A Mini Project Report On

**CONNECT-FOUR GAME**

Submitted in partial fulfilment of the requirement of

**University of Mumbai**

For the Degree of

**Bachelor of Engineering (SE)**

*in*

**COMPUTER ENGINEERING**

*Submitted by*

**Reuben Brian D’silva**

**Jeevan Kunder**

**Aryan Hirlekar**

**Atharv Kadam**

*Supervised by*

**Mrs. Prachi Verma (Guide)**



**Department of Computer Engineering**

**Fr. Conceicao Rodrigues Institute of Technology**

**Sector 9A, Vashi, Navi Mumbai - 400703**

**UNIVERSITY OF MUMBAI**

**2022-2023**

**APPROVAL SHEET**

This is to certify that the Mini project entitled

**“CONNECT-FOUR GAME”**

**Submitted by**

Reuben Brian D’silva 1021135

Jeevan Kunder 1021163

Aryan Hirlekar 1021148

Atharv Kadam 1021156

**Supervisors: Mrs. Prachi Verma**

**Project Coordinator: Mrs. Dakshayani R**

**Examiners: 1.**

**2.**

**Head of Department: Dr. Lata Ragha**

**Date:**

**Place: Vashi, Navi Mumbai**

i

**Declaration**

We declare that this written submission for S.E. Mini Project entitled “**CONNECT FOUR GAME**” represent our ideas in our own words and where others’ ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

**Project Group Members:**

Reuben Brian D’silva, 1021135

Jeevan Kunder, 1021163

Aryan Hirlekar, 1021148

Atharv Kadam, 1021156

ii

**Abstract**

Trying to develop games where a human player is matched against a computer has been in development for decades. Along with this development, efforts have also been made to make the games enjoyable for longer periods of time. One method that can be employed is by providing a sense of randomness. For a two-player game as simple as Connect Four, there will hardly be anything that can prove to be a surprise during a regular game. It can help to include events that can occur at any time, supposedly at random. Having a player play against the computer can be the best way to implement randomness if properly executed. Algorithms such as the Minimax algorithm could provide the best move to perform to minimize the chances of the player winning. But seeing as the calculations are done from the point of view of both the computer and the player, and knowing that the human player won’t be able to compete, finding ways to handicap the computer could be a solution. The Alpha-Beta algorithm will prune out options that provide worse results than the ones already determined. This could also be used to omit certain possibilities to make sure the computer doesn’t always win. To add to the aspect of making the game experience engaging, features such as three-player games, time-based modes, and a high score leaderboard can be added.

iii

**Contents**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [**Abstract**](#bkmpage1) | | | | | **iii** | |
| [**List**](#bkmpage4)[**of**](#bkmpage4)[**Figures**](#bkmpage4) | | | | | **vi** | |
| [**List**](#bkmpage7)[**of**](#bkmpage7)[**Tables**](#bkmpage7) | | | | | **vii** | |
| **1** | [**Introduction**](#bkmpage10) | | | | | **9** | |
|  | [1.1](#bkmpage11) | [Background](#bkmpage11) | | | | **10** | |
|  | [1.2](#bkmpage11) | [Motivation](#bkmpage11) | | | | **10** | |
|  | [1.3](#bkmpage11) | [Aim](#bkmpage11) [and](#bkmpage11) [Objective](#bkmpage11) | | | | **10** | |
|  | [1.4](#bkmpage12) | [Report](#bkmpage12) [Outline](#bkmpage12) | | | |  | |
|  |  |  | | | |  | |
| **2** | [**Study**](#bkmpage13)[**Of**](#bkmpage13)[**the**](#bkmpage13)[**System**](#bkmpage13) | | | | | **11** | |
|  | 2.1 | [Literature](#bkmpage15) [Review](#bkmpage15) | | | | **12** | |
|  |  | 2.1.1 | | Real Time Connect 4 Game Using Artificial Intelligence | | **12** | |
|  |  | 2.1.2 | | Alpha-Beta Pruning in Min-Max Algorithm | | **12** | |
|  |  | 2.1.3 | | Bridging the Blank: A Defining from Property Uncertainty to Felt Uncertainty in Games | | **13** | |
|  |  | 2.1.4 | | Questioning Wizard of Oz: Effects of Revealing the Wizard behind the Robot | | **14** | |
|  | 2.2 | Existing System | | | | **16** | |
|  |  | 2.2.1 | Helpfulgames.com Connect Four Game | | | **16** | |
|  |  | 2.2.2 | CBC Kids Connect4 Game | | | **16** | |
|  |  | 2.2.3 | Papergames.io’s connecct4 game | | | **16** | |
|  |  |  |  | | |  | |
| **3** | [**Proposed**](#bkmpage27)[**System**](#bkmpage27) | | | | | **17** | |
|  | 3.1 | [Problem](#bkmpage28) [Statement](#bkmpage28) | | | | **18** | |
|  | 3.2 | [Scope](#bkmpage28) | | | | **18** | |
|  | 3.3 | Hardware and Software Requirements | | | | **18** | |
|  |  |  |  | | |  | |
| **4** | [**Design**](#bkmpage38)[**Of**](#bkmpage38)[**the**](#bkmpage38)[**System**](#bkmpage38) | | | | | **19** | |
|  | 4.1 | Approach | | | | **20** | |
|  |  | 4.1.1 | Flow Chart | | | **21** | |
|  | 4.2 | Technology Stack | | | | **22** | |
|  |  | 4.2.1 | Front End | | | **22** | |
|  |  | 4.2.2 | Back End | | | **22** | |
|  |  |  |  | | |  | |
|  |  |  |  | | |  | |
| **5** | **Result and Discussion** | | | | | **23** | |
|  | 5.1 | Observation | | | | 24 | |
|  | 5.2 | Implementation | | | | 24 | |
|  |  | 5.2.1 | Landing Page | | | 25 | |
|  |  | 5.2.2 | Account Page | | | 27 | |
|  |  | 5.2.3 | Mode Select | | | 25 | |
|  |  | 5.2.4 | 2-Players | | | 26 | |
|  |  | 5.2.5 | 3-Players | | | 27 | |
|  |  | 5.2.6 | Leaderboard | | | 27 | |
|  |  | 5.2.7 | Step-by-step working | | | 28 | |
|  | 5.3 | Important Code Snippets | | | | 29 | |
|  |  | 5.3.1 | Code Snippet 1 | | | 29 | |
|  |  | 5.3.2 | Code Snippet 2 | | | 32 | |
|  |  |  |  | | |  | |
| **6** | [**Conclusion**](#bkmpage64)[**&**](#bkmpage64)[**Future**](#bkmpage64)[**Scope**](#bkmpage64) | | | | | **34** | |
|  | [**References**](#bkmpage64) | | | | |  | |
|  | [**Appendix**](#bkmpage66)[**A:**](#bkmpage66)[**Timeline**](#bkmpage66)[**Chart**](#bkmpage66)**s** | | | | |  | |

