ДОДАТОК А

АЛГОРИТМ НАДСИЛАННЯ ЗОБРАЖЕННЯ

```
const handleFileUpload = useCallback(
  async (file: File) => {
    if (!socket || !selectedUser) {
      return
    }

    const url = URL.createObjectURL(file)

    const socketFile = await formatFileToSocketFile(file)
    const fr = new FileReader()

    fr.onload = function () {
      const img = new Image()

      img.src = url

      img.onload = function () {
        let width = img.width
        let height = img.height
        const ratio = img.width / img.height

        if (width > 300) {
          width = 300
          height = width / ratio
        }

        if (height > 300) {
          width = width * ratio
          height = 300
        }

        const message: Message = {
          id: new Date().toISOString(),
          isViewed: false,
          senderId: user.googleId,
          timestamp: new Date().toISOString(),
          type: 'file',
          url: url,
          contentType: MAP_MIME_TO_TYPE [socketFile.mimetype],
```

```
          isPending: true,
          width,
          height,
        }
        setMessages((state) =>  [message, ...state])
        socket.emit('send_file', {
          userId: selectedUser.googleId,
          file: socketFile,
          identity: message.id,
          width,
          height,
        })
      }
    }

    fr.readAsDataURL(file)
  },
   [selectedUser, socket, user],
)
```

ДОДАТОК Б

АЛГОРИТМ ПЕРЕДАЧІ ПОВІДОМЛЕННЯ МІЖ КОРИСТУВАЧАМИ

```
private async sendMessage(
  senderId: string,
  recepientId: string,
  message: Message,
) {
  const selectedUser =
    await this.userService.findByGoogleId(recepientId)
  const currentUser = await this.userService.findByGoogleId(senderId)

  const userChat = selectedUser.chats.find(({ chatId }) =>
    currentUser.chats.some((chat) => chat.chatId === chatId),
  )

  if (userChat) {
    await this.chatService.createMessage(userChat.chatId, message)
    await this.userService.updateChat(recepientId, userChat.chatId, {
      type: 'new',
      payload: { message },
    })
    await this.userService.updateChat(senderId, userChat.chatId, {
      type: 'new',
      payload: { message },
    })
  } else {
    const chat = await this.chatService.create()
    await this.userService.createChat(recepientId, chat.chatId)
    await this.userService.createChat(senderId, chat.chatId)
    await this.chatService.createMessage(chat.chatId, message)

    await this.userService.updateChat(recepientId, chat.chatId, {
      type: 'new',
      payload: { message },
    })
    await this.userService.updateChat(senderId, chat.chatId, {
      type: 'new',
      payload: { message },
    })
  }
}
```

```
@SubscribeMessage('send_message')
async handleMessage(
  client: Socket,
  payload: { userId: string; text: string; identity: string },
) {
  const { userId, text, identity } = payload
  const connectedUserId = client.handshake.query ['id'] as string

  const message: Message = {
    id: randomUUID(),
    senderId: connectedUserId,
    timestamp: new Date().toISOString(),
    isViewed: false,
    type: 'text',
    text,
  }
  console.log('sender:', connectedUserId)
  console.log('rec:', userId)
  await this.sendMessage(connectedUserId, userId, message)

  client.to(userId).emit('new_message', message)
  client.nsp.to(connectedUserId).emit('sended_message', {
    message, identity
  })
}
```

ДОДАТОК В

АЛГОРИТМ ЗБЕРІГАННЯ ФАЙЛУ У ХМАРНОМУ СХОВИЩІ

```
@Injectable()
export class S3Service {
 private readonly AWS_S3_BUCKET = process.env.AWS_S3_BUCKET
 private readonly s3 = new AWS.S3({
   accessKeyId: process.env.AWS_ACCESS_KEY_ID,
   secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
 })

 async uploadFile(name: string, buffer: Buffer, mimetype: string) {
   const params = {
     Bucket: this.AWS_S3_BUCKET,
     Key: String(name),
     Body: buffer,
     ACL: 'public-read',
     ContentType: mimetype,
     ContentDisposition: 'inline',
     CreateBucketConfiguration: {
       LocationConstraint: 'ap-south-1',
     },
   }

   try {
     const s3Response = await this.s3.upload(params).promise()

     return s3Response.Location
   } catch (e) {
     console.log(e)
   }
 }
}
```