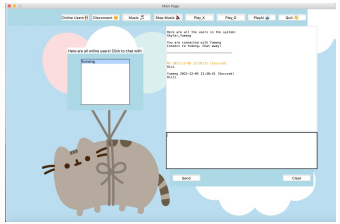
Cat Theme Chat System

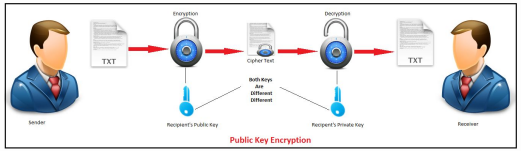
ICS Fall 2021

Presented by:
Skyler Chen
Shavarsh Melikyan
Arghya Sarkar

1 Introduction & Overview



Main Functions Overview

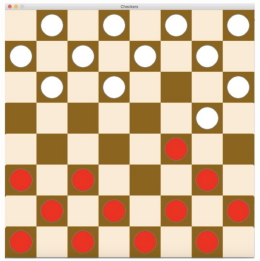


GUI

Cat theme interface designed for cat lovers

Secure Msg

Use public key & private key to secure messages



Multiplayer Game

Multiplayer tic-tac-toe game

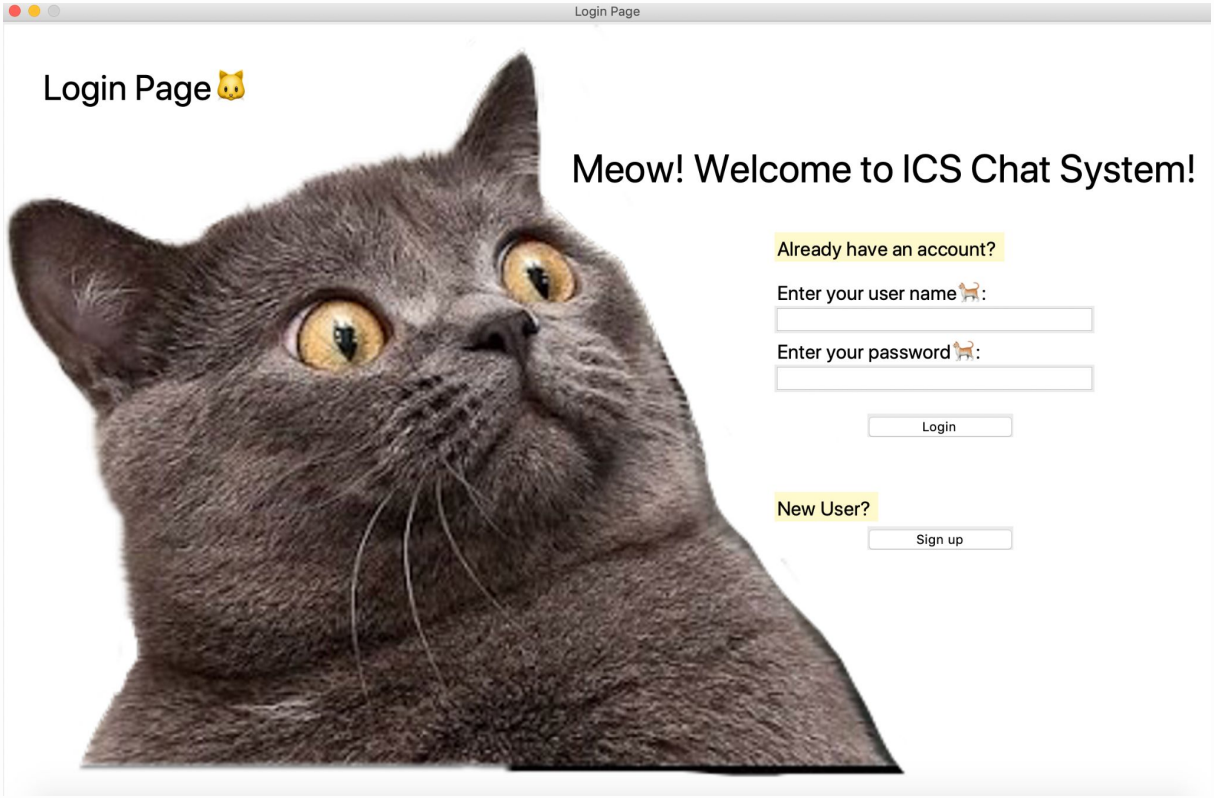
Game with AI

Single player checker game with AI

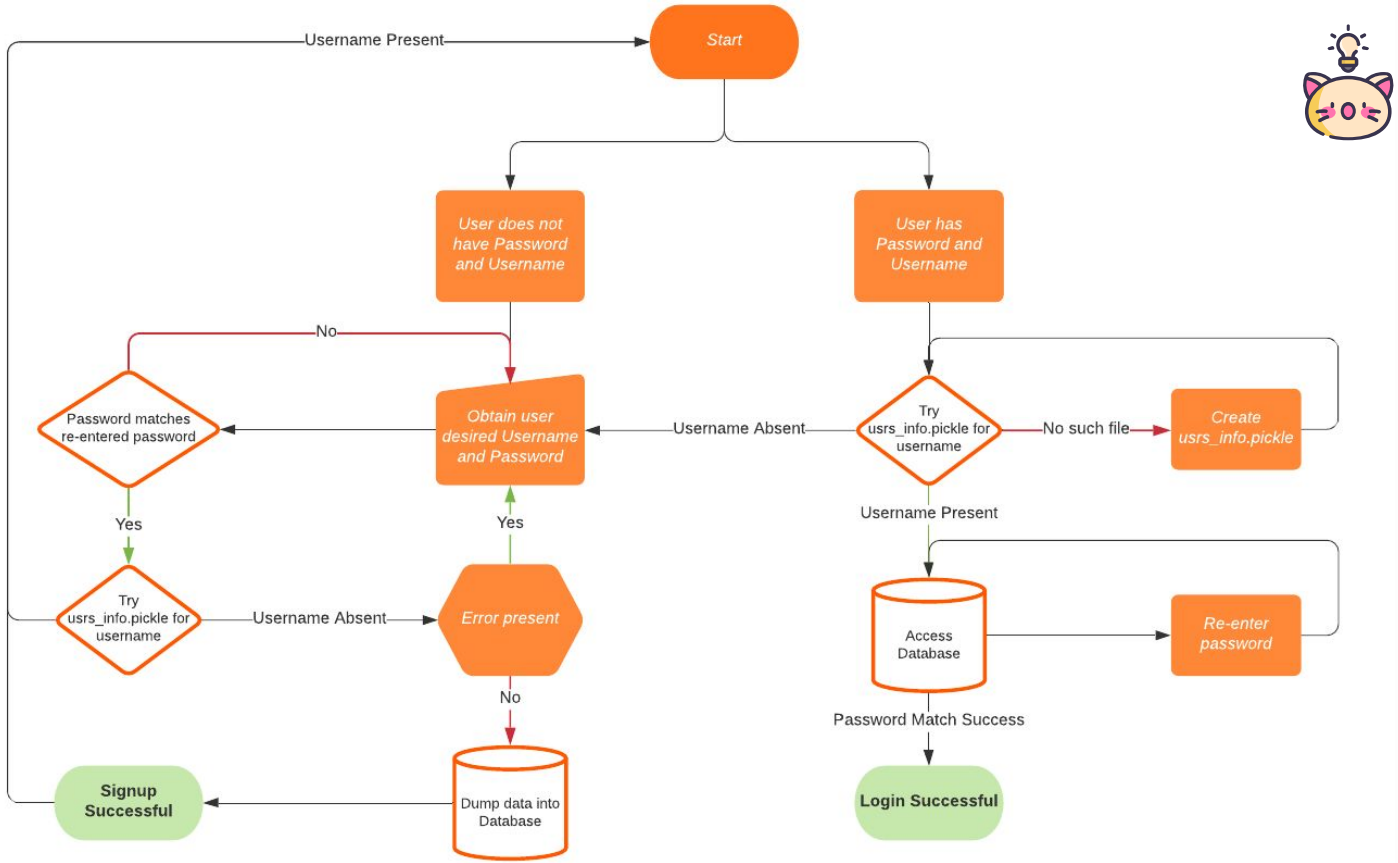
2
Discussion & Codes Explain

Let’s take a closer look!

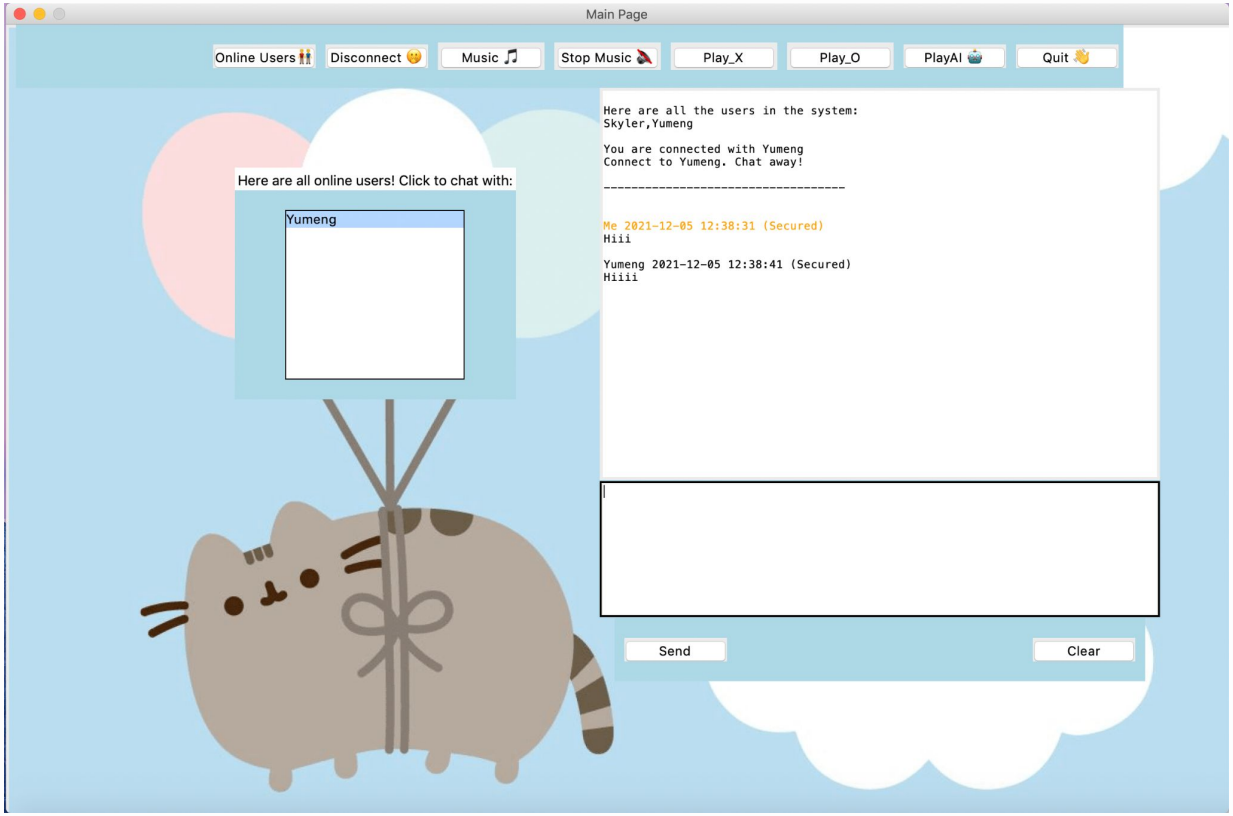
Login Page (GUI)