v

**List of Figures**

|  |  |  |
| --- | --- | --- |
| 1 | Final Game Tree Without Alpha-Beta Pruning | 12 |
| 2 | Final Game Tree with Alpha-Beta Pruning | 12 |
| 3 | Final average analysis of the collected data | 14 |
| 4 | Trust in the robot for the 3 experimental conditions | 15 |
| 5 | Intelligence perception for the 3 experimental conditions | 15 |
| 6 | Likeability perception for the 3 experimental conditions | 15 |
| [7](#bkmpage16) | Flow chart | 21 |
| 8 | Landing page | 23 |
| 9 | Account page | 25 |
| 10 | Mode Select | 25 |
| 11 | 2 players (red winning) | 26 |
| 12 | 3 players screen | 27 |
| 13 | Leaderboard display | 27 |

vi

**List of Tables**

1. [Comparison between Two Algorithms](#bkmpage39) 13

vii

**Chapter 1**

**Introduction**

* 1. **Background**

Connect Four is a two-player connection board game, in which the players choose a color and then take turns dropping colored tokens into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. A player wins if they manage to get four consecutive tokens of their color within the grid.

**1.2 Motivation**

The game, while seemingly easy, actually has 4,531,985,219,092 possible tokens configurations. Creating a program with respect to all those possibilities is difficult. So, finding ways to make it easy to for computer is a major motive. As beginners implementing features like creating a good scoring system, having time-based modes, the ability to customize the in-game color themes is a good starting point. This process will include concept generation, proper design, development and testing.

**1.3 Aim and Objective**

The main aim is to about various methods and algorithms deployed to create games. Implementing those methods, mainly to create a computer bot to play against. To improve upon the simple playstyle of the game by adding some factors of uncertainty and randomness, as games where the computer performs the same predefined steps, people will find it boring. To learn more about the challenges and difficulties faced within game development and trying to overcome the same.

**1.4 Report Outline**

This report presents our attempt at making a playable game on a single device regardless the number of players. We have provided our approach to make the program suitable for taking multiple user inputs as well as for a single user and a machine learning algorithm.

**Chapter 2**

**Study of the System**

**2.1 Literature Survey**

**2.1.1 Real Time Connect 4 Game Using Artificial Intelligence [1]**

To help the AI play Connect Four using heuristics with influence mapping. The different types of AI implemented were: Random, Defensive and Aggressive. The different modes were: Two players, with and w/o a timer, Against the AI, with and w/o a timer. The scoring system worked by checking the immediate up, down, left, right tokens of the same color. If more than one was present, it kept counting till required number of tokens were reached. The algorithms mentioned were: Minimax without Alpha-Beta, Minimax with Alpha-Beta, A\* algorithm, A\* finds the shortest path to reach the destination by creating trees for each path. Though it was not used because high memory consumption. Programs used while testing: VICTOR, Velena.

