Cute Name

Smoc George-Marian **Lucaci Alexandru-Constantin**

Minea Ioan Mihai **Sbârcea Ștefan-Vladimir**

Contents

1 Introduction **2**

2 Case study **2**

3 Technical Details of Existing Solutions **3**

4 Overview of Technologies and Motivation **5**

4.1 Google Cloud Platform 5

4.2 Flask framework 7

4.3 Sveltte framework 8

4.4 SSH 9

5 Business Canvas **9**

6 System Architecture Overview **10**

7 Use Case Scenarios **12**

7.1 User Authentication 12

7.2 File Management 12

7.3 System Administration 13

References **14**

1 Introduction

This technical report presents a proposed project for developing a file storage website with a secure shell (SSH) frontend and a file viewing application. The purpose of this project is to provide users with a convenient platform to store and access their files in various formats. The report covers a case study, technical details of existing solutions, overview of technologies, business canvas, architectural diagram, use-case diagrams, and API documentation.

2 Case study

The proposed project aims to address the need for a user-friendly file storage solution with enhanced security features. In the current international/national reality, individuals and businesses often struggle with managing and accessing their files across different devices and formats. By developing a file storage website with an SSH frontend and a file viewing application, users will be able to securely store, organize, and view their files from anywhere, using different devices.

3 Technical Details of Existing Solutions

Text documents, spreadsheets, and presentations may all be created and worked on together using Google Docs. It lacks built-in assistance for displaying database files in a user-friendly format. Even while the marketing strategy for this solution emphasizes accessibility, which allows users to access and work on their files from any device with an internet connection, there were numerous instances where customers were unable to view files because their device did not support the format. A user-friendly graphical interface can be provided with little system resources by integrating a Linux terminal that makes use of the GView application (the graphic interface is made using the terminal's own capabilities). [Dra21]

- Client-side: Google Docs' client-side features a web browser-based user interface with an extensive document editing environment. Users can generate and update documents using the client-side tools, which also include an editor, formatting tools, collaborative features, and other resources. Desktops, laptops, tablets, and smartphones can all use the user interface since it is responsive and has been built to do so.

- Server-side: Google Docs' server-side is made up of a document processing engine, a database, and a file system, among other things. The collaborative features, document content, and formatting are handled by the document processing engine. The file system offers users a secure space to store and manage their files, while the database offers persistent storage for documents and other data.

- Cloud infrastructure: A cloud infrastructure serves as the platform's supporting infrastructure and is used to host Google Docs. The servers, storage, and networking components of the cloud infrastructure offer scalable and dependable performance. In order to safeguard user data and documents, the cloud infrastructure also offers security measures including encryption, firewalls, and intrusion detection.

- Google Drive, Google Sheets, and Google Slides are just a few of the Google services with which Google Docs is integrated. These integrations allow users to collaborate with other users, access their papers from any location with an

internet connection, and import and export documents from and to other Google services.

Repl.it is an online development tool that has a built-in terminal emulator and a variety of programming languages and tools. Its support is the most similar to what our application provides. What it lacks is the ability to integrate the software we use to see various file kinds, which forces users to learn more about using the command line to set up and maintain the software. These processes are automated by our platform.
Repl.it's limited computing resources won't be used to effectively view files because it is designed to be used for coding. When a large file is loaded and attempted to be visualized, things take longer than they would in a situation when the file is relatively small in size.

- Client-side: Repl.it's client-side features a user interface that is based on a web browser and offers a complete coding environment. Developers can write and test code on the client-side using an editor, console, debugger, and other tools.

- Server-side: Repl.it's server-side is made up of a code executor, a file system, and a database, among other things. The platform's code executor executes developer-written code and offers real-time feedback on output and failures. While the database offers persistent storage for projects, the file system gives developers a safe place to keep their code and data.

