Course:
CIS-17B

Project:
Online Connect 4

Assignment:
C++ Progress Report

Group Members:
Kelby Knight, Ryan Westfall, Patrick Pascual,Francisco Sanchez, Anthony Nguyen, Amare Terrell, Janaye Jackson, Kyle Riebeling

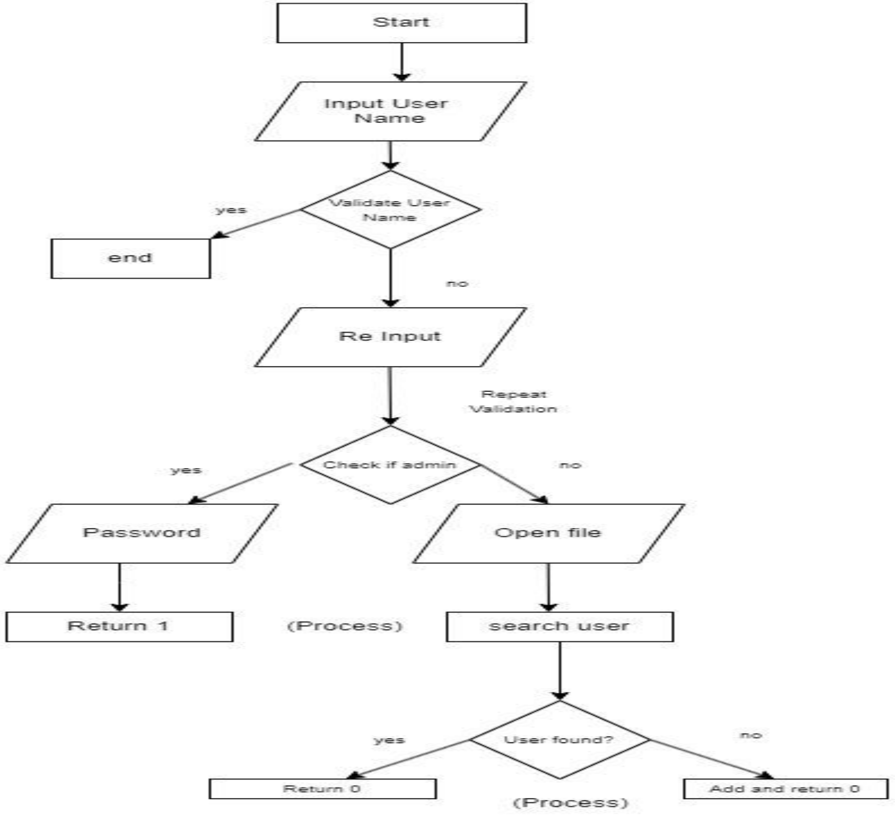
Table of Contents:

Introduction:

- Connect 4 also known as Connect four is a two player game where each player chooses a color and drops said colored circle tokens into a grid. This grid is often six by four by sometimes bigger or smaller.
- A player wins the game by getting four of their colored tokens in a row either, horizontally, vertically or diagonally. A player can stop another player from winning by blocking them with their own colored tokens from getting four in a row.

Development Summary:

- Assigned group members to groups A or B
  - Gave each group tasks
  - Gave members responsibilities
  - Assigned meeting days and times
- Developed Classic Board for Connect 4 (6 x 7 board)
- Implemented Admin and Player front
- Implemented Binary
- Conversion to Java
- Java to HTML



Binary code flowchart

Major Variables(UML):

BinaryInterface
userFile string currUser User userSpot int
findUser(fstream &, string) int addUser() void BinaryInterface() login() int updateRecord(int **, int) void adminMenu(void;

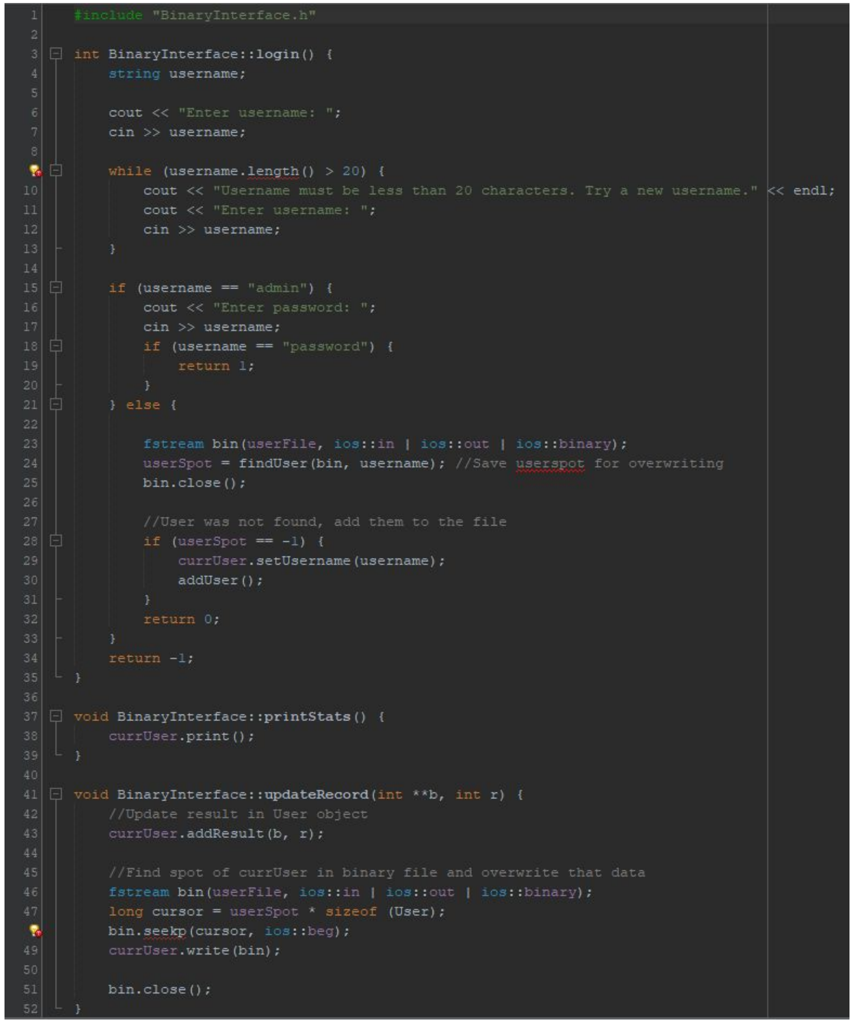
Board
row int col int game **int
*A structure that is used to help with building the Game class

Major Variables(UML):

