

# Muhammad Saad

## Saad Reprot

-  Quick Submit
-  Quick Submit
-  National University of Computer and Emerging Sciences, Islamabad

### Document Details

**Submission ID****trn:oid:::1:3418185770****9 Pages****Submission Date****Nov 20, 2025, 8:20 AM GMT+5****789 Words****Download Date****Nov 20, 2025, 8:24 AM GMT+5****4,369 Characters****File Name****connectFour\_PF\_Lab\_-\_Final\_Project\_Report.docx****File Size****1.7 MB**

# 0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

## Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

## Detection Groups

### 0 AI-generated only 0%

Likely AI-generated text from a large-language model.

### 0 AI-generated text that was AI-paraphrased 0%

Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

## Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

## Frequently Asked Questions

### How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

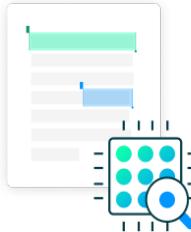
AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (\*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

### What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.



A very advanced, console-based implementation of the classic 'Connect Four' game.

# connectFour+

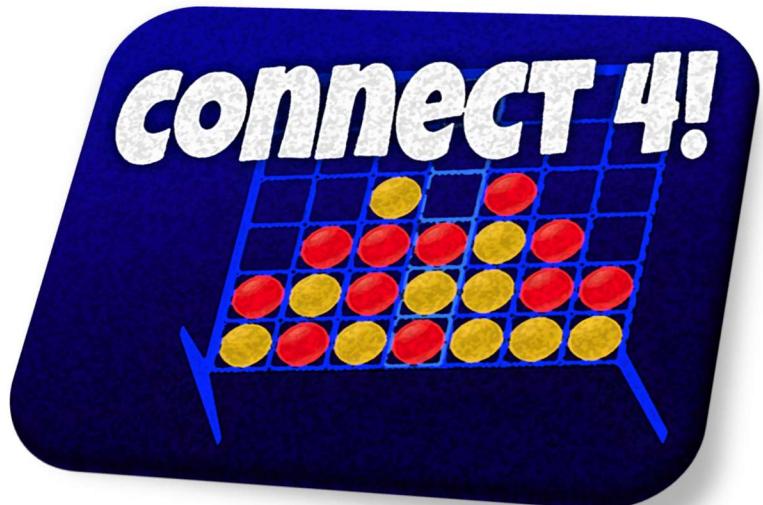
## PF Lab – Final Project Report

Muhammad Saad (25K-0604)

# FINAL PROJECT REPORT

*PROGRAMMING FUNDAMENTALS*

**University Name:** FAST Nuces KHI  
**Department:** Department of Computer Science  
**Course:** Programming Fundamentals  
**Project Title:** connectFour+  
**Submitted By:** Muhammad Saad (25K-0604)  
**Submitted To:** Sir Sheeraz Iqbal  
**Semester:** Fall 2025  
**Date:** 21 November, 2025



## CONTENTS

<b>Final Project Report.....</b>	1
<b>1. Introduction.....</b>	3
<b>2. Objectives.....</b>	4
<b>3. System Design.....</b>	5
<b>System Overview/Flow.....</b>	5
<b>4. Implementation.....</b>	6
<b>Key Features.....</b>	6
<b>Code Snippet.....</b>	6
<b>5. Conclusion, Limitations &amp; References.....</b>	8
<b>Conclusion.....</b>	8
<b>Limitations.....</b>	8
<b>Future Enhancements.....</b>	8
<b>References .....</b>	8

## 1. INTRODUCTION

`connectFour+` is an advanced console-based version of the famous “Connect Four” game. The game consists of a grid-like gameboard with openings on the top of each column. Players take turns to drop their pieces (red or blue usually) into one of the available columns. The objective of the game is to make a consecutive row of 4 with only 1 types of piece; this could be horizontally, vertically, or diagonally. Whichever player manages to perform this, wins.

This implementation of the game consists of various `matchModes`, `gameModes`, gameboard-sizes, players, persistence, `gameHistory`, `leaderBoards`, an about section, as well as many other features. All of these options and features are neatly arranged in menus which could easily be accessed and used by the user.