- Cloud infrastructure: Repl.it is hosted on a cloud infrastructure, which provides the underlying infrastructure for the platform. The cloud infrastructure includes servers, storage, and networking components that provide scalable and reliable performance. Additionally, the cloud infrastructure provides security features, such as encryption, firewalls, and intrusion detection, to protect user data and code.

- Integrations: Repl.it integrates with a range of third-party services and platforms, including GitHub, Bitbucket, and Google Drive. These integrations enable developers to import code from external sources, collaborate with other developers, and deploy code to various cloud services.

4 Overview of Technologies and Motivation

For the purposes of our project, we have many technologies to choose from, and our goal is to decide what would be the best possible options. The more we have the most appropriate technologies, the more everything will integrate smoothly with everything we set out to do. We will present in order of importance which technology we used, why we choose it and what other options we would have had.

4.1 Google Cloud Platform

One of the most important choices we have is which suite of cloud computing services we will choose. The answer to this is the Google Cloud Platform. Google Cloud Infrastructure is a global infrastructure that securely delivers incredible, sustainable performance and accessibility. Google's cloud infrastructure is trusted around the world and is designed to deliver the right services to users, wherever they are. Google has secure and efficient data centers, as well as a fast and reliable global network. Whether you need Google Compute Engine to launch virtual machines on demand, Google Container Engine for managing containers and clusters, or general Google Cloud Storage for storing and accessing data, Google Cloud Infrastructure has you covered.

Google Cloud Platform (GCP) is a suite of cloud computing services offered by Google. It provides a wide range of infrastructure and platform services that enable users to build, deploy, and scale applications and services on Google's global infrastructure.

The reasons we chose GCP are the very reasons that make GCP stand out from other services:

- The most important thing is that GCP is the most reliable of the options we wanted to choose. AWS and Azure, GCP's competing services, have, at least in the past, had quite a few problems over time. According to the following article Amazon Web Services(AWS) had outages in January 2022 that affected its customers. The article also mentioned that AWS believed clients should stick to multiple cloud regions within AWS, rather than entering multi-cloud agreements with other vendors to manage exposure to outages.

- Price, as far as we have looked, GCP offers the lowest price. Like its competitors,

Google Cloud Platform (GCP) offers a pay-as-you-go pricing structure where you only pay for the services you use. Google Cloud Platform (GCP) has been known for its competitive pricing and offers a range of pricing options, including on-demand, sustained use discounts, and committed use discounts As this article says, Google Cloud Functions have the lowest unit price for GBs consumed per second along with free-tier offerings. Also, being students GCP offers better "deals" for students compared to others.

- We simply like it better than Azure. In the cloud computing course we had to practice with both services and GCP was the most preferred by the whole team.

Google Cloud Platform was one of the 3 options we had at hand, the other 2 being AWS and Microsoft Azure. All 3 represent the top three cloud platforms with high cloud market share. AWS is the largest and most widely used cloud service provider, offering a comprehensive suite of services and a vast global infrastructure. It has a wide range of pricing options and a strong ecosystem of third-party integrations. Azure is a popular cloud platform that offers a broad set of services, including integration with Microsoft products and services. It provides a hybrid cloud environment and is often chosen by organizations already heavily invested in Microsoft technologies. All 3 have advantages and disadvantages, but for our project the most suitable was GCP.

The project is based on the following services:

- Google App Engine- is a fully managed serverless platform for developing and hosting web applications at scale. It is one of the services offered by Google Cloud Platform (GCP) that allows developers to build and deploy applications on Google's infrastructure.

- Google API Gateway- is a managed service used for Google Cloud services and external systems. It's the easiest way to create/ deploy and manage API backends

- Cloud CDN- It is used to deliver content to users from Google's global network of edge locations. This means that when a user requests content from your website, the content is delivered from the edge location closest to the user, which reduces latency and improves performance.

