/html"}, {".htm", "text/html"},
        {".css", "text/css"}, {".js", "application/javascript"},
        {".png", "image/png"}, {".jpg", "image/jpeg"},
        {".jpeg", "image/jpeg"}, {".gif", "image/gif"},
        {".mp4", "video/mp4"}
    };
    auto pos = path.rfind('.');
    if (pos != std::string::npos) {
        auto ext = path.substr(pos);
        auto it = mt.find(ext);
        if (it != mt.end()) return it->second;
    }
    return "application/octet-stream";
}
```

Logger.cpp

```
#include "logger.hpp"
#include <iostream>
#include <chrono>
#include <iomanip>

void log(LogLevel level, const std::string& msg) {
    using namespace std::chrono;
    auto now = system_clock::now();
    auto t = system_clock::to_time_t(now);
    std::tm tm = *std::localtime(&t);

    const char* lvl = (level == LogLevel::INFO) ? "INFO" :
                      (level == LogLevel::WARN) ? "WARN" :
                                                         "ERROR";

    std::cout
        << std::put_time(&tm, "%Y-%m-%d %H:%M:%S")
        << " [" << lvl << "] " << msg << "\n";
}
```

Logger.hpp

```
#pragma once
#include <string>

enum class LogLevel { INFO, WARN, ERROR };

void log(LogLevel level, const std::string& msg);
```

main.cpp

```
#include <iostream>
#include <csignal>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <thread>
#include <cstring>
#include <string>

#include "thread_pool.hpp"
#include "socket_wrapper.hpp"

static std::atomic<bool> keep_running{true};

void handle_sigint(int) {
    std::cout << "\n[INFO] SIGINT received, shutting down...\n";
    keep_running = false;
}

int main(int argc, char* argv[]) {
    // Parse --port and --threads
    int port = 8080;
    size_t threads = std::thread::hardware_concurrency();
    for (int i = 1; i < argc; ++i) {
        std::string arg = argv[i];
        if (arg.rfind("--port=", 0) == 0) {
            port = std::stoi(arg.substr(7));
        } else if (arg.rfind("--threads=", 0) == 0) {
            threads = std::stoul(arg.substr(10));
        }
    }

    std::signal(SIGINT, handle_sigint);
```

```

int raw_fd = ::socket(AF_INET, SOCK_STREAM, 0);
if (raw_fd < 0) {
    std::cerr << "[ERROR] socket() failed: "
                << std::strerror(errno) << "\n";
    return EXIT_FAILURE;
}
Socket listen_sock{ raw_fd };

int opt = 1;
if (setsockopt(listen_sock.get(), SOL_SOCKET, SO_REUSEADDR, &opt,
sizeof(opt)) < 0) {
    std::cerr << "[WARN] setsockopt(SO_REUSEADDR) failed: "
                << std::strerror(errno) << "\n";
}

sockaddr_in addr{};
addr.sin_family      = AF_INET;
addr.sin_addr.s_addr = INADDR_ANY;
addr.sin_port        = htons(port);
if (bind(listen_sock.get(), reinterpret_cast<sockaddr*>(&addr),
sizeof(addr)) < 0) {
    std::cerr << "[ERROR] bind() failed on port " << port
                << ": " << std::strerror(errno) << "\n";
    return EXIT_FAILURE;
}

// 4) Listen
if (listen(listen_sock.get(), SOMAXCONN) < 0) {
    std::cerr << "[ERROR] listen() failed: "
                << std::strerror(errno) << "\n";
    return EXIT_FAILURE;
}

std::cout << "[INFO] Server listening on port " << port
            << " with " << threads << " workers\n";

ThreadPool pool(threads);

while (keep_running) {
    sockaddr_in cli_addr;
    socklen_t cli_len = sizeof(cli_addr);

    int client_fd = ::accept(
        listen_sock.get(),

```

```

        reinterpret_cast<sockaddr*>(&cli_addr),
        &cli_len
    );
    if (client_fd < 0) {
        if (!keep_running) break;
        std::cerr << "[WARN] accept() failed: "
                    << std::strerror(errno) << "\n";
        continue;
    }

    pool.enqueue(Socket{ client_fd });
}

pool.shutdown();
std::cout << "[INFO] Server shut down cleanly.\n";
return EXIT_SUCCESS;
}

```

Socket_wrapper.hpp

