JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Assemble - A Decentralized Modular Platform Connecting Institutes Together

**[1]MURTAZA UDAIPURWALA, [2]BHAVYA GADA, [3]SARIKA MANE**
[1]Student, [2]Student, [3]Professor
[1]Department of Artificial Intelligence & Data Science,
[1]K J Somaiya Institute of Technology, Mumbai, India

***Abstract:*** The primary motive of the Assemble project appears to be the creation of a collaborative, efficient, and innovative ecosystem for educational institutions by leveraging a decentralized network and community-driven development. It is focused on addressing several key objectives: Facilitating Collaboration, Enhancing Educational Environments, Promoting Decentralization, Encouraging Innovation, and Promoting Decentralization. Assemble is a platform with a community-driven plugin architecture, encompassing all your needs. The core of the platform is powered by a decentralized network. It is a platform built to create a cohesive environment for institutes and at the same time allow them to connect with other institutes over the Assemble Decentralized Network (ADN).

*Index Terms* - **Golang, Kubernetes**

## I. INTRODUCTION

The world has shifted into the digital realm, making many activities that once required physical presence now accessible with a simple click. The COVID-19 pandemic led educational institutions to transition their daily operations to digital platforms, giving rise to online lectures, digital notes, and online exams. Today, numerous institutions rely heavily on digital platforms, such as web applications and smartphone apps. Assemble takes the concept of a digital platform to the next level by adopting a modular, plugin-based architecture, all powered by a decentralized network

## II. OBJECTIVE

Key takeaways from this introduction to Assemble include:

- Community-Driven Collaboration: Assemble thrives on community contributions, with plugin architecture designed to encourage innovation and customization. This collaborative spirit fosters a dynamic and adaptable ecosystem.
- Decentralization for Security: The project's core infrastructure leverages decentralization to enhance security and privacy. This approach is pivotal in safeguarding sensitive educational data and ensuring the resilience of the platform.
- Efficiency and Performance: The choice of Go (Golang) for backend development underscores the commitment to efficiency and scalability, critical for handling diverse needs and a potentially large user base.
- User-Friendly Frontend: SvelteJS is chosen for the frontend, prioritizing a streamlined and high-performance user interface. This simplifies development, reduces load times, and enhances the overall user experience.
- Educational Advancements: Assemble's primary motive is to empower educational institutes by providing tools that improve administrative processes, enhance teaching and learning experiences, and enable seamless collaboration with peers.
- Adaptability: While the initial concepts are outlined, the project remains open to evolution, reflecting a commitment to staying responsive to the evolving needs and challenges within the education sector.
- In conclusion, Assemble emerges as a promising initiative poised to reshape the landscape of educational collaboration, driven by community innovation and underpinned by security, efficiency, and user-friendliness. While the project's journey is still unfolding, its vision and principles provide a compelling foundation for the future of educational institutions worldwide

## III. LITERATURE SURVEY

- gRPC simplifies inter-process communication by efficiently exchanging data and function calls, making it suitable for micro services and distributed systems. [1]
- The OpenAPI specification standardizes services through machine-readable API definitions, enabling seamless integration, documentation, and code generation for consistency and collaboration. [2]
- Svelte.js compiles components into efficient JavaScript, resulting in faster load times and a smoother user experience, making it a compelling choice for modern web development. [3]
- JWTs streamline web app authentication and authorization by providing tokens with user credentials and permissions for each request, enhancing security and performance.[4]
- OIDC simplifies social media authentication by allowing users to log in using their social accounts, boosting user convenience and ensuring data privacy compliance. [5]

