LaurentianUniversity
UniversitéLaurentienne

Project Report

Secure File Sharing System over Google Cloud Platform

Under the supervision of:

Prof. Jaspreet Bhatia

CPSC-5207

LAURENTIAN UNIVERSITY

BHARTI SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

Group - 5:

Amandeep

Rattan Nagi

Noman Rashid

Kanwal Preet

Md Jubayer

Table of Contents

1. Abstract

The Secure File Sharing System over Google Cloud Platform is a web application that enables secure and efficient file sharing among users. It is designed to meet the unique needs of startups and small businesses. By utilizing Firebase's suite of tools and Google Cloud Platform's robust infrastructure, the system provides a scalable and secure environment for file storage and management. The application includes user authentication via Firebase Authentication, file storage using Cloud Storage, and deployment through Firebase Hosting. Developed using React.js, the web application's frontend is both responsive and user-friendly. This project demonstrates the practicality of integrating cloud-based services to create secure and scalable web solutions, addressing common challenges in file sharing, such as data protection, ease of access, and performance.

2. Introduction

2.1 Background

File-sharing systems have become an important part of both personal and professional environments, driven by the need for remote collaboration and data exchange. However, traditional methods often face issues like data breaches, limited storage, and poor scalability. The introduction of cloud platforms like Google Cloud has drastically changed this era by offering reliable, secure, and globally accessible solutions.

2.2 Business Case

Startups and small businesses often face challenges in securely sharing and collaborating on documents across their teams. Traditional solutions either lack advanced security features or are extremely expensive for smaller organizations. As businesses increasingly rely on remote work and digital tools, the risks associated with unauthorized access and data breaches become significant, especially when handling sensitive customer data.

The project addresses some challenges by offering a secure, scalable, and affordable solution tailored for small businesses and startups. Key aspects of the business case include:

1. **Controlled File Sharing**: The application allows users to share files selectively, ensuring only authorized individuals can access them.
2. **Permission Management**: Comprehensive access control mechanisms ensure that sensitive customer data remains protected, reducing the likelihood of security breaches.
3. **Real-Time Information**: The system provides features such as activity logs and audit trails, offering businesses real-time insights into file usage and access. These logs help in monitoring unauthorized activities and maintaining regulatory compliance.
4. **Hybrid Deployment**: For businesses with stringent data security requirements, the app supports hybrid deployment. This allows sensitive files to be stored on private infrastructure while leveraging cloud services for less sensitive data and application hosting.

2.3 Objectives

1. **Secure File Upload and Sharing:** Allow users to upload files and generate secure, dynamic links for authorized access only.
2. **Authentication and Access Control:** Use Firebase-based authentication to verify users and role-based access control for sensitive data protection.
3. **Cost-Effective Solution:** Utilize GCP's free-tier and pay-as-you-go pricing model to develop an affordable platform.
4. **Scalability**: Ensure the system can handle increasing user demands without compromising performance or security.
5. **Global Accessibility**: Enable users to access and share files seamlessly from anywhere with just a simple internet connection.

2.4 Scope

The system provides a secure interface for file uploading and generates unique, dynamic links for controlled file access. These links are accessible only to authorized users, demonstrating the emphasis on user authentication and data security. Core functionalities are implemented using GCP services, including Firebase Authentication for secure user management, Compute Engine for pre-processing, Cloud Storage for file storage, and Firebase for hosting the web application.

Key interactions include:

1. Uploading a file and securely storing it in Firebase Storage.
2. Authenticating users and controlling their roles through Firebase Authentication.
3. Pre-processing the files for security like scanning it for virus (eg. malware) through a back-end logic in Compute Engine.

The operations and features are conducted entirely in a simulated environment designed for demonstration purposes, ensuring no use of real user data or integration with external services.

