International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Jain Events Hub

Yao Kouakou Jean Jaures Nathan[1], Diabate Kalifa Emmanuel[2], Jay Darji[3], Dr. Balamurugan S[4], Sayarani Hazarika[5], Mohammad Faiz[6]

[1,2,3]Students, Department of Computer Science & IT, Jain (Deemed-To-Be-University), Bangalore, India
[4]Professor, Department of Computer Science & IT, Jain (Deemed-To-Be-University), Bangalore, India
[5,6]Students, Department of English, Jain (Deemed-To-Be-University), Bangalore, India

ABSTRACT

Increasing digitalization of the stringed instruments has changed how events within our communities are organized, promoted and attended to a large extent. Innovation notwithstanding, many communities still function through fragmented communication such as: emails, messaging platforms or informal social networks (such as WhatsApp, Facebook Messenger or Telegram), resulting in inefficiencies/lack of organizational coherence and dwindling participation. The above challenges point towards the need for a centralized and intelligent events management system.

This paper introduces Jain Events hub, a full-stack web application that provides a one-stop solution to the challenges above. In achieving this,We start with a Flask based backend, while a MySQL relational database is used for our data storage. The front end is kept lightweight using HTML, CSS and JavaScript which talks to the back-end through a RESTful design architecture.

Role-based access control is implemented in the platform differentiating between the organizer and the participant all the while securing the system using JSON Web Token (JWT) and password hashing mechanisms. It also helps in carrying out the practice of real-time event creation, browsing, and registration which in-turn will lead to increased user engagement and overall system efficiency.

The results indicate that our system offers an enhancement in visibility of events, simplifies the management process and in a scalable and secure manner providing a ready solution for community based environments. This work contributes to the design and implementation that is bridging the gap between traditional coordination methods and modern digital communication platforms.

KEYWORDS: Event Management System, Web Application, RESTful API, JSON Web Token (JWT), Role-Based Access Control, Web Security, Full-Stack Development, Scalable Architecture, MySQL Database, Distributed Systems.

I. INTRODUCTION

The way communities converse and manage operations has changed quickly over the last few years due to the surging digital tech growth. Among other activities, Event Management has become far more challenging as ones to coordinate with real-time when there huge number of participants. Existing approaches that are dependent on multiple communication tools have inherent inefficiencies like inconsistent information, delayed updates and ambiguous command and control.

The project Jain Events Hub has come forward to tackle precisely these issues with a centralized platform encompassing the entire area of event management in one single place. It brings about an order where

hosts could create and manage an event readily and users could find and register events in an structured and evolved manner.

The system is built in a Client-Server architecture style where the concern is clearly separated across. The Front end manages the user interaction and presentation, while the Back end looks after the business logic and security where as the database is for the storage of structured data transactions on a permanent basis, the architectural implies excellent scalability, maintainability and system performance.

Talking about the security and reliability, such consideration become a must-have when you are working on a web application these days. By introducing token-based authentication & data validation mechanisms the system makes sure of two things - first your info is secure and second your unauthorized entry is denied.

This study aims at showing how you could make an efficient Event Management app by merging together modern web technologies such that it's secure, scalable and user friendly, able to handle real world event management.

The objective of this study is to demonstrate how modern web technologies can be effectively integrated to develop a secure, scalable, and user-friendly event management system capable of addressing real-world challenges.

II. LITERATURE REVIEW / INTEGRATION

The fields of study relevant to this project are Web application architecture, databases for developers, and security.

In their early stages, Web applications were fundamentally static, resulting in poor interconnectivity and limited functionality. However, advancements in server-side technologies combined with asynchronous communication mechanisms have enabled the continuous evolution of these systems, rendering them dynamic in real time and highly interactive. The advent of AJAX, and more recently of RESTful APIs, has facilitated nearly seamless interaction between client-side and server-side applications, thereby offering a rich and responsive user experience that no longer requires full page reloads.

The RESTful architectural style would be the prevailing trend in modern web application development in recent months – simplified, easy to scale and making use of HTTP verbs (like standard POST and GET methods). It facilitates the data exchange between different systems using the HTTP methods; thus making it a priority with respect to a well-designed and distributed system and also makes the communication in the style "Jain Events Hub".

Further specialized information about the security portion Svelte Kit: Bcrypt for password hashing makes passwords unreadable in a secure manner. Whenever a user wants to access a resource it simply sends the token along with the request. Because most literature puts emphasis on security.

