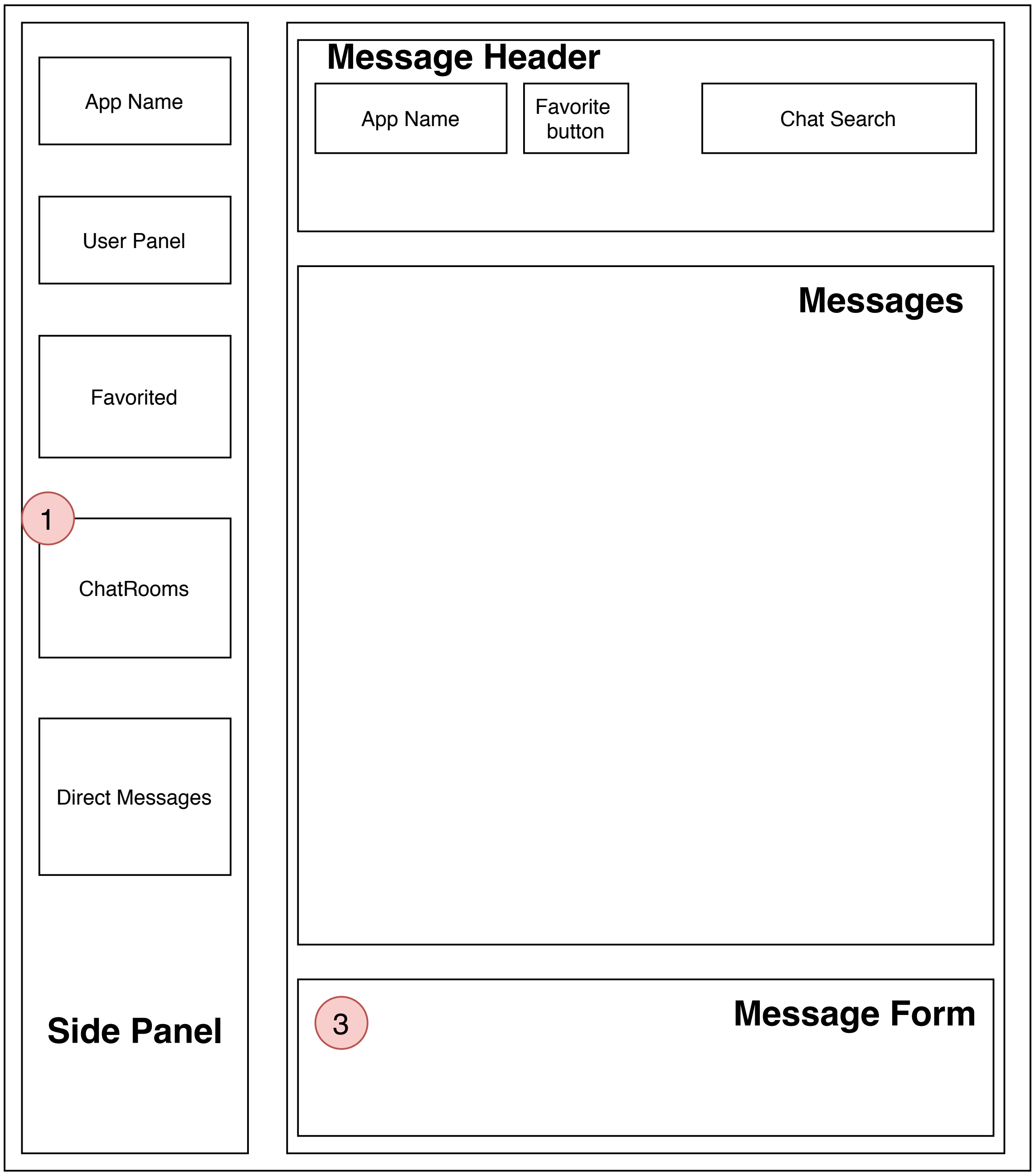
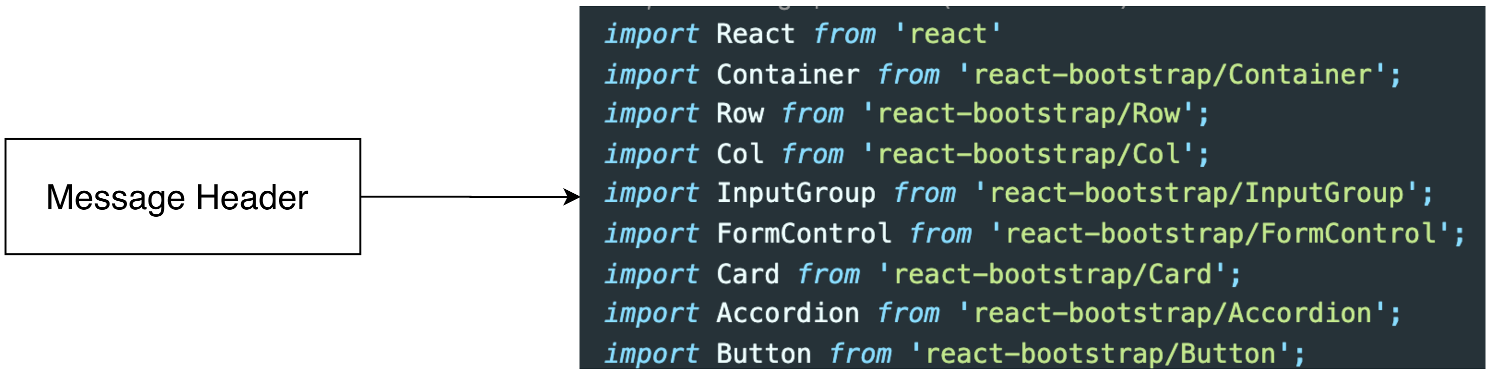
메시지 부분 UI

