REVUP

[1]Nazneen Kiresur, [2]Prathik N, [3]Peeyush Yadav, [4]Nithya Shakti R B, [5]Nikhil S Tengli

Authors affiliation, [1]nazneenkiresur@gmail.com,[2]prathik252416@gmail.com, [3]piyuankit@gmail.com, [4]jeevika75@gmail.com, [5]nikhil.tengli@reva.edu.in

Abstract

An event management system assists event organizers with event planning, execution, and reporting. Planning, organizing, and executing an event is a time-consuming task for any institution; therefore, with such systems in place, campus event planners will be able to book large-scale events, engage with service providers, and maintain a single source of record for repeatable, quantifiable events with the assistance of a comprehensive event solution. With an event of substantial magnitude, it is understandable that the infrastructure and development expenses for the same management systems would be prohibitively expensive. As a result, we seek to suggest an ideal solution for the challenge using a parallel-microservice-led architecture that aids in effective load balancing schemes and resource utilization. In addition, a large-scale notification system based on parallel computing is proposed in this study in order to provide quicker responses than traditional approaches. This would also effectively use the platform as a means for information exchange about such events and occasions.

Keywords. Event Management System (EMS), Microservices, Large-scale Notification Systems (LSNS)

1. INTRODUCTION

An event is entirely about individuals collaborating to design, operate, and engage in an experience. It is an activity that brings the target group together in space and time, a gathering where a message is transmitted, and actions take place. Numerous events are held in the modern world, including workshops, conferences, and fests[1]. An event involves organizers, participants, and a variety of other roles. Without a doubt, most organizations are devoting time and resources to developing effective strategies. The organizing procedure is time-consuming and involves a great deal of documentation. Apart from that, organizers are required to publicize their event using a banner and a social media platform. The most prevalent medium for sharing knowledge is social media, however, because organizers only communicate or publish information once or twice, it may not reach another student[1][2]. Organizers are passively posting events, which has a significant impact on program engagement. When the number of participants is large, managing the events without a proper system becomes quite difficult.

However, some systems focus on managing events, as well as issues in ensuring that organizers and attendees engage effectively. Certain systems are quite costly. In the current situation, existing systems have numerous flaws that render them ineffective in

carrying out events. Currently, all working systems are manual. As a result, there are numerous concerns about the process's security, validity, and feasibility. It's challenging to keep track of all events, clients, and services. Unmanaged planning might cause event execution to be delayed. Hence effective EMS is required to resolve this issue.

Our project is an online EMS, or more accurately, a gateway in the form of a website that will assist an event organizer as well as participants and stakeholders in the event's functionality. The following are some of the system's features: The system allows event organizers to log in and publish information about their events. This system has three major goals: assess and design a system for managing events, construct the EMS using a relational approach, and notify users about the event happenings through different mediums like mail, WhatsApp, etc.

The system's benefits include lower advertising costs, reduced paperwork, more efficiency and effectiveness, and fewer human errors. The system also provided an interface for maintaining all the reports of the events which help the organization during audits. It will undoubtedly assist organizers in digitally promoting events and exponentially increasing registrations and participation.

2. RELATED WORKS

Many architectures and designs have been proposed for EMS. This section enlists the efforts made in the prior:

1. The authors of [1] propose a generic academic event management platform capable of dynamically creating web pages for any academic event. A person could quickly create a website for his or her event by selecting certain options for the website, such as venue, registration, and call for papers, and populating the appropriate data.
2. The authors of [3] proposed a .NET framework-based architecture for the application's entire design. However, the application was limited in terms of advertising and showing pertinent information about upcoming events. The team's primary objective was to create a login-based EMS that would provide additional information on the participants.
3. The design presented in [4], in which the authors briefly discuss the many characteristics of the microservice architecture, and the architecture that has recently gained attention, in which big modules and components of system applications are broken down into simpler and more elaborate modules.
4. In [5], Amir Saleem and his colleagues proposed a system for event booking in which hotels and clubs can utilize a web-based application system. Additionally, the system can be used as a piece of software to promote all possible booking places. Rather than searching for materials throughout the site, the user may find them all in one location. This strategy was efficient and assisted the user in saving time and money.
5. The author and his colleagues present a study in [6] that examines a variety of companies and their techniques for managing large-scale events that occur on their premises. As a result, they developed and offered a mobile application allowing users to register for a range of events.

