Client server network project

By

Group C

Submitted to

The University of Liverpool

MASTER-OF-COMPUTER-SCIENCE

*Software Development in Practice*

Word Count: XXX

09/Oct/2023

REPORT TITLE

Submitted to

The University of Liverpool

Word Count: XXX

09/Oct/2023

TABLE OF CONTENTS

Page

[Chapter 1. Introduction 2](#_Toc147156570)

[Chapter 2. Solution Overview 3](#_Toc147156571)

[2.1 Solution overview 3](#_Toc147156572)

[2.2 Technical Flow 4](#_Toc147156573)

[Chapter 3. Solution Design (Client) 5](#_Toc147156574)

[3.1 Solution Design (Server) 5](#_Toc147156575)

[3.2 Solution Design (Server) 5](#_Toc147156576)

[3.3 Error Handling 6](#_Toc147156577)

[3.4 Design Decisions 6](#_Toc147156578)

[Chapter 4. Testing 6](#_Toc147156579)

[4.1 Serialization Unit Testing 6](#_Toc147156580)

[4.2 Encryption Unit Testing 6](#_Toc147156581)

[4.3 Network Unit Testing 6](#_Toc147156582)

[Chapter 5. Conclusion 7](#_Toc147156583)

[REFERENCES 8](#_Toc147156584)

[APPENDICES 8](#_Toc147156585)

# Introduction

This Project is to establish a simple Client Server Network. This summative assignment is a practical exploration of a real-life project which involves gathering requirements, designing, and collaborating with the Project Manager, Architect, Developer and Tester to implement a simple client/server network. This assignment equips the team with essential skills for designing and managing networked systems in real-world scenarios.

Through this Project, Team C will demonstrate an understanding of network communication, the creation and manipulation of data structures, the transmission of data between client and server, and the implementation of advanced features such as data serialisation in various formats and encryption of text files.

# Solution Overview

Solution overview

Figure 1 is a high-level solution of a simple client-server network to perform data serialisation, encryption, and transfer from a server to a client.

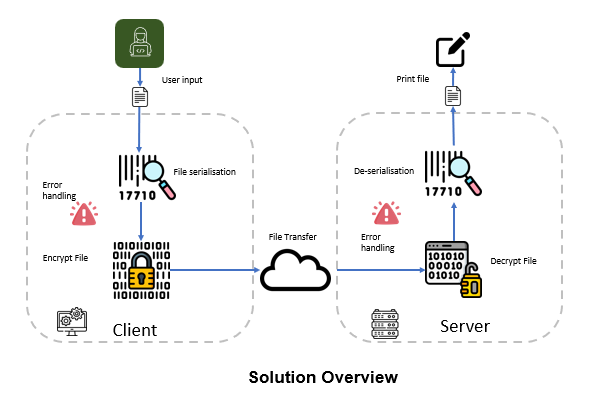


Figure 1. Solution overview

Technical Flow

The Technical Flow diagram (Figure 2) showcases the process flow of a Client-Server network that accomodates various data formats and encrypts the data to protect transmitted information. It also showcases server actions to accept the transmited data, de-seralise, unencrpt and print the received data.