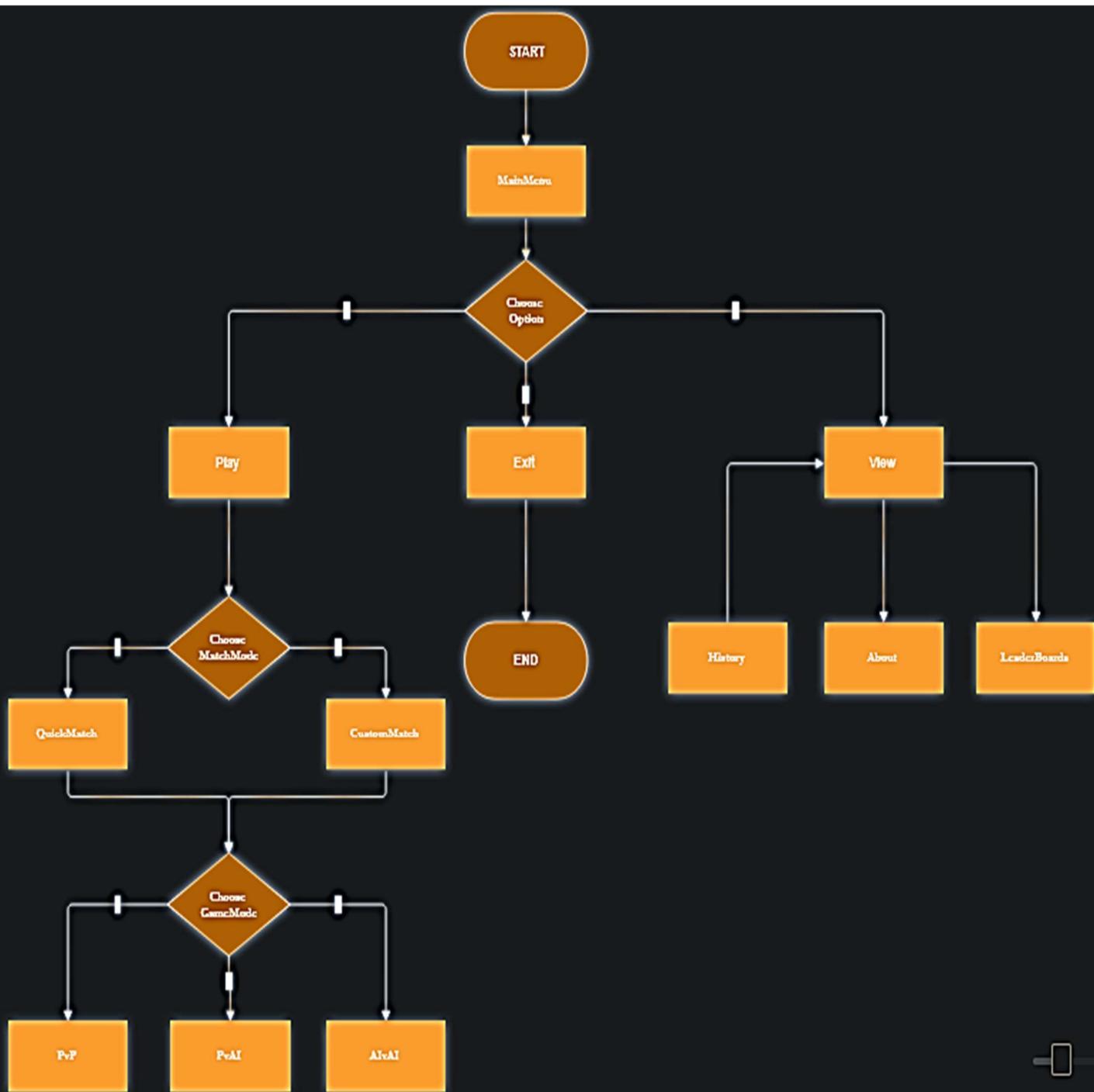
The project took almost a full month to make (almost the complete month of October, 2025) and consists of 2600-3200 loc (range due to different measures of counting).

