The Ultimate Connect Four!

Final Report

Your Team's Name & Logo

Daniel Medina,

Katie Park,

Farhan Moustafa,

McKinley Martsolf,

Liya Tekie

Course: CPSC 224 - Software Development

I. Introduction and Project Description

The game that our group recreated is *Connect Four*, a board game traditionally played with two players. The objective of a traditional game of *Connect Four* is for two players, both with different colored discs, to drop their discs, in alternate turns, into the vertical grid columns provided by the game set in an attempt to have four of their colored discs in a line. A player wins once they connect four of their discs continuously, either vertically, horizontally, or diagonally, with no empty spaces or discs from their opponent, hence the name "*Connect Four*." If the board fills up with discs and no more discs can be added, the game is a draw. Although there are many different variations and house rules to this board game, such as having a bigger grid, connecting five instead of four, etc., this is the conventional way to play the game.

As for a digital version of this game, this game will be expanded to allow more than 2 players to play, with the user being given 3 different board size options. However, this project will limit the game to four players to maintain the readability of the grid size during output for users. Additionally, due to the challenge of outputting colors and mimicking the different colored discs representing players in the real-life game, this project will instead use tokens (such as O, X, etc.) to represent discs. Otherwise, this project will simulate the conventional gameplay of a traditional game of *Connect Four*, in terms of players taking turns in adding their "disks" or pieces to the grid to connect four in a row to win the game. See the Appendices for game rules.

II. Team Members - Bios and Project Roles

McKinley Martsolf is a computer science and criminology student striving to become a cybercrime investigator of any kind. His prior projects have included a Farkle game, a tank game, a Gonzaga student mental and physical health app, and building a language transformer

from scratch. McKinley's skills include C/C++, Python, Java, Balsamiq Wireframe, and Mockitt. For this project, his responsibilities were helping write code, adding and editing the slideshow, adding to the final summary report, and scheduling group meeting times.

Liya Tekie is a Computer Science major with a concentration in Cybersecurity and Economics. Her previous projects include developing a Farkle game and creating a website to raise awareness about the lack of clean water in Africa. Liya's technical skills include Python, C/C++, Java, and HTML. For this project, Liya helped with coding, designing the slides, and organizing the final results.

Daniel Medina is a computer science and business student with a concentration in Management Information Systems (MIS). He is passionate about software engineering, mobile development, machine learning, and artificial intelligence. Daniel has built a variety of projects, including a Farkle game, a Battleship game, several API-driven applications, a clothing store website, and a collaborative rowing app. His technical skills include programming languages such as C/C++, Python, Java, and JavaScript, as well as web development with HTML and data handling using libraries like Pandas.

Katie Park is a student currently majoring in Computer Science with an interest in Software Development and Machine Learning. Her prior projects include building a Farkle game, creating a program that allows users to navigate between trees on East-West streets in the Logan neighborhood using doubly linked lists, and creating a program that allows users to navigate the Columbia River (and its tributaries) using binary trees. The coding languages she is familiar with include C++, Python, and Java. For this project, she mainly contributed to coding the program and its functions, as well as helping with the project's documentation and slideshow.

Farhan Moustafa is a student-athlete currently majoring in Marketing with a minor in Software Development. He has a passion for interface design and consumer interaction. His most recent projects include a Farkle game, a restaurant website, and a social media video for a restaurant. He holds Coding abilities in C++, HTML, and Java. For this project he contributed to coding the program, interface design, report templates, and group meetings.

III. Project Requirements

The following table outlines the major features included in the game. These features range from an introductory screen and player setup options to piece selection and board size choices. Core gameplay mechanics are also implemented, including taking turns, recognizing when a player wins, and presenting a winner's screen. Additionally, players are given the option to quit the game at any point during the game.