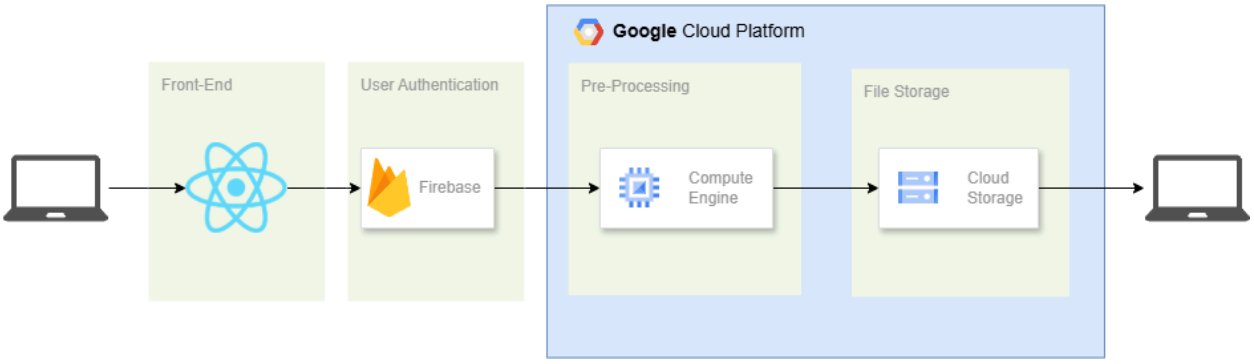
While the implementation is limited to a small-scale demonstration, this analysis provides insights into extending the system for larger organizations or more complex use cases.

2.5 Significance

The project showcases the use of Firebase and Google Cloud services to build a modern, efficient, and scalable application. The secure handling of files ensures data protection, while the integration of cloud tools eliminates the need for a complex server-side infrastructure.

3. Model

3.1 System Architecture



The architecture comprises three main layers:

1. **Frontend Layer**:
    a. Built with React.js, this layer provides users with an intuitive and responsive interface for interaction.
    b. It includes pages for user authentication, file upload/download, and file management.
2. **Middleware Layer**:
    a. Firebase SDK acts as a bridge between the frontend and the backend, enabling seamless communication.
3. **Backend Layer**:
    a. Firebase Authentication handles user sign-ups and sign-ins.
    b. Compute Engine pre-processes the files for viruses.
    c. Cloud Storage manages file uploads, metadata, deletion, and retrievals.
    d. Firebase Hosting ensures the application is globally available with SSL encryption.

3.2 Tools and Technologies used

1. **Frontend Development:**
    a. React.js:
        i. React.js was chosen for building the user interface due to its component-based architecture, which enables reusable, modular code.

ii. React Hooks were used to manage state efficiently, ensuring a seamless user experience.
iii. Integrated libraries like Material-UI provided pre-designed, responsive components, reducing development time while maintaining a professional appearance.
iv. It enabled navigation between pages (e.g., login, dashboard, file upload) without refreshing the browser, improving the application's responsiveness.

2. **Firebase Services:**
    a. Firebase Authentication:
        i. Provides a secure and easy-to-implement authentication mechanism.
        ii. Supports multiple sign-in methods, such as email/password and Google Sign-In, ensuring flexibility for users.
        iii. Session persistence ensures users remain logged in across page reloads, enhancing usability.
    b. Storage:
        i. Used for storing uploaded files securely.
        ii. Supports resumable uploads, ensuring all files can be uploaded even with intermittent network connectivity.
        iii. Enforces access control via Firebase Security Rules, restricting file access based on user authentication.
    c. Firebase Hosting:
        i. Used to deploy and host the web application.
        ii. Provides a globally distributed content delivery network (CDN), ensuring fast load times for users worldwide.
        iii. HTTPS is enabled by default, ensuring all communication between the client and server is encrypted.

3. **Backend Logic:**
    a. Google Compute Engine (GCE):
        i. A virtual machine (VM) instance is used to implement backend logic for preprocessing files uploaded by users.
        ii. This includes a file scanning feature to detect potential viruses or malware before allowing further actions, ensuring the system remains secure.
        iii. The VM runs a script integrated with antivirus software to scan files in real-time.
        iv. Processed files flagged as "safe" shall be uploaded, otherwise not.

4. **Development and Deployment Tools:**

a. Node.js and npm:
    i. Node.js was used as the runtime for managing development workflows.
    ii. npm handled project dependencies, including Firebase SDKs and React-specific libraries.
b. Firebase CLI:
    i. Facilitates seamless deployment of the application to Firebase Hosting.
    ii. Enables testing of Firebase rules locally before deploying them to production.
c. GitHub:
    i. GitHub provided a centralized repository for storing and sharing project code.

3.3 Key Features

1. **User Authentication:**
    a. Registration and login functionality using Firebase Authentication.
    b. Google Sign-In integration for quick and seamless user access.

