SP-14 Red Build Chess Game Using AI

Presentation video : https://youtu.be/5X08BBjj99

Date: 4/28/23   Website: https://sites.google.com/view/sp14red/home

Roles	**Name**	**Major responsibilities**	**Contact (Email and/or Phone)**
Team leader	Aaron Dailey	Team lead, Documentarian, quality tester, and designer	aarondailey9@gmail.com 706-312-4017
Team members	Nicholas Kennel	UI designer and Implementer	nickovertime9@gmail.com 770-899-8206
	Haige Zhu	AI designer and implementer	hzhu5@students.kennesaw.edu 847-848-5735
Advisor / Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	Sperry46 in D2L !!

Aaron Dailey   Nicholas Kennel   Haige Zhu

Contents

1.0 Introduction

The following project is a chess game that allows the user to play against an AI on a Windows computer made from C#, that uses the Min-Max, Negamax, and Alpha-Beta pruning algorithms running on WinForms. The project started after the group chose to do this back in January. Originally the project was a chess Ai game being developed in Unity, but around late February we changed our development program to WinForms. More details on why we changed can be found in 5.3. The main algorithms used for the program are Alpha-Beta pruning, Min-max, and Negamax.

Alpha-Beta pruning is an optimization technique for the Minimax algorithm, which involves cutting off certain branches during the search process. This can save computational resources and increase the depth of the algorithm. The pruning algorithm abandons a branch if it discovers that the branch will lead to a worse outcome. This method does not affect the Minimax algorithm but can speed up the algorithm. Of course, if the optimal path can be found at the beginning, the alpha-beta algorithm will be more.

PVS (Principal Variation Search) is a variant of the Alpha-Beta algorithm. The PVS algorithm is also commonly known as the NegaScout search algorithm. The ordinary minimax algorithm appears more complex, as one side attempts to take the maximum value while the other side tries to take the minimum value. Therefore, we need to check which side is going to take the maximum or minimum value, to perform different actions. In 1975, Knuth and Moore proposed the Negamax method, which eliminated the difference between the two sides and was concise and elegant. Using the Negamax method, both sides of the game take the maximum value. The core of the Negamax algorithm is that the value of the parent node is the negation of the maximum value of the values of its child nodes. To ensure the correct operation of this algorithm, there is an additional point to consider. For example, in chess, the evaluation function must be sensitive to which player is making the move. That is, if a positive evaluation value is returned for a position where the red player makes a move, then a negative evaluation value should be returned for a position where the black player makes a move. Note the negative sign involved. In terms of principles, the MINMAX and NEGAMAX algorithms can be equivalent. However, their search efficiency is different. From the third level onwards, PVS search surpasses the speed of MINMAX + Alpha-Beta search. Of course, the number of evaluated leaf nodes in the same depth search is also less than that of the Alpha-Beta search. When the search depth is 5, the speed of the PVS search reaches 250% of that of the MINMAX + Alpha-Beta search.

1.1 Project Goals

The goal of this project is to develop an AI system that can automatically evaluate the current state of a chess game and provide the best move according to its analysis. The system will be presented through a graphical interface that displays the chessboard and the current state of the game as it progresses through human and AI moves. It mainly has the following functions:

• Graphical interface display (showing window controls for user interaction, displaying the chessboard and chess pieces)

• AI evaluation settings (search depth, setting values for the relative strength of different chess pieces) • Game player selection of game mode (both sides controlled by AI, one side controlled by AI and switchable, both sides manually controlled by players)

