Account Manager

Technical Specification

Damian Zylski and Barry Hoinacki JR

2019

Table of Contents

[Revision History 2](#_Toc5548106)

[Introduction 3](#_Toc5548107)

[Project Scope 3](#_Toc5548108)

[Glossary 3](#_Toc5548109)

[Requirements Description 3](#_Toc5548110)

[Summary 3](#_Toc5548111)

[Assumptions and Dependencies 3](#_Toc5548112)

[Use Cases 4](#_Toc5548113)

[Actors 5](#_Toc5548114)

[Use Case Details 5](#_Toc5548115)

[Use Case 12 5](#_Toc5548116)

[Use Case 16 6](#_Toc5548117)

[Use Case 19 7](#_Toc5548118)

[Use Case 17 8](#_Toc5548119)

[Use Case 18 9](#_Toc5548120)

[External Interface Requirements 9](#_Toc5548121)

[Technical Solutions Description 10](#_Toc5548122)

[Interfaces 10](#_Toc5548123)

[Architecture 10](#_Toc5548124)

[Component 1 10](#_Toc5548125)

[Component 2 10](#_Toc5548126)

[Component n 10](#_Toc5548127)

[Data Model 10](#_Toc5548128)

[Table 1 10](#_Toc5548129)

[Table 2 10](#_Toc5548130)

[Table n 10](#_Toc5548131)

[Class Model 10](#_Toc5548132)

[Appendix A: Issues List 10](#_Toc5548133)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Summary |
| 1.0 | Damian Zylski, Barry Hoinacki Jr | 04/23/2019 | Initial document creation, Introduction. |
| 1.1 | Damian Zylski | 04/23/2019 | Typed up the project scope and out of scope fields. |
| 1.2 | Barry Hoinacki Jr | 04/28/2019 | Started to update glossary, typed up summary, started on assumptions and dependencies. |

# Introduction

An account manager is a software application that stores a list of accounts, such as a gmail account, and their respective details, such as passwords, to a database file. This document describes requirements and technical documentation for a new account manager.

## Project Scope

The software considers the following to be in scope

* The ability to use the account manager anywhere (portable)
* The ability create a database and add an account to it
* The ability to load an existing account
* The ability to edit and delete accounts in a database
* The ability to store account information securely using Blowfish encryption

The software considers the following to be out of scope

* The ability to use autofill on a particular website or login screen
* The ability to export database file in different file formats
* The ability to set database policies and usage rules
* File recovery in the case the file is accidently edited or deleted
* Client distribution and installation.

# Glossary

For this document, the terms below are defined as shown.

|  |  |
| --- | --- |
| Term | Definition |
| Encryption | Process of taking words and other such data and scrambling it by using a specific method so it would look like a jumbled mess. |
| Decryption | Process of taking information that was encrypted and trying to reverse the effects of it to make it readable/usable. |
| Blowfish | A method of encryption/decryption |
| Account | A [word] to store information that is needed to either label or be used to login to a specific user in a service such as an email or social media. |
| Database | A storage of various information, in this case storing the master password and the various accounts and their information. |
| Master Password | The password used to be able to get into the database so you can add, edit, or see the accounts in the database. |
| Scrum | An incremental software development process with defined work periods after which a workable piece of software is delivered to the user. |
|  |  |
| Client | The machine running the Account manager software. |

# Requirements Description

This section describes the business requirements.

## Summary

An account manager is used to store accounts for various things such as email or social media sites so that you will be able to securely keep all the login information for all the accounts so that the client would not need to constantly try different passwords/username combinations to log into an account. As such the requirements center around how the account manager activities will happen.