- Firebase Auth- Firebase Authentication is a service that provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to

your app. Firebase Authentication supports authentication using email and password, phone numbers, Google, Facebook, Twitter, and more

- Google Vision AI- Google Vision AI is a machine learning service offered by Google Cloud Platform that allows developers to train their own image recognition models or use pre-trained models for image analysis

4.2 Flask framework

For the server side, the choice of programming language was quite simple, I chose Python. Compared to other variants (nodeJS, GO, PHP) all project members know Python very well and it is about the simplest variant to create a server. Also, python provides a wide range of libraries and frameworks that make our lives easier, libraries and frameworks that have a very large community, where we can find almost any problem that arises. An important thing to mention, which is quite important for the realization of the project, is that Python is a cross-platform language, and that gives us the possibility to switch from one operating system to another without much trouble.

As a framework we stopped at Flask. Flask is often considered a popular choice due to its simplicity, flexibility, and ease of use. Flask is a micro-framework, which means it provides the core functionalities needed for web development without imposing rigid conventions. This flexibility allows developers to have more control over the structure and design of their applications. Flask's minimalist design keeps the learning curve low and makes it easy to get started. Additionally, Flask has an extensive ecosystem of extensions that add functionality when needed, without bloating the core framework.

We chose Flask because it represented the simplest variant to understand and use, it is considered to have a slight learning curve, and is often considered easier to learn due to its simplicity and minimalistic approach. It has a gentle learning curve, making it accessible to beginners. A problem we found with Python and implicitly with its frameworks is that there are many, each variant for a specific type of problem. So choosing one was not that simple. We needed to know exactly what we wanted to do and how we wanted to do it. We had to choose from the following frameworks:

- Flask framework- Flask has emerged as a popular and lightweight Python web framework. Known for its simplicity and minimalism, Flask offers a range of

benefits that make it a preferred choice for developers seeking an efficient and customizable server-side solution. This essay delves into the key aspects of Flask, highlighting its simplicity, flexibility, Pythonic syntax, active community, and versatility.

- Fast API- FastAPI is a modern Python web framework that has gained popularity for its high performance and efficient handling of asynchronous operations. While FastAPI has its strengths, here are a few reasons why it might not necessarily be considered better than Flask in situations like complexity, Ecosystem and Community and Compatibility and Integration.

- Django framework- is a robust and feature-rich Python web framework that follows a batteries-included approach. It provides a comprehensive set of built-in tools and functionalities for the rapid development of complex web applications. However, its extensive nature and conventions may be considered overkill for small or simple projects where a lightweight and minimalistic approach is desired.

- Pyramid framework-Pyramid is a flexible and scalable Python web framework that is known for modularity and extensibility. It offers a wide range of features and options, making it suitable for large-scale applications. However, its learning curve may be steeper compared to Flask, and it may not be the best choice for developers who prefer a simpler and more lightweight framework.

- web2py - is a full-stack Python web framework that focuses on simplicity and ease of development. It provides an all-in-one solution with an integrated database abstraction layer, form handling, and security features. However, its comprehensive nature and conventions are a deal breaker for us.

4.3 Sveltte framework

After deciding that we will use a framework for the server side, we also chose to use a framework for the front end. After several searches, we found the Svelte framework. Svelte is a tool for building fast web applications. It is similar to JavaScript frameworks such as React and Vue, which share the goal of making it easy to build slick interactive user interfaces. But there's a crucial difference: Svelte converts your app into ideal JavaScript at build time, rather than interpreting your application code

at run time like traditional frameworks do. This means you don't pay the performance cost of the framework's abstractions, and you don't incur a penalty when your app first loads. Taking into account that the vast majority of the team are beginners when it comes to a front-end framework, the simplest and easiest-to-use variant is Svelte, especially since our project is not very complex on the front-end side. According to this article, Svelte is an ideal option for what we need. Svelte also has a simpler syntax than other frameworks, which makes it easier to learn and use. It uses a template language that is similar to HTML, which makes it easy to read and write. Svelte has been gaining popularity in recent years, and many developers are choosing it over other frameworks because of its simplicity and speed. However, it is still a relatively new framework, so it may not have as many resources or community support as some of the more established frameworks.