Login Loop


Start
Username Present
User does not have Password and Username
User has Password and Username
No
Password matches re-entered password
Obtain user desired Username and Password
Username Absent
Try usrs_info.pickle for username
No such file
Create usrs_info.pickle
Yes
Yes
Username Present
Try usrs_info.pickle for username
Username Absent
Error present
Access Database
Re-enter password
No
Password Match Success
Signup Successful
Dump data into Database
Login Successful

Main Chatting Page (GUI)




```python
def online_update(): #update
    if client.sm.get_state() != S_OFFLINE:
        onlinedict=client.sm.whoOnLine
        # print(onlinedict)
        # onlinedict = onlinedict.split(",")

    else:
        onlinedict = {}
    whoOnline.set(list(onlinedict))
    whoListBox.after(1000, online_update)

online_update()
```

Packages used:
socket , tkinter, threading,
runpy, argparse, time, pickle

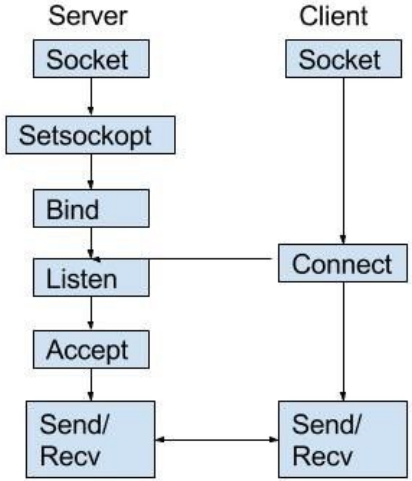
There is also a nice BGM!

Socket & Base Chat

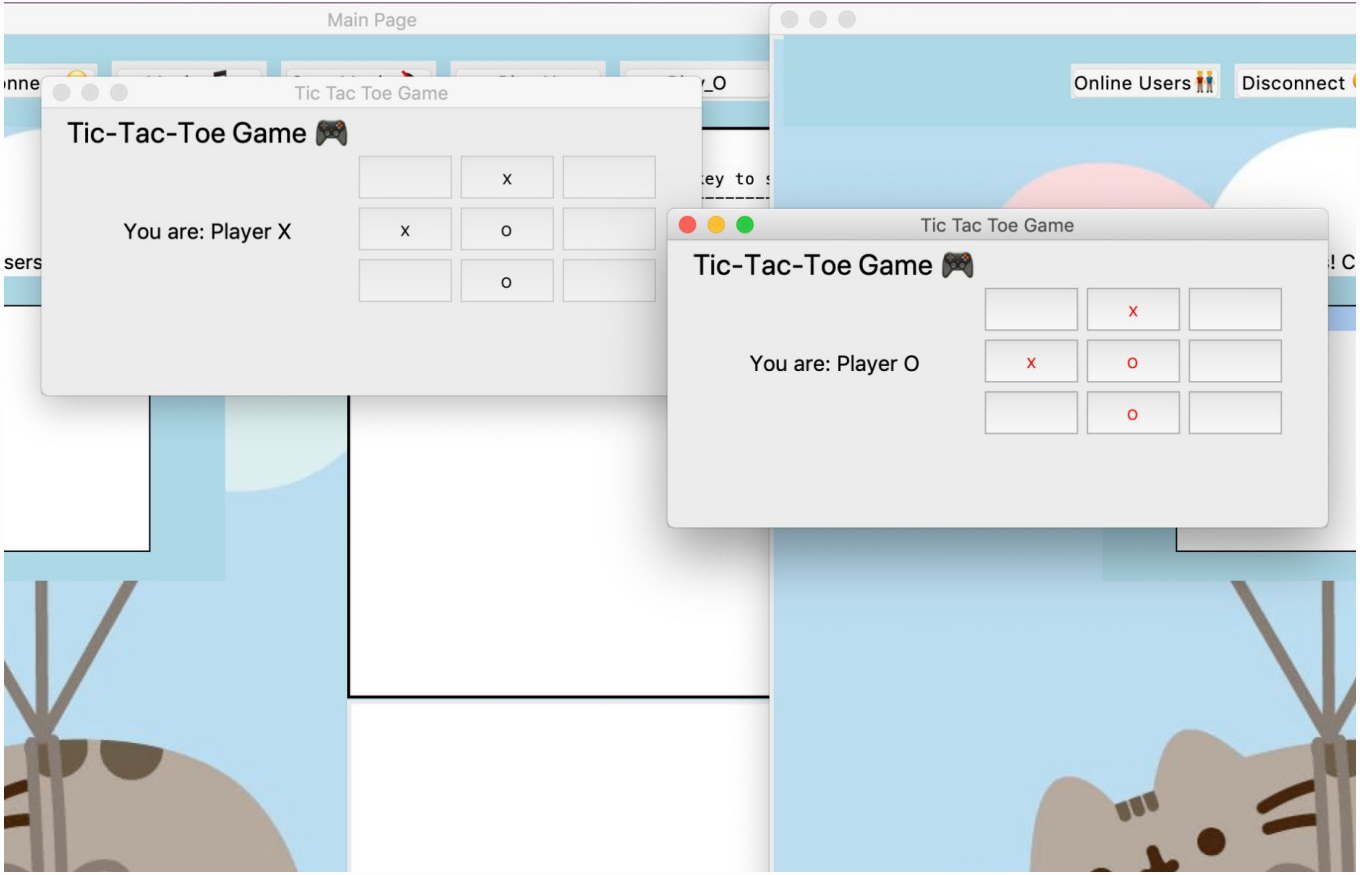
```
def init_chat(self):
    self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    svr = SERVER if self.args.d == None else (self.args.d, CHAT_PORT)
    self.socket.connect(svr)
    self.sm = csm.ClientSM(self.socket)
    reading_thread = threading.Thread(target=self.read_input)
    reading_thread.daemon = True
    reading_thread.start()
```