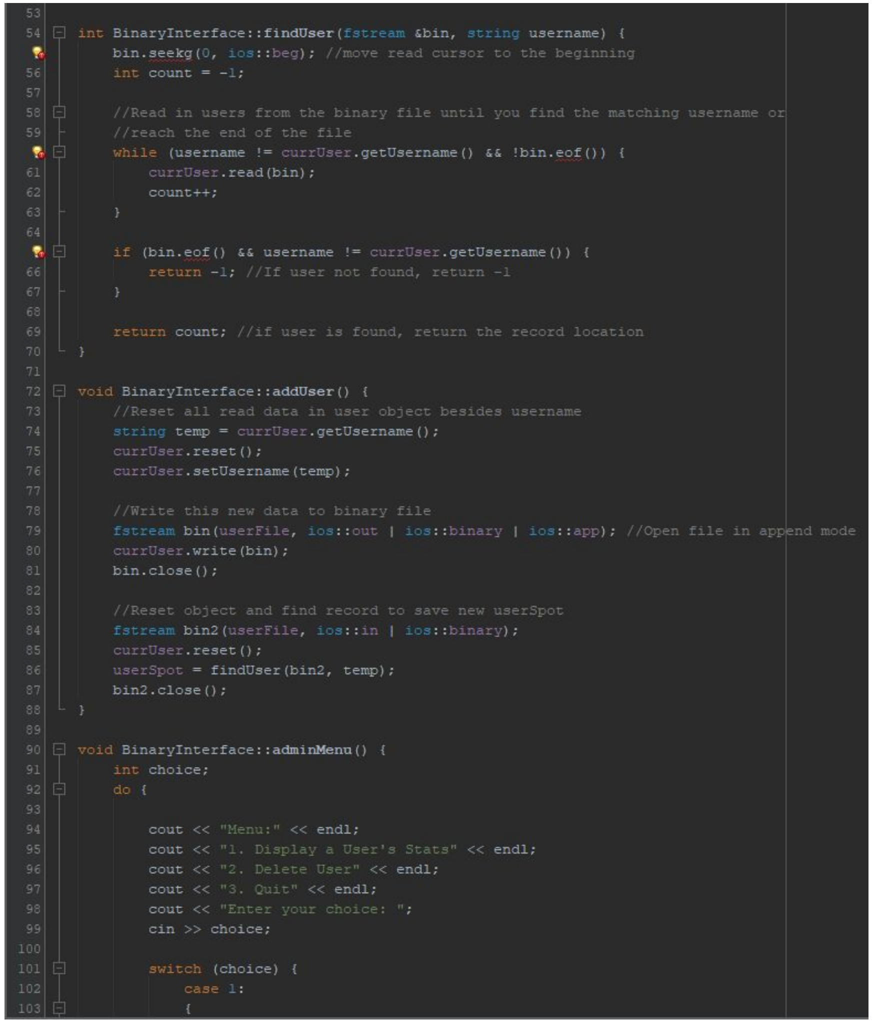
Game
gameBoard *BOARD data BinaryInterface gOver bool player, comp, pMove int
Game(BinaryInterface) ~Game() display() void plaMove(int *, int) int AIMove(int *) int upDown(int, int, int) int downUp(int, int, int) int win(int) bool playerVsPlayer() void playerVsComputer() void printStats() void menu() void reset() void

User
username[20] char gameResult[3] int winLoss[3] int
User() reset() void setUsername(string) void addResult(int **, int) void print() void write(fstream &bin) void read(fstream &bin) void

BinaryInterface.cpp

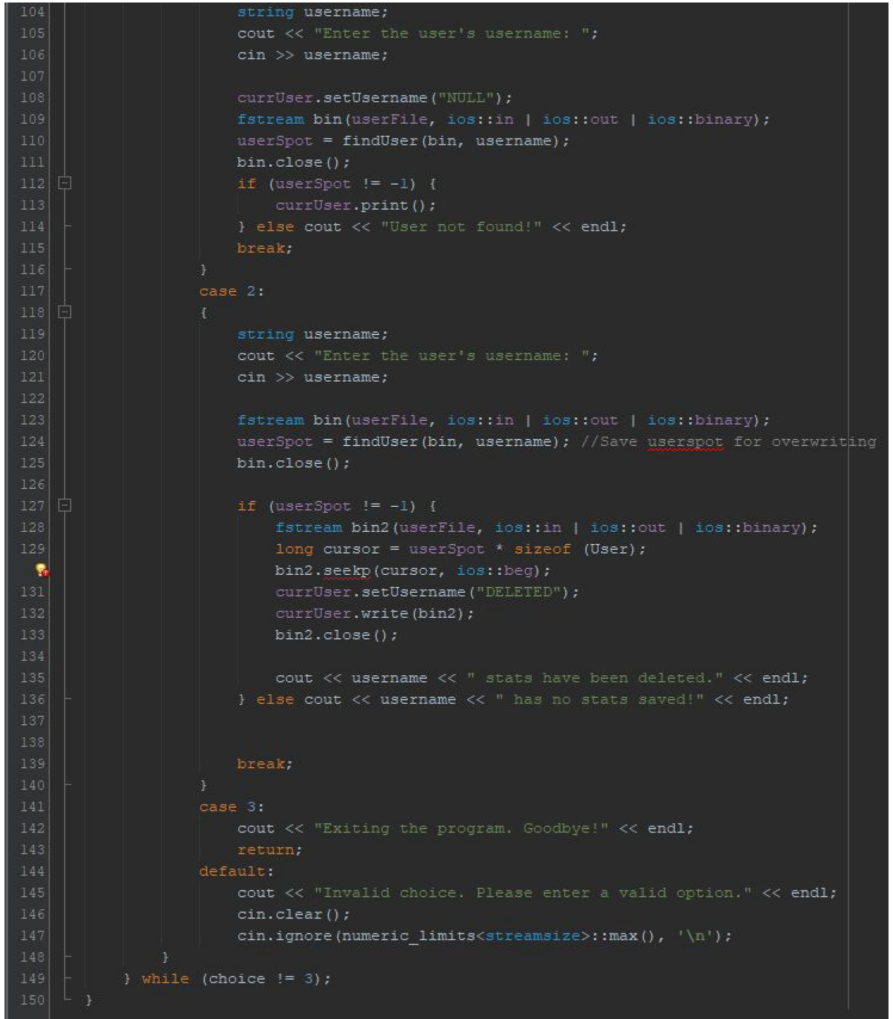


```cpp
#include "BinaryInterface.h"

int BinaryInterface::login() {
    string username;

    cout << "Enter username: ";
    cin >> username;

    while (username.length() > 20) {
        cout << "Username must be less than 20 characters. Try a new username." << endl;
        cout << "Enter username: ";
        cin >> username;
    }

    if (username == "admin") {
        cout << "Enter password: ";
        cin >> username;
        if (username == "password") {
            return 1;
        }
    } else {

        fstream bin(userFile, ios::in | ios::out | ios::binary);
        userSpot = findUser(bin, username); //Save userspot for overwriting
        bin.close();

        //User was not found, add them to the file
        if (userSpot == -1) {
            currUser.setUsername(username);
            addUser();
        }
        return 0;
    }
    return -1;
}

void BinaryInterface::printStats() {
    currUser.print();
}

void BinaryInterface::updateRecord(int **b, int r) {
    //Update result in User object
    currUser.addResult(b, r);

    //Find spot of currUser in binary file and overwrite that data
    fstream bin(userFile, ios::in | ios::out | ios::binary);
    long cursor = userSpot * sizeof (User);
    bin.seekp(cursor, ios::beg);
    currUser.write(bin);

    bin.close();
}
```



