Moyasar Hiring: Simple Drive Project

Assign	
Status	In Progress

Simple Drive

You are required to write a Ruby on Rails application that provides APIs to store and retrieve objects/files using an id, name or a path.

This project will showcase your abilities and skills in Software Engineering principles and techniques like object orientation and abstraction and see your ability to explore and learn the underlying structure of different technologies.

Think of this project as developing a simple object storage system that provides a single interface for multiple storage backends.

The backends should be the following:

- Amazon S3 Compatible Storage Service
- Database Table
- Local File System
- FTP (Bonus Points)

Storing a Blob of Data

A user can store a blob of data using the service by making a `POST` request to the endpoint `/v1/blobs`. The endpoint should expect the following JSON object:

```
{
    "id": "any_valid_string_or_identifier",
    "data": "SGVsbG8gU2ltcGxlIFN0b3JhZ2UgV29ybGQh"
}
```

The first field `id` is a unique reference for the blob. It can be any valid text or string like a UUID, a path or a random set of characters, it should only serve as a unique identifier and should not have any meaning or semantics.

The second field `data` is a Base64 encoded binary data. If the service cannot decode the Base64 binary string upon receiving the request, the request should be rejected.

Retrieving a Blob

A user can retrieve a blob using the same `id` used when saving the data to the storage server.

Retrieving data should be available through the following endpoint `/v1/blobs/<id>`.

The response should look like the following:

```
{
    "id": "any_valid_string_or_identifier",
    "data": "SGVsbG8gU2ltcGxlIFN0b3JhZ2UgV29ybGQh",
    "size": "27",
    "created_at": "2023-01-22T21:37:55Z"
}
```

Beside the first two fields, the response should return the `size` in bytes and `created_at` timestamp in UTC.

Storage Backends

After we have described our interface to the outside world, we need to store the data somewhere. The service should be configurable to store the data in one of the storage services mentioned above.

We will describe each one of them next.

Amazon S3 Compatible Storage

Amazon S3 protocol is an open protocol for storing and retrieving data in logical containers called buckets. The protocol has some popular implementation like:

- AWS S3, the original.

- Minio, open-source and built in Golang.
- Digital Ocean Spaces.
- Linode Object Storage.

Amazon S3 protocol is built on top of HTTP. There are a lot of libraries from different vendors to connect and use S3 compatible servers.

Your challenge is to explore the S3 protocol and only use an HTTP client to store and retrieve files from the service.

You are not allowed to use any S3 library, and in doing so your project will be rejected.

Database

One of the storage options should be a separate database or database table that can be configured when using the system.

You can design the table as you see fit, but it should not be the same table used to track blob information within the storage service.

Local Storage

A system administrator can configure the service to store files within the local system instead of an external service or a database.

The only required configuration should be the path to the storage directory, e.g. `/home/john/server_storage`.

You are free to decide how to store the files within the directory and how to retrieve them.

FTP (Bonus Point)

As a bonus adapter, you can add the ability to store files on an FTP server. You can use any library you want to connect to an FTP server.

Tracking Blobs

Your application database should have at least a single table to track uploaded blobs and and their information.

Only metadata about the blob/file is stored in this table, the actual data should be stored in the storage service.

Authentication

Any request to the service should be authenticated using `Bearer` token authentication.

You are free to decide on the token type and its generation.

Remember to keep it simple.

Unit Testing (Bonus Points)

Writing integration and unit tests for the project will grant you extra points for this project.