Assignment 1(LinuxOS)

Objective: Develop a console-based file explorer application in C++that interfaces with the Linux operating system to manage files and directories.

Day-wise Tasks:  
Day 1: Design the application structure and setup the development environment. Start with basic file operations like listing files in a directory.  
  
Code:  
#include <iostream>

#include <dirent.h>

#include <cstring>

#include <unistd.h>

**void** listFiles(**const** std::string **&**path)

{

    DIR \*dir = opendir(path.c\_str());

    if (dir == nullptr)

    {

        std::cerr << "Error opening directory: " << path << std::endl;

        return;

    }

**struct** dirent **\***entry;

    while ((entry = readdir(dir)) != nullptr)

    {

        std::cout << entry->d\_name << std::endl;

    }

    closedir(dir);

}

**int** main()

{

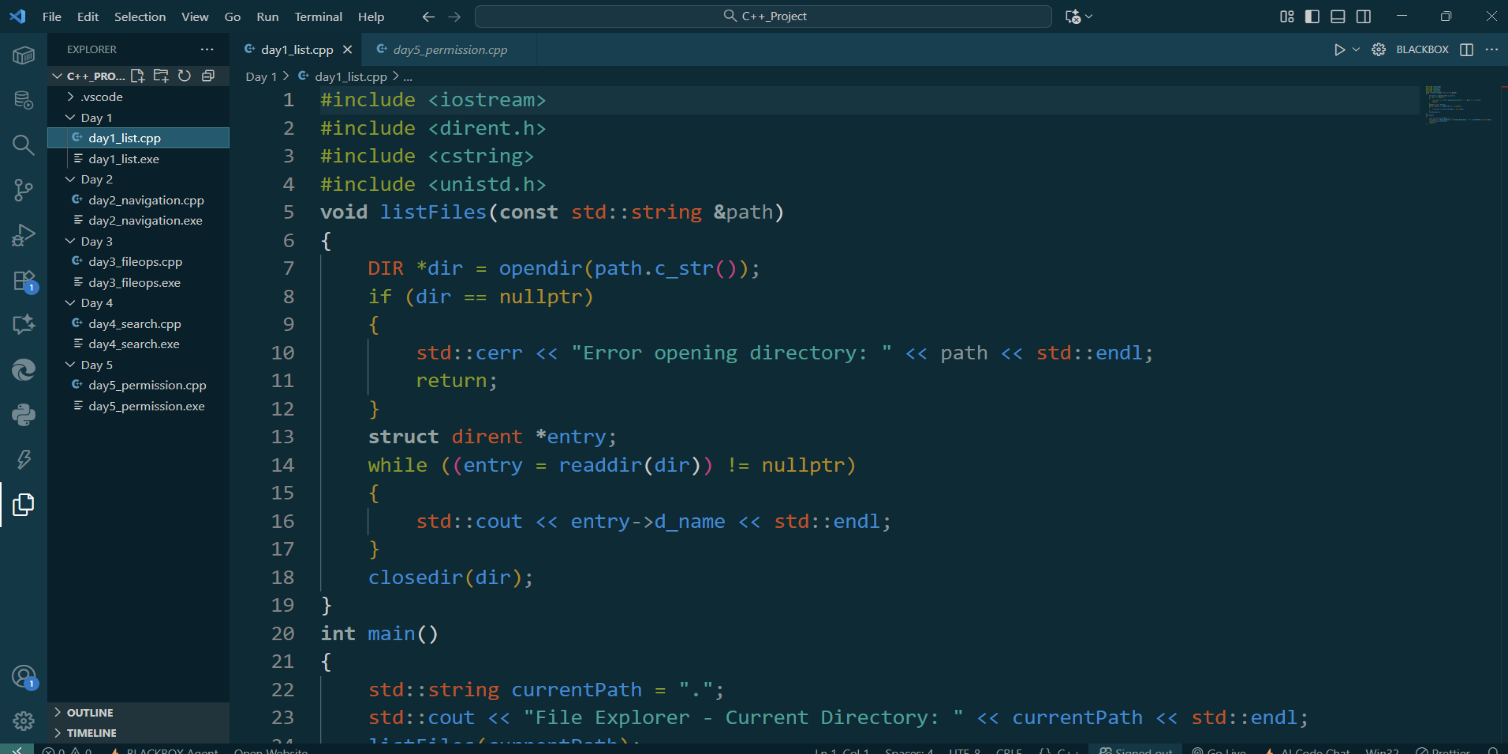
    std::string currentPath = ".";

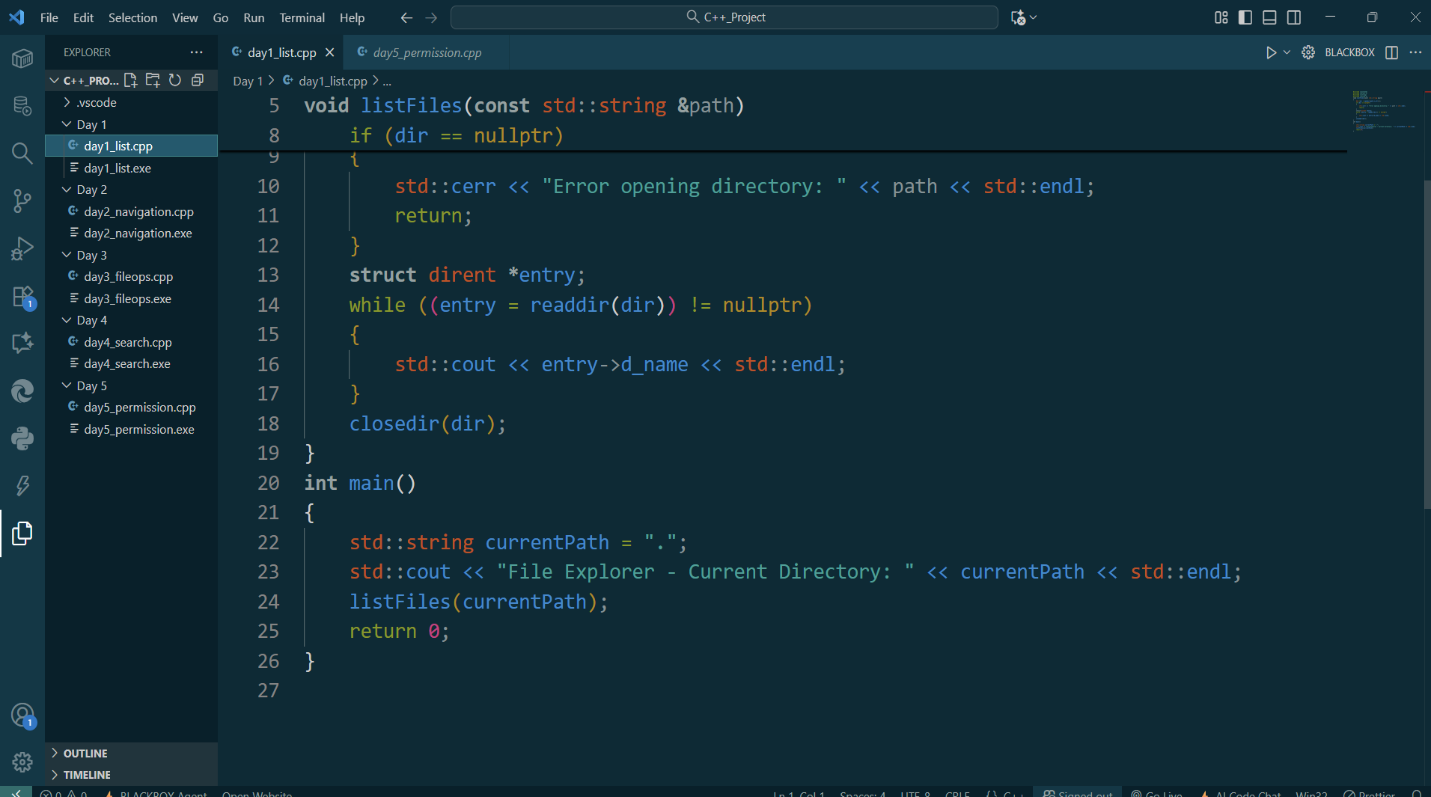
    std::cout << "File Explorer - Current Directory: " << currentPath << std::endl;

    listFiles(currentPath);

    return 0;