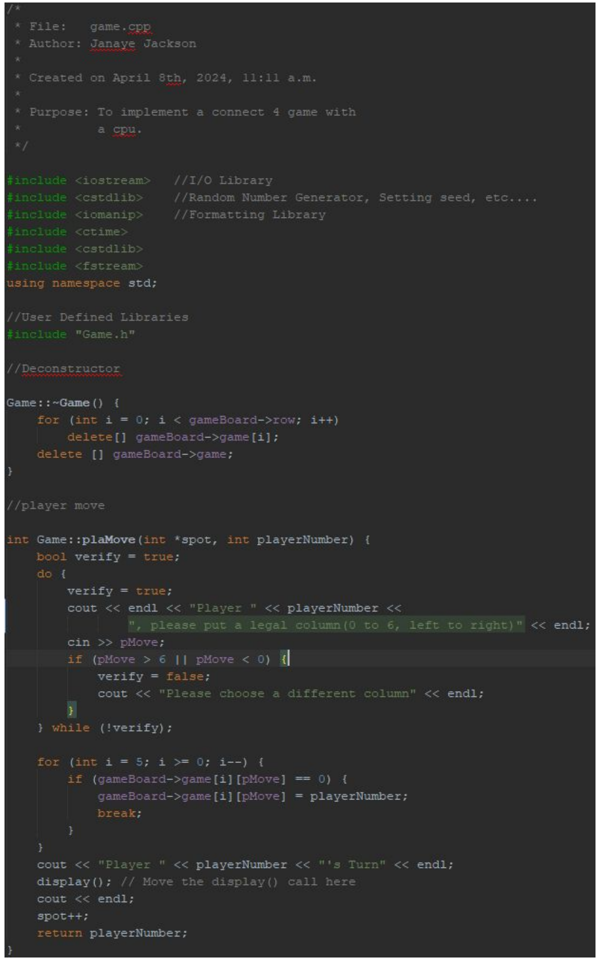
```cpp

int BinaryInterface::findUser(fstream &bin, string username) {
    bin.seekg(0, ios::beg); //move read cursor to the beginning
    int count = -1;

    //Read in users from the binary file until you find the matching username or
    //reach the end of the file
    while (username != currUser.getUsername() && !bin.eof()) {
        currUser.read(bin);
        count++;
    }

    if (bin.eof() && username != currUser.getUsername()) {
        return -1; //If user not found, return -1
    }

    return count; //if user is found, return the record location
}

void BinaryInterface::addUser() {
    //Reset all read data in user object besides username
    string temp = currUser.getUsername();
    currUser.reset();
    currUser.setUsername(temp);

    //Write this new data to binary file
    fstream bin(userFile, ios::out | ios::binary | ios::app); //Open file in append mode
    currUser.write(bin);
    bin.close();

    //Reset object and find record to save new userSpot
    fstream bin2(userFile, ios::in | ios::binary);
    currUser.reset();
    userSpot = findUser(bin2, temp);
    bin2.close();
}

void BinaryInterface::adminMenu() {
    int choice;
    do {

        cout << "Menu:" << endl;
        cout << "1. Display a User's Stats" << endl;
        cout << "2. Delete User" << endl;
        cout << "3. Quit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
            {
```

BinaryInterface.cpp continued

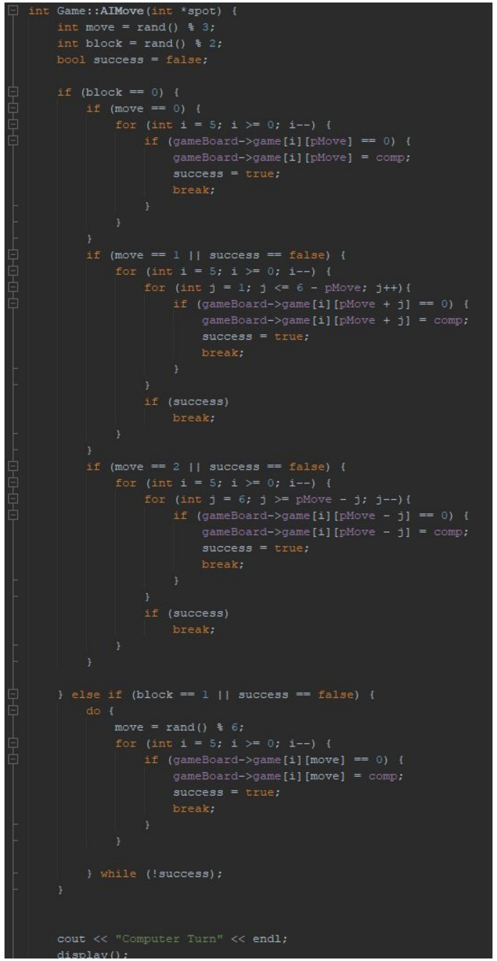


```cpp
                string username;
                cout << "Enter the user's username: ";
                cin >> username;

                currUser.setUsername("NULL");
                fstream bin(userFile, ios::in | ios::out | ios::binary);
                userSpot = findUser(bin, username);
                bin.close();
                if (userSpot != -1) {
                    currUser.print();
                } else cout << "User not found!" << endl;
                break;
            }
            case 2:
            {
                string username;
                cout << "Enter the user's username: ";
                cin >> username;

                fstream bin(userFile, ios::in | ios::out | ios::binary);
                userSpot = findUser(bin, username); //Save userspot for overwriting
                bin.close();

                if (userSpot != -1) {
                    fstream bin2(userFile, ios::in | ios::out | ios::binary);
                    long cursor = userSpot * sizeof (User);
                    bin2.seekp(cursor, ios::beg);
                    currUser.setUsername("DELETED");
                    currUser.write(bin2);
                    bin2.close();

                    cout << username << " stats have been deleted." << endl;
                } else cout << username << " has no stats saved!" << endl;


                break;
            }
            case 3:
                cout << "Exiting the program. Goodbye!" << endl;
                return;
            default:
                cout << "Invalid choice. Please enter a valid option." << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
    } while (choice != 3);
}
```