The program was implemented in C Lab Windows/CVI by National Instruments written for a Windows environment

**2.1.2 Alpha-Beta Pruning in Min-Max Algorithm [2]**

Using Minimax algorithm with and without Alpha-beta pruning and showing the distinction between programs running with above-mentioned methods

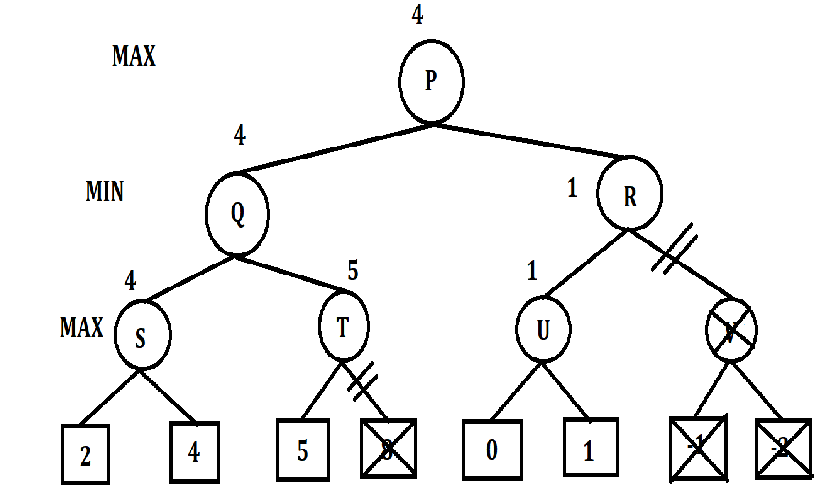
 

Fig 1: Final Game Tree Without Alpha-Beta Pruning**[2]** Fig 2:Final Game Tree with Alpha-Beta Pruning**[2]**

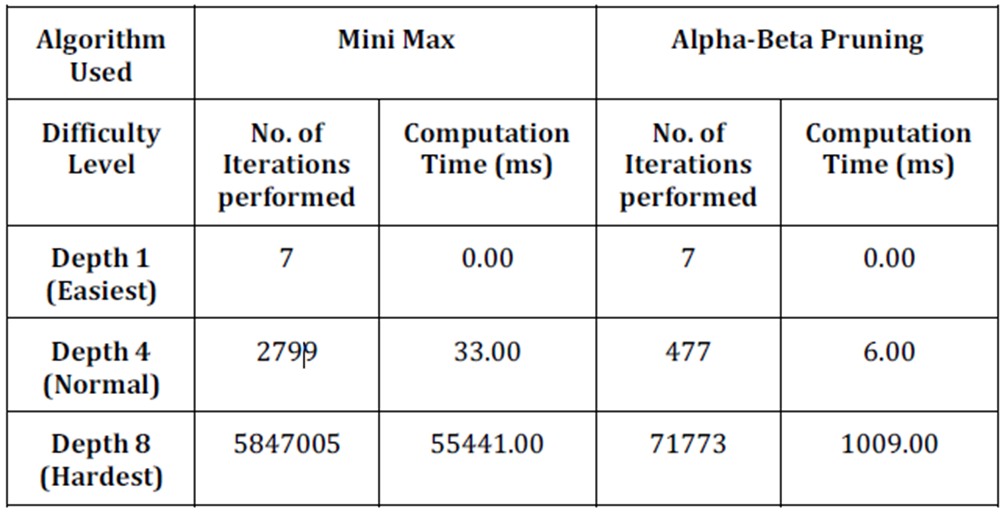


Table 1: Comparison between Two Algorithms**[2]**

**2.1.3 Bridging the Blank: A Defining from Property Uncertainty to Felt Uncertainty in Games [3]**

Importance of striking a balance of uncertainty in algorithms and in turn videogames. Having less randomness, the format becomes predictable and eventually boring. Having to much randomness causes difficulties in to learn patterns, and maybe even frustration

Different iterations of Connect Four developed were:

* FairGame7x6
* FairGame5x6
* FairGame9x6
* FairGameWithMCTS
* FairGameWithMCTSResultTold

Each participant was required to finish a total of seven games, and each game variant was played seven times.

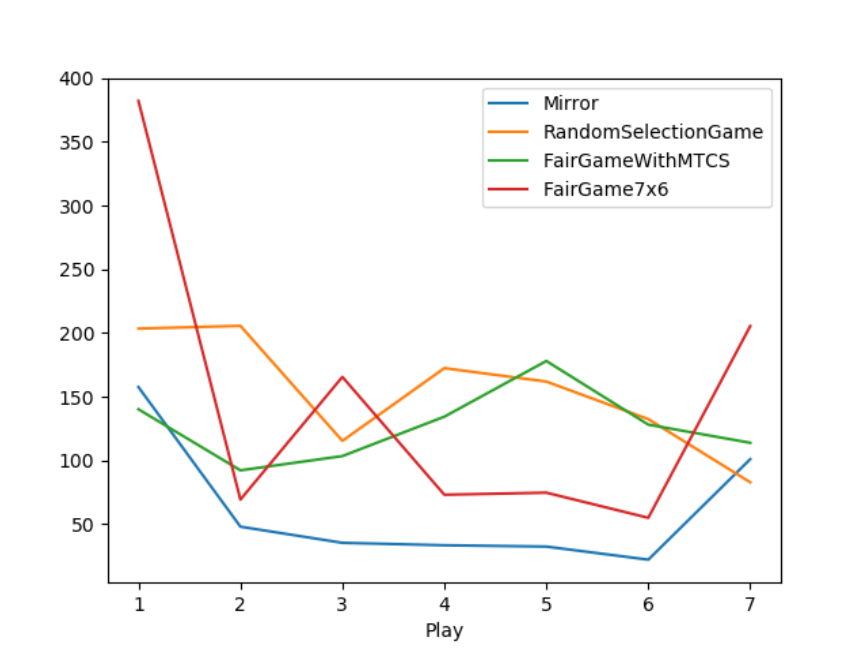
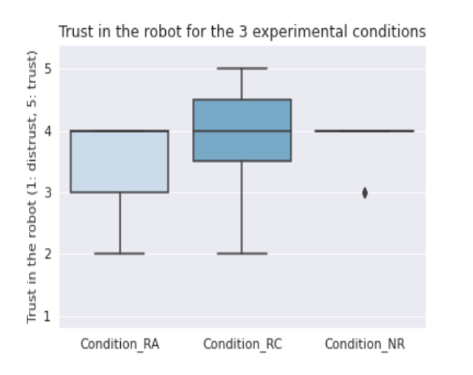


