SCRUM 5

Fonty’s hogeschool  ICT, S3, Infrastructure

DESIGN DOCUMENT

Contents

[Introduction 2](#_Toc181694804)

[Document history. 2](#_Toc181694805)

[Terms & abbreviation 2](#_Toc181694806)

[1. Project description 3](#_Toc181694807)

[1.1 Context 3](#_Toc181694808)

[1.1.1 File Storage & Management 3](#_Toc181694809)

[1.1.2 Data Synchronization 3](#_Toc181694810)

[1.1.3 Versioning and Backup 3](#_Toc181694811)

[1.1.4 Hybrid Cloud Implementation 3](#_Toc181694812)

[1.1.5 User Authentication & Permissions 3](#_Toc181694813)

[1.1.6 File Sharing and Collaboration 3](#_Toc181694814)

[1.1.7 Security and Compliance 3](#_Toc181694815)

[1.1.8 Scalability and Cost Optimization 4](#_Toc181694816)

[1.1.9 API for External Integrations 4](#_Toc181694817)

[2. Project scope 4](#_Toc181694818)

[2.1 Design Web Application & Features 4](#_Toc181694819)

[2.2 Informal Model 5](#_Toc181694820)

[2.3 API specifications 5](#_Toc181694821)

[2.3.1 Search 5](#_Toc181694822)

[2.3.2 User Authentication 6](#_Toc181694823)

[2.4 Functional requirements 6](#_Toc181694824)

[2.4.1 Main functionality/ Goal of website and the cloud 6](#_Toc181694825)

[2.4.2 Functional Requirements 6](#_Toc181694826)

[2.5 Administrative Functions 8](#_Toc181694827)

[2.6 Code requirements 8](#_Toc181694828)

[2.7 Hosting web application 8](#_Toc181694829)

[3. Risk management 9](#_Toc181694830)

[3.1 Configuration management 9](#_Toc181694831)

[4. Use cases & User stories 9](#_Toc181694832)

[4.1 User Stories 9](#_Toc181694833)

[4.2 Use Cases 14](#_Toc181694834)

[5. Deliverables per sprint 17](#_Toc181694835)

[References 18](#_Toc181694836)

# Introduction

This is a live document for the design of this application. Changes might be made to the design of this document and/or implemented differently and added later. For more information of the stakeholder, working method and users, we would like to refer to the Project Goal document for SCRUM 5. However functional requirements, use cases and user stories are included in this document.

# Document history.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Author | Date | Description | Status |
| 0.1 | Isabelle Gruijs | 3/11/2024 | First draft Design Document |  |
|  |  |  |  |  |

# Terms & abbreviation

|  |  |
| --- | --- |
| Terms | Abbreviation |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 1. Project description

### 1.1 Context

This group project is going to be a web application that acts like a Dropbox platform. It allows you to download and upload files and manage and store them.

### 1.1.1 File Storage & Management

Store user files in Amazon S3, with options for encrypted storage (using S3’s server-side encryption or client-side encryption). Users can share files using pre-signed URLs or role-based access control (RBAC) to restrict access to specific users or groups. On-premises file storage can be used for highly sensitive data with secure synchronization to AWS using AWS Storage Gateway.

### 1.1.2 Data Synchronization

Implement a system where files can be seamlessly synced between on-premises storage and AWS S3. AWS DataSync or Storage Gateway can be used for efficient data movement, ensuring minimal latency for critical files stored on-premises.

### 1.1.3 Versioning and Backup

Use S3’s built-in versioning feature to track file changes and enable recovery of previous versions. Set up automatic backups using AWS Backup for cloud-stored data and a similar solution for on-premises storage.

### 1.1.4 Hybrid Cloud Implementation

Provide local file access using an on-premises file system like NFS or SMB, integrated with AWS Storage Gateway, allowing files to be cached locally but stored securely in the cloud for global access and disaster recovery.

### 1.1.5 User Authentication & Permissions

Utilize AWS IAM for managing user permissions in the cloud. On-premises Active Directory or LDAP servers can be integrated for user authentication, with AWS Managed Microsoft AD as a potential bridge between on-premises and cloud-based user authentication. Make sure your solution doesn't imply keeping access keys on the users’ hard drives

### 1.1.6 File Sharing and Collaboration

Build a sharing module that allows users to invite others to collaborate on files. Use pre-signed URLs with expiration for time-bound file sharing or OAuth2 for secure, long-term collaboration access.

### 1.1.7 Security and Compliance

Implement stringent security measures such as file encryption, multi-factor authentication (MFA), and logging using AWS CloudTrail. On-premises systems would adhere to local compliance needs, while cloud infrastructure could follow standards like HIPAA, GDPR, or SOC 2.

### 1.1.8 Scalability and Cost Optimization

Use S3 storage classes (Standard, Intelligent-Tiering, or Glacier) to optimize costs depending on file access patterns. Enable lifecycle policies to archive infrequently accessed files to cheaper storage automatically. On-premises storage can be used for high-frequency, low-latency access.

### 1.1.9 API for External Integrations

Develop an API that allows other apps or services to upload, manage, or retrieve files programmatically, making the service extensible for various business or academic use cases. AWS API Gateway and Lambda can be used for managing API requests efficiently.

# 2. Project scope

The functionality and accessibility are important. Below there are some important points of the inside scope.

|  |  |
| --- | --- |
| **Inside scope:** | **Outside scope:** |
| 1. Functionality & Accessibility | 1. Managing environment after delivery |
| 1. Prevential Problem Management | 1. Supporting older OS |
| 1. Availability Management | 1. No updates |
| 1. Security | 1. Supports plugins |
| 1. Web application hosting | 1. Checkpoints (snapshots) |
| 1. Database for login |  |
| 1. Cloud environment |  |
| 1. Support Multiplatform OS |  |

## 2.1 Design Web Application & Features

This is the current slight preview of the application and the services that it offers. It should be able to post, get, edit and possibly delete, which are basic API functions.

Some considerations that must be made is with the logins is security within the code, as well with the information that is displayed. Since it is a web application that will later be deployed in a cloud environment, there are other security managing options available. Below you can find the design of the application of the functional requirements of the application.

The application will apply the CRUD method of POST, GET, PUT and delete. Below is a table with the different queries that will be included in the application. This is currently not relevant for this document, but this will further be enlightened in the Design Document with different queries

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C | R | U | D | OPERATIONS |
| CREATE | READ | UPDATE | DELETE | x |
| POST | GET | PUT | DELETE | HTTP |
| INSERT | SELECT | UPDATE | DELETE | RDS |

## 2.2 Informal Model

**Compatibility Framework**

Description: The application is a web application, so each user should be able to reach the platform.

**Data Collection**

Description: The application uses some processes to gather information such as what you uploaded or want to download. These processes might involve checking system APIs or running commands to collect relevant data.

**User Interface Design**

Description: The dashboard design adheres to a structured framework focusing on user-friendliness and accessibility. It could use front-end tools like LEGO blocks or building blocks to ensure a well-organized layout and presentation.

**Search Organization**

Description: The application employs a method for efficiently searching software versions called a search organization technique. This method organizes relevant data elements and enables quick retrieval based on user queries. This can be done within the web application.

**Authentication Method**

Description: User authentication is handled using a formal method such as OAuth2. This method outlines the steps for verifying users, issuing access tokens, and enforcing access rules to protect the application's security. Amazon Cognito is a strong contender for this implementation.

## 2.3 API specifications

### 2.3.1 Search

* + **Endpoint:** **/search**
  + **Method:** GET
  + **Description:** Search for specific hardware or software components.
  + **Input:**
    - **Query**: The component or software name to search for.

### 2.3.2 User Authentication

* + **Endpoint:** **/login**
  + **Method:** POST
  + **Description:** Authenticates the user to access vFetcher services.
  + **Input:**
    - **username**: The user's username.
    - **password**: The user's password.

## 2.4 Functional requirements

### 2.4.1 Main functionality/ Goal of website and the cloud

The website's primary objective is to provide a secure interface that allows users to upload and download files to and from AWS S3 buckets, Aswell as monitoring costs and logs. The system must ensure secure file transfer, access control, and data integrity while interacting with AWS S3 storage in a cost-efficient manner.

### 2.4.2 Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirements: | Description: | Priority: |
| The system must provide secure user authentication | Users must create an account and log in to gain access to the file upload and download functionalities. | High |
| The system must enforce role-based access control (RBAC) to determine who can upload, download, or view files. | Different user roles (e.g., admin, regular user) will have different permissions for accessing files. | High |
| Users must be able to upload files securely to specific AWS S3 buckets. | Provide a user-friendly interface where users can select files from their local device and upload them to AWS S3. | High |
| The system must support multiple file types for upload. | images, PDFs, documents, and videos. | Medium |
| The system should limit the size of each file upload | Administrators should be able to configure the maximum file size allowed for upload (e.g., 10MB). | Medium |
| The system should display upload progress and success/failure messages to the user. | Show real-time progress indicators and messages for successful uploads or errors. | Medium |
| Files must be automatically encrypted using server-side encryption when uploaded to AWS S3 | (e.g., AES-256 encryption). | High |
| Users must be able to download files securely from AWS S3 buckets. | After login, users can view a list of files they have access to and download them to their local device. | High |
| The system should log all download actions for auditing purposes. | Maintain a log of when files were downloaded, by whom, and from which IP address. | Medium |
| Users should be able to view a list of uploaded files with details. | such as file name, size, upload date, and uploader. | Medium |
| The system should support file versioning in AWS S3. | If the same file is uploaded multiple times, previous versions should be retained in the S3 bucket (versioning enabled). | High |
| Users should be able to delete files securely from the S3 bucket, with access restrictions in place. |  | Medium |
| The system must support SSL/TLS encryption for all data transfers between the website and AWS S3. | Files should be encrypted both in transit and at rest (e.g., SSL in transit, S3 server-side encryption at rest). | High |
| Users should receive notifications (e.g., email, on-screen) when file uploads or downloads are successful or encounter errors. |  | Low |
| The system must log all critical operations, including uploads, downloads, and file deletions, for auditing purposes. | Use AWS CloudTrail/CloudWatch and server-side logging to track user actions. | High |
| MYSQL Database | to track user information, file metadata, and access logs. | High |
| Some Automation to create Infrastructure | Ansible to create infrastructure and docker for containerization | High |
| Frontend and Backend security | Public Frontend and private Backend in separate subnets/VPCs. | High |
| Demonstrate the upgrade of HTTP to HTTS connection without any changes to EC2 instances (use of Certification Manager and load balancer) |  | High |
| Cost management System | Have alerts and a monitoring system for all costs | High |
|  |  |  |

## 2.5 Administrative Functions

There should be two administrators. A platform administrator and a system administrator.

Users with a platform administrator authorization should be able to create new credentials for login purposes besides the log in query, updating account information, and removing accounts. As well as possibly freezing accounts that have been possibly breached.

Users with system administrators should be allowed on to the console of AWS, as well as maintaining the system required to host the application. The role should also include the rights to update, delete and make changes to the users.

Users should be able to download and upload files, and search from results of the output.

## 2.6 Code requirements

-

-

-

## 2.7 Hosting web application

Currently it will be hosted on AWS solely, with the intend to make it hybrid for redundancy and high availability purposes.

# 3. Risk management

Some of the risks include data leaks, software problems, and issues with different systems. The web application will stay safe by using strong security, updating our software regularly, and testing it well.

Security is a big part of infrastructure, for the web application there are some countermeasures and risks, but currently there is no specific details of actual potential problems arising.

*X = Little to no impact.*

*X = Medium impact.*

*X = Big Impact.*

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Impact | Countermeasures |
| Leaks in code | 50/100 | x | Security testing code |
| Cloud intrusion | 1/100 | x | Testing environment |
| Credential leaks | 70/100 | x | Isolate leaks |
| Host infection | 50/100 | x | No infected data traffic |

## 3.1 Configuration management

Utilize Git for version control and host repositories on platforms like GitHub. Carefully manage configuration files for the database, security settings, and AWS hosting to ensure the consistency and reliability.

# 4. Use cases & User stories

Below are twelve descriptions for user stories, and three use cases for the end goal for the project.

### 4.1 User Stories

1. File Upload & Storage

* **User Story**

*As a user, I want to upload files to cloud storage securely, so that I can access my files from any device.*

* + **Acceptance Criteria**
    - Users can drag and drop files into the application to upload.
    - Files are uploaded securely to Amazon S3 with encryption (client-side or server-side).
    - The system displays progress indicators during uploads.
    - Upon completion, users receive a confirmation notification.

2. File Sharing with Pre-Signed URLs

* **User Story**

*As a user, I want to share files with others using pre-signed URLs, so that I can securely share time-bound access without giving permanent permissions.*

* + **Acceptance Criteria**
    - Users can generate pre-signed URLs with customizable expiration times.
    - URLs grant access to specific files only, without requiring the receiver to log in.
    - Expired URLs are automatically invalidated after the set time.

3. Role-Based Access Control (RBAC)

* **User Story**

*As a user, I want to control access to my files using role-based permissions, so that I can restrict access to specific users or groups.*

* + **Acceptance Criteria**
    - Users can assign roles (e.g., viewer, editor, admin) to individuals or groups for each file or folder.
    - Permissions are enforced both on-premise and in the cloud using AWS IAM.
    - Users can modify permissions at any time, and changes are reflected instantly.

4. On-Premise File Synchronization

* **User Story**

*As a system admin, I want to sync on-premise files with AWS S3, so that sensitive data can be stored locally but also backed up in the cloud.*

* + **Acceptance Criteria**
    - AWS Storage Gateway is configured to automatically sync changes between on-premise and cloud storage.
    - Users experience minimal latency when accessing high-priority files stored on-premise.
    - Changes to files are synced in real-time, with no manual intervention needed.

5. File Versioning & Recovery

* **User Story**

*As a user, I want to access previous versions of my files, so that I can recover older versions if needed.*

* + **Acceptance Criteria**
    - Each file version is stored using S3’s versioning feature.
    - Users can view and restore previous versions directly from the file interface.
    - Automatic backups are performed using AWS Backup for cloud files and a similar solution for on-premise storage.

6. Local File Caching for Fast Access

* **User Story**

*As a user, I want to access frequently used files quickly via local storage, so that I can work with minimal delays.*

* + **Acceptance Criteria**
    - Frequently accessed files are cached on-premise using AWS Storage Gateway.
    - Users can access cached files with low latency through local file systems like NFS or SMB.
    - Files not accessed recently are offloaded to S3 and can be accessed when needed.

7. User Authentication via AWS IAM & Active Directory

* **User Story**

*As an admin, I want to integrate on-premise Active Directory with AWS IAM, so that users can authenticate securely across both environments without duplicating credentials.*

* + **Acceptance Criteria**
    - Users authenticate using existing on-premise Active Directory credentials.
    - AWS Managed Microsoft AD bridges on-premise and cloud authentication.
    - Multi-factor authentication (MFA) is enforced for additional security.
    - No sensitive access keys are stored on users’ local machines.

8. File Sharing & Collaboration with OAuth2

* **User Story**

*As a user, I want to invite collaborators to work on files, so that we can edit and manage files together securely.*

* + **Acceptance Criteria**
    - Users can send invitations for collaboration via email.
    - Collaborators log in using OAuth2, without needing to create new accounts.
    - Role-based permissions (viewer, editor) can be set for each collaborator.
    - Collaboration activities (file edits, comments) are logged for transparency.

9. File Encryption & Security

* **User Story**

*As a security officer, I want all files to be encrypted in transit and at rest, so that sensitive data is protected from unauthorized access.*

* + **Acceptance Criteria**
    - All files are encrypted using S3’s server-side encryption (SSE-S3 or SSE-KMS) or client-side encryption (CSE).
    - Files are transmitted securely using SSL/TLS during uploads and downloads.
    - Access to files is logged using AWS CloudTrail for auditing purposes.
    - On-premise files adhere to the same encryption and security standards.

10. Lifecycle Management for Cost Optimization

* **User Story**

*As a user, I want infrequently accessed files to be automatically archived, so that I can reduce storage costs without losing access to my data.*

* + **Acceptance Criteria**
    - Files not accessed in a specified period are moved to S3 Glacier automatically.
    - Users are notified when files are archived and can restore them if needed.
    - Lifecycle policies are customizable by folder or file type.

11. API for External Integrations

* **User Story**

*As a developer, I want to use an API to programmatically upload and manage files, so that I can integrate the service into my own application.*

* + **Acceptance Criteria**
    - The API allows for file uploads, downloads, and metadata management.
    - API requests are authenticated using OAuth2 or API keys, depending on the use case.
    - AWS API Gateway and Lambda are used to efficiently handle requests, with rate limits in place.

12. Compliance with Security Standards (HIPAA, GDPR, SOC 2)

* **User Story**

*As a compliance officer, I want the file storage solution to meet industry standards such as HIPAA, GDPR, and SOC 2, so that I can ensure the service is secure and legally compliant.*

* + **Acceptance Criteria**
    - Files in the cloud are stored in AWS regions that comply with data residency regulations (e.g., GDPR).
    - Auditing and logging features (AWS CloudTrail) are enabled for tracking access to sensitive files.
    - The system supports data encryption, multi-factor authentication, and secure file-sharing features to meet compliance requirements.
    - Regular security audits and vulnerability scans are conducted to maintain compliance.

### 4.2 Use Cases

#### Use Case 1

**Upload and Manage Files**

* **Name:** Upload and Manage Files
* **Actors:** User, AWS S3
* **Preconditions:** User is logged into the system and has available storage space.
* **Main Flow**
  1. User selects the file(s) to upload from their device.
  2. The system initiates the upload and shows progress indicators.
  3. The file is uploaded to AWS S3, with encryption applied.
  4. Once the upload is complete, the file appears in the user's file management dashboard.
  5. User can rename, delete, or move the file into folders.
* **Alternative Flow**
  1. If the upload fails, the system retries or notifies the user of the error.
* **Postconditions:** The file is securely stored in AWS S3 and accessible via the user's dashboard.
* **Trigger:** User initiates a file upload.

**A screenshot of a computer screen

Description automatically generated**

#### Use Case 2

**Share File with Pre-Signed URL**

* **Name:** Share File with Pre-Signed URL
* **Actors:** User, AWS S3
* **Preconditions:** User is logged into the system and owns the file to be shared.
* **Main Flow**
  1. User selects a file to share.
  2. The system generates a pre-signed URL with an expiration time.
  3. User copies the URL and shares it with the recipient.
  4. The recipient accesses the file via the URL.
* **Alternative Flow**
  1. If the URL expires, the recipient cannot access the file and receives an error message.
* **Postconditions:** The file is shared securely via a time-limited URL.
* **Trigger:** User selects a file and requests to share it.

**A diagram of a software

Description automatically generated**

#### Use Case 3

**Sync On-Premise Files with Cloud Storage**

* **Name:** Sync On-Premise Files with Cloud Storage
* **Actors:** System Admin, AWS Storage Gateway
* **Preconditions:** On-premise file system is configured with AWS Storage Gateway.
* **Main Flow**
  1. Admin configures file folders for synchronization with AWS S3.
  2. The system automatically syncs files between on-premise storage and the cloud.
  3. Changes made to files on-premise are reflected in the cloud, and vice versa.
  4. Users can access synced files from both local storage and cloud storage.
* **Alternative Flow**
  1. If the connection to the cloud fails, changes are queued for syncing when the connection is restored.
* **Postconditions:** Files are synced between on-premise storage and AWS S3, ensuring data availability and backup.
* **Trigger:** Admin config

A diagram of a string

Description automatically generated

# 5. Deliverables per sprint

The end product of the second (1) sprint will be the front end and some of the features usable by the user thus far.

The end product of the third sprint (2) will contain the deliverable of sprint 1 and authentication and authorization included in the second sprint.

The end product of the fourth sprint (3) will contain every deliverable of sprint 1 and 2, as well as the goal discussed with the client/stakeholder.

The end product in the last sprint (final delivery) is a web application allows you to log in to your account and for you to upload and download files, in a secure and easy manner.

# References

*Methods DOT framework:* [*https://www.ictresearchmethods.nl/*](https://www.ictresearchmethods.nl/)