• Game player initiation and progression of gameplay

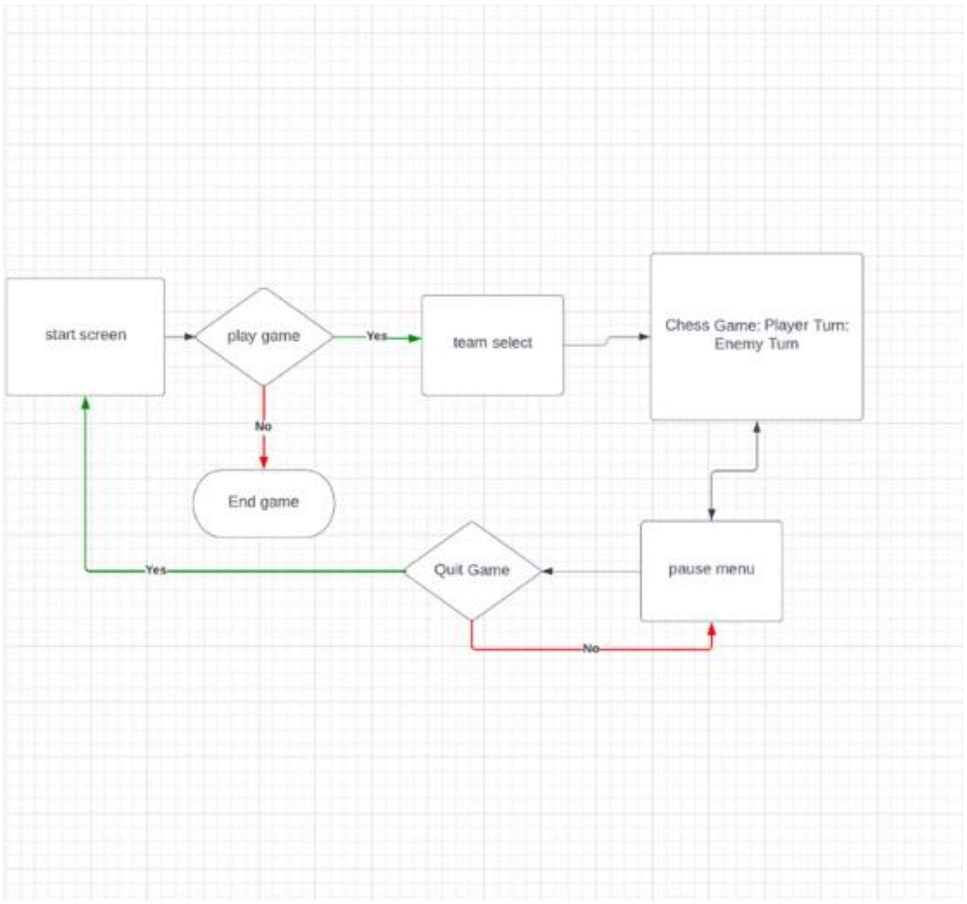
• AI automatic evaluation and completion of human-machine gameplay

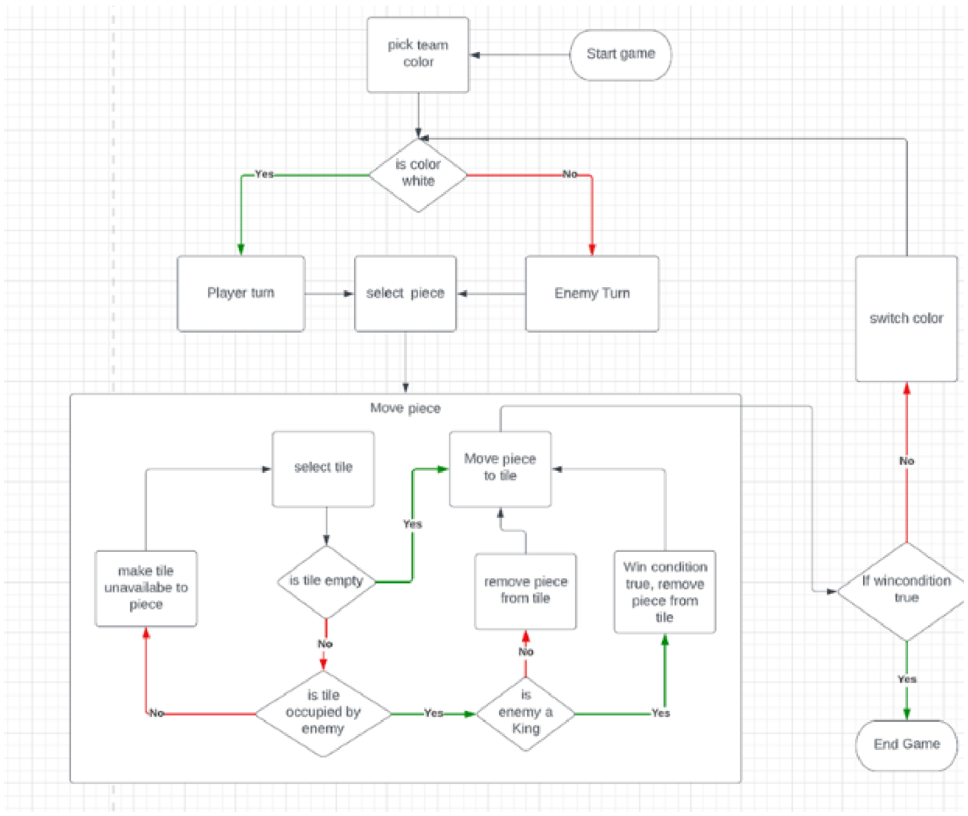
• Software checks for compliance with game rules in move decisions

