Contents

[Design 1](#_Toc499154018)

[Overview 1](#_Toc499154019)

[Objectives 1](#_Toc499154020)

[Design Philosophy 1](#_Toc499154021)

[Server UML 1](#_Toc499154022)

[Client UML 1](#_Toc499154023)

[Implementation 1](#_Toc499154024)

[Overview 1](#_Toc499154025)

[Objectives 1](#_Toc499154026)

[Application Layer 1](#_Toc499154027)

[Presentation Layer 1](#_Toc499154028)

[Service Layer 1](#_Toc499154029)

[User Manual 1](#_Toc499154030)

[Conclusion 1](#_Toc499154031)

# Design

## Overview

For this project I have developed a UDP based file transfer system. The application uses datagrams to pass Strings and Byte arrays from client to server and vice versa. Alongside being able to log in and log out, users can also upload and download files to the server. This process creates a folder on the server side which store the files the user uploads. To access these features, message code must be passed between the server and client.

This system uses the DatagramPacket and DatagramSocket java API to implement its functionalities.

## 

## Objectives

The objectives for this project are as follows:

* To allow a client to log in, log out, upload and download files using datagrams
* To design and follow the rules set in the protocol for the interactions with the file transfer server
* To show understanding of the implementation of a protocol

## Design Philosophy

For the design philosophy behind the Client, the logic is split amongst the three layers. These three layers are the Presentation Layer (ClientGUI.java), Application Layer (ClientHelper.java) and the service layer (ClientDatagramSocket.java). By moving the application logic to the ClientHelper.java, we can satisfy the open/closed principle allowing for extension as the program develops in the later stages. The user interactions with the GUI can be separated from overall logic by calling the methods in the ClientHelper.java. This means that the GUI will only have to deal with handling input data and responses from the server. The ClientDatagramSocket class will implement overloads for its sendMessage() method to send byte array data when uploading files. It will also implement a method to read byte arrays sent from the server.

For the design philosophy of the server, the application and presentation layer are split between the Server.java class. The service layer of the application will be the ServerDatagramSocket class. Similarly to the client, methods will be implemented to deal with sending and receiving byte arrays to and from the client.

## Server UML

## Client UML

# Implementation

## Overview

## Objectives

## Application Layer

## Presentation Layer

## Service Layer

# User Manual

# Conclusion