Refers to the address-family ipv4

Connection-oriented TCP protocol



Game 1 - Multiplayer tic-tac-toe (Tkinter Window)



Game 1 - Multiplayer tic-tac-toe (Tkinter Window)

Command: firstClick()

Command: secondClick()

etc... All similar structures within function

1	2	3
4	5	6
7	8	9

```
turn = True
def firstClick():
    global turn
    if turn == True and firstBtn["text"] == " ":
        firstBtn["text"] = 'X'
        value = '1'
        client.send(value.encode('utf-8'))
        turn = False
        check()
```

Display the choice on own screen

Encode and send choice to component

Set turn to F so it's the other's turn (players take turn)

Packages used:
socket, tkinter, threading,

Game 1 - Multiplayer tic-tac-toe (Tkinter Window)

How to display choices on the other's screen?

```
def receive_thread(client):
    global turn
    while True:
        message = client.recv(2048).decode('utf-8')
        if message == '1' and firstBtn["text"] == " ":
            firstBtn["text"] = 'X'
            turn = True
        elif message == '2' and secondBtn["text"] == " ":
            secondBtn["text"] = 'X'
            turn = True
        elif message == '3' and thirdBtn["text"] == " ":
            thirdBtn["text"] = 'X'
            turn = True
```

Decode message received

Display component's choice

Set turn to T after receiving msg, so that the receiver can make a choice now

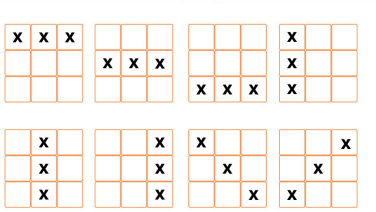
Game 1 - Multiplayer tic-tac-toe (Tkinter Window)

```
def check():
    global flag
    b1 = firstBtn["text"]
    b2 = secondBtn["text"]
    b3 = thirdBtn["text"]
    b4 = fourthBtn["text"]
    b5 = fifthBtn["text"]
    b6 = sixthBtn["text"]
    b7 = seventhBtn["text"]
    b8 = eighthBtn["text"]
    b9 = ninthBtn["text"]

    flag = flag + 1
    winner = 0

    if ((b1 == b2 and b2 == b3 and b1 == 'O') or
        (b1 == b2 and b2 == b3 and b1 == 'X')):
        winner = 1
        win(b1)
```

How to determine who wins?

```
def win(player):
    if player == 'X':
        playerNumber = 1
    else:
        playerNumber = 2
    messagebox.showinfo(f"CONGRATULATIONS",
    window.destroy()
```

Check whether X or O is at the winning box



List all possible winning outcomes

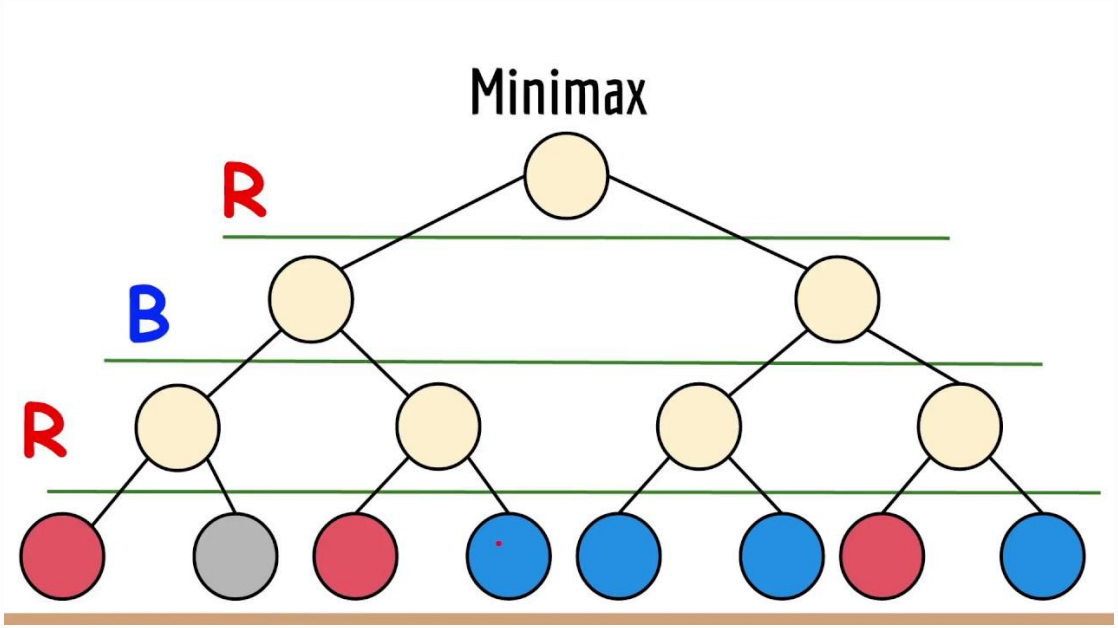
If all nine boxes taken and none of winning outcomes satisfied, no one wins

```
if flag == 9 and winner != 1:
    messagebox.showinfo("Dead End!!", "No one can win now!")
    window.destroy()
```

Game 2 - Checker game with AI (Pygame with minimax function)

First of all, What is Minimax?

