|  |
| --- |
| Temporary File Hosting System |
| Analysis & Design Document |

|  |
| --- |
| Project Manager: Connor Oliver  System Analyst: Jordan Campbell  System Analyst: David Campbell  System Analyst: Anson Carmody  January 29, 2016 |

# Table of Contents

[Table of Contents 1](#_Toc441830294)

[Approval 3](#_Toc441830295)

[Document Changes 4](#_Toc441830296)

[Management Summary 5](#_Toc441830297)

[Goal Statement 6](#_Toc441830298)

[Deliverables 8](#_Toc441830299)

[Assumptions 9](#_Toc441830300)

[Constraints 10](#_Toc441830301)

[Key Requirements 11](#_Toc441830302)

[Outputs 11](#_Toc441830303)

[Reports 11](#_Toc441830304)

[Inputs 11](#_Toc441830305)

[Login Information 11](#_Toc441830306)

[User Registration 11](#_Toc441830307)

[File Upload 11](#_Toc441830308)

[Data 12](#_Toc441830309)

[User Information 12](#_Toc441830310)

[File Information 12](#_Toc441830311)

[Computer Information 12](#_Toc441830312)

[Reports 12](#_Toc441830313)

[Download Information 12](#_Toc441830314)

[User agent Information 12](#_Toc441830315)

[Process 13](#_Toc441830316)

[Data Backup 13](#_Toc441830317)

[Data Access 13](#_Toc441830318)

[Encryption 13](#_Toc441830319)

[Client Side Security 13](#_Toc441830320)

[Security 13](#_Toc441830321)

[Administrative Access Controls 13](#_Toc441830322)

[Data Security 13](#_Toc441830323)

[Modeling 14](#_Toc441830324)

[System Use Case Diagram 14](#_Toc441830325)

[System Domain Class Diagram 15](#_Toc441830326)

[Event Table 16](#_Toc441830327)

[Primary Use Case Diagrams 18](#_Toc441830328)

[Primary Use Case Detailed Descriptions 22](#_Toc441830329)

[Primary Use Case Activity Diagrams 28](#_Toc441830330)

[Primary Use Case Sequence Diagrams 33](#_Toc441830331)

[Program Design 38](#_Toc441830332)

[Output Design 39](#_Toc441830333)

[Input Design 40](#_Toc441830334)

[Database Design 43](#_Toc441830335)

[Entity Relationship Diagram 43](#_Toc441830336)

[DBDL Notation 43](#_Toc441830337)

[Data Dictionary 44](#_Toc441830338)

[Support Processing Design 46](#_Toc441830339)

[Environmental Requirements 47](#_Toc441830340)

[Implementation Requirements 48](#_Toc441830341)

# Approval

**Client:** Mr. Gerald Caissy

**Project:** Temporary File Hosting System

**Start Date:** February 1, 2016

By signing this document you confirm that you have reviewed its contents and provide approval to proceed with the project as described in this document. Any desired changes after this point will be subject to review by the client and project manager.

|  |  |
| --- | --- |
| Mr. Gerald Caissy  Holland College | Date |

# Document Changes

|  |  |  |
| --- | --- | --- |
| Date | Changed By | Change Description |
| 24-01-15 | Connor Oliver | Created Document. |
| 28-01-28 | Connor Oliver | Updated wording on assumptions and constraints |
| 28-01-28 | Connor Oliver | Added project goal section |

# Management Summary

This system is designed to fill a gap in current file hosting solutions, and provide users with a useful service that gives them the features they need, but without any of the annoying things that are a part of other services.

Functional requirements of the temporary file upload system are as follows:

* Allow user to upload file(s) for a short amount of time.
* Host files for a short amount of time before deleting it from server.
* Allow anonymous downloads of uploaded files.
* Allow user to report file and by extension the file’s uploader.
* Allow user to view a gallery of public files on website and allow user to view contents of non-binary files.
* Allow admin to ban users, manually delete files, and review reports.
* Allow other applications to interact with our API.

This document provides our initial design and analysis for the implementation of a temporary file upload system. This document contains written explanations, observations, and recommendations for the system, as well as various diagrams which help show the structure of our proposed design for the temporary file upload system.

# Goal Statement

Many times a person wants to upload and quickly share a file with their friends or family, but many current file hosting services don’t fill this need. Because the user is uploading a file to share, they may not want it to stay available on the internet long term or may want to share it privately. Our goal with this project is to create a simple temporary file upload system where a user can quickly upload and share a file, but have the file automatically removed after a short period of time when they’re done sharing it.

With our system the user will be able to download a desktop side client that allows them to quickly upload files to our servers. This application will reside in the system taskbar so it’s always available for the user to upload files at a moment’s notice. The application will allow uploading of all sorts of files to our server, and when uploaded the server will return a URL to the client so they can quickly share it to someone else.

