CSCE 3550
Tais Loye
Tal0174

AI Usage Disclosure & References

cpp-httplib

Yhirose. (n.d.). *cpp-httplib* [Computer software]. GitHub. https://github.com/yhirose/cpp-httplib

nlohmann/json

Nlohmann, N. (n.d.). *JSON for Modern C++* [Computer software]. GitHub. https://github.com/nlohmann/json

RFC 7519 - JSON Web Token

RFC 7517 - JSON Web Key

AI Usage Disclosure (As per syllabus)

This is a disclosure that I used AI to assist me during the course of this project. I want to be transparent that I have never had to code anything like this before and didn't know where to start. Despite using AI to assist me, it still took me ~12 hours to complete this assignment, as I was very adamant about trying to take notes on everything I was doing.I was able to grasp these concepts in depth and carefully analyze every step. AI assistance was used to develop most of my code, but with my thoughtful prompts that facilitated learned and smooth progression.

Prompts fed into chatgpt:

- Please breakdown where to even start on this project like I am a complete beginner who has never touched a computer. Make sure it is in C++: Implementing a basic JWKS Server
  Objective
  Develop a RESTful JWKS server that provides public keys with unique identifiers (kid) for verifying JSON Web Tokens (JWTs), implements key expiry for enhanced security, includes an authentication endpoint, and handles the issuance of JWTs with expired keys based on a query parameter.

  Chooses an appropriate language and web server for the task.

  Due to the simplicity of this assignment, I would prefer you complete it with an unfamiliar language… but as I have no way to verify it, it's not considered part of the rubric.

  This project is for educational purposes. In a real-world scenario, you'd want to integrate with a proper authentication system and ensure security best practices.

Background
HTTP/web servicesLinks to an external site.
Familiarize yourself with client/server HTTP services.
RESTLinks to an external site.
Familiarize yourself with correct HTTP methods/headers/status codes for RESTful APIs.
JOSE: JWTLinks to an external site., JWK (and JWKS):Links to an external site.
Familiarize yourself with the concepts of JWT, JWK.
Understand the importance of key expiry, and kid.
Requirements
Key Generation
Implement RSA key pair generation.
Associate a Key ID (kid) and expiry timestamp with each key.
Web server with two handlers
Serve HTTP on port 8080
A RESTful JWKS endpoint that serves the public keys in JWKS format.
Only serve keys that have not expired.
A /auth endpoint that returns an unexpired, signed JWT on a POST request.
If the “expired” query parameter is present, issue a JWT signed with the expired key pair and the expired expiry.
Documentation
Code should be organized.
Code should be commented where needed.
Code should be linted per your language/framework.
Tests
Test suite for your given language/framework with tests for you.
Test coverage should be over 80%.
Blackbox testing
Ensure the included test clientLinks to an external site. functions against your server.
The testing client will attempt a POST to /auth with no body. There is no need to check authentication for this project.
NOTE: We are not actually testing user authentication, just mocking authentication and returning a valid JWT for this user
Note:
Using kid in JWKS is crucial for systems to identify which key to use for JWT verification. Ensure that the JWTs include the kid in their headers and that the JWKS server can serve the correct key when requested with a specific kid.

Expected Outcome
At the end of the project, you should have a functional JWKS server with a RESTful API that can serve public keys with expiry and unique kid to verify JWTs.

The server should authenticate fake users requests, issue JWTs upon successful authentication, and handle the “expired” query parameter to issue JWTs signed with an expired key.

This project should take 1-12 hours, depending on your familiarity with your chosen language/framework, and web servers in general.

Deliverables
Provide a link to your GitHub repo containing your code.
Include in the repo a screenshot of the test clientLinks to an external site. running against your server.
Include in the repo a screenshot of your test suite (if present) showing the coverage percent.

As always with every screenshot, please include identifying information.