Packages used:
Pygame, minimax



- **Recursive Call**
- **Maximization & Minimization**
- **Node Checking**
- ***Alpha-Beta Pruning**

Game 2 - Checker game with AI (Pygame with minimax function)

First of all, What is Minimax?

```python
def minimax(position, depth, max_player, game):
    if depth == 0 or position.winner() != None:
        return position.evaluate(), position

    if max_player:
        max_eval = float('-inf')
        best_move = None
        for move in get_all_moves(position, WHITE, game):
            evaluation = minimax(move, depth-1, False, game)[0]
            max_eval = max(max_eval, evaluation)
            if max_eval == evaluation:
                best_move = move

        return max_eval, best_move
    else:
        min_eval = float('inf')
        best_move = None
        for move in get_all_moves(position, RED, game):
            evaluation = minimax(move, depth-1, True, game)[0]
            min_eval = min(min_eval, evaluation)
            if min_eval == evaluation:
                best_move = move

        return min_eval, best_move
```

- **Recursive Call**
- **Maximization & Minimization**
- **Node Checking**
- ***Alpha-Beta Pruning**

Game 2 - Checker game with AI (Pygame with minimax function)

For Checkers we actually need to make 3 classes:

- **Board Class, responsible for functions related to the game's board (drawing squares, creating board, moving pieces etc.)**
- **Piece Class, responsible for pieces on a board (pieces' position calculation, pieces' movements, drawing an other functions)**
- **Game class, responsible for game-related functions' realization (game update after each move, resetting, valid-move drawing AKA possible moves etc.)**

Main working file: main_checkers.py, addition: constants' file

```python
import pygame
from checkers.constants import WIDTH, HEIGHT, SQUARE_SIZE, RED, WHITE
from checkers.game import Game
from minimax.algorithm import minimax

FPS = 60

WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Checkers')

def get_row_col_from_mouse(pos):
    a, b = pos
    row = b // SQUARE_SIZE
    col = a // SQUARE_SIZE
    return row, col

def main():
    run = True
    clock = pygame.time.Clock()
    game = Game(WIN)

    while run:
        clock.tick(FPS)

        if game.turn == WHITE:
            value, new_board = minimax(game.get_board(), 4, WHITE, game)
            game.ai_move(new_board)

        if game.winner() != None:
            print(game.winner())
            run = False

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                row, col = get_row_col_from_mouse(pos)
                game.select(row, col)

        game.update()

    pygame.quit()

main()
```

Game 2 - Checker game with AI (Pygame with minimax function)

board.py

```python
import pygame
from .constants import BLACK, ROWS, RED, SQUARE_SIZE, COLS, WHITE, GREY, AW, ROD
from .piece import Piece

class Board:
    def __init__(self):
        self.board = []
        self.red_left = self.white_left = 12
        self.red_kings = self.white_kings = 0
        self.create_board()

    def draw_squares(self, win):
        win.fill(ROD)
        for row in range(ROWS):
            for col in range(row % 2, COLS, 2):
                pygame.draw.rect(win, AW, (row*SQUARE_SIZE, col *SQUARE_SIZE, SQUARE_SIZE,

    def evaluate(self):
        return self.white_left - self.red_left + (self.white_kings * 0.5 - self.red_kings *

    def get_all_pieces(self, color):
        pieces = []
        for row in self.board:
            for piece in row:
                if piece != 0 and piece.color == color:
                    pieces.append(piece)
        return pieces

    def move(self, piece, row, col):
        self.board[piece.row][piece.col], self.board[row][col] = self.board[row][col], sel
        piece.move(row, col)

        if row == ROWS - 1 or row == 0:
            piece.make_king()
            if piece.color == WHITE:
                self.white_kings += 1
            else:
                self.red_kings += 1

    def get_piece(self, row, col):
        return self.board[row][col]
```

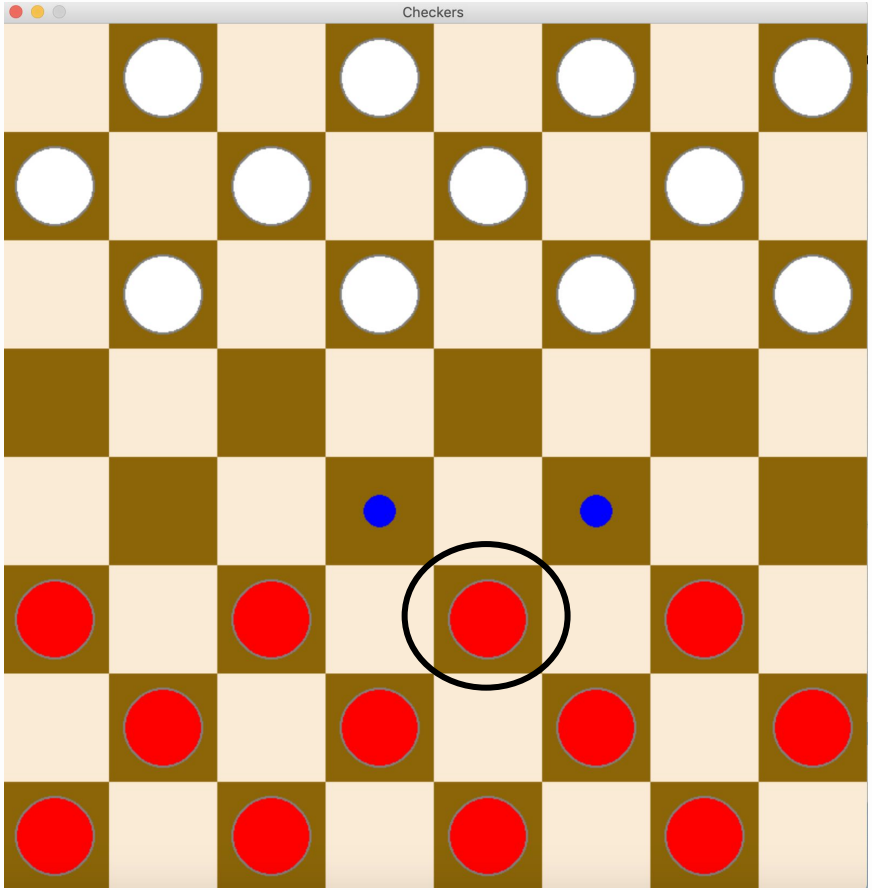
piece.py

```python
from .constants import RED, WHITE, SQUARE_SIZE, GREY, CROWN
import pygame

class Piece:
    PADDING = 15
    OUTLINE = 2

    def __init__(self, row, col, color):
        self.row = row
        self.col = col
        self.color = color
        self.king = False
        self.x = 0
        self.y = 0
        self.calc_pos()

    def calc_pos(self):
        self.x = SQUARE_SIZE * self.col + SQUARE_SIZE // 2
        self.y = SQUARE_SIZE * self.row + SQUARE_SIZE // 2

    def make_king(self):
        self.king = True

    def draw(self, win):
        radius = SQUARE_SIZE//2 - self.PADDING
        pygame.draw.circle(win, GREY, (self.x, self.y), radius + self.OUTLINE)
        pygame.draw.circle(win, self.color, (self.x, self.y), radius)
        if self.king:
            win.blit(CROWN, (self.x - CROWN.get_width()//2, self.y - CROWN.get_height()

    def move(self, row, col):
        self.row = row
        self.col = col
        self.calc_pos()

    def __repr__(self):
        return str(self.color)
```

