International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 SJIF 2023: 8.176 ISSN: 2582-3930

We Meet –Let's Connect with the world (instant messaging platform) using Full Stack Development

Priya Pandey [1],Anya Kumari [2],
Sweety Tomar[3],Riya Srivastava[4]
(Information Technology , NIET, Greater Noida Uttar Pradesh)[1,2,3,4]

Mr.Rajeev Kumar
Professor IT
NIET, Greater Noida
Uttar Pradesh

Abstract-"We Meet" is a chat application that aims to facilitate real-time Communication technology has revolutionized the way people connect, transcending local boundaries and facilitating the exchange of ideas. While this technology has been available for some time, its widespread adoption has only recently occurred. This research project presents an illustrative example of a chat server, comprising two applications: a client application running on a user's web browser, and a server application hosted on network servers [1] Users can engage in private and group conversations by connecting the client to a server. Late-stage safety measures were implemented to ensure secure were implemented to ensure secure communication.

Keywords-Chat-service, Full Stack Development, Networking Model, API,RDBMS, private chat, Group chat, React, java , javascript programming language, Online Communication

Abbreviations and Acronyms- Graphical User Interface(GUI), Hypertext Mark-up Language (HTML),Cascading Style Sheets (CSS), Application Program Interface (API), Transmission Control Protocol/Internet Protocol (TCP/IP), Hypertext Transfer Protocol (HTTP), Open System's Interconnection (OSI), Apache Tom-Cat 6.0, Windows, Cross Platform, International Organization for Standardization (ISO), Hypertext Transfer Protocol Secure (HTTPS), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Operating System (OS), File Transfer Protocol (FTP), Structured Query Language (SQL), Java Script, Extensible Mark-up Language (XML), Domain Naming System (DNS), Internet Protocol (IP), Uniform Resource Locator (URL), User Interface (UI), Development and Operation (DevOps), Relational Database Management System (RDMS).

I . INTRODUCTION

"We Meet - Let's Connect with the World" aims to create a platform where users can interact through full-stack web development. The platform will provide a web-based interface for instant messaging, allowing users to engage in one-on-one conversations using different devices. Instant messaging (IM) is a popular form of communication, primarily through chat, where individuals exchange

text-based messages via computers. IM has evolved with technology, incorporating features like Wireless Application Protocol (WAP) and Short Message Service (SMS) integration for handheld devices like mobile phones. Developers around the world strive to enhance user experience and streamline their workflow. To achieve these goals, they rely on frameworks like Spring Boot for quick and efficient web application development. These frameworks leverage pre-existing frameworks such as Spring, simplifying the development process and boosting productivity.

The frontend design of the platform includes features such as user sign-up with name, contact number, email, and password. Users can log in with their email and password, view all their chat rooms, create new chat rooms by adding other users' email addresses, and search for existing rooms by name or email. They can post messages in chat rooms, with notifications and indicators for unread messages. Additional stretch goals include displaying online status and enabling group chat.

On the backend, the User Management Service provides REST APIs for creating and retrieving user information. Optional features include email address verification using OTP and secure password storage using the SHA algorithm. REST APIs are also available for creating, retrieving, and deleting chat rooms, with Role-Based Access Control (RBAC) implemented to ensure authorized user access.

Please note that this text has been provided as a single paragraph without numbering.

1. II. RELATED WORK

There are numerous chat service web applications available today, offering instant messaging capabilities for users across various platforms. Popular chat services such as WhatsApp, Facebook Messenger, and Telegram have become integral parts of people's daily communication. While popular chat service apps like WhatsApp and Telegram have gained significant user bases, they are not without their limitations. These limitations present opportunities for the development of a new chat service app that addresses these issues. Some of the common problems encountered in existing apps are[8]:

Privacy Concerns [9]: There have been concerns about data privacy and security in popular chat apps. A new chat service app can prioritize strong end-to-end encryption, transparent data handling practices, and user-controlled privacy settings to address these concerns.

Advertisements and Monetization: Many chat apps rely on advertising as a revenue source, which can disrupt the user experience. A new chat service app could explore alternative monetization models, such as subscription-based plans or innovative partnerships, to reduce reliance on intrusive ads.

Platform Lock-In: [10] Some chat apps restrict interactions between users on different platforms or limit integration with other services. A new chat service app could prioritize cross-platform compatibility, allowing users to communicate seamlessly regardless of their device or preferred chat app.

User Interface Complexity: Existing chat apps may have complex user interfaces that can be overwhelming for some users. A new chat service app can focus on a clean and intuitive user interface, making it easy for users to navigate and communicate effortlessly.

