Hawkeye Software Design

V1.0

Contents

[Amendment Records 4](#_Toc209449151)

[Architecture 5](#_Toc209449152)

[Roles 6](#_Toc209449153)

[Requestors 6](#_Toc209449154)

[Designers 6](#_Toc209449155)

[Root [OC/DOC] 6](#_Toc209449156)

[Business Flows 6](#_Toc209449157)

[Requestor Flow 6](#_Toc209449158)

[Designer Flow 7](#_Toc209449159)

[Root [OC/DOC] Flow 7](#_Toc209449160)

[Technology Stacks & Tools 8](#_Toc209449161)

[Identity Access Management [Keycloak] 9](#_Toc209449162)

[oAUTH2 9](#_Toc209449163)

[Default Token Settings 10](#_Toc209449164)

[Default Session Settings 10](#_Toc209449165)

[Groups & Users 10](#_Toc209449166)

[Software Security 11](#_Toc209449167)

[CDR>RL>CDR Return>Alert>Approve 11](#_Toc209449168)

[Database Security 12](#_Toc209449169)

[1. Network Security 🌐 12](#_Toc209449170)

[2. Authentication and Authorization 👥 12](#_Toc209449171)

[3. Encryption 🛡️ 13](#_Toc209449172)

[Air Flow Workflow 13](#_Toc209449173)

[Distributed File System [GlusterFS] 13](#_Toc209449174)

[Introduction 13](#_Toc209449175)

[Why? 14](#_Toc209449176)

[Setup Steps 14](#_Toc209449177)

[Software Design 17](#_Toc209449178)

[Clean Architecture 17](#_Toc209449179)

[Database Design 19](#_Toc209449180)

# Amendment Records

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Item** | **Author** | **Remarks** |
| 10/9/2025 | Initial | Eric See |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Architecture

A diagram of a software company

AI-generated content may be incorrect.

# Roles

## Users

* Login to system
* Create requests
* Download request forms to telco
* Upload returned data
* View reports
* View Route Lists
* View Archived Requests & Route Lists

## Designers

* Design form templates
* Design base Tables
* Approved extra fields

## Root [OC/DOC]

* Create Users
* View Audit Logs
* View Alerts

# Business Flows

## Users Flow

* Login
* Inbox
  + Routine List updated alerts
* Request
  + Upload Attachment to extract text
  + Select Fields
  + Select Telco
  + Download Request Form
* Send to Telco Manually
* Upload Returned Request Form
  + Confirm Extra Fields

## Designer Flow

* Login
* Inbox
  + Approved Extra Fields Requests
  + Go to Master Template to confirm
* Go to Base Tables
  + Add new value to List
* Go to Master Templates
  + Add telco

## Root [OC/DOC] Flow

* Login
* Inbox
  + Alerts
* View Audit Logs
* CRUD users

# Technology Stacks & Tools

* Golang 1.25
* Python 3.13.7
* NodeJS 24
* Keycloak 26.3.3
* Airflow 2.9
* Nginx 1.29.1
* NextJS 15.1
* MongoDB Server CE 7.0
* Redis 8.0
* Elasticsearch 9.1.3
* Kibana 9.1.3
* Grafana 12.0
* Prometheus 3.6.0
* GlusterFS 11.0
* Gitlab CE 18.0.1
* SonarQube Enterprise Server 10.8
* Altassian Jira

# Identity Access Management [Keycloak]

### oAUTH2

A diagram of a process

AI-generated content may be incorrect.

1. The application requests authorization to access service resources from the user

2. If the user authorized the request, the application receives an authorization grant

3. The application requests an access token from the authorization server (API) by presenting authentication of its own identity, and the authorization grant

4. If the application identity is authenticated and the authorization grant is valid, the authorization server (API) issues an access token to the application. Authorization is complete.

5. The application requests the resource from the resource server (API) and presents the access token for authentication

If the access token is valid, the resource server (API) serves the resource to the application

### Default Token Settings

* **Access Token Lifespan**: By default, an access token is valid for **5 minutes**. ⏳ This short lifespan is a security best practice, as it minimizes the window of opportunity for a compromised token to be misused.
* **Refresh Token**: The refresh token's lifespan is tied to the SSO session. It is used to get a new access token once the current one expires, without requiring the user to log in again.

