FIFA World Cup Quiz Game – Socket Programming Project

CYSE 250: Cybersecurity Programming and Networking

Emmanuel Dade

FIFA World Cup Quiz Game – Project Idea

What It Does

- Lets users play a trivia game about FIFA World Cup history.
- Users answer multiple-choice questions about teams, players, and tournaments.
- Shows the user's final score at the end.

How It Works

- User connects to the quiz server.
- Log in or create a new account.
- Server sends one question at a time.
- User picks an answer.
- Server checks correctness and updates score.
- Repeat for 10 questions.
- Server shows final score.
- Will give option to play again or end

Source Code

encryption.py

```
# encryption.py - Simple Caesar Cipher with shift
def encrypt_message(message, shift=3):
    """Encrypt message using Caesar cipher"""
    encrypted = ""
    for char in message:
        if char.isalpha():
            ascii_offset = 65 if char.isupper() else 97
            encrypted += chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
        else:
            encrypted += char
    return encrypted

def decrypt_message(encrypted_msg, shift=3):
    """Decrypt message using Caesar cipher"""
    return encrypt_message(encrypted_msg, -shift)

def hash_password(password):
    """Create a simple hash for password storage"""
    hashed = 0
    for char in password:
        hashed = (hashed * 31 + ord(char)) % (10**9 + 7)
    return str(hashed)
```

server.py

```
# server.py
import socket
import threading
from encryption import encrypt_message, decrypt_message, hash_password

# Server configuration
HOST = '127.0.0.1'
PORT = 5555
USERS_FILE = 'users.txt'
QUESTIONS_FILE = 'questions.txt'

# Global data structures
users_db = {}  # Dictionary: {username: hashed_password}
questions_list = []  # List of question dictionaries

def load_users():
    """Load users from file into dictionary"""
    try:
        with open(USERS_FILE, 'r') as f:
            for line in f:
                if ':' in line:
                    username, password = line.strip().split(':')
                    users_db[username] = password
        print(f"Loaded {len(users_db)} users from database")
    except FileNotFoundError:
        print("No existing user database found. Creating new one.")
        open(USERS_FILE, 'w').close()

def save_user(username, password):
    """Save new user to file"""
    with open(USERS_FILE, 'a') as f:
        f.write(f"{username}:{password}\n")

def load_questions():
    """Load questions from file into list of dictionaries"""
    try:
        with open(QUESTIONS_FILE, 'r') as f:
            for line in f:
                parts = line.strip().split('|')
                if len(parts) == 6:
                    question_dict = {
                        'question': parts[0],
                        'options': [parts[1], parts[2], parts[3], parts[4]],
                        'correct': parts[5]
                    }
                    questions_list.append(question_dict)
        print(f"Loaded {len(questions_list)} questions from database")
    except FileNotFoundError:
        print("ERROR: questions.txt not found!")

def authenticate_user(username, password):
    """Check if user credentials are valid"""
    hashed_pw = hash_password(password)
    if username in users_db:
        # Returning user
        if users_db[username] == hashed_pw:
            return "LOGIN_SUCCESS"
        else:
            return "WRONG_PASSWORD"
    else:
        # New user - create account
        users_db[username] = hashed_pw
```

```
        save_user(username, hashed_pw)
        return "NEW_USER_CREATED"

def handle_client(client_socket, address):
    """Handle individual client connection"""
    print(f"[NEW CONNECTION] {address} connected")

    try:
        # Authentication phase
        while True:
            # Receive encrypted credentials
            encrypted_data = client_socket.recv(1024).decode('utf-8')
            credentials = decrypt_message(encrypted_data)

            if '|' not in credentials:
                continue

            username, password = credentials.split('|')
            auth_result = authenticate_user(username, password)

            # Send encrypted response
            response = encrypt_message(auth_result)
            client_socket.send(response.encode('utf-8'))

            if auth_result in ["LOGIN_SUCCESS", "NEW_USER_CREATED"]:
                print(f"[AUTH SUCCESS] {username} authenticated")
                break

        # Game phase
        score = 0
        total_questions = min(10, len(questions_list))

        for i in range(total_questions):
            question = questions_list[i]

            # Prepare question message
            question_msg = f"{question['question']}|"
            question_msg += "|".join(question['options'])

            # Send encrypted question
            encrypted_q = encrypt_message(question_msg)
            client_socket.send(encrypted_q.encode('utf-8'))

            # Receive encrypted answer
            encrypted_answer = client_socket.recv(1024).decode('utf-8')
            answer = decrypt_message(encrypted_answer)

            # Check answer (handle letter inputs)
            correct_answer = question['correct'].lower()
            if answer.lower() == correct_answer:
                score += 1
                result = "CORRECT"
            else:
                result = f"WRONG|{question['correct']}"

            # Send encrypted result
            encrypted_result = encrypt_message(result)
            client_socket.send(encrypted_result.encode('utf-8'))

        # Send final score
        final_msg = f"GAME_OVER|{score}|{total_questions}"
        encrypted_final = encrypt_message(final_msg)
        client_socket.send(encrypted_final.encode('utf-8'))

        print(f"[GAME COMPLETE] {username} scored {score}/{total_questions}")
```

```
    except Exception as e:
        print(f"[ERROR] {address}: {e}")
    finally:
        client_socket.close()
        print(f"[DISCONNECTED] {address}")

def start_server():
    """Initialize and start the server"""
    load_users()
    load_questions()

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((HOST, PORT))
    server.listen()

    print(f"[LISTENING] Server is listening on {HOST}:{PORT}")

    while True:
        client_socket, address = server.accept()
        thread = threading.Thread(target=handle_client, args=(client_socket, address))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.active_count() - 1}")

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

client.py

