

CHATROOM APPLICATION

Guide: Rukmini

B.Chetana, K.Charan, I.Charan, Charan Teja, Chandrakanth, Chandra Sai Teja, Rukmini

School of Engineering

Malla Reddy University

Abstract: This project aims to create a simple chat application using Python and Django that allows users to create chat rooms and chat in real-time. The app will use Django Channels to enable real-time communication between the server and the client. The app will not store all previous chat records, only displaying a few recent chats, and older chats will be deleted. The app will use WebSocket technology to enable real-time messaging, and users will be able to join chat rooms and send messages. The application will also include a chat room creation feature and user authentication to secure the messaging. Django and Channels are great tools for building realtime applications with Python. Here are some general steps to follow to get started: Set up a new Django project, Install and configure Django Channels, Create a Chat Room model, Create a Message model, Create views and templates, Implement WebSocket communication, Handle message storage, Test and deploy. Overall, this project will enable users to have a seamless chat experience while maintaining their privacy and security.

Keywords:- Python, Django, Chats, WebSocket, Privacy.

I. INTRODUCTION

In this project, we will develop a simple chat application using Python and Django that allows users to create chat rooms and chat with each other in real-time.

The app will also have the feature of displaying recent chat history, but older messages will be deleted to optimize the performance of the application.

The chat app will have an user interface and will be easy to use. Users will be able to register themselves using their email and password and will be able to login to their account.

They will then be able to create a new chat room or join an existing one. Once they have joined a chat room, they will be able to chat with other users in real-time.

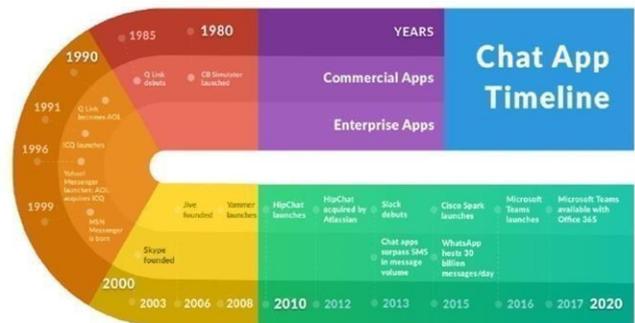


Fig. 1. Chat App Timeline

II. LITERATURE REVIEW

In 2020, Jhalak Mittal, "Arushi Garg, Shivani Sharma" and "Online Chat Request" was published by the International Journal of Research in Engineering, IT and Social Sciences, ISSN 2250-0588. This research paper contains the information we have provided to maintain the security and protection of the request for a speech. We have identified many requirements for secure speech and make it more realistic by using modern day techniques and weight to provide speed and good assurance to its customers. XSalsa20 calculator is ideal for mobile phones due to its high security, high performance and battery life. Customers can be sure that no one can read their messages, even if the cell phone gets in the wrong hands, you can't access the app and you can't access local information. In 2020, R. Gayathri, C. Kalieswari and published by the International Journal of Engineering and Advanced Technology (IJEAT). This research paper contains that the chat app provides a better and more flexible program. for discussion. Developed with the latest technology in the way of providing a reliable system. The main, added security, group chat, etc. This app can find the best demand in the market for most organizations that aim to have private applications. Additional features will also be added to the program based on community needs that include conference call, video chat. Location sharing, etc. based on need.

III. PROBLEM STATEMENT

To keep pace with this fast-moving world, we need to be adaptable to changing circumstances. Communication is one of the most important aspects of any business.

And being able to develop these features is essential to growing a business. Having a good customer service team is essential to improving business relationships with the customer. But it also helps to reach out to your clients where you are. Showing that you are also improving can be important to your customers. This feature indicates that you are willing to adjust to a variety of changes for potential customers.

In addition, chat apps allow employees to interact and work together. Teams of employees can share ideas and solve company problems with a single click. At a time when people are sitting at computers all day, it makes sense that conversations take place there. The chat app works very well in email communication. And the main goal of a chat request is cooperation. This means that its integration can lead to better employee performance and productivity.

IV. REQUIRED TOOLS

- Python 3.x(Jupyter)
- Django 2.x(models, extensions)
- Web Server (Apache, Nginx)
- Database (MySQL, PostgreSQL)
- JavaScript Libraries (jQuery, React)

V. METHODOLOGY

In a chat system, clients can be mobile applications or web applications. Clients do not communicate directly. Instead, each client connects to a chat service, which supports all of the features mentioned above. Let's focus on the basic functionality.

The chat service should support the following activities:

- 1) Get messages from other clients.
- 2) Find the right recipients of each message and pass the message on to the recipients.
- 3) If the recipient is offline, hold the recipient's messages on the server until they are online.

If the client intends to start a conversation, it connects to the chat service using one or more network protocols. For chat service, network protocol selection is important.

Applications are started by the client in most client / server applications. The same is true for the sender of a chat request.



When the sender sends a message to the recipient via chat service, he is using a time-tested HTTP protocol, which is the standard web protocol.

In this case, the client opens the HTTP connection with the chat service and sends the message, notifying the

service to send the message to the recipient. Keeping alive works well in this regard because the topic of keep alive allows the client to maintain ongoing communication with the chat service. It also reduces the amount of TCP handshake. HTTP is a good option on the sender's side, and many popular chat apps such as Facebook used HTTP in the beginning to send messages. However, the recipient side is much more complex.

Since HTTP is client-initiated, it is no small feat to send messages from a server. Over the years, many techniques have been used to mimic server-initiated communication: voting, long voting, and WebSocket

VI. EXPERIMENTAL RESULTS

For a chatroom application can vary based on factors such as the implementation approach, technology stack, scalability, user load, and user engagement. Some common metrics to evaluate the performance and effectiveness of a chatroom application include:

Response time: Measure the time it takes for a message to be delivered and displayed in the chatroom after being sent by a user.

Latency: Evaluate the delay between sending a message and receiving it on the recipient's side.

Concurrent user handling: Determine how well the chatroom application handles multiple users sending and receiving messages simultaneously without significant performance degradation.

Scalability: Test the ability of the chatroom application to handle increasing user loads without sacrificing performance or experiencing system failures.

User engagement: Assess user satisfaction, activity levels, and frequency of interactions within the chatroom application. This can include metrics such as the number of messages sent, active users, and user retention rates.

Reliability and stability: Evaluate the stability and reliability of the chatroom application by monitoring system uptime.

FUTURE WORK

Other enhancements will be involved

- 1) A place of safety
- 2) Video call
- 3) large size
- 4) Conference call
- 5) Voice recording will be added
- 6) Improving different text style and font size.

REFERENCES

- [1] Django-documentation-Channels:
<https://channels.readthedocs.io/en/stable/>
- [2] Django-documentation-Web-Sockets:
<https://docs.djangoproject.com/en/3.2/topic/websockets/>
- [3] Django-documentation-Django-ORM:
<https://docs.djangoproject.com/en/3.2/topics/db/>
- [4] Python-documentation-socket-programming:
<https://docs.python.org/3/howto/sockets.html>
- [5] Real-time chat app tutorial using Django Channels:
<https://medium.com/@devdilip/building-a-real-timechat-app-with-django-channels-and-react-3c2c6c1f16fe>

These references can provide you with more information and examples on how to implement the different features of your chat app using Python and Django.

VII. CONCLUSION

In conclusion, we have successfully developed a simple chat app using Python and Django. The chat app allows users to create chat rooms and chat in real-time with other users who are in the same room. We have implemented a WebSocket server using Django Channels to handle real time messaging, which makes the app more responsive and interactive for users.

We have also implemented a simple data storage solution where older chats are deleted and only a few recent chats are displayed. This ensures that the app is lightweight and doesn't take up unnecessary server space.

Overall, this project has provided us with valuable experience in working with Python and Django to create a real-world application. It has helped us to better understand web development concepts such as WebSocket servers, real-time messaging, and data storage.

Chatroom applications provide several benefits, including:

Instant Communication: Chatroom applications enable instant communication, allowing users to exchange messages and receive responses in real-time. This enhances the efficiency of communication and fosters quick decision-making and problem-solving.

Collaboration and Teamwork: Chatrooms serve as virtual spaces for teams or groups to collaborate and work together. Users can share ideas, coordinate tasks, and provide updates, leading to improved teamwork and productivity.

Accessibility and Convenience: Chatroom applications are often accessible across various devices and platforms, making it easy for users to join discussions and participate

from anywhere, anytime. This accessibility enhances flexibility and convenience in communication.

Multimodal Communication: Chatroom applications often support various communication modes, including text, multimedia files, and sometimes audio or video calls. This versatility allows users to choose the most appropriate form of communication for their needs.

Organization and History: Chatrooms typically archive conversations, providing a history of discussions and allowing users to refer back to previous messages. This feature enables easy retrieval of information and ensures important details are not lost.

Integration and Extensibility: Chatroom applications

often offer integration with other tools and services, such as project management platforms, file sharing systems, or thirdparty applications. This integration enhances workflow efficiency and centralizes information within the chatroom environment.

However, it is important to address potential challenges, such as ensuring data privacy and security, managing message overload in busy chatrooms, and maintaining a positive and inclusive communication environment.

In conclusion, chatroom applications play a crucial role in fostering real-time communication, collaboration, and information sharing. With continuous advancements in technology and user expectations, chatroom applications are likely to evolve further, incorporating features that enhance user experience, integration capabilities, and overall productivity in various personal and professional settings.