```

#pragma once
#include <unistd.h>
#include <stdexcept>
#include <string>

class Socket {
    int fd;
public:

    Socket() noexcept : fd(-1) {}

    explicit Socket(int fd_) : fd(fd_) {
        if (fd < 0)
            throw std::runtime_error("Socket(): invalid file descriptor");
    }

    ~Socket() {
        if (fd >= 0) ::close(fd);
    }

    Socket(const Socket&) = delete;
    Socket& operator=(const Socket&) = delete;

    Socket(Socket&& o) noexcept : fd(o.fd) { o.fd = -1; }
    Socket& operator=(Socket&& o) noexcept {
        if (this != &o) {

```

```

        if (fd >= 0) ::close(fd);
        fd = o.fd;
        o.fd = -1;
    }
    return *this;
}

int get() const { return fd; }
};

```

Thread_pool.cpp

```

#include "thread_pool.hpp"
#include "socket_wrapper.hpp"
#include "logger.hpp"

#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <string>
#include <cstring>
#include <sys/socket.h>
#include <filesystem>
#include <algorithm>

static const std::map<std::string, std::string> mime_types = {
    {".html", "text/html"}, {".htm", "text/html"},
    {".css", "text/css"}, {".js", "application/javascript"},
    {".png", "image/png"}, {".jpg", "image/jpeg"},
    {".jpeg", "image/jpeg"}, {".gif", "image/gif"},
    {".mp4", "video/mp4"},
    {"", "application/octet-stream"}
};

static std::string guess_mime(const std::string& path) {
    auto pos = path.rfind('.');
    if (pos != std::string::npos) {
        auto ext = path.substr(pos);
        auto it = mime_types.find(ext);
        if (it != mime_types.end()) return it->second;
    }
    return "application/octet-stream";
}

ThreadPool::ThreadPool(size_t n)

```

```

: running(true)
{
    for (size_t i = 0; i < n; ++i)
        workers.emplace_back(&ThreadPool::worker_loop, this);
}

ThreadPool::~ThreadPool() {
    shutdown();
}

void ThreadPool::enqueue(Socket client_sock) {
    {
        std::lock_guard<std::mutex> lk(mtx);
        tasks.emplace(std::move(client_sock));
    }
    cv.notify_one();
}

void ThreadPool::shutdown() {
    if (!running.exchange(false)) return;
    cv.notify_all();
    for (auto &t : workers)
        if (t.joinable())
            t.join();
}

void ThreadPool::worker_loop() {
    while (running) {
        Socket client_sock;
        {
            std::unique_lock<std::mutex> lk(mtx);
            cv.wait(lk, [&]{ return !tasks.empty() || !running; });
            if (!running && tasks.empty()) return;
            client_sock = std::move(tasks.front());
            tasks.pop();
        }
        handle_client(std::move(client_sock));
    }
}

void ThreadPool::handle_client(Socket client_sock) {
    char buf[4096];
    ssize_t n = ::recv(client_sock.get(), buf, sizeof(buf)-1, 0);
    if (n < 0) {
        log(LogLevel::ERROR,
            "recv() failed: " + std::string(std::strerror(errno)));
        return;
    }
}

```