}

Screenshots:  




Day 2: Implement file and directory navigation features. Enable the user to move through directories.  
Code:  
#include <iostream>

#include <dirent.h>

#include <cstring>

#include <unistd.h>

#include <limits>

**void** listFiles(**const** std::string **&**path)

{

    DIR \*dir = opendir(path.c\_str());

    if (dir == nullptr)

    {

        std::cerr << "Error opening directory: " << path << std::endl;

        return;

    }

**struct** dirent **\***entry;

    while ((entry = readdir(dir)) != nullptr)

    {

        std::cout << entry->d\_name << std::endl;

    }

    closedir(dir);

}

**int** main()

{

    std::string currentPath = ".";

    std::string command;

    while (true)

    {

        std::cout << "File Explorer - Current Directory: " << currentPath << std:endl;

        listFiles(currentPath);

        std::cout << "Enter command (cd <dir> or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")

            break;

        if (command.substr(0, 3) == "cd ")

        {

            std::string newDir = command.substr(3);

            if (chdir(newDir.c\_str()) == 0)

            {

**char** cwd[1024];

                getcwd(cwd, sizeof(cwd));

                currentPath = cwd;

            }

            else

            {

                std::cerr << "Error changing directory." << std::endl;

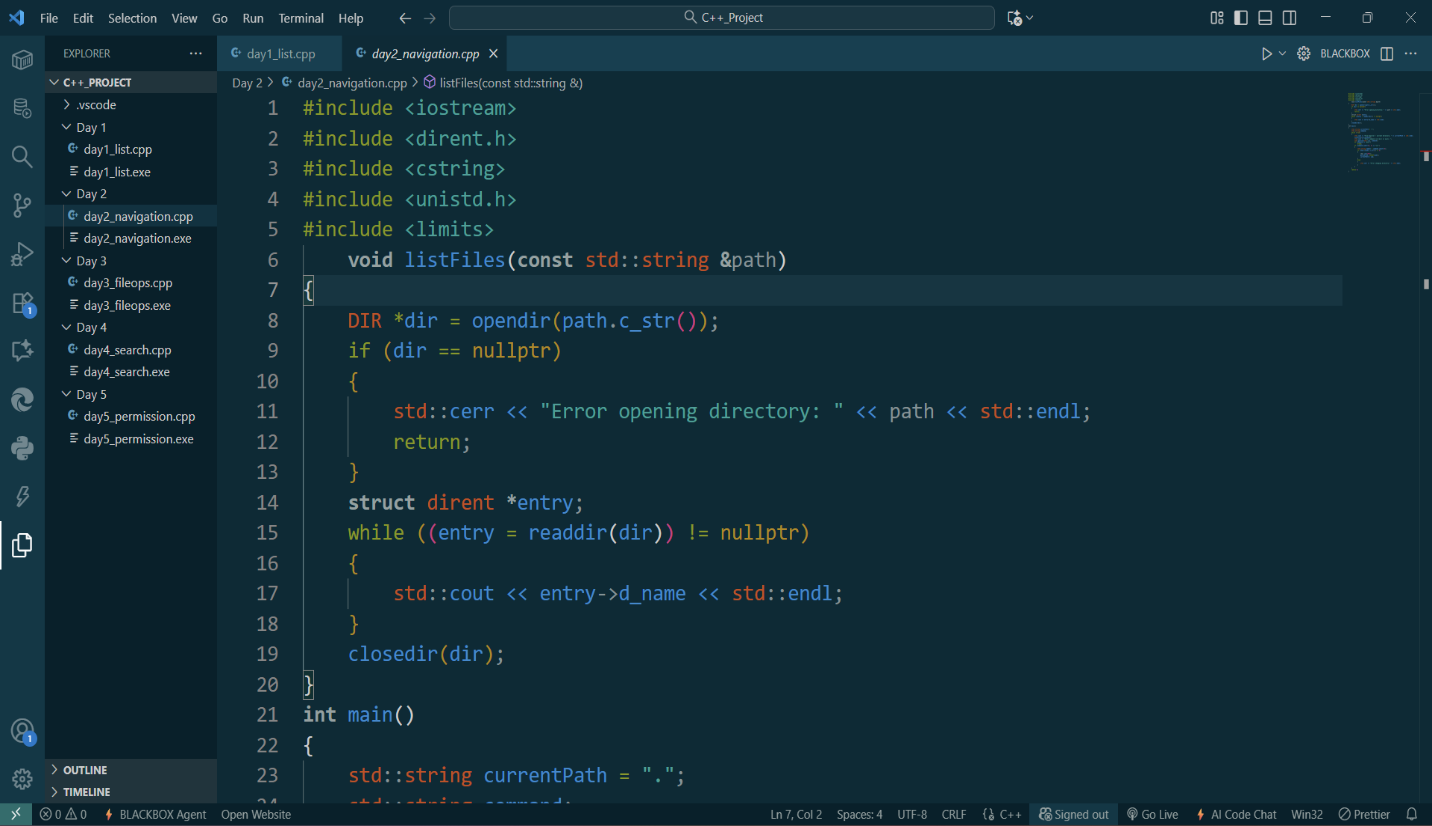
            }

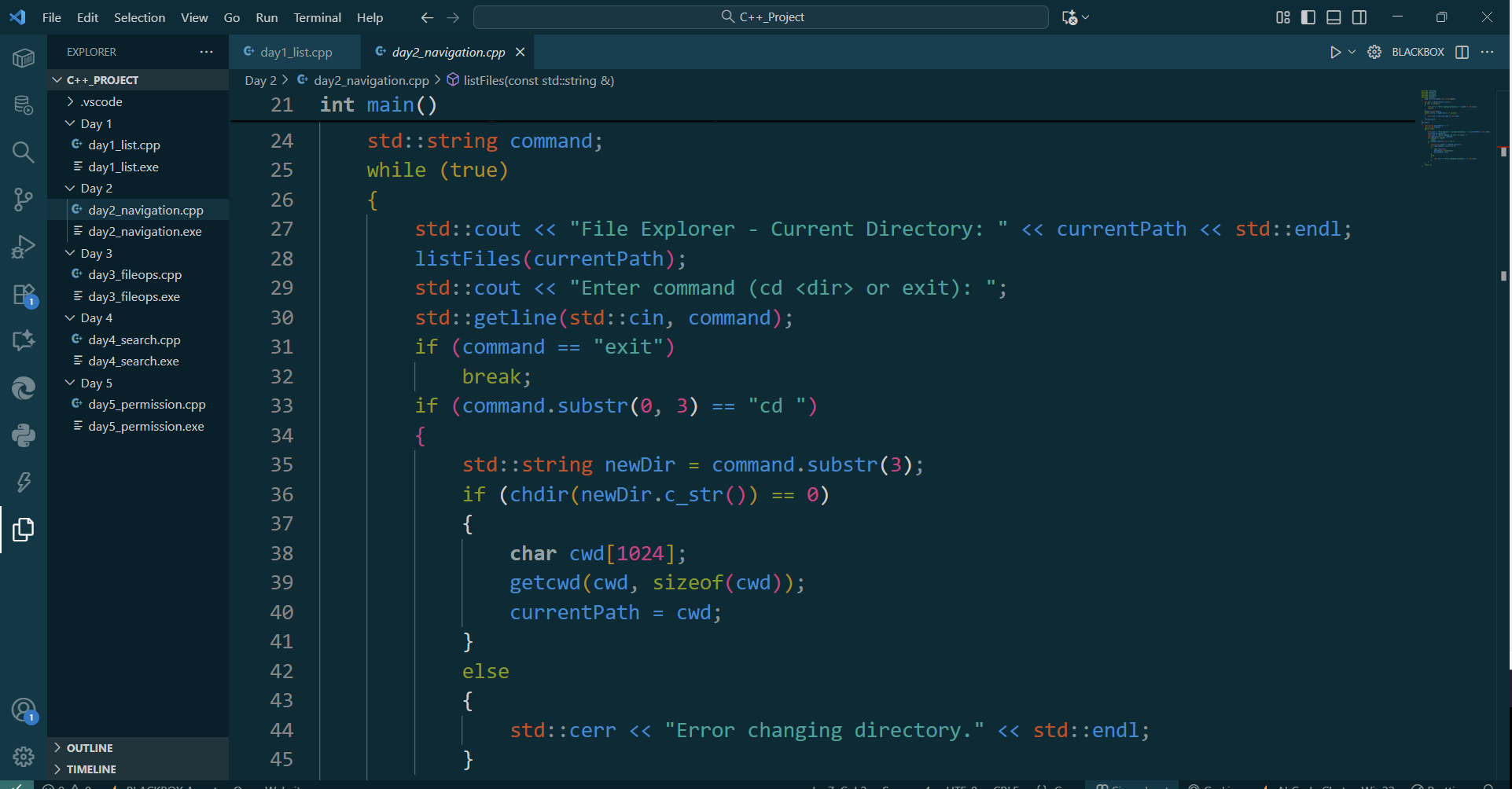
        }

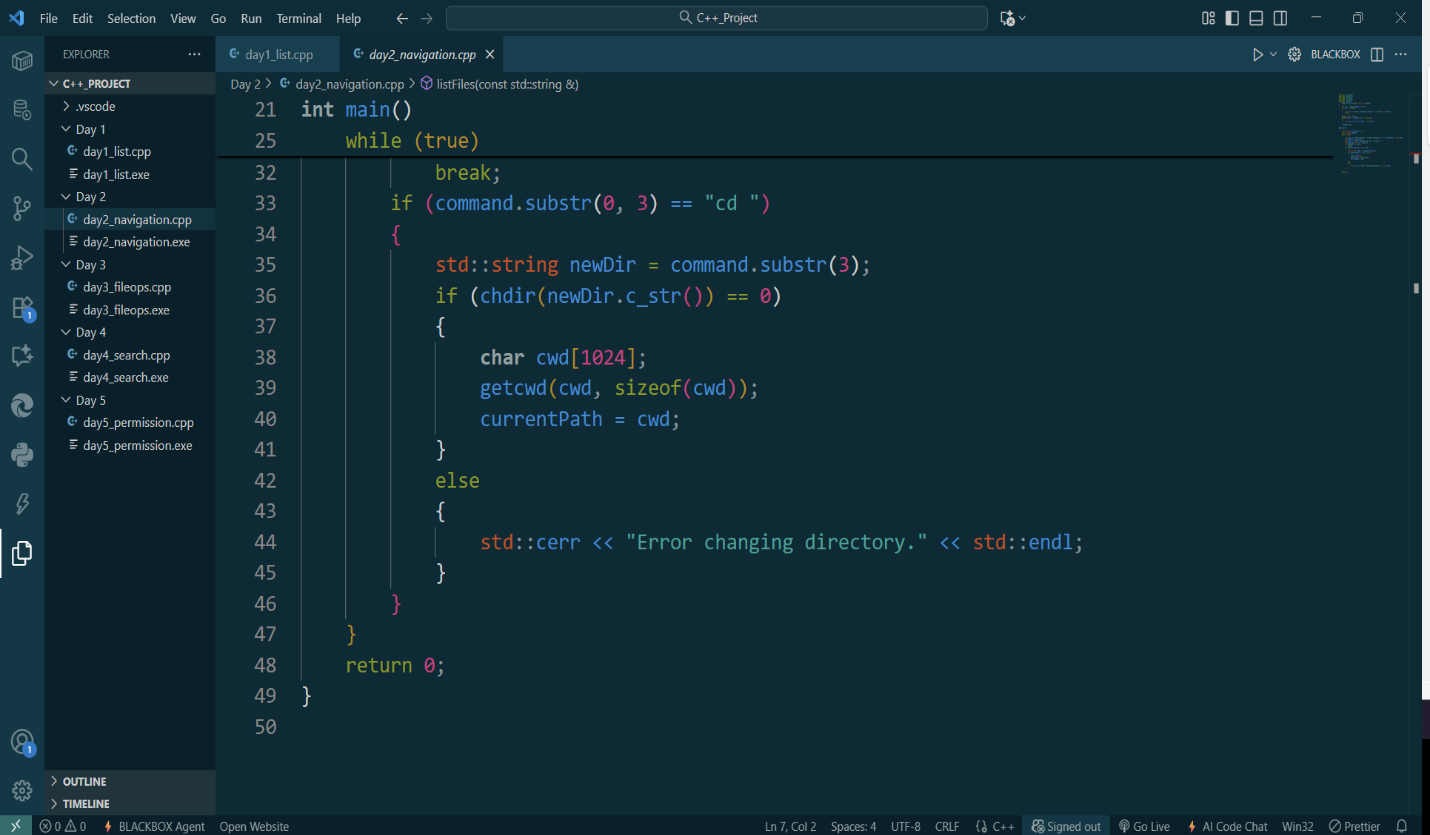
    }

    return 0;

}

Screenshot:  






Day 3: Add file manipulation capabilities (copy, move, delete, create).  
Code:  
#include <iostream>

#include <dirent.h>

#include <unistd.h>

#include <cstring>