4.4 SSH

SSH stands for Secure Shell. It is a cryptographic network protocol that allows secure communication between two computers over an insecure network. SSH is used for remote login and other network services such as file transfer, email, and X11 tunneling. SSH provides a secure channel over an unsecured network by using a client-server architecture and providing strong encryption of data during transmission. As this article says SSH is widely used in the front-end development community for secure remote access to servers and other network services.

5 Business Canvas

A business canvas is a strategic management tool that helps you to visualize and develop your business model. It consists of nine building blocks that describe the key elements of your business, including your value proposition, customer segments, channels, customer relationships, revenue streams, key resources, key activities, key partnerships, and cost structure.

- Value proposition: Secure file storage and organization with easy access from anywhere. Also, we aim to seamless access across devices, emphasizing the convenience and flexibility it brings to users

- Customer segments: Individuals and businesses who need to store and access files securely. When we say about individuals we mean, first of all, people who are not in the field, who just want the application to work, even if it would be perfectly usable and useful for people who are in the field, but also students/pupils who want to experiment, to solve assignments/projects and not consume a lot of resources when it comes to storage.
- Channels: Online advertising and social media marketing. One idea would be the creation of bundles for students, the creation of free trials.
- Customer Relationships: We want to keep communication as open as possible with all our customers, so we try to have all forms of communication available, so we offer Self-service with email support, and a virtual chat with common problems and ways to solve the problems that appear.
- Revenue streams: Subscription plans paid monthly/yearly, rewards for people who use the application for a long period of time, personalized plans for students/people.
- Key resources: includes the necessary hardware and software infrastructure, development tools, secure servers, updated frameworks, skilled personnel, and partnerships with Google cloud service providers.
- Key activities: Developing and maintaining the website and applications.
- Key partnerships: None at this time.
- Cost structure: includes development and maintenance costs, server hosting and bandwidth expenses, marketing and advertising expenditures, customer support resources, and licensing and legal requirements.

6 System Architecture Overview

In this section, we provide an overview of the system architecture for our file storage website. The architecture is represented using the C4 model, which offers a structured approach to visualize the system's key components and their relationships at different levels of abstraction.