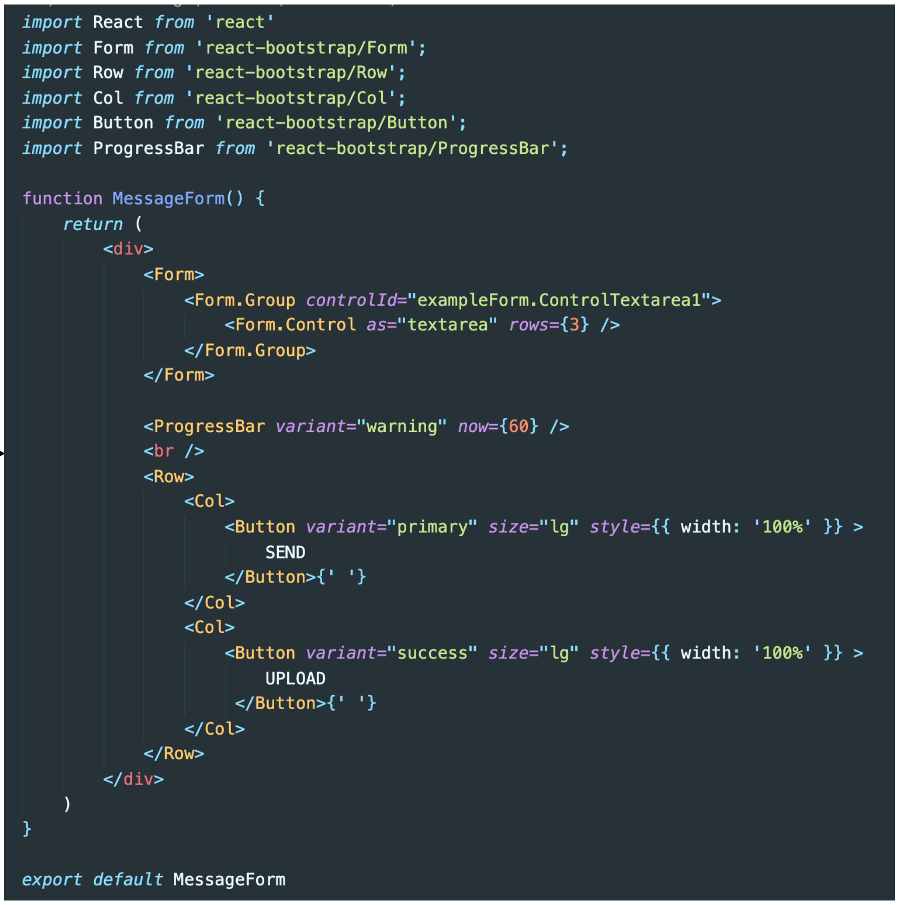




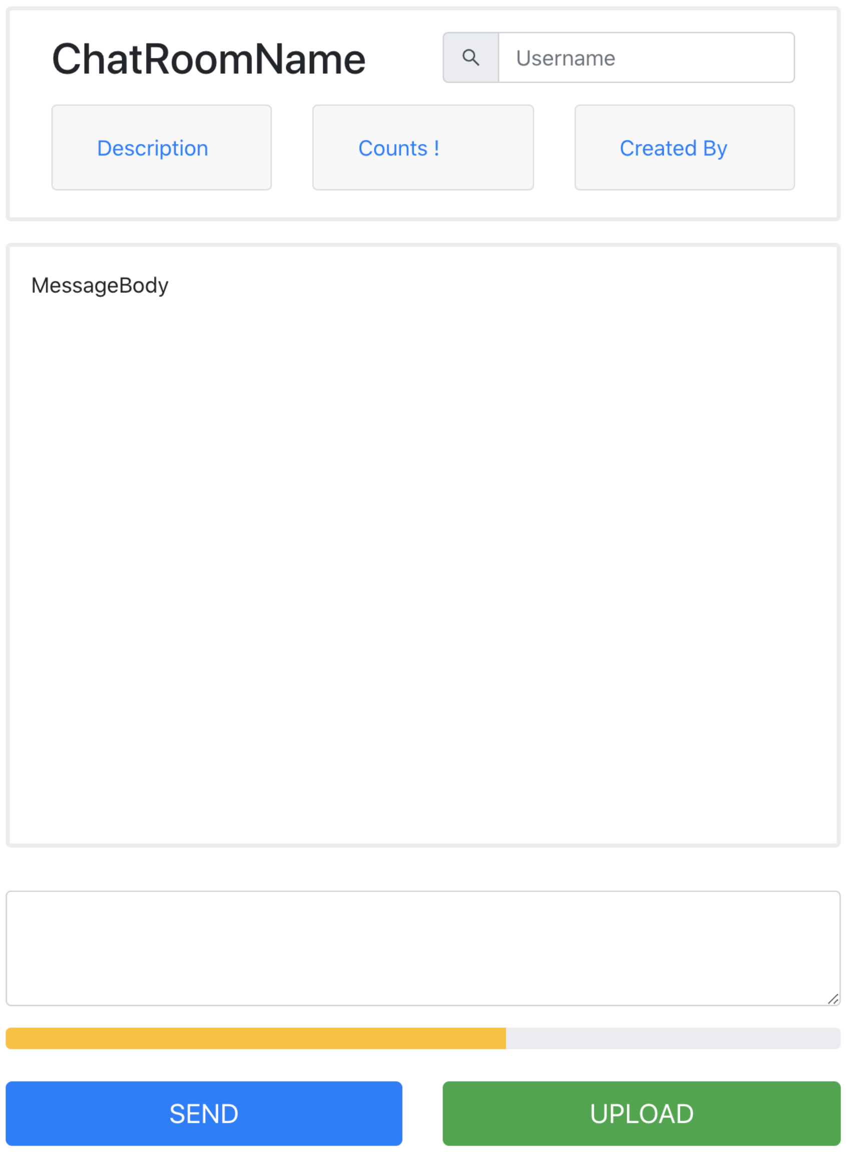
```
import React from 'react'
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import InputGroup from 'react-bootstrap/InputGroup';
import FormControl from 'react-bootstrap/FormControl';
import Card from 'react-bootstrap/Card';
import Accordion from 'react-bootstrap/Accordion';
import Button from 'react-bootstrap/Button';
```

