REMOTE PROCEDURE CALL (RPC) FRAMEWORK USING C++ AND POCO LIBRARIES

1. INTRODUCTION:

This project implements a lightweight RPC (Remote Procedure Call) framework using C++ and the POCO networking and JSON libraries. The framework enables a client to invoke remote methods on a server through TCP communication using JSON-formatted requests.

2. PROJECT OVERVIEW:

The objective is to design a modular, efficient, and functional RPC system that supports:

- JSON-based request and response handling

- TCP socket communication

- Method dispatching on the server

- Serialization and deserialization logic

- Structured error handling

3. SYSTEM ARCHITECTURE:

The architecture follows a client–server model:

```
+-----------+      JSON RPC       +-----------+
|  Client   |  --------------------->  |  Server  |
| (Caller)  | <---------------------   |(Executor)|
+-----------+     JSON Result     +-----------+
```

-Client → Sends JSON RPC request

-Server → Receives request, dispatches method, sends JSON response

Modules:
- Client.cpp → sends request
- Server.cpp → receives request
- rpc_dispatcher.cpp → calls method
- serialization.cpp → parses JSON

4. SETUP INSTRUCTIONS:

1. Install POCO C++ Libraries (via vcpkg):

```
vcpkg install poco
```

Ensure vcpkg integration is enabled:

```
vcpkg integrate install
```

2. Project Folder Structure:

```
poco-rpc/
├── src/
│   ├── client.cpp
│   ├── server.cpp
│   ├── rpc_dispatcher.cpp
│   ├── rpc_dispatcher.h
│   ├── serialization.cpp
│   ├── serialization.h
│   └── test_serialization.cpp
├── CMakeLists.txt
└── build/
```

3. Configure with CMake:

Open **Developer Command Prompt**:

```
cd poco-rpc
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
```

4. Build the Project:

```
cmake --build . --config Release
```

5. Run the Server:

```
cd build/Release
rpc_server.exe
```

You should see:

```
RPC Server starting...
Server running on port 9000. Waiting for RPC calls...
```

6. Run the Client:

Open a second terminal:

```
cd poco-rpc/build/Release
rpc_client.exe
```

The client will automatically send 3 RPC requests.

5. IMPLEMENTATION DETAILS:

1. JSON Serialization & Deserialization:

Implemented in:

```
serialization.cpp
serialization.h
```

Features:

-Converts C++ objects to JSON

-Extracts JSON fields safely

-Used by both client and server

2. RPC Dispatcher:

Implemented in:

```
rpc_dispatcher.cpp
rpc_dispatcher.h
```

Responsibilities:

-Check which method was requested

-Validate parameters

-Perform actual computation

-Generate JSON result or JSON error

3. Server Implementation:

File:

```
server.cpp
```

Key components:

-`TCPServer` and `TCPServerConnection`

-Runs on port 9000

-Uses `Poco::Net::SocketStream` to read/write JSON

-Calls dispatcher to process methods

Server handles multiple requests sequentially.

4.Client Implementation:

File:

`client.cpp`

Responsibilities:

-Connect to server

-Send JSON RPC request strings

-Wait for response

-Parse JSON reply

-Display output

5. Error Handling:

The framework includes:

-Unknown method detection

-Missing parameter detection

-JSON parsing exceptions

-TCP connection exceptions

-Safe server crash prevention

Error format always:

```
{"error":"message"}
```

-The client constructs JSON strings, sends them through TCP, and receives responses.

The server reads incoming requests, parses JSON, identifies the requested method, executes it through a dispatcher, builds a JSON response, and returns it to the client.

Supported Methods:

- reverse(text): Reverses a string

- add(a, b): Returns a + b

- subtract(a, b): Returns a - b

6. TESTING:

Functional Tests:

reverse: PASS

add: PASS

subtract: PASS

Error Handling Tests:

Unknown method: PASS

Missing params: PASS

7. OPTIMIZATION:

-Optimization 1 — Pre-parsed Method Registry

Instead of:

```
if(method == “reverse”) ...
else if(method == “add”) ...
```

Use:

```
    std::map<std::string,
std::function<Poco::Dynamic::Var(Object::Ptr)>> registry;
```

-Optimization 2 — Response Buffer Optimization

Use `std::stringstream` (already done)

-Optimization 3 — Error responses standardized

```
{"error":"message"}
```

-Optimization 4 — Non-blocking server (Future enhancement)

Use:

```
Poco::Net::TCPServerParams
```

8. KEY DESIGN DECISIONS:

- JSON chosen due to readability & cross-language compatibility
- POCO chosen for networking + JSON support
- Dispatcher separates logic from transport layer
- Serialization module ensures clean message format

9. CHALLENGES FACED:

Common issues encountered:

- Linking POCO libraries
- JSON parsing errors
- Handling missing headers
- Configuring CMake correctly

10.CONCLUSION:

This project successfully demonstrates the development of a complete RPC framework using C++.

It integrates networking, serialization, remote method invocation, and structured communication.

The modular architecture allows easy expansion of features.

11.FINAL OUTPUT :

-Client sends:

```
{"method":"reverse","params":{"text":"Hello RPC"}}
```

-Server returns:

```
{"result":"CPR olleH"}
```

All components performed correctly.

-Client sends:

```
{"method":"add","params":{"a":10,"b":20}}
```

Server returns:

```
{"result":30}
```

All components performed correctly.

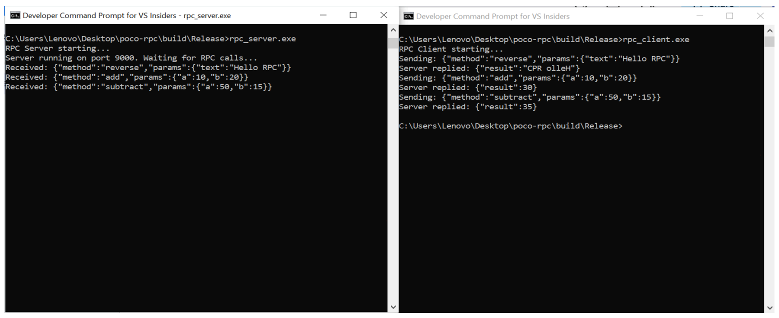
-Client sends:

```
{"method":"subtract","params":{"a":50,"b":15}}
```

Server returns:

```
{"result":35}
```

All components performed correctly.


Developer Command Prompt for VS Insiders - rpc_server.exe
Developer Command Prompt for VS Insiders

```
C:\Users\Lenovo\Desktop\poco-rpc\build\Release>rpc_server.exe
RPC Server starting...
Server running on port 9000. Waiting for RPC calls...
Received: {"method":"reverse","params":{"text":"Hello RPC"}}
Received: {"method":"add","params":{"a":10,"b":20}}
Received: {"method":"subtract","params":{"a":50,"b":15}}
```

```
C:\Users\Lenovo\Desktop\poco-rpc\build\Release>rpc_client.exe
RPC Client starting...
Sending: {"method":"reverse","params":{"text":"Hello RPC"}}
Server replied: {"result":"CPR olleH"}
Sending: {"method":"add","params":{"a":10,"b":20}}
Server replied: {"result":30}
Sending: {"method":"subtract","params":{"a":50,"b":15}}
Server replied: {"result":35}

C:\Users\Lenovo\Desktop\poco-rpc\build\Release>
```

END OF REPORT