Data Storage and Backups: Some chat apps have limitations on data storage or lack robust backup

options, leading to potential data loss. A new chat service app can offer ample storage capacity and automatic backup features to ensure users' conversations and media are securely preserved.

Integration with Other Services: While some chat apps offer integrations with popular services, there may be room for expansion and improvement. A new chat service app could provide seamless integration with a wider range of applications, enabling users to access various services within the chat interface.

Group Chat Management: [11] Managing large group chats can be challenging in existing apps, with limited options for organization and control. A new chat service app can introduce enhanced group management features, such as moderation tools, topic categorization, and advanced search capabilities.

By addressing these problems and offering innovative solutions, a new chat service app can carve its own niche in the market, attracting users who seek a better user experience, improved privacy measures, and more flexible functionalities.

Our paper aims to create a platform that addresses all the problems mentioned earlier by overcoming the challenges faced by related works like

Enhanced Security: Implementing robust encryption and security measures to protect user data and ensure privacy. This can include end-to-end encryption, two-factor authentication, and secure data storage practices.

Advanced Notification System: Developing a smart notification system that allows users to customize their preferences, including the ability to mute specific conversations, receive push notifications, or choose different alert sounds for different types of messages.

Rich Media Support: [12] Enabling the seamless sharing of various types of media within the chat, such as images, videos, documents, and audio files. This can enhance the user experience and facilitate efficient communication.

Customization Options: Offering users the ability to personalize their chat experience by allowing them to customize themes, colours, fonts, and other visual elements according to their preferences.

Collaboration Tools: Introducing collaboration features that enable users to work together in real-time, such as document sharing and editing, task management, and integrated video conferencing capabilities.

Smart Search Functionality: Implementing a powerful search feature that allows users to quickly find past conversations, specific messages, or shared files, making it easier to retrieve important information.

Integration with Third-Party Services: Providing seamless integration with popular services and platforms, such as calendars, project management tools, or social media platforms, allowing users to access and share information from within the chat app.

Cross-Platform Compatibility: Ensuring the chat application works smoothly across multiple devices and platforms, including web browsers, desktop applications, and mobile devices, enabling users to stay connected regardless of their preferred device.

By incorporating these additional benefits into the real-time chat web application, we can create a unique and improved user experience, offering features and functionalities.

III. PROBLEM STATEMENT

In today's rapidly evolving world, adaptability is crucial to staying ahead. Effective communication plays a vital role in the success of any business. It is imperative to be able to develop communication

features that align with changing circumstances, as they are instrumental in business growth.

A strong customer service team is essential for fostering positive relationships with customers. However, it is equally important to meet customers where they are. Demonstrating a commitment to improvement and adaptability can significantly impact customer perception. By incorporating new features and functionalities into communication channels, businesses can showcase their willingness to adjust and cater to the needs of potential customers.

Furthermore, chat apps facilitate seamless collaboration among employees. Teams can effortlessly exchange ideas, solve company problems, and work together towards shared objectives. In an era where people spend much of their time at computers, it is logical for conversations to take place within these digital environments. Chat apps integrate well with email communication, and their primary focus is on fostering collaboration. Such integration can lead to enhanced employee performance and productivity by providing a streamlined platform for effective teamwork.

By embracing chat apps and their diverse capabilities, businesses can harness the power of communication to drive customer satisfaction, adapt to changes, and foster an environment of collaboration among employees, ultimately contributing to overall business success.

IV. TECHNOLOGY USED

When creating our Web Application, we use HTML to structure its content and CSS to style and design its appearance. However, if we want to make the website dynamic and interactive, we require server-side technology. A web server is responsible for handling client requests and responding to them. It processes these requests using specific protocols. Its primary function is to store and serve web pages to clients. Essentially, the web server acts as a mediator between the client and the server. Apache is a well-known example of a web server.HTML, which stands for Hypertext Mark-up Language, serves as a common language for communication between the web server and the web client. As both the web server and web client are separate software components, there needs to be a language that facilitates their communication. HTTP, short for Hypertext Transfer Protocol, is the communication protocol utilized between the client and the server. It operates on top of the TCP/IP protocol.

In summary, while HTML and CSS are crucial for creating the structure and design of a website, server-side technology, including web servers, HTTP, and server-side scripting languages, is required to make the website dynamic and enable communication between the client and the server.

Servlet

A Servlet [13] is a Java program that runs within a web server; it receives requests and responds to them using related protocols, typically HTTP. Servlets are versatile and capable of handling various types of requests, making them a common choice for functional application development.