Table 1: Major Features

Feature	*Description*
Intro Screen	The game will automatically output an intro screen with the game and programmer names. The user will also have the choice to play the game or quit the program.
Multiple Players	The game will ask the user the number of players that will play the game (2-4 players), as well as the names of each player. If a

	name is not inputted, the game will provide a default name (Player 1, Player 2, etc.).
Piece Types	The game will ask the player to choose which piece they would like to have as a representation of the "disks" in the real-life game, with the options being: "X," "O," "#," and "$."
Size of Board	The game will ask the user how large they would like the game board to be and will be given the options of small (6x7), medium (7x8), and large (8x9)
Gameplay	The game will then "play" the game *Connect Four*, in which each player takes turns inputting a column they would like their token added to. The token will be added to the bottom-most empty board space.
Recognizing a Win	The game will continuously check to see if any player has connected four of their tokens in a continuous line (either vertically, horizontally, or diagonally) after every player ends their turn.
Winner's Screen	The game will recognize which player has won the game and output a winner's screen. The user will then be given the option to quit the program or play again.

Option to Quit	At any point during gameplay, any player will have the option to end the game.

IV. Solution Approach

This project's solution design included several different components that this team thought were integral to the team's take on the game:

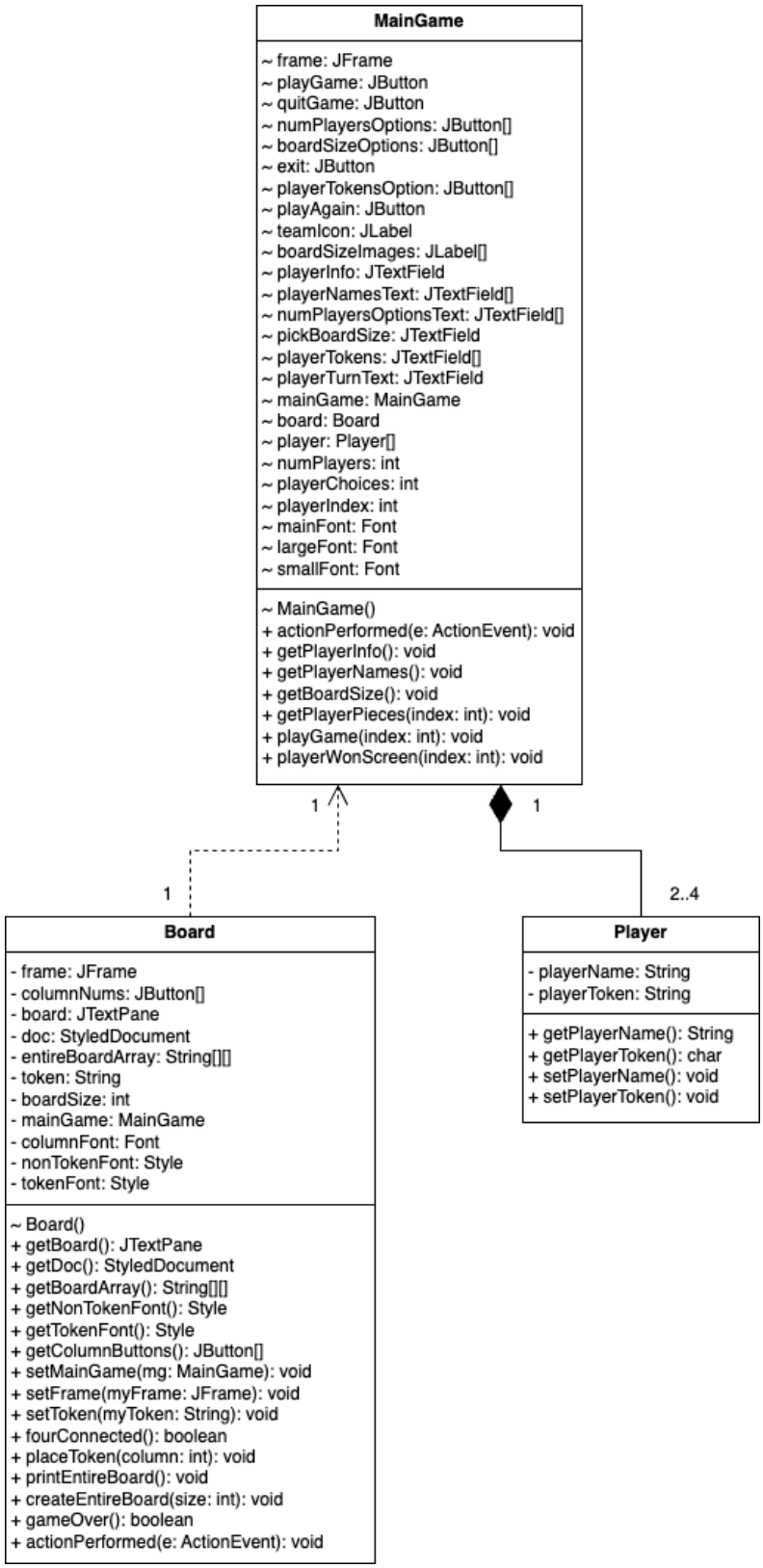
1. Multiplayer Gameplay: Automatically have the next player's turn start after a player inserts their token.
    a. Setting the number of players was done using buttons to ensure valid input.
    b. Having the next player's turn start was solved by having an array of objects from the Player class, which is iterated through whenever a player clicks a column button.
2. Player Customizability: Giving the players chances to enter a name and choose their token to represent their disk.
    a. Player name customization was addressed by having a text field that only reads from the input when the user clicks the "Done" button.
    b. Token options were implemented through buttons, and a button option disappeared after a player chose it, ensuring multiple players didn't have the same token
3. Automatic Win Recognition: Instantly detects that a player has won after the player puts their token into a column
    a. This component was solved by immediately running a method after the column