- A concurrency model manages plugin communication by overseeing concurrent execution and resource sharing, ensuring efficiency and control in a modular system.[6]
- Multi-threading architecture enables programs to perform multiple tasks concurrently using independent threads, improving performance, especially on multicore processors.[7]
- Concurrent maps in Go ensure safe and efficient data handling across go routines, preventing race conditions and maintaining data integrity. [8]
- AWS ELB distributes incoming traffic to multiple targets, such as EC2 instances, ensuring application availability and performance in AWS. [9]
- Kubernetes efficiently manages containerized applications, providing scalability, load balancing, and automation for project consistency and reliability. [10]
- Mobile devices are essential for accessing and interacting with cloud-native services, offering flexibility and security on smartphones and tablets. [11]
- Optimizing a cloud-native environment involves continuous resource monitoring and auto-scaling for efficient operations and cost savings. [12]
- TLS, or Transport Layer Security, safeguards data during transit, ensuring secure and private communication over networks like the internet. [13]
- Multi-tenant services allow multiple users or organizations to share the same infrastructure or application while keeping their data and operations isolated. [14]
- Creating a decentralized network involves designing a distributed system without central authority, promoting resilience and autonomy. [15]
- Advanced memory technologies like non-volatile memory (NVM) provide faster access and persistent storage capabilities, enhancing overall system performance and reliability. [16]
- Leader election algorithms, such as Raft or Paxos, are used in distributed systems to choose a single node responsible for coordination and decision-making among peers. [17]
- Zero-trust security verifies every user and device in a network, regardless of location, before granting access to resources, enhancing network security. [18]
- Cloud computing offers benefits like scalability, cost-efficiency, and global accessibility, enabling rapid application deployment and scaling while reducing infrastructure management overhead. [19]

## IV. METHODOLOGY

Assemble aims to assist institutions in creating their own customized platforms tailored to their specific needs. Recognizing that each institution may have unique requirements, we introduce the concept of plugins/extensions available for download and installation from the Assemble marketplace. This process resembles setting up a WordPress site or installing a VSCode plugin, making it accessible with just a few clicks. We expand this concept further by establishing a decentralized network connecting all participating institutions, known as the Assemble Decentralized Network (ADN). This decentralized network enables institutions to exchange data and resources while maintaining control over their own assemble instances due to its decentralized nature. ADN is built upon the ActivityPub protocol, enhancing Assemble's ability to connect with other ActivityPub projects such as Mastodon and Pleroma.

### *A. RPC-Based Plugins:*

Assemble's plugins communicate with the parent process through Remote Procedure Calls (RPC). Since these plugins are developed by the community and are not subject to regulation by the Assemble Team, we have designed them to prevent a malfunctioning plugin from crashing the main process. This is why we have chosen an RPC-based plugin model.

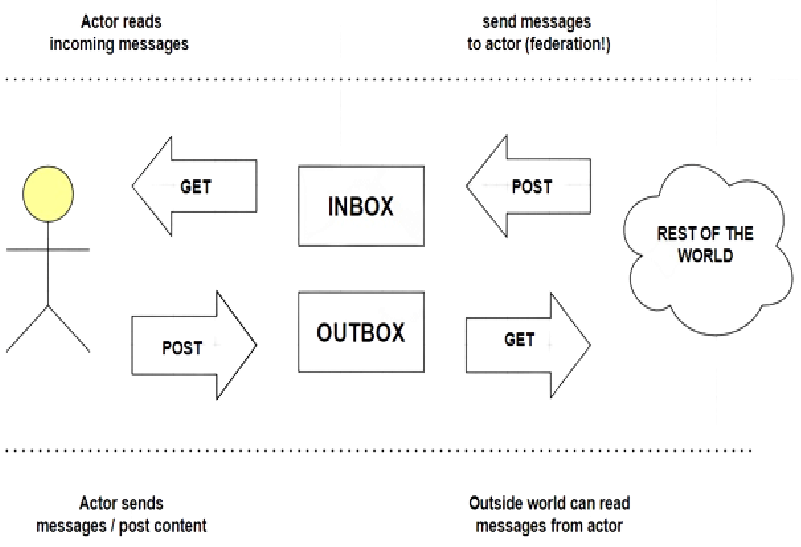
Actor reads incoming messages
send messages to actor (federation!)
GET
INBOX
POST
REST OF THE WORLD
POST
OUTBOX
GET
Actor sends messages / post content
Outside world can read messages from actor

Fig. 1. *RPC-Based Plugins*

*B. Dynamic UI Generation:*

Assemble's user interface (UI) will primarily be developed using SvelteJS. To simplify plugin development and integration with UI, we have devised a query-parse format. This format allows plugin developers to ship plugins without the need to worry about UI creation and integration; everything is handled through code, eliminating the need for YAML, JSON, TOML, or any other text formats. Defining everything in code provides developers with features like type-checking and intelligent code suggestions.