Message Body

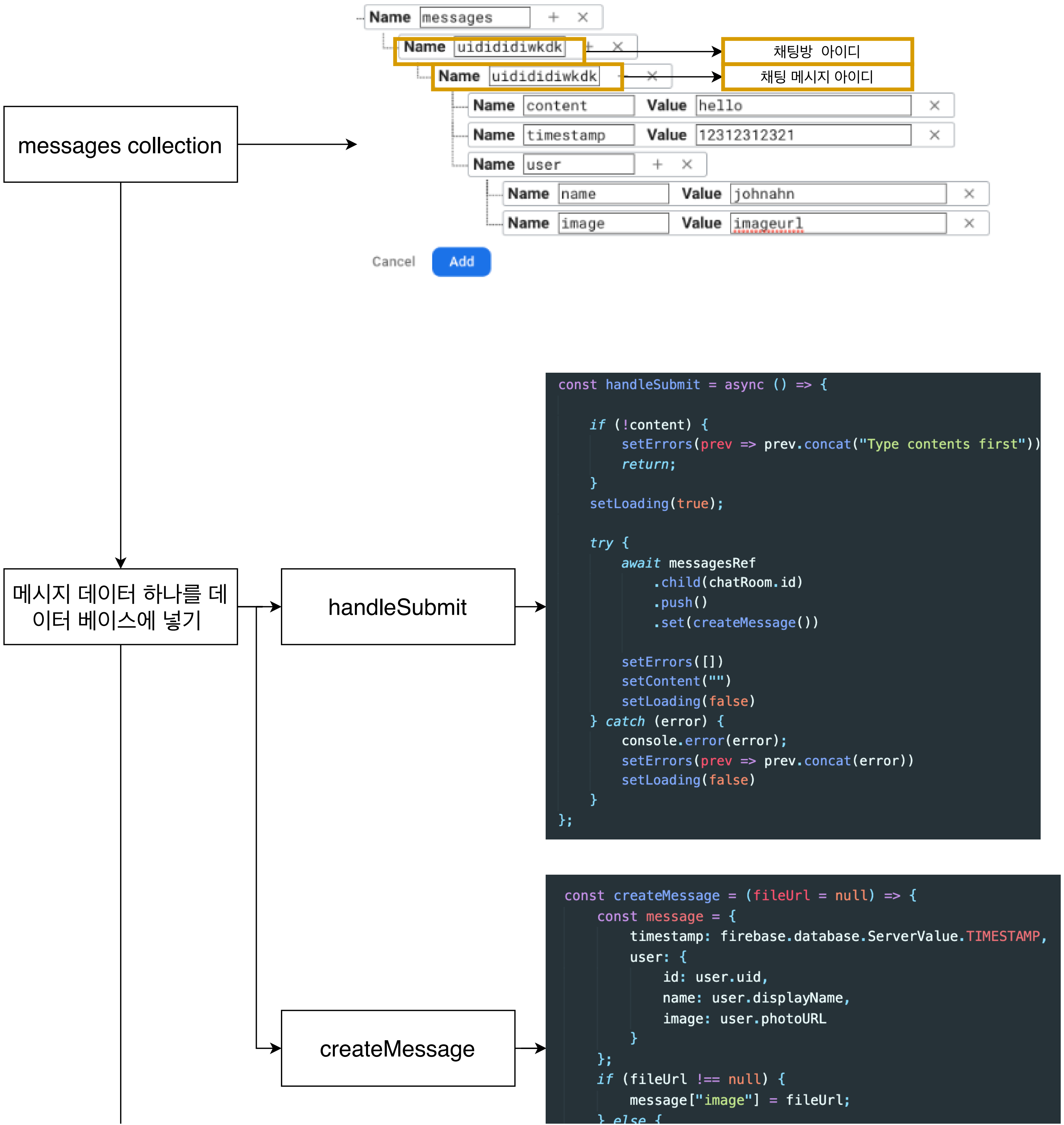
Message Form



```
import React from 'react'
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import Button from 'react-bootstrap/Button';
import ProgressBar from 'react-bootstrap/ProgressBar';

function MessageForm() {
    return (
        <div>
            <Form>
                <Form.Group controlId="exampleForm.ControlTextarea1">
                    <Form.Control as="textarea" rows={3} />
                </Form.Group>
            </Form>

            <ProgressBar variant="warning" now={60} />
            <br />
            <Row>
                <Col>
                    <Button variant="primary" size="lg" style={{ width: '100%' }} >
                        SEND
                    </Button>{' '}
                </Col>
                <Col>
                    <Button variant="success" size="lg" style={{ width: '100%' }} >
                        UPLOAD
                     </Button>{' '}
                </Col>
            </Row>
        </div>
    )
}

export default MessageForm
```