6. "Event optimizer" was proposed by Mr. Nagesh and his colleagues in [7]. There were numerous tightly coupled modules in use across the application, all of which serve a specific purpose in this proposal.
7. Rowstron and his teammates proposed a large-scale notification system-based infrastructure in [8], which was built on top of a generic peer-to-peer object location and routing substrate called Pastry. Scribe is capable of scaling to a high number of subscribers, publishers, and subjects while providing efficient application-level message multicasts.
8. In [9] they proposed a system with clearly defined roles and modules, each with distinct tasks and accessibilities across the proposed event management system. The module layout was well-supported in terms of institutional organization.

3. ARCHITECTURE DESIGN

As demonstrated by the history of software and system development over the last fifty years, software architecture is critical to software systems, providing plausible insights, eliciting the appropriate questions, and providing general tools for thought. Table 1. summarizes the existing software architectures.

Module Name	Description
Service Oriented	SOA is a business-IT-aligned approach in which applications rely on available services to facilitate business processes. A service is a self-contained reusable software component provided by a service provider and consumed by service requestors.
Distributed Computing	A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.
Parallel Computation	Parallel computing is a type of computing architecture in which several processors simultaneously execute multiple, smaller calculations broken down from an overall larger, complex problem.
Microservices	Inspired By Service-Oriented, Microservices - also known as

	microservice architecture - is an architectural style that breaks systems and applications down to a more granular, modular level.

Table 1: Existing Architectures

A unique architecture based on existing approaches was developed to improve the performance of conventional load balancing schemes. The proposed design is intended to host computationally intensive operations as separate entities, hence boosting the response time of the server. Figure 1. illustrates a high-level explanation of the proposed software architecture.

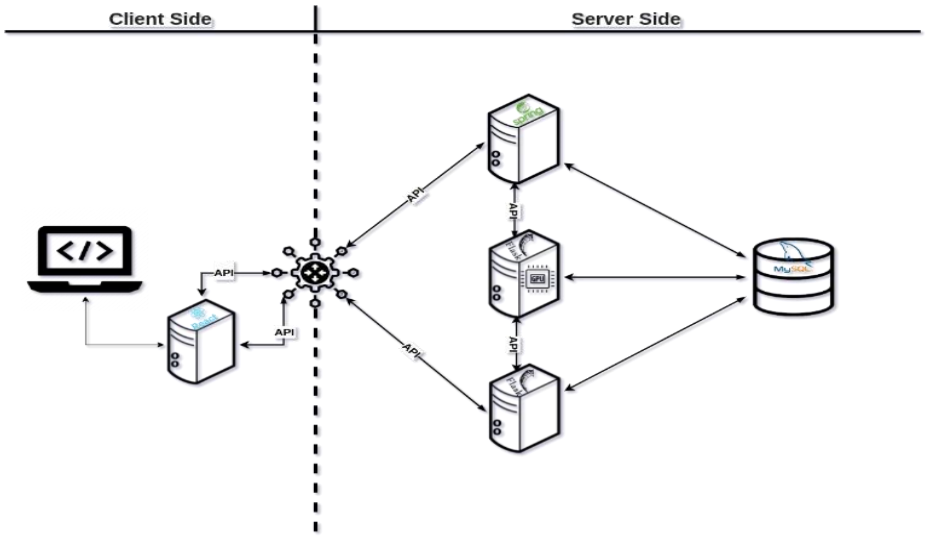

Figure 1: Proposed Architecture

The following are some of the most popular frameworks now available on the market, which are employed in this architecture:

1. Spring Boot: Spring Boot is used to create small, self-contained, ready-to-run applications, which may give your code greater flexibility and robustness. Spring Boot's numerous purpose-built capabilities simplify the process of designing and running microservices in production at scale[10].
2. Flask: Flask is a Python-based micro-web framework. It is characterized as a microframework due to the fact that it is not dependent on any specific tools or libraries [11].