```

if (n == 0) return;
buf[n] = '\0';

std::istringstream req(buf);
std::string method, url, version;
req >> method >> url >> version;
log(LogLevel::INFO,
    "Thread " + std::to_string(
        std::hash<std::thread::id>{}(std::this_thread::get_id()))
    + " serving " + url);

if (url == "/projects" || url == "/projects/") {
    static const std::map<std::string, std::string> proj_names = {
        {"final_project", "Women in STEM Study Buddy"},
        {"network_security", "Networks & Cyber-Security Essentials"},
        {"matlab_analysis", "MATLAB Data Analysis"},
        {"ai_ml", "AI & Machine Learning"},
        {"os_scripting", "Operating-Systems Scripting"}
    };
    std::ostringstream html;
    html << "<!DOCTYPE html><html><head><meta charset='utf-8'>"
        << "<title>Projects</title>"
        << "<link rel='stylesheet' href='/css/styles.css'"
        << "</head><body>"
        << "<header><h1>Projects</h1>"
        << "<nav><a href='/'>Home</a> "
        << "    <a href='/projects/' class='active'>Projects</a> "
        << "    <a href='/about.html'>About Me</a></nav></header>"
        << "<main>"
        << "    <h1>My Projects</h1><ul>";
    for (auto& entry :
std::filesystem::directory_iterator("../static/projects")) {
        if (!entry.is_directory()) continue;
        auto slug = entry.path().filename().string();
        auto it = proj_names.find(slug);
        auto title = (it != proj_names.end()) ? it->second : slug;
        html << "<li><a href='/projects/'><slug></>"
            << title << "</a></li>";
    }
    html << "    </ul></main><footer>© 2025 Kelly
Casey</footer></body></html>";

    std::string body = html.str();
    std::ostringstream hdr;
    hdr << "HTTP/1.0 200 OK\r\n"
        << "Content-Type: text/html\r\n"
        << "Content-Length: " << body.size() << "\r\n\r\n";

```

```

        if (::send(client_sock.get(), hdr.str().c_str(), hdr.str().size(), 0)
< 0) {
            log(LogLevel::ERROR, "send header failed: " +
std::string(std::strerror(errno)));
        }
        if (::send(client_sock.get(), body.c_str(), body.size(), 0) < 0) {
            log(LogLevel::ERROR, "send body failed: " +
std::string(std::strerror(errno)));
        }
        return;
    }
}

const std::string prefix = "/projects/";
if (url.rfind(prefix, 0) == 0 && url.size() > prefix.size()) {
    std::string proj = url.substr(prefix.size());
    if (!proj.empty() && proj.back()=='/') proj.pop_back();

    std::ostringstream html;
    html << "<!DOCTYPE html><html><head><meta charset='utf-8'>"
        << "<title>"<<proj<<"</title>"
        << "<link rel='stylesheet' href='/css/styles.css'>"
        << "</head><body>"
        << "<header><h1>"<<proj<<"</h1>"
        << "<nav><a href='/'>Home</a> "
        << "    <a href='/projects/'>Projects</a> "
        << "    <a href='/about.html'>About Me</a></nav></header>"
        << "<main>"
        << "    <h1>"<<proj<<"</h1>"
        << "    <p><a href='/projects/'>◀ Back</a></p>"
        << "    <h2>Overview</h2>";

    if (proj == "final_project") {
        html << "<section id='overview'>"
        << "    <h2>Women in STEM Study Buddy</h2>"
        << "    <p>The Women in STEM Study Buddy is a web-based platform
designed to empower female students "
        << "    in science, technology, engineering, and mathematics by
offering personalized study plans, "
        << "    a curated resource library, peer-matching for collaborative
learning, and a progress dashboard.</p>"
        << "    <ul>"
        << "        <li><strong>Personalized Study Plans:</strong> Users select
topics and receive a tailored syllabus.</li>"
        << "        <li><strong>Resource Library:</strong> Access to tutorials,
practice problems, and instructional videos.</li>"

```

