# Introduction

## Purpose and Scope

The purpose of this software is to provide an FTP client with the ability to receive/send archived directories. The software will allow users to connect to an FTP server and perform various operations, including downloading and uploading files and directories in the form of archives. The scope of the software includes the implementation of necessary functionality and features to provide an easy-to-use interface for users to interact with the FTP server. The software will also be able to handle errors and exceptions that may occur during the execution of its functions.

## Background and Context

FTP (File Transfer Protocol) is a standard protocol used for transferring files over the internet. It is commonly used by web developers, software engineers, and IT administrators for uploading and downloading files to and from servers. However, traditional FTP clients lack the ability to receive/send archived directories which can be very useful in situations where multiple files need to be transferred at once. This is where the need for an FTP client with the ability to receive/send archived directories comes in.

Such a program would allow users to select multiple files or directories and compress them into a single archive file, such as a ZIP or TAR file. This archive file can then be transferred over FTP to a remote server or downloaded from the server to the user's local machine. This feature can save a lot of time and effort for users who need to transfer multiple files or directories at once.

Furthermore, the program can also include additional features such as support for resumable downloads/uploads, folder synchronization, and multi-threading to improve performance. These features are essential for large file transfers and can greatly improve the overall user experience.

Overall, the purpose of this software is to provide users with a more efficient and convenient way to transfer files over FTP, especially when dealing with multiple files or directories.

## Project Objectives

1. Develop a user-friendly interface that allows users to easily navigate and transfer files between their local machine and an FTP server.
2. Implement the ability to send and receive compressed directories, such as zip or tar files, to reduce transfer times and improve efficiency.
3. Ensure the security of file transfers by implementing encryption and authentication protocols.
4. Provide users with the ability to resume interrupted transfers and track transfer progress.
5. Implement features such as remote file editing and file/folder synchronization to further streamline the user experience.
6. Test the software thoroughly to ensure it works reliably and efficiently under various network conditions and with different FTP servers.
7. Optimize the software to improve transfer speeds and reduce resource usage.
8. Provide clear documentation to assist users with installation, setup, and troubleshooting.
9. Continuously improve the software based on user feedback and new technologies in the FTP client space.
10. Ensure the software complies with all relevant legal and regulatory requirements.

## Key Stakeholders

1. Users: The end-users who will use the FTP client to transfer files to and from an FTP server.
2. Developers: The team responsible for designing and developing the FTP client software.
3. Project managers: The team responsible for managing the project, ensuring timely delivery, and overseeing the work of the developers.
4. Quality assurance (QA) team: The team responsible for testing the software to ensure that it meets the required quality standards.
5. Technical support team: The team responsible for providing technical support to end-users who may encounter issues while using the FTP client.
6. Marketing and sales team: The team responsible for promoting and selling the FTP client to potential customers.
7. Management: The management of the organization that is developing and/or distributing the FTP client.

# Functional Requirements

## File Transfer

The specific functionalities related to file transfer that the software should provide:

1. Upload and download files: The software should allow the user to upload and download files to and from the remote FTP server.
2. File transfer modes: The software should support different transfer modes such as ASCII and binary modes.
3. Transfer progress tracking: The software should provide a progress indicator to track the file transfer status and completion.
4. Resume support: The software should support the resumption of interrupted file transfers.
5. Folder transfers: The software should support transferring entire folders/directories recursively.
6. File compression and extraction: The software should allow the user to compress and extract files and directories to be transferred to and from the remote FTP server.
7. File encryption: The software should support encryption of files during transfer to ensure data security.
8. Error handling: The software should provide appropriate error messages and handling mechanisms for common errors such as invalid credentials or network errors

## Directory Transfer

The next functional requirements should meet the program to provide directory transfer:

1. Ability to transfer directories containing multiple files and sub-directories
2. Ability to choose the source and destination directories for the transfer
3. Ability to create a new directory at the destination and transfer the source directory into it
4. Ability to overwrite or skip existing files or directories at the destination
5. Ability to resume the transfer from the point where it was stopped if the transfer is interrupted
6. Ability to provide progress updates during the transfer
7. Ability to handle transfer errors and provide appropriate error messages to the user
8. Ability to cancel a transfer in progress

