Cloudboard - A Cloud Based Clipboard

CSCI 5253 Datacenter Scale Computing

Team

Aditya Srivastava , Harsh Gupta , Aishwarya Jayaramu

Goal

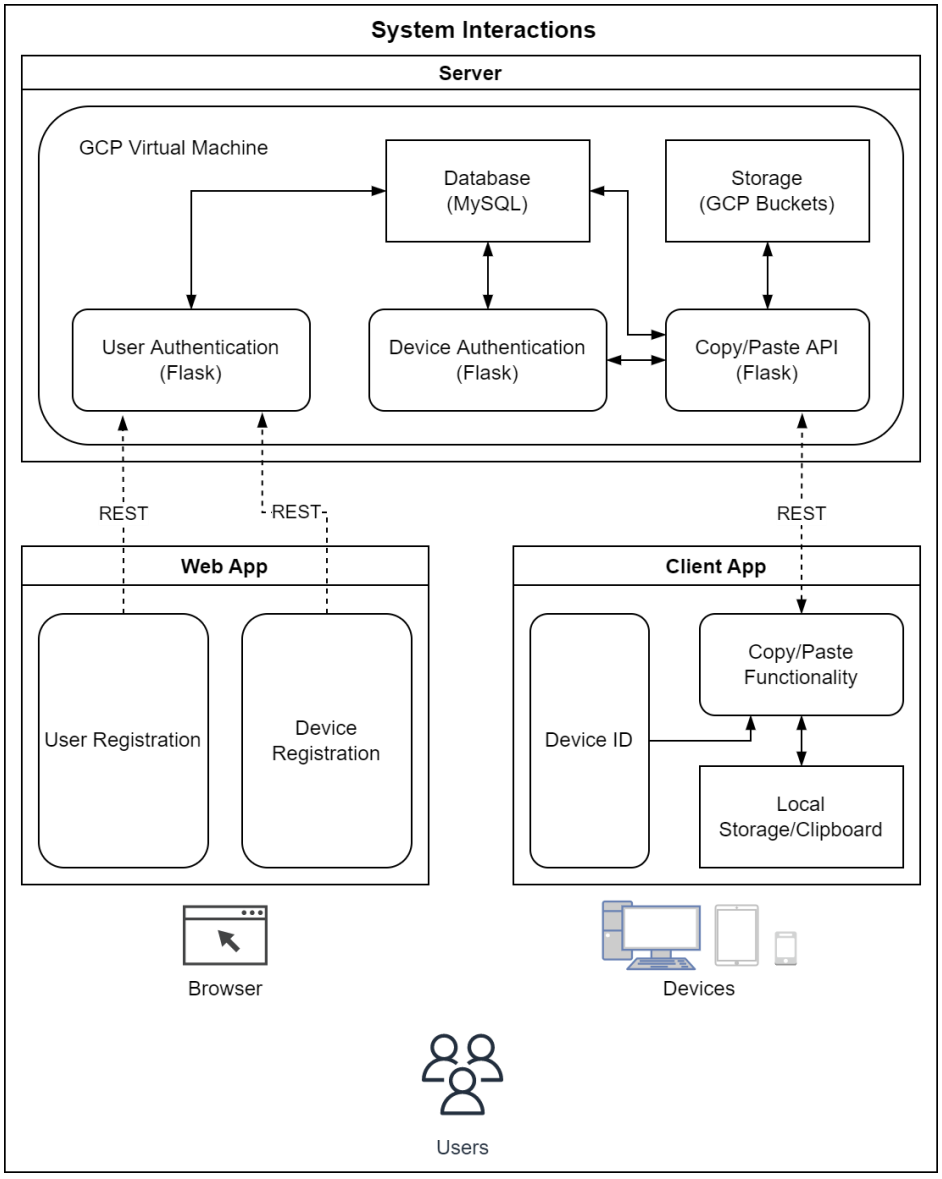
We aim to make a cloud-based clipboard service to which the user can seamlessly copy objects (text, images and files) from one of their devices and then paste the same objects on any of their other devices.

Motivation

Most people these days own more than one personal computing device, and often work on them simultaneously. Thus there are many instances where one might want to transfer data between two devices quickly and conveniently, such as when transferring files between two devices, copying URLs from your mobile device to open on your laptop's browser, and copying a verification or access code received on one device to use on another (such as when using 2-factor authentication). For most devices this continues to be a challenging task, especially considering the variability in per device operating systems and the lack of such a unified data transferring service. With Cloudboard, we are trying to solve this problem by developing a system which allows users to synchronize the clipboards on all of their devices.

Our solution employs a server on the cloud, running in a **virtual machine**, and clients applications on user devices, which use **API interfaces** to pass copy and paste requests between each other. We handle all of the user and device authentication, and their data using **storage services**, **databases** and **key-value stores** on the cloud server, and provide them with an unobtrusive solution that is easy and reliable to use.

