

SERVER

```
import socket
import threading
```

```
# ENCRYPTION / DECRYPTION
```

```
def encrypt(text, shift=3):
    result = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result
```

```
def decrypt(text, shift=3):
    return encrypt(text, -shift)
```

```
# USERS DATABASE
```

```
users = {}
balances = {}
```

```
def load_users():
    global users, balances
    users = {}
    balances = {}
    try:
        with open("users.txt", "r") as f:
            for line in f:
                line = line.strip()
                if not line or ":" not in line:
                    continue
                user, enc_pwd, balance = line.split(":", 2)
                users[user] = decrypt(enc_pwd)    # decrypt password when loading
                balances[user] = float(balance)
    except FileNotFoundError:
        users = {}
        balances = {}
```

```
def save_users():
```

```

with open("users.txt", "w") as f:
    for user in users:
        # encrypt the password before saving
        enc_pwd = encrypt(users[user])
        f.write(f"{user}:{enc_pwd}:{balances[user]}\n")

# HANDLE CLIENTS
def handle_client(client, addr):
    print(f"Connected: {addr}")
    client.send(encrypt("Welcome to the banking server!").encode())
    logged_in_user = None

    while True:
        try:
            data = client.recv(1024).decode()
            if not data:
                break

            command = decrypt(data).strip()
            parts = command.split()
            if len(parts) == 0:
                continue

            action = parts[0].lower()

            # REGISTER
            if action == "register":
                if len(parts) < 3:
                    client.send(encrypt("Usage: register <username> <password>").encode())
                    continue
                username = parts[1]
                password = parts[2]
                if username in users:
                    client.send(encrypt("Username already exists!").encode())
                else:
                    users[username] = password
                    balances[username] = 0.0
                    save_users()
                    client.send(encrypt("Registration successful! You are now logged
in.").encode())

```

```

        logged_in_user = username

# LOGIN
elif action == "login":
    if len(parts) < 3:
        client.send(encrypt("Usage: login <username> <password>").encode())
        continue
    username = parts[1]
    password = parts[2]
    if username in users and users[username] == password:
        logged_in_user = username
        client.send(encrypt("Login successful!").encode())
    else:
        client.send(encrypt("Invalid username or password.").encode())

# BALANCE
elif action == "balance":
    if logged_in_user:
        client.send(encrypt(f"Balance: ${balances[logged_in_user]:.2f}").encode())
    else:
        client.send(encrypt("You must login first.").encode())

# DEPOSIT
elif action == "deposit":
    if logged_in_user:
        if len(parts) < 2:
            client.send(encrypt("Usage: deposit <amount>").encode())
            continue
        try:
            amount = float(parts[1])
            balances[logged_in_user] += amount
            save_users()
            client.send(encrypt(f"Deposited ${amount:.2f}. New balance:
${balances[logged_in_user]:.2f}").encode())
        except:
            client.send(encrypt("Invalid amount.").encode())
    else:
        client.send(encrypt("You must login first.").encode())

# WITHDRAW

```

```

elif action == "withdraw":
    if logged_in_user:
        if len(parts) < 2:
            client.send(encrypt("Usage: withdraw <amount>").encode())
            continue
        try:
            amount = float(parts[1])
            if amount > balances[logged_in_user]:
                client.send(encrypt("Insufficient funds.").encode())
            else:
                balances[logged_in_user] -= amount
                save_users()
                client.send(encrypt(f"Withdrew ${amount:.2f}. New balance:
${balances[logged_in_user]:.2f}").encode())
            except:
                client.send(encrypt("Invalid amount.").encode())
        else:
            client.send(encrypt("You must login first.").encode())

# QUIT
elif action == "quit":
    client.send(encrypt("Goodbye!").encode())
    break

# INVALID COMMAND
else:
    client.send(encrypt("Invalid command.").encode())

except:
    break

client.close()
print(f"Disconnected: {addr}")

# START SERVER
load_users()
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("127.0.0.1", 5000))
server.listen(5)
print("Starting Banking Server...")

```

```
while True:
    client, addr = server.accept()
    threading.Thread(target=handle_client, args=(client, addr)).start()
```

Client

```
import socket
```

```
# ENCRYPTION / DECRYPTION
```

```
def encrypt(text, shift=3):
    result = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result
```

```
def decrypt(text, shift=3):
    return encrypt(text, -shift)
```

```
# CONNECT TO SERVER
```

```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("127.0.0.1", 5000))
```

```
print(decrypt(client.recv(1024).decode()))
```

```
logged_in = False
```

```
# MENUS
```

```
def main_menu():
    print("\n=== MAIN MENU ===")
    print("1. Register")
    print("2. Login")
    print("3. Quit")
    return input("Enter a number: ")
```

```
def banking_menu():
    print("\n=== BANKING MENU ===")
    print("1. Check Balance")
    print("2. Deposit")
    print("3. Withdraw")
    print("4. Logout")
    return input("Enter a number: ")
```

```

# MAIN LOOP
while True:
    if not logged_in:
        choice = main_menu()

        if choice == "1": # Register
            username = input("Create a username: ")
            password = input("Create a password: ")
            command = f"register {username} {password}"
            client.send(encrypt(command).encode())
            response = decrypt(client.recv(1024).decode())
            print(response)
            if "successful" in response.lower():
                logged_in = True

        elif choice == "2": # Login
            username = input("Enter username: ")
            password = input("Enter password: ")
            command = f"login {username} {password}"
            client.send(encrypt(command).encode())
            response = decrypt(client.recv(1024).decode())
            print(response)
            if "successful" in response.lower():
                logged_in = True

        elif choice == "3": # Quit
            client.send(encrypt("quit").encode())
            print("Disconnected.")
            break

    else:
        choice = banking_menu()

        if choice == "1": # Check balance
            client.send(encrypt("balance").encode())
            print(decrypt(client.recv(1024).decode()))

        elif choice == "2": # Deposit
            amount = input("Deposit amount: ")

```

```
command = f"deposit {amount}"
client.send(encrypt(command).encode())
print(decrypt(client.recv(1024).decode()))

elif choice == "3": # Withdraw
    amount = input("Withdraw amount: ")
    command = f"withdraw {amount}"
    client.send(encrypt(command).encode())
    print(decrypt(client.recv(1024).decode()))

elif choice == "4": # Logout
    logged_in = False
    print("Logged out.")

else:
    print("Invalid option.")
```