Fig 3: Final average analysis of the collected data**[3]**

**2.1.4 Questioning Wizard of Oz: Effects of Revealing the Wizard behind the Robot [4]**

Wizard of Oz: A humans pretending to be AI behind the system till the AI gets properly implemented. A robot and the participant cooperate to beat a computer AI. The robot provides suggestions to the human about the next best move. While at the end of the game the robot engages the participant in a short conversation. Criticism: Users made to believe they are interacting with a fully AI integrated system and the secret is revealed only at the end of the experiment. For the purposes of our study, the game selected was Connect 4 and probability AI, based on Monte Carlo tree search (MCTS), that can predict the probability of winning the round associated with a move was selected. As the focus is on the interactions between the game and the player factors such as the case in which the robot reveals to be controlled by a human, the case in which it declares to be fully autonomous and the case in which it says nothing. The conditions were defined as: Where it lies about being an AI, the robot truthfully reveals that it is being controlled by a human, The robot does not reveal anything. The analysis was done on the following data: How frequently the participants accept the suggestions of the robot, how much did the player trust the robot to give the correct option? Feedback from the players on if the robot was entirely controlled by humans or entirely autonomous



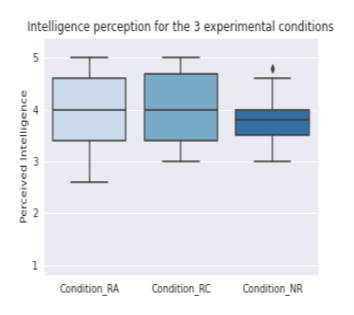
Fig 4: Trust in the robot for the 3 experimental conditions **[4]**

Fig 5: Intelligence perception for the 3 experimental conditions **[4]**

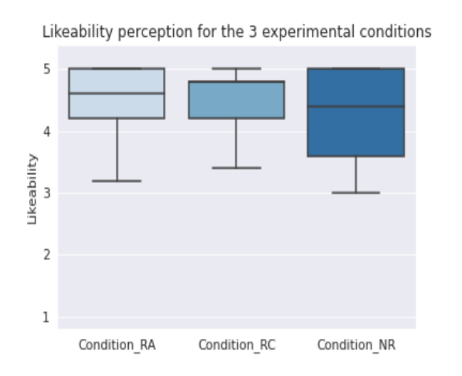


Fig 6: Likeability perception for the 3 experimental conditions **[4]**

**2.2 Existing System**

**2.2.1 Helpfulgames.com Connect Four Game [5]**

It has a very ordinary yet concise interface. 14 difficulty levels are available which range from “EASY” all the way up to “EXCEPTIONALLY HARD”. It displays a bar which shows the player’s knowledge points (exp points). The bottom of the screen has a tab which stores the player’s statistics (rounds played, levels cleared, accuracy) alongside the recent activities in the game. The mode in which the player starts first the computer sometimes goes faster and skips some of the players placement. The player always has to face the computers AI, also no other game modes are available. It has a bug in which if we click really fast more than one chip is deployed.

**2.2.2 CBC Kids Connect4 Game [6]**

The graphics were very well designed. It only has 2 modes i.e., User vs AI & User vs User. Game is designed in such a way that it is easy to play. The board had a 7x6 ratio. The game doesn't show any leaderboard. Only 2 modes are available.

**2.2.3 Papergames.io’s connecct4 game [7]**

The front end of this site is very well designed with a user-friendly interface. It has 4 different game mode play online play with a friend play vs robot create a tournament. It has a feature of ranked matches as well which is missing in most online connect 4 games. It also has an interactive leaderboard. Rules of the game were not explained in the beginning. A lot of bots in online matches as well. You cannot play this game without signing in.

**Chapter 3**

**Proposed System**

**3.1 Problem Statement**

The topic chosen for this project isn’t a new idea by any means and various implementations with varying degrees of complexity are already being made. The complexity comes in the algorithm chosen for the computer. Having a fully functioning application with the projected requirements is the final goal.