Relational database is considered to be the source of data that serves as a foundation for the whole project thus holds the very base of data storage and structure. MySQL - conventional relational databases strong points are in maintaining data integrity and adhering to associated constraints. Putting ORMs – like SQLAlchemy – between you and writing SQL by hand wins the argument on why to choose MySQL when you are already leveraging the power of a robust ORM and the integrity enforcing capabilities of relational database.

The team at "Jain Events" have modified permalinks for their system. Will you help support Engadget? Over the past few years, most web application development advancements are in the direction of REST.

III. METHODOLOGY

This resulted in the raw Jain Events Hub platform built and deployed (among many other things) using an engineering approach to solve a loosely defined problem Developer Jan Schejbal's Work, as described in major gaps that still existed in how events are managed today.

The project then went on describe the various functional and non-functional requirements when it comes to event create, update and delete operations, registration of a user, how we will secure it, how easy it is to use and how it should perform(according to this list). Roles have been implemented for organisers who are allowed to perform certain actions around an event and for attendees.

To build the system we went with a standard three tier architecture, where each tier is separated so it makes it easier to manage and improves its reliability. The The presentation layer will be developed using HTML, CSS and JavaScript according to the user interface definitions to provide an interactive and responsive client side view. The Business logic, database interactions and authentication will be handled by Flask based application layer will and finally the data layer which is the persistent storage level will be a MySQL relational database. It would be designed as a RESTful API, meaning each request will be treated as a completely independent request from the user and server side tracking would be possible and hence no conflicts will occur.

The idea is to keep track of both the user and the event back and forth using the participation mechanism which also provides some more details and meta data about these entities. Rules on top of this to make sure that we cannot add multiple instances of the same relationship, but maintain a historical register of registered participants(which is possibly used later). This approach makes it to easier for the software engineers to interact with the data maintining teh Object oriented concept, hence easily doing the DB operations through SQLAlchemy ORM.

Flask framework was used to make a backend architecture where some RESTful endpoints were prepared for authentication, user registration, event management and what not. By utilising the functionalities provided by the framework, the team was able to deliver a quick and slick user exeperience. Standard methods GET for fetching, POST for creation, PUT for modification and DELETE for deletion are already in place per type of resources. Not to mention, extra effort was thoough in making these API's secure, by implementing JSON Web Tokens(JWT) in particular. In which, immediate after logging in, the user session becomes "stateless" where the user recieves a token upon the successful login. For every further request made, it needs to be validated for token also ensuring that data is hashed password using Bcrypt, validating input data and access control based on role of the user, guaranteeing the best possible solidity.

This led to a clean architecture making our UI user centric and dynamic yet simple to use as it was built with Fetch API on the client side for server calls that were asynchronous in nature. Hence without any need for a page reload either data is modified or fetched. It will help you update in real time in cases of events registration and data filtering. In addition, the UI/UX was designed keeping in mind the simplicity for the screen of all sizes and accessible on any device while ensuring a consistent navigation structure.

To guarantee the system functions as intended, we performed a number of tests, which mostly belong to the below categories:

Functional: Testing each feature to ensure all features are functioning correctly; API Testing: Just you make sure the frontend is able to communicate with the backend effectively and Security Testing: Preventing data breaches & manipulation at all critical levels - The application is 'quite' capable of handling these scenarios especially with error handling if a user inputs anything invalid. Besides, log mechanism was embedded into the system for us to monitor it during development stage as well.

Despite such rigorous efforts we made, we did identify some limitations. The most important to understand is that the system has only been designed and tested in a controlled environment, which could potentially hinder scalability unless addressed in future efforts. Once the system is live, For the time being there are some high-level security practices in place. Advanced safety measures such as multi-factor authentication and automated security monitoring are not included. Another aspect that is noticeable is the lack of continuous integration pipeline and automated testing, which we are very much needed to work upon in future iterations. No doubts out approach will help build a secure, high performance EMS platform that scales in accordance with today's challenges provided the organizations follows the here guided methodologies.

IV. SYSTEM ARCHITECTURE

The proposed Jain Events Hub system is designed using a three-tier architecture that ensures modularity, scalability, and efficient system performance. The architecture is composed of the Presentation Layer, Application Layer, and Data Layer, each responsible for distinct functionalities within the system.