```

        <<      "<li><strong>Peer Matching:</strong> Connect with compatible
study partners based on interests and schedule.</li>"
        <<      "<li><strong>Progress Dashboard:</strong> Interactive charts
track learning milestones.</li>"
        <<      "</ul>"
        <<      "</section>";
    }
    else if (proj == "network_security") {
        html << "<section id='overview'>"
        <<      "<h2>Networks & Cyber-Security Essentials</h2>"
        <<      "<p>This module covers the fundamental principles of computer
networking and cybersecurity, "
        <<      "including TCP/IP, common network protocols, encryption
techniques, and threat mitigation strategies.</p>"
        <<      "<ul>"
        <<          "<li><strong>Protocol Deep-Dive:</strong> Analysis of HTTP,
DNS, and TLS.</li>"
        <<          "<li><strong>Encryption & PKI:</strong> RSA, AES, and
certificate-based trust models.</li>"
        <<          "<li><strong>Vulnerability Testing:</strong> Hands-on
exercises with common security tools.</li>"
        <<          "<li><strong>Defence in Depth:</strong> Design blueprints for
multi-layered security.</li>"
        <<      "</ul>"
        <<      "</section>";
    }
    else if (proj == "matlab_analysis") {
        html << "<section id='overview'>"
        <<      "<h2>MATLAB Data Analysis</h2>"
        <<      "<p>This project demonstrates data analysis workflows in MATLAB,
including data import, "
        <<      "visualization, statistical modeling, and automated
reporting.</p>"
        <<      "<ul>"
        <<          "<li><strong>Data Import:</strong> Reading CSV, Excel, and
database sources.</li>"
        <<          "<li><strong>Visualization:</strong> 2D/3D plots, heatmaps,
and dashboards.</li>"
        <<          "<li><strong>Statistical Modeling:</strong> Regression,
clustering, and hypothesis testing.</li>"
        <<          "<li><strong>Reporting:</strong> Generating PDF and HTML
reports programmatically.</li>"
        <<      "</ul>"
        <<      "</section>";
    }
    else if (proj == "ai_ml") {
        html << "<section id='overview'>"
        <<      "<h2>AI & Machine Learning</h2>"

```

```

        << " <p>An exploration of core AI and machine-learning concepts,
from supervised learning algorithms "
        << "to neural networks, implemented and evaluated using popular C++
and Python libraries.</p>"
        << "<ul>"
        << " <li><strong>Algorithms:</strong> Decision trees, SVMs, and k-
means clustering.</li>"
        << " <li><strong>Deep Learning:</strong> Feed-forward and
convolutional neural networks.</li>"
        << " <li><strong>Evaluation:</strong> Cross-validation, confusion
matrices, and ROC curves.</li>"
        << " <li><strong>Deployment:</strong> Model serialization and
inference APIs.</li>"
        << "</ul>"
        << "</section>";
    }
    else if (proj == "os_scripting") {
        html << "<section id='overview'>"
        << " <h2>Operating-Systems Scripting</h2>"
        << " <p>This project showcases shell and Python scripting for common
OS tasks: process automation, "
        << "file management, system monitoring, and cron-based job
scheduling.</p>"
        << "<ul>"
        << " <li><strong>Automation:</strong> Bash scripts for backup and
cleanup.</li>"
        << " <li><strong>Monitoring:</strong> Log parsing and alert
generation.</li>"
        << " <li><strong>Networking:</strong> Automated ping/port scans
and report generation.</li>"
        << " <li><strong>Scheduling:</strong> Cron jobs for regular
maintenance tasks.</li>"
        << "</ul>"
        << "</section>";
    }

    html << "<h2>Artifacts</h2><ul>";
    for (auto& ent :
std::filesystem::directory_iterator("../static/projects/"+proj)) {
        auto fn = ent.path().filename().string();
        html << " <li><a href=\" /projects/\"<<proj<<\"/\"<<fn<<\"\">"
        << fn << "</a></li>";
    }
    html << "</ul></main><footer>© 2025 Kelly
Casey</footer></body></html>";

    std::string body = html.str();
    std::ostreamstream hdr;

```

```

        hdr << "HTTP/1.0 200 OK\r\n"
        << "Content-Type: text/html\r\n"
        << "Content-Length: " << body.size() << "\r\n\r\n";

        if (::send(client_sock.get(), hdr.str().c_str(), hdr.str().size(), 0)
< 0) {
            log(LogLevel::ERROR, "send header failed: " +
std::string(std::strerror(errno)));
        }
        if (::send(client_sock.get(), body.c_str(), body.size(), 0) < 0) {
            log(LogLevel::ERROR, "send body failed: " +
std::string(std::strerror(errno)));
        }
        return;
    }