button is pressed that checks the entire board for any tokens that are four in a row.

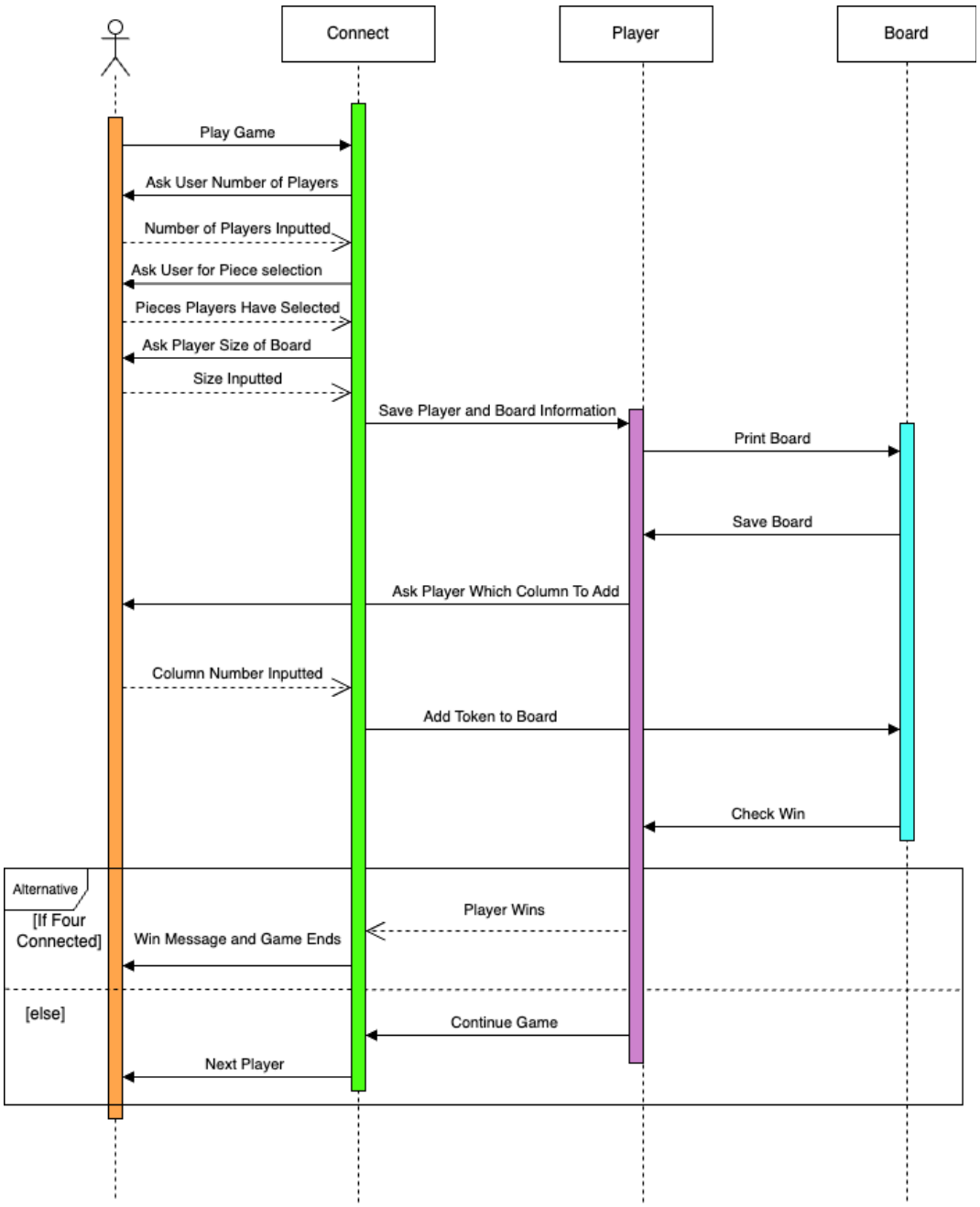
   b. Outputs a winner's screen that shows which player won and the final board.