ChatRoomName
Username
Description
Counts !
Created By
MessageBody
SEND
UPLOAD

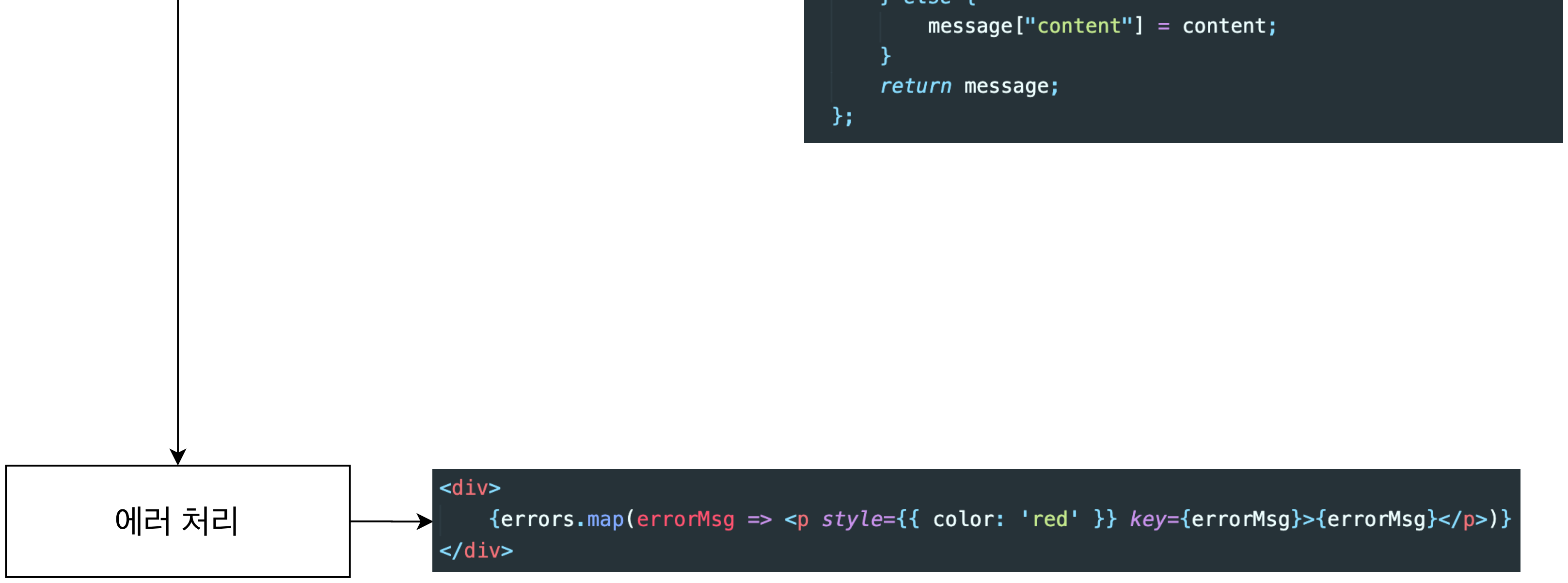
메시지 저장하기



```
Name messages
  Name uidididiwkdk
    Name uidididiwkdk
      Name content      Value hello
      Name timestamp    Value 12312312321
      Name user
        Name name       Value johnahn
        Name image      Value imageurl
```

```
const handleSubmit = async () => {

    if (!content) {
        setErrors(prev => prev.concat("Type contents first"))
        return;
    }
    setLoading(true);

    try {
        await messagesRef
            .child(chatRoom.id)
            .push()
            .set(createMessage())

        setErrors([])
        setContent("")
        setLoading(false)
    } catch (error) {
        console.error(error);
        setErrors(prev => prev.concat(error))
        setLoading(false)
    }
};
```

```
const createMessage = (fileUrl = null) => {
    const message = {
        timestamp: firebase.database.ServerValue.TIMESTAMP,
        user: {
            id: user.uid,
            name: user.displayName,
            image: user.photoURL
        }
    };
    if (fileUrl !== null) {
        message["image"] = fileUrl;
    } else {
```