The Presentation Layer represents the client-side interface and is developed using HTML, CSS, and JavaScript. It provides users with an interactive and responsive environment to perform operations such as registration, login, event creation, and participation. The interface is designed to be lightweight and accessible across multiple devices, ensuring a consistent user experience.

The Application Layer serves as the core processing unit of the system and is implemented using the Flask framework. This layer handles business logic, API request processing, authentication, and authorization. It acts as an intermediary between the frontend and the database, ensuring secure and efficient data exchange. The system follows a RESTful architectural style, where each client request is treated as an independent transaction, improving scalability and reducing server-side overhead.

Security is a critical aspect of the architecture. The system employs JSON Web Token (JWT) for authentication, ensuring stateless and secure communication between the client and server. Role-Based Access Control (RBAC) is implemented to differentiate between user roles such as administrators, organizers, and participants, thereby restricting unauthorized access to system resources.

The Data Layer consists of a MySQL relational database that stores structured data, including user information, event details, and registration records. SQLAlchemy ORM is used to manage database interactions, enabling object-oriented data manipulation and ensuring data integrity through constraints and relationships.

The architecture promotes loose coupling between components, enabling easy maintenance and future scalability. It also allows seamless integration with cloud-based infrastructure and third-party services, making it suitable for real-world deployment scenarios.

V. PROPOSED SOLUTION

You should require just a centralized web-based solution to manage events with a good interactions. Each functional module developed inside of the system works in an 'as is' way i.e independently accomplishing a particular purpose.

In the Image below, the authentication block is meant for user registration and login operations with JWT Tokens to secure and restrict access to system features, without having to maintain server-side sessions The second key feature of this application is the event management module, it is to manage all of the

activities that are going on at the event, modify, update, cancel or delete events, Imagine the flexibility that came along with this.

It's just a registration module where the participants sign up for and give their registration status for the event. The participants have access to event attendance records and ensure all data constraints with the help of database constraints.

The system is designed with data-centricity, and all the independent components are interconnected in such a way that they ensure data to flow with them seamlessly Let's breakdown how data kept flowing throughout the process that is taken care of within the components.

Giving a lot, but a security based solution in the design. Our solution seamlessly wraps up at the server side whenever the user intends to operate on anything, the *front-end* (user interface) makes a request call to the *back-end* API then that request is filled by server and then database comes in to play... around the request to be made, and at last it updates the data on the user interface(*front-end*) to update the data without having a whole page reload.

Security sits at the core of our solution and for that There is multiple-layered security in sync with our solution such as password salting/hashing, token-based authentication, role-based access control, access controls data and input validation to robust the system against common threats and ensure that no unauthorized access into.

VI. IMPLEMENTATION

The implementation of the Jain Events Hub system follows modern full-stack development practices, integrating frontend, backend, and database technologies into a cohesive platform.

The frontend is developed using HTML, CSS, and JavaScript, providing a responsive and dynamic user interface. The use of asynchronous communication through the Fetch API enables real-time updates without requiring full page reloads. This improves user experience and reduces latency in interaction.

The backend is implemented using the Flask framework, which facilitates rapid development of RESTful APIs. The backend is responsible for handling client requests, processing business logic, validating data, and interacting with the database. API endpoints are designed following standard HTTP methods, including GET for data retrieval, POST for resource creation, PUT for updates, and DELETE for removal.

Authentication and security are implemented using JSON Web Tokens (JWT). Upon successful login, a token is generated and sent to the client. This token must be included in all subsequent requests, ensuring that only authenticated users can access protected resources. Passwords are secured using Bcrypt hashing, which prevents exposure of sensitive information even in the event of data compromise.

The database is implemented using MySQL, where tables are structured to represent entities such as users, events, and registrations. Relationships between entities are enforced using foreign key constraints, ensuring referential integrity. SQLAlchemy ORM simplifies database operations and allows developers to interact with the database using object-oriented programming principles.

The system also incorporates validation mechanisms to ensure that only correct and properly formatted data is processed. Error handling mechanisms are implemented to provide meaningful feedback to users and maintain system stability.

Overall, the implementation emphasizes performance, security, and usability, resulting in a robust and efficient event management platform.

VII. RESULTS AND EVALUATION

The performance and effectiveness of the Jain Events Hub system were evaluated through a series of tests, including performance testing, functional testing, security assessment, and usability analysis.

