Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 3, July 2021:2566- 2579
Research Article

Cloud based Project Management System

Blessin George Varghese [1], T.R. Rohith [2], B. Amutha [3]

Abstract

A fully cloud based cross platform codebase management application that makes ensures secure and reliable access to all developers to a centralised codebase that's maintained by a company / group.

Keywords: *Cloud, IDE, Development, UI, Flutter, Codebase, Management, Storage, Live Feed, Access, Control, Version Control, Code Integrity.*

[1] Computer Science Engineering, SRM IST, Kattankulathur, India, bv4837@srmist.edu.in

[2] Computer Science Engineering, SRM IST, Kattankulathur, India, tt1878@srmist.edu.in

[3] Dr., HoD, Department of CSE, SRM IST, Kattankulathur, India, amutha.b@ktr.srmuniv.ac.in

Introduction

A Cloud based, cross platform application that facilitates easy collaboration and sharing of projects and files. This application focuses primarily on:

1. Role Based User Authentication.
2. Code Integrity Verification.
3. Sharing of projects via shortened links.
4. Version Control.
5. Feed that updates changes in an organisation / group.

As work from home and other forms of remote jobs are becoming increasingly popular, this platform enables quick access to work files from anywhere on any device at any time, securely.

The platform empowers developers to work from anywhere at any time, especially during a Pandemic like the current Covid-19.

State of the Art (Literature Survey)

A fully GUI based application so that the developers focus more on their implementation rather than learning to use our system.

Buttons, text fields, and other components will be used for a fully touch/pointer interactive UI.

A live feed will make all collaborators aware of the dev-happenings in their organisation.

1. To avoid mismanagement/unauthorized access of codebases.
2. The provide ubiquitous access for faster development.
3. To make everyone in the team aware of codebase updates.
4. To improve the quality of code.

This platform can be used in companies, start-ups and even for small group projects. This solution fits well with Rapid Application Development strategies.

Proposed Work

A cloud based accessible interface that shows the various files being edited in real time just like Google Docs but for Code. [9]

Ubiquitous access is ensured by the deployment of a cross platform client application, which would work on any modern internet enabled device. A comprehensive log is maintained to get metrics or to generate reports on who accessed/added a file and when. [10]

Uses a load balancer to mitigate additional cost and unavailability of the service. Also, a firewall is used to mitigate any possible DDoS attacks.

Traditional load balancers are very costly and inflexible hardware. A substitute of this hardware is to use software-defined network load balancers. The software-defined networking (SDN) load balancers offer the facility to programmers to design and construct their own strategy of load balancing, which makes the software-defined network load balancer flexible and programmable[2].

The present counterparts offer a more complex way of code management, which leads to more time being spent on learning the system, which inherently blocks quick start of development.

Compute Engine provides auto scaling, is something that allows us to create or remote Virtual Machines for a controlled group community based on changes in usage. Automatic Scaling allows the services to manage spikes in activity without any impact while still lowering costs when resource use is limited. The auto scaler conducts automatic redesign of the number of Virtual Machines based on the calculated required usage after a specific usage policy is defined.

Regulation for Auto-scaling

At least one autoscale signal must be defined when creating an autoscale. CPU usage, load balanced usage power, cloud-specific data, and other policy designs can be used to generate signals. If you have multiple signals in your autoscaling policy, the autoscale ups or downs a tagged virtual machine based on the token that most virtual machines allow.

At least one autoscale signal must be defined when creating an autoscale. CPU usage, load balanced usage power, cloud-specific data, and other policy designs can be used to generate signals. If we have multiple signals in your autoscaling policy, the autoscaler ups or downs a tagged virtual machine based on the token that most virtual machines allow.

The basic and fundamental way to automatically scale is CPU utilization. This policy instructs the autoscaler to monitor the average CPU usage of a specific or specifically tagged virtual machine and to connect or remove machines as necessary to maintain configured utilization. This is particularly useful for CPU intensive system designs that can vary in usage. Serving power load balancing.

When you configure an automatic system to create new systems as per our load utilization policy and ability, our automatic scaling system monitors an instance group's serving capacity and increases the number of Virtual Machines when existing ones reach their maximum capacity.

The load balancing backend system will define an instance's serving power, which would be enabled on usage or the various request that are happening in any given second.