**3.2 Scope**

Playing a game such as this develops the skill of noticing patterns. To some extent it can also develop fine motor skills. As some amount of planning is required on playing the next move it can boost strategic thinking capabilities. Creates a quick-thinking mindset. It teaches competitiveness and how to handle associated feelings/emotions. And probably the most important of all it provides fun and entertainment

**3.3 Hardware and Software Requirements**

We have chosen to create the game in the programming language Python. The reason was it extremely popular, simple, easy to follow and flexible to use. In python we will require the Numpy, Pygame, Sys, math libraries. From existing systems, it was observed the games utilized about 150 MB of Memory. Our first implementation was in Jupyter Notebook and the game was CPU heavy and the RAM usage was similar as above. On later versions we ran the game on Vscode, and the game as a python file on its own.

**Chapter 4**

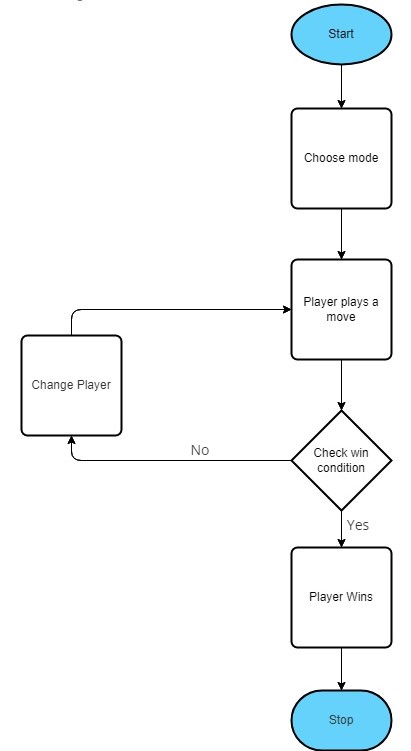
**Design of the System**

**4.1 Approach**

Features to be implemented:

* A proper menu system
* Support to play against both real players and AI
* Support for up to 3 real players
* Having a good amount of game modes
* Rewarding scoring system
* Leaderboards for multiple modes

**4.1.1 Flow Chart**

****Fig 7: Game basics flow chart

The flowchart in Fig 7 shows the barebones of the functioning of the program. The user will be prompted with a mode selection page after which the game will begin. Players will take in turns to play. The win condition is checked after each move. If it is satisfied, the current player wins and the game ends. If not, the play continues till a player wins.

**4.2 Technology Stack**

**4.2.1 Front End**

For creating the menu, we have used DearPyGui. DPG is a simple, bloat-free, and powerful Python GUI framework. It is easy to work with to provide a good interface for selecting game modes as well as for account creation.

**4.2.2 Back End**

All data regarding the match, i.e., score, points, will go through DPG to locally connected database on MySQL. Leaderboard information can be displayed by fetching from the database and user account details can also be viewed.

**Chapter 5**

**Result and Discussion**

**5.1 Observations**

Checking stats through the task manager, the python program running through Jupyter notebook was CPU heavy average 150 MB RAM throughout the multiple instances. The exit button doesn’t perform it’s intended operation and the entire program crashes and needs to be restarted. Same stats observed while running through Vscode and as a python file on its own.

**5.2 Implementation**

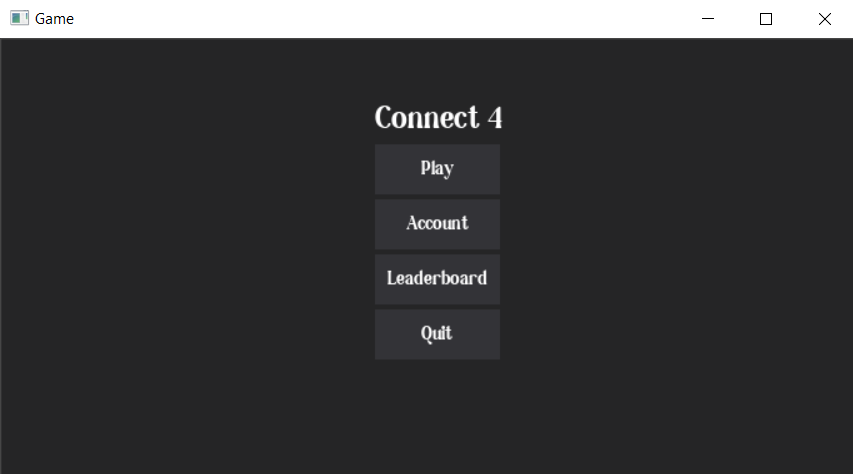
**5.2.1 Landing Page (screen shots of each step)**

Fig 8: Landing page

This is the first page the user will see. It includes the options to start a game, look at account options, see the leaderboards and to quit the game.

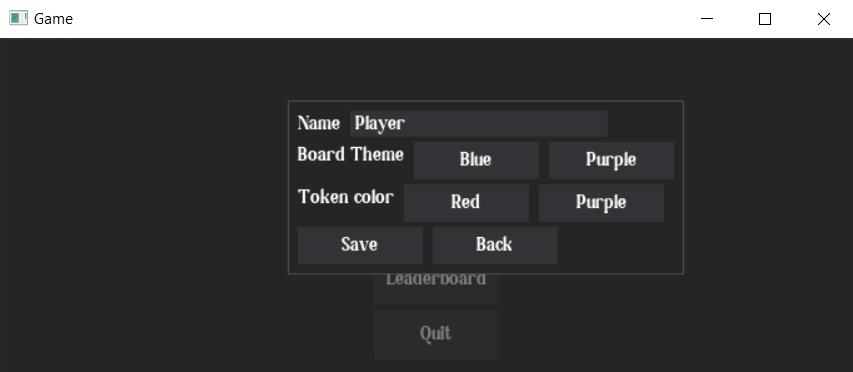
**5.2.2 Account Page**

Fig 9: Account page

On this page, the user will be able to create an account and select preferences for the board and token themes. This data will be stored in a local database.