    std::string req_path = (url == "/" ) ? "index.html" : url.substr(1);

    if (req_path.find("..") != std::string::npos) {
        log(LogLevel::WARN, "Blocked illegal path: " + req_path);
        req_path = "404.html";
    }

    std::error_code ec;
    auto full = std::filesystem::weakly_canonical(
        std::filesystem::path("../static") / req_path, ec
    );
    auto static_root = std::filesystem::weakly_canonical("../static", ec);

    if (std::mismatch(static_root.begin(), static_root.end(),
full.begin()).first != static_root.end()) {
        log(LogLevel::WARN, "Escape static dir: " + full.string());
        full = static_root / "404.html";
    }

    std::ifstream file(full, std::ios::binary);
    if (!file) {
        log(LogLevel::WARN, "File not found: " + full.string());
        full = static_root / "404.html";
        file.open(full, std::ios::binary);
    }

    std::ostringstream body_buf;
    body_buf << file.rdbuf();
    std::string body = body_buf.str();
    std::string mime = guess_mime(full.string());

```

```

std::ostringstream hdr;
hdr << "HTTP/1.0 "
    << (body.empty() ? "404 Not Found" : "200 OK") << "\r\n"
    << "Content-Type: " << mime << "\r\n"
    << "Content-Length: " << body.size() << "\r\n\r\n";

if (::send(client_sock.get(), hdr.str().c_str(), hdr.str().size(), 0) < 0)
{
    log(LogLevel::ERROR, "send header failed: " +
std::string(std::strerror(errno)));
}
if (::send(client_sock.get(), body.data(), body.size(), 0) < 0) {
    log(LogLevel::ERROR, "send body failed: " +
std::string(std::strerror(errno)));
}
}

```

Thread_pool.hpp

```

#pragma once

#include <vector>
#include <queue>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <atomic>

#include "socket_wrapper.hpp"
#include "logger.hpp"

class ThreadPool {
public:
    explicit ThreadPool(size_t n);
    ~ThreadPool();

    void enqueue(Socket client_sock);

    void shutdown();

private:
    void worker_loop();

```

```

    void handle_client(Socket client_sock);

    std::vector<std::thread> workers;
    std::queue<Socket> tasks;
    std::mutex mtx;
    std::condition_variable cv;
    std::atomic<bool> running;
};

```

url_utils.hpp

```

#pragma once
#include <string>

inline std::string sanitize_url(const std::string& url) {
    if (url == "/") return "index.html";
    if (url.size() > 1) {
        auto p = url.substr(1);
        if (p.find("..") == std::string::npos && !p.empty())
            return p;
    }
    return "404.html";
}

```

Styles.css

```

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
body {
    font-family: sans-serif;
    line-height: 1.5;
    background-color: #ffe5ec;
    color: #333;
}

header {
    background-color: #ff6699;
    padding: 1rem 2rem;
}

```

```
}
header h1 {
  display: inline;
  color: #fff;
  font-size: 1.5rem;
  vertical-align: middle;
}
nav {
  display: inline-block;
  margin-left: 2rem;
  vertical-align: middle;
}
nav a {
  color: #fff;
  text-decoration: none;
  margin-right: 1.5rem;
  font-weight: bold;
}
nav a.active,
nav a:hover {
  text-decoration: underline;
}

.home-content {
  padding: 2rem;
  text-align: center;
}
#intro h2 {
  color: #b30059;
  margin-bottom: 1.5rem;
}
.home-buttons {
  margin-bottom: 1.5rem;
}
.home-buttons .btn {
  display: inline-block;
  margin: 0 0.5rem;
  padding: 0.75rem 1.5rem;
  border-radius: 4px;
  text-decoration: none;
  font-weight: bold;
}
.btn.solid {
  background-color: #d60057;
  color: #fff;
}
.btn.outline {
  border: 2px solid #d60057;
```