React

React [14] is a popular library for building user interfaces in JavaScript. It was created by Facebook and is supported by a community of open-source developers. Although it is not a programming language itself, React is widely used in web development. It was introduced in May 2013 and has since become one of the most commonly used frontend libraries. React provides various features and extensions, such as Flux and React Native, that go beyond just creating user interfaces.

One of the advantages of using React is that it simplifies the creation of dynamic web applications.

Compared to plain JavaScript, React requires less code and offers more functionality, making it easier to build complex applications. Additionally, React improves performance by utilizing a technique called Virtual DOM. It compares the previous and current states of components and updates only the parts of the real DOM that have changed, resulting in faster application rendering.

Another benefit of React is its emphasis on reusable components. Components are the fundamental building blocks of React applications, and they can be reused throughout the application. This reusability significantly reduces development time by eliminating the need to write the same code multiple times.

React also follows a unidirectional data flow, where data is passed from parent components to child components. This makes it easier to debug errors and identify the source of problems in an application since the data flow is predictable and traceable.

Overall, React is a powerful library that simplifies UI development, improves performance, promotes code reuse, and provides a clear data flow structure for building web applications.

Node.js

Node.js [15] is a runtime environment for JavaScript that is open-source and works across different platforms. It is widely used in various types of projects. Instead of running within a web browser, Node.js runs the V8 JavaScript engine, which is the core of Google Chrome, outside of the browser environment. This design allows Node.js to be highly efficient and performant. Unlike traditional web servers that create a new thread for each request, Node.js operates in a single process. It provides a set of built-in asynchronous I/O functions that prevent JavaScript code from blocking and prioritize non-blocking programming patterns. This means that Node.js can efficiently handle tasks like network operations, database access, and file system operations without wasting CPU cycles. When an I/O operation is performed, Node.js doesn't block the thread but instead continues executing other tasks. It resumes the operation once the response is ready. This unique feature enables Node.js to handle a large number of concurrent connections without the complexities of managing thread concurrency. One of the significant advantages of Node.js is that it allows frontend developers who are already familiar with JavaScript to write server-side code without having to learn a completely different language. Node.js supports the latest when running Node.js.

Spring Boot

Spring [16] is widely recognized for its ability to create scalable applications. In the realm of web applications, Spring offers Spring MVC, a highly popular module known for building scalable web applications. However, a notable drawback of Spring projects is the time-consuming and somewhat overwhelming configuration process, particularly for new developers. Making an application production-ready can be challenging, especially for those new to Spring. Fortunately, Spring Boot emerges as a solution to this challenge. Spring Boot is built on top of Spring and encompasses all its features. It has gained favour among developers due to its rapid and production-ready environment, allowing developers to focus directly on logic implementation instead of struggling with complex configurations and setups.

Spring Boot, being a microservice-based framework, significantly reduces the time required to create a production-ready application. It streamlines the development process, minimizing the effort needed for configuration. Prior knowledge of the Spring framework is a prerequisite for working with Spring Boot.

My SQL

MySQL [17] is a widely-used open-source relational database management system InnoDB being the default choice since version 5.5, providing transactional support and referential integrity.

In addition to its core features, MySQL offers advanced functionality such as indexing, allowing for efficient data retrieval, query optimization, and performance improvement. It supports triggers, which are automatic actions executed in response to specific events, views for virtual table creation, and stored procedures for encapsulating reusable database logic.

MySQL provides robust security mechanisms, including user authentication and access control, as well as encryption capabilities for protecting sensitive data. It supports replication, enabling data redundancy and high availability through various replication topologies. (RDBMS) known for its speed, reliability, and scalability. It supports multiple operating systems and follows a client-server architecture, allowing multiple clients to connect to a single MySQL server. MySQL is ACID-compliant, ensuring data integrity by supporting atomicity, consistency, isolation, and durability. It offers a range of storage engines, with

MySQL integrates seamlessly with popular programming languages and frameworks, making it a preferred choice for web applications and content management systems. Its compatibility with languages like PHP, Python, Java, and .NET simplifies database interactions.

MySQL offers tools such as MySQL Workbench for database design, administration, and query development. It has a thriving community that provides extensive documentation, support, and regular updates. With its versatility, performance, and broad range of features, MySQL continues to be a leading database management system used by major websites, applications, and enterprises worldwide

Kafka

Kafka [18]is a distributed streaming platform developed by the Apache Software Foundation. It is designed to handle high-throughput, fault-tolerant, and real-time data streaming. Kafka provides a messaging system that allows data to be published, subscribed to, and processed in a scalable and reliable manner

At its core, Kafka follows a publish-subscribe model, where producers write data to topics, and consumers read from those topics. It acts as a distributed commit log, meaning it stores all messages persistently on disk, allowing for fault tolerance and durability. The messages in Kafka are organized into partitions, which are spread across multiple servers in a cluster, ensuring scalability and parallel processing.