#include <fstream>

**void** listFiles(**const** **char** **\***path)

{

    DIR \*dir = opendir(path);

    if (dir)

    {

**struct** dirent **\***entry;

        while ((entry = readdir(dir)))

        {

            std::cout << entry->d\_name << std::endl;

        }

        closedir(dir);

    }

}

**void** copyFile(**const** **char** **\***src, **const** **char** **\***dst)

{

    std::ifstream in(src, std::ios::binary);

    std::ofstream out(dst, std::ios::binary);

    out << in.rdbuf();

}

**int** main()

{

**char** currentPath[1024] = ".";

    std::string command;

    while (true)

    {

        std::cout << "Current Directory: " << currentPath << std::endl;

        listFiles(currentPath);

        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>, or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")

            break;

        if (command.substr(0, 3) == "cd ")

        {

            std::string newDir = command.substr(3);

            if (chdir(newDir.c\_str()) == 0)

            {

                getcwd(currentPath, sizeof(currentPath));

            }

        }

        else if (command.substr(0, 5) == "copy ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            copyFile(src.c\_str(), dst.c\_str());

        }

        else if (command.substr(0, 5) == "move ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            rename(src.c\_str(), dst.c\_str());

        }

        else if (command.substr(0, 7) == "delete ")

        {

            std::string file = command. substr(7);

            remove(file.c\_str());

        }

        else if (command.substr(0, 7) == "create ")

        {

            std::string file = command.substr(7);

