1. Describe a redesign of your short message service as a microservice using RPC or REST as an interface – explain why you choose the interface method you chose (REST/RPC) and describe the microservice principles you will apply in your design. Using pseudo-code describe the implementation of the four methods of your service: **15 marks**

In this Project I have used ONC RPC as an interface. While REST is better at getting information RPC is better for calling functions. This means that functions like login and logout should work better with RPC, whereas upload and download functions would be more suitable for REST.

Very little refactoring is needed in turn reducing the amount of time and effort required to redesign the project. RPC is easy to implement for this type of application.

RPC is lightweight allowing for a more efficient implementation. It also allows preservation of the business logic of the application.

ECHOPROG contains a method login\_1() which takes two strings: uName, a username and pWord, a password.

Psuedo Code:

1. Implement **one** of the four methods (note you don’t have to host it on Docker): **15 marks** . Look at the labs on RPC (ONC RPC/Thrift/gRPC) or REST(IntelliJ or DropWizard) to help you.

For this I chose to implement the login method

1. Compare and contrast your original protocol-based solution with the new microservices solution **20 marks (**make four points supported with examples**)**

The new microservice based approach is much simpler to implement when compared to the previous protocol-based one. While the protocol-based solution required datagrams and sockets whereas the RPC based microservice did not require these. This made the microservice significantly simpler to implement as it used an IDL file. This IDL file generated two classes with given methods and parameters allowing for a faster and easier setup.

The protocol-based solution would be more difficult to upscale over the RPC microservice. The protocol-based solution would require a significant amount of redesigning. This would cost a lot of time and effort that could be used elsewhere. The RPC microservice does not have this issue. The IDL file would be the only major change in this case where it would be redesigned to generate any new classes.

The protocol-based solution required far more code. This approach had far more classes, each having a significant amount of code within them. This made debugging more difficult as you had to sort through more code. They could also become more difficult to read though with amount of classes and methods. The RPC microservice had fewer classes, with each class being shorter with fewer methods than comparable classes from the protocol based solution. Several of these classes were autogenerated. This approach saved time by lessening the amount of code which had to be written and kept track of.

The protocol-based solution is much lower level requiring more setup. I first designed the protocol that would be implemented and then I designed and implemented the four methods. The RPC microservice is layer higher than the protocol approach. It requires less direct interaction with the computer as it does not use datagrams and sockets unlike the protocol approach.