• Software detects game end conditions (checkmate, stalemate, etc., or if more than 50 moves have been played without a capture)

• Software timing move decisions.

2.0 Design Constraints





2.1 Environment

Chess is a class that controls the logical operation of the game and forms the core of the software. It has two independent auxiliary classes: ChessBoard and LegalMoveSet. ChessBoard mainly records the current state of the chessboard in various forms. LegalMoveSets is a chess rule class used to determine whether a move is legal and whether the game is over. It can also form the action set of each chess piece under the current state.

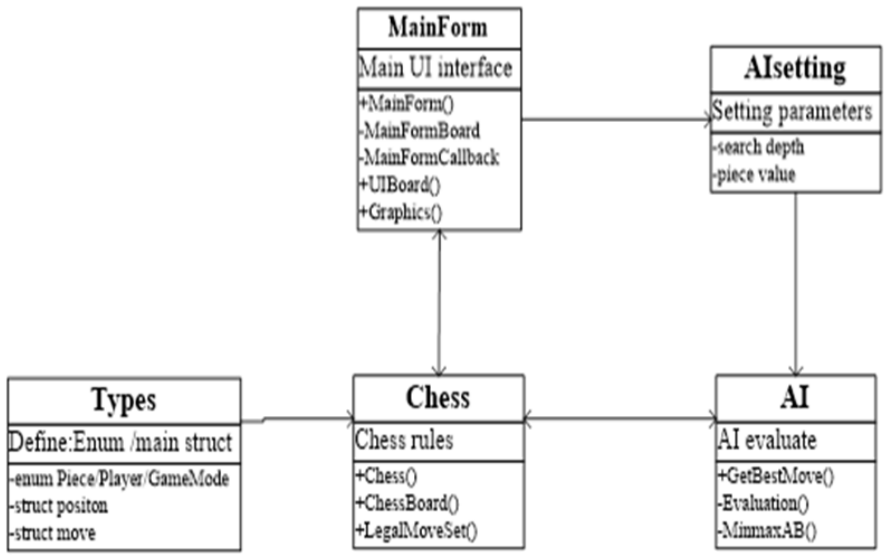
MainForm is the form that runs the game and serves as the UI. It has two shadow forms, MainFormBoard and MainFormCallbacks, to ensure smoother interface operation.

AI is an AI that evaluates values and uses a minimax search with alpha-beta pruning to make the best move decisions.

