LAB 3

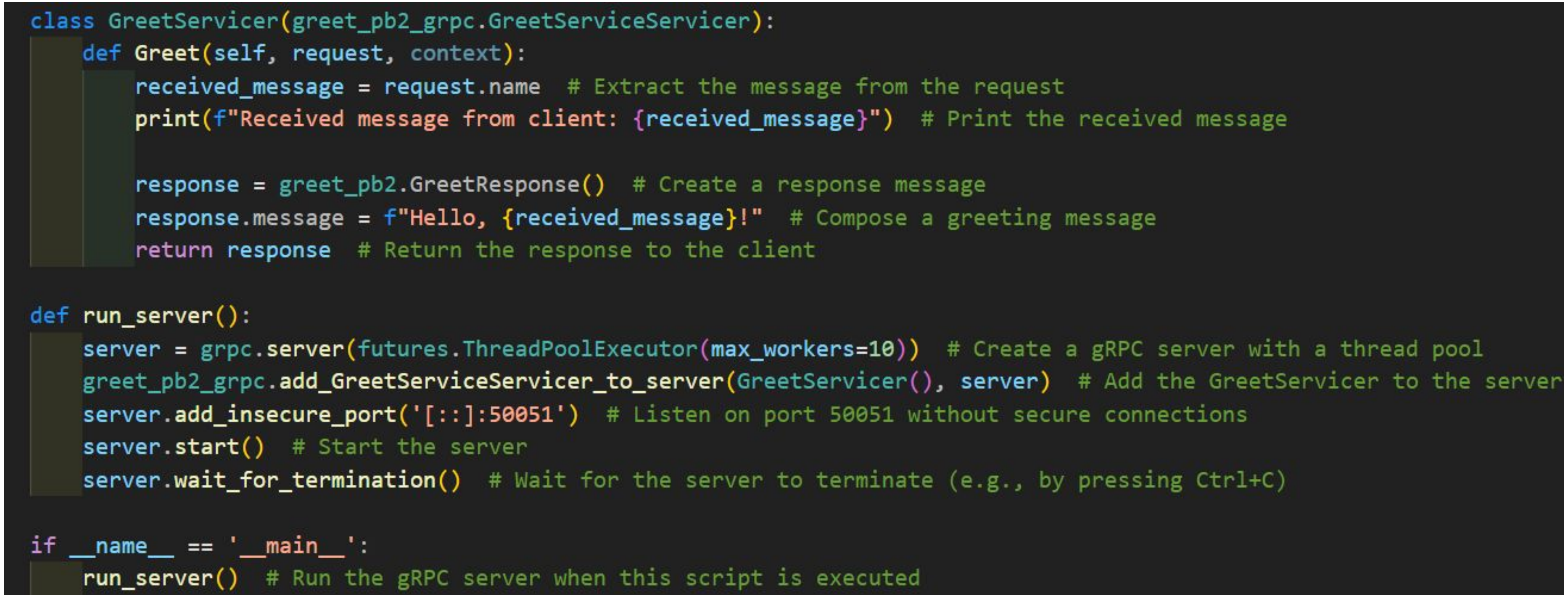
DISTRIBUTED SOFTWARE SYSTEMS

SCENARIO: CREATING A GRPC SERVER

Client 1 should interact with the gRPC server, connect to the server and call the Greet method with a custom name and print the response received from the server, which should be a greeting message..

Client 2 should interact with the same gRPC server, connect to the server and call the Greet method with a different custom name and print the response received from the server, which should be a different greeting message.