```
# client.py
import socket
from encryption import encrypt_message, decrypt_message

HOST = '127.0.0.1'
PORT = 5555

def display_menu():
    """Display welcome menu"""
    print("\n" + "="*50)
    print("   FIFA WORLD CUP TRIVIA GAME")
    print("="*50)

def authenticate(client_socket):
    """Handle user authentication"""
    while True:
        print("\n--- LOGIN / REGISTER ---")
        username = input("Enter username: ").strip()
        password = input("Enter password: ").strip()

        if not username or not password:
            print("Username and password cannot be empty!")
            continue

        # Send encrypted credentials
        credentials = f"{username}|{password}"
        encrypted_creds = encrypt_message(credentials)
        client_socket.send(encrypted_creds.encode('utf-8'))

        # Receive encrypted response
        encrypted_response = client_socket.recv(1024).decode('utf-8')
        response = decrypt_message(encrypted_response)

        if response == "LOGIN_SUCCESS":
            print(f"\n✓ Welcome back, {username}!")
            return True
        elif response == "NEW_USER_CREATED":
```

```
            print(f"\n✓ Account created! Welcome, {username}!")
            return True
        elif response == "WRONG_PASSWORD":
            print("\n✗ Incorrect password. Try again.")
        else:
            print("\n✗ Authentication failed. Try again.")

def play_game(client_socket):
    """Main game loop"""
    question_num = 1

    while True:
        # Receive encrypted question
        encrypted_question = client_socket.recv(2048).decode('utf-8')
        question_data = decrypt_message(encrypted_question)

        # Check if game is over
        if question_data.startswith("GAME_OVER"):
            parts = question_data.split('|')
            score = parts[1]
            total = parts[2]
            print("\n" + "="*50)
            print(f"   GAME OVER!")
            print(f"   Your Final Score: {score}/{total}")
            print(f"   Percentage: {int(score)/int(total)*100:.1f}%")
            print("="*50)
            break

        # Parse question
        parts = question_data.split('|')
        question_text = parts[0]
        options_list = parts[1:5]

        # Display question
        print("\n" + "-"*50)
        print(f"Question {question_num}:")
        print(question_text)
        print()

        for idx, option in enumerate(options_list):
            letter = chr(97 + idx)  # 97 is ASCII for 'a'
            print(f"  {letter}. {option}")

        # Get user answer
        while True:
            answer = input("\nYour answer (a/b/c/d): ").strip().lower()
            if answer in ['a', 'b', 'c', 'd']:
                break
            print("Please enter a letter: a, b, c, or d.")