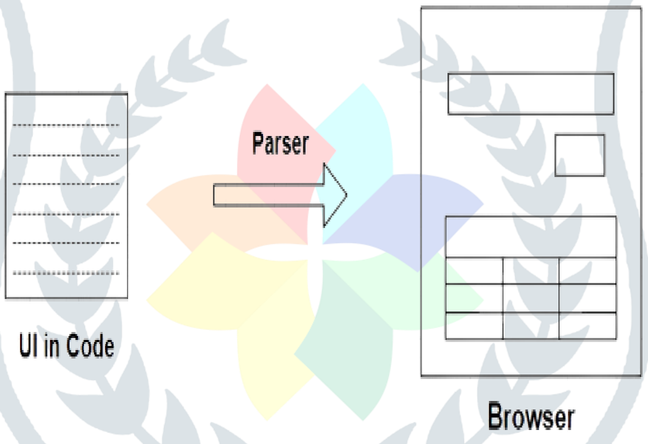

Fig.2. *UI Generation*

*C. Using Go at the backend*

Efficiency and Speed: Go compiles to fast, efficient native machine code, which is crucial for handling tasks in a decentralized network and a large user base. Concurrency: Go's built-in support for concurrency helps manage multiple tasks running simultaneously, ensuring responsiveness. Scalability: Go's lightweight goroutines and low memory usage make it suitable for handling growth in users and plugins without performance bottlenecks. Networking: Go's standard library provides robust networking capabilities, essential for building and maintaining a decentralized network.. Developer Community: Go has an active developer community, offering a broad pool of potential contributors and access to useful libraries. Simplicity and Readability: Go's straightforward syntax simplifies code understanding and maintenance, important when multiple developers work on plugins. Cross-Platform Compatibility: Go allows easy deployment on various platforms, accommodating diverse user needs. Security: Go's design focuses on security, reducing common vulnerabilities, which is crucial for a secure decentralized network.

*D. Using SvelteJS at the backend*

Svelte compiles code at build time, resulting in faster load times and better performance. Simplicity: Svelte's syntax is straightforward, making it easier for developers to learn and use. No Virtual DOM: Svelte doesn't rely on a virtual DOM, which can reduce complexity and improve performance. Reactive Data Binding: Svelte provides a simple way to handle real-time data updates. Component-Based: Svelte encourages the use of reusable UI components, making code easier to maintain. Small Bundle Sizes: Svelte generates smaller bundles, improving loading times. Integration: It can be integrated with various backend technologies, offering flexibility for different projects.

## V. CONCLUSION

In summary, Assemble represents an ambitious project aimed at revolutionizing the way educational institutions collaborate and operate. It envisions a future where a community-driven, decentralized platform powers seamless connections among institutes.