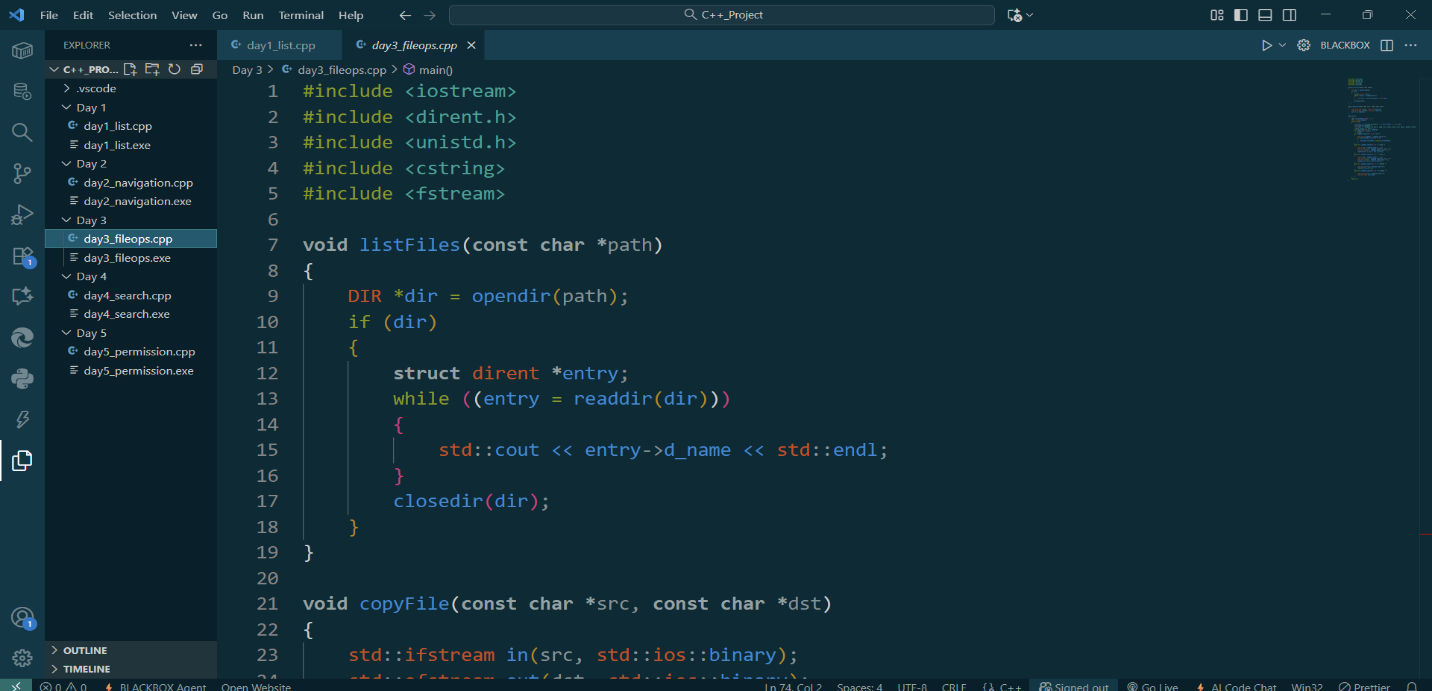
            std::ofstream out(file);