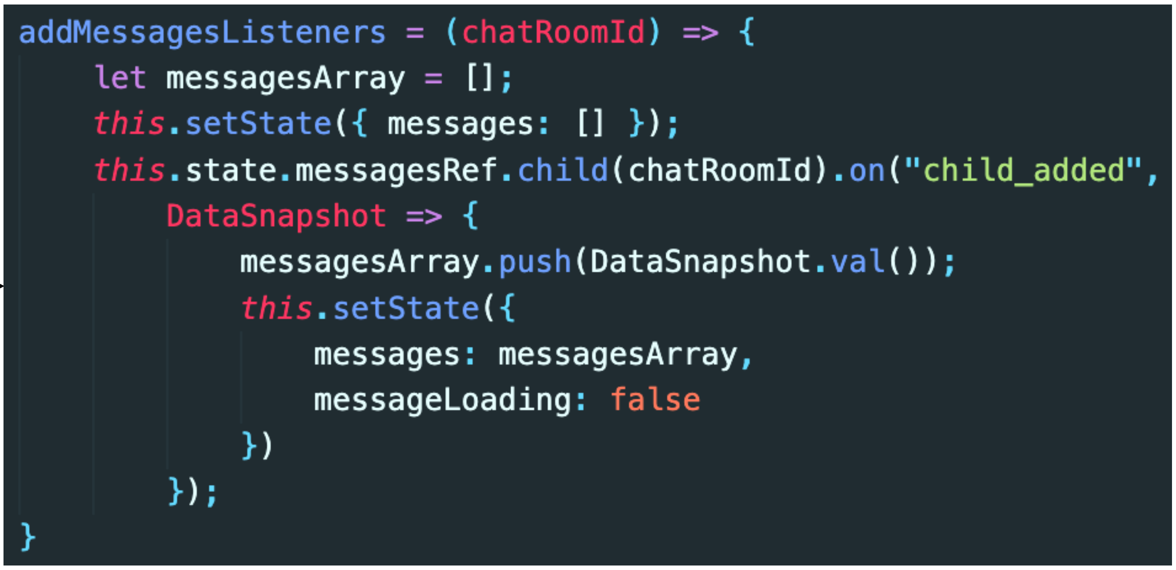


```
    } else {
        message["content"] = content;
    }
    return message;
};
```

```
<div>
    {errors.map(errorMsg => <p style={{ color: 'red' }} key={errorMsg}>{errorMsg}</p>)}
</div>
```

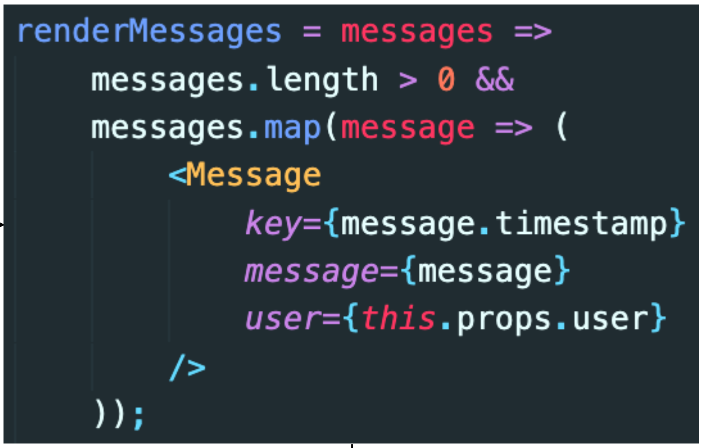
메시지 보여주기 (Read)

메시지 데이터 실시간으로 가져오기

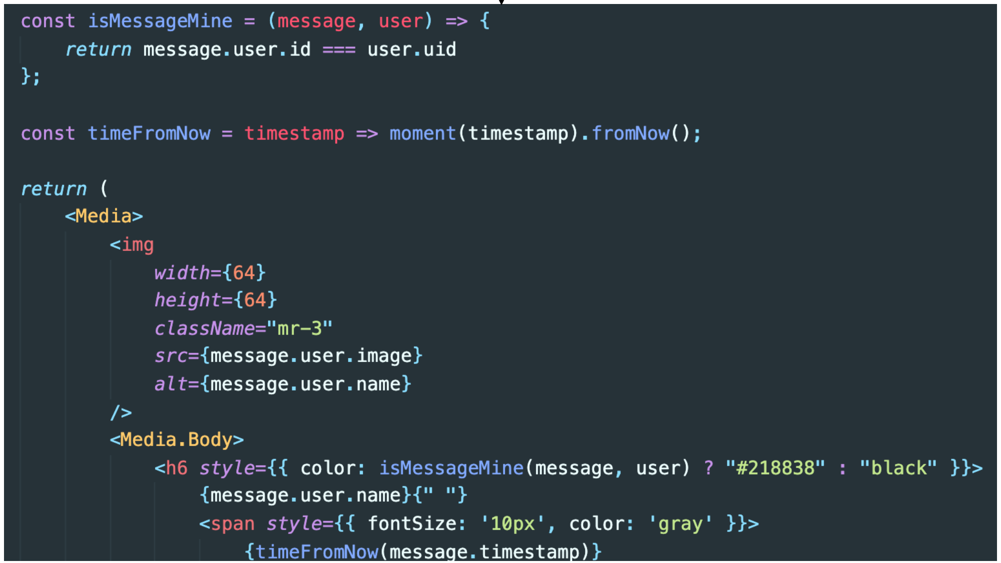


```
addMessagesListeners = (chatRoomId) => {
    let messagesArray = [];
    this.setState({ messages: [] });
    this.state.messagesRef.child(chatRoomId).on("child_added",
        DataSnapshot => {
            messagesArray.push(DataSnapshot.val());
            this.setState({
                messages: messagesArray,
                messageLoading: false
            })
        });
}
```