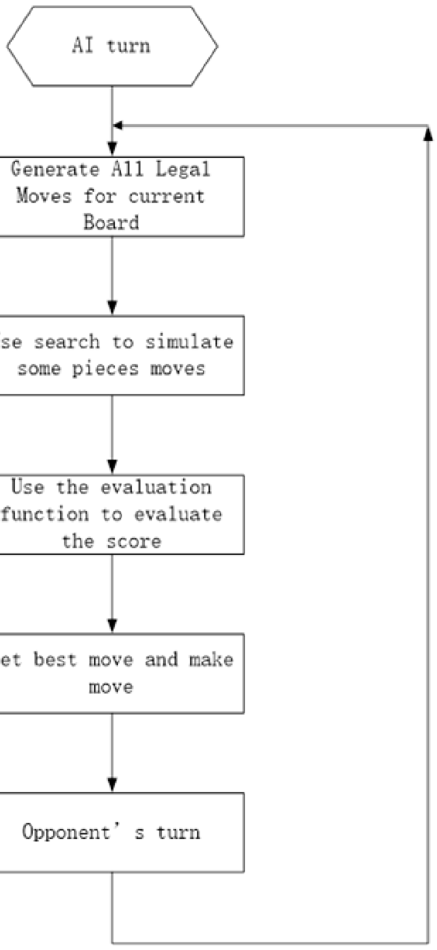
Types define various enumeration values and structures required by the system, such as chess piece enumeration, player enumeration, game mode enumeration, chess piece position structure, and chess piece movement model structure.

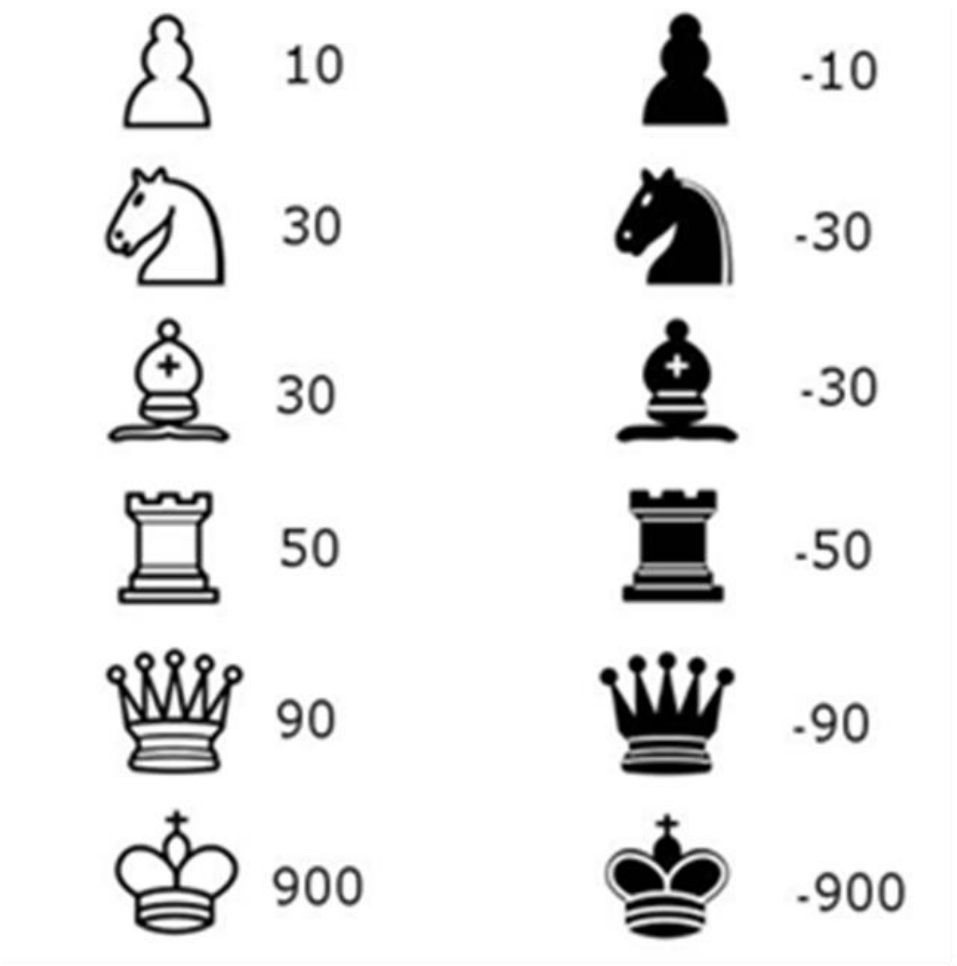
AIsetting allows users to set the game difficulty (search depth) and chess piece strength values through the FrmSetting form interface to affect the AI evaluation results.