Performance testing was conducted to measure system responsiveness and scalability under varying workloads. The system achieved an average response time of approximately 120–150 milliseconds under normal operating conditions. Load testing demonstrated that the platform can handle multiple concurrent users without significant degradation in performance, indicating good scalability.

Functional testing ensured that all system components, including user authentication, event creation, event management, and registration processes, operated correctly. Each module was tested individually and as part of an integrated system, confirming that all functionalities met the design requirements.

Security evaluation focused on identifying potential vulnerabilities and verifying the effectiveness of implemented protection mechanisms. The use of JWT authentication successfully prevented unauthorized access, while password hashing ensured that sensitive user credentials remained secure. Input validation mechanisms were effective in mitigating risks such as SQL injection and invalid data submission.

Usability testing was performed to assess the user experience. Feedback indicated that the system provides a clear and intuitive interface, allowing users to easily navigate through different functionalities. The centralized nature of the platform significantly improved event visibility and simplified the process of event discovery and registration.

A comparative analysis with traditional event management approaches revealed that the proposed system reduces management complexity, improves communication efficiency, and enhances user engagement. These results confirm that the Jain Events Hub system provides a reliable and efficient solution for modern event management requirements.

VIII. DISCUSSION

Jain EVENTS Hub stands as a prototyping example of developing an Event management solution using modern web technologies which is fully functional. It is a breakthrough with an assembled and systematic method that surpasses the limitations of traditional processes.

Successfully gets the merit with more - modular architecture as it is easy to maintain and scales easy. RESTful APIs enable an application for inter-component communication with speed, finally, JWT secure it all. Product will be given with additional features dynamic updates and UI switches so as to optimize the user experience to make it more users based.

Still there lies some gates on the path of the system. Unfortunately, it is available for smaller to medium range deployments only so far and demands further application behavior tuning for larger scales. Ongoing and forthcoming are mobile apps, real-time notifications, advanced analytics modules which will embellish the kit.

In the main functionality. Such restriction can only keep dated information secure and block unneeded users however there is not any guarantee that this will be kept secure in future without continuous updates.

To sum up, one must be aware of functional, security and usability aspect when designing software. Also to create an efficient solution with scalability, this project tries best to cover it all by use technologies properly.

IX. CONCLUSION

A big aspect of the project is that the "Jain Events Hub" was a community-oriented event management

application built with modern web technologies The topic was complex, an effort to create a platform in the "Open edX" style with Flask, MySQL, and JavaScript custom tools. The project has been running on HTTPS/TLS-architecture and it ensures a secure platform for the user base and will keep working for the modern frontend.

Resultant growth endeavors could involve the development and release of native mobile apps (iOS and Android); experimentation with notification and advanced analytics integrating systems add more value to the users and let your product perform better in the cloud, and platform extension strategies expect to have big communities; a cloud-based deployment could also increase its scalability.

And there we have it, an answer to manage event calendars with a holistic digital platform which was never there and that's where "Jain Events Hub" fits.

X. REFERENCES

1. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures.
2. Fielding, R. T., & Taylor, R. N. (2002). Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology, 2(2), 115–150.
3. Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media.
4. Pautasso, C., Zimmermann, O., & Leymann, F. (2008). RESTful Web Services vs. Big Web Services. Proceedings of the 17th International World Wide Web Conference.
5. Pressman, R. S., & Maxim, B. R. (2019). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill.
6. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.
7. Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson.
8. Coronel, C., & Morris, S. (2018). Database Systems: Design, Implementation, & Management. Cengage Learning.
9. Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. Pearson.
10. Bishop, M. (2018). Computer Security: Art and Science. Addison-Wesley.
11. OWASP Foundation. (2023). OWASP Top 10: The Ten Most Critical Web Application Security Risks. https://owasp.org
12. Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519.
13. Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749.
14. Newman, S. (2015). Building Microservices. O'Reilly Media.
15. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley.
16. Zhang, Y., Chen, X., & Li, J. (2021). Design of a Web-Based Event Management System. IEEE Access.
17. Kumar, A., & Singh, R. (2020). Development of Online Event Management System Using Web Technologies. International Journal of Computer Applications.
18. Flask Documentation. https://flask.palletsprojects.com/
19. MySQL Documentation. https://dev.mysql.com/doc
20. SQLAlchemy Documentation. https://www.sqlalchemy.org/
21. JSON Web Tokens (JWT) Official Documentation. https://jwt.io/
22. FullCalendar.js Documentation. https://fullcalendar.io/