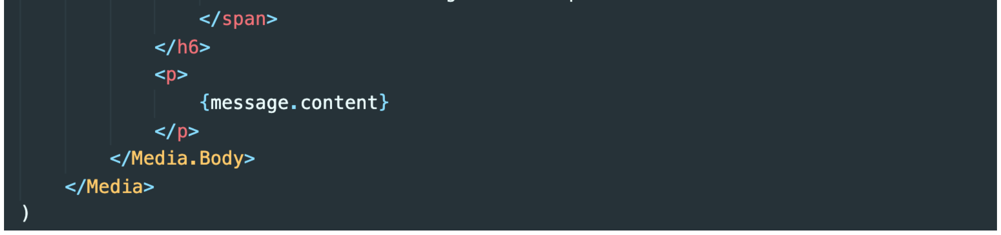
메시지 보여주기



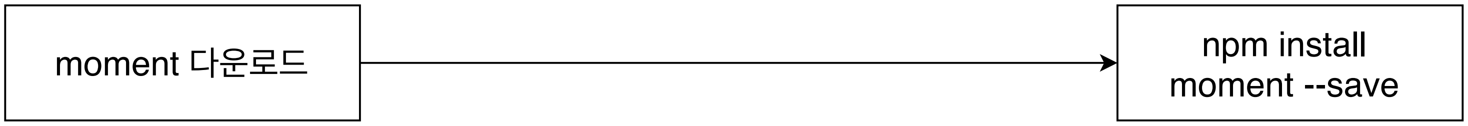
```
renderMessages = messages =>
    messages.length > 0 &&
    messages.map(message => (
        <Message
            key={message.timestamp}
            message={message}
            user={this.props.user}
        />
    ));
```



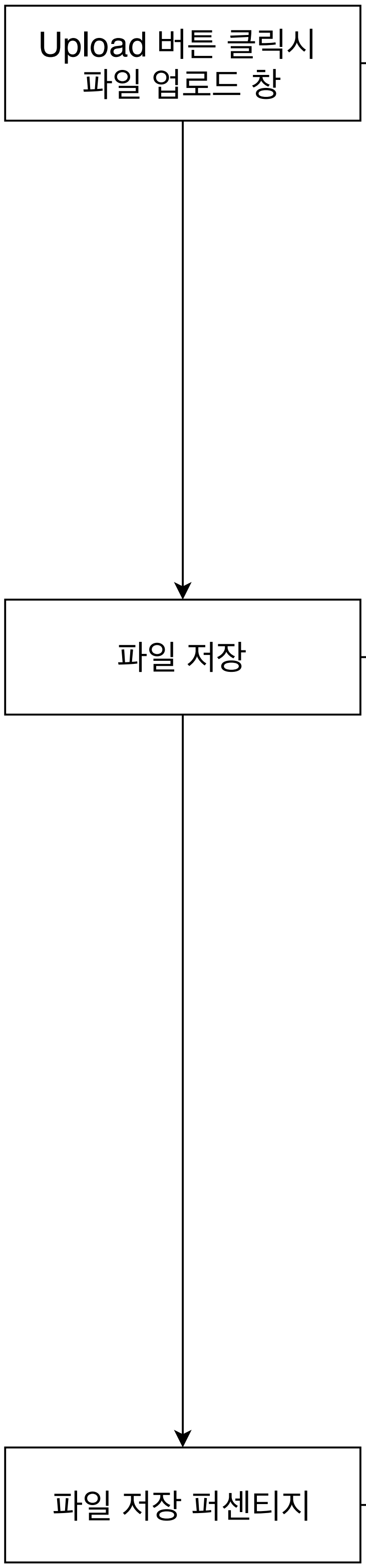
```
const isMessageMine = (message, user) => {
    return message.user.id === user.uid
};

const timeFromNow = timestamp => moment(timestamp).fromNow();

return (
    <Media>
        <img
            width={64}
            height={64}
            className="mr-3"
            src={message.user.image}
            alt={message.user.name}
        />
        <Media.Body>
            <h6 style={{ color: isMessageMine(message, user) ? "#218838" : "black" }}>
                {message.user.name}{" "}
                <span style={{ fontSize: '10px', color: 'gray' }}>
                    {timeFromNow(message.timestamp)}
```

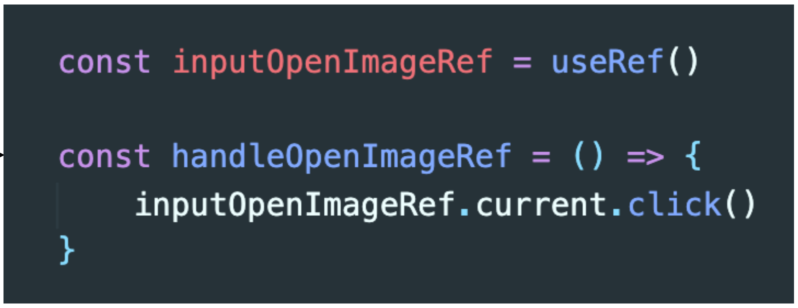


```
                </span>
            </h6>
            <p>
                {message.content}
            </p>
        </Media.Body>
    </Media>
)
```