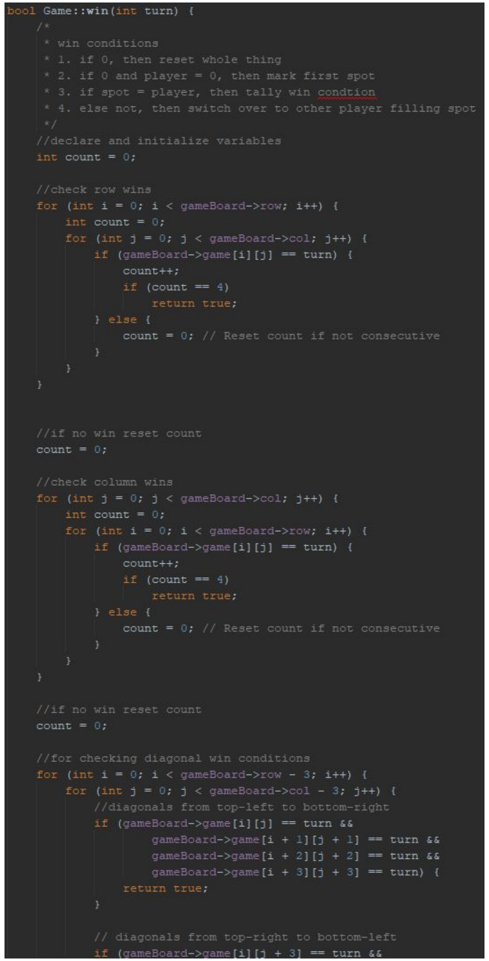
Game.cpp



```cpp
/*
 * File:   game.cpp
 * Author: Janaye Jackson
 *
 * Created on April 8th, 2024, 11:11 a.m.
 *
 * Purpose: To implement a connect 4 game with
 *          a cpu.
 */

#include <iostream>   //I/O Library
#include <cstdlib>    //Random Number Generator, Setting seed, etc....
#include <iomanip>    //Formatting Library
#include <ctime>
#include <cstdlib>
#include <fstream>
using namespace std;

//User Defined Libraries
#include "Game.h"

//Deconstructor

Game::~Game() {
    for (int i = 0; i < gameBoard->row; i++)
        delete[] gameBoard->game[i];
    delete [] gameBoard->game;
}

//player move

int Game::plaMove(int *spot, int playerNumber) {
    bool verify = true;
    do {
        verify = true;
        cout << endl << "Player " << playerNumber <<
                ", please put a legal column(0 to 6, left to right)" << endl;
        cin >> pMove;
        if (pMove > 6 || pMove < 0) {
            verify = false;
            cout << "Please choose a different column" << endl;
        }
    } while (!verify);

    for (int i = 5; i >= 0; i--) {
        if (gameBoard->game[i][pMove] == 0) {
            gameBoard->game[i][pMove] = playerNumber;
            break;
        }
    }
    cout << "Player " << playerNumber << "'s Turn" << endl;
    display(); // Move the display() call here
    cout << endl;
    spot++;
    return playerNumber;
}
```

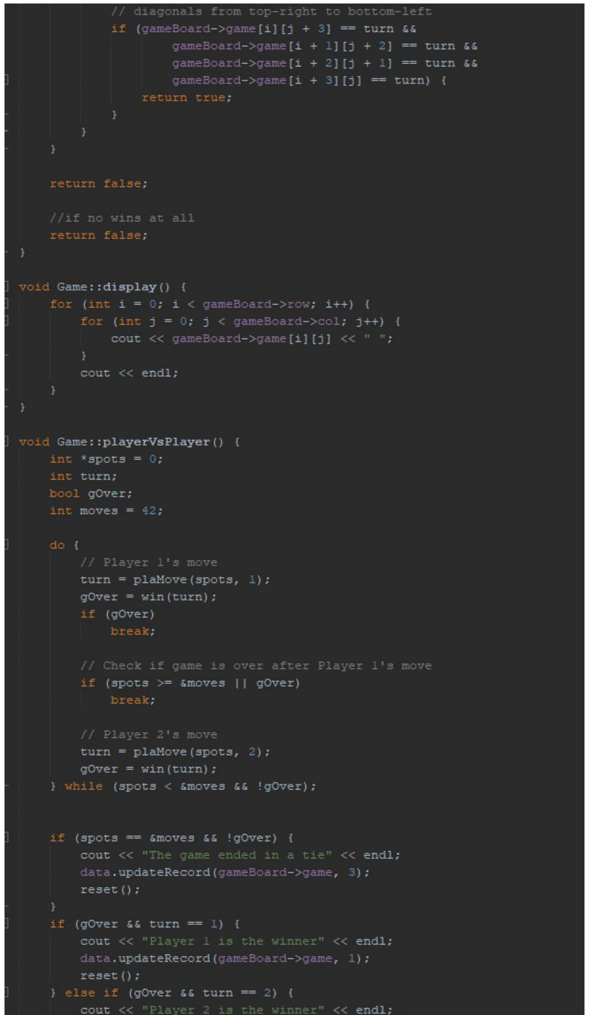


```cpp
int Game::AIMove(int *spot) {
    int move = rand() % 3;
    int block = rand() % 2;
    bool success = false;

    if (block == 0) {
        if (move == 0) {
            for (int i = 5; i >= 0; i--) {
                if (gameBoard->game[i][pMove] == 0) {
                    gameBoard->game[i][pMove] = comp;
                    success = true;
                    break;
                }
            }
        }
        if (move == 1 || success == false) {
            for (int i = 5; i >= 0; i--) {
                for (int j = 1; j <= 6 - pMove; j++){
                    if (gameBoard->game[i][pMove + j] == 0) {
                        gameBoard->game[i][pMove + j] = comp;
                        success = true;
                        break;
                    }
                }
                if (success)
                    break;
            }
        }
        if (move == 2 || success == false) {
            for (int i = 5; i >= 0; i--) {
                for (int j = 6; j >= pMove - j; j--){
                    if (gameBoard->game[i][pMove - j] == 0) {
                        gameBoard->game[i][pMove - j] = comp;
                        success = true;
                        break;
                    }
                }
                if (success)
                    break;
            }
        }

    } else if (block == 1 || success == false) {
        do {
            move = rand() % 6;
            for (int i = 5; i >= 0; i--) {
                if (gameBoard->game[i][move] == 0) {
                    gameBoard->game[i][move] = comp;
                    success = true;
                    break;
                }
            }

        } while (!success);
    }


    cout << "Computer Turn" << endl;
    display();
```