## 2. OBJECTIVES

- 1) To allow users to play the connectFour game
- 2) To provide the user with a menu with 3 options: Play, View, & Exit
- 3) To provide the user with 3 gameModes: PvP, PvAI, AIvAI
- 4) To allow the user to configure the settings (like gameboard size and the chosen AI)
- 5) To animate each move played
- 6) To display the results of the game in the end
- 7) To allow the user to see the complete game history, and to replay each individual game later on
- 8) To track the rankings of each player and allow the user to see the leaderBoards in a tabulized manner
- 9) To allow the user to get help about the game (both brief and detailed)
- 10) To allow the user to exit the program whenever he/she wishes to do so

### 3. SYSTEM DESIGN

#### SYSTEM OVERVIEW/FLOW



## 4. IMPLEMENTATION

Language: C

Compiler/IDE: VS Code / Dev C++ / GCC

### KEY FEATURES

#### ➤ LOCs

- 2596 Lines of Pure-Code
- 3007 loc after formatting with C/C++ extension in VSCode
- +210 loc if ABOUT included

- Menu
- QuickMatch & CustomMatch options
- Multiple GameModes
- Multiple GameBoard sizes
- Animated printing of each move
- Persistence (Game History)
- Replay past games
- LeaderBoards
- Help (About)

### CODE SNIPPET

```
void PvAI() //  
player vs AI mode  
{  
    setGame(); //  
initializing game. globals  
    if (game.quickMatch) //  
quickMatch() mode  
    {  
        choosePlayer(0); //  
initially did this: strcpy(game.player1Name, "Player 1");  
        strcpy(game.player2Name, "Thalith"); //  
defaulting ai with thalith  
        game.rowCount = 6; //  
CClassic (7 x 6) gameBoard  
        game.colCount = 7;  
    }  
    else //  
customMatch() mode
```

```
{  
    setPlayers(); //  
choose players  
    setGameBoard(); //  
choose gameBoard size  
}  
if (!program.randomSeeded){ srand(time(NULL)); program.randomSeeded = true; } //  
seeding rand() with time(NULL) once per program  
startGame(); //  
printing the gameBoard & a little pause before starting game  
  
while (game.playGame) //  
infinite gameLoop until a terminal state is reached  
{  
    switchActivePlayer(); //  
switches game.activePlayer  
    if (game.activePlayer == 1) { getPlayerMove(); } //  
gets move based on game.activePlayer  
    else { getAIMove(); } //  
getting AI's move when game.activePlayer is even (2 + 2n)  
    updateGameBoard(); //  
updates the gameBoard row by row & calls printGameBoard in between them  
    updateGameState(); //  
updates game.gameState based upon the 5 checking functions  
    evaluateGameBoard(); //  
evaluating game based upon game.gameState  
    saveGameBoardDetails(); //  
updates the gameBoardHistory.txt file with the current playerMove  
}  
  
saveGameDetails();  
updateLeaderBoards(); //  
LeaderBoards will only be updated if the mode is not AIvAI (is PvP or PvAI)  
pressEnterToContinue(); //  
waits for user input before clearing the screen & reprinting mainMenu  
}
```

## 5. CONCLUSION, LIMITATIONS & REFERENCES

### CONCLUSION

Hence the report demonstrates that this implementation of the connectFour game is very good. :)

<https://github.com/The-Saad-E1/Connect-Four>

### LIMITATIONS

- 1) Lack of a GUI
- 2) Lack of an advanced AI algorithm (miniMax)
- 3) Lack of a resumeGame feature
- 4) Lack of multiplayer capability using networking
- 5) Major blinking when printing gameboard
- 6) Used global variables and functions without any parameters

### FUTURE ENHANCEMENTS

- 1) Add a GUI layer using RAYLIB & RAYGUI
- 2) Implement MiniMax & LookAhead AIs (with depth 5+)
- 3) Implement resumeGame feature using filing
- 4) Implement multiplayer feature using socket-networking (client-peer model)
- 5) Disable cursor when printing the gameboard (using \033 escape sequences)
- 6) Pass arguments to functions when called (so to properly see program-flow)

### REFERENCES

- ❖ Let Us C by Yashavant P. Kanetkar
- ❖ <https://www.programiz.com/c-programming>
- ❖ <https://www.geeksforgeeks.org/c-programming-language/>