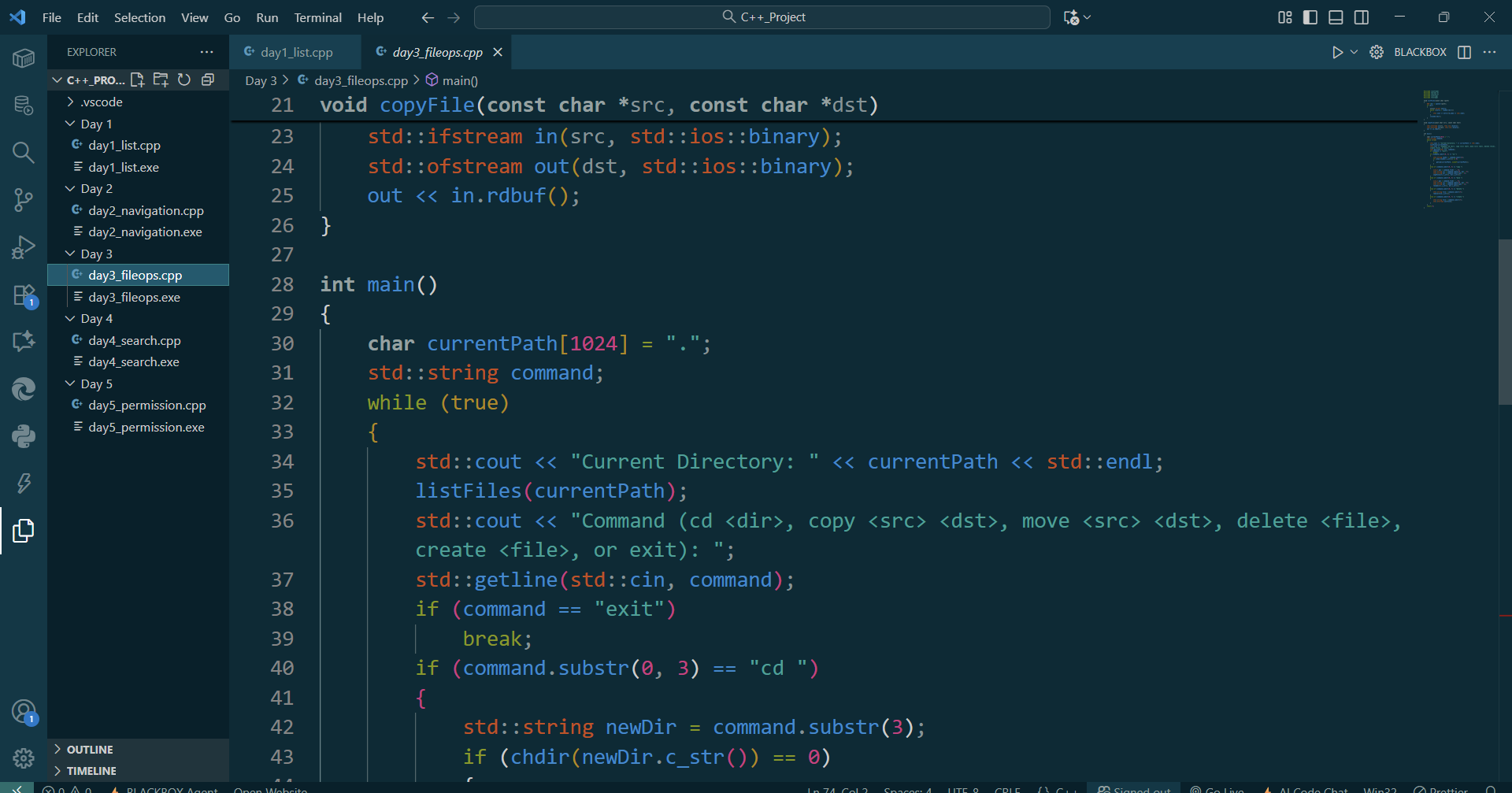
        }

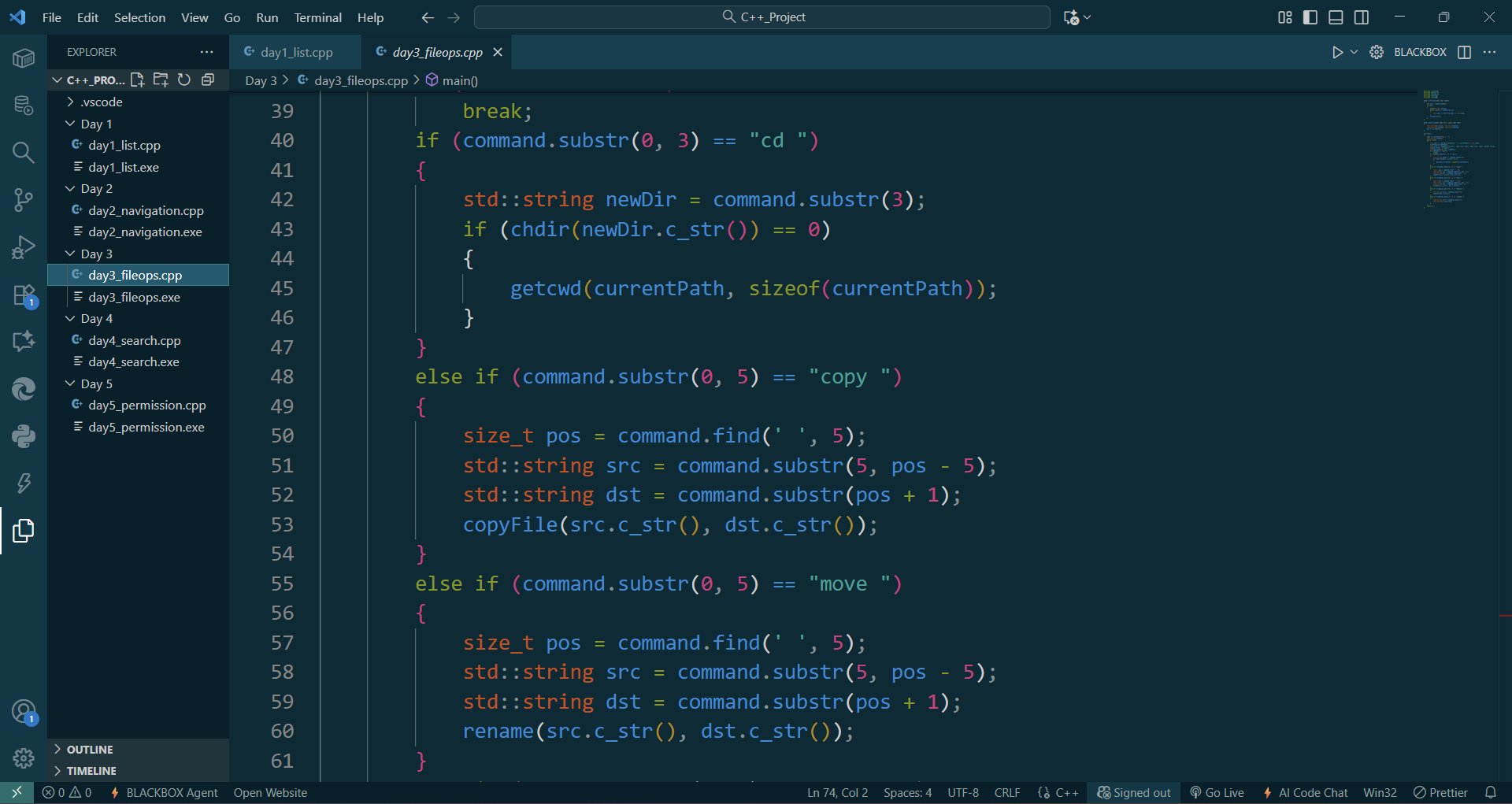
    }

    return 0;

}

Screenshots:  






Day 4: Implement file search functionality within the file explorer.  
Code:  
#include <iostream>

#include <dirent.h>

#include <unistd.h>

#include <cstring>

#include <fstream>

#include <stack>

**void** listFiles(**const** **char** **\***path)

{

    DIR \*dir = opendir(path);

    if (dir)

    {

**struct** dirent **\***entry;

        while ((entry = readdir(dir)))

        {

            std::cout << entry->d\_name << std::endl;

        }

        closedir(dir);

    }

}

**void** copyFile(**const** **char** **\***src, **const** **char** **\***dst)

{

    std::ifstream in(src, std::ios::binary);

    std::ofstream out(dst, std::ios::binary);

    out << in.rdbuf();

}

**void** searchFiles(**const** **char** **\***root, **const** **char** **\***name)

{

    std::stack<std::string> dirs;

    dirs.push(root);

    while (!dirs.empty())

    {

        std::string current = dirs.top();

        dirs.pop();

        DIR \*dir = opendir(current.c\_str());

        if (dir)

        {

**struct** dirent **\***entry;

            while ((entry = readdir(dir)))

            {

                if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)

                    continue;

                std::string fullPath = current + "/" + entry->d\_name;

                if (strcmp(entry->d\_name, name) == 0)

                {

                    std::cout << "Found: " << fullPath << std::endl;

                }

                if (entry->d\_type == DT\_DIR)

                {

                    dirs.push(fullPath);

                }

            }

            closedir(dir);

        }

    }

}

**int** main()

{

**char** currentPath[1024] = ".";

    std::string command;

    while (true)

    {

        std::cout << "Current Directory: " << currentPath << std::endl;

        listFiles(currentPath);

        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>, search <name>, or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")

            break;

        if (command.substr(0, 3) == "cd ")

        {

            std::string newDir = command.substr(3);

            if (chdir(newDir.c\_str()) == 0)

            {

                getcwd(currentPath, sizeof(currentPath));

            }

        }

        else if (command.substr(0, 5) == "copy ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            copyFile(src.c\_str(), dst.c\_str());

        }

        else if (command.substr(0, 5) == "move ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            rename(src.c\_str(), dst.c\_str());

        }

        else if (command. substr(0, 7) == "delete ")

        {

            std::string file = command.substr(7);

            remove(file.c\_str());

        }

        else if (command.substr(0, 7) == "create ")  
 {

            std::string file = command.substr(7);

            std::ofstream out(file);

        }

        else if (command.substr(0, 7) == "search ")

        {

            std::string name = command.substr(7);

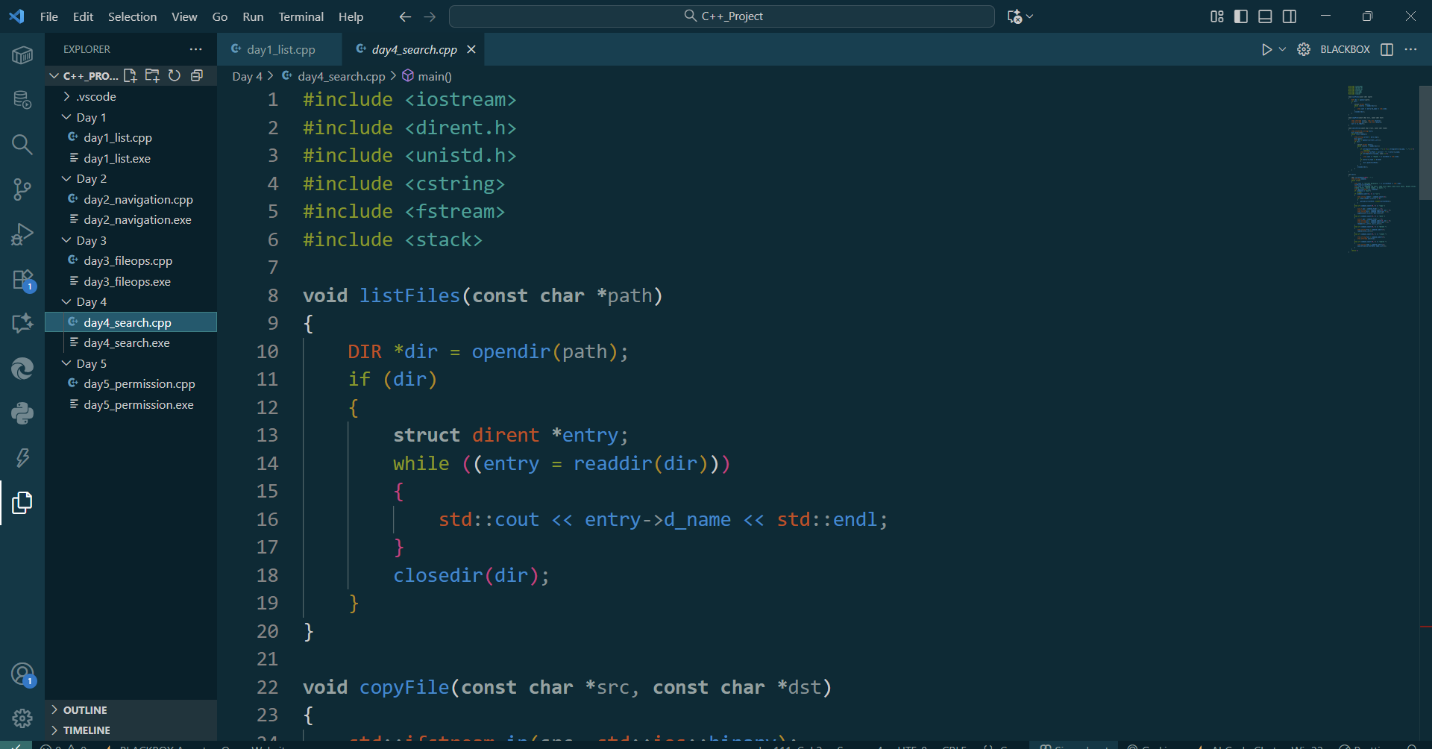
            searchFiles(currentPath, name.c\_str());