        # Send encrypted answer
        encrypted_answer = encrypt_message(answer)
        client_socket.send(encrypted_answer.encode('utf-8'))

        # Receive encrypted result
        encrypted_result = client_socket.recv(1024).decode('utf-8')
        result = decrypt_message(encrypted_result)

        # Display result
        if result == "CORRECT":
            print("\n✓ Correct! Well done!")
        else:
            correct_answer = result.split('|')[1]
            print(f"\n✗ Wrong! The correct answer was option {correct_answer}.")
        question_num += 1
```

```
def main():
    """Main client function"""
    display_menu()

    while True:
        try:
            # Connect to server
            client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            client_socket.connect((HOST, PORT))
            print(f"\n[CONNECTED] Connected to server at {HOST}:{PORT}")

            # Authenticate
            if authenticate(client_socket):
                input("\nPress Enter to start the quiz...")
                play_game(client_socket)

            # Ask to play again
            play_again = input("\nPlay again? (yes/no): ").strip().lower()
            client_socket.close()

            if play_again not in ['yes', 'y']:
                print("\nThanks for playing! Goodbye!")
                break

        except ConnectionRefusedError:
            print("\n[ERROR] Could not connect to server. Is it running?")
            break
        except Exception as e:
            print(f"\n[ERROR] {e}")
            break

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

Question Bank (questions.txt)

```
Which country won the first FIFA World Cup in 1930?|Uruguay|Brazil|Argentina|Italy|a
Who is the all-time top scorer in FIFA World Cup history?|Pele|Ronaldo|Miroslav Klose|Diego Maradona|c
How many times has Brazil won the FIFA World Cup?|4 times|5 times|6 times|7 times|b
In which year did Egypt first participate in the FIFA World Cup?|1934|1950|1962|1970|a
Which country hosted the 2018 FIFA World Cup?|Brazil|Germany|Russia|Qatar|c
Who won the Golden Boot at the 2022 FIFA World Cup?|Lionel Messi|Kylian Mbappe|Cristiano Ronaldo|Neymar|b
Which team has appeared in the most FIFA World Cup finals?|Brazil|Germany|Argentina|Italy|b
In which year did Diego Maradona score the famous Hand of God goal?|1982|1986|1990|1994|b
What is the only African nation to reach the FIFA World Cup quarterfinals in 1990?|Nigeria|Cameroon|Senegal|Egypt|b
How many teams participated in the FIFA World Cup finals tournament in 2022?|24|32|48|64|b
```

Assignment Instructions and Rubric

CYSE-250 Cybersecurity Programming and Networking

Total Points: 100

Learning Objective: The project will provide you with the opportunity to demonstrate your understanding of the applied aspect of Python and socket programming in your concentration.

1. The project will be related to the contents of this course. This can be an individual or a group project. A group of maximum two students will develop and design their own project.
2. The group should find a related topic to socket programming.
3. You MUST include encryption and decryption algorithms to send and receive messages.

4. You MUST authenticate the user login information (username and password) if it is a returned user or create a new user account.
5. You MUST show up and present your work during the lecture time.
6. You MUST run your code and demonstrate it, explain the steps included in your project.
7. The presentation MUST include following sections:
   • The project's description.
   • The project's goal. Your project should have a goal. (educational, sales, entertainment, security)
   • Simple chatting applications are NOT considered for this project and will NOT be accepted.
8. Your code MUST include the following:
   • Loops
   • Functions
   • Lists and Dictionaries
   • Socket Programming
   • Files
   • Encryption and Decryption
   • Authentication
9. You must submit your code on Canvas.

(No submission No grade)
(No presentation No grade)
(No show up No grade)
(No Socket Programming No grade)

Grading Rubrics (100 points)

- (40 Points) Show up, Discussion and Presentation
- (40 Points) Running Program including socket programming including Files, Functions, Lists and dictionaries (*.py file).
- (10 Points) Encryption and Decryption
- (10 points) Authentication