```
    color: #d60057;
}
.btn.outline:hover {
    background-color: #d60057;
    color: #fff;
}
.home-stats {
    list-style: none;
    display: flex;
    justify-content: center;
    gap: 2rem;
    margin-top: 1rem;
}
.home-stats li strong {
    color: #d60057;
}

.page-content {
    max-width: 800px;
    margin: 2rem auto;
    background-color: #fff;
    padding: 2rem;
    border-radius: 6px;
    box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.about-content img.profile-pic {
    float: left;
    width: 150px;
    height: 150px;
    margin-right: 1.5rem;
    border-radius: 50%;
    border: 4px solid #ff6699;
}

.project-listing .page-banner {
    background-color: #ff99c8;
    padding: 1rem;
    margin-bottom: 2rem;
    border-radius: 4px;
    text-align: center;
}
.project-listing .page-banner h1 {
    color: #b30059;
    margin: 0;
}
```

```
footer {
  text-align: center;
  padding: 1rem 0;
  background-color: #ffccd9;
  color: #333;
  margin-top: 2rem;
}
```

```
h2, h3 {
  color: #b30059;
  margin-bottom: 1rem;
}
```

```
a {
  color: #b30059;
}
```

```
a:hover {
  color: #ff0066;
}
```

```
/* 404 ERROR PAGE */
```

```
.error-page {
  background-color: #ffe5ec;
  color: #333;
  text-align: center;
  padding: 4rem 2rem;
  font-family: sans-serif;
}
```

```
.error-page h1 {
  color: #b30059;
  font-size: 3rem;
  margin-bottom: 1rem;
}
```

```
.error-page p {
  margin-bottom: 1rem;
}
```

```
.error-page a {
  color: #d60057;
  text-decoration: none;
  font-weight: bold;
}
```

```
.error-page a:hover {
  color: #ff0066;
}
```

404.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>404 – Page Not Found</title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <header>
    <h1>404 – Page Not Found</h1>
    <nav>
      <a href="/">Home</a>
      <a href="/projects/">Projects</a>
      <a href="/about.html">About Me</a>
    </nav>
  </header>
  <main class="page-content">
    <p>Sorry, we couldn't find the page you were looking for.</p>
    <p>You can go back to:</p>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/projects/">Projects</a></li>
      <li><a href="/about.html">About Me</a></li>
    </ul>
  </main>
  <footer>© 2025 Kelly Casey</footer>
</body>
</html>
```

About.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>About Me – Kelly Casey</title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <header>
    <h1>About Me</h1>
    <nav>
      <a href="/">Home</a>
      <a href="/projects/">Projects</a>
      <a href="/about.html" class="active">About Me</a>
    </nav>
  </header>
```

```
<main class="page-content about-content">

  <h2>Who I Am</h2>
  
  <p>
    Hello-I'm <strong>Kelly Casey</strong> and I'm a 35-year-old student
    currently enrolled in the
    <a href="https://www.uwtsd.ac.uk/programme-
    courses/foundation/computing/computing-software-engineering-foundation-year-
    full-time">
      Computing & Software Engineering Foundation Year
    </a>
    at the
    <a href="https://www.uwtsd.ac.uk/">University of Wales Trinity Saint
    David</a>.
  </p>
  <p>
    I plan to continue to do my Master's in
    <a href="https://www.uwtsd.ac.uk/programme-courses/postgraduate-
    pgce/computing/software-engineering-and-artificial-intelligence-full">
      Software Engineering and Artificial Intelligence
    </a>
    and then move into a professional software development role.
  </p>

  <h2>Skills & Technologies</h2>
  <ul>
    <li>C++17</li>
    <li>POSIX sockets & networking</li>
    <li>Multithreading (std::thread, mutex, condition_variable)</li>
    <li>Problem-solving & software design</li>
    <li>Data analysis & visualization</li>
    <li>Database fundamentals (SQL & NoSQL)</li>
    <li>Network & cybersecurity basics</li>
    <li>HTML5, CSS3, JavaScript</li>
    <li>Linux (WSL/Ubuntu)</li>
    <li>Git</li>
    <li>Firebase Authentication & Realtime Database</li>
    <li>Android Studio & Kotlin</li>
    <li>CMake & build systems</li>
    <li>RESTful APIs</li>
    <li>Python scripting</li>
  </ul>