The server will provide a simple API that allows clients to easily connect to it and interact with it. This API will provide all the basic features needed for the system like login, registration, file upload, etc. By using an API we’ll be able to implement client side software on any operating system or device we wish with ease.

All files uploaded to the system will have a set amount of time they are allowed to remain on the server before being deleted. This part of the system design makes the system unique from other file hosting providers, but also significantly lowers the resources needed for server storage making it cheaper to provide the service. When a file’s lifetime is up it will initially be made unavailable for download, so that users still downloading it can complete their download, and when all connections to the file have been closed it will then be deleted from the server permanently.

The service will track users and files only as necessary to create a basic trail of files uploaded and downloaded by a user as well as computers used. The reason for keeping this data is due to the fact that with a file uploading system some users could upload illegal material, and if law enforcement were to inquire about a user they could be provided with data. By keeping this data we’re also able to create statistics about the use of our system, which could be useful long term.

The service will also have a simple web portal that displays the most recently uploaded public files, as well as basic administrator tools and the ability for people browsing the site to report files for review. Uploads will not be permitted through this web portal, only browsing uploaded files and downloading them.

# Deliverables

The final implementation of the Temporary File Upload System and we envision it will include the following deliverables:

1. A server application that provides an API for user management, file management and other system management tools.
2. A desktop tray application that allows the user to login to their account and then upload files.
3. A website that server as a gallery for files that have been uploaded publicly and that allows users to view individual files.
4. An admin panel that allows an admin to review files, reports, delete files manually, ban users, and view statistics.

# Assumptions

The following assumptions about the system, resources available, and use have been made when creating document:

1. There will be unlimited storage capacity on the server
2. Appropriate hardware running appropriate operating systems for all components will be available
3. Server will run on an optimal internet connection for file distribution
4. Client and server applications will always have an internet connection available
5. Users will not upload corrupt or malicious files
6. Users will provide us with personal to collect tracking data

# Constraints

The following constraints for the system have been identified during the design of this system and will be implemented in the project:

1. All user inputs such as login information and file information will be sanitized and validated.
2. Usernames and passwords will be encrypted/hashed when in transmission between the server and client(s).
3. There will be a maximum file size for uploads to ensure users don’t abuse the service and so as to not strain the systems resources.
4. A user will need to be authenticated with the system in order to upload files.

# Key Requirements

## Outputs

### Reports

* User statistics report
* File upload statistics report

## Inputs

### Login Information

* Administration
* Users

### User Registration

* Id
* Username
* Password salt and hash
* Email address
* Registration date

### File Upload

* File format information
* File size information

## Data

### User Information

* Id
* Username
* Password salt and hash
* Email address
* Registration date

### File Information

* File format information
* File size information

### Computer Information

* Computer Id
* MAC address
* Bios Serial
* Operating System
* Log date

### Reports

* Holds information user and file reporting

### Download Information

* IP Address
* Download Time
* Download Date

### User agent Information

* Hashed user agent Id

## Process

### Data Backup

* Regular backup provided for user information data

### Data Access

* Secure methods of accessing data

### Encryption

* Encryption for user information

### Client Side Security

* Secure method of sending user information

## Security

### Administrative Access Controls

* Employee Security Levels
  + Level 1 – Administration

### Data Security

* Data backup
* Data encryption

# Modeling

## System Use Case Diagram

This diagram depicts an overview of all the primary use case events for the system, grouped together by User Administration, File Service (upload and download), and File Removal.



## System Domain Class Diagram

This first pass diagram of the system shows relationships between the various system classes, as well as approximations of the information and functionality they will be responsible for. Most interactions with the system will be handled by either the client application or the user’s browser of choice, and will go through an application running on an external server which will handle webpage serving and data access.



## Event Table

The following event table documents the use cases which have been identified for the purposed system. The event tables shows important information about the events as well and the type of event they are, for example state, external, and temporal.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Event | Trigger | Source | Use Case | Response | Destination | Event Type |
| **User Account Management** |  |  |  |  |  |  |
| User wants to register for the system | User creation request | User | Register new user | User added to system | User  System | External |
| User wants to log in | User login request | User | User logs in to the system | User login accepted or denied | User | External  State |
| User wants to log out | User logout request | User | User logs out of the system | User is logged out | User | External  State |
| **File Management** |  |  |  |  |  |  |
| User wants to upload file | User selects file | User | User uploads file | User’s file is uploaded | System | External  State |
| System removes outdated files | Outdated file check | System | Remove outdated files | Flagged files are removed | System | Temporal |
| User wants to download file | Download link followed | User | User downloads files | File is downloaded | User | External |
| **Reporting** |  |  |  |  |  |  |
| Time to produce users registered report | “End of week, month, quarter and year” | System | User report generated | Report is generated and saved | Administration | Temporal |
| Time to produce file report | “End of week, month, quarter and year” | System | File report generated | Report is generated and saved | Administration | Temporal |
| **Website** |  |  |  |  |  |  |
| User wants to view files available | User visits web page | User | View files available | Gallery of files are displayed | User | External |