Timetables

Algorithmic based automatic scale allows us to increase the access and up time of our service and solution by provisioning space prior to the expected demand. For repeated usage pattern along with one of high spike event, you may set a minimum instance group size.

Compute Engine File Storage

Disk storage gives applications file specific access for reading and updating data that would be spread amongst many devices. Scale-up architecture is used by some on-premises file storage systems, which easily add storage to a defined scale of computing resources.

Apart from this we have more file design systems which use a single machine starting and then pan it out design, which allows space and CPU cores (for speed) to be attached to an already present file design incrementally as required. A single or more virtual machines (VMs) would be able to access the files in both storage architectures.

Multiple file design systems for storing utilize a standard that allows user machines to attach a File system and utilize or view the files as though they were connected to their own system, despite the fact that certain file systems use a native POSIX client. Network File System (NFS) for Linux (and in some cases Windows) and Server Message Block (SMB) for Windows are the most general protocols for exporting file shares.

Local SSDs and Persistent Discs on Compute Engine

Local SSDs offer millions of IOPS and GB/s of bandwidth without completely filling the deployment network bandwidth. It's particularly prudent for us to consider, that the on premise Solid State Drives have availability, reliability, and flexibility trade-offs.

When it comes to choosing a file storage solution, there are a few things to keep in mind.

When selecting a file storage solution, we must consider manageability, expense, efficiency, and scalability. If there is a well scheduled load on the system, which won’t be necessarily always be the situation, making the decision is easier. When workloads change based on a schedule or is extremely variable, it would be a good idea for swapping running costs for freedom and explicit change to be able to scale the entire service. If you have a temporary and well-known workload, on the other hand, you would be able to design an extremely specific built file storage design solution that can potentially be quickly torn down and restored to accommodate the storage requirements.

One of the foremost choices you'll have to perform is which option to choose, paying for an included storage systems, or a system with platform specific support, or to go with a completely self-managed solution.

The next step is to figure out our specific reliability and availability needs are. Most system designs are regional and will not have cover if a region fails by default. As a result, it's critical to think about whether a disaster recovery (DR) design that prevents and helps us recover in case of regional failures is necessary. It's particularly crucial for us to comprehend the system’s maximum load and up time specifications.

The choice of a local solid state drive or permanent disks in the deployment, for example, would have a significant effect, as would file solution software policy management. To enable our application to be highly durable, available, and even secure against multiple specific geo-location failures, each solution requires meticulous planning.