game.py

```python
import pygame
from .constants import RED, WHITE, BLUE, SQUARE_SIZE
from checkers.board import Board

class Game:
    def __init__(self, win):
        self._init()
        self.win = win

    def update(self):
        self.board.draw(self.win)
        self.draw_valid_moves(self.valid_moves)
        pygame.display.update()

    def _init(self):
        self.selected = None
        self.board = Board()
        self.turn = RED
        self.valid_moves = {}

    def winner(self):
        return self.board.winner()

    def reset(self):
        self._init()

    def select(self, row, col):
        if self.selected:
            result = self._move(row, col)
            if not result:
                self.selected = None
                self.select(row, col)

        piece = self.board.get_piece(row, col)
        if piece != 0 and piece.color == self.turn:
            self.selected = piece
            self.valid_moves = self.board.get_valid_moves(piece)
            return True

        return False
```

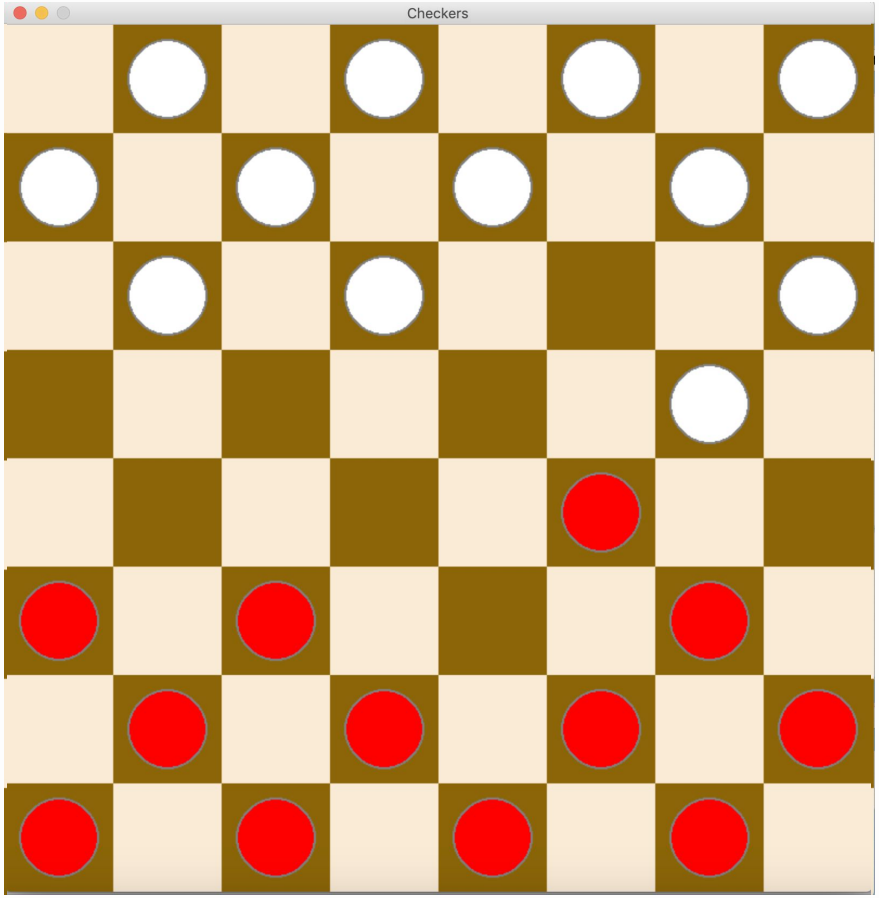
Game 2 - Checker game with AI (Pygame with minimax function)



AI side

Player side when click

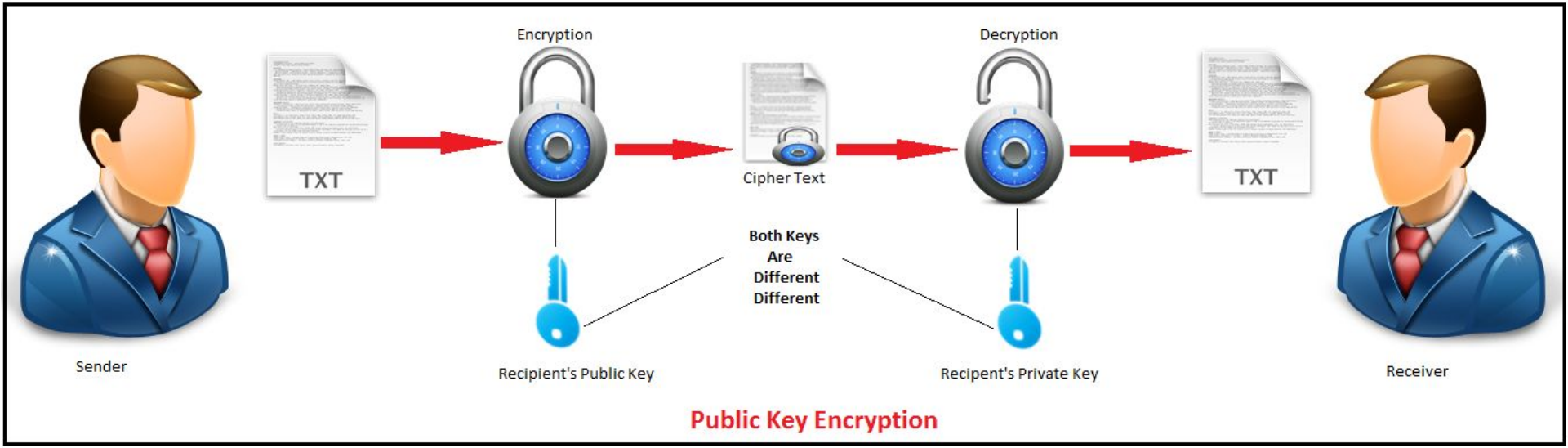
When AI made its decision....