2. **File Management:**
    a. Real-time notification to track upload/download.
    b. Metadata tagging for files, enabling quick retrieval and categorization.

3. **Security Measures:**
    a. Role-based access control to ensure users only access their files.
    b. Implementation of Firebase Storage rules for protecting sensitive data.
    c. Secure token generation to prevent unauthorized API access.

4. **User Experience:**
    a. Mobile-first design to ensure compatibility across devices.
    b. Interactive dashboards displaying file statistics (e.g., total uploads, file sizes).
    c. Notifications and alerts for upload/download completion.

5. **Deployment:**
    a. Firebase Hosting for deploying the application globally with minimal configuration.
    b. Configured HTTPS to encrypt all communications.

4. Implementation

4.1 Workflow

1. **Frontend Development:**
    a. The frontend was developed using React.js, emphasizing a modular and maintainable structure. Reusable React components were designed to ensure scalability and easy updates.
    b. Material-UI components were integrated to provide a responsive and an aesthetic layout.
    c. React Router was implemented for seamless navigation between pages such as login, dashboard, and file upload, enhancing the user experience.

2. **Google Cloud Configuration:**
    a. A Google Cloud project was set up as the backbone of the application's backend services.
    Key configurations included:
        a. Google Cloud Storage:
            i. Configured to store user-uploaded files securely.
            ii. Access control rules were implemented to restrict unauthorized access, ensuring compliance with security best practices.
        b. Google Compute Engine (GCE):
            i. Deployed a virtual machine (VM) instance to handle backend file processing.
            ii. This VM runs an antivirus script to preprocess files uploaded to Cloud Storage, scanning them for viruses or malware.
            iii. Processed files flagged as "safe" shall be uploaded, otherwise not.

3. **Integration:**
    a. The application integrates Firebase and Google Cloud services to deliver a seamless user experience.

4. **Authentication and File Management:**
    a. The Firebase SDK was used to incorporate user authentication features, allowing secure login and role-based access control.
    b. APIs were developed to manage file uploads, virus scans, and secure file sharing.

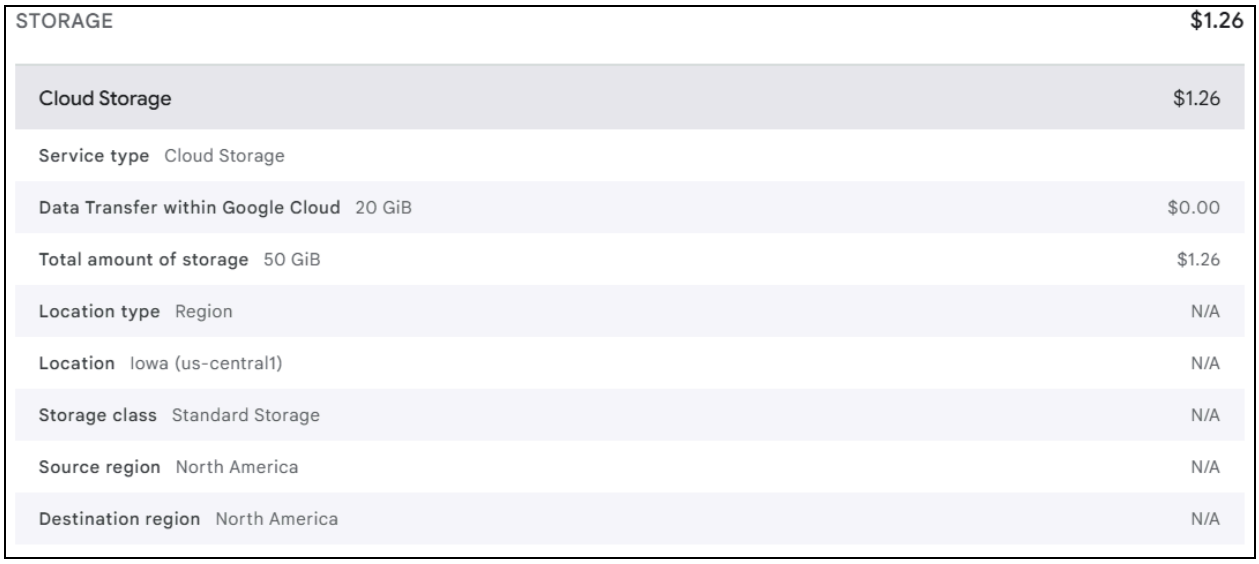
5. **File Preprocessing Workflow:**
    a. Google Compute Engine retrieves the uploaded files for scanning and preprocessing, ensuring they are virus-free before generating secure download links.