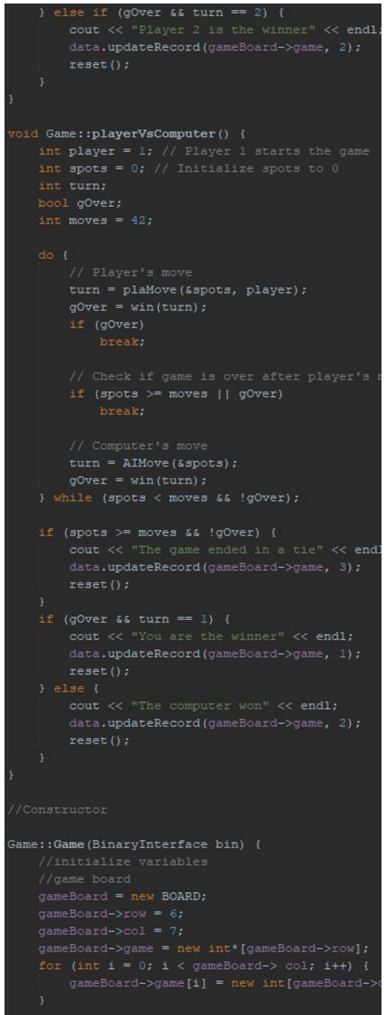


```cpp
bool Game::win(int turn) {
    /*
     * win conditions
     * 1. if 0, then reset whole thing
     * 2. if 0 and player = 0, then mark first spot
     * 3. if spot = player, then tally win condtion
     * 4. else not, then switch over to other player filling spot
     */
    //declare and initialize variables
    int count = 0;

    //check row wins
    for (int i = 0; i < gameBoard->row; i++) {
        int count = 0;
        for (int j = 0; j < gameBoard->col; j++) {
            if (gameBoard->game[i][j] == turn) {
                count++;
                if (count == 4)
                    return true;
            } else {
                count = 0; // Reset count if not consecutive
            }
        }
    }


    //if no win reset count
    count = 0;

    //check column wins
    for (int j = 0; j < gameBoard->col; j++) {
        int count = 0;
        for (int i = 0; i < gameBoard->row; i++) {
            if (gameBoard->game[i][j] == turn) {
                count++;
                if (count == 4)
                    return true;
            } else {
                count = 0; // Reset count if not consecutive
            }
        }
    }

    //if no win reset count
    count = 0;

    //for checking diagonal win conditions
    for (int i = 0; i < gameBoard->row - 3; i++) {
        for (int j = 0; j < gameBoard->col - 3; j++) {
            //diagonals from top-left to bottom-right
            if (gameBoard->game[i][j] == turn &&
                    gameBoard->game[i + 1][j + 1] == turn &&
                    gameBoard->game[i + 2][j + 2] == turn &&
                    gameBoard->game[i + 3][j + 3] == turn) {
                return true;
            }

            // diagonals from top-right to bottom-left
            if (gameBoard->game[i][j + 3] == turn &&
```

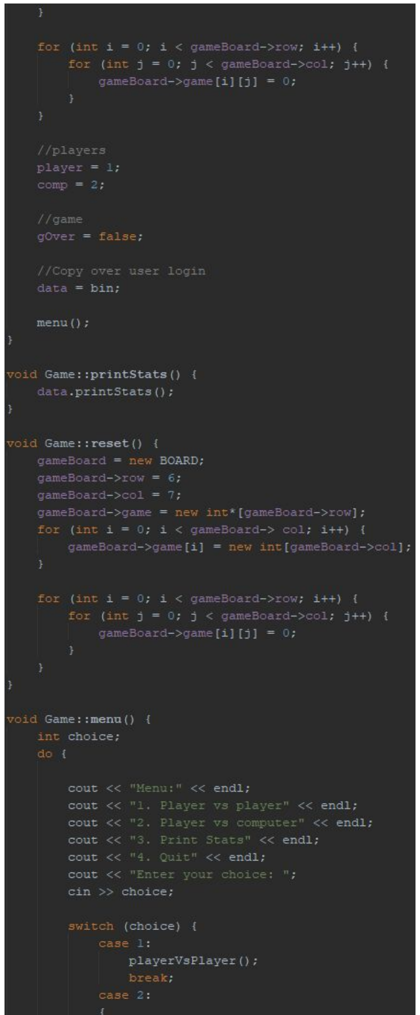
Game.cpp continued



```cpp
            // diagonals from top-right to bottom-left
            if (gameBoard->game[i][j + 3] == turn &&
                    gameBoard->game[i + 1][j + 2] == turn &&
                    gameBoard->game[i + 2][j + 1] == turn &&
                    gameBoard->game[i + 3][j] == turn) {
                return true;
            }
        }
    }

    return false;

    //if no wins at all
    return false;
}

void Game::display() {
    for (int i = 0; i < gameBoard->row; i++) {
        for (int j = 0; j < gameBoard->col; j++) {
            cout << gameBoard->game[i][j] << " ";
        }
        cout << endl;
    }
}

void Game::playerVsPlayer() {
    int *spots = 0;
    int turn;
    bool gOver;
    int moves = 42;

    do {
        // Player 1's move
        turn = plaMove(spots, 1);
        gOver = win(turn);
        if (gOver)
            break;

        // Check if game is over after Player 1's move
        if (spots >= &moves || gOver)
            break;

        // Player 2's move
        turn = plaMove(spots, 2);
        gOver = win(turn);
    } while (spots < &moves && !gOver);


    if (spots == &moves && !gOver) {
        cout << "The game ended in a tie" << endl;
        data.updateRecord(gameBoard->game, 3);
        reset();
    }
    if (gOver && turn == 1) {
        cout << "Player 1 is the winner" << endl;
        data.updateRecord(gameBoard->game, 1);
        reset();
    } else if (gOver && turn == 2) {
        cout << "Player 2 is the winner" << endl;
```