Creating a GRPC server

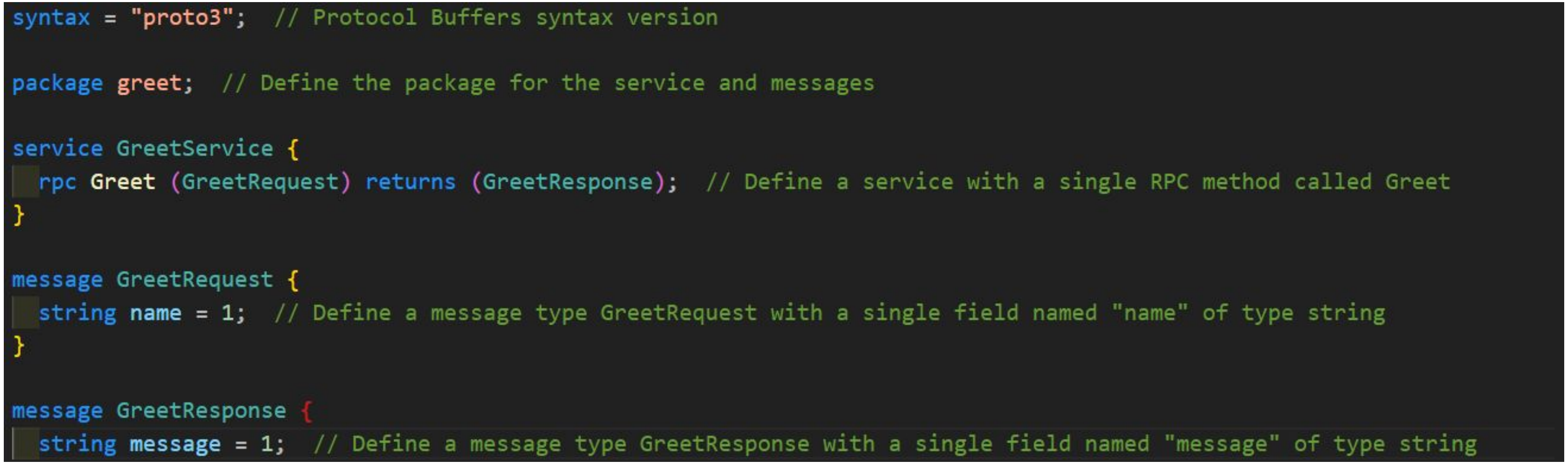


```python
class GreetServicer(greet_pb2_grpc.GreetServiceServicer):
    def Greet(self, request, context):
        received_message = request.name  # Extract the message from the request
        print(f"Received message from client: {received_message}")  # Print the received message

        response = greet_pb2.GreetResponse()  # Create a response message
        response.message = f"Hello, {received_message}!"  # Compose a greeting message
        return response  # Return the response to the client

def run_server():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))  # Create a gRPC server with a thread pool
    greet_pb2_grpc.add_GreetServiceServicer_to_server(GreetServicer(), server)  # Add the GreetServicer to the server
    server.add_insecure_port('[::]:50051')  # Listen on port 50051 without secure connections
    server.start()  # Start the server
    server.wait_for_termination()  # Wait for the server to terminate (e.g., by pressing Ctrl+C)

if __name__ == '__main__':
    run_server()  # Run the gRPC server when this script is executed
```

The server provides a method called Greet that accepts a single string parameter and responds with a greeting message.

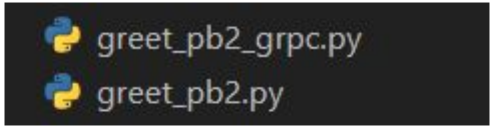
Creating a GRPC server (Protofile)

I defined the Greet method and the service in the .proto file



```
syntax = "proto3";  // Protocol Buffers syntax version

package greet;  // Define the package for the service and messages

service GreetService {
  rpc Greet (GreetRequest) returns (GreetResponse);  // Define a service with a single RPC method called Greet
}

message GreetRequest {
  string name = 1;  // Define a message type GreetRequest with a single field named "name" of type string
}

message GreetResponse {
  string message = 1;  // Define a message type GreetResponse with a single field named "message" of type string
```