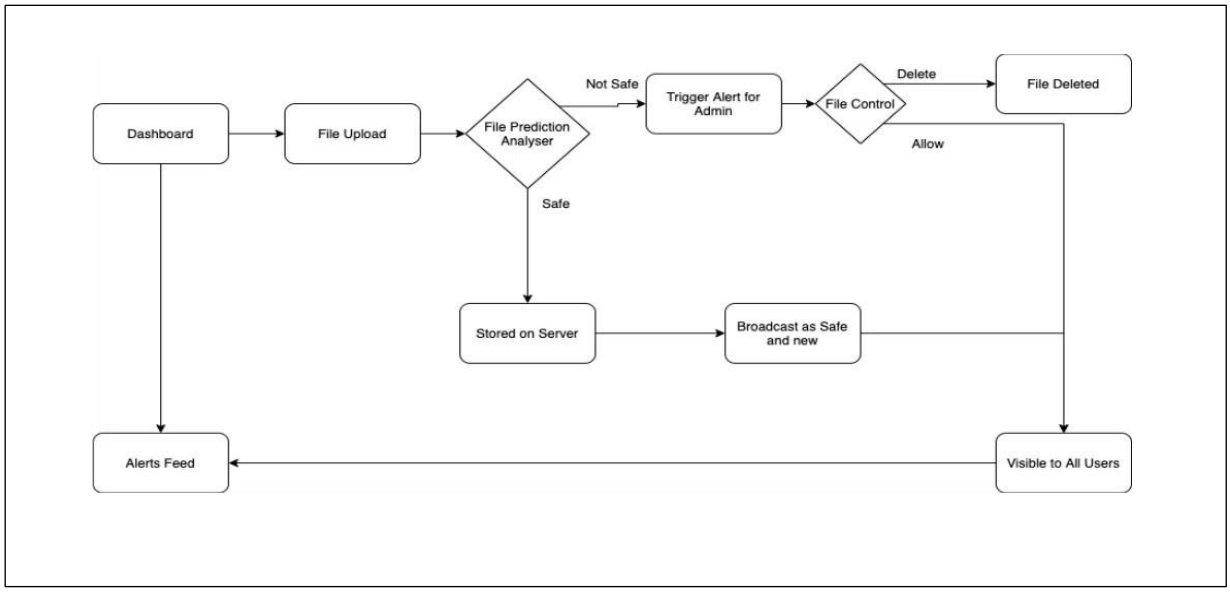

Fig. 1. The General Flow Diagram of the Application

Lastly, think about where you will need to access your data (zones, countries and local data centers). Since only certain archiver solutions would allow hybrid, on-premises, and cloud access, the location of the various data centers accessing the data will influence your choice of solution to use.

We have modules in our system such as:

1. Role Based User Authentication

A centralised authentication module, that gives restricted access to user based on privilege levels as:

a. Admin
    1. Has authorization at the highest level. Can add / remove users, view and delete any file, block a user, turn off the platform and other administrative accesses.
b. Moderator
    1. Maintains a team. Can add / remove users, view and delete any file, block a user and others at a group level.
c. Developer
    1. Has create / read / update access to team's files and delete access to own files.

These will allow the organization to ensure only those who should have access or are allowed to can do so. [11]

2. User Environment

An interactive place to write code for the user. This will serve as the input and output for all parts of our application. It runs in the cloud as such can be loaded on any browser. Thereby requiring no installation. All bug fixes can quickly be rolled out as the user need not update instead merely refresh the page. [8]

A place where all the recent updates from a member's organisation are updated with short texts which keeps him/her updated with the happenings around codebases.

Examples:

a) New Member in the team
b) Update to package
c) Bug Found

These will allow the developer to get timely updates without having to scramble and search for them. They will get them as and when they require it. Thus, enabling higher productivity and focus.

3. Dashboard

The Single Page View where all of the metrics and graphs along with logs are visible to the user.

4. Backup

The file data is kept in backup with the latest version for lifetime and previous versions for up to 30 days (can be set by the organizational policy).

5. Automatic file reporting

We've implemented an ML based file analysis so as to automatically detect and report potentially harmful files. [1]

Algorithm for training the model for this works as follows:

- Start
- Load Dataset
- Using ExtraTrees Classifier, find out important features. Example of features:
    - Characteristics
    - DllCharacteristics
    - VersionInformationSize
    - ImageBase
    - Checksum
- Build an ML model using:
    - Decision Tree
    - Random Forest
    - Ada Boost
    - Linear Regression
- Train each model, and pick the model with best accuracy
- Save the model
- Check :
    - False positives
    - False negatives
- Stop

6. Cloud Security

As our platform process and store files on the cloud, it is very important to ensure the integrity of them. We've taken the following measures to ensure security:

i. Discover and evaluate cloud applications

Most of us tend to take IaaS (Infrastructure as a Service) or PaaS (Platform as a Service) security for granted and don't hesitate to add a new application or platform to the corporate

cloud environment.. However, each new application added can present a potential risk and should be evaluated accordingly.

ii. Manage access to cloud applications and user behaviour

As with many cloud apps and storage options, there is usually more than one user who regularly needs to access the apps. To ensure sensitive data is protected, configure user access permissions and manage access to limit access to information within the core group. Apply a strong password policy that requires a minimum of 14 characters containing at least one uppercase letter, lowercase letter, special character, and number. Also limit the number of unsuccessful attempts to connect to the cloud.

iii. Identify, categorize and protect sensitive data stored in the cloud

Identify sensitive data: Identify the data or application to which you want to manage access. Sensitive data, such as customer data, organizational policies, and other information such as keys, encrypted passwords, etc., that need to be protected should ideally be in a separate folder or storage, with limited access. Restrict downloading of sensitive data to risky or insecure devices [5]

Despite the strictest access controls, data loss often occurs due to downloading files to devices. When sharing data or information externally, be sure to create security policies to block and protect downloads to unknown devices and monitor low-trust sessions as much as possible.

iv. Automate and remediate cloud application security risks

Information security is essential for all organizations, large or small, but these functions are often understaffed and underfunded. The use of tools and automation can help the application security team stay current without being overwhelmed by high-risk situations. [6]