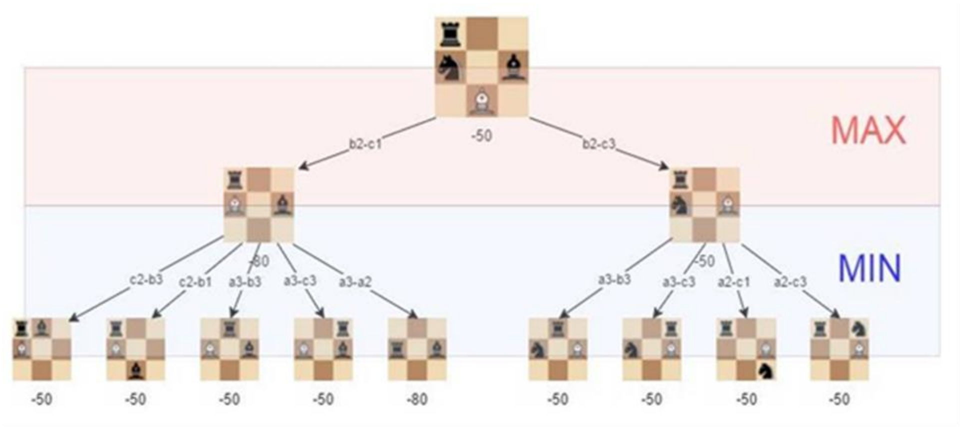
2.2 User Characteristics



2.3 System


AI turn
Generate All Legal Moves for current Board
Use search to simulate some pieces moves
Use the evaluation function to evaluate the score
Get best move and make move
Opponent's turn





3.0 Requirements

3.1 User Requirements

1. The game begins with a start screen where it would allow you to do 3 options: one starts the game; One opens the credit screen; one exits the game.
2. Players can choose the start color. From there on, it is going to go to the main chess game scene where the player can play the AI. The chess game scene is going to consist of some canvases for the UI, chess pieces, and the chess board.

3. It is going to be a basic UI that lets you be able to just click on a chess piece, then it is going to enable you to see where that piece can move as well as where it can move and be vulnerable to enemy pieces.
4. After that you move a chess piece, it will automatically switch from the player's turn to the AI's turn, and then the AI's turn will then select a chess piece and then move their chess piece. It will switch back and forth until it gets to a point where one of the chess pieces will take another chess piece.
5. When a piece takes another chess piece it would replace that piece or delete the enemy's chess piece. Then that will continue until you take the enemy (AI) king or until the enemy takes your king and at that point, it will change a win condition through either you or the AI to true. Once the king is taken, instead of going to the enemy's turn or your turn it will, then just go to the victory screen or defeat screen.
6. Finally, you could choose to replay the game again.

3.2 AI Requirements

1. We use the training data as chess games and rules.
2. Our goal for the AI part is at least for the AI to be able to move the piece by the logic and the rules, while AI plays with the user. For example, if the player the user uses white and he makes a move with the pawn to E4, then the AI will use the minimax algorithm to evaluate the optimal solution based on the user's moves, and then the Alpha-beta algorithm will optimize the calculation time, trying to optimize the time to within a few seconds
3. Deep learning and data sets to train AI to achieve AI to play against people. It can set the difficulty.
4. The AI part is to calculate the possibility of winning each step through the Alpha -Beta pruning algorithm and reduce the calculation time through algorithm optimization.
5. The Minmax algorithm is used to evaluate the optimal solution based on the user's moves.

4.0 External Interface Requirements

4.1 User Interface Requirements

An interactive chess piece, available chess movement, pause screen, start screen, and a victory/lose menu.

5.0 Hardware Interface Requirements

A computer that has x64 architecture for its CPU, and GPU that has DX10, DX11, and DX12-capable GPUs.

5.1 Software Interface Requirements

Required software need is Windows 7 (SP1+) or Windows 10, 64-bit versions or Windows 11.

5.2 Tech platform

The platform we used was WinForms which runs off the programming language C# and runs on Windows. The platform uses min max and alpha pruning algorithms for the chess game to operate. The platform can run AI vs. AI, player vs. player, and variations of AI vs. player. The Winforms app runs different levels of difficulty for artificial intelligence from level one (easiest)to level 5(hardest). The app has the ability to change the value of all chess pieces.