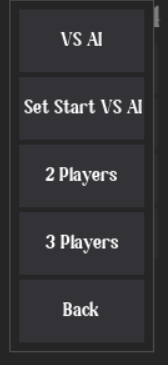
**5.2.3 Mode Select**

Fig 10: Mode Select

The user will be able to select from a variety of modes including options to play with two and three human, players or different modes and difficulties against the AI.

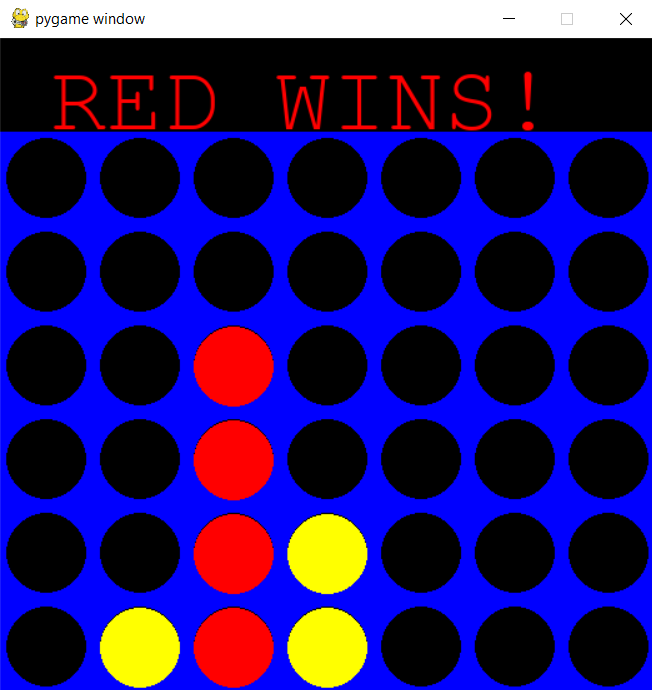
**5.2.4 2 Players**

Fig 11: 2 players (red winning)

This is the screen that will appear after selecting the 2-player mode and if the player with the red token wins.

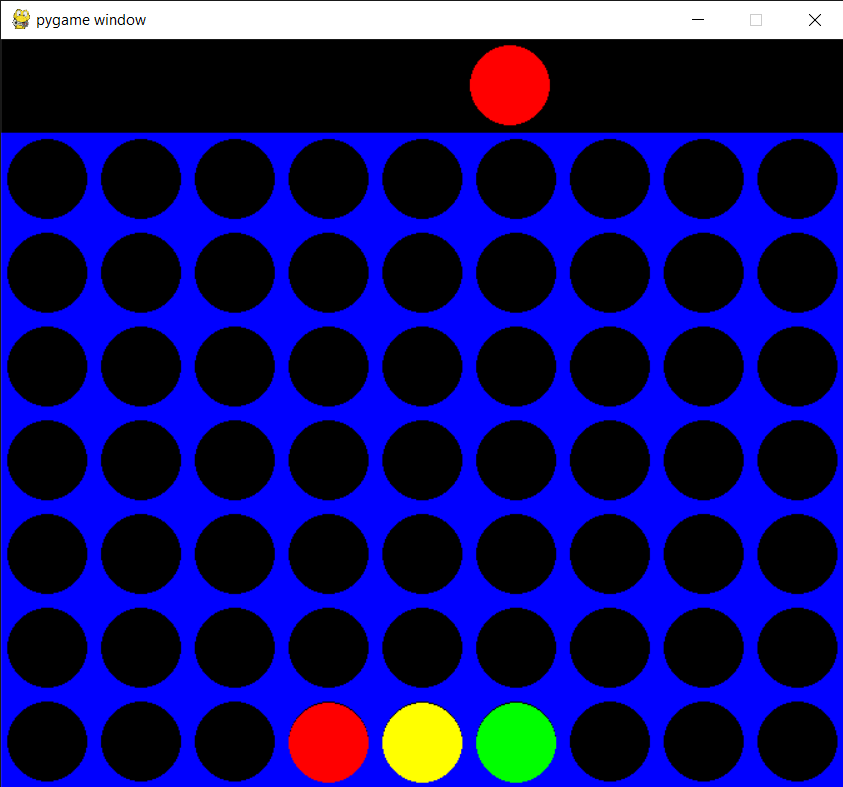
 **5.2.5 3 players**

Fig 12: 3 players

This is the starting screen of the 3-player mode. The board is bigger than the other modes for the extra player

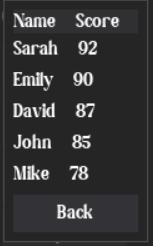
**5.2.6 Leaderboard**

Fig 13: Leaderboard display

This is the leaderboard option in the main menu which will display the in-game highscores

**5.2.2 Step-by-step working**

Step 1:

First, we import the NumPy package as np. Then create a python function named create\_board(). np.zeros( ) function is used to create a matrix full of zeroes.6 rows and 7 columns are the dimensions. Then we return the board. We will begin writing the main game loop as while not game-over. Our loop is going to run as long as this game\_over variable is false. The only time its going to switch to true is if someone gets 4 circles in a row. To increase the turn by 1 we will use turn += 1.