One of Kafka's key strengths is its ability to handle large volumes of data in real-time. It achieves this by employing a distributed architecture that allows data to be processed in parallel across multiple servers. This makes it an ideal choice for use cases such as data streaming pipelines, real-time analytics, log aggregation, and event-driven architectures.

Kafka is known for its high performance and low-latency characteristics. It achieves this by implementing efficient disk and OS-level I/O operations, as well as utilizing a zero-copy messaging design. It can handle millions of messages per second, making it suitable for high-throughput applications.

Additionally, Kafka provides fault tolerance through replication. Each partition can have multiple replicas, with one serving as the leader and the others as followers. If a leader replica fails, one of the followers is automatically promoted as the new leader, ensuring continuous operation without data loss.

Websocket

WebSockets[19] is a communication protocol that enables full-duplex, real-time communication between a client and a server over a single, long-lived connection. Unlike traditional HTTP, which follows a request-response model, WebSockets allow for bidirectional communication, facilitating data exchange between the client and the server in real time. This persistent connection eliminates the need for repeated HTTP requests, reducing overhead and latency associated with establishing new connections for every communication.

WebSockets are highly beneficial for applications that require instant data updates and interactive features. They are extensively used in real-time chat applications, collaborative tools, financial trading platforms, online gaming, and live streaming applications. WebSockets provide a more efficient and responsive solution compared to other techniques like long polling or AJAX.

By establishing a persistent connection, WebSockets enable real-time data push from the server to the client without the need for client-initiated requests. This allows for immediate updates and notifications, enhancing user experience and enabling seamless, dynamic content updates.

V. PROPOSED METHODOLOGY

The proposed application aims to eliminate the need for a centralized system, such as the one used by Skype, for registration, login, and buddy lists. In Skype, user profiles are stored in a centralized database, allowing users to log in from anywhere using their credentials. While this centralized approach offers some level of robustness, concerns arise regarding the security of the centralized database and its susceptibility to attacks.

A recent study focused on decentralizing the buddy list functionality of a chat application. The study suggested the development of a robust index system using a distributed hash table for a decentralized chat application. This index system would store the IP addresses and ports of all users who join the chat. Users would establish their own buddy lists by connecting to a centralized indexing system after logging in.

When user A wants to communicate with user B, user B would act as a server and authenticate user A. However, since the authentication process is one-way, it creates an opportunity for attackers to impersonate user B. To address some of these issues, our proposed application incorporates the following ideas:

Decentralization: The application will avoid relying on a central server for registration, login, and buddy lists, distributing these functions among the users themselves.

Enhanced Security: We prioritize implementing robust authentication Users would establish their own buddy lists by connecting to a centralized indexing system after logging in.

CLIENT

A client refers to a software application or a system that sends requests to another application running on a dedicated machine called a server. The communication between the client and the server does not necessarily have to be through wired connections; wireless communication is also possible. The client, which is connected to a network, can initiate a request and send it to the server for processing and response.

Chat application or Client Side

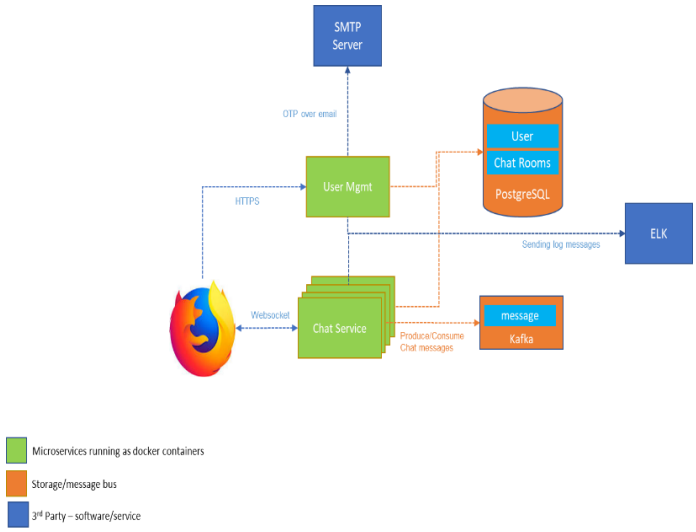
The chat application [20] is a crucial component of the chat architecture, serving as the user-facing interface. It consists of two main root components:

Chat Client Engine: This component is responsible for managing all communication with the Chat Server Engine. It utilizes internal components, namely the Chat REST API Client Library and the Chat Web Socket Client Library, to facilitate communication and data exchange between the client and the server. The Chat Client Engine handles the underlying mechanisms required to establish and maintain a connection with the server.

