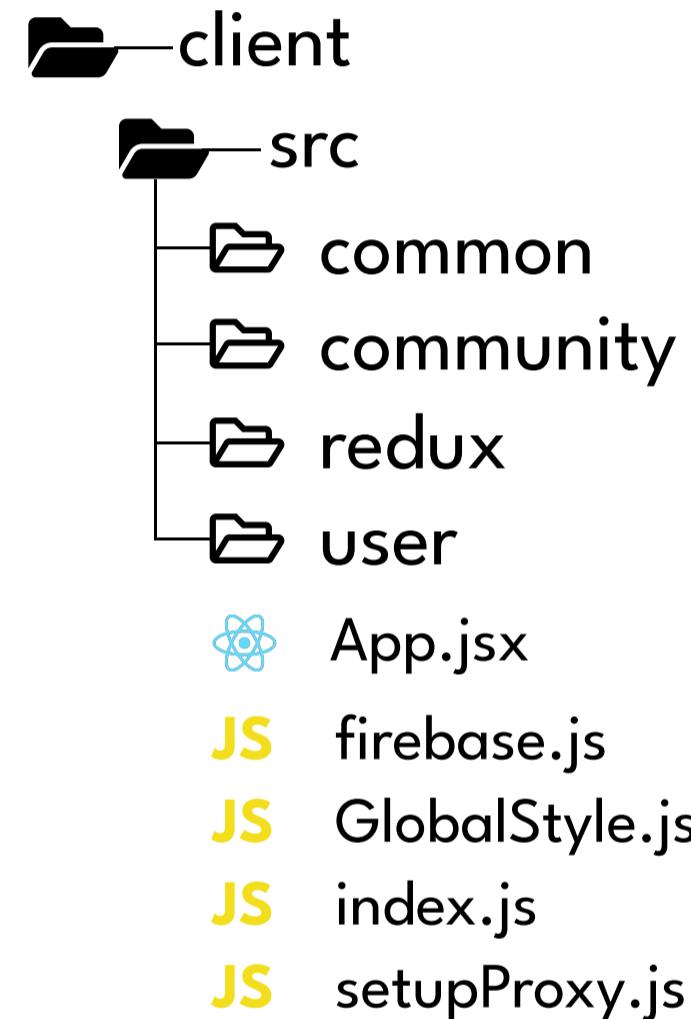


# Intro

**name** : Notice  
**Skills** : HTML, JAVASCRIPT, FIREBASE, MONGO DB



## MAIN

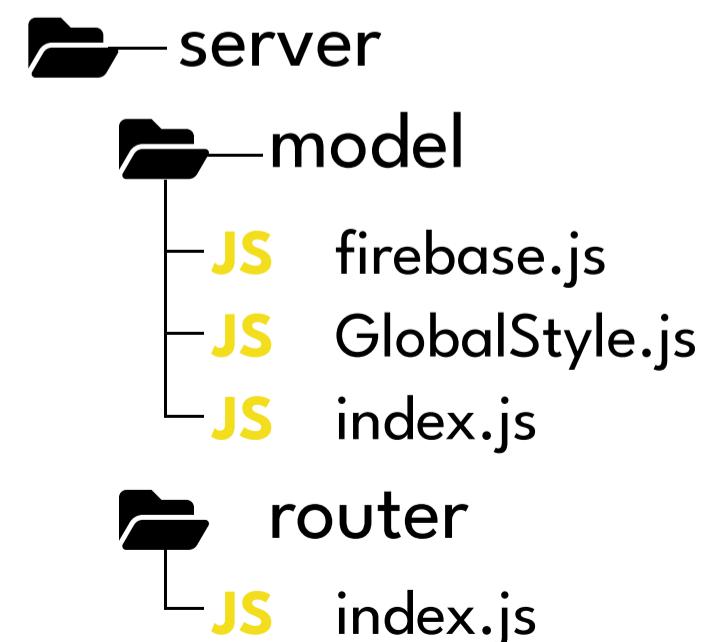
# NOTICE

Hello.  
This is Recent Newest

MAIN

안녕하세요!반갑습니다

작성자: jjjshlove  
수정일: 2023-03-30



jjjshlove님 반갑습니다. 로그아웃

# Firebase / MongoDB 01

---

## client / App.jsx

```
useEffect(() => {
  firebase.auth().onAuthStateChanged((userInfo) => {
    컴포넌트 마운트시 firebase로 받은 유저정보가 없으면 (로그인 상태가 아니면)
    logout함수를 호출해서 유저정보를 비우는 액션객체 반환하고 해당 액션객체를 dispatch로 리듀서에 전달
    if (userInfo === null) dispatch(logoutUser());
    firebase로 받은 유저정보가 있으면 (로그인 상태이면)
    loginUser함수 호출시 인수로 로그인된 정보값을 전달해서 해당 정보가 담긴 액션객체를 생성후 dispatch로 전달
    else dispatch(loginUser(userInfo.multiFactor.user));
  });
}, [dispatch]);
```

## server / index.js

```
mongoDB 접속 구문
app.listen(port, () => {
  mongoose