## REFERENCES

**[1]** Nieman, Katherine, and Sayeed Sajal. 2023. A Comparative Analysis on Load Balancing and gRPC Microservices in Kubernetes. Intermountain Engineering, Technology and Computing (IETC), pp. 322-327. IEEE, 2023.
[2] Karavisileiou, Aikaterini, Nikolaos Mainas, and Euripides GM Petrakis. 2020. Ontology for Openapi rest services descriptions. IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), pp. 35-40. IEEE, 2020.
[3] Bhardwaz, Saumyamani, and Rohan Godha. 2023. Svelte. js: The Most Loved Framework Today. 2nd International Conference for Innovation in Technology (INOCON), pp. 1-7. IEEE. 2023.
[4] Melton, Ryan. 2021. Securing a Cloud-Native C2 Architecture Using SSO and JWT. IEEE Aerospace Conference (50100), pp. 1-8. IEEE, 2021.
[5] Li, Kunying, An Ren, Yu Ding, Ying Shi, and Xiaobo Wang. 2020. Research on decentralized identity and access management model based on the oidc protocol. International Conference on E-Commerce and Internet Technology (ECIT), pp. 252-255. IEEE, 2020.
[6] Taheri, Saeed, and Ganesh Gopalakrishnan. 2021. GoAT: Automated Concurrency Analysis and Debugging Tool for Go. IEEE International Symposium on Workload Characterization (IISWC), pp. 138-150. IEEE, 2021.
[7] Gao, Chan, Hengxi Lv, and Yunzhang Tan. 2023. Multithreading Technology Based on Golang Implementation. 3rd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), pp. 603-608. IEEE, 2023.
[8] Jenkins, Louis, Tingzhe Zhou, and Michael Spear. 2017. Redesigning Go's Built-In Map to Support Concurrent Operations. 26th International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 14-26. IEEE, 2017.
[9] Noviani, Erina Fika, Bayu Kembara, Bakti Anugrah Yudha Pratama, Dyah Ayu Permata Sari, Ary Mazharuddin Shiddiqi, and Bagus Jati Santoso. 2022. Performance Analysis of AWS and GCP Cloud Providers. IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), pp. 236-241. IEEE, 2022.
[10] Todorov, Milen Hrabarov. 2021. Design and deployment of Kubernetes cluster on raspberry pi OS. 29th National Conference with International Participation (TELECOM), pp. 104-107. IEEE, 2021.
[11] Apostolakis, Nikolaos, Marco Gramaglia, and Pablo Serrano. 2022. Design and validation of an open source cloud native mobile network. IEEE Communications Magazine 60, no. 11 (2022): 66-72.
[12] Kim, Chorwon, Ryangsoo Kim, Geon-Yong Kim, and Sungchang Kim. 2021. Kubernetes-based DL Offloading Framework for Optimizing GPU Utilization in Edge Computing. International Conference on Information and Communication Technology Convergence (ICTC), pp. 143-146. IEEE, 2021.
[13] Wazan, Ahmad Samer, Romain Laborde, David W. Chadwick, Francois Barrere, and Abdelmalek Benzekri. 2016. How can I trust an X. 509 certificate? An analysis of the existing trust approaches. IEEE 41st Conference on Local Computer Networks (LCN), pp. 531-534. IEEE, 2016.
[14] Nguyen, Nguyen Thanh, and Younghan Kim. 2022. A Design of Resource Allocation Structure for Multi-Tenant Services in Kubernetes Cluster. 27th Asia Pacific Conference on Communications (APCC), pp. 651-654. IEEE, 2022.
[15] Čučko, Špela, and Muhamed Turkanović. "Decentralized and self-sovereign identity: Systematic mapping study." IEEE Access 9 (2021): 139009-139027.
[16] Ma, Perng-Yi Richard. 1982. A task allocation model for distributed computing systems. IEEE Transactions on Computers 100, no. 1 (1982): 41-47.
[17] Garcia-Molina, Hector. 1982. Elections in a distributed computing system. IEEE transactions on Computers 31, no. 01 (1982): 48-59.
[18] Yang, Heecheol, and Jungwoo Lee. 2018. Secure distributed computing with straggling servers using polynomial codes. IEEE Transactions on Information Forensics and Security 14, no. 1 (2018): 141-150.
[19] Foster, Ian, Yong Zhao, Ioan Raicu, and Shiyong Lu. 2018. Cloud computing and grid computing 360-degree compared. Grid computing environments workshop, pp. 1-10. IEEE, 2018, pp. 255-260, doi: 10.1109/ICSCCC.2018.8703351