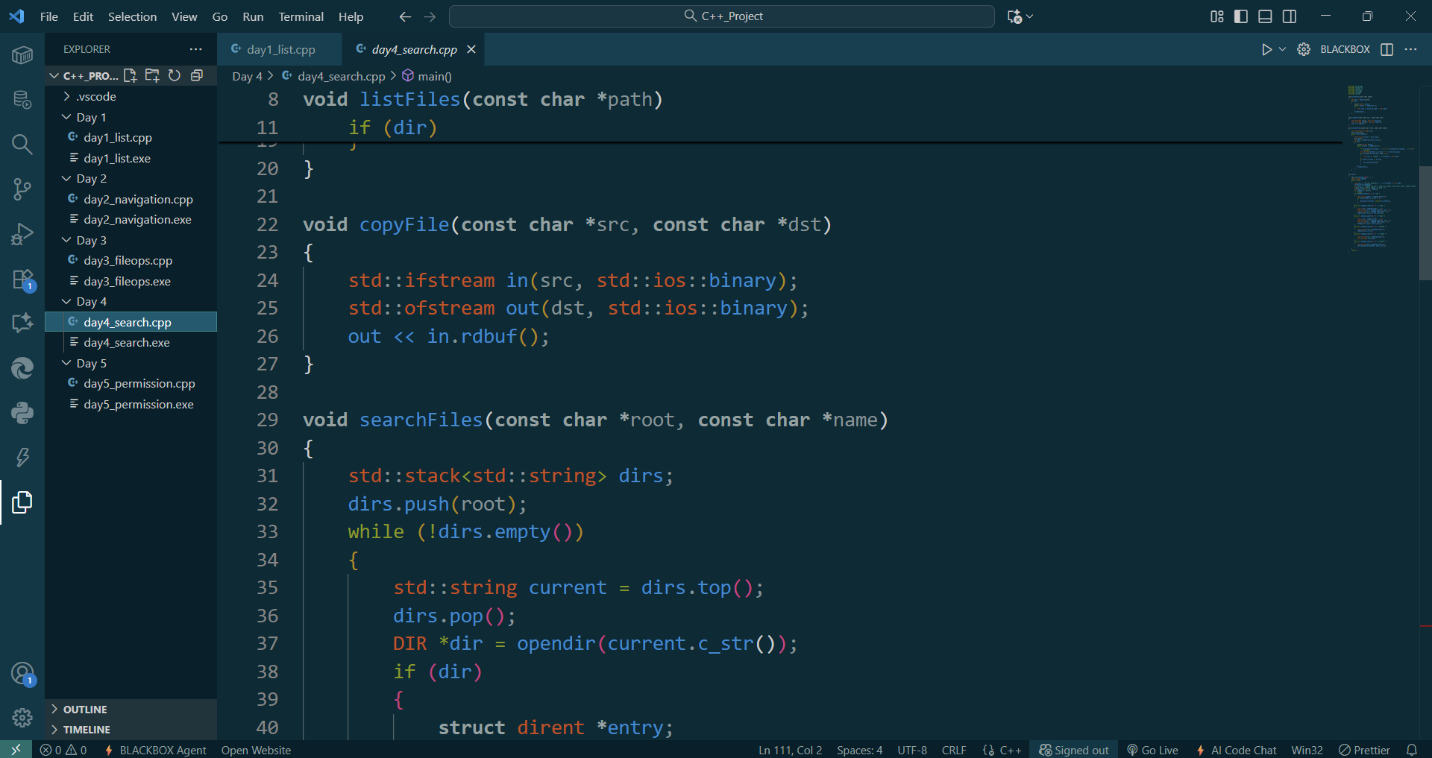
        }

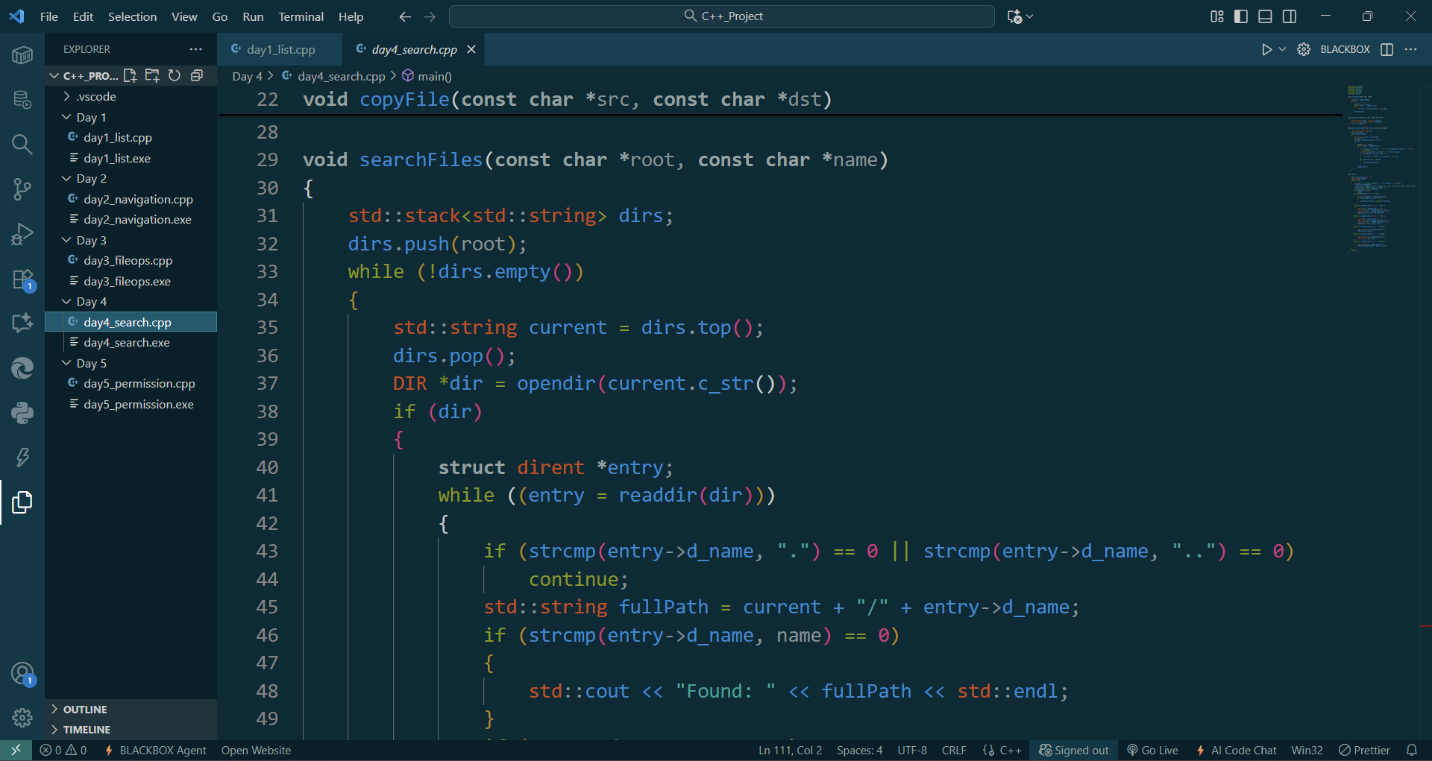
    }

    return 0;