5.3 Challenges and issues

The development originally was going to use unity but as we were working on it we decided to choose WinForms since we progressed a lot there so we could train the AI. There were challenges into finding how we would do the columns and rows for the game but we eventually figure it out by using the min max and alpha pruning algorithms. The development of the software   has mostly gone on time and the only problems we are facing now are the speed of how level 4 and 5 complete their moves while every other feature has been completed or has been done.

6.1 Results

The current results is that the Ai can run efficiently on all levels and don’t have any bugs or problems while running on any Windows computer.

6.2 Project Plan and Management

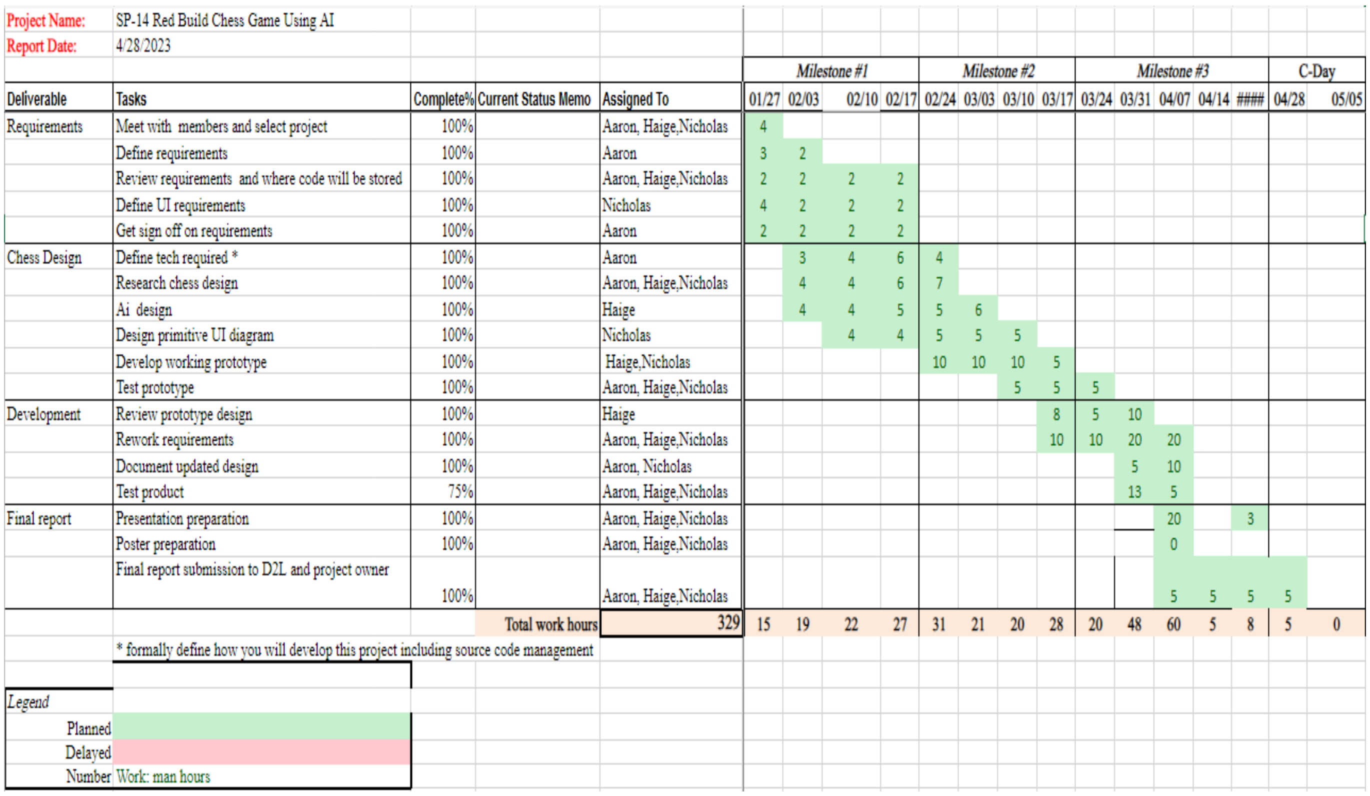
Meeting Schedule Date/Time

Mondays 4- 5 pm and Wednesdays 4-5 pm

Collaboration and Communication Plan

- Discord will be the main form of communication otherwise other forms of communication will be via email or phone. Regular meetings will be held on Mondays and Wednesdays 4-5 pm in discord if needed meetings can be arranged at other times. The requirements for meetings is a microphone and pc or phone to communicate in discord. Meeting and distributing notes will be handled by the Team leader to keep everyone on what is needed; otherwise, if the leader cannot attend, the notes will be delegated to one of the other team members.
- The tool we will use for communication will be Discord while file sharing of documents will be by OneDrive and sharing of code will be by GitHub. Discord is how we will communicate where all files are in OneDrive and GitHub as well as post shared links to the files. GitHub will be used as our website but also where we will share code and versions of our game. OneDrive will be the documentation for our deliverables will be stored.
- The regular status updates and progress will be reported bi-weekly in each meeting Mondays and Wednesdays. If needed other updates may come outside those frames are to keep the workflow ahead of schedule or to accomplish a certain task that was not available previously. All members of the team are responsible for reporting their updates to other members, so we all know what we need to work on for the project

Project Schedule and Task Planning (Ganatt chart)



Project Name:	SP-14 Red Build Chess Game Using AI																		
Report Date:	4/28/2023																		
					Milestone #1				Milestone #2				Milestone #3					C-Day	
Deliverable	Tasks	Complete%	Current Status Memo	Assigned To	01/27	02/03	02/10	02/17	02/24	03/03	03/10	03/17	03/24	03/31	04/07	04/14	####	04/28	05/05