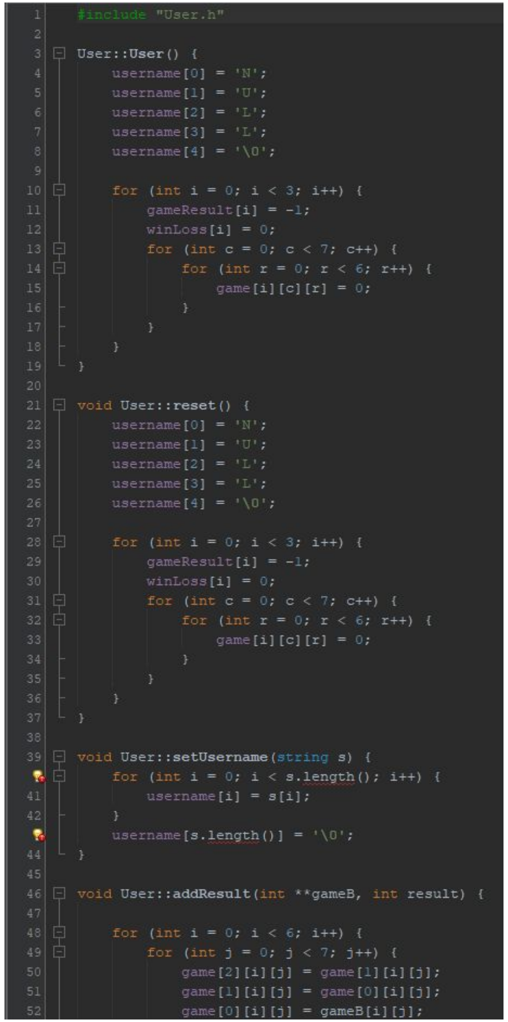


```cpp
    } else if (gOver && turn == 2) {
        cout << "Player 2 is the winner" << endl;
        data.updateRecord(gameBoard->game, 2);
        reset();
    }
}

void Game::playerVsComputer() {
    int player = 1; // Player 1 starts the game
    int spots = 0; // Initialize spots to 0
    int turn;
    bool gOver;
    int moves = 42;

    do {
        // Player's move
        turn = plaMove(&spots, player);
        gOver = win(turn);
        if (gOver)
            break;

        // Check if game is over after player's m
        if (spots >= moves || gOver)
            break;

        // Computer's move
        turn = AIMove(&spots);
        gOver = win(turn);
    } while (spots < moves && !gOver);

    if (spots >= moves && !gOver) {
        cout << "The game ended in a tie" << endl
        data.updateRecord(gameBoard->game, 3);
        reset();
    }
    if (gOver && turn == 1) {
        cout << "You are the winner" << endl;
        data.updateRecord(gameBoard->game, 1);
        reset();
    } else {
        cout << "The computer won" << endl;
        data.updateRecord(gameBoard->game, 2);
        reset();
    }
}

//Constructor

Game::Game(BinaryInterface bin) {
    //initialize variables
    //game board
    gameBoard = new BOARD;
    gameBoard->row = 6;
    gameBoard->col = 7;
    gameBoard->game = new int*[gameBoard->row];
    for (int i = 0; i < gameBoard-> col; i++) {
        gameBoard->game[i] = new int[gameBoard->c
    }
```

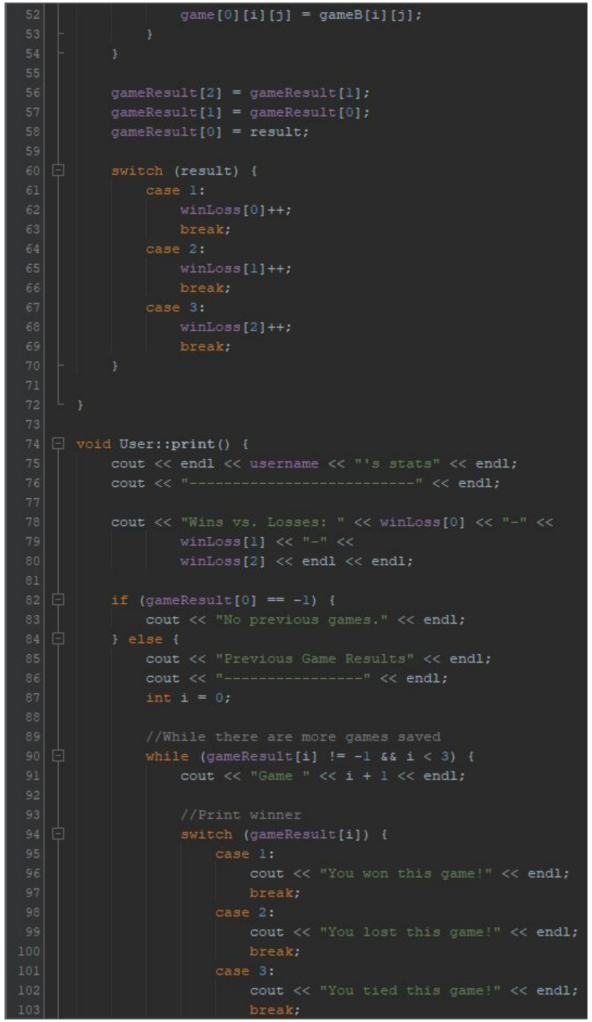


```cpp
    }

    for (int i = 0; i < gameBoard->row; i++) {
        for (int j = 0; j < gameBoard->col; j++) {
            gameBoard->game[i][j] = 0;
        }
    }

    //players
    player = 1;
    comp = 2;

    //game
    gOver = false;

    //Copy over user login
    data = bin;

    menu();
}

void Game::printStats() {
    data.printStats();
}

void Game::reset() {
    gameBoard = new BOARD;
    gameBoard->row = 6;
    gameBoard->col = 7;
    gameBoard->game = new int*[gameBoard->row];
    for (int i = 0; i < gameBoard-> col; i++) {
        gameBoard->game[i] = new int[gameBoard->col];
    }

    for (int i = 0; i < gameBoard->row; i++) {
        for (int j = 0; j < gameBoard->col; j++) {
            gameBoard->game[i][j] = 0;
        }
    }
}

void Game::menu() {
    int choice;
    do {

        cout << "Menu:" << endl;
        cout << "1. Player vs player" << endl;
        cout << "2. Player vs computer" << endl;
        cout << "3. Print Stats" << endl;
        cout << "4. Quit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                playerVsPlayer();
                break;
            case 2:
            {
```

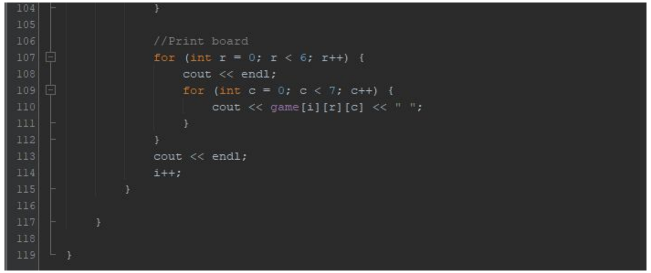
User.cpp



```cpp
#include "User.h"

User::User() {
    username[0] = 'N';
    username[1] = 'U';
    username[2] = 'L';
    username[3] = 'L';
    username[4] = '\0';

    for (int i = 0; i < 3; i++) {
        gameResult[i] = -1;
        winLoss[i] = 0;
        for (int c = 0; c < 7; c++) {
            for (int r = 0; r < 6; r++) {
                game[i][c][r] = 0;
            }
        }
    }
}

void User::reset() {
    username[0] = 'N';
    username[1] = 'U';
    username[2] = 'L';
    username[3] = 'L';
    username[4] = '\0';

    for (int i = 0; i < 3; i++) {
        gameResult[i] = -1;
        winLoss[i] = 0;
        for (int c = 0; c < 7; c++) {
            for (int r = 0; r < 6; r++) {
                game[i][c][r] = 0;
            }
        }
    }
}

void User::setUsername(string s) {
    for (int i = 0; i < s.length(); i++) {
        username[i] = s[i];
    }
    username[s.length()] = '\0';
}

void User::addResult(int **gameB, int result) {

    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 7; j++) {
            game[2][i][j] = game[1][i][j];
            game[1][i][j] = game[0][i][j];
            game[0][i][j] = gameB[i][j];
```