Cloud automation helps improve the security and resiliency of applications within an organization, because when sensitive tasks are automated, you don't need to rely on manual resource tracking and connecting IT people to critical systems. [7]

v. Protection against malware threats

Protecting against malware threats becomes increasingly difficult as attackers use advanced components to pose serious threats to the cloud infrastructure. [4]

We’ve added an ML based malware detection layer that automatically identifies potential threat and generates a report based on it. [3]

7. Cloud Scalability

A web application is mainly meant to be used by multiple users at the same time from multiple locations, with varying workloads. Scalability is the factor that decides the

robustness of the platform in this scenario. To ensure standard scalability, we've covered the following:

i. Seamless scalability from data centre to cloud
ii. Elastic scaling on systems on both vertical and horizontal scales, using Kubernetes clusters : Horizontal Pod Auto scaling and Vertical Node Auto scaling

8. Uptime

Uptime refers to the availability metric of a platform. If the platform is always up, running and accessible, it is said to have a good uptime or in other words, zero downtime.

Implementation

Frontend : Flutter
Backend : Python / JavaScript
Database : MySQL
Cloud : GCP

Usage of Flutter enables quick prototyping, flexible cross platform support, easy updates and much more. Dart language with improved null safety measures enables us to build it without null safety issues.

JavaScript has been used for the backend to enable seamless asynchronous and near real time communication with servers. Python is primarily used for analysis.

This platform uses a serverless backend architecture and thus improvises on cost efficiency and load balancing. [2]

A content delivery network (CDN) is designed to have multiple servers and networks in multiple regions. This allows users to connect to their nearest one reducing latency. This also helps in reducing load as multiple networks and servers are being accessed. It also enables a provider to scale up relatively easily. As latency is determined by distance, the need or requirement to have it in multiple regions is mandatorily required.

This huge network that is present allows for extremely quick transfer of static content that does not change. Things like images, videos, and design assets along with style information

that directly impacts the user experience is served by this network. These networks are extremely popular with very large companies maintaining dedicated teams for the same. All this is required to ensure a smooth experience for the users of our application.