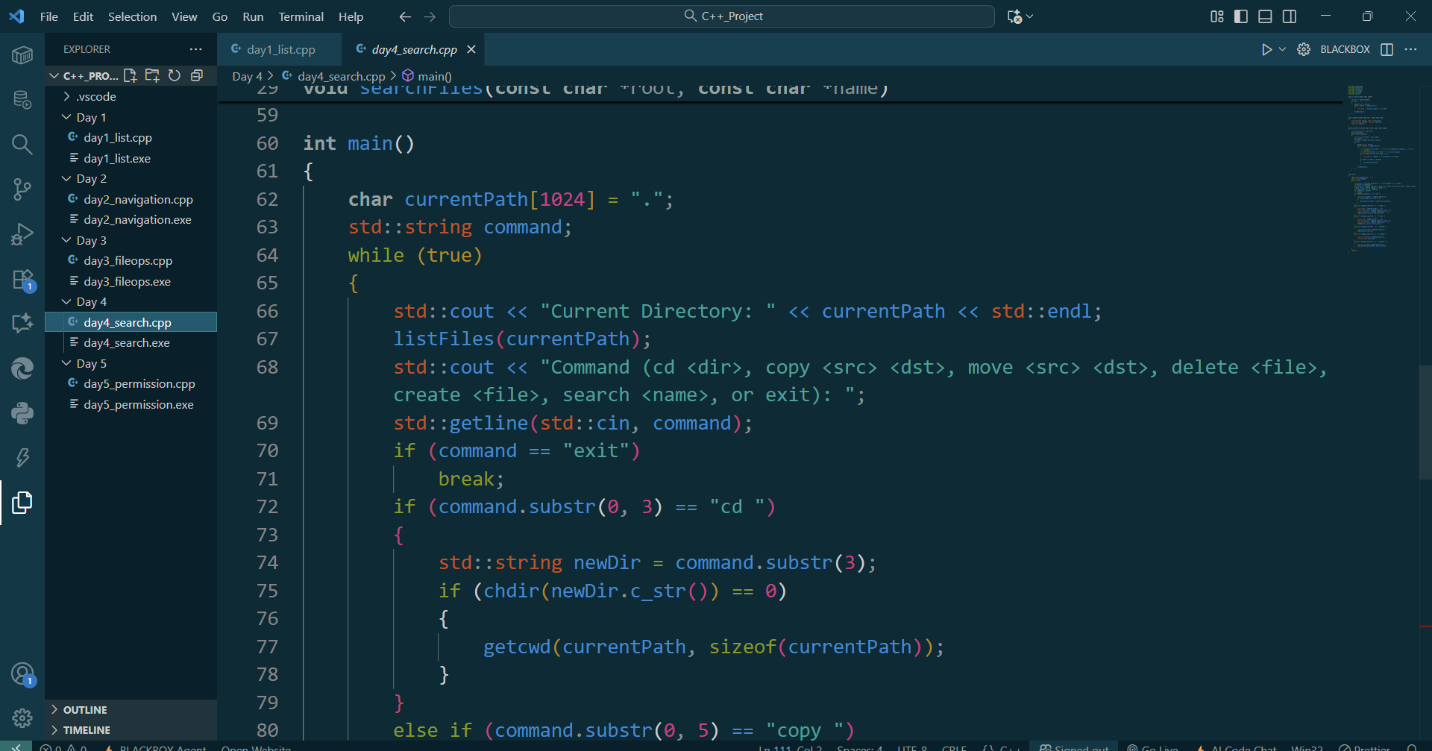
}

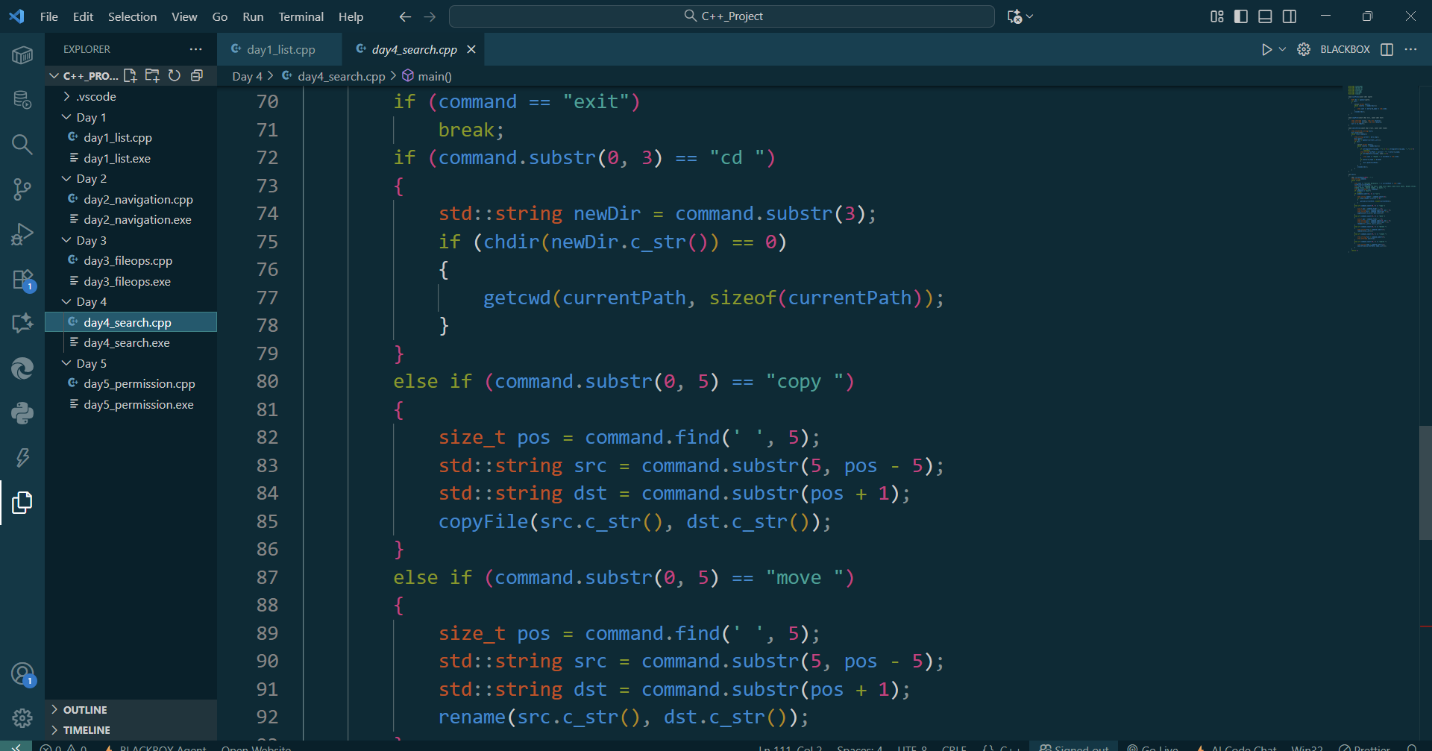
Screenshots:  