- Is this the simplest way to do this? How long does this take? What resources can I use to help me with this?
- HTTP/web servicesLinks to an external site. Familiarize yourself with client/server HTTP services. RESTLinks to an external site. Familiarize yourself with correct HTTP methods/headers/status codes for RESTful APIs. JOSE: JWTLinks to an external site., JWK (and JWKS):Links to an external site. Familiarize yourself with the concepts of JWT, JWK. Understand the importance of key expiry, and kid.
- I dont know what json is or any of these acronyms
- So what should I take away and be able to do on my own after completing this assignment? Should I have known where to start only having a little bit of C++ experience?
- Valid JWT authentication
  view longer description

  Full Marks
  15 pts

  No Marks
  0 pts
  /15 pts
  Expired JWT authentication
  view longer description

  Full Marks
  5 pts

  No Marks
  0 pts
  /5 pts
  Proper HTTP Methods And Status Codes
  view longer description

  Full Marks
  10 pts

  No Marks
  0 pts
  /10 pts
  Valid JWK Found In JWKS
  view longer description

  Full Marks
  20 pts

  No Marks
  0 pts
  /20 pts
  Expired JWK Not Found In JWKS
  view longer description

  Full Marks
  10 pts

  No Marks
  0 pts
  /10 pts

Expired JWK is expired
view longer description

Full Marks
5 pts

No Marks
0 pts
/5 pts
Test suite is present
view longer description

Full Marks
15 pts

No Marks
0 pts
/15 pts
Test coverage
view longer description

Full Marks
5 pts

No Marks
0 pts
/5 pts
Documentation/Linting/Organization

- Okay im trying to sit down and do the jwks project and i have a jwks.cpp file. Now what
- what is this file path? /nfs/home/STUDENTS/tal0174/CSCE3550/Project1/jwks.cpp
- tal0174@cell08-cse:~/CSCE3550$ g++ jwks.cpp -o jwks -std=c++17
  cc1plus: fatal error: jwks.cpp: No such file or directory
  compilation terminated.
- This showed up: The file is not displayed in the text editor because it is either binary or uses an unsupported text encoding.
- it said server running on port 8080. but it said curl not found. Do I still have to be in my cpp file? it looks like im back in just the csce 3550 folder
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -q0- http://localhost:8080/
  wget: invalid option -- '0'
  wget: invalid option -- '-'
  Usage: wget [OPTION]... [URL]...

  Try `wget --help' for more options.
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -qO- http://localhost:80wget -qO-
  http://localhost:8080/.well-known/jwks.json
  {"keys":[{"alg":"RS256","e":"AQAB","kid":"my-key-id","kty":"RSA","n":"sample-modulus","use":"sig"}]}tal0174
  @cell08-cse:~/CSCE3550/Project1$
- Nothing happens when I do that last wget command
- wget -qo- --post data="http://localhost:8080/auth
  wget: option '--post' is ambiguous; possibilities: '--post-data' '--post-file'
  Usage: wget [OPTION]... [URL]...

  Try `wget --help' for more options.
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -S -O- --timeout=3 --post-data='' http://localhost:8080/auth

echo
--2026-02-14 00:56:48-- http://localhost:8080/auth
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8080... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 404 Not Found
  Content-Length: 0
  Connection: close
2026-02-14 00:56:48 ERROR 404: Not Found.

- tal0174@cell08-cse:~/CSCE3550/Project1$ cd ~/CSCE3550/Project1
grep -n 'Post("/auth' jwks.cpp
34:    svr.Post("/auth", [](const httplib::Request& req, httplib::Response& res) {
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -S -O- --timeout=3 --post-data='' http://localhost:8080/auth
--2026-02-14 00:59:37-- http://localhost:8080/auth
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8080... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Keep-Alive: timeout=5, max=100
  Content-Length: 108
  Content-Type: application/json
Length: 108 [application/json]
Saving to: ‘STDOUT’

-                   0%[   ]   0 --.-KB/s   {"kid":"my-key-id","note":"placeholder token for wiring up the endpoint","token":"heade-   100%[=======================>]   108 --.-KB/s   in 0s

2026-02-14 00:59:37 (19.5 MB/s) - written to stdout [108/108]

- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -S -O- --timeout=3 http://localhost:8080/auth
--2026-02-14 01:05:56-- http://localhost:8080/auth
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8080... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 405 Method Not Allowed
  Connection: close
  Content-Length: 18
  Content-Type: text/plain
2026-02-14 01:05:56 ERROR 405: Method Not Allowed.