6. **Testing and Debugging:**
    a. Functional testing was performed to validate critical application features, including user authentication, file upload, file scanning, and download functionality. Bugs related to session persistence, file scanning workflows, and UI responsiveness were identified and resolved.
    b. Thorough debugging ensured that the interaction between Google Cloud Storage, Compute Engine, and Firebase services operated seamlessly.
7. **Deployment:**
    a. The application was prepared for production by creating a build using the npm run build command.
    b. Frontend Deployment:
        i. The web application was deployed using Firebase Hosting, leveraging its global content delivery network for efficient and secure hosting.
    c. Backend Deployment:
        i. The Compute Engine instance and Google Cloud Storage were configured to handle backend logic and file storage in real-time.
        ii. Security configurations ensured that backend services operated only under authorized access.

4.2 Cost Estimation

Cost estimation for this project involves analyzing the resources and services used on Google Cloud Platform (GCP) and Firebase, as well as other associated operational and infrastructure expenses. The following are the key areas would be:

1. **Google Cloud Services:**
    a. Cloud Storage (ref. Fig 2):
        i. The total storage shall be *50 GB*.
        ii. The region would be *Iowa (us-central1)*.
        iii. The storage class would be *Standard Storage*.
        iv. The initial data transfer within Cloud would be up to *20 GB*.



STORAGE	$1.26
Cloud Storage	**$1.26**
Service type Cloud Storage	
Data Transfer within Google Cloud 20 GiB	$0.00
Total amount of storage 50 GiB	$1.26
Location type Region	N/A
Location Iowa (us-central1)	N/A
Storage class Standard Storage	N/A
Source region North America	N/A
Destination region North America	N/A

Fig. 2

b. Compute Engine(ref. Fig 3):
   i. The machine type would be *e2-medium* (vCPUs: 1, RAM: 4 GB).
   ii. The instance shall run for *2 hours/day*.
   iii. The Boot Disk size shall be *10 GB*.
   iv. The region is the same as Cloud Storage (*Iowa (us-central1)*).

COMPUTE	$4.27
Instances (Compute Engine)	**$4.27**
Service type Instances	
Instance-time 2 Hours	N/A
Machine type e2-medium, vCPUs: 1, RAM: 4 GB	$2.86
Boot disk type Balanced persistent disk	N/A
Boot disk size (GiB) 10 GiB	$1.40
Number of Instances 0.1	N/A
Operating System / Software Free: Debian, CentOS, CoreOS, Ubuntu or BYOL (Bring Your Own License)	N/A
Provisioning Model Regular	N/A
Threads per core 2 threads per core	N/A
Enable Confidential VM service false	N/A
Add GPUs false	N/A
Region Iowa (us-central1)	N/A

Fig. 3

2. **Firebase Services:**
    a. Authentication:
        i. It is free for up to *50k users/month*, then GCP pricing takes place accordingly.
    b. Hosting:
        i. It is free up to *10GB/month*, then *$0.10/GB*.
    c. Data Transfer:
        i. It is free up to *360MB/day*, then *$0.15/GB*.

The total estimated cost is to be referred to in Fig. 4.

Fig. 4

5. Results and Analysis

5.1 Functional Testing

The application underwent rigorous functional testing to ensure its core features worked as intended. The key results were as follows:

1. The login and logout functionality was verified using both valid and invalid credentials, ensuring secure access.
2. File upload limits were tested, including restrictions on file size, to prevent oversized files from disrupting the system.
3. Access control was validated by ensuring that only authenticated users could view and manage their respective files.

5.2 Performance Metrics

The application demonstrated strong performance during testing, with the following metrics:

1. File uploads averaged between 3 to 5 seconds for files up to 5 MB, reflecting efficient backend integration.
2. User authentication processes were completed in less than 2 seconds, ensuring a smooth login experience.
3. Hosting latency was minimal, with global page load times averaging under 500 milliseconds, providing a fast and responsive user interface.

5.3 User Feedback

A test group of users provided valuable feedback after using the application:

1. The interface was praised for its ease of use and intuitive design, making it accessible to users with varying levels of technical expertise.
2. Users highlighted the system's security and reliability, especially regarding controlled access and file sharing.
3. Suggestions for future improvements included the addition of shared folder functionality and enhanced collaboration features.

6. Future Scope

The project demonstrates the effective use of core Google Cloud Platform (GCP) services to develop a scalable, secure, and user-friendly file-sharing system. Building on this architecture, several enhancements could be implemented in the future to expand the system's capabilities:

1. **Advanced Notifications:**
    a. Introduce real-time notifications to alert users when their shared files are accessed or downloaded.
    b. Provide customizable notification preferences, allowing users to choose between email, SMS, or in-app alerts.

2. **Regional Scalability:**
    a. Extend the system to support deployment across multiple regions, ensuring lower latency and improved performance for a globally distributed user base.
    b. Implement region-specific storage to comply with data residency and regulatory requirements.

3. **Support for Additional File Formats:**
    a. Expand compatibility to include additional file types, such as semi-structured (eg. *HTML, XML, JSON*) and advanced document formats.
    b. Provide tailored features, such as video previews, audio playback, or rendering for CAD files, enhancing usability for various industries.

4. **Hybrid Deployment Options:**
    a. Enable hybrid deployment to allow sensitive files to be stored on-premises while using GCP for scalable application hosting and file sharing.
    b. Offer seamless integration between on-premises and cloud environments for businesses with robust security and compliance needs.

5. **Enhanced Collaboration Features:**
    a. Include functionality for shared folders and group-level permissions, facilitating easier collaboration across teams.
    b. Develop tools for real-time editing and annotation of shared documents, similar to collaborative platforms like Google Docs.

7. Conclusion

The *Secure File Sharing System over Google Cloud Platform* successfully addresses the need for a secure, scalable, and user-friendly file-sharing solution. By integrating Firebase and React.js, the project highlights the potential of modern web technologies in solving real-world challenges. This project not only met its objectives but also laid the foundation for future enhancements, such as collaborative features and advanced file management tools.

For a live demo of the project, visit https://cloud-project-demo-da9e3.web.app.

Github Repository: https://github.com/amandeep0826/Secure-File-Sharing-System-using-GCP

References

B. S. Rawal and S. S. Vivek, "Secure Cloud Storage and File Sharing," 2017 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 2017, pp. 78-83, doi: 10.1109/SmartCloud.2017.19

P. Pandiaraja, D. P, K. J and L. S, "An Efficient Data Sharing System with Enhanced Authentication Factors and Secure Access Privilege Delegation," 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024, pp. 392-399, doi: 10.1109/IDCIoT59759.2024.10467256.

Adi Wijaya; António Vilares, Data Engineering with Google Cloud Platform: A guide to leveling up as a data engineer by building a scalable data platform with Google Cloud , Packt Publishing, 2024.

A. Al-Said Ahmad and P. Andras, "Scalability analysis comparisons of cloud-based software services," J. Cloud Comput., vol. 8, no. 1, p. 10, Jul. 2019, doi: 10.1186/s13677-019-0134-y.

Kovuru, H. K. (2024). Secured File Sharing on the Cloud using Encryption Techniques [California State University, Sacramento]. https://scholars.csus.edu/esploro/outputs/graduate/Secured-File-Sharing-on-the-Cloud/99258164563601671#file-0

G. Griffis, "How to scale in the cloud," Work Life by Atlassian. Accessed: Dec. 09, 2023.[Online]
https://www.atlassian.com/blog/platform/how-to-scale-in-the-cloud

https://cloud.google.com/storage/docs/introduction

https://github.com/johnmarksinclair/secure-cloud-storage

https://github.com/forscht/ddrive

https://github.com/EKarton/RClone-Drive-WebUI

Google Compute Engine Pricing:
https://cloud.google.com/compute/pricing

Google Firebase Pricing:
https://firebase.google.com/pricing

Firebase Authentication Overview:
https://firebase.google.com/docs/auth

Google Compute Engine Overview:
https://cloud.google.com/compute

Compute Engine Documentation:
https://cloud.google.com/compute/docs

GCP pricing calculator:
https://cloud.google.com/products/calculator