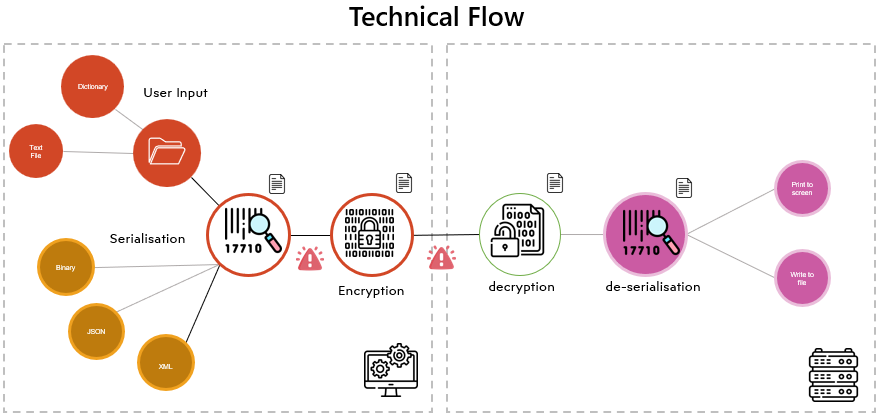
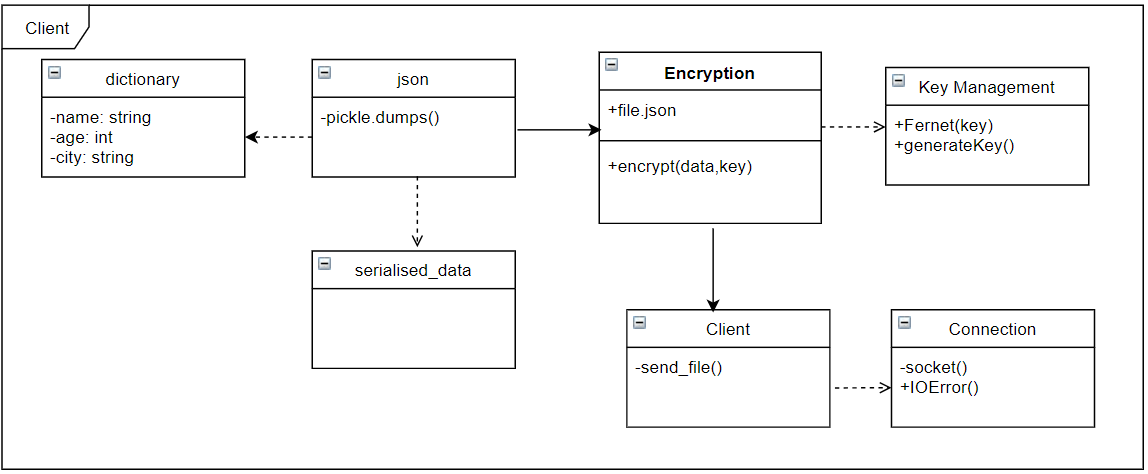


Figure 2. Technical Flow

# Solution Design

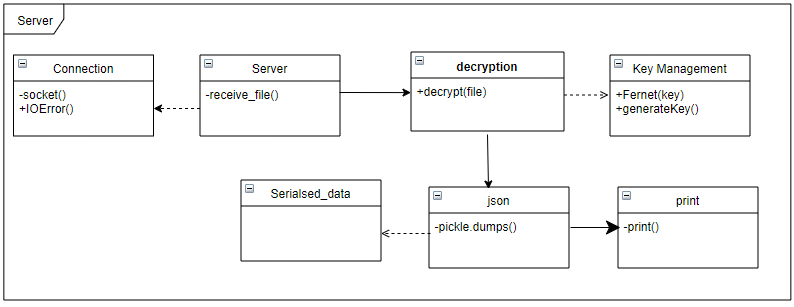
Solution Design (Client)



Key components within the Client design

* + Serialisation: Serialise the data into a structured format to maintain data integrity and interoperability.
  + Key Management: The key management component is responsible for generating and managing encryption keys.
  + Encryption: The encryption component is responsible for encrypting and decrypting data using the encryption keys provided by the key management component.
* Client: The client is responsible for sending encrypted data

Solution Design (Server)



Key components within the Server design

* **Server:** The server opens a socket to listen for incoming client connections. Once a connection is established, the server receives the file from the client. The server receives the file from the client. using a TCP Protocol.
* **Decryption:** The server decrypts the file using the same key used to encrypt it. This will be done via the Fernet Key generated by the client. This key will be supplied along with the file.
* **De-serialisation:** The server de-serialises the data in the file to a json format.
* **Print:** The server prints the contents of the dictionary.

Error Handling

Design Decisions

**Serialisation**: Pickle module is used to serialise the dictionary into JSON format for the following reasons –

* JSON is a text-based format that is easy to read and write.
* It is a language-agnostic format and interoperability between systems written in different languages.
* It is a lightweight format with minimal overhead.

**Encryption**: Fernet Cryptographic protocol will be used as our encryption tool of choice for the following reasons -

* Fernet uses symmetric key cryptography, which is simple and easy to use.
* Fernet is built on AES (Advance Encryption Standard) and ensures data encrypted is well-protected.

# Unit Testing

Serialization Unit Testing

Encryption Unit Testing

# Conclusion

REFERENCES

APPENDICES

|  | **Group Artifacts** |
| --- | --- |
| 1 | GitHub Repository |
| 2 | Client Server Network Project report |
| 3 | Requirement.txt |
| 4 | Readme.md |
| 5 | Screenshot of Github repo |
| 6 | Screenshot of Logs |