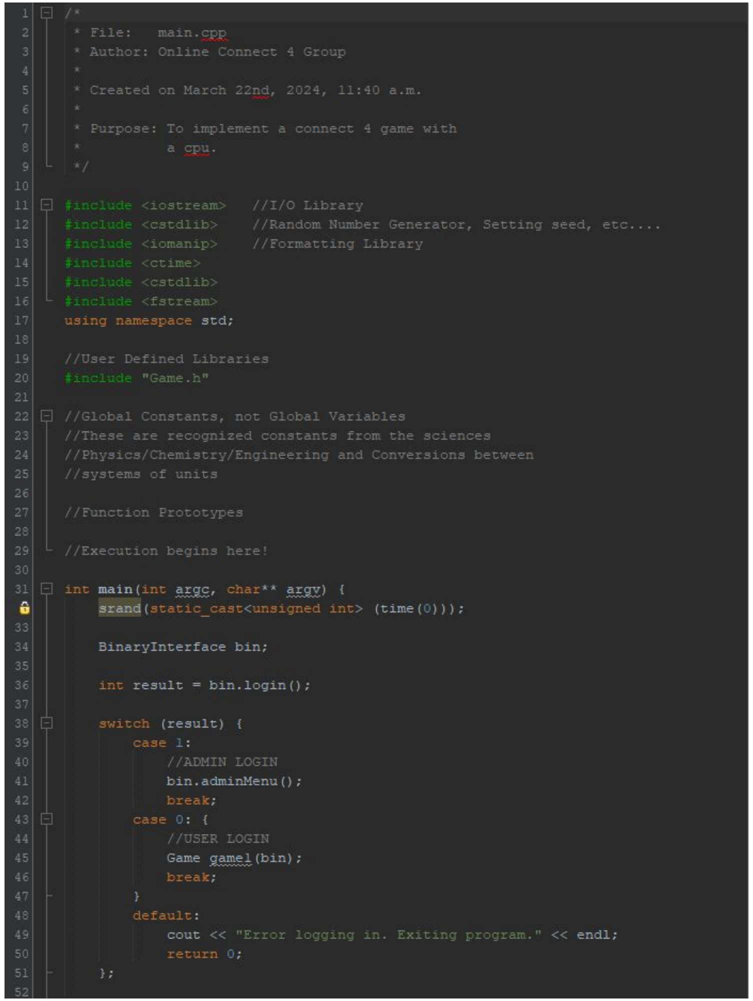


```cpp
            game[0][i][j] = gameB[i][j];
        }
    }

    gameResult[2] = gameResult[1];
    gameResult[1] = gameResult[0];
    gameResult[0] = result;

    switch (result) {
        case 1:
            winLoss[0]++;
            break;
        case 2:
            winLoss[1]++;
            break;
        case 3:
            winLoss[2]++;
            break;
    }

}

void User::print() {
    cout << endl << username << "'s stats" << endl;
    cout << "---------------------------" << endl;

    cout << "Wins vs. Losses: " << winLoss[0] << "-" <<
            winLoss[1] << "-" <<
            winLoss[2] << endl << endl;

    if (gameResult[0] == -1) {
        cout << "No previous games." << endl;
    } else {
        cout << "Previous Game Results" << endl;
        cout << "-----------------" << endl;
        int i = 0;

        //While there are more games saved
        while (gameResult[i] != -1 && i < 3) {
            cout << "Game " << i + 1 << endl;

            //Print winner
            switch (gameResult[i]) {
                case 1:
                    cout << "You won this game!" << endl;
                    break;
                case 2:
                    cout << "You lost this game!" << endl;
                    break;
                case 3:
                    cout << "You tied this game!" << endl;
                    break;
```



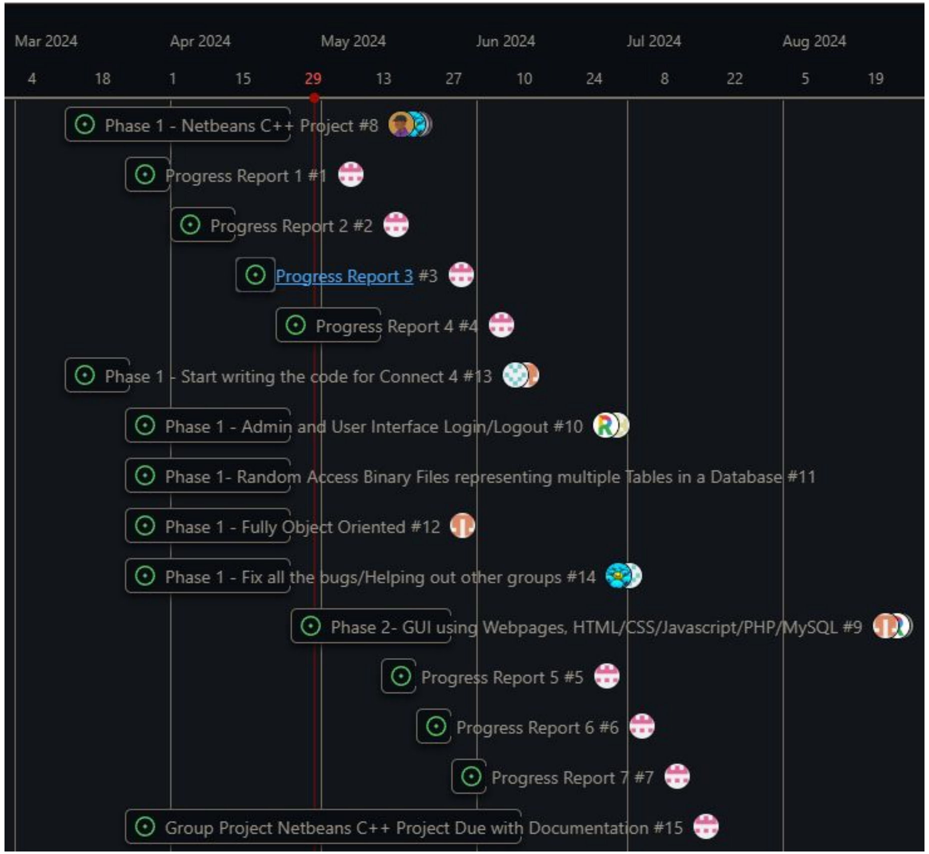
```cpp
            }

            //Print board
            for (int r = 0; r < 6; r++) {
                cout << endl;
                for (int c = 0; c < 7; c++) {
                    cout << game[i][r][c] << " ";
                }
            }
            cout << endl;
            i++;
        }

    }

}
```