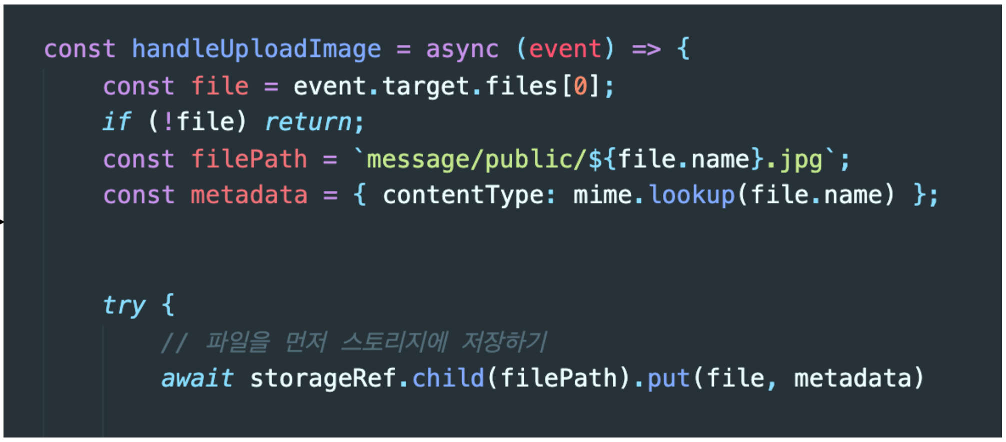


파일 업로드

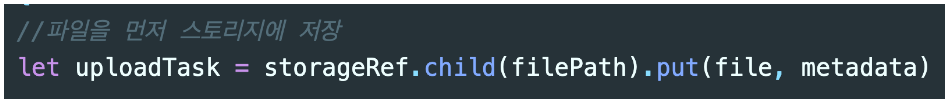




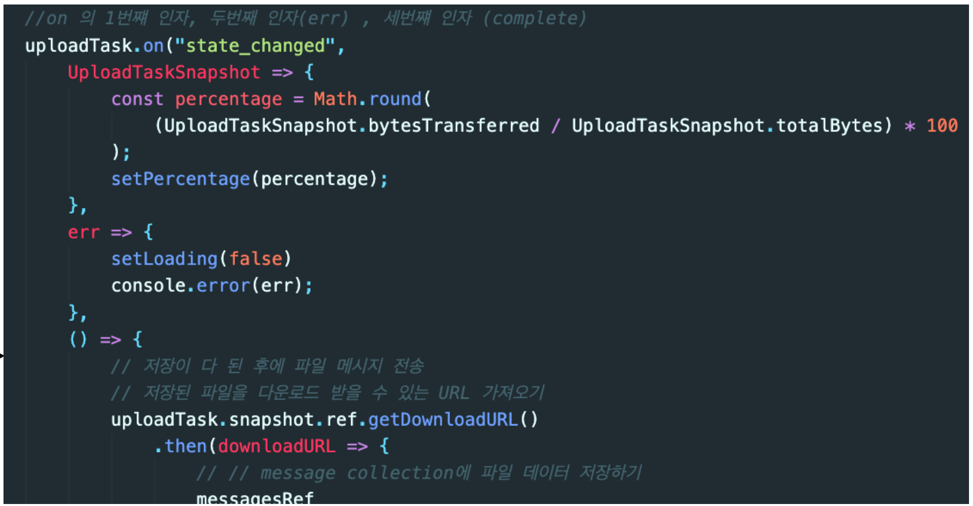
```
const inputOpenImageRef = useRef()

const handleOpenImageRef = () => {
    inputOpenImageRef.current.click()
}
```



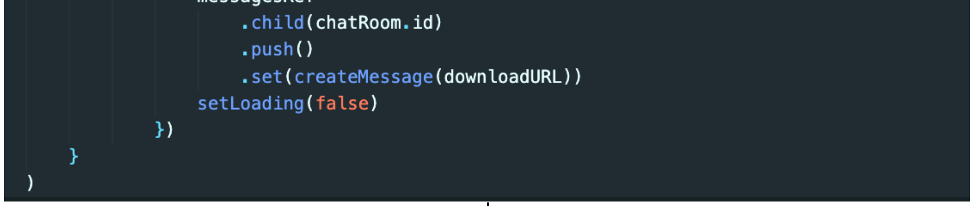
```
const handleUploadImage = async (event) => {
    const file = event.target.files[0];
    if (!file) return;
    const filePath = `message/public/${file.name}.jpg`;
    const metadata = { contentType: mime.lookup(file.name) };


    try {
        // 파일을 먼저 스토리지에 저장하기
        await storageRef.child(filePath).put(file, metadata)
```



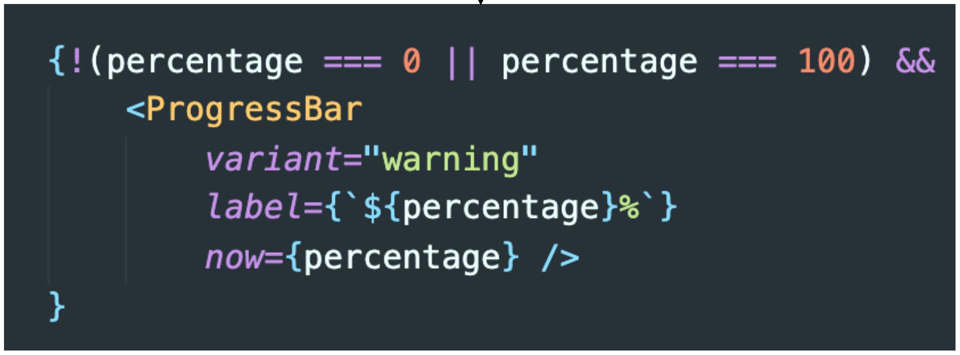
```
//파일을 먼저 스토리지에 저장
let uploadTask = storageRef.child(filePath).put(file, metadata)
```



```
//on 의 1번째 인자, 두번째 인자(err) , 세번째 인자 (complete)
uploadTask.on("state_changed",
    UploadTaskSnapshot => {
        const percentage = Math.round(
            (UploadTaskSnapshot.bytesTransferred / UploadTaskSnapshot.totalBytes) * 100
        );
        setPercentage(percentage);
    },
    err => {
        setLoading(false)
        console.error(err);
    },
    () => {
        // 저장이 다 된 후에 파일 메시지 전송
        // 저장된 파일을 다운로드 받을 수 있는 URL 가져오기
        uploadTask.snapshot.ref.getDownloadURL()
            .then(downloadURL => {
                // // message collection에 파일 데이터 저장하기
                messagesRef
```