System Architecture



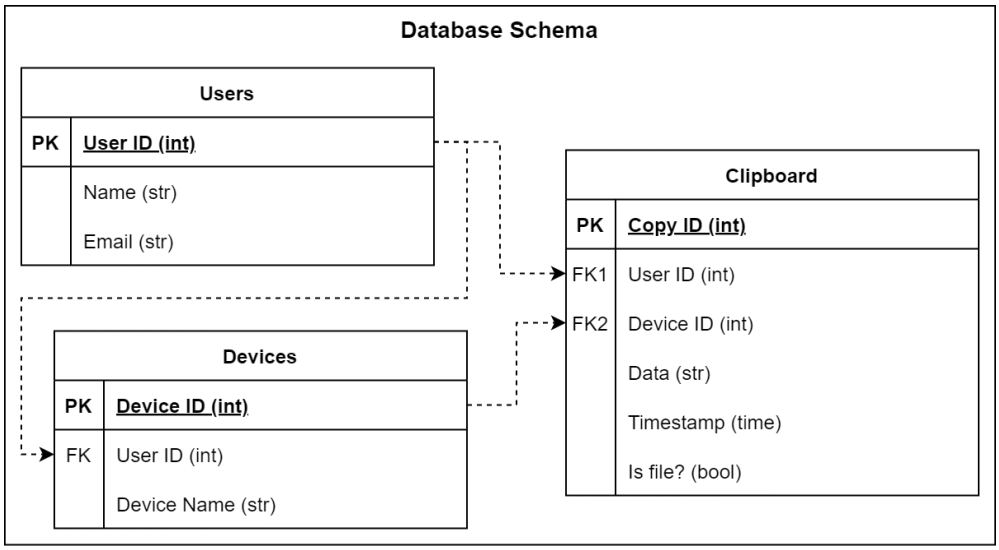
Overview of Key Components

As seen above, the service requires the following components;

1. Server that serves a REST API which
    a. Allows users to create accounts
    b. Allows users to register devices
    c. Listens for any copy and paste requests from the clients
2. Web portal that exposes the user/device registration API to the user

3. Client application installed locally on users' registered devices, that exposes the copy/paste API to them
4. A storage system for storing and indexing copied objects for retrieval later

Database Schema



The database schema has been designed with the following ideas in mind;

1. Each user can own multiple devices which are uniquely identifiable
2. Each copy operation in the *Clipboard* table has its own identifier and we store which device it originates from
3. In the *Clipboard* table, the *Data* field can be one of two possible strings:
    a. If the user copies text, the *Data* field holds the text itself
    b. If the user copies a file (or files), the *Data* field holds either the location of the file on the user's device or its location on the storage bucket in the server
4. The *Timestamp* reflects the time of the copy operation so that we can retrieve the most recent file
5. The *Is file?* field indicates whether the data field holds copied text or location to the copied file(s)

Software and Hardware Interactions

Software

The **server** is being written in Python using the Flask microframework, and will serve the REST APIs responsible for the following functionalities;

1. **User Registration:** Users can register an account using an email and password. This is required as each users' clipboard must be isolated from the others.

2. **Device Registration:** Users can register each of their devices to their account using token based authentication.
3. **Copy/Paste:** Users can send copy and paste requests to the server, authenticated by the token of the device that the request is sent from.

The server will also interface with the MySQL database and the storage bucket while handling the requests.

The **client application** will be installed locally on the user's devices and will run as a service in the background. When the user initiates a copy or paste operation, the client application will forward the requests to the server, using the copy/paste API. The application will not require users to login, and instead use the device authentication tokens generated during device registration to authenticate the requests.

The **web application** is a simple webpage for users to first register, and then later login to manage their devices, which includes the generation of authentication tokens for new devices.

Hardware

The **server** runs on the Google Cloud Platform, where our hardware comprises a single virtual instance, along with a bucket for storage.

The **client app** will run on personal computing devices, such as laptops, mobile phones and tablets. Since this will require us to write a new client application for each operating system, at the time of writing, we are only aiming to do so for Linux based devices.

The **web app** runs in-browser, and does not have hardware specific requirements.

Interactions

Each time a user performs a copy operation, the server receives a time-stamped copy request containing either the text copied, or the locations of the files. This request is authenticated with the device specific token.

Each time a user performs a paste operation, the server retrieves the most recent copy operation performed by the user. If the copied data has been cached in either the database or the bucket, it is forwarded to the device that sent the paste request. Otherwise, the server requests the device with the data to first upload the data, caches it, and then forwards it to the device making the paste request. If the copied data is text, then it is cached in the database itself, otherwise the files are cached in the storage bucket.

Testing and Debugging

We will start with testing the individual components;

1. **Functionality and Interface Testing**
    a. **Website**: Validate the REST API carrying the authentication credentials.
    b. **Client Application**: Test whether requests are sent correctly to the server and the output at the client for the paste command is displayed correctly. Errors if any must be caught by the application and must be shown to the end user and logged at the server.
    c. **Server**: Test if the server is handling all application requests without any service denial. Ensure that it receives the credentials and the copied data sent by the web app and the client app. Confirm that the data is sent when the client app requests a paste. Measure the latency and keep it within the usable limits. Ensure that the data cannot be overwritten by a copy request when a paste request is ongoing.
    d. **Database Server:** Make sure queries sent to the database give expected results.
2. **Database Testing:**
    a. We will test if any errors are shown while executing queries and ensure data integrity while creating or updating data on the database.
    b. We will check the response times of queries and confirm that the test data retrieved from the database is shown accurately in the client application.

In order to debug the APIs we will be using Postman to send and retrieve data from the server. We will also use the server logs to figure out what went wrong

Stretch Goals

Some stretch goals that we may work on are:

1. A way for users to clear their clipboard and/or view their copy history.
2. Data encryption on the copied objects for privacy.