main.cpp



```cpp
/*
 * File:   main.cpp
 * Author: Online Connect 4 Group
 *
 * Created on March 22nd, 2024, 11:40 a.m.
 *
 * Purpose: To implement a connect 4 game with
 *          a cpu.
 */

#include <iostream>   //I/O Library
#include <cstdlib>    //Random Number Generator, Setting seed, etc....
#include <iomanip>    //Formatting Library
#include <ctime>
#include <cstdlib>
#include <fstream>
using namespace std;

//User Defined Libraries
#include "Game.h"

//Global Constants, not Global Variables
//These are recognized constants from the sciences
//Physics/Chemistry/Engineering and Conversions between
//systems of units

//Function Prototypes

//Execution begins here!

int main(int argc, char** argv) {
    srand(static_cast<unsigned int> (time(0)));

    BinaryInterface bin;

    int result = bin.login();

    switch (result) {
        case 1:
            //ADMIN LOGIN
            bin.adminMenu();
            break;
        case 0: {
            //USER LOGIN
            Game game1(bin);
            break;
        }
        default:
            cout << "Error logging in. Exiting program." << endl;
            return 0;
    };

```

Overall progress for project:



Mar 2024 Apr 2024 May 2024 Jun 2024 Jul 2024 Aug 2024

4 18 1 15 29 13 27 10 24 8 22 5 19

Phase 1 - Netbeans C++ Project #8

Progress Report 1 #1

Progress Report 2 #2

Progress Report 3 #3

Progress Report 4 #4

Phase 1 - Start writing the code for Connect 4 #13

Phase 1 - Admin and User Interface Login/Logout #10

Phase 1- Random Access Binary Files representing multiple Tables in a Database #11

Phase 1 - Fully Object Oriented #12

Phase 1 - Fix all the bugs/Helping out other groups #14

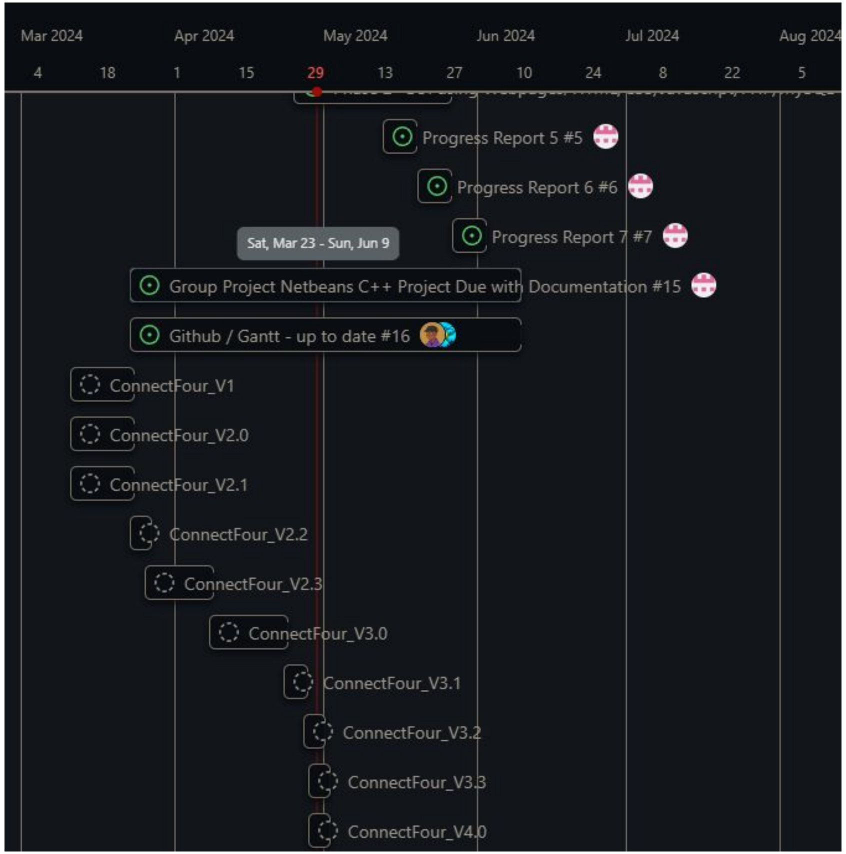
Phase 2- GUI using Webpages, HTML/CSS/Javascript/PHP/MySQL #9

Progress Report 5 #5

Progress Report 6 #6

Progress Report 7 #7

Group Project Netbeans C++ Project Due with Documentation #15



Mar 2024 Apr 2024 May 2024 Jun 2024 Jul 2024 Aug 2024

4 18 1 15 29 13 27 10 24 8 22 5

Progress Report 5 #5

Progress Report 6 #6

Sat, Mar 23 - Sun, Jun 9

Progress Report 7 #7

Group Project Netbeans C++ Project Due with Documentation #15

Github / Gantt - up to date #16

ConnectFour_V1

ConnectFour_V2.0

ConnectFour_V2.1

ConnectFour_V2.2

ConnectFour_V2.3

ConnectFour_V3.0

ConnectFour_V3.1

ConnectFour_V3.2

ConnectFour_V3.3

ConnectFour_V4.0

Useful Project & Group Information 1:

Group Name: Connect 4 Online

Meeting dates and time: 4/27/24 12:00pm

Members: Kelby Knight, Kyle Riebeling, Ryan Westfall, Amare Terrell, Aleksander Videv, Patrick Pascual, Janaye Jackson, Francisco Sanchez, Cristian Magana, Anthony Nguyen

Rafaan Hyder

Group A: Kyle R., Ryan W., Patrick P., Janaye J., Rafaan H.

Group B: Kelby K., Amare T., Aleksander V., Francisco S., Cristian M., Anthony N.

Useful Project & Group Information part 2 Responsibilities:

Group Leaders(s):	Group A: Patrick P. Group B: Kelby K.
Group A Tasks:	● AI Functions, Debug base game, Multiplayer functions, Classification, Admin/User Interface Login/Logout, Develop Classic Board for CON4
Group B Tasks:	● Binary Files, Random Access Binary files, Represent multiple Tables in database, Convert to Java, and Java to HTML, Documentation

Useful Project & Group Information part 3
Work Schedule & Links:

Meetings:	● Online (Discord) ● Thursday and Saturday @ 5:40PM
GitHub Repository:	● https://github.com/4mxr3/Connect4ON