RESEARCH ARTICLE OPEN ACCESS

VENUE: A Real-Time Event and Movie Booking Platform

Aman Karar, Pulkit Pandey, Piyush Soni, Rohan Choudhary, Laxmikant Vashishtha

Department of Computer Science Engineering, Global Institute of Technology, Jaipur, India

ABSTRACT: Effective People buy tickets for movies and events in a way now because of online booking sites. However a lot of these systems have issues. For example sometimes people book the seat at the same time. It also takes a time to update the seats.. Sometimes the information about the seats is not the same everywhere. This paper is about VENUE, which's a new system for booking events and movies. VENUE is designed to make sure that booking seats is reliable and that updates happen away. The system uses MongoDB, Express.js, React and Node.js. It also uses real-time communication and special database operations. The main goal of VENUE is to prevent people from booking the seat at the same time. It does this by using database operations that make sure booking a seat is a single step. VENUE also uses a technology called WebSocket to update all users about which seats are available right away. VENUE has a system that can handle a lot of users at the time. It includes features like security checks different levels of access, managing bookings and organizing events. The way VENUE is built shows that modern web technologies can work together to create a system for booking tickets. VENUE can handle users without any issues. VENUE is an event and movie booking platform that is designed to guarantee dependable seat management and instantaneous booking updates, for VENUE.

Keywords — MERN stack, real-time apps, concurrency control, WebSockets, transactional databases, and event booking systems.

1. INTRODUCTION

When we talk about booking systems they are a crucial part of the way we manage events and entertainment these days. We need systems that let people buy movie tickets or book seats for events to be really good at what they do. These systems have to be reliable and respond quickly.

One big problem with these systems is when a lot of people try to book something at the time. This can cause problems like people getting the seat or the database getting mixed up. Older booking systems often just update the database in a way, which does not work well when a lot of people are using it at the same time. If a lot of people try to book the seat at almost the same time a system that is not good at handling this can give the seat to more than one person. This makes people not trust the system and it hurts the reputation of the platform.

This paper is about VENUE, which's a system, for booking things in real time. VENUE is designed to deal with these problems by making sure that everything is done correctly and the system is reliable. The system uses ways of making websites and strong backend logic to make sure that seats are given out in a way that is safe and uses database transactions.

When we use a booking platform we want to know if a seat is available away. We expect the system to show us which seats are taken so we do not try to book something that's not available. VENUE makes this happen by using a way of communicating that instantly tells all connected devices when a seat is taken or becomes available. The main goal of this project is to create a system for booking events that can handle a lot of people using it at the time. We want this event booking platform to be able to handle real-time interactions with users while making sure that all the information is correct and up, to date.

2. LITERATURE REVIEW

Online booking systems have evolved significantly with the growth of cloud computing and real-time web technologies. Several large platforms such as BookMyShow, Eventbrite, and Ticketmaster have demonstrated the importance of scalable architectures capable of handling high volumes of concurrent requests. Early booking systems relied primarily on synchronous server-client communication models. These systems often faced performance limitations when user traffic increased, particularly during popular events or movie releases. In such cases, the lack of concurrency management could result in seat overbooking.

Research in modern web system architecture suggests that transactional database operations play a crucial role in preventing inconsistencies during concurrent updates. Atomic operations ensure that database states remain valid even when multiple users attempt to modify the same resource simultaneously. Additionally, the adoption of real-time communication technologies such as WebSockets has enabled more responsive interfaces. Instead of relying on repeated polling requests, WebSockets allow servers to push updates directly to clients whenever relevant data changes.

Recent studies on web applications show that JavaScript-based full-stack frameworks, like the MERN stack work well.

They make development easier by using the programming language for both frontend and backend. This also makes it simpler to add real-time features. The VENUE platform uses these ideas. Adds booking logic live seat updates and access control based on user roles all, within one full-stack system MERN stack.

The VENUE platform and MERN stack make it easier to build and manage applications.

3. SYSTEM ARCHITECTURE AND TECHNOLOGY STACK

The architecture of VENUE follows a layered design consisting of three primary components:

1. Frontend Layer
2. Application Server
3. Database Layer

The frontend of this system is made with React. This means the website is easy to use and it changes quickly when you are looking at events picking seats and handling bookings. React is good at updating the website when the seats that are change. The backend of the system is built with Node.js and Express.js. This part of the system does important things. It makes sure people are who they say they are. It checks to see if booking requests are okay. It handles money transactions. It also helps people talk to each other in time.

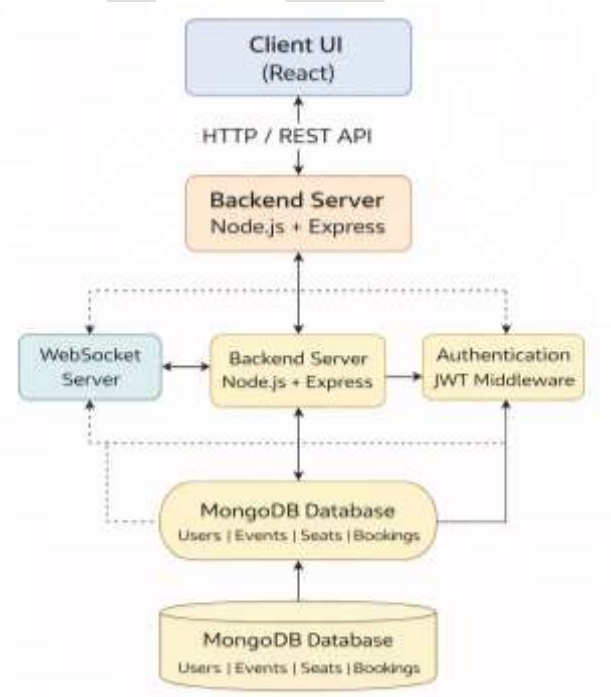

Fig. 1. Overall system architecture of the VENUE booking platform

The database part of the system uses MongoDB. This is a way to store information. It also supports something called -document transactions. These transactions are very important, for making sure that booking operations work correctly and do not get messed up. This means that when you book something MongoDB makes sure everything is done properly and consistently so you do not have to worry about the booking not working right. The system also uses WebSocket communication.

This helps to give users real-time updates when the seat availability changes. So when a seat is no longer available, during the booking process the users get to know about it away. This is really helpful because it means users do not have to wait to find out if a seat is available or not. The system tells them about seat availability changes soon as it happens.

4. SYSTEM IMPLEMENTATION

The implementation of VENUE focuses on several key functional modules.

A. User Authentication

The platform uses JSON Web Tokens for logging in. When a user logs in the server checks the users credentials. Makes a special token that is stored in a safe place in a cookie that can only be seen by the web server. This JSON Web Tokens token is then used to check who is making requests to the server. The platform keeps passwords safe by using methods to mix up the passwords like Argon2 so the users login details are still safe even if someone gets into the database. The platform uses methods, like Argon2 to keep passwords safe..

B. Booking Management

The booking system is made to stop problems, with seats by using a way of updating information in MongoDB. This special way is called atomic update operations. It is done inside MongoDB transactions. When someone wants to book a seat the server does these things in order:

1. Start a database transaction
2. Verify seat availability
3. Decrease available capacity
4. Create a booking record
5. Commit the transaction

If the seat is not available anymore the whole thing gets. The user gets a message saying that someone else has already booked the seat.

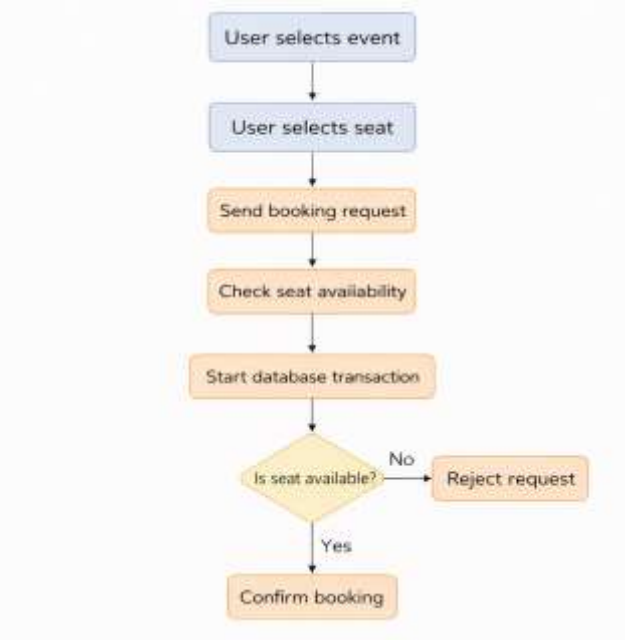

Fig. 2. Seat booking workflow in the VENUE system

C. Real-Time Seat Updates

The system uses a way of talking to computers that are connected to it this is called WebSocket communication, between the server and the computers that are connected to it. When someone books a seat the server tells all the computers that are looking at the same thing that something has changed. The server does this by sending out a message to all these computers.

All the computers that are looking at the thing get this message right away. They update the picture of the seats.

This helps to make sure that people do not see information about the seats. The system makes sure that people see the information, about the seats by doing this.

D. Event Organizer Module

The platform is really helpful for people who organize events. These event organizers can do a lot of things. They can make events and add pictures and videos to promote them. They can also decide how many seats are available and make a schedule. The platform makes sure that only the people who are supposed to do these things can actually do them. This is so that nobody can make changes without permission. The platform does this by letting certain people

do certain things, based on what their role is, as an event organizer.

5. RESULTS AND ANALYSIS

The VENUE system was checked to see if it works properly. We did this by testing it and trying to book seats at the time, with many users. This was done to see how the VENUE system behaves when a lot of people are trying to book seats at the time.

Fig. 3. Seat selection interface showing real-time seat availability

The results show that using operations is a good way to stop duplicate seat allocation. When two users try to reserve the seat the first one to do it successfully gets the seat and the second one does not. The second request does not go through.

The system also uses WebSockets to give users real-time updates. This means that when someone books a seat all the other users see the change away. This helps because users are less likely to try to select seats that are already taken.

The system works well when a lot of people are using it at the same time like when a small event is being booked. The transactional operations and WebSockets work together to make sure that the seat allocation process is safe and the user experience is good. The use of operations prevents duplicate seat allocation and the real-time updates make the system more user friendly.

6. CHALLENGES AND SOLUTIONS

One of the challenges we faced while building the VENUE platform was handling many seat booking requests at the same time.

- In real-life event booking systems lots of users might try to book the seat almost at the same time.
- If not handled properly this can cause bookings or weird database issues.
- To fix this we used MongoDB transactions to make sure each booking request is processed as one unit.
- If a seat becomes unavailable during the booking process the request gets safely rejected, preventing bookings of the same seat.
- Another challenge was keeping seat availability updates live for users on the platform.
- Traditional ways of updating data require users to refresh the page to see changes, which can lead to seat info.
- To solve this the VENUE platform uses WebSocket technology to let the server instantly tell all users about seat updates whenever a booking happens.
- This way users get the seat availability info without needing to refresh the page manually.
- We also had to make sure the platform had authentication and role-based access control.
- The VENUE platform supports types of users: customers, event organizers and administrators.
- Each type of user needs levels of access to platform features.
- We implemented JSON Web Token (JWT) based authentication to solve this.
- User roles are included in the token. Checked for each protected request.

- This approach stops access to sensitive operations like event creation, booking management and administrative controls.
- Lastly we had to make sure the user interface was smooth and responsive.

- Seat maps, event listings and booking confirmations had to update without slowing down the platform.
- We used React's component-based architecture, efficient state management and asynchronous API requests.
- This helped keep interactions even when users navigated fast between events or seat selections, on the VENUE platform.
- We made sure the VENUE platform could handle users and bookings at the same time.
- We focused on making the VENUE platform user-friendly and reliable.
- The VENUE platform now handles bookings and user interactions smoothly.

7. CONCLUSION

The VENUE platform is a system that helps people book event and movie tickets. It does this in time so people can see right away if the seats they want are available. The VENUE platform uses a kind of database that can handle a lot of transactions at the same time. It also uses something called WebSocket to update the user interface. The people who made the VENUE platform used web technologies like the MERN stack to build it. This means that the VENUE platform can handle a lot of people using it at the time without slowing down. The way the VENUE platform is designed also makes it easy to add features later on like working with multiple services at the same time handling payments and managing big events. There are some things that the people who made the VENUE platform might want to work on in the future. They might want to make the VENUE platform even better, at handling a lot of users by using something called microservices. They might also want to add a system that can handle a lot of requests at the time so the VENUE platform does not get overwhelmed when there are a lot of people trying to book tickets. The VENUE platform could also be improved by adding something that can predict how many tickets will be sold so the people running the events can plan ahead.

REFERENCES

[1] K. P. Birman, *Reliable Distributed Systems: Technologies, Web Services, and Applications*. Springer, 2019.

[2] MongoDB Inc., "MongoDB Transactions and Data Consistency," MongoDB Documentation, 2024.

[3] T. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[4] D. Johnson, *MERN Stack for Scalable Web Applications*. O'Reilly Media, 2022.

[5] React Documentation, "React Documentation and Best Practices," Meta Open Source, 2025.

[6] Socket.IO Documentation, "Real-Time Bidirectional Event-Based Communication," 2024.

[7] R. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," University of California, Irvine, Doctoral Dissertation, 2000.

[8] R. Ajmera and C. S. Lamba, "Exalting complexity measures for software testing accompanying multi agents," Int. J. of Emerging Trends & Technology in Computer Science, vol. 2, no. 4, Jul.–Aug. 2013, pp. 56–61.

[9] R. Ajmera and C. S. Lamba, "Multi agent framework for measuring software reliability," World Academy of Informatics and Management Sciences, vol. 1, no. 2, Apr.–May 2012, pp. 21–24.

[10] Sanu Kumar, Sara Kanwar, Sara Kumawat, Laxmikant Vashishtha, Yoganand Sharma, "Social Media Sentiment Analysis Using Machine Learning: Frameworks, Methods, and Challenges", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 206-210, 2025.

[11] Kritika Paliwal, Laxmikant Vashishtha, Ratna Ram Tanwar, "Big Data Analytics in Modern Engineering and Industry", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 326-330, 2025.

[12] Pankaj Jain, Shristi Arora, Santosh Kumar, "Companion App: A Flutter-Based Mobile

Application for Mental Health Tracking and Personalized Well-Being Support", International Journal of Global Research in Science and Technology (IJGRST), Vol. 10, pp. 136-152, 2025.

[13] R. Ajmera and D. Dharamdasani, "Comparative study of existing food delivery app," Global Research Journal, pp. 454–463, 2022.

[14] R. Ajmera, "Study and analysis of software design models using Symphony .NET tool," in Proc. 2nd World Conf. on SMART Trends in Systems, Security and Sustainability, IEEE, Oct. 2018.

[15] A. Kalwar and R. Ajmera, "ARQI: Model for developing web application," Int. J. on Technical and Physical Problems of Engineering (IJTPE), vol. 13, no. 47, pp. 7–13, Jun. 2021.

[16] R. Ajmera, A. Kalwar, and C. S. Lamba, "A modern study on progressions & issues of web applications development in small firms," Int. J. of Scientific Research in Science and Technology, vol. 3, no. 8, Nov.–Dec. 2017.