Requirements	Meet with members and select project	100%		Aaron, Haige,Nicholas	4														
	Define requirements	100%		Aaron	3	2													
	Review requirements and where code will be stored	100%		Aaron, Haige,Nicholas	2	2	2	2											
	Define UI requirements	100%		Nicholas	4	2	2	2											
	Get sign off on requirements	100%		Aaron	2	2	2	2											
Chess Design	Define tech required *	100%		Aaron		3	4	6	4										
	Research chess design	100%		Aaron, Haige,Nicholas		4	4	6	7										
	Ai design	100%		Haige		4	4	5	5	6									
	Design primitive UI diagram	100%		Nicholas			4	4	5	5	5								
	Develop working prototype	100%		Haige,Nicholas					10	10	10	5							
	Test prototype	100%		Aaron, Haige,Nicholas							5	5	5						
Development	Review prototype design	100%		Haige								8	5	10					
	Rework requirements	100%		Aaron, Haige,Nicholas								10	10	20	20				
	Document updated design	100%		Aaron, Nicholas										5	10				
	Test product	75%		Aaron, Haige,Nicholas										13	5				
Final report	Presentation preparation	100%		Aaron, Haige,Nicholas											20		3		
	Poster preparation	100%		Aaron, Haige,Nicholas											0				
	Final report submission to D2L and project owner	100%		Aaron, Haige,Nicholas											5	5	5	5	
			Total work hours	329	15	19	22	27	31	21	20	28	20	48	60	5	8	5	0
	* formally define how you will develop this project including source code management																		
Legend																			
Planned	🟩																		
Delayed	🟥																		
Number	Work: man hours																		

Version Control Plan

The plan for the version control is to make the coding and software in WinForms and then export it to our GitHub to maintain and document different versions of the game. The GitHub will have the most updated version of the game and will reflect what code has already been done.

6.3 Test Plan

The minimum amount of tests should be 2 for each level for each game type except player vs player will be tested 2 times since it's not reliant on artificial intelligence, and the tests must be conducted on a Windows computer. The total amount of tests should be around 32, and to see if it has any other problems on other computers. It will be tested on another computer for the same making it 64 tests. All ai will be timed to see if it’s the game loading time is dependent on the CPU and GPU. If the CPU and GPU are, it shows that we need to change our current code to be more efficient. The overall test plan is to make sure all game modes work and make sure the ai is efficient in playing the game of chess with the user.