## Assumptions and Dependencies

The project assumes the following

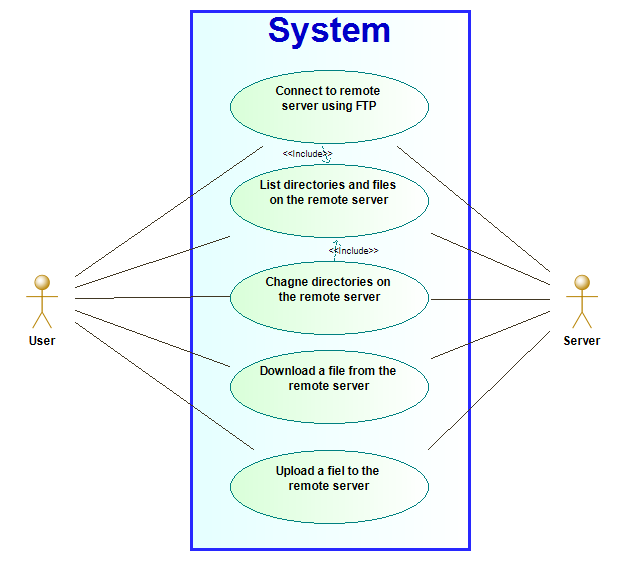
* The user will be able to remember where the database file is located and the master password for that database file.
* The user already has accounts they want to store into the account manager.
* The target operating system is Windows 7 or newer.

## Use Cases

This documentation focuses on the key uses cases the software will meet. While other use cases are present, and documented in the scrum tool, the following have been identified as either being key functionalities or the most difficult to implement. For details on the other use cases see the project on [Taiga](https://tree.taiga.io/project/bart-c137-s2019-ftpclient/).

|  |  |
| --- | --- |
| Use Case ID | Use Case Summary |
| 12 | Connect to remote server using FTP |
| 16 | List directories and files on the remote server |
| 19 | Change directories on the remote server |
| 17 | Download a file from the remote server |
| 18 | Upload a file to the remote server |

The interactions between the use cases is shown in the diagram below.



### Actors

All the use cases identified in section Use Cases reference one of the following actors.

|  |  |
| --- | --- |
| Actor | Description |
| User | The user who is using the software. |
| System | The software the user is using. This is also the software described in this document. |
| Server | The remote server the user is accessing while using the System. |

# Use Case Details

## Use Case 12

|  |  |
| --- | --- |
| **Use Case** | Connect to remote server using FTP |
| **Primary Actor** | System |
| **Goal in Context** | To use the FTP protocol to connect to a remote server. |
| **Preconditions** | The user can start the application. |
| **Trigger** | The user initiates a new connection |
| **Scenario** | |
| 1. The user provides the server name. 2. The user provides the user ID. 3. The user provides the password for the provided user ID. 4. The user acknowledges the input is complete. 5. The system verifies the format of the server name is a FQN, IP address, or a valid URL. 6. The system pings the server. 7. The system uses FTP to log into the server. 8. The system shows the root folder. | |
| **Exceptions** | |
| 1. The server name is not a valid FQN, IP address, or URL.    1. Follow Scenario steps 1 through 3.    2. The system checks the server name and determines it’s not valid.    3. The system displays a message to the user stating the server name is not valid.    4. The system displays the server and user information screen. 2. The server is not reachable.    1. Follow Scenario steps 1 through 5.    2. The system does not get a response from the server.    3. The system displays a message to the user stating the server is not reachable.    4. The system displays the server and user information screen. 3. The host, user ID, password, or port are not specified.    1. Follow Scenario steps 1 through 4.    2. The system verifies the field is missing.    3. The system notifies the user information is missing. 4. The user ID or password is incorrect.    1. Follow Scenario steps 1 through 5.    2. The system gets an error code from the server.    3. The system reports the error back to the user.    4. The system attempts to provide a user-friendly message to the user along with the error. | |
| **Priority** | High |
| **When Available** | 1st or 2nd Sprint |
| **Frequency of Use** | Every day |
| **Secondary Actors** | * User * Server |
| **Open Issues** | None |

## Use Case 16

|  |  |
| --- | --- |
| **Use Case** | List directories and files on the remote server |
| **Primary Actor** | System |
| **Goal in Context** | Retrieve the list of directories and/or files on the server. |
| **Preconditions** | * The user can start the application. * The user can log into the server. |
| **Trigger** | * The user first logs into the remote server. * The user navigates to a new directory. |
| **Scenario** | |
| 1. The system determines it must fetch the remote files from the server. 2. The system asks the server for the files. 3. The server responds with the list of files. 4. The system asks for a list of directories. 5. The server responds with a list of directories. 6. The system displays the files and folders to the user. | |
| **Exceptions** | |
| 1. The user does not have permission to access the target directory.    1. Follow Scenario steps 1 and 2.    2. The system gets an error code from the server.    3. The system displays the message to the user in a user-friendly message. 2. The server is not reachable to fetch the files    1. Follow Scenario steps 1 and 2.    2. The system cannot complete the operation and does not receive a response from the server.    3. The system displays a message to the user in a user-friendly message. 3. The directory is empty (no files or folders)    1. Follow Scenario steps 1 through 6.    2. The system displays no files or folders. | |
| **Priority** | High |
| **When Available** | 1st or 2nd Sprint |
| **Frequency of Use** | Every day |
| **Secondary Actors** | * User * Server |
| **Open Issues** | None |