4. Having a User-Interface: Display the game through a GUI, rather than the terminal.
   a. Done through using Java Swing components that were set to be visible at certain times, so the game has different "screens" at different times

Class Diagram UML:



Sequence Diagram UML:



V. Test Plan

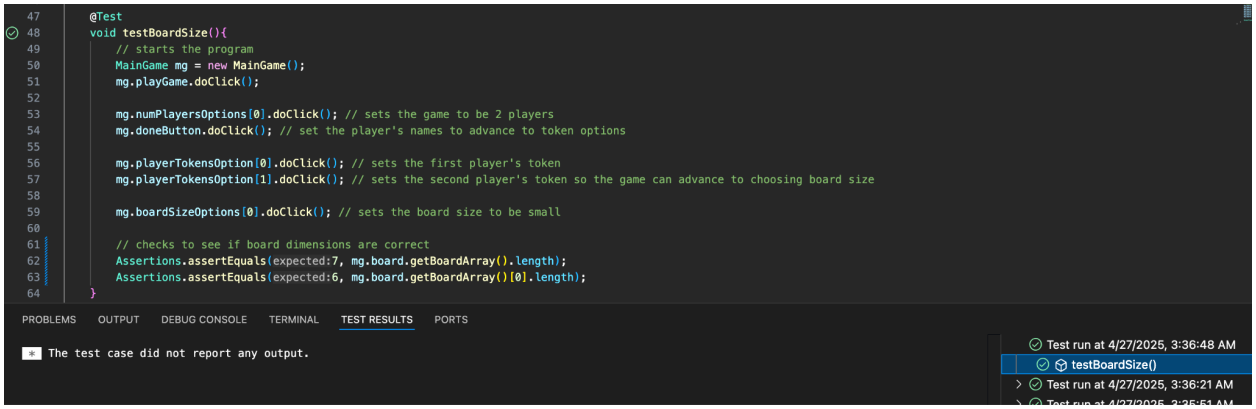
This project used unit testing, with 4 tests being implemented, and all 4 passing. These tests include:

1. Testing the player's name is set.
    a. This is done by setting 2 players' names:
        i. "GU Student"
        ii. An unnamed player to test if the game sets the player's name as "Player 2" if nothing is entered
    b. This is tested by checking to see if the player's name is fully set within the Player class

2. Testing the player's token is set for each specific player.
    a. This is done by setting the player's token for both players:
        i. Player one clicks the first button, meaning they chose "X" as a token.
        ii. Player two clicks the second button, meaning they chose "O" as a token.

b. After both players choose their token, the unit test makes sure to check the player's token is the token that is expected within the Player class.

3. Testing the board size is correct for what the user chose.
    a. This is done by setting the players' names and tokens, and then having the game advance to choosing the board size.
    b. The unit test is coded so the user chooses the first button, meaning they chose the smallest board size.
    c. After the smallest board size is chosen, the program then creates the board with the certain dimensions corresponding to that board size.
    d. This unit test checks if the correct board size is created by checking that the number of rows and columns is the same as what is expected of a small board in the Board class.