## Primary Use Case Diagrams

### View Available Files

The Spring server will host a gallery page through a static url which users will be able to follow to view public files currently in storage. The server will query the database for currently active files and display a thumbnail gallery with image links to download the files. If the files are images a thumbnail of the image will be shown, otherwise an icon identifying the file type (such as pdf or word icons) will be shown along with the file names. This gallery will be ordered by upload date and time by default.



### User Downloads File

Downloads will be accessed by users through a unique download link created for each download. Once a link is followed the server will create a new temporary file object and redirect to a new page from which the file can be downloaded to avoid providing users with direct access to the file path. From this new download page the user will be able to save the file to their local machine by normal means.



### User Uploads File

This chart shows a simple flow of events for uploading a file and from where the events are triggered. Files will be uploaded by the user through the use of a desktop application. The User will select which file to upload and in doing so will also select options for how long the file will remain available and whether or not the file will show up on the public file page. Once the user submits a file to upload the system will take the file information and store that information for identification and retrieval of the file. Once this is done the file will be uploaded from the User’s machine and downloaded by the server for storage.



### Register New User

New Users will register with the system through the desktop application. The User will enter their email, username and password. The system will verify that the email format is correct and that the username is unique. If the User’s information is valid a new User will be created and their information stored, with the password being encrypted during storage.



### Remove Outdated Files

Expired files will be removed automatically by the server application. Every minute the system will go through the list of active files and compare their expiry times to the current time. If a file has just expired it will be flagged as “unavailable”. Links to unavailable files will no longer function but the file will remain in storage for a 30-min grace period to allow anyone still downloading it to finish their download. Any files with expiry dates 30 minutes passed the check time will be removed from storage, though an inactive entry for the file will remain in the database for tracking and reporting purposes.



## Primary Use Case Detailed Descriptions

Primary use case detailed descriptions provide a more in depth view of the previously seen primary use cases. These detailed descriptions help show the workflows of each use case in more detail, conditions related to them, exception conditions, and related use cases.

|  |  |
| --- | --- |
| **Use case name:** | *User Registers* |
| **Scenario:** | Allow the user to enter a username and password and add it to the Systems database of registered users. |
| **Triggering event:** | User wishes to register a username and password |
| **Brief description:** | User enters a username and password and the System validates the username and adds it to the database if it is valid. |
| **Actors:** | User, System |
| **Related use cases:** | User logs in to system |
| **Stakeholders:** | User, System |
| **Pre-conditions:** | User must download application  User must be running application |
| **Post-conditions:** | Users is account is successfully registered in the System |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | | |  |  | | --- | --- | | 1. | User opens application | | |  |  | | --- | --- | | 1.1 | System opens to main screen | | |  |  | | |  |  | | --- | --- | | 2. | User selects registration option. | | |  |  | | --- | --- | | 2.1 | System opens registration screen and prompts for username and password. | | | |  |  | | --- | --- | | 3. | User enters username and password. | | |  |  | | --- | --- | | 3.1 | System validates username and password and returns registration confirmation. | | |
| **Exception conditions:** | |  |  | | --- | --- | | 3.1 | User name and/or password was not valid. | |

|  |  |
| --- | --- |
| **Use case name:** | *User Uploads File* |
| **Scenario:** | Allow user to upload a file to the System |
| **Triggering event:** | User would like to upload a file |
| **Brief description:** | The user selects the file they would uploaded then they select the amount of time they would like it to be available for and whether its shared status is public or private. The System then checks the file and adds it to the list of available documents. |
| **Actors:** | User, System |
| **Related use cases:** | File Download |
| **Stakeholders:** | User, System |
| **Pre-conditions:** | User must have a registered account  User must be logged in |
| **Post-conditions:** | User file is uploaded to the System and displayed in the gallery |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | | |  |  | | --- | --- | | 1. | User selects upload file option | | |  |  | | --- | --- | | 1.1 | System opens to file upload screen | | | |  |  | | --- | --- | | 2. | User selects file, time and privacy settings and submits file | | |  |  | | --- | --- | | 2.1 | System validates upload request and returns upload confirmation | | |  | |  |  | | --- | --- | | 2.2 | System adds file to server | | |  | |  |  | | --- | --- | | 2.3 | System adds file to file gallery | | |  |  | |  |  | |
| **Exception conditions:** | 2.1 File options were incorrect |

|  |  |
| --- | --- |
| **Use case name:** | *User Downloads File* |
| **Scenario:** | Allow user to download a file from the System |
| **Triggering event:** | User would like to download a file |
| **Brief description:** | User selects the file they would like to download based on a link they were provided with or a file they have selected from the Systems file gallery. |
| **Actors:** | User, System |
| **Related use cases:** | File Upload |
| **Stakeholders:** | User, System |
| **Pre-conditions:** | User must have application open or web portal open |
| **Post-conditions:** | File is downloaded on user’s device |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | | |  |  | | --- | --- | | 1. | User follows download link | | |  |  | | --- | --- | | 1.1 | System checks to see file availability | | |  | |  |  | | --- | --- | | 1.2 | File is retrieved from archive | | |  | |  |  | | --- | --- | | 1.3 | File is streamed to browser | | |  |  | | |  |  | | --- | --- | | 2. | User selects to save file | |  | |  |  | |
| **Exception conditions:** | 1.2 File may not be available |

|  |  |
| --- | --- |
| **Use case name:** | *System Deletes File* |
| **Scenario:** | The system removes files that have expired |
| **Triggering event:** | Time limit set on file expires |
| **Brief description:** | The system does a check for any outdated files that have expired. Any files that have expired are then flagged to no longer be visible or to be deleted entirely from the system based on the file life time. |
| **Actors:** | System |
| **Related use cases:** | File Upload |
| **Stakeholders:** | System |
| **Pre-conditions:** | There must be at least one file uploaded |
| **Post-conditions:** | Any files that have reached their time limit are flagged |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | |  | |  |  | | --- | --- | | 1.1 | System checks for expired file | | |  | |  |  | | --- | --- | | 1.2 | Compares lifetime to current time | | |  | |  |  | | --- | --- | | 1.3 | System flags any files that have reached expiration | | |  |  | |  | |  |  | | --- | --- | | 2. | System removes outdated files | | |  |  | |
| **Exception conditions:** |  |

|  |  |
| --- | --- |
| **Use case name:** | *User Logs In* |
| **Scenario:** | Allow user to log in to system |
| **Triggering event:** | User selects login option |
| **Brief description:** | The user selects the login option, then enter their username and password. The system then validates the user’s inputs and either allows or declines their login request. |
| **Actors:** | User, System |
| **Related use cases:** | User Registration |
| **Stakeholders:** | User, System |
| **Pre-conditions:** | User must have created an account |
| **Post-conditions:** | User will be logged in to the system |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | | |  |  | | --- | --- | | 1. | User selects log in screen | | |  |  | | --- | --- | | 1.1 | System returns log in screen and prompts for username and password | | | |  |  | | --- | --- | | 2. | User enters credentials | | |  |  | | --- | --- | | 2.1 | System validates user’s credentials | | |  | |  |  | | --- | --- | | 2.2 | User granted access | | |  |  | |  |  | |  |  | |
| **Exception conditions:** | 2.1 User enters incorrect credentials |

|  |  |
| --- | --- |
| **Use case name:** | *User Logs Out* |
| **Scenario:** | Allow user to log out of system |
| **Triggering event:** | User selects log out option |
| **Brief description:** | The user selects the log out option, the system then changes the user’s login status and restricts access. |
| **Actors:** | User, System |
| **Related use cases:** | User Log In |
| **Stakeholders:** | User, System |
| **Pre-conditions:** | User must be logged in |
| **Post-conditions:** | User will be logged out of the System |
| **Flow of activities:** | |  |  | | --- | --- | | **Actor** | **System** | | |  |  | | --- | --- | | 1. | User selects log out option | | |  |  | | --- | --- | | 1.1 | System changes login status | | |  | |  |  | | --- | --- | | 2.1 | System restricts user access | | |  |  | |  |  | |  |  | |  |  | |
| **Exception conditions:** |  |

## Primary Use Case Activity Diagrams

Available public files will be viewed by navigating to a gallery page. Once the page is requested the server will query the database and create a list of all active, public files. The list will be ordered by most recently uploaded and displayed to the user as a thumbnail gallery, where the images may be clicked to access the download link for that file.



Users will download files by following shared links in their browsers. Once a link has been followed the server will access the database to check if the link is valid and whether or not the file is still active. If the file is available the downloader’s UserAgent information will be saved for tracking purposes and a new File object will be created by the server. This file object will be served to the browser where the user will be able to save the file. If the file is no longer available or hasn’t existed show a screen telling the user so.



Users will upload files via the file upload portion of the desktop app. Users will be presented with a button to choose a file which will open a file chooser dialogue. The user will then be able to browse to the file they wish to upload. Once a file has been chosen the user may select options for the file’s lifetime and whether it will be publicly shown in the gallery page or only reachable through a shared link. Either way once the file has been submitted it will be stored on the server and the user will be given a link to share the file.



Users will begin account registration by navigating to the account section of the app. The system will prompt for new account information and after the user has filled in the forms validate the information. If information is invalid the user will be prompted to try again. Once all fields are valid a new user will be entered into the database and the user may sign in.



The system will remove outdated files by periodically querying the database and comparing the expiration times for currently active files to the current time. If the file is just expiring it will be flagged as unavailable. If the file has been expired longer than 30 minutes it will be flagged for removal. Once all files are done being checked any that are flagged for removal will be deleted from storage.



## Primary Use Case Sequence Diagrams

This diagram shows a basic breakdown of the user’s interaction with the system for accessing the public file gallery and the system messages involved with this interaction. The user requests access to the page, the server queries a list of the available files from the database and then provides links to those files to the user.



This diagram shows the basic interaction between the user and the system for downloading a file, as well as the messages sent between them during this interaction. The user follows a url link requesting the download, the server provides a page with the requested content and the user saves the file to their machine.



This diagram shows a basic breakdown of the user’s interaction with the system for uploading a file to the temporary storage application and the system messages involved with this interaction. The user chooses to upload a file and is given a dialogue allowing them to browse to the file they wish to upload. Once a file has been chosen the user may select various options about the upload, such as its lifetime and its accessibility. Once the file has been submitted to the system a unique link will be provided to download the file.



This diagram shows a basic breakdown of the user’s interaction with the system for registering a new account as well as the system messages involved with this interaction. The user decides to register a new account and the application provides a form for them to enter information. Once the user’s info has been submitted the system will validate the information and prompt again if the information is not valid. Once a user has entered valid information the system will confirm the submission and they may log in.



This diagram shows a very basic breakdown of the System’s process for automatically removing expired files. Periodically the system will check the active files and compare their expiration times to the current time. Newly expired files will be hidden and unable to be downloaded, while any files expired for more than 30 minutes will be removed from storage, though a reference to the download will remain for auditing purposes.



## Internet Architecture Diagram

This chart gives a logical overview of the Internet Architecture for the deployed system. The system itself will primarily consist of two parts: A client application running on an uploading user’s machine and a central server that accepts uploads and serves download and gallery pages. The Server Machine will provide handling for data access and file storage as well as hosting a packaged tomcat server for serving web pages. Download and Gallery pages do not need the client application installed and can be accessed from any computer with an installed web browser.



# Program Design

## Design Class Diagram

This is a more in-depth diagram of the class relationships for the proposed system. Information about the user and uploads will be handled through the client application on their machine. Gallery access and downloads will be handled by the server application, which will be responsible for serving the web links, accessing information from the database and generating reports about the system’s use.



## Sequence Diagrams

This diagram shows an in-depth depiction of the processes and messages for the user accessing the available file gallery. The user visits the page through a standard browser window which requests the file information from the server. The server loops through all the active files and adds them to a list. Once the list has been generated it is sorted and is then served to the page as a collection of download links.



This diagram shows the processes and messages for the user downloading a file from the system. The user follows a download link, whether shared by another user or accessed through the gallery. The browser requests the page for the link from the server and the server retrieves the file details corresponding to the link from the database. Since files are kept in a non-publicly-accessible location the server then creates a new DownloadLink with the path to the file. The DownloadLink object creates a file object and returns it to the server to be streamed to the browser. For file tracking purposes the browser sends the IP address and UserAgent of the downloader and then provides the file to the user to be downloaded to their machine.



This diagram shows the processes and messages for the user uploading a new file to the system. The user will access the file upload through the client application which will provide a FileChooser dialogue where the user can upload their file. Once the file has been chosen the user can select options for how long the file will remain in storage and whether or not it will show up in the public gallery. Once the file has been submitted it will be sent to the server along with the User’s information as well as some identifying information on the computer it was uploaded from, for file tracking purposes. This information can only be accessed by an installed and authorized upload client application with the user’s knowledge and authorization. Once the file is received by the server it is saved to storage and details on the file and its uploader are stored to the database. The application then confirms submission of the file to the user.



This diagram shows an in-depth depiction of the process and messages involved with registering a new account. The user accesses the registration option through the desktop client and is provided with a form to fill in their information. When the form is submitted the information is validated by the application and if cleared is sent to the server, otherwise the user is prompted to correct any errors. Once the information is received by the server a new user is created and added to the database, and the submission is confirmed to the user, allowing them to log in and begin uploading files.



This diagram shows an in-depth depiction of the internal process for automatically removing expired files, and the interactions between the server application and the database to accomplish this. Every minute the server will query all the active files in the database and compare their expiration times to the current time. If a file has just expired it will be flagged as inactive and will neither show up in the gallery if public or be available for download by following the link. Expired files will be kept for 30 minutes allowing time for ongoing downloads to finish. If a file has been expired for more than 30 minutes it will be flagged as removed and the actual file will be removed from storage, though a reference to it will still be kept in the database for file tracking purposes.



## Package Diagram

The view layer for our system is comprised of tabs in the client application which provide login and upload functionality to the user and web pages for file downloading, gallery access and reports which are served by the Server Application. Data access will be handled through SpringJPA objects which correspond to their respective database tables. The DownloadLink business layer actor will be responsible for retrieving a file from storage and providing it to the server to be served as a download page, to keep the file storage directory inaccessible to the public.



# Output Design

The following contains designs of the outputs of the Temporary File Upload Service. The designs include several reports pertaining to user information and file information as well as document download links which are sent to users.

## Reports

Reports regarding user registration numbers and file upload numbers are tracked and provided to administration through a web page. Administration has the ability to visit this page and run these reports based on the time period they have selected.

### Weekly User Information Report Documentation

SYSTEM DOCUMENTATION

NAME OF SYSTEM                                      DATE                      PAGE 1 OF 1

Temporary File Hosting System  January 29, 2016

ANALYST                                     PURPOSE OF DOCUMENTATION

A. Carmody                                     Report Analysis- Weekly User Information Report

                    FIELD             FIELD TYPE          FIELD LENGTH

                       Total Registered Users            Numeric                         10

                       New Registered Users            Numeric              10

Registered User Increase Numeric 8

COMMENTS

          1. Date is printed at the top of the report.

2. New Registered Users are based upon users that have registered during the week leading up to the date of the requested report.

TOTALS REQUIRED

1. Registered User Increase is calculated using Total Registered Users and New Registered Users.

MEDIA

          The report is generated on a webpage that is provided to the administration.

FREQUENCY

The report is printed as per the request of authorized parties.

DISTRIBUTION

The report is generated for administration purposes. Data should only be shared between trusted authorized parties.

ATTACHMENTS

Printer spacing chart and mock-up report are attached.

### Weekly User Information Report Mock-up

Temporary File Upload Service

Weekly User Information Report January 24, 2015

Week of January 17th

Total Registered Users: 14000

New Registered Users: 400

User Registration Increase: +2.85%

Active Registered Users: 645

### File Information Report Documentation

SYSTEM DOCUMENTATION

NAME OF SYSTEM                                      DATE                      PAGE 1 OF 1

Temporary File Hosting System  January 29, 2016

ANALYST                                     PURPOSE OF DOCUMENTATION

C. Oliver                                     Report Analysis-File Information Report

                    FIELD             FIELD TYPE          FIELD LENGTH

                       Total Uploaded Files            Numeric                         10

File Type Name Alphanumeric 10

File Type Total Numeric 10

File Type Distribution Numeric 10

COMMENTS

          1. Date is printed at the top of the report.

1. Total Uploaded File Type consists of many different fields depending on the amount of file types uploaded. Every file type will have its own field.

SORT SEQUENCE

          Uploaded File Types are ordered descending based on total amount of files uploaded.

TOTALS REQUIRED

1. File Type Distribution is calculated by dividing the number of upload per file type by the total number of file uploads. It is represented as a percentage of the file total.

MEDIA

          The report is generated on a webpage that is provided to the administration.

FREQUENCY

The report is printed as per the request of authorized parties.

DISTRIBUTION

The report is generated for administration purposes. Data should only be shared between trusted authorized parties.

ATTACHMENTS

Printer spacing chart and mock-up report are attached.

### File Information Report Mock-up

Temporary File Upload Service

File Upload Information Report January 24, 2015

Total Uploaded Files: 7000

Total Uploaded File Types:

TYPE Number Percentage

DOC: 2582 36.89%

PDF: 1987 28.39%

TXT: 1007 15.39%

ODT: 382 5.46%

OTT: 357 5.10%

PPS: 262 3.74%

XLSX: 189 2.70%

EPUB: 97 1.39%

ODX: 71 1.01%

PPTX: 66 0.94%

## File Download Link

The file download link is provided to users in order to download a file from the Temporary Upload System.

### File Download Link Mock-up

Temporary File Upload Service

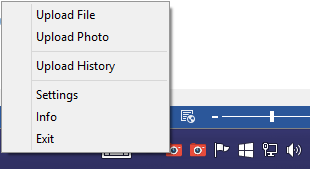
The following is a download link provided by TFUS:

Click to download Test.DOC

# Input Design

## Tray App Menu

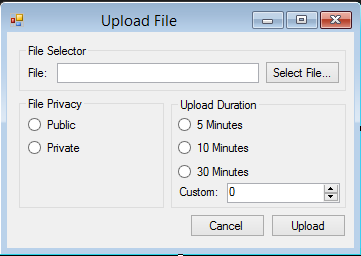
The following UI mockup shows an example of the client side tray application for uploading files.



* User can right click icon to bring up content menu as seen above
* User can select one of several options in the menu
* User can manually exist the application if desired
* While not shown, the user can use global hotkeys to select any of these options at any time

## Upload Dialog

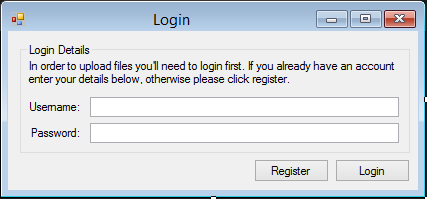
The following UI design shows an example of what the upload file dialog for the client application may look like in Windows.



* The user may select any file on their hard drive to upload
* The user may select whether to upload the file to be publicly or privately viewable
* The user may select predefined amounts of time for the file to be uploaded or select their own duration
* The user may hit cancel at any time to exit the upload dialog

## Login Dialog

The following UI design shows an example of what the user login dialog for the client application may look like in Windows.



* The user is told that they either need to login or create an account
* The user is able to enter their existing login details
* The user is given the option to login with the provided data or to click register which will open a registration screen.

# Database Design

## Entity Relationship Diagram



## DBDL Notation

Users (**UserId**, username, passwordSalt, passwordHash, email, registrationDate)

Uploads (**UploadId**, FileId, UserId, ComputerId)

FK FileId 🡪 FileDetails

FK UserID 🡪 Users

FK ComputerId 🡪 Computers

FilesDetails (**FileId**, originalName, storedName, downloadKey, fileSize, fileStatus, fileAccess, updateDate, expirationTime)

Computers (**ComputerId**, macAddress, biosSerial, operatingSystem, logDate)

Downloads (**DownloadId**, ipAddress, userAgent, downloadTime, downloadDate)

Reports (**ReportId**, FileId, details, reportTime, ipAddress)

FK FileId 🡪 Files

## Data Dictionary

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table: Users | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | userId | User ID | Integer | 10 | 0 | No | Number | Auto Increment |
|  | userName | Username | Character | 10 | 0 | No | String | None |
|  | passwordSalt | Password Salt | Character | 16 | 0 | Yes | String | Random Generation |
|  | passwordHash | Password Hash | Character | 64 | 0 | Yes | String | SHA-256 of password |
|  | email | Email | Character | 50 | 0 | Yes | String | None |
|  | registrationDate | Registration Date | Date | 8 | 0 | No | YYYYMMDD | ISO 8601 Date |
|  |  |  |  |  |  |  |  |  |
| Table: Computers | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | computerId | Computer ID | Integer | 10 | 0 | No | Number | Auto Increment |
|  | macAddress | MAC Address | Character | 12 | 0 | Yes | String | None |
|  | biosSerial | BIOS Serial | Character | 64 | 0 | Yes | String | None |
|  | operatingSystem | Operating System | Charater | 64 | 0 | Yes | String | None |
|  | logDate | Log Date | Date | 8 | 0 | No | YYYYMMDD | ISO 8601 Date |
|  |  |  |  |  |  |  |  |  |
| Table: Upload | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | uploadId | Upload ID | Integer | 12 | 0 | No | Number | Auto Increment |
| FK | fileId | File ID | Integer | 12 | 0 | No | Number | None |
| FK | userId | User ID | Integer | 10 | 0 | No | Number | None |
| FK | computerId | Computer ID | Integer | 10 | 0 | No | Number | None |
|  |  |  |  |  |  |  |  |  |
| Table: FileDetails | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | fileId | File ID | Integer | 12 | 0 | No | Number | Auto Increment |
|  | originalName | Original File Name | Character | 100 | 0 | No | String | None |
|  | downloadKey | Download Key | Character | 12 | 0 | No | String | Randomly Generated |
|  | storedName | Stored File Name | Character | 64 | 0 | No | String | None |
|  | fileSize | File Size | Integer | 10 | 0 | No | Number | Kilobyte Size of File |
|  | fileStatus | File Status | Integer | 1 | 0 | No | Number | 0: Available 1: Hidden 2: Deleted 3: Archived |
|  | fileAccess | File Access |  |  |  |  |  | 0: Public 1: Private |
|  | uploadDate | Upload Date | Date | 8 | 0 | No | YYYYMMDD | ISO 8601 Date |
|  | expirationTime | Expiration Time | Datetime | 14 | 0 | No | Datetime | None |
|  |  |  |  |  |  |  |  |  |
| Table: Reports | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | reportId | Report ID | Integer | 12 | 0 | No | Number | Auto Increment |
| FK | fileId | File ID | Integer | 12 | 0 | No | Number | None |
|  | details | Details | Character | 250 | 0 | No | String | None |
|  | reportDate | Report Date | Date | 8 | 0 | No | YYYYMMDD | ISO 8601 Date |
|  | reportTime | Report Time | Time | 5 | 0 | No | HH:MM | None |
|  |  |  |  |  |  |  |  |  |
| Table: Downloads | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | downloadId | Download ID | Integer | 12 | 0 | No | Number | Auto Increment |
| FK | fileId | File ID | Integer | 12 | 0 | No | Number | None |
| FK | userAgentId | Useragent ID | Integer | 12 | 0 | No | Number | None |
|  | ipAddress | IP Address | Character | 15 | 0 | Yes | String | None |
|  | downloadTime | Download Time | Time | 5 | 0 | No | HH:MM | None |
|  | downloadDate | Download Date | Date | 8 | 0 | No | YYYYMMDD | ISO 8601 Date |
|  |  |  |  |  |  |  |  |  |
| Table: Useragents | |  |  |  |  |  |  |  |
| Key Type | Field Name | Full Name | Datatype | Length | Decimals | Security | Format | Defined Values/Info |
| PK | userAgentId | Useragent ID | Integer | 12 | 0 | No | Number | Auto Increment |
|  | userAgentRaw | Useragent Raw | Character | Long | 0 | Yes | String | None |
|  | userAgentHash | Useragent Hash | Character | 40 | 0 | Yes | String | SHA-1 of useragent |

# Support Processing Design

First time users for the Temporary File Upload System will first need to complete and initial account registration in order to use the service. This registration will be done though the client side application and will require the user to provide a username, password and email address. We will not collect things like a user’s real name or location, because this data is unneeded as the only purpose to having an account is to track uploads relative to a computer and it’s user as well as and rate limit users.

In order to make the application, API and other parts of the system secure and to protect a user and their account all passwords will be stored hashed in the database, be hashed when in transit between parts of the system, and be salted and hashed when logging a user in.

To protect the system from malicious use most of the API endpoints will require authentication to be used. Those that won’t will only provide limited data back and won’t accept entering any data into the system. To ensure system uptime and reliability the API will also have rate limiting enabled to prevent excessive requests from a user that would use an excessive amount of system resources. Further details on system security will be discussed below in the Implementation Requirements section.

In addition to all this we will have the system setup to take regular backups of its database. We’re not going to backup files on the server because as their lifetime on the server is already limited, it’s not a huge problem if we lost any. The database is more important because it stores our tracking data and user information which would be much harder to replace in the event of system failure.

# Environmental Requirements

The implementation of the Temporary File Upload System is a brand new system therefore no consideration about existing data or infrastructure will need to be made. The environmental requirements that have been identified are as follows:

## Server Hardware

* A high speed server grade internet connection
* A server load balancer if multiple servers are to used
* A load balancer in the event of multiple servers
* Uninterruptable power supplies
* External hard drive(s) for backups
* At least 8GB ram
* At least 1TB hard drive

## Server Software

* Java 7+
* Java Spring
* Apache Tomcat
* Debian 8 Linux OS
* Postgres Database
* Automated backup script or software for database

## Client Hardware

* 512MB RAM
* High speed internet connection

## Client Software

* Windows 7+
* .Net 4.5

## Human Resources

* One system administrator to set the system up and do periodic maintenance

# Implementation Requirements

## Implementation

There will be very minimal work involved for the implementation of this system. After the server is installed and running effectively the remaining elements will mostly lie with the user. Administrative privileges will belong to the up-loader for the existing file. The system admin will oversee the user’s behaviour on the site and will monitor the security as needed.

Any passwords for logins will be encrypted and will be safely stored in the database. Only Administrators for the web page will have over riding privileges when such modifications to the web page must be made.

## Post-implementation reviews

As we are responsible for hosting this service it is in our best interest to constantly provide support during its operation. If we are pleased with the efficiency, office productivity, and user satisfaction, then we are to review what has been working well. If problems surface, the users are able to contact our team to request assistance. If the system requires changes on forms or other areas, a potential maintenance shut down may be necessary. At this time a full analysis will be done to evaluate adequate use of the system, as well as performance and security issues discovered in the early stages of deployment. The solutions to the discovered problems from the analysis will determine any lengths of shut downs if required.