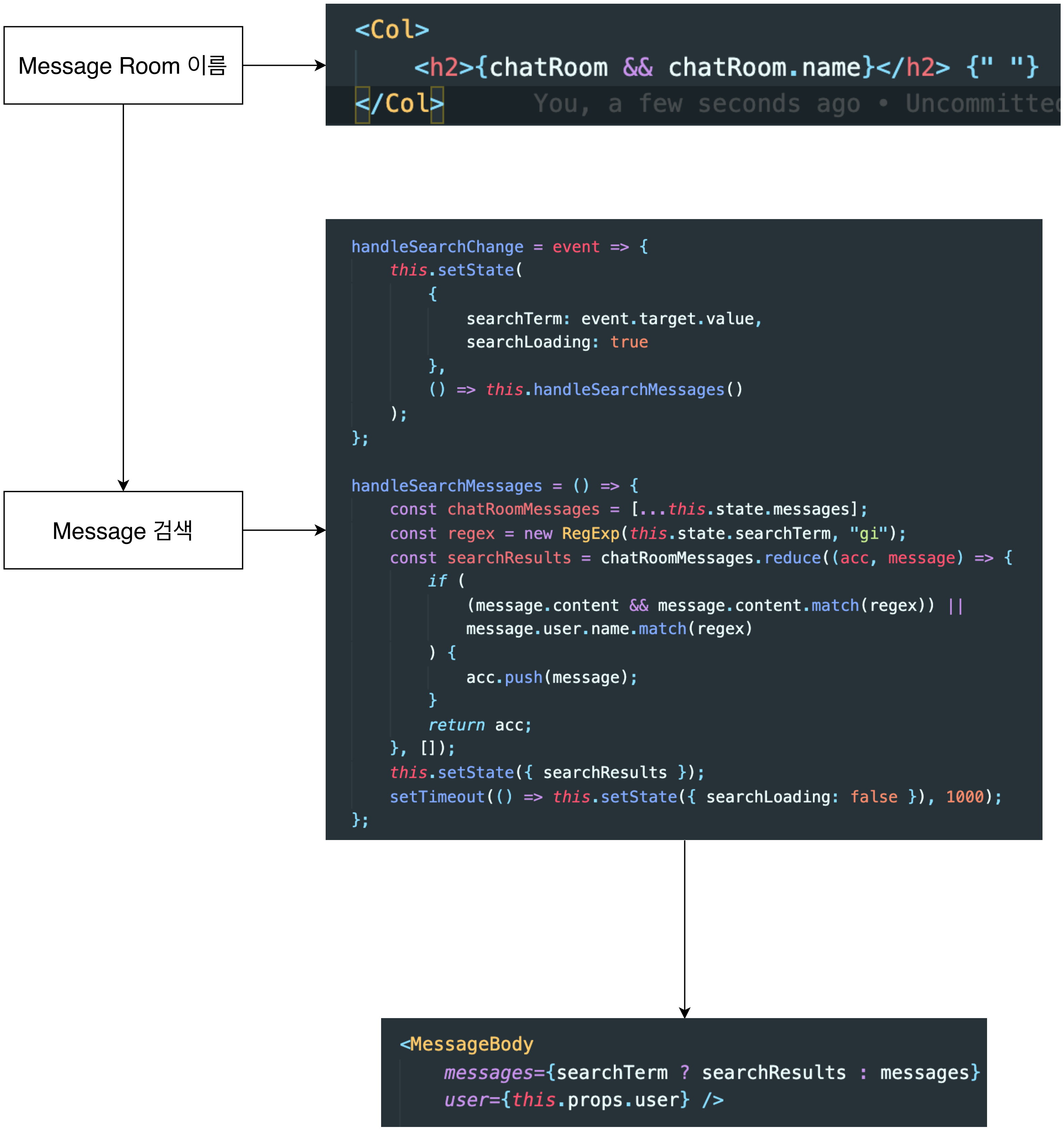


```
                messagesRef
                    .child(chatRoom.id)
                    .push()
                    .set(createMessage(downloadURL))
                setLoading(false)
            })
    }
)
```



```
{!(percentage === 0 || percentage === 100) &&
    <ProgressBar
        variant="warning"
        label={`${percentage}%`}
        now={percentage} />
}
```

메시지 검색



```
<Col>
    <h2>{chatRoom && chatRoom.name}</h2> {" "}
</Col>        You, a few seconds ago • Uncommitte
```

```
handleSearchChange = event => {
    this.setState(
        {
            searchTerm: event.target.value,
            searchLoading: true
        },
        () => this.handleSearchMessages()
    );
};

handleSearchMessages = () => {
    const chatRoomMessages = [...this.state.messages];
    const regex = new RegExp(this.state.searchTerm, "gi");
    const searchResults = chatRoomMessages.reduce((acc, message) => {
        if (
            (message.content && message.content.match(regex)) ||
            message.user.name.match(regex)
        ) {
            acc.push(message);
        }
        return acc;
    }, []);
    this.setState({ searchResults });
    setTimeout(() => this.setState({ searchLoading: false }), 1000);
};
```

```
<MessageBody
    messages={searchTerm ? searchResults : messages}
    user={this.props.user} />
```

accumulator

1. 축적자 2. 축재자 3. 누산기

```
g modifier: global. All matches (don't return on first match)

i modifier: insensitive. Case insensitive match (ignores case of [a-zA-Z])
```