Chat UI: The Chat UI component is responsible for presenting data and information to the users. It includes two main sub-components:

Chat Contact List UI: This component displays the list of contacts or users that the current user can interact with. It provides an organized view of available contacts, allowing users to select and initiate conversations with specific individuals.

Chat Dialog UI: This component is responsible for rendering the actual conversation or dialog between users. It presents the messages, notifications, and other relevant information exchanged during the chat session.



VI. CONCLUSION

The current focus of the project is on text communication within the chat application. However, there is always room for improvement and expansion. Many existing chat apps that serve a similar purpose tend to have complex and confusing interfaces, making them difficult to use. It is crucial to make a positive first impression in both human relationships and human-computer interaction. Therefore, the goal of this project is to develop a chat service Web app with a high-quality user interface that is intuitive and user-friendly.

In the future, the application aims to incorporate additional features to enhance the user experience. These features may include:

File Transfer: Allowing users to share files with each other through the chat application.

Voice Message: Enabling users to send and receive voice messages, adding a more dynamic form of communication.

Video Message: Allowing users to exchange video messages, further enriching the communication experience.

Audio Call: Introducing the capability for users to have voice calls with each other directly within the app.

Video Call: Enabling users to have video calls, facilitating real-time face-to-face communication.

Group Call: Extending the functionality to support group calls, allowing multiple users to participate in a conversation simultaneously.

By incorporating these features, the project aims to provide a comprehensive and versatile chat service that goes beyond text communication, ultimately enhancing the overall user

experience and making it a preferred choice for users seeking a high-quality and user-friendly chat application.

VII. REFERENCES

https://www.ijraset.com/research-paper/development-of-chat-application [1]

https://catalogimages.wiley.com/images/db/pdf/0764549537.01.pdf [2]

https://www.academia.edu/40977586/WEB_BASED_CHAT_APPLICATION_MINOR_PROJECT_REPORT_COMPUTER_SCIENCE_and_ENGINEERING [3]

https://dev.to/jeremyling/how-to-build-a-loginsignup-form-with-validation-in-2-minutes-in-react-1a27 [4]

https://www.callicoder.com/spring-boot-websocket-chat-example/ **[5]**

https://dev.to/subhransu/realtime-chat-app-using-kafka-springboot-reactjs-and-websockets-lc [6]

https://www.researchgate.net/publication/360483603_Research_paper_on_Group_chatting_Application [7]

https://www.infoq.com/articles/challenges-realtime-chat-service-pusher/#:~:text=Despite%20the%20progress%20made%20in%20realtime%20systems%20design%2C,pushing%20updates%20and%20persisting%20changes%20to%20separate%20subsystems. [8]

https://onlinelibrary.wiley.com/doi/full/10.1002/cpe.6426 [9]

https://en.wikipedia.org/?title=Platform_lock-in&redirect=no [10]

(PDF) Research paper on Group chatting Application (researchgate.net) [11]

https://support.google.com/richmedia/answer/2417545 [12]

Learn Servlet Tutorial – javatpoint [13]

The Best Guide to Know What Is React [Updated] (simplilearn.com) [14]

Introduction to Node.js (nodejs.dev) [15]

What is Java Spring Boot? —Intro to Spring Boot | Microsoft Azure [16]

https://www.mysql.com/[17]

https://www.confluent.io/lp/apache-kafka/?utm_medium=sem&utm_source=google&utm_campaign=ch.sem_br.nonbrand_tp.prs_tgt.kafka_mt.xct_rgn.apac_lng.eng_dv.all_con.kafka-stream&utm_term=kafka%20streams&creative=&device=c&placement=&gad=1&gclid=CjwKCAjw1MajBhAcEiwAagW9MTXnMtvO1zbmUXTzve3LL

MGAewkI3ihPslZzoTRGsRKLciyvgU_7BBoCHqwQAvD_BwE[18]

https://en.wikipedia.org/wiki/WebSocket#:~:text=WebSocket%20is%20a%20computer%20communications,protocol%20is%20known%20as%20WebSockets.[19]

https://www.researchgate.net/publication/360483603_Research_paper_on_Group_chatting_Application [20]