### Default Session Settings

* **SSO Session Idle**: The default idle timeout for a user's session is **30 minutes**. 😴 If a user is inactive for this period, their session will expire, and they will need to re-authenticate to continue.
* **SSO Session Max**: The maximum lifespan of a user's session, regardless of activity, has a default of **10 hours**. ⏰ After this time, the user is automatically logged out. This is a hard limit to ensure sessions do not persist indefinitely.
* **Offline Session Idle**: For offline tokens, the default idle timeout is **30 days**. This determines how long an offline token can remain unused before being revoked. This is particularly relevant for applications that require long-lived access.
* **Offline Session Max Limited**: By default, this is **off**. When enabled, it sets a maximum lifespan for offline sessions.

### Groups & Users

|  |
| --- |
| **Groups** |
| requestor |
| designer |
| admin |

|  |
| --- |
| **Roles** |
| add\_user |
| update\_user |
| delete\_user |
| view\_user |
| all\_user |
| add\_template |
| update\_template |
| delete\_template |
| view\_template |
| all\_template |
| add\_cdr\_request |
| add\_ctc\_request |
| add\_ss\_request |
| add\_all |
| upload\_cdr\_return\_data |
| upload\_ctc\_return\_data |
| upload\_ss\_return\_data |
| view\_audit\_log |
| view\_archived |
| add\_base\_list |
| update\_base\_list |
| view\_base\_list |
| approved\_extra\_fields |
| confirm\_extra\_fields |

# Software Security

## CDR>RL>CDR Return>Alert>Approve

A diagram of a diagram

AI-generated content may be incorrect.

# Database Security

## 1. Network Security 🌐

This is your first line of defense. By default, MongoDB might be configured to listen on all network interfaces, which is a major security risk.

* **Bind to Specific IPs:** Configure MongoDB to only accept connections from trusted IP addresses. The bindIp setting in the mongod.conf file should be set to 127.0.0.1 for local connections or to the specific IP addresses of your application servers.
* **Firewall Rules:** Use a firewall to restrict access to the MongoDB port (default 27017). Only allow connections from your application servers and trusted administrators.
* **Change Default Port:** While not a security silver bullet, changing the default port can help deter automated port-scanning attacks.

## 2. Authentication and Authorization 👥

Authentication verifies who a user is, and authorization determines what they can do. Both are off by default in older MongoDB versions and must be enabled.

* **Enable Access Control:** You must enable authentication in the mongod.conf file by setting security.authorization: enabled. Once enabled, no one can access the database without a valid user account.
* **Create Users:** Create a separate user for each application or service that needs to access the database. The first user you create should be an administrator with the userAdminAnyDatabase role.
* **Role-Based Access Control (RBAC):** This is MongoDB's primary authorization mechanism. Instead of granting permissions directly to users, you create **roles** with specific privileges (e.g., read, readWrite) and then assign those roles to users. This follows the principle of least privilege, ensuring an application or user only has the permissions it absolutely needs to function.

## 3. Encryption 🛡️

Encryption protects your data both when it's being sent over the network and when it's stored on disk.

* **Encryption in Transit (TLS/SSL):** By default, data transmitted between MongoDB clients and servers is not encrypted. To prevent man-in-the-middle attacks, you should enable TLS/SSL (Transport Layer Security/Secure Sockets Layer) for all connections. This encrypts the data as it travels across the network.
* **Encryption at Rest:** This secures your data files on the disk, making them unreadable to anyone who gains unauthorized access to the storage. MongoDB Enterprise and MongoDB Atlas offer built-in encryption at rest using the **WiredTiger storage engine**. For other versions, you can use filesystem-level or disk-level encryption.
* **Client-Side Field-Level Encryption (CSFLE):** This is the most robust form of encryption. It allows your application to encrypt sensitive data fields **before** they are sent to the database. The data remains encrypted in transit and at rest, and only your application, with the correct encryption keys, can decrypt it.

# Air Flow Workflow

Process uploaded telco returned zip files

# Distributed File System [GlusterFS]

## Introduction

GlusterFS is an open-source, scalable, distributed file system. It's designed to aggregate storage from multiple servers into a single, large global namespace. Unlike traditional distributed file systems, GlusterFS has a "shared nothing" architecture, meaning it doesn't rely on a central metadata server, which eliminates a single point of failure and a potential performance bottleneck. This architecture allows it to scale linearly and provides high availability.

## Why?

GlusterFS is a popular choice for several reasons, particularly in environments that need flexible, scalable, and resilient storage without high costs.

* **Scalability:** You can easily add more storage capacity and performance by simply adding new servers (nodes) and their associated storage (bricks) to the cluster. This "building block" approach allows you to start small and grow to petabyte-scale storage as your needs increase.
* **High Availability and Redundancy:** GlusterFS provides different volume types to ensure data redundancy. A **replicated volume** mirrors data across multiple bricks, so if one server fails, the data is still available from its replica. A **distributed replicated volume** combines this redundancy with scalability, spreading replicated data across multiple sets of servers.
* **Cost-Effective:** GlusterFS can be built on **commodity hardware**. You don't need expensive, proprietary storage arrays. This makes it a highly economical solution for building large-scale storage systems.
* **No Central Metadata Server:** This is a key architectural advantage. By avoiding a central server, GlusterFS eliminates a single point of failure and a potential performance bottleneck. It uses an elastic hashing algorithm to locate files, allowing for high performance and linear scalability.
* **Flexibility:** GlusterFS is POSIX-compliant, which means it behaves like a standard file system. It can also use any on-disk file system (like XFS or ext4) that supports extended attributes. It's accessible via standard protocols like NFS and SMB.

## Setup Steps

**1.Install and Configure GlusterFS**

The installation process is similar on most Linux distributions.

1. **Install the GlusterFS server package** on **all** your server nodes. On Ubuntu, you'd use sudo apt-get install glusterfs-server, while on CentOS/RHEL, you'd use sudo yum install glusterfs-server.
2. **Start and enable the GlusterFS daemon** (glusterd) on all nodes to ensure it runs automatically at boot time. Use sudo systemctl start glusterd and sudo systemctl enable glusterd.
3. **Prepare a storage brick on each server.** A brick is a directory that GlusterFS will use to store files. It is best practice to use a separate partition formatted with a file system like XFS for this. Create a mount point and then a directory for the brick. For example: sudo mkdir -p /data/brick1/gv0.

**2. Create a Trusted Storage Pool**

This step establishes trust between your servers, allowing them to form a cluster. You only need to run this command from **one** of the servers.

1. From your first server, use the gluster peer probe command to add the other server(s) to the trusted pool. For example, sudo gluster peer probe <hostname\_or\_IP\_of\_second\_server>.
2. You can verify the status of the peers with the gluster pool list command. You should see all your servers connected in the pool.

**3. Create a GlusterFS Volume**

A volume is the logical file system that clients will mount. GlusterFS supports several volume types, each with a different purpose.

* **Distributed Volume:** Spreads files across bricks without replication. It offers maximum storage capacity but no data redundancy. A file is stored on only one brick.
* **Replicated Volume:** Creates a mirror of the data on all bricks in the volume. This provides high availability and data redundancy, but the total usable storage is limited to the size of a single brick. The number of bricks must be an even number.
* **Distributed Replicated Volume:** Combines both of the above. It distributes files across sets of replicated bricks, providing both scalability and high availability. The total number of bricks must be a multiple of the replica count. This is a common choice for production environments.

To create a **replicated volume** (a good starting point for high availability), run this command from any server in the trusted pool.

sudo gluster volume create <volume\_name> replica 2 <server1\_IP>:/data/brick1/gv0 <server2\_IP>:/data/brick1/gv0

Once the volume is created, you must **start it** for it to become active: sudo gluster volume start <volume\_name>.

You can view the volume's status and details with sudo gluster volume info.

**5. Mount the DFS on a Client**

Once the volume is created and started, clients can mount it to access the shared storage.

1. **Install the GlusterFS client package** on the client machine: sudo apt-get install glusterfs-client or sudo yum install glusterfs-client.
2. **Create a mount point** directory on the client: sudo mkdir -p /mnt/glustervol.
3. **Mount the volume** using the mount command with the GlusterFS native client: sudo mount -t glusterfs <server\_IP>:/<volume\_name> /mnt/glustervol.
4. To ensure the volume is automatically mounted at system startup, add an entry to the /etc/fstab file: <server\_IP>:/<volume\_name> /mnt/glustervol glusterfs defaults,\_netdev 0 0. The \_netdev option prevents the system from attempting to mount the volume before the network is available.

# Software Design

# Clean Architecture

The structure breaks down the application into logical layers, which is a key principle of Clean Architecture and its related patterns (like Hexagonal and Onion Architecture). Here's a breakdown of how the provided structure maps to Clean Architecture's layers:

* **scripts/services/**: This likely corresponds to the **Application Layer** or **Use Cases**. This is where the core business logic and specific application rules would reside. These services define the "what" the application does.
* **scripts/routers/**: This folder acts as the **Interface Adapters** or **Presentation Layer**. In a web application, this is where you'd handle incoming requests (e.g., from a REST API) and translate them into a format that the services layer can understand.
* **scripts/models/**: This could represent the **Entities** or **Domain Layer**. These would be the data structures or business objects that encapsulate the core business rules and are independent of any framework, database, or UI.
* **docs/**, **scripts/utils/**, **scripts/config/**, **.env**, **Dockerfile**, **requirements.txt**, **mongo\_update.py**: These files and directories represent the **Frameworks and Drivers** or **Infrastructure Layer**. They handle the "how"—how the application is deployed, configured, interacts with the database (mongo\_update), and handles external concerns. The mongo\_update.py is a particularly strong indicator of this layer, as it directly deals with a specific database implementation (MongoDB).

|  |
| --- |
| data2/  ├── docs/  ├── scripts/  │ ├── \_\_pycache\_\_/  │ ├── config/  │ ├── logs/  │ ├── models/  │ ├── routers/  │ ├── services/  │ └── utils/  ├── .env  ├── api\_server.py  ├── dependencies.py  ├── make\_aws\_data\_key.py  ├── make\_data\_key.py  ├── master-key.txt  ├── mongo\_update.py  ├── .dockerignore  ├── Dockerfile  ├── readme.md  └── requirements.txt |

# Database Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **user** |  | **setting\_info** |  |  |
| id |  | id |  |  |
| email |  | color\_theme |  |  |
| first\_name |  | abv |  |  |
| last\_name | |  | | --- | |  | | dashboard\_details |  |  |
| mobile |  | created\_at |  |  |
| created\_at |  | updated\_at |  |  |
| updated\_at |  |  |  |  |
| role |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **alert** |  | **audit\_log** |  |  |
| id |  | user\_id |  |  |
| event\_type |  | action |  |  |
| links |  | remarks |  |  |
| content |  | created\_at |  |  |
| created\_at |  | updated\_at |  |  |
| updated\_at |  | created\_by |  |  |
| status [Read, New] |  |  |  |  |
| user\_id |  |  |  |  |
|  |  |  |  |  |
| **toi** |  | **mobile\_info** |  | **call\_log** |
| id |  | id |  | id |
| first\_name |  | telco |  | duration |
| last\_name |  | mobile |  | telco |
| sin |  | imei |  | call\_location |
| created\_at |  | created\_at |  | start\_time |
| updated\_at |  | updated\_at |  | end\_time |
| created\_by |  | created\_by |  | call\_direction |
| []mobile\_info | |  | | --- | |  | | end\_at |  | created\_at |
| []call \_log |  |  |  | updated\_at |
|  |  |  |  | created\_by |
|  |  |  |  |  |
|  |  |  |  |  |
| **telco** |  | **template\_info** |  | **field** |
| id |  | id |  | id |
| name |  | name |  | label |
| |  | | --- | | [] template\_info | |  | bass\_template |  | name |
| contact\_number |  | remarks |  | type(string,int, baselist) |
| contact\_person |  | |  | | --- | | []field | |  | created\_at |
| contact\_email |  | created\_at |  | updated\_at |
| created\_at |  | updated\_at |  | created\_by |
| updated\_at |  | created\_by |  |  |
| created\_by |  |  |  |  |
|  |  |  |  |  |
| **base\_list** |  |  |  |  |
| id |  |  |  |  |
| label |  |  |  |  |
| name |  |  |  |  |
| type(string,int) |  |  |  |  |
| version |  |  |  |  |
| created\_at |  |  |  |  |
| updated\_at |  |  |  |  |
| created\_by |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **template** |  | **field** |  |  |
| id |  | id |  |  |
| name |  | label |  |  |
| type (CDR, SS CTC) |  | name |  |  |
| version |  | type(string,int, baselist) |  |  |
| |  | | --- | | []field | |  | created\_at |  |  |
| version |  | updated\_at |  |  |
| created\_at |  | created\_by |  |  |
| updated\_at |  | created\_at |  |  |
| created\_by |  | updated\_at |  |  |
|  |  | created\_by |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **form** |  | **uploadfile\_info** |  |  |
| id |  | id |  |  |
| |  | | --- | | []uploadfile\_Info | |  | file\_name |  |  |
| created\_at |  | file\_name |  |  |
| updated\_at |  | created\_at |  |  |
| created\_by |  | updated\_at |  |  |
| template\_id |  | created\_by |  |  |
| []toi |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **return\_form** |  | **process\_log** |  |  |
| id |  | id |  |  |
| name |  | status |  |  |
| telco |  | fields[] |  |  |
| created\_at | |  | | --- | |  | | updated\_at |  |  |
| updated\_at |  | created\_by |  |  |
| created\_by |  |  |  |  |
| []toi |  |  |  |  |
| remarks |  |  |  |  |

The End