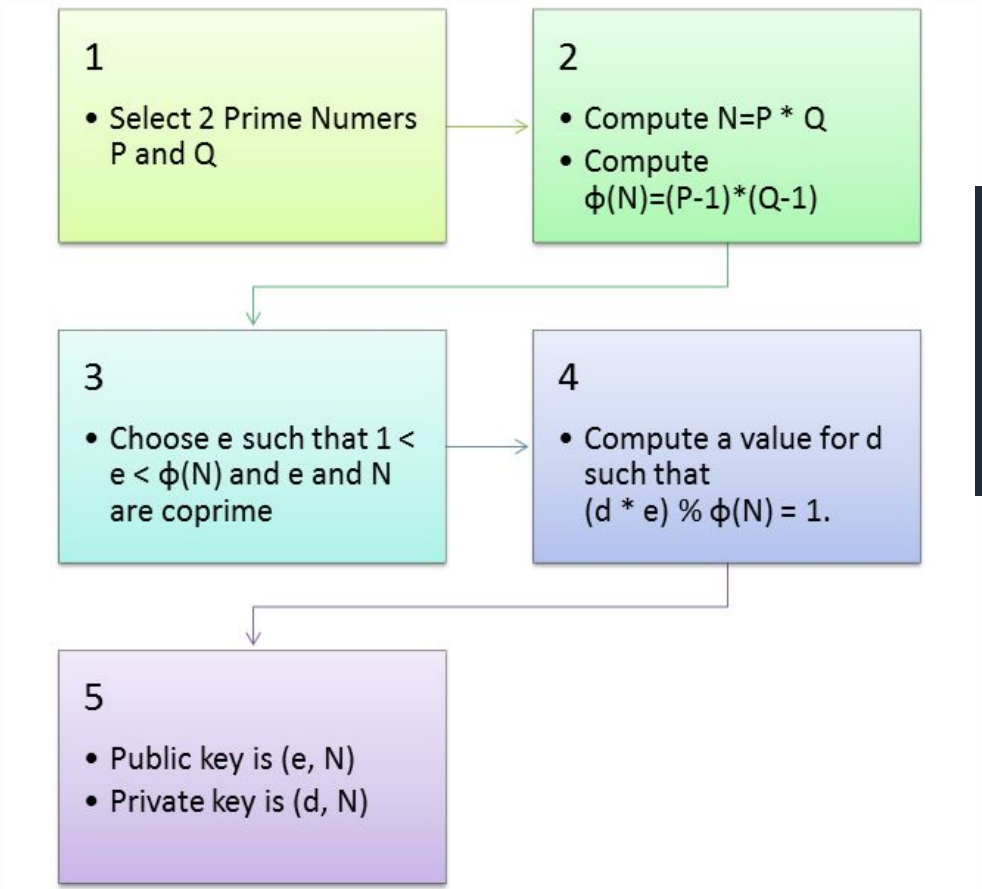
Game 2 - Checker game with AI (Pygame with minimax function)

1. Doing our step
2. Computer scores possible movements for each case using Minimax Algorithm
3. Basing on the Algorithm's outcome the choice is being taken