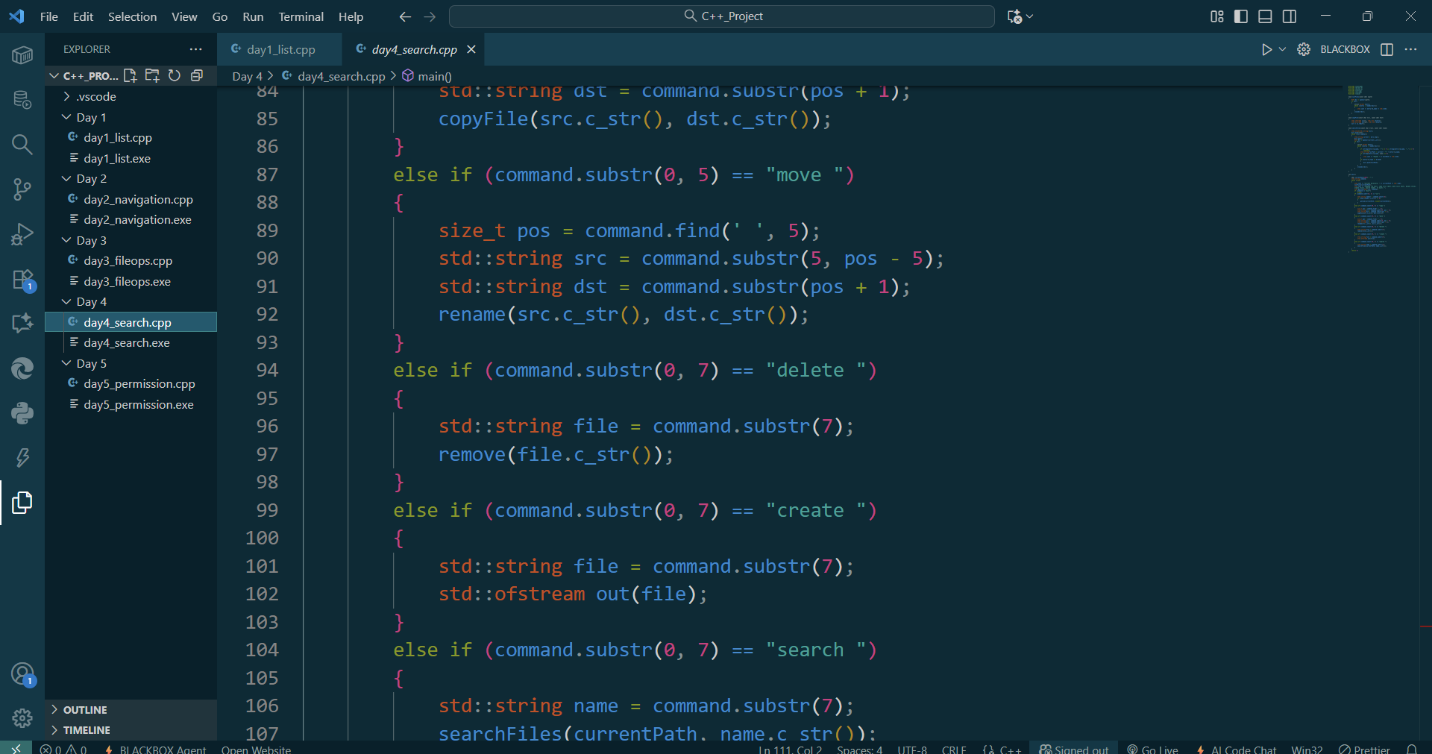


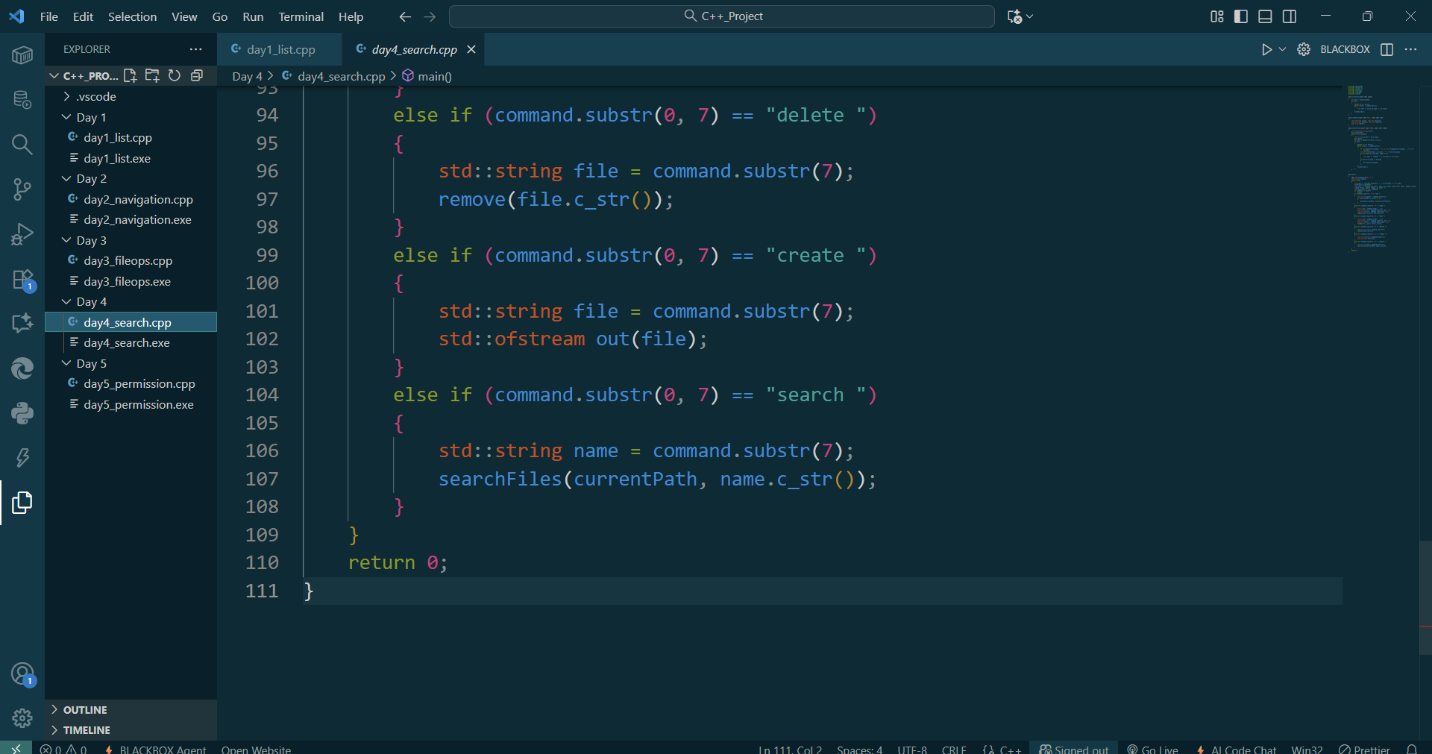












Day 5: Add file permission management features.  
Code:  
#include <iostream>

#include <dirent.h>

#include <unistd.h>

#include <cstring>

#include <fstream>

#include <stack>

#include <sys/stat.h>

**void** listFiles(**const** **char** **\***path)

{

    DIR \*dir = opendir(path);

    if (dir)

    {

**struct** dirent **\***entry;

        while ((entry = readdir(dir)))

        {

            std::cout << entry->d\_name << std::endl;

        }

        closedir(dir);

    }

}

**void** copyFile(**const** **char** **\***src, **const** **char** **\***dst)

{

    std::ifstream in(src, std::ios::binary);

    std::ofstream out(dst, std::ios::binary);

    out << in.rdbuf();

}

**void** searchFiles(**const** **char** **\***root, **const** **char** **\***name)

{

    std::stack<std::string> dirs;

    dirs. push(root);

    while (!dirs.empty ())

    {

        std::string current = dirs.top();

        dirs.pop();

        DIR \*dir = opendir(current.c\_str());

        if (dir)

        {

**struct** dirent **\***entry;

            while ((entry = readdir(dir)))

            {

                if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)

                    continue;

                std::string fullPath = current + "/" + entry->d\_name;

                if (strcmp(entry->d\_name, name) == 0)

                {

                    std::cout << "Found: " << fullPath << std::endl;

                }

                if (entry->d\_type == DT\_DIR)

                {

                    dirs.push(fullPath);

                }

            }

            closedir(dir);

        }

    }

}

**int** main()

{

**char** currentPath [1024] = ".";

    std::string command;

    while (true)

    {

        std::cout << "Current Directory: " << currentPath << std::endl;

        listFiles(currentPath);

        std::cout << "Command (cd <dir>, copy <src> <dst>, move <src> <dst>, delete <file>, create <file>, search <name>, permissions <file>, chmod <mode> <file>, or exit): ";

        std::getline(std::cin, command);

        if (command == "exit")

            break;

        if (command.substr(0, 3) == "cd ")

        {

            std::string newDir = command.substr(3);

            if (chdir(newDir.c\_str()) == 0)

            {

                getcwd(currentPath, sizeof(currentPath));

            }

        }

        else if (command.substr(0, 5) == "copy ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            copyFile(src.c\_str(), dst.c\_str());

        }

        else if (command.substr(0, 5) == "move ")

        {

            size\_t pos = command.find(' ', 5);

            std::string src = command.substr(5, pos - 5);

            std::string dst = command.substr(pos + 1);

            rename(src.c\_str(), dst.c\_str());

        }

        else if (command.substr(0, 7) == "delete ")

        {

            std::string file = command.substr(7);

            remove(file.c\_str());

        }

        else if (command.substr(0, 7) == "create ")

        {

            std::string file = command.substr(7);

            std::ofstream out(file);

        }

        else if (command.substr(0, 7) == "search ")

        {

            std::string name = command.substr(7);

            searchFiles(currentPath, name.c\_str());

        }

        else if (command.substr(0, 12) == "permissions ")

        {

            std::string file = command.substr(12);

**struct** stat st;

            if (stat(file.c\_str(), &st) == 0)

            {

                std::cout << "Permissions: " << std::oct << (st.st\_mode & 0777) << std::dec << std::endl;

            }

        }

        else if (command.substr(0, 6) == "chmod ")

        {

            size\_t pos = command.find(' ', 6);

            std::string modeStr = command.substr(6, pos - 6);

            std::string file = command.substr(pos + 1);

**int** mode = strtol (modeStr.c\_str(), nullptr, 8);

            chmod(file.c\_str(), mode);

        }

    }

    return 0;

}

Screenshot:

