```
#Server
import socket
import threading
import json
import os
import time

SERVER_HOST = 'localhost'
SERVER_PORT = 5000
USERS_FILE = "users.json"
VOTES_FILE = "votes.json"
BUFSIZE = 1024
SHIFT_KEY = 5  #ASCII Shift

#Encryption
def ascii_encrypt(text):
    return ''.join(chr((ord(char) + SHIFT_KEY) % 256) for char in text)
#Decryption
def ascii_decrypt(text):
    return ''.join(chr((ord(char) - SHIFT_KEY) % 256) for char in text)

if not os.path.exists(USERS_FILE):
    with open(USERS_FILE, 'w') as file:
        json.dump({}, file)

if not os.path.exists(VOTES_FILE):
    with open(VOTES_FILE, 'w') as file:
        json.dump({"tally": {}, "voters": []}, file)

#Server Used Functions
def register_user(username, password):
    with open(USERS_FILE, 'r+') as file:
        users = json.load(file)
        if username in users:
            return "Another account is using this username already!"
        users[username] = ascii_encrypt(password)
        file.seek(0)
        json.dump(users, file, indent=4)
        return "Registration successful!"

def authenticate_user(username, password):
    with open(USERS_FILE, 'r') as file:
        users = json.load(file)
    return users.get(username) == ascii_encrypt(password)

def initialize_votes_file():
    if not os.path.exists(VOTES_FILE):
        with open(VOTES_FILE, 'w') as file:
            json.dump({"tally": {}, "voters": []}, file)
    else:
        try:
            with open(VOTES_FILE, 'r') as file:
                data = json.load(file)
                if not isinstance(data, dict) or "tally" not in data or "voters"
not in data:
                    raise ValueError("Error")
        except (json.JSONDecodeError, ValueError):
            with open(VOTES_FILE, 'w') as file:
                json.dump({"tally": {}, "voters": []}, file)
```

```
def record_vote(username, candidate):
    try:
        with open(VOTES_FILE, 'r+') as file:
            data = json.load(file)
            if username in data["voters"]:
                return "You cannot vote twice! Please exit from the server."

            data["voters"].append(username)
            if candidate in data["tally"]:
                data["tally"][candidate] += 1
            else:
                data["tally"][candidate] = 1
            file.seek(0)
            json.dump(data, file, indent=4)
            return "Your vote has successfully been cast! Please exit from the
server."
    except Exception as e:
        return f"Error recording vote: {str(e)}"

def get_tally():
    try:
        with open(VOTES_FILE, 'r') as file:
            data = json.load(file)
        return data["tally"]
    except Exception as e:
        print(f"Error reading tally: {str(e)}")
        return {}


def client_server_connection(conn, addr):
    print(f"{addr} connected successfully.")
    try:
        while True:
            try:
                data = conn.recv(BUFSIZE).decode()
                if not data:
                    break
                request = json.loads(data)

                if request["action"] == "register":
                    username, password = request["username"], request["password"]
                    message = register_user(username, password)
                    conn.send(json.dumps({"message": message}).encode())

                elif request["action"] == "login":
                    username, password = request["username"], request["password"]
                    success = authenticate_user(username, password)
                    conn.send(json.dumps({"status": "success" if success else
"failure"}).encode())

                elif request["action"] == "vote":
                    username, candidate = request["username"], request["candidate"]
                    message = record_vote(username, candidate)
                    conn.send(json.dumps({"message": message}).encode())

                elif request["action"] == "tally":
                    tally = get_tally()
                    conn.send(json.dumps({"tally": tally}).encode())
```

```
            except socket.timeout:
                print(f"Connection with {addr} timed out.")
                break
    except Exception as e:
        print(f"Exception with {addr}: {str(e)}")
    finally:
        conn.close()
        print(f"{addr} disconnected from the server.")

#Main server loop
def main():
    initialize_votes_file()
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((SERVER_HOST, SERVER_PORT))
    server.listen(5)
    print(f"Server is listening on {SERVER_HOST}:{SERVER_PORT}")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=client_server_connection, args=(conn,
addr))
        thread.start()

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