Secure Message Basic Idea RSA



Generation of Private Key and Public Key in RSA



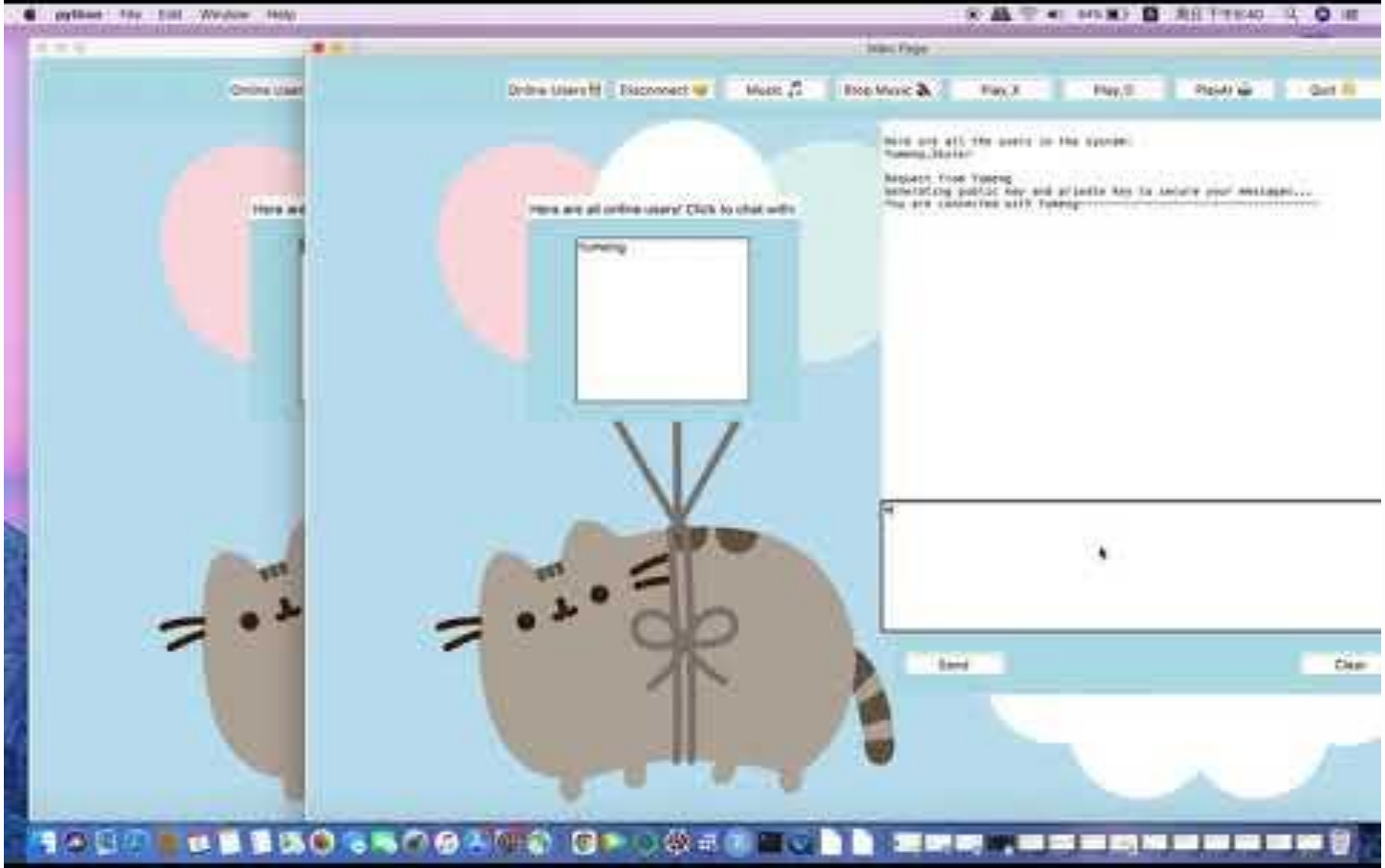
```
def main():
    public_key, private_key = generate_key()
    print(public_key, private_key)
    message = "hi"
    print(message)
    encrypted_message = encryption(message, public_key)
    print('Here is the encrypted message: ', end = '')
    print(encrypted_message)
    received_message = decryption(encrypted_message, private_key)
    print('Here is the message you receive: ' +  received_message)

if __name__ == "__main__":
    main()
```

Secure Message Basic Idea RSA

```python
import random
import time

#Greatest Common Divisor
def gcd(x, y):
    while y != 0:
        x, y = y, x % y
    return x


def qck_pow(a,b,n):
    if b == 0:
        return 1
    if b == 1:
        return int(a)**b%n
    if b%2 == 0:
        return (qck_pow(int(a),b//2,n)**2)%n
    return (a * (qck_pow(int(a),b//2,n)**2)%n)%n

#Determining the the multiplicative inverse of two relative p[rime numbers
def mlt_inv(m,n):
    if gcd(m,n) != 1:
        return None
    i,j,k = 1,0,m
    x,y,z = 0,1,n
    while z != 0:
        q = k//z
        x,y,z,i,j,k = (i-q*x), (j-q*y),(k-q*z),x,y,z
    return i % m

#Determining whether the number is a prime number in O(n**0.5) runnning time
def is_prime(k):
    for i in range(2, int(k ** 0.5) + 1):
        if k % i == 0:
            return False
    return True

def generate_prime_list(m, n):
    prime_list = []
    for i in range(m, n):
        if is_prime(i):
            prime_list.append(i)
    return prime_list

prime_list = generate_prime_list(100, 10000)
```

```python
def generate_key():
    p = random.choice(prime_list)
    q = random.choice(prime_list)
    while p == q:
        p = random.choice(prime_list)
    n = p*q
    # calculate euler function phi(n)
    phi = (p-1) * (q-1)
    # Choose an integer e that is relative prime with phi(n)
    e = random.randint(1, phi)
    while gcd(e, phi) != 1:
        e = random.randrange(1, phi)
    # Use Extended Euclid's Algorithm to generate the private key
    d = mlt_inv(e, phi) #d = (p-1)*(q-1) - 1
    public_key = (e,n)
    private_key = (d,n)
    return public_key, private_key

#encryption
def encryption(message,public_key):
    e,n = int(public_key[0]), int(public_key[1])
    encrypted_number = []
    #message += "xxx"
    for i in message:
        encrypted_number.append(qck_pow(ord(i),e,n))
    print(str(encrypted_number))
    return str(encrypted_number)

#decryption
def decryption(message,private_key):
    d,n = int(private_key[0]), int(private_key[1])
    decrypted_message = []
    if len(message) > 0:
        message = str(message)
    message = message[1:-1]
    message = message.split(",")

    for i in message:
        t = int(i)
        decrypted_message.append(chr(qck_pow(t,d,n)))
    message_text = "".join(decrypted_message)
    return message_text
```

3
Live Show Case


Online Users
Disconnect
Music
Stop Music
Play X
Play O
Pickle
Quit
Here are all online users! Click to chat with:
Send
Clear

4
Analysis

Logic & Improvement

Design Logic

To attract users, we made:

- Fun interface, login/signup page with clear instructions, bgm...
- Encryption to secure users’ messages
- Online & offline game for leisure :)
- Clear and user-friendly structure overall

Potential Improvements

Future plan

- Better interface for tic-tac-toe game. Implement Minimax Algorithm
- Request function: only chatting when request is accepted
- Potential future games include 2+ players
- AI Checkers: Alpha-Beta Pruning Algorithm + Reinforcement Learning Techniques
- Change existing libraries with the better ones right after examining other libraries

Thank You