  <h2>Contact</h2>
  <ul>
```

```

    <li>Email: <a
href="mailto:kelly.casey1689@gmail.com">kelly.casey1689@gmail.com</a></li>
    <li>LinkedIn: <a href="https://www.linkedin.com/in/kelly-casey-
11b89299">kelly-casey-11b89299</a></li>
  </ul>

  <h2>Hobbies & Interests</h2>
  <ul>
    <li>Spending quality time with my children</li>
    <li>Baking and experimenting with new recipes</li>
    <li>Reading</li>
    <li>Solving puzzle games (Sudoku, crosswords, word searches)</li>
    <li>Building small C++ network-programming demos</li>
    <li>Crafting front-end web page experiments</li>
  </ul>

</main>

<footer>
  © 2025 Kelly Casey. All rights reserved.
</footer>
</body>
</html>

```

Index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My Portfolio – Kelly Casey</title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <header>
    <h1>Kelly Casey</h1>
    <nav>
      <a href="/" class="active">Home</a>
      <a href="/projects/">Projects</a>
      <a href="/about.html">About Me</a>
    </nav>
  </header>

  <!-- Home page uses its own wrapper so it remains full-width pink -->
  <main class="home-content">
    <section id="intro">
      <h2>Turning data into insights, one line of code at a time.</h2>

```

```

    <div class="home-buttons">
      <a href="/files/Kelly%20Casey%20CV%20040225.pdf" class="btn
solid">Download CV</a>
      <a href="/projects/" class="btn outline">See my work</a>
      <a href="/about.html" class="btn outline">Learn more about me</a>
    </div>
    <ul class="home-stats">
      <li><strong>5+</strong> Projects</li>
      <li><strong>3</strong> Languages</li>
      <li><strong>2</strong> Shells</li>
      <li><strong>1</strong> Master's planned</li>
    </ul>
  </section>

  <section id="featured">
    <h2>Featured Project</h2>
    <a href="/projects/final_project/">
      
    </a>
    <h3><a href="/projects/final_project/">Women in STEM Study
Buddy</a></h3>
    <p>
      A native Android app that connects female-identifying STEM students
with study partners
      based on shared academic interests. Secure registration, profiles,
partner discovery
      & real-time chat via Firebase.
    </p>
    <p><a href="/projects/final_project/">View project details →</a></p>
  </section>
</main>

<footer>
  © 2025 Kelly Casey. Built with C++ & ♥
</footer>
</body>
</html>

```

Test_utils.cpp

```

#define CATCH_CONFIG_MAIN
#include "catch.hpp"
#include "../src/url_utils.hpp"
#include "../src/src/mime.hpp"

TEST_CASE("URL sanitization", "[sanitize_url]") {

```

```
    REQUIRE(sanitize_url("/")      == "index.html");
    REQUIRE(sanitize_url("/about.html") == "about.html");
    REQUIRE(sanitize_url("../secret.txt") == "404.html");
    REQUIRE(sanitize_url("")        == "404.html");
    REQUIRE(sanitize_url("../")      == "404.html");
}

TEST_CASE("MIME type lookup", "[guess_mime]") {
    REQUIRE(guess_mime("index.html") == "text/html");
    REQUIRE(guess_mime("styles.css") == "text/css");
    REQUIRE(guess_mime("script.js")  == "application/javascript");
    REQUIRE(guess_mime("image.png")  == "image/png");
    REQUIRE(guess_mime("unknown.xyz") == "application/octet-stream");
}
```

socket programming implementation kelly casey

GRADEMARK REPORT

FINAL GRADE

94/100

GENERAL COMMENTS

This is a really fantastic piece of work, Kelly. Exceptionally well designed, far beyond the expected scope of this module. Really well done.

Just so you know, you submitted the assessments the wrong way around (i.e. to the opposite turnitin portals) - doesn't matter, I'll just swap them when submitting marks.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