3. MySQL: MySQL Database Service enables developers to quickly create and deploy safe applications using the open-source database[12].
4. React JS: React JS is a free and open-source JavaScript library for developing user interfaces for single-page applications. Since it's an open-source framework, it is wide open to various experimentations and resources that are available throughout the community[13]. React is also known to make the designs more dynamic and alive, and hence it was imbibed as the sole front-end framework for the proposed system
5. Apache Tomcat Server: This is an open-source Java servlet container that supports various Java Enterprise Specifications, such as Websites API, Java-Server Pages, etc.[14]. It is one of the most extensively used Java servers owing to various features such as high extensibility, a well-tested core engine, and a long-lasting design.

4. CONCEPTUAL DESIGN AND IMPLEMENTATION

Events of various types, from seminars to sports to conferences to cultural festivals can be handled and maintained by a generic EMS. Our research has led us to include the following requirements for our website:

1. User Access Privileges.
2. Event Report Management.
3. Large Scale Notification system

A. Conceptual Design

The proposed EMS's features are briefly summarized below:

1. User Access Privileges.

   Our application supports six distinct user roles which are anonymous user, user, Institutional user, organizer, admin, and super admin. Table 2. shows the breakdown of the different types of roles each user plays in our application.

USER ROLE	**DESCRIPTION**	**PRIVILEGES**
Anonymous User	All visitors to our website are regarded as anonymous users who must first register in our application to become a user/Institutional User.	View all events Register for the application.

User	Users of the application include all registered users who are not affiliated with the institutions.	Sign In. View all events. Register for events. Raise queries.
Institutional User	Institutional Users of the application include all registered users who are affiliated with the institutions.	Sign In. View all events. Register for events. Raise queries. Apply for Organizer.
Organizers	The organizers of the application may be students, faculty, or staff members responsible for organizing the events at the institution.	Sign In. View and register for events. Create and manage events. Create and manage event reports. Raise queries. Apply for Admin.
Admin	Every department in institutions will have an admin who will be in charge of monitoring all the activities regarding the events in the department.	Sign In. View and register for events. Create and manage events. Create and manage event reports of the entire department. Create and manage event types. Create and manage Organizers. Create and manage queries.

Super Admin	There will be only one super admin in the institution who is responsible for monitoring the entire application.	Sign In. Manage all event reports. Create and manage event types. Create and manage departments. Create and manage Admins.

Table 2: Types of User Role in the Application

2. Event Report Management:

Event reports are the documents that support the insights gathered from the events which are to be curated from time to time during or after the completion of the events. Our web application support uploading the downloading the event reports. This is limited to the respective authorities which are essentially the Organizer, or the admin associated with the event. The reports then can be downloaded by the same user. Admins, however, have special privileges that allow them to see the events where reports have been uploaded directly, a case not provided to the organizers.

3. LSNS:

Notifications are alerts that are used to notify users about the updates that are of concern to them. A publish-subscribe model has become the fore parameter for distributed systems over time[8]. In our architecture, we proposed a parallel computed, large-scale notification system, which not only adheres to making use of the resources efficiently but also makes the entire notification aspect of the EMS faster as compared to the traditional practices of circulating notices through the same medium.

B. Implementation

RevUp is a web-based event management application that allows institutions to plan and manage events. All our application servers are hosted on the Apache Tomcat server. React Js was used as the sole front-end framework. This was done to ensure we have a user-friendly, feature-rich, and lightweight user interface. MySQL was used to interact and handle the entire application's data, right from user sign-in to handling and changing of user roles amongst the institution. It is also used for storing the event data and all its related data as well.

The backend servers consisted of 3 modules which were built using Spring Boot and Flask. The first model completely dealt with all the event-related information like event details, department details, event reports, and event classification which was completely built using spring boot which delivers a high level of security using secure authentication. The second module was developed using a flask that mainly dealt with user-related information

which required a higher level of security compared to events. So, we incorporated key encryptions to authenticate the user which prevents threats like SQL Injection protection, and guards against XSS (Cross-Site Scripting) attacks, etc. As for the LSNS, as this required proper load balancing and efficient utilization of the resources, this final module was developed over Flask, owing to the fact that it is a lightweight WSGI framework[15] and it supports a varied array of Python modules to support multithreading.

One of the benefits of this application is that it requires no prior training to operate and can be used to create and maintain both technical and non-technical events.

C. *System Workflow*

The proposed system of the platform is as follows:

1) *Handling User Privileges:* Like in any organization, the platform allows the users to have distinct roles and responsibilities. These can be assigned to the users only if the higher role accepts the requests of the candidates. The system maintains the hierarchy in such a manner that an institutional user can only become an organizer and thus this request can only be verified by the said peer, the admin. Similarly, the organizer and admin can be reviewed by their respective peers. Figure 2 depicts a scenario, where an organizer has requested to be promoted to *Admin*. The same will come into action once Admin approve his/her request.

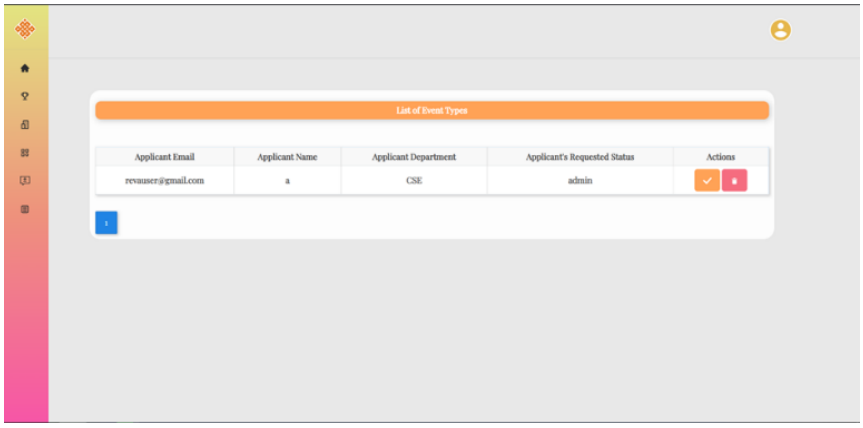

Figure 2: Admin approving requests for organizers

2) *Creation of a New Event:* The creation of an event is limited to only the organizers and admins of the department. To do this, the event creator needs to fill-up the form (shown in Figure 3.) and enter the details that prevail regarding the event. This also includes uploading an event poster which would

be used throughout the platform. Once created the event created gets enlisted on the dashboard where the other users can also view the same from the same point of time.

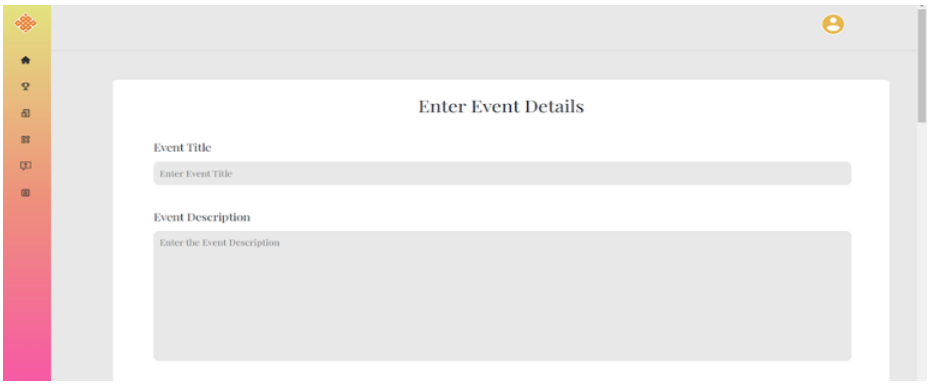

Figure 3: Form for creation of a new Event

3) *Data Updating and Manipulation:* Updating or manipulation of the present data associated with an event can be edited by its creator or the admin of the corresponding handling department of the organization. The user gets the present data in the respective fields which aids them to make the modifications easily. The view also allows uploading and downloading of the event reports provided the event has been completed (shown in Figure 4).

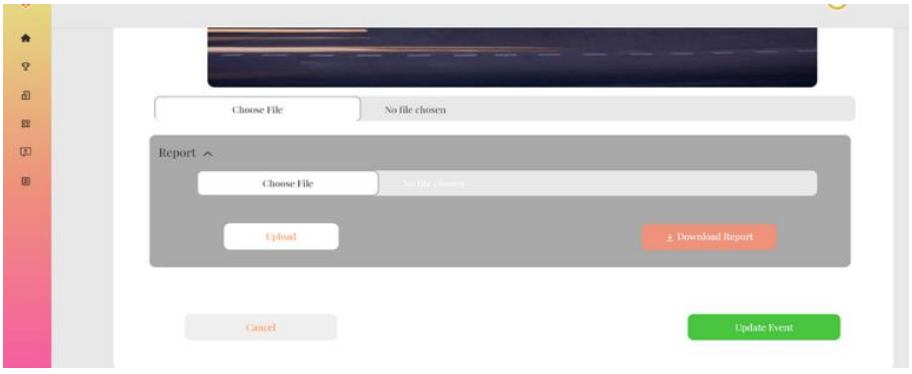

Figure 4: Update Event Details along with privileges to upload and download report

4) *User Views:* Based on the user roles, views have been assigned to each role wherein the User and Institutional user roles have the read-only view for the events. As for the Organizer and Admin, they have been provided with the option to edit or upload the events that they oversee. As for the super admin, the privileges are only to maintain the user privileges and the institutional data regarding the departments, or the requests made by different users in the institute. Figure 5 shows the Home Page (*Event Page)* which is prescribed only for Anonymous users. The sidebars and other functionalities get changed based on the different user roles that are defined.

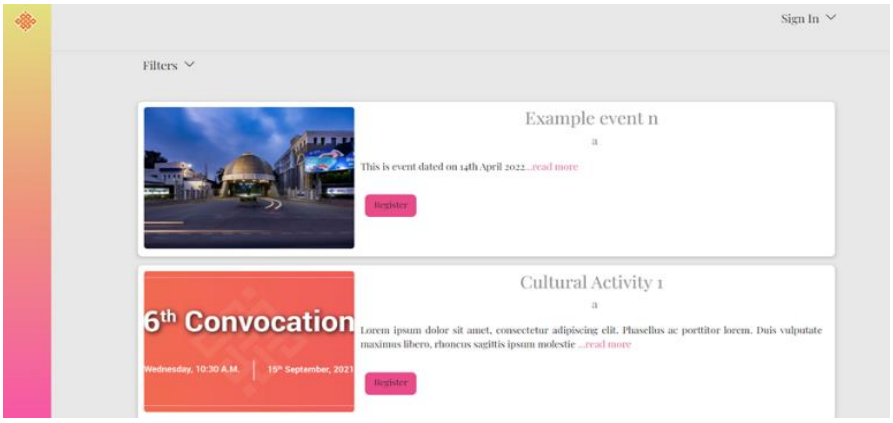

Figure 5: Event Page loaded for an Anonymous user

5) *Privileges:* Privileges have been assigned throughout different user roles that are allotted during the signup. This can be elaborated as
   a) Anonymous User: The user can only view the events that are currently "active" or whose registration deadline has not been passed yet. This is to only lead the user to either Signup or Sign-In to the platform.
   b) User: The role of the user is for an event that is open to all candidates, regardless of their association with the hosting organization. The users can register for such events and thus can participate.
   c) Institutional User: These are the types of users that are students or members of the institution other than the Organizer or the Admins roles. Such users can register as well as request to be promoted to the user role of Organizer provided, they are entrusted with the responsibilities, and the application for the same is approved by their peers.
   d) Organizer: Organizers or the members of the platform who have been assigned to host or handle the events associated with their departments. They can create or update the events and thus can also upload and view the reports of the same.

e) Admin: Admins are the users that manage the platform when it comes to their department. They can handle the creation and updating of events. Admins also have the privilege to create or update the Event Type or the category to which the event belongs. This category comes in handy whilst creating or updating the events. Admins can also view and download the report for the events completed in their department. Finally, they also have the privilege to approve the requests of the Institutional Users or the Organizer to be promoted to the requested roles.

f) Super Admin: This user overlooks the creation and management of the User Privileges. They are not to create events but can have all other privileges that other admins may not have. They can create or modify the details regarding the department or the event categories. Departments are only to be viewed by other event creators (Organizer and admin) wherein during any creation or updating, the same is taken from the user details itself.

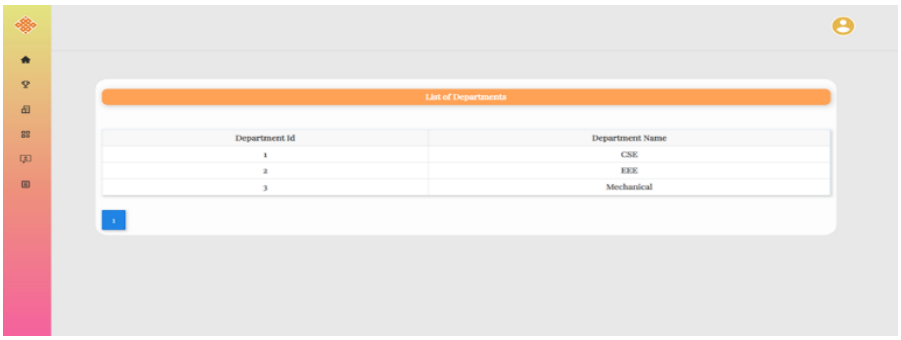

Figure 6: List of Available Departments

5. EXPERIMENTAL RESULTS

Numerous experiments and plannings were conducted in order to execute the proposed architecture over the web application. These addressed a variety of topics, including testing and confirming the accesses permitted to each user role and the way these roles' requests are handled. The platform was distributed to various users on the premises in order to obtain feedback and approvals on the platform's functionality. This aided in the implementation and testing of our project's needs, as well as in broadening the platform's usefulness. With regards to the Large-scale notification system, we experimented with traditional means of alerting people through WhatsApp and email, but the audience pool can be very enormous (as little as 10,000 individuals), and in such circumstances, distributing alerts in a timely manner was not possible. Thus, after doing further experiments with the multithreading strategy, we saw a significant improvement in notification delivery time for the audience size that we had available. The experimental findings from evaluating the server response time for a test size of 10,000 test user emails in the large-scale notification system are summarized in Table 3. The y-axis indicates the

response time (in seconds), while the x-axis indicates the computing methodology utilized to distribute the emails during the test runs.

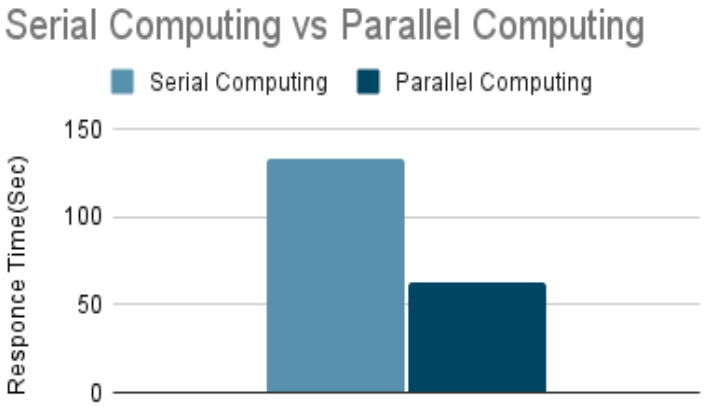

Table 3:Performance analysis for Serial vs Parallel Computing in delivering large scale notifications.

6. CONCLUSION

Our goal was to minimize human effort and make event planning easier for users, administrators, and event organizers. We sought to create an EMS that could be used to manage any type of event effectively and efficiently for any organization. This approach is now applicable to a variety of events, including conferences, seminars, cultural functions, and festivals. Our long-term objective is to incorporate a machine learning model that will provide event organizers with enhanced insights into how to increase event registrations. However, there are some drawbacks, such as the fact that user registration has not yet been implemented. However, registration will be implemented in the future, as well as an expansion of the spaces available for the creation of various types of events.

7. REFERENCES

[1] S. Islam, R. Majumder, S. Sultana, S. Nasrin, and R. Islam, "Toward a Generic Event Management System for Academia," 2019 5th International Conference on Advances in Electrical Engineering (ICAEE), 2019, pp. 706-711, DOI: 10.1109/ICAEE48663.2019.8975626.

[2] P. A.-A. S. Ulul Azmi and N. Ibrahim, “UTHM Students’ Event Management System”, aitcs, vol. 2, no. 2, pp. 697–716, Nov. 2021. [3] Gartner Report, Financial Times, 2007.

[3] Arsheen. Khan, Aarti. Pundalik, Tanvi. Shinde, Sneha. Gupta, S.J. Patil,” Event Management System” for International Research Journal of Engineering and Technology (IRJET)

[4] Dragoni, N. et al. (2017). Microservices: Yesterday, Today, and Tomorrow. In: Mazzara, M., Meyer, B. (eds) Present and Ulterior Software Engineering. Springer, Cham. https://doi.org/10.1007/978-3-319-67425-4_12

[5] Amir Saleem, Davood Ahmed Bhat, Mr. Omar Farooq Khan, “Review Paper on an Event Management System” for International Journal of Computer Science and Mobile Computing ISSN 2320–088X.

[1] Akash Verma, Gunjan Srivastava, Himanshu Verma, Mayank Johri, Archana Bhalla, “Study on Event Management Applications” International Journal of Innovative Science and Research Technology ISSN No:2456 – 2165.

[2] Mr. J Nagesh Babu, Ms. Srujana J M, Ms. Srusti U M, Ms. Sushma Kulkarni, “Event Management System” International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 6, Issue 5, May 2019

[3] Rowstron, A., Kermarrec, AM., Castro, M., Druschel, P. (2001). Scribe: The Design of a Large-Scale Event Notification Infrastructure. In: Crowcroft, J., Hofmann, M. (eds) Networked Group Communication. NGC 2001. Lecture Notes in Computer Science, vol 2233. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45546-9_3

[4] Dhanawade Phulabai Pandurang1, Mohite Damini Maruti, Sakhare Dipali Balu, S. T. Shirkande “A Design on Centrally College Event Management System” International Journal of Research in Engineering, Science and Management Volume-3, Issue-6, June-2020

[5] Spring | Microservices https://spring.io/microservices

[6] Flask https://flask.palletsprojects.com/en/2.1.x/

[7] MySQL /https://www.mysql.com/

[8] React https://reactjs.org/

[9] Apache Tomcat https://tomcat.apache.org/

[10] Flask | The Pallets Projects https://palletsprojects.com/p/flask/