## Use Case 19

|  |  |
| --- | --- |
| **Use Case** | Change directories on the server |
| **Primary Actor** | System |
| **Goal in Context** | Change the working directory on the server |
| **Preconditions** | * There is an existing connection to a remote server. * The user can see a list of available folders. |
| **Trigger** | The user selects a new folder to change to and initiates changing the folder. |
| **Scenario** | |
| 1. The system requests the server to change to the working director to the folder selected by the user. 2. The server responds with a success code. 3. Follow Use Case 16. | |
| **Exceptions** | |
| 1. The connection between the client and the server is terminated.    1. The system requests the server to change the working directory.    2. The system realizes the connection is terminated.    3. The system asks the user if the user want to reconnect.    4. The user responds Yes.    5. The systems logs into the server.    6. The system asks the remote server to change the working directory.    7. Follow the Scenario. | |
| **Priority** | High |
| **When Available** | 1st or 2nd Sprint |
| **Frequency of Use** | Every day |
| **Secondary Actors** | * User * Server |
| **Open Issues** | Is it possible to determine if the system can detect a terminated session? |

## Use Case 17

|  |  |
| --- | --- |
| **Use Case** | Download a file from the server |
| **Primary Actor** | System |
| **Goal in Context** | Download the contents of a file on the server to the local client. |
| **Preconditions** | * There is an existing connection to a remote server. * The user can see a list of available files on the server. |
| **Trigger** | The user selects a file and initiates the download. |
| **Scenario** | |
| 1. The system requests the server to start sending the file contents. 2. The server starts sending the bytes to the system. 3. The server identifies the file send is complete. 4. The system writes the data to a file in the working directory on the client with the same name as the file on the server. 5. The system notifies the user the download is complete. | |
| **Exceptions** | |
| 1. The connection is terminated before the download completes.    1. The system requests the server to start sending the file contents.    2. The server starts sending the bytes to the client.    3. The system receives an error during the download.    4. The system discards the received bytes.    5. The system displays a user-friendly message to the user. 2. The user is out of space on the client.    1. Follow Scenario steps 1 through 3.    2. The system attempts to write the bytes to a file on the disk and gets an error.    3. The system discards the received bytes.    4. The system displays a user-friendly message to the user. 3. The user does not have permission to create a new file in the client working directory.    1. Same flow as in Exception 2. | |
| **Priority** | High |
| **When Available** | No later than Sprint 4. |
| **Frequency of Use** | Every day |
| **Secondary Actors** | * User * Server |
| **Open Issues** | None |

## Use Case 18

|  |  |
| --- | --- |
| **Use Case** | Upload a file to the server |
| **Primary Actor** | System |
| **Goal in Context** | Upload the contents of a local file to the server. |
| **Preconditions** | * There is an existing connection to a remote server. * The user can see a list of available files on the client. |
| **Trigger** | The user selects a local file and initiates the upload. |
| **Scenario** | |
| 1. The system starts sending the bytes to the server. 2. The system sends a send complete message to the server. 3. The server saves the received bytes in the working directory with the same name as the local client file. | |
| **Exceptions** | |
| 1. The connection is terminated before the upload completes.    1. The system starts sending the bytes to the server.    2. The system realizes the connection is terminated.    3. The system discards the bytes it has read from the file.    4. The system displays a user-friendly message to the user. 2. The server is out of space.    1. Follow Scenario steps 1 through 2.    2. The server attempts to write the bytes to a file on the disk and gets an error.    3. The server discards the received bytes.    4. The server notifies the system of the error.    5. The system displays a user-friendly message to the user. 3. The user does not have permission to create a new file in the client working directory.    1. Same flow as in Exception 2. | |
| **Priority** | High |
| **When Available** | No later than Sprint 4. |
| **Frequency of Use** | Every day |
| **Secondary Actors** | * User * Server |
| **Open Issues** | None |

# External Interface Requirements

There are no external interface requirements.