The test was conducted on a KSU (Kennesaw State University) library computer and Aaron’s laptop. The time for each chess black or white is the avg time of two trials for each computer for each level. The avg time is for the times completed for each game type. The Ai game type will have to be the avg of the black and white chess ai but for the other game types, it will only have one since the other chess side is the player. * The avg time is rounded to the closet millisecond Ex: 0:00:00.8

Game Type	Level	Avg Time	Specs	Black chess	White Chess
Ai	1	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00	0:00:00
ai	1	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00	0:00:00
AI	2	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00	0:00:00
Ai	2	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00	0:00:00
Ai	3	0:00:00.7	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00.8	0:00:00.6
Ai	3	0:00:00.4	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00.3	0:00:00.5
Ai	4	0:00:02.5	CPU: Intel i7-7700	0:00:03.0	0:00:02.7

			GPU: Intel(R) Hd Graphics 630		
Ai	4	0:00:02.5	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:03.3	0:00:02.8
Ai	5	0:00:58.3	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:55.0	0:01:01.7
Ai	5	0:00:40.5	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:41.9	00:00:40.4
Black Ai vs player	1	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00	N/A
Black Ai vs player	1	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00	N/A
Black Ai vs player	2	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00	N/A

Black Ai vs player	2	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00	N/A
Black Ai vs player	3	0:00:00.8	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:00.8	N/A
Black Ai vs player	3	0:00:00.7	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:00.7	N/A
Black Ai vs player	4	0:00:04.4	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:04.4	N/A
Black Ai vs player	4	0:00:02.7	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:02.7	N/A
Black Ai vs player	5	0:00:33.2	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	0:00:33.2	N/A
Black Ai vs player	5	0:00:19.8	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	0:00:19.8	N/A

White Ai vs player	1	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	N/A	0:00:00
White Ai vs player	1	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	N/A	0:00:00
White Ai vs player	2	0:00:00	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	N/A	0:00:00
White Ai vs player	2	0:00:00	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	N/A	0:00:00
White Ai vs player	3	0:00:00.8	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	N/A	0:00:00.8
White Ai vs player	3	0:00:00.2	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	N/A	0:00:00.2
White Ai vs player	4	0:00:04.2	CPU: Intel i7-7700 GPU: Intel(R) Hd	N/A	0:00:04.2

			Graphics 630		
White Ai vs player	4	0:00:01.5	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	N/A	0:00:01.5
White Ai vs player	5	0:01:33.3	CPU: Intel i7-7700 GPU: Intel(R) Hd Graphics 630	N/A	0:01:33.3
White Ai vs player	5	0:01:33.7	CPU: Intel i7-11800H GPU: RTX 3060 Laptop	N/A	0:01:33.7

6.4 Development Process

We researched how to build a chess board in unity and started working on that. But after going through, making some design choices, and discovering that it's better to stay using WinForms, we stayed and use that. We also went and added some additional screens to the chess game, including the start screen pause screen and the credit screen does not pause screen, just the start and credit screen.

First We constructed a model and relay the message for chess movement rules based on the international or rule of international chess and 2nd We implement AI that automatically responds to player moves to form a game and 3rd is implement a graphic interface to display the game progress and the board state for both players. The force provides a prompt for capturing peace or putting their opponents in check and 5th just record the game history log for both players. Then We did a couple of evaluations for the AI algorithm. To improve the speed of the AI to save some time. Our first game demo was one minute or three, three to four minutes. All right, now We add a couple of different algorithms such as transaction tables and then Negamax to reduce the time. So right now, our AI reaction time is reduced to one minute.

6.5 Summary and Conclusion

The project we created was a Chess game with Artificial intelligence with C# in the platform WinForms. The goal was to make a chess game where the player could play against Artificial intelligence. The goal was completed, and we added more features which were player vs player, AI vs Ai and changed values of the chess pieces, and the program's ability to track all pieces' movement and timing. The project is completed and is fully functional for the public and doesn’t have any speed problems in the previous version of the program. The program can be used on any Windows computer.