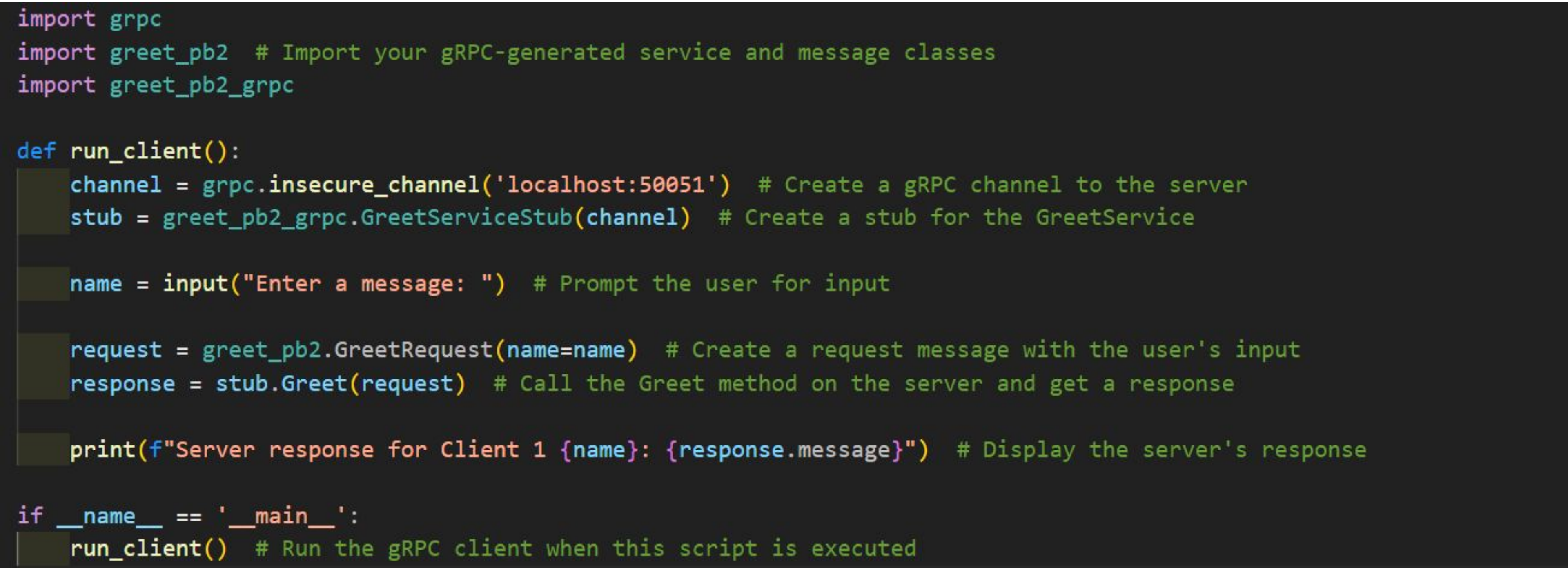
One time I execute the proto.file it creates this two new automatic files:



greet_pb2_grpc.py
greet_pb2.py

Clients

The client interacts with the gRPC server to connect with and call the Greet method with his custom name.



```python
import grpc
import greet_pb2  # Import your gRPC-generated service and message classes
import greet_pb2_grpc

def run_client():
    channel = grpc.insecure_channel('localhost:50051')  # Create a gRPC channel to the server
    stub = greet_pb2_grpc.GreetServiceStub(channel)  # Create a stub for the GreetService

    name = input("Enter a message: ")  # Prompt the user for input

    request = greet_pb2.GreetRequest(name=name)  # Create a request message with the user's input
    response = stub.Greet(request)  # Call the Greet method on the server and get a response

    print(f"Server response for Client 1 {name}: {response.message}")  # Display the server's response

if __name__ == '__main__':
    run_client()  # Run the gRPC client when this script is executed
```

Execution

The client print the response received from the server, which should be a greeting message, and the server also prints when it receives the messages from clients.