Step 2:

In the step 02, we make a few updates to the previous code. We want the selection variable to actually drop a piece on the board. For that, we will create another three functions as def drop\_piece( ), def is\_valid\_location( ), def get\_next\_open\_row( ). How these functions are going to Work together is as follows, the (0-6) in the code represents the column where they want to drop their piece. Hence we update the name of selection variable to column variable (col).Then we will take this col in the current board that we have and pass it as parameters in all the three functions with board. We will initialize global variables called ROW\_COUNT and COLUMN\_COUNT. The np.flip( ) function reverses the order of array elements along the specified axis, preserving the shape of the array.

Step 3:

In step 03, we will create a game with a GUI. With the new modifications that we do will make the game look like an actual board game. First we will import all the necessary libraries. Next we will define the colours blue, black, red and yellow as global static variables. These values are going to be rgb values. We will initialize global variables called ROW\_COUNT and COLUMN\_COUNT. Number of rows are 6 and number of columns are 7.Then we create 5 functions named create\_board( ), drop\_piece( ), is\_valid\_location( ), get\_next\_open\_row( ) and print\_board( ).Then we create a function called winning\_move() and we check for horizontal locations to win, vertical locations to win, positively and negatively sloped diagonals to win. In horizontal and vertical locations, we create a nested for loop for the rows and columns and check an if condition statement to see if the piece has been dropped to that location on the board. If the if condition is satisfied, it will return TRUE. We will repeat the same procedure for vertical locations, positively and negatively sloped diagonals as well. the function def draw\_board( ), pygame.draw is a module for drawing shapes.pygame.draw.rect is used to draw a rectangle. Now we will define the triangle. Define the height and width and the position. So the position is going to be, c\*SQUARESIZE and the position of the y-axis is going to be r\*SQUARESIZE+SQUARESIZE. Height and width are going to be the other two parameters and thats just going to be SQUARESIZE, SQUARESIZE. We will repeat the same procedure for the circle.

**5.3 Important Code Snippets**

**5.3.1 Code Snippet 1**

dpg.create\_context()

dpg.create\_viewport(title='Game', width=700, height=600)

dpg.setup\_dearpygui()

def ModeValue(sender, app\_data):

if app\_data[1]=='1Play':

oneplay()

elif app\_data[1]=='2Play':

twoplay()

elif app\_data[1]=='3Play':

threeplay()

else:

ssoneplay()

with dpg.window(label='Select Mode', modal=True, show=False, tag='b0', no\_title\_bar=True,pos=(280,50)):

dpg.add\_button(label="VS AI", tag='1Play',width=100, height=50)

dpg.add\_button(label="Set Start VS AI", tag='SS1Play',width=100, height=50)

dpg.add\_button(label="2 Players", tag='2Play',width=100, height=50)

dpg.add\_button(label="3 Players", tag='3Play',width=100, height=50)

dpg.add\_button(label="Back", callback=lambda: dpg.configure\_item("b0", show=False),width=100, height=50)

with dpg.window(label='Account', modal=True, show=False, tag='b1', no\_title\_bar=True,pos=(230, 50)):

with dpg.group(horizontal=True):

dpg.add\_text("Name")

InName=dpg.add\_input\_text(default\_value="Player", tag="Pname")

with dpg.group(horizontal=True):

dpg.add\_text("Board Theme")

dpg.add\_button(label="Blue", tag="BColBl", width=100, height=30)

dpg.add\_button(label="Purple", tag="BColPurp", width=100, height=30)

with dpg.group(horizontal=True):

dpg.add\_text("Token color")

dpg.add\_button(label="Red", tag="TColRd",width=100, height=30)

dpg.add\_button(label="Purple", tag="TColPurp",width=100, height=30)

with dpg.group(horizontal=True):

dpg.add\_button(label="Save", tag="SaveChanges",width=100, height=30)

dpg.add\_button(label="Back", callback=lambda: dpg.configure\_item("b1", show=False),width=100, height=30)

a

with dpg.window(label='Leaderboards', modal=True, show=False, tag='b2', no\_title\_bar=True, pos=(300, 50)):

with dpg.group():

with dpg.table(header\_row=True):

dpg.add\_table\_column(label='Name Score')

rows=showldb()

for i in rows:

print(i)

with dpg.table\_row():

dpg.add\_text(f"{i[1]} {i[2]}")

dpg.add\_button(label="Back", callback=lambda: dpg.configure\_item("b2", show=False),width=100, height=30)

with dpg.window(tag="Primary Window", width=700, height=600):

with dpg.group(pos=(300,50)):

title=dpg.add\_text("Connect 4")

dpg.bind\_item\_font(title, default\_font)

dpg.add\_button(label="Play", callback=lambda: dpg.configure\_item('b0', show=True),width=100, height=40)

dpg.add\_button(label="Account", callback=lambda: dpg.configure\_item('b1', show=True),width=100, height=40)

dpg.add\_button(label="Leaderboard", callback=lambda: dpg.configure\_item('b2', show=True), width=100, height=40)

dpg.add\_button(label="Quit", callback=lambda: dpg.configure\_item('Primary Window', show=False),width=100, height=40)

dpg.bind\_font(second\_font)