Screenshots

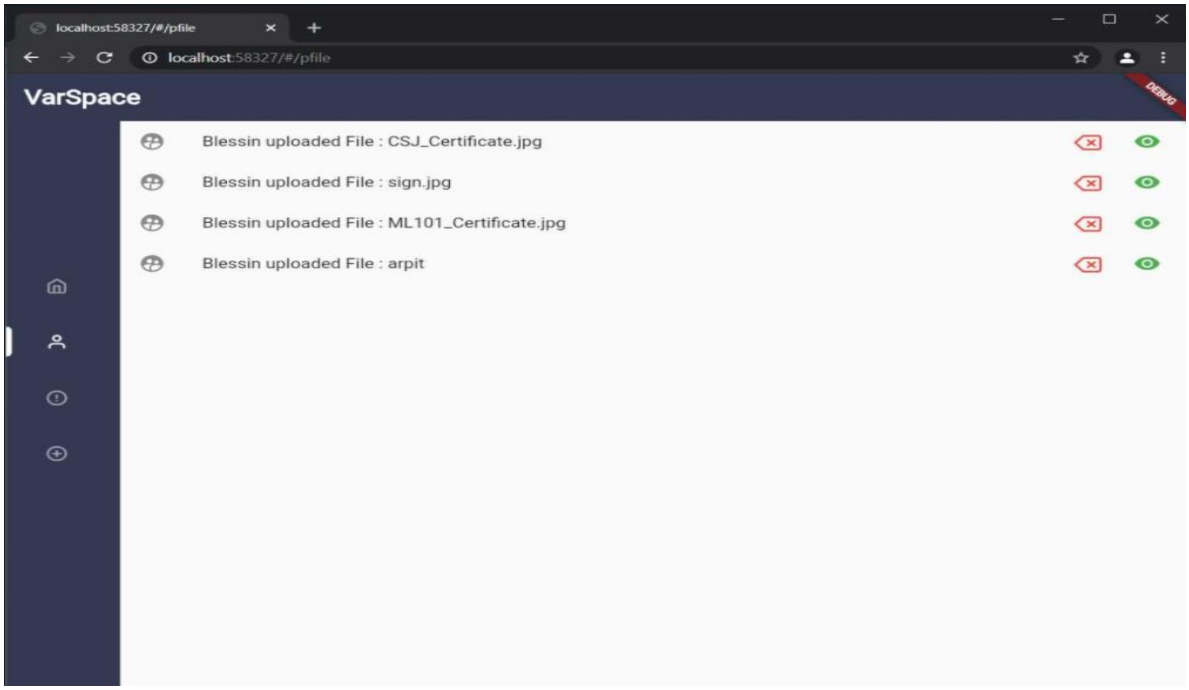

Fig. 2. My Files Section: Shows Personal Files

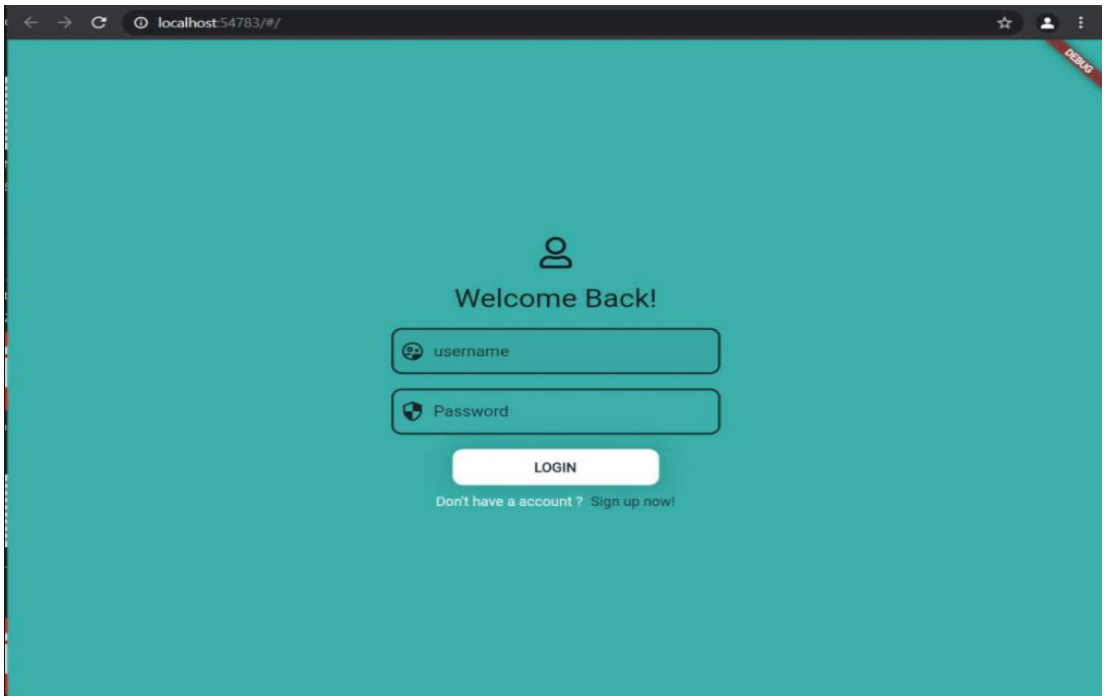

Fig. 3. The Login Screen

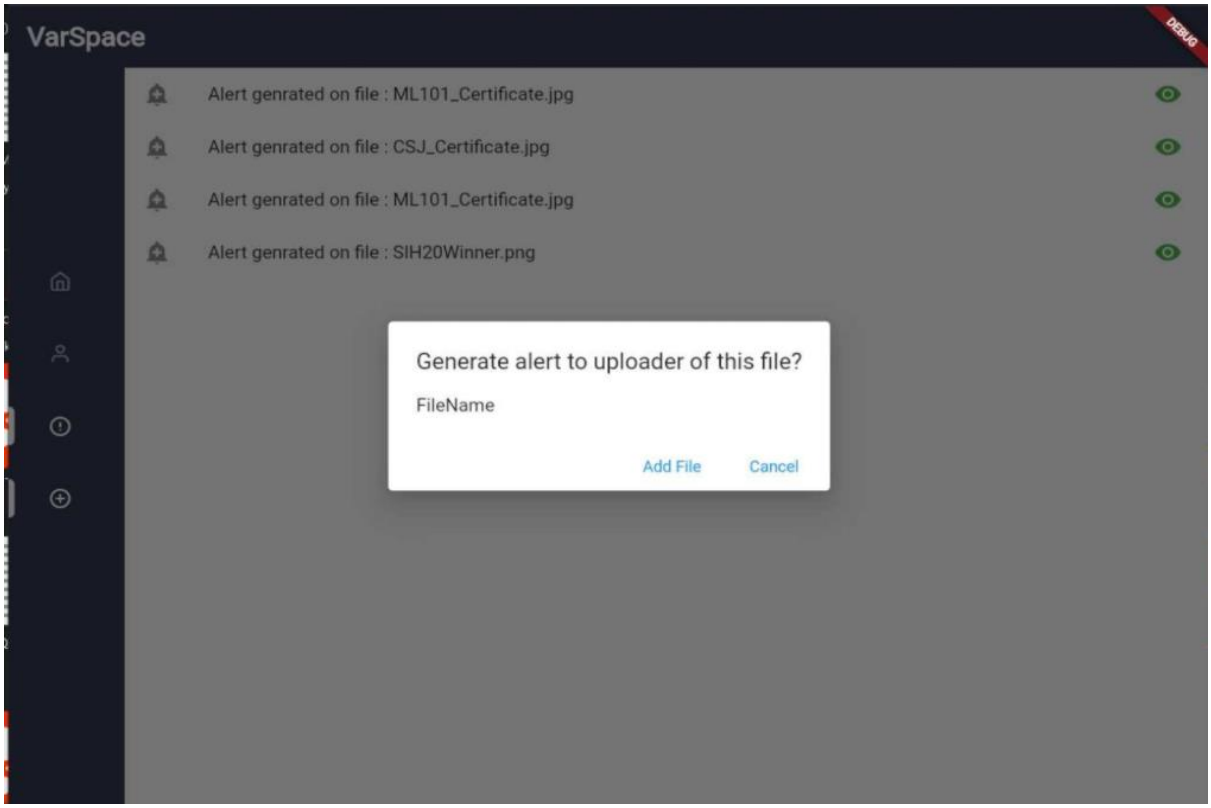

Fig. 4. Generate Alert Against a File

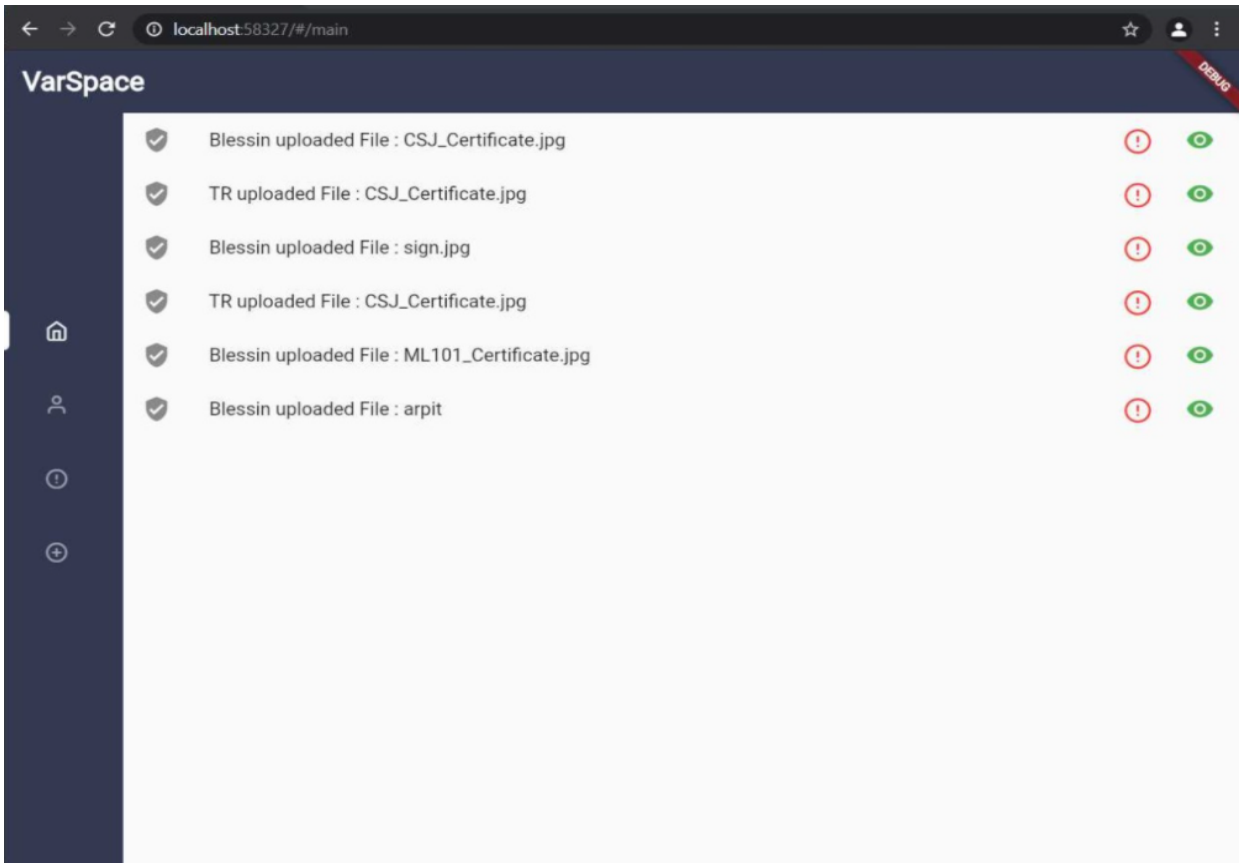

Fig. 5. All Files Section : Shows Files from All Users

Results Discussion

The platform has been able to perform up to the expectations put forward in the beginning, with All implemented modules working seamlessly. Implementation of the same in production would definitely take more resources and testing but is definitely going to help people manage the way they share work files.

Also, preventing DDoS and other forms of attack is taken care of, by using firewall to stop malicious requests. A backup for the past 30 days at any point in time is preserved, to assist in disaster recovery.
On the security front, JWTs are used to block and restrict any unauthorized API calls / access to the database.

When comparing with existing platforms of the same kind, our platforms offer a relatively easier on boarding and has an easier learning curve.

Conclusion