## Archiving and Compression

Functional requirements the program should meet to provide work with the archiving and compression:

1. Ability to create archives: The system should provide the ability to create archives of files and directories.
2. Support for various archive formats: The system should support various archive formats such as .zip, .tar, .tar.gz, .tar.bz2, etc.
3. Compression options: The system should allow users to choose from different compression options such as no compression, low, medium, or high compression.
4. Decompression: The system should also allow users to decompress existing archives.
5. Error handling: The system should handle errors that may occur during archiving and compression, such as insufficient disk space, read or write errors, and file format errors.
6. Progress tracking: The system should provide progress indicators during the archiving and compression process.
7. Encryption: The system should also provide encryption options to secure the archives created by the user.
8. Verification: The system should verify the integrity of the archives created and ensure that the compressed and decompressed files are identical.

## Authentication and Security

The program should meet the next requirements to provide security:

1. The system shall provide secure authentication for users accessing the FTP client.
2. The system shall enforce password policies, such as minimum length and complexity requirements.
3. The system shall allow users to change their passwords.
4. The system shall maintain a list of authorized users and their permissions.
5. The system shall log all user activity for auditing and security purposes.
6. The system shall use encryption to protect the confidentiality of files during transmission.
7. The system shall prevent unauthorized access to files by implementing access controls.
8. The system shall protect against common attacks, such as denial-of-service attacks and buffer overflow attacks.

## Error Handling and Recovery

The program should meet the next requirements to provide error handling and recovery:

1. The software should provide appropriate error messages for the user when an error occurs, indicating what went wrong and suggesting possible solutions.
2. The software should have a robust error handling mechanism that can gracefully recover from errors and continue with the operation if possible.
3. The software should allow the user to retry failed operations, such as file transfers, in case of errors.
4. The software should provide a mechanism to log errors and other important events, such as successful file transfers, for debugging and auditing purposes.
5. The software should provide a mechanism to roll back incomplete or failed operations, such as directory transfers, to a known good state.
6. The software should have appropriate security measures in place to prevent unauthorized access and protect user data, including error messages and log files.
7. The software should be able to recover from unexpected events, such as power outages or network failures, without losing user data or corrupting files.

## Logging and Reporting

The program should meet the next requirements to provide logging and reporting:

1. The program should log all significant events and errors during the file transfer process.
2. The program should provide a report on the status and success of the file transfer process, including the number of files transferred and any errors encountered.
3. The program should allow users to configure the logging level and output format (e.g., text file, database, email notification).
4. The program should include timestamps in all log entries and reports to aid in troubleshooting and analysis.
5. The program should be able to generate graphical representations of the transfer process status, such as progress bars or charts, to make it easier for users to monitor the progress of the transfer.
6. The program should allow users to export and save logs and reports for future reference or analysis.
7. The program should notify the user when an error occurs, including a brief description of the error and potential solutions.
8. The program should provide recovery options in case of errors, such as retrying the transfer or resuming the transfer from the point of failure.
9. The program should have a mechanism for storing and archiving logs and reports for future reference or analysis.
10. The program should provide the ability to search and filter logs and reports based on specific criteria, such as date, time, or error type.

# Non-Functional Requirements

## Performance

## Reliability and Availability

## Usability and Accessibility

## Compatibility and Interoperability

## Scalability and Extensibility

## Maintainability and Supportability

## Legal and Regulatory Compliance

# User Interface Requirements

## User Interface Design

## Navigation and Layout

## Input and Output Controls

## Help and Documentation

## Error Messages and Alerts

# Technical Requirements

## Hardware Requirements

## Software Requirements

## Network Requirements

## Integration Requirements

## System Architecture

## Data Management and Storage

# Testing Requirements

## Test Plan

## Test Cases

## Test Environment

## Test Data

## Test Results

## Bug Tracking and Reporting

# Deployment Requirements

## Installation and Configuration

## System Requirements

## Compatibility and Interoperability

## Deployment Plan

## Rollout and Migration

# Support and Maintenance Requirements

## Bug Fixing and Updates

## User Training and Documentation

## Support Services

## Maintenance and Upgrades

## End-of-Life and Disposal

# References

## Standards and Regulations

## Technical Documentation

## User Guides and Manuals

## Training Materials

## Other Relevant Resources