#========binding item to registry=========

dpg.bind\_item\_handler\_registry('1Play', "widget handler")

dpg.bind\_item\_handler\_registry('SS1Play', "widget handler")

dpg.bind\_item\_handler\_registry('2Play', "widget handler")

dpg.bind\_item\_handler\_registry('3Play', "widget handler")

dpg.bind\_item\_handler\_registry('SaveChanges', "dbvalues")

dpg.bind\_item\_handler\_registry('BColBl', "dbvalues")

dpg.bind\_item\_handler\_registry('BColPurp', "dbvalues")

dpg.bind\_item\_handler\_registry('TColRd', "dbvalues")

dpg.bind\_item\_handler\_registry('TColPurp', "dbvalues")

dpg.show\_viewport()

dpg.set\_primary\_window("Primary Window", True)

dpg.start\_dearpygui()

dpg.destroy\_context()

**5.3.2 Code Snippet 2**

while not game\_over:

for event in pygame.event.get():

if event.type == pygame.QUIT:

sys.exit()

if event.type == pygame.MOUSEMOTION:

pygame.draw.rect(screen, BLACK, (0,0, width, SQUARESIZE))

posx = event.pos[0]

if turn == 0:

pygame.draw.circle(screen, RED, (posx, int(SQUARESIZE/2)), RADIUS)

else:

pygame.draw.circle(screen, YELLOW, (posx, int(SQUARESIZE/2)), RADIUS)

pygame.display.update()

if event.type == pygame.MOUSEBUTTONDOWN:

pygame.draw.rect(screen, BLACK, (0,0, width, SQUARESIZE))

if turn == 0: #RED

posx = event.pos[0]

col = int(math.floor(posx/SQUARESIZE))

if is\_valid\_location(board, col):

row = get\_next\_open\_row(board, col)

drop\_piece(board, row, col, 1)

if winning\_move(board, 1):

label = myfont.render("RED WINS!", 1, RED)

screen.blit(label, (40,10))

game\_over = True

else : #YELLOW

posx = event.pos[0]

col = int(math.floor(posx/SQUARESIZE))

if is\_valid\_location(board, col):

row = get\_next\_open\_row(board, col)

drop\_piece(board, row, col, 2)

if winning\_move(board, 2):

label = myfont.render("YELLOW WINS!", 1, YELLOW)

screen.blit(label, (40,10))

game\_over = True

print\_board(board)

draw\_board(board)

turn += 1

turn = turn % 2

if game\_over:

pygame.time.wait(3000)

**Chapter 6**

**Conclusion and Future Scope**

**6.1 Conclusion**

In conclusion, the Connect 4 game project offers a fun and challenging gaming experience with its classic gameplay mechanics. We have added different modes that will be fun and challenging for the player to play alone or with friends. There is also a level of customizability to make the game more personalized according to the player’s choosing. The interface is kept simple so that new players aren’t overwhelmed and can get easily accustomed to the game.

**6.2 Future Scope**

In the future, the game could be enhanced with improved AI opponents, varied game modes, graphical enhancements, social features, customization options, and regular updates based on player feedback to create a more engaging and dynamic gaming experience.

**References**

1. Real-Time Connect 4 Game Using Artificial Intelligence, 2009, Ahmad M. Sarhan, Adnan Shaout and Michele Shock
2. Alpha-Beta Pruning in Mini-Max Algorithm –An Optimized Approach for a Connect-4 Game, 2018, Rijul Nasa, Rishabh Didwania, Shubhranil Maji, Vipul Kumar
3. Questioning Wizard of Oz: Effects of Revealing the Wizard behind the Robot, 2022, Shano Liang
4. Bridging the Blank: A Defining from Property Uncertainty to Felt Uncertainty in Game, 2022, Jauwairia Nasir, Pierre Oppliger, Barbara Bruno, and Pierre Dillenbourg)
5. helpfulgames.com ’s connect 4 game
6. CBC kids
7. <https://papergames.io/en/connect4>

**Acknowledgement**

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work to-gather. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to appreciate the constant interest and support of our mentor **Mrs. Prachi Verma** in our project and aiding us in developing a flair for the field of Your Domain. We would always cherish the journey of transforming the idea of our project into reality. We would like to show our appreciation to **Mrs. Prachi Verma** for her tremendous support and help, without whom this project would have reached nowhere. We would also like to thank our project coordinator **Mrs. Dakshayani R** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

**Project Group Members:**

Reuben Brian D’silva, 1021135

Jeevan Kunder, 1021163

Aryan Hirlekar, 1021148

Atharv Kadam, 1021156

**Appendix A: Timeline Char****ts**