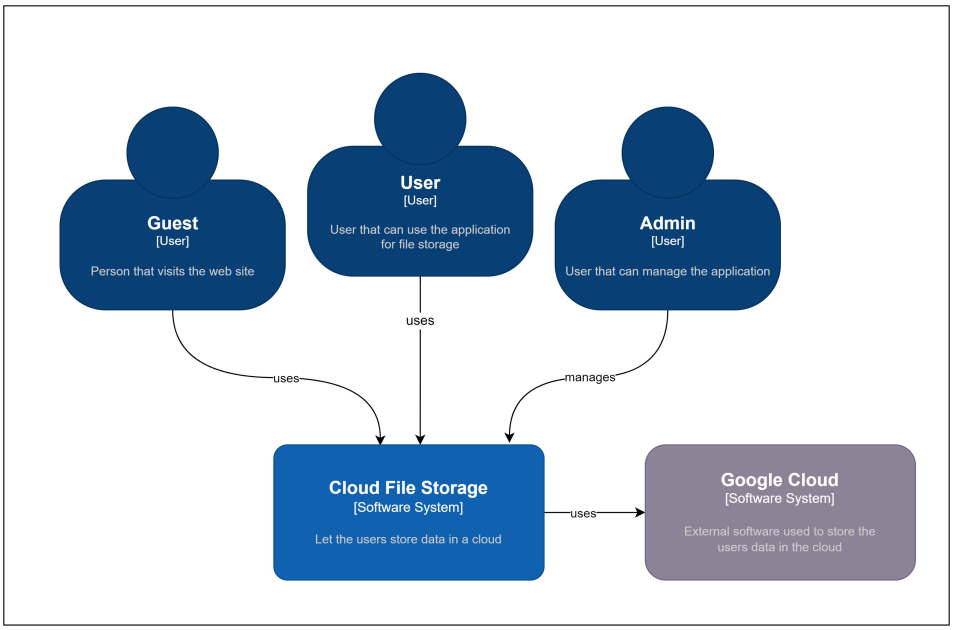

Figure 1: System context Diagram

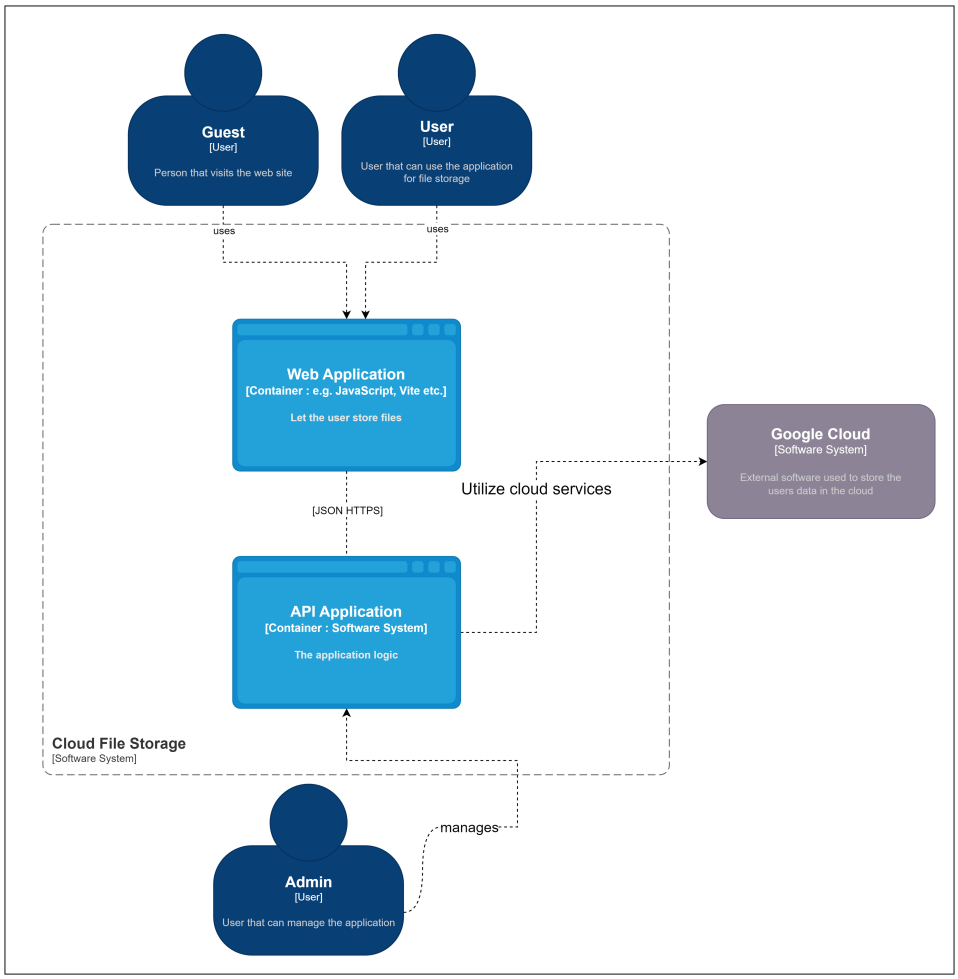

Figure 2: Container Diagram

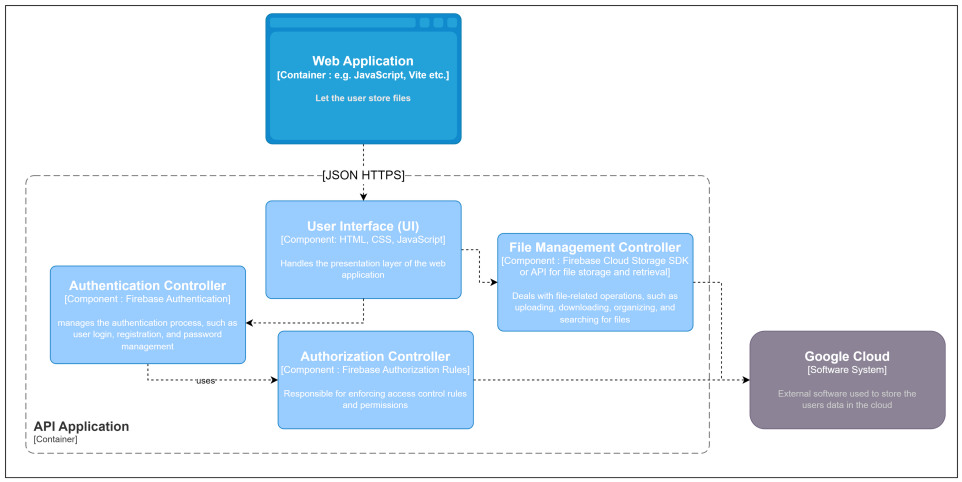

Figure 3: Component Diagram

7 Use Case Scenarios

7.1 User Authentication

Actors: User, Authentication System

Use Cases:

- User Registration
- User Login
- Forgot Password
- Change Password

Relationships: User initiates the authentication processes, such as registering a new account, logging in, resetting a forgotten password, or changing the password.

7.2 File Management

Actors: User, File Management System

Use Cases:

- Upload File
- Download File
- Delete File

- Create Folder
- Move File
- Share File