- why did he say that this could take an hour for someone who is familiar with this? what would they do that only takes an hour? would it be bare bones? I want to produce a product that is realistic with what a student should be able to produce in my stage while understanding what is happening and getting full marks
- Okay i'm rested and starting back up on the jwks project and I want to refresh you on the rubric requirements: Valid JWT authentication (/auth returns a valid JWT), Expired JWT authentication (/auth returns an expired JWT), Proper HTTP Methods And Status Codes (/auth and JWKS endpoint each have proper HTTP status codes to invalid RESTFUL methods), Valid JWK Found In JWKS (Valid JWT's kid is found in JWKS), Expired JWK Not Found In JWKS (Expired JWT's kid is not found in JWKS), Expired JWK is expired (JWT exp claim is in the past), Test suite is present (There exists a test suite with tests), Test coverage (Test coverage is >= 80%), Documentation/Linting/Organization (Code has appropriate documentation Code has been linted). Tell me how what I've done so far fits into this rubric and how I can get back on track to finish in 8 hours.
- tal0174@cell08-cse:~/CSCE3550/Project1$ cd ~/CSCE3550/Project1
ss -ltnp | grep :8080 || echo "NOT RUNNING"
LISTEN 0   5   0.0.0.0:8080   0.0.0.0:*   users:(("jwks",pid=447188,fd=3))

tal0174@cell08-cse:~/CSCE3550/Project1$ ls
httplib.h json.hpp jwks jwks.cpp
tal0174@cell08-cse:~/CSCE3550/Project1$