We have a fully cloud based cross platform codebase management application that makes ensures secure and reliable access to all developers to a centralized codebase that's maintained by a company / group.

References

Binkley, D. (2007). Source code analysis: A road map. *In Future of Software Engineering (FOSE'07)*, 104-119. https://doi.org/10.1109/FOSE.2007.27.

Chakravarthy, V.D., & Amutha, B. (2019). A novel software-defined networking approach for load balancing in data center networks. *International Journal of Communication Systems*, e4213.

Zhang, R., Huang, S., Qi, Z., & Guan, H. (2011). Combining static and dynamic analysis to discover software vulnerabilities. *In Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 175-181.

https://doi.org/10.1109/IMIS.2011.59.

Barstad, V., Goodwin, M., & Gjøsæter, T. (2014). Predicting source code quality with static analysis and machine learning. *In Norsk IKT-konferanse for forskning og utdanning*.

Shivaji, S., Whitehead, E.J., Akella, R., & Kim, S. (2012). Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, *39*(4), 552-569.

Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., & Bener, A. (2010). Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, *17*(4), 375-407.

https://doi.org/10.1007/s10515-010-0069-5.

Seyster, J., Dixit, K., Huang, X., Grosu, R., Havelund, K., Smolka, S.A., & Zadok, E. (2010). Aspect-oriented instrumentation with GCC. *In International Conference on Runtime Verification, Springer, Berlin, Heidelberg*, 405-420.

Dalipaj, D., Gonzalez-Barahona, J.M., & Izquierdo-Cortazar, D. (2016). Software Engineering Artifact in Software Development Process–Linkage Between Issues and Code Review Processes. *In New Trends in Software Methodologies, Tools and Techniques*, 115-122. https://doi.org/10.3233/978-1-61499-674-3-115.

Maata, R.L.R., Cordova, R., Sudramurthy, B., & Halibas, A. (2017). Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment. *In IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 1-4.

https://doi.org/10.1109/ICCIC.2017.8524573.

Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., & Baldi, P. (2007). Mining internet-scale software repositories. *Advances in neural information processing systems*, *20*, 929-936.

Ferraiolo, David & Kuhn, D. (2009). Role-Based Access Controls.