Backups Made Simple

Team Info

Members

- Adison Daggett - Front End Designer
- Kathryn Butler - Technical Documentation
- Luke Scovel - Developer
- William Brennan - Developer
- Anshu Avinash - Front End Designer
- Finlay Curtiss - Developer
- Jake Thompson - Developer

Repository

Github - https://github.com/Flameis/CS362-Team3

Communication Channel and Rules

We will use our Group 3 Teams Channel as our main communication channel. We will not have regular meetings besides the ones after class. If we need to have an impromptu meeting in the future we will communicate with each other on teams.

We will use Git as our version control due to our comfortability with the tool. We will make sure that our file structure is clean and well organized with proper .gitignore and README setup. We will be clear with our commit messages, and notify each other when we are working on code so our teammates know to pull before making more changes.

In order to promote creative discussions, we can encourage one another to do our best, and notify each other politely if some of the pushed code broke the whole website. We will consider each other's technical design decisions in an unbiased manner, promoting a space where we can both speak our minds openly.

If we get confused about what we are supposed to do, we will reach out to each other and discuss strategies to understand where we are in the development process. Additionally, we can independently reread the instructions/design document to refresh our understanding.

If a member gets sick or has other extenuating circumstances, they will notify the other team members and redistribute the workload accordingly. If a team member falls behind in their work, the other members will reach out and ask how they're doing and attempt to assist them if they're stuck on something challenging in the course.

Product Description

Abstract

The overcomplication of file backup systems has caused needless problems for the average consumer. Backups Made Simple (BMS) aims to provide a simple interface for creating automated secure backups of specified files. BMS allows users to define a target backup location, such as a server URL or LAN storage system, and automates the backup process without interfering with other applications. BMS focuses on delivering a hands-off approach to data preservation.

Goal

Modern backup features are too complicated. They're buggy forces of nature that you cannot stop without uninstalling your operating system completely. We want to change that. The best way to do so is by creating software with the opposite mentality that file backup systems use. We want to make backup software that only backs up and downloads what a user wants, without the bulky overhead.

Current Practice

Current cloud backup systems such as OneDrive are often unwieldy and can cause complications with the local files system itself, for example, OneDrive by default automatically stores the My Documents folder in the cloud and makes everything go through a shortcut to access the files. This can cause read-write errors for many programs that use the My Documents folder to store configuration files, save files, and more.

Novelty

File backup software is annoying. Companies like Microsoft design breathe down a user's neck and fearmonger the idea that one day, someone will hack into your computer and steal all your images. Nobody wants to deal with that. Our team wants to create a simple backup software that does not breathe down a user's neck. File backup software has been done before, but we want to give users the autonomy of nonintrusive backups.

Effects

By resolving the typical issues and constraints of current solutions, Backup Made Simple (BMS) will improve the way the users interact with backup software. The simple, clear-cut design will increase the effectiveness of file backups and remove the clutter associated with current backup systems.

Technical Approach

We will be using Python 3 while following the PEP 8 style guide. We will also use GitHub to assign and track tasks. A couple libraries that may be needed are a SFTP/File Manager package and a GUI package.

Risks

One of the largest risks will likely be related to connecting the program with some file storage location. Our initial plan is to use the Stak home directory that OSU offers for Engineering students. This might cause some delays in allowing our software to connect and run well with OSU's servers. One way to mitigate this is to design our program for use with the OSU Stak storage and test how our program interacts with it as soon as possible. If, either during our design or prototype stages, we see that this idea isn't working, we can pivot to another solution before we have sunk too much time into the first choice.

Major Features

- Upload: Allows users to easily upload the files or folders to a specific backup location.
- Download: Enables users to retrieve and restore backed-up files from their specified storage locations.
- Automatic backups: Implements an automated system that periodically backs up specific files or folders based on a user-defined preference schedule.
- User interface: Designs a simple, intuitive, and visually appealing interface for an elegant backup process.

Stretch Goals

- Encryption: Merges the encryption capabilities to secure files during both backup and download processes. Ensure that users with the correct decryption key can access backed-up files, intensifying data security.
- Compression: Adds file compression functionality to minimize storage space requirements. Enables users to back up large files or folders efficiently, reducing upload and download times.
- File conversion: Provides options for file conversion during backup (such as converting images to more space-efficient formats). Enhances flexibility by supporting a variety of file types and formats for specific use cases.