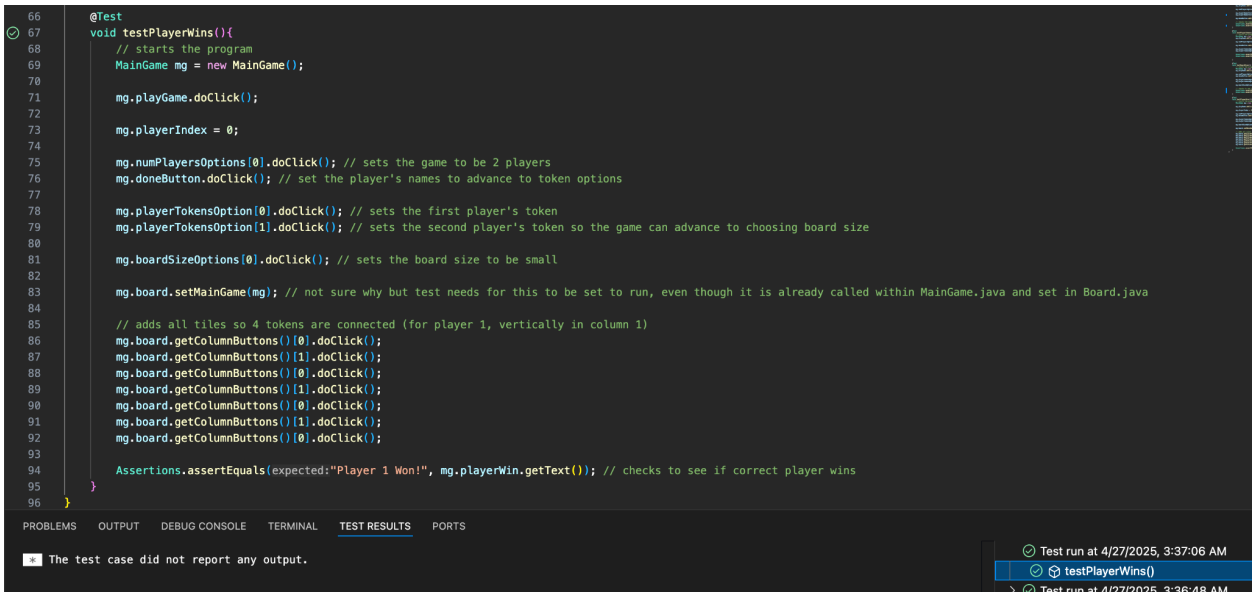


4. Testing the program correctly identifies a win, and which player has won the game.
    a. After the players' names, tokens, and board size is chosen, the unit test then clicks the buttons to make sure each player adds their token into a column.
    b. The buttons that are clicked ensure that Player 1 wins by having four tokens in a row (vertically).

c. The unit test checks if the program:

i. Recognizes a player has won, by checking if a winner's screen/banner is outputted

ii. Recognizes which player won, by checking the player's name in the winner's screen/banner is "Player 1"



VI. Project Implementation Description

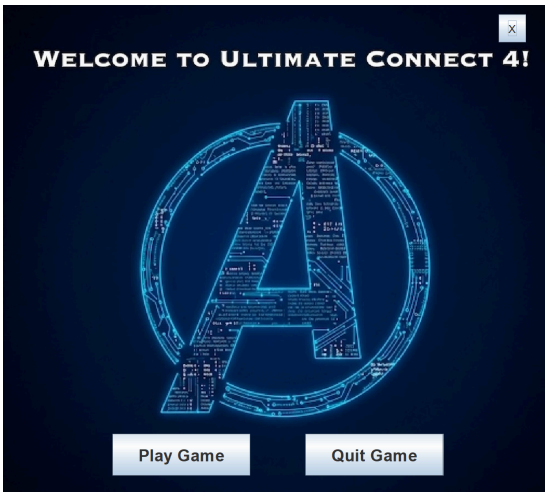
The parts of the proposed architecture that were completed in this project were mostly achieved, which include:

1. Having three different files: Connect, Board, and Player
    a. Although the game was limited to 3 files, as proposed, the "main" file that calls the other classes was renamed to MainGame
2. As the game used a UI to get all input from the user, more variables were created as a part of the constructor for all three classes, much more than what was proposed

3. Certain methods, such as getBoardColumn() from the Board class, are no longer included, and instead, new methods were created, such as gameOver() in the Board class
4. To ensure easier passing of variables between different classes, more “getter” methods were included for each class.

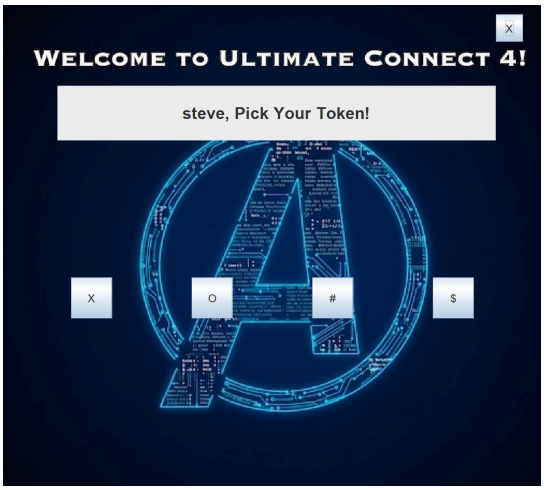
There are 6 screens for the main game that can be seen below and a test results screen that is seen in the test terminal.

1) Intro Screen / Game Start Menu

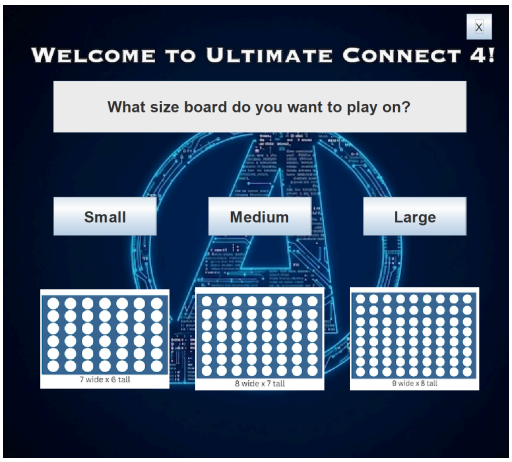


2) Number of Players and Names Screen

3) Players Picking Token Screen



4) Board Size Screen



5) The Game Screen

6) The Win Screen

7) The Test Results Screen

VI(a). Project Statistics

Overall, this project had 827 lines of code written, with the amount of code written by each individual member being:

- Katie writing half (½) of the code
- McKinley, Liya, Farhan, and Daniel writing an eighth (⅛) of the code each

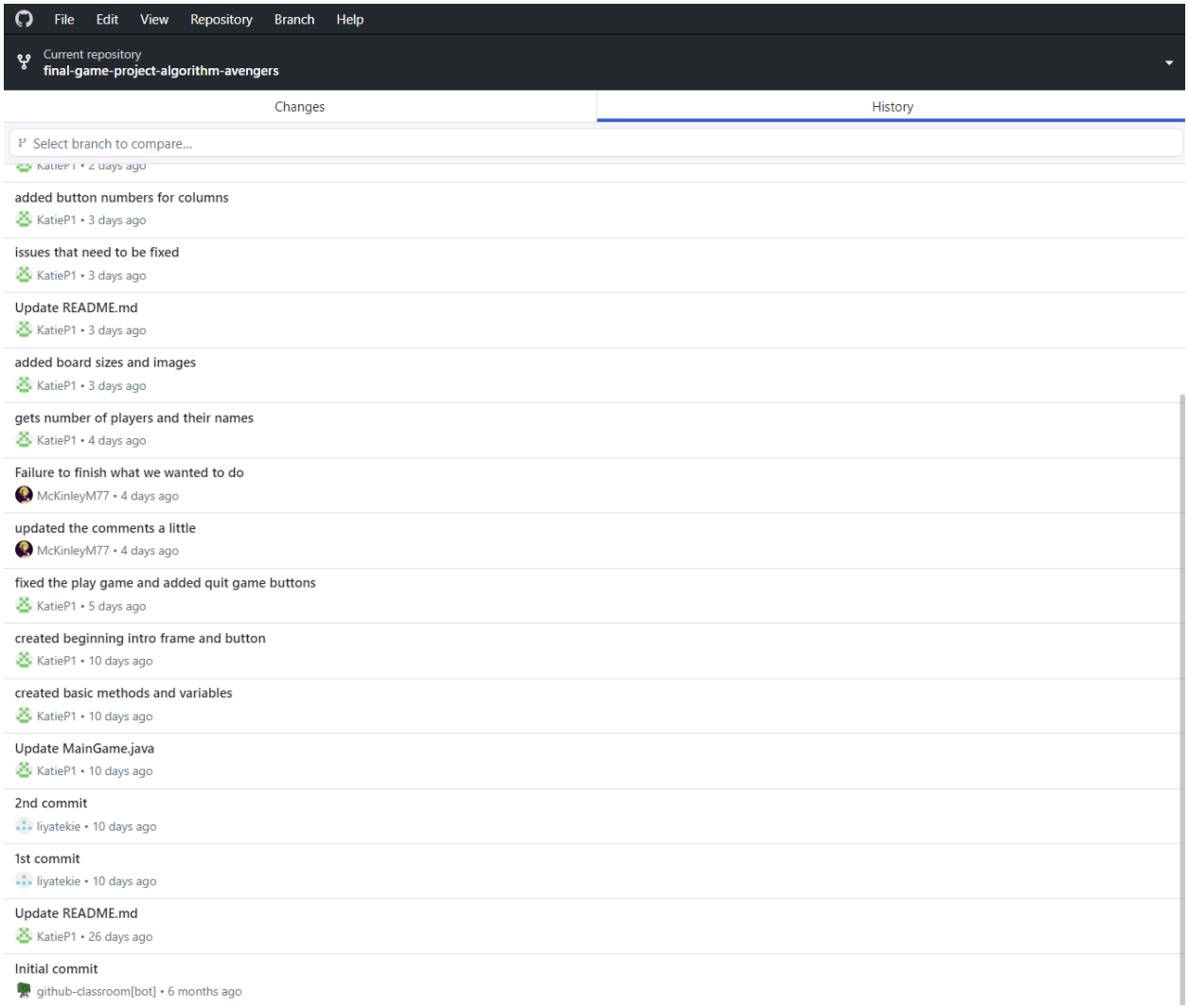
However, a lot of the code was completed during group meetings, where everyone was present and collaborated, so many lines written were a team effort. This also means that since a lot of the coding was completed during group meetings, many of the commits that were pushed were done on a single laptop/login (specifically Katie's), despite members taking turns coding. Overall, there were about 22 commits:

- 16 commits from Katie
- 3 commits from McKinley
- 3 commits from Liya

As for branching, this project did not use any, and so there were also no pull requests. Also, this project uses 4 unit tests (see section V above), which were all passed successfully. Additionally, 4 images were used in this project:

- A team logo that was set as the background image throughout the game
- Three images of board sizes so the user can visually see and choose which board size they wish to play (rather than being given the options of Small, Medium, and Large).

Gathering the images, as well as correctly formatting them, took several hours.



Git Repo Link:

https://github.com/GU-2024-Fall-CPSC224/final-game-project-algorithm-avengers

VII. Team Collaboration and Tools Used

As a team, communication was mainly done through messages, wherein group meetings were set up to fully discuss certain issues or coding. The number of group meetings initially was around 1-2 meetings a week, and as the deadline got closer, the meetings became more frequent. By the time it was the last week before the deadline, there were 5 meetings throughout the week.

Much of the code and the goals of what the code would do were done as a group as a collaborative effort, however, there were also times the code would be worked on individually, just as a way to complete any tasks that were unable to be finished during the meetings. However, the individual code was always discussed and explained during the group meetings before any other tasks were started. Additionally, as explained earlier, no Git branches or pull requests were done, and the coding was fully done on main.

The code was usually reviewed and explained by the coder to other group members if it was completed individually, and in this process, a lot was learned on how there were many ways different game elements could be implemented, as one member would write code that completed a certain task, while another member had a way to complete that same task but in a completely different way. And from this, a lesson learned about building software as a team was mainly about compromise and flexibility.

Additionally, this project did not use any planning tools, as most of the project planning and issue discussion was done in person or over group messages. Overall, there were a couple of issues that arose throughout the project, such as certain members not being able to run their code (due to laptop issues), and the game formatting differently on different laptops when launched. However, these issues were resolved by mainly using one laptop (that worked) to run everything and base the formatting on how that laptop outputted the program when launched.

VIII. Future Work

For a future project, the game could implement:

1. A way for there to be an unlimited number of players.
    a. A one-player game could be the program inputting a disk in a random column, meant to simulate a robot, so the player could play by themselves.
2. Users can input the dimensions of the board themselves, rather than having preset board sizes.
    a. The game would most likely have a minimum range of 4 by 4, so a player would be able to win, but the maximum range could go up to infinity (or however large the user wants).
3. If given the information on how it could be implemented, having the game be online so users can compete with other users who are not physically present in competitive matches.

IX. Glossary

Term	Definition
Connect Four	A classic two-player board game where players try to connect four of their colored discs in a row vertically, horizontally, or diagonally.
Disc	The physical round piece each player drops into the game grid; in the digital version, the discs, also called tokens, are represented by symbols like "X," "O," “#,” or “$.”
Grid	The vertical board with rows and columns where discs are dropped during Connect Four gameplay.
Draw	A tie in the game where no player wins because the board fills up with discs without any player connecting four in a row.
Tokens	In the digital game version, a character (like "X," "O," "#," or "$") that represents a player’s disc.
Vertical	A direction running up and down. In Connect Four, four discs stacked vertically wins the game.

Horizontal	A direction running left to right. Four discs in a horizontal line wins the game.
Diagonal	A direction slanting from one corner to the opposite corner (like bottom-left to top-right). Four diagonally connected discs also win.
User Interface (UI)	The screen layout and visual elements that players interact with when playing the game.
Intro Screen	The first screen of the game showing the title, credits, and options to start or quit the game.
Player Setup	The process where players input names, choose their symbols, and set up their board before the game begins.
Gameplay Mechanics	The rules and actions that govern how the game is played, including taking turns and placing discs.
Winner’s Screen	The screen displayed after someone wins the game, showing the winning player's name and offering a replay or quit option.
Unit Testing	A software testing method where individual parts of the code (units) are tested to make sure they work correctly.

UML Diagram	(Unified Modeling Language) A visual chart that shows the structure and design of a system, such as class relationships.
Class Diagram	A type of UML diagram that shows the system's classes, their attributes, and relationships.
Sequence Diagram	A type of UML diagram that shows how objects interact in a particular sequence (order of operations).
API	(Application Programming Interface) A set of rules allowing different software programs to communicate with each other.
Repository	A digital storage location (such as GitHub) where a project's code and files are saved and managed.
Pull Request	A request to merge code changes from one part (branch) of a repository into the main project, often used for team collaboration.
Branch (Git)	A separate version of the project code that can be worked on independently before merging into the main version.
GUI (Graphical User Interface)	A visual way for users to interact with a computer program, often with buttons, images, and text rather than text commands.

Cybersecurity	Protecting computer systems, networks, and data from digital attacks or unauthorized access.
Criminology	The study of crime, criminal behavior, and the law enforcement system.
Management Information Systems (MIS)	A field combining business and computer science to manage and analyze business data and technology systems.
Planning Tools	Software used by teams to organize tasks, assign work, and track progress (examples: Trello, Git Issues).

X. References

Team, Hasbro Content. 2025. “Connect 4 Game.” Hasbro Instructions. 2025. https://instructions.hasbro.com/en-in/instruction/connect-4-game.

XI. Appendices

Appendix A: The Official Connect 4 Rules (from the Hasbro website):

- Challenge a friend to a game of Connect Four.
    - For this project, we allowed the game to support from 2 to 4 players.
- Drop your red or yellow discs in the grid and be the first to get 4 in a row to win.

- If your opponent is getting too close to 4 in a row, block them with your disc!
- Whoever wins can pull out the slider bar to release all the discs and start all over again.
- Meant for ages 6 and up