Relationships: User interacts with the file management system to perform operations like uploading files, downloading files, deleting files, creating folders, moving files between folders, and sharing files with other users.

7.3 System Administration

Actors: Administrator, System Administration System

Use Cases:

- Manage User Accounts
- Change System Settings
- Generate Reports
- Remove User

Relationships: The administrator interacts with the system administration system to manage user accounts, changes system settings, generate reports on system usage or user activities and removes suspicious users.

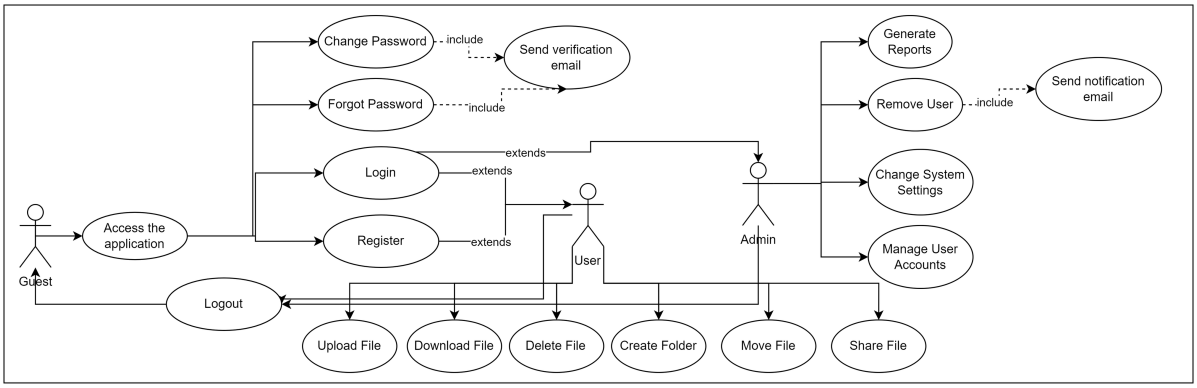

Figure 4: Use case diagram

References

[Dra21] Gavrilut Dragos. Gview project. https://github.com/gdt050579/GView.git, 2021. Accessed on May 14, 2023.