# Technical Solutions Description

The FTP Client is developed as a JavaFX application designed to run on an individual user’s machine. This solution takes advantage of the [Apache Commons Net](https://commons.apache.org/proper/commons-net/) component to simplify and shorten the development cycle. The FTP Client will use the Apache Commons SDK to access the server and retrieve the files and folders allowing the user to download and upload the files they need to. The FTP Client will also allow the user to save frequently used servers on the local client to make connection easier.

It is important to note; the Apache Commons Net component is open source and is provided as is. This means support the for the SDK if bugs are found will not be possible.

# Interfaces

The FTP Client does not use any external interfaces.

# Architecture

The architecture is a simple client-server; however, the FTP Client exists only on the client. The FTP Client establishes the connection to the server after which the user can browse the files and folders no the server, download files to the local client, and upload files to the server.

A screenshot of a cell phone

Description automatically generated

## Client

The client is the user’s local machine. This is where the FTP Client will be installed and where the local files and directories are located.

## FTP Client

The software described in this document. The FTP Client is responsible for establishing a connection to the server, accessing the files on the client, and providing a mechanism to upload and download files to and from the remote server. The FTP Client uses the Apache Commons NET SDK to accomplish these tasks. The FTP Client will also save frequently used server configurations on the client so the user can easily access important servers.

## Server Configurations

The server configuration is a JSON file stored on the client containing information about serves the user connects to frequently. The FTP Client provides a method for the user to save the server information so only the user selected servers are saved.

The information needed to store are

* Host – server name, IP, or URL.
* Port – the port to use for connecting to the server.
* User ID – the ID to use to connect to the server.
* Password – the password for the saved user ID. The user has an option if the password should be saved or not.

The password is saved as an encrypted string, all other fields are saved as plain text. The JSON structure of the file is shown below.

[

{

"host": "server1.domain.com",

"port": 22,

"user": "userA",

"password": "cGFzc3dvcmQ="

},

{

"host": "server2.other.domain.com",

"port": 44,

"user": "userB"

}

]

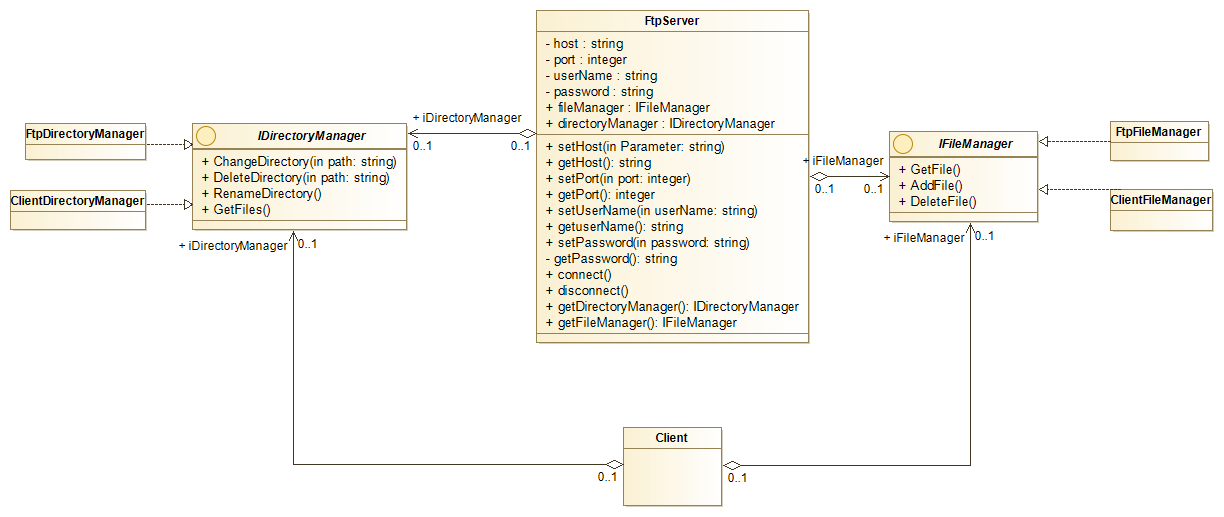
## Server

The server is the remote machine which is running FTP server software. It hosts the files the user wants to download.

# Data Model

There are no databases used in the FTP Client.

# Class Model



# Appendix A: Issues List