- was i supposed to do that in terminal 1 or 2
- You're telling me stop the server but I only have it running in terminal A can you specify which terminal im meant to be doing these things in
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -qO- http://localhost:8080/.well-known/jwks.json; echo {"keys":[{"alg":"RS256","e":"AQAB","kid":"valid-kid","kty":"RSA","n":"sample-modulus-valid","use":"sig"}]}
- tal0174@cell08-cse:~/CSCE3550/Project1$ wget -qO- --post-data='' http://localhost:8080/auth; echo {"kid":"valid-kid","note":"placeholder token for wiring up the endpoint","token":"header.payload.signature"}
- How do I undo my last action
  I messed up. CAn you append my existing code with the changes you suggested: #include "httplib.h"

```
#include "json.hpp"
#include <iostream>
#include <ctime>
#include <string>
#include <vector>

using json = nlohmann::json;

struct KeyEntry {
    std::string kid;
    json jwk;              // public JWK fields
    std::time_t expires_at; // epoch seconds
};

static bool is_expired(std::time_t now, const KeyEntry& k) {
    return k.expires_at <= now;
}

static std::string base64url_encode(const std::string& in) {
    static const char* b64 =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    std::string out;
    int val = 0, valb = -6;
    for (unsigned char c : in) {
        val = (val << 8) + c;
        valb += 8;
        while (valb >= 0) {
            out.push_back(b64[(val >> valb) & 0x3F]);
            valb -= 6;
        }
    }
    if (valb > -6) out.push_back(b64[((val << 8) >> (valb + 8)) & 0x3F]);
    while (out.size() % 4) out.push_back('=');

    // Convert to base64url
    for (char& ch : out) {
        if (ch == '+') ch = '-';
        else if (ch == '/') ch = '_';
    }
    while (!out.empty() && out.back() == '=') out.pop_back();
    return out;
}
```

```
int main() {
    httplib::Server svr;

    // ---- Key store: one valid, one expired ----
    std::time_t now = std::time(nullptr);

    KeyEntry validKey{
        "valid-kid",
        {
            {"kty", "RSA"},
            {"kid", "valid-kid"},
            {"use", "sig"},
            {"alg", "RS256"},
            {"n", "sample-modulus-valid"},
            {"e", "AQAB"}
        },
        now + 3600 // valid for 1 hour
    };

    KeyEntry expiredKey{
        "expired-kid",
        {
            {"kty", "RSA"},
            {"kid", "expired-kid"},
            {"use", "sig"},
            {"alg", "RS256"},
            {"n", "sample-modulus-expired"},
            {"e", "AQAB"}
        },
        now - 3600 // expired 1 hour ago
    };

    std::vector<KeyEntry> keys{ validKey, expiredKey };
    // ----------------------------------------

    svr.Get("/", [](const httplib::Request& req, httplib::Response& res) {
        res.set_content("JWKS Server Running", "text/plain");
    });

    // JWKS endpoint: return ONLY non-expired keys
    svr.Get("/.well-known/jwks.json", [keys](const httplib::Request& req, httplib::Response& res) mutable {
        std::time_t now = std::time(nullptr);

        json out;
        out["keys"] = json::array();

        for (const auto& k : keys) {
            if (!is_expired(now, k)) {
                out["keys"].push_back(k.jwk);
            }
        }

        res.status = 200;
```

```
    res.set_content(out.dump(), "application/json");
  });

  // Auth endpoint: placeholder token for now, but choose valid vs expired kid
  // Example:
  //   POST /auth            -> uses valid-kid
  //   POST /auth?expired=true -> uses expired-kid (later we'll make exp claim past + sign with expired key)
  svr.Post("/auth", [keys](const httplib::Request& req, httplib::Response& res) mutable {
  bool wantExpired = req.has_param("expired") && req.get_param_value("expired") == "true";
  std::string chosenKid = wantExpired ? "expired-kid" : "valid-kid";

  std::time_t now = std::time(nullptr);

  json header = {
    {"alg", "RS256"},
    {"typ", "JWT"},
    {"kid", chosenKid}
  };

  json payload = {
    {"sub", "student"},
    {"iat", now},
    {"exp", wantExpired ? (now - 300) : (now + 300)} // expired: 5 min ago, valid: 5 min ahead
  };

  std::string encodedHeader = base64url_encode(header.dump());
  std::string encodedPayload = base64url_encode(payload.dump());

  // Signature placeholder FOR NOW (next step we make it real RS256 with OpenSSL)
  std::string token = encodedHeader + "." + encodedPayload + "." + "signature";

  json response = {
    {"token", token},
    {"kid", chosenKid}
  };

  res.status = 200;
  res.set_content(response.dump(), "application/json");
});




  // Proper method handling: GET /auth should be 405
  svr.Get("/auth", [](const httplib::Request& req, httplib::Response& res) {
    res.status = 405;
    res.set_content("Method Not Allowed", "text/plain");
  });

  std::cout << "Server running on port 8080...\n";
  svr.listen("0.0.0.0", 8080);
}
```

- what terminal do i do step 2-4 in
- was step 4 supposed create a ton of files

- Lines executed:0.00% of 8723
- tal0174@cell08-cse:~/CSCE3550/Project1$ python3 -m unittest -v test_jwks.py
  ls -l *.gcda

  Ran 0 tests in 0.000s

  OK
  ls: cannot access '*.gcda': No such file or directory
- tal0174@cell08-cse:~/CSCE3550/Project1$ python3 -m unittest discover -v
  test_auth_expired (test_jwks.TestJWKS) ... ok
  test_auth_valid (test_jwks.TestJWKS) ... ok
  test_get_auth_405 (test_jwks.TestJWKS) ... ok
  test_jwks_filters_expired (test_jwks.TestJWKS) ... ok
  test_root (test_jwks.TestJWKS) ... ok

  Ran 5 tests in 0.457s

  OK
- tal0174@cell08-cse:~/CSCE3550/Project1$ ls -l *.gcda
  ls: cannot access '*.gcda': No such file or directory
- tal0174@cell08-cse:~/CSCE3550/Project1$ ls -l *.gcda
  -rw-r--r-- 1 tal0174 domain^users 287712 Feb 15 18:25 jwks.gcda
  tal0174@cell08-cse:~/CSCE3550/Project1$ gcov jwks.cpp | grep "Lines executed"
  Lines executed:90.51% of 158
  Lines executed:81.67% of 60
  Lines executed:90.00% of 10
  Lines executed:100.00% of 13
  Lines executed:54.55% of 33
  Lines executed:84.21% of 19
  Lines executed:89.62% of 106
  Lines executed:65.81% of 117
  Lines executed:77.27% of 22
  Lines executed:38.17% of 985
  Lines executed:28.03% of 2758
  Lines executed:52.11% of 190
  Lines executed:88.68% of 53
  Lines executed:84.13% of 63
  Lines executed:100.00% of 2
  Lines executed:80.49% of 205
  Lines executed:100.00% of 14
  Lines executed:58.06% of 31
  Lines executed:72.41% of 29
  Lines executed:100.00% of 2
  Lines executed:50.00% of 4
  Lines executed:33.33% of 57
  Lines executed:55.17% of 29
  Lines executed:0.00% of 48
  Lines executed:58.97% of 78
  Lines executed:0.00% of 34
  Lines executed:66.67% of 6
  Lines executed:100.00% of 16
  Lines executed:65.07% of 146

Lines executed:22.48% of 129
Lines executed:8.27% of 133
Lines executed:71.43% of 7
Lines executed:82.04% of 362
Lines executed:83.90% of 118
Lines executed:36.17% of 94
Lines executed:12.24% of 49
Lines executed:75.54% of 139
Lines executed:86.67% of 15
Lines executed:32.00% of 75
Lines executed:47.67% of 86
Lines executed:29.28% of 304
Lines executed:29.41% of 68
Lines executed:50.00% of 38
Lines executed:100.00% of 4
Lines executed:100.00% of 13
Lines executed:21.12% of 232
Lines executed:62.35% of 170
Lines executed:84.62% of 13
Lines executed:12.24% of 49
Lines executed:35.76% of 453
Lines executed:14.03% of 221
Lines executed:100.00% of 8
Lines executed:0.00% of 1
Lines executed:100.00% of 19
Lines executed:100.00% of 13
Lines executed:42.86% of 35
Lines executed:83.33% of 6
Lines executed:26.83% of 41
Lines executed:87.50% of 64
Lines executed:0.00% of 14
Lines executed:100.00% of 3
Lines executed:0.00% of 112
Lines executed:58.33% of 12
Lines executed:87.50% of 24
Lines executed:0.00% of 9
Lines executed:0.00% of 4
Lines executed:57.58% of 33
Lines executed:0.00% of 9
Lines executed:100.00% of 13
Lines executed:71.43% of 14
Lines executed:82.35% of 17
Lines executed:0.00% of 12
Lines executed:100.00% of 4
Lines executed:0.00% of 18
Lines executed:64.29% of 14
Lines executed:92.00% of 25
Lines executed:31.43% of 35
Lines executed:0.00% of 5
Lines executed:42.86% of 14
Lines executed:0.00% of 4
Lines executed:0.00% of 2
Lines executed:38.89% of 18
Lines executed:0.00% of 23

Lines executed:80.00% of 10
Lines executed:0.00% of 10
Lines executed:100.00% of 5
Lines executed:100.00% of 6
Lines executed:0.00% of 1
Lines executed:0.00% of 4
Lines executed:0.00% of 2
Lines executed:66.67% of 3
Lines executed:0.00% of 1
Lines executed:42.10% of 8727
tal0174@cell08-cse:~/CSCE3550/Project1$

- Im scared by removing things that i will omit something i need for submission
- Okay I havent done anything with git yet I also havent removed anything. Or made a readme. What do I need screenshots of?
- Can you help me make a read.me
- I havent removed the temp files and stuff. Should I do that
- yes. Should i do this command: git remote add origin https://github.com/Tlloye/CSCE3550-jwks.git
  git branch -M main
  git push -u origin main