#contacts
[
  {
    _id: new ObjectId('66e48153ebd8eef9f2ae137a'),
    lastMessageTime: 2024-10-19T18:58:49.859Z,
    email: 'aditya',
    firstName: 'Aditya',
    lastName: 'Shinde',
    image: null,
    color: 2
  },
  {
    _id: new ObjectId('66fe43a06fd49e3352d57f95'),
    lastMessageTime: 2024-10-19T05:52:35.543Z,
    email: 'tilakRvermaPilot@gmail.com',
    firstName: 'tilak',
    lastName: 'verma',
    image:
'upload/profiles/1727939515103wp9052004-hd-4k-art-wallpapers.jpg'
,
    color: 3
  },

]

#each contact after selecting from the list

each contact: {_id: '66fe43a06fd49e3352d57f95',
lastMessageTime: '2024-10-19T05:52:35.543Z',
 email: 'tilakRvermaPilot@gmail.com',
firstName: 'tilak',
lastName: 'verma', …}

##

```
##########################################################
#########################

#sending message


   1)const handleSendMessage = async () => {


    if(selectedChatType ==="contact" ){
      socket.emit("sendMessage",{
        sender:userInfo.id,
        content:message,
        recipient:selectedChatData._id,
        messageType:"text",
        fileUrl:undefined,

       })

    }


   };

@@@ socket.js(backend)
22222)  socket.on("sendMessage",sendMessage );

const sendMessage =async(message)=>{
       const senderSocketId = userSocketMap.get(message.sender);
       const recipientSocketId =
userSocketMap.get(message.recipient);
         // Create the message
       const createdMessage = await Message.create(message);

         // Finding the created message by its ID and populate sender
and recipient details
       const messageData =await
```

```
Message.findById(createdMessage._id)
        .populate("sender","id email firstName lastName image color")
        .populate("recipient","id email firstName lastName image color")
        .exec();

        if(recipientSocketId){

io.to(recipientSocketId).emit("recieveMessage",messageData);
        }

        //confirmation message to sender
        if(senderSocketId){
            io.to(senderSocketId).emit("recieveMessage",messageData);
        }

    }
```

|||||||||||||||||| data generated by populate function (messageData)

```
{
  "_id": "64abc12345...",
  "message": "Hello!",

"sender": {
    "id": "64f123abc...",
    "email": "sender@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "image": "profile.jpg",
    "color": "#ff0000"
  },

"recipient": {
    "id": "64f456def...",
    "email": "recipient@example.com",
    "firstName": "Jane",
    "lastName": "Smith",
    "image": "profile2.jpg",
```

```
   "color": "#00ff00"
  }
}
```

@@@@socketContext.jsx

```
socket.current.on("recieveMessage",handleRecieveMessage);

 const handleRecieveMessage = (message) => {
              const { selectedChatData, selectedChatType, addMessage
} = UseAppStore.getState();

          if (
                selectedChatType !== undefined &&
(selectedChatData._id === message.sender._id ||
selectedChatData._id === message.recipient._id)
              ) {

                console.log("message received")
                addMessage(message); // change state of
selectedchatmeeesage
              }
          }
```

@@@ store

```
 addMessage: (message) => {
       const selectedChatMessages = get().selectedChatMessages;
       const selectedChatType = get().selectedChatType;


       set({
          selectedChatMessages: [
             ...selectedChatMessages,

             {
```

```
            ...message,
            recipient:
              selectedChatType == "channel"
                ? message.recipient
                : message.recipient._id,
            sender:
              selectedChatType == "channel"
                ? message.recipient
                : message.recipient._id,
          },
        ],


useEffect(() => {

  if (selectedChatData._id) {

    const getMessages = async () => {

      const response = await apiCLient.post(
        GET_ALL_MESSAGES_ROUTES,
        {
          id: selectedChatData._id,
        },
        { withCredentials: true }
      );


      if (response.data.messages) {

        setSelectedChatMessages(response.data.messages);
      }
    };


    if (selectedChatType === "contact") getMessages();
  }
```

```
  }, [selectedChatType, selectedChatData, setSelectedChatMessages])



 const renderMessages = () => {
    let lastDate = null;
    return selectedChatMessages.map((message, index) => {
      const messageDate =
moment(message.timestamp).format("YYYY-MM-DD");
      const showDate = messageDate !== lastDate;
      lastDate = messageDate;

      return (
        <div key={index} className="">
          {showDate && (
            <div className="text-center text-gray-500 my-2">
              {moment(message.timestamp).format("LL")}
            </div>
          )}
          {selectedChatType === "contact" &&
renderPersonalMessages(message)}
          {/* {selectedChatType === "channel" &&
renderChannelMessages(message)} */}
        </div>
      );
    });
  };


      });
    }
});
```