A
PROJECT REPORT
ON

Cloud-based File Storage System

SUBMITTED BY

Vineet Maurya

ACADEMIC YEAR 2024-25

S.Y. B.C.A. SEM-3

UNDER THE GUIDANCE OF

Name of Faculty

Navrachna University

SUBMITTED TO

Navrachna University

A
PROJECT REPORT
ON

Cloud-based File Storage System

SUBMITTED BY

Vineet Maurya

ACADEMIC YEAR 2024-25

S.Y. B.C.A. SEM-3

UNDER THE GUIDANCE OF

Name of Faculty

Navrachna University

SUBMITTED TO

Navrachna University

ABSTRACT

ACKNOWLEDGEMENT

PROJECT PROFILE

Student Information	
Name	**Enrollment Number**
Vineet Maurya	23000068
Project Details	
Project Title	The File Spot - A Cloud based File Storage System
Duration	Insert Duration Here
Name of Project	Insert Project Name Here
Platform	Web
Team Size	1

INDEX

SDLC OVERVIEW

REQUIREMENT GATHERING AND ANALYSIS

Organization Details

Meetings

Data which will be Input into the System

Data which will be Output from the System

Type of Project

Method of collecting Data

SYSTEM REQUIREMENT SPECIFICATIONS

Introduction

The Files Spot (henceforth reffered to as TFS) is a simple file-storage service. It allows for simple and easy file backup, organization, and sharing.

3.1.1 Purpose

Many cloud storage solutions are currently available for general use for the public. These are usually either barebones storage services like Amazon S3 aimed at developers to build on top of, or customer-friendly services like Google Drive and Microsoft Onedrive. Unfortunately, services between these two extremes are few and far between. The Files Spot aims to fill this gap in the cloud storage space.

TFS is aimed squarely at power users who want a simple cloud storage service to store and backup their files in. Towards this end, TFS aims to deliver on two fronts - a simple user interface that makes it easy to get started, and a flexible API that allows for easy automation and integration with existing automation tools.

3.1.2 Document Conventions

The File Spot is henceforth reffered to as TFS in the rest fo this document.

Users in this document is used to refer to anyone who uses the service, either via the provided web client or via the API.

In this document, clients refer to both the web client and to any service that consumes the service via the provied API. This includes, but is not limited to, various automation services that may use the service by scraping/parsing the web client instead of using the API.

3.1.3 Intended Audience and Reading Suggestions

The intended audience for this documernt includes all stakeholders and any user who wishes to know more about the workings and design principles of the service. This should be especially useful for users who are wanting to know about the rationale behind certain decisions made during the development process.

3.1.4 Project Scope

The project explicitly aims to tackle the problem of uploading and downloading files. As such, support any and all file formats is within hthe scope of this project.

Security is provided via a basic login/token wall. The communication may be encrypted using standard HTTPS/TLS protocols, discussion about said protocols is out of the scope of this document.

File and data/metadata storage is a core part of this service and will be the main point of discussion in this document. Strategies to manage file access, data storage, and verification of file integrity will be an integral part of this project.

Encryption of files at rest is an explicit non-goal of this service. This is to avoid potential legal complexities regarding hosting of potentially illegal content. As such, all files uploaded to this service will be visible to the admins of the service. This is ordinarily meant to allow admins to comply with legal demands, but users should make sure that they don't upload any sensitive data to this service.

3.1.5 References

Overall Description

On the user side, The Files Spot (TFS) is a simple web application that can be accessed from any web browser. It will have a UI for uploading new files to the cloud and a UI to manage files that have already been uploaded to the cloud. The UI will be very simple and minimalistic to ensure that it can be easily parsed by tools and other accessiblity tools.

On the backend, it will be a simple file storage service written in PHP that will manage files for multiple users. It will also allow users to share files - both to specific users or to the public at large. File management will be primarily accomplished via tags set by the user.

3.2.1 Product Perspective

3.2.2 Product Features

The web client is designed to be easy to use for all kinds of people on all kinds of connections. Specific care is taken to ensure that the design is easy to understand and interpret.

The underlying DOM of the web client is designed to be minimal and clean to ensure that various accessiblity services can easily surface relevant information to the user. Although this is a non-goal, an attempt will be made on a best-effort basis to ensure that

the DOM itself remains relativey stable so that any scrapers that rely on it can remain functional as long as possible.

The API is meant to be very easy and intuitive to use. This is accomplished via the use of standard HTTP verbs (GET, POST, PUT, PATCH, DELETE) in the various file descriptors. The API will also be JSON-based and stateless (as far as possible) so that it is easy to inetgarte into existing CLI-based and app-based workflows.

3.2.3 Use Cases and Characteristics

The primary usecase for this service is to easily backup files to the cloud. In addition, the simple protocols make this a very accessible application.

Another use case for this service could be file transfer and sharing. It explicitly supports sharing with various people (or the world) to enable this usecase.

3.2.4 Operating Environment

This service can be run on any service that can run PHP.

People looking to deploy this service should first check what version of PHP their host supports. If a supported version is available, refer to the PHP deployment guides for that host.

Developers wanting to fork/make changes should refer to the comments in the source code. Additionally, we reccomend that they install just as a command runner to simplify their lives. This project makes heavy use of just recipies to automate mundate CLI tasks. For reference, this service was originally built and tested on PHP 8.3.10 running on Windows 10.

3.2.5 Design and Implementation Constratints

A major implementation constraint here is the need to make this service as user firendly as possible.

The web client aims to follow the principles of progressive enhancement. As a part of it, we ensure that the web app is useable in all three loading stages (HTML, HTML+CSS, HTML+CSS+Javascript). This therefore rules out JS-only frameworks like React, Vue, etc.

Another design decision to not use any large external libraries. This means that we mostly avoid large CSS libraries like Bootstrap. Instead, we opt for custom, minimal CSS and JS to power the site.

On the server side, the decision to avoid React, etc means that we need a language which can stringly augment the existing HTML instead of generating a new one everytime one is requested. This leaves us with two major, battle-tested options - Python (via Jinja2

templates in Django/Flask), or PHP. Here, we opt for PHP as it is a lighter, much simpler language to make a web server in. Additionally, PHP comes with a lot of server utilities built in (database methods, etc) which removes the need for external dependencies.

3.2.6 User Documentation

3.2.7 Assumption and Dependencies

System Features

The system storage will be backed by a standard disk on the server. Each file will then also be recorded in a Postgresql database along with some metadata. This metadata will allow the server to verify that the file is in the state it was uploaded in.

3.3.1 System Feature 1

External Interface Requirements

3.4.1 User Interfaces

3.4.2 Hardware Interfaces

3.4.3 Software Interfaces

3.4.4 Communication Interfaces

Other Non-Functional Requirements

3.5.1 Performance Requirements

3.5.2 Safety Requirements

3.5.3 Security Requirements

3.5.4 Software Quality Attributes

Other Requirements

SYSTEM ANALYSIS AND MODELLING

Use case Diagram

Normalization and E-R Diagram

Data Dictionary

Functional and Behavioural Modelling

Gantt Chart

TEST CASES

SCREENSHOTS

LIMITATIONS AND FUTURE ENHANCEMENTS

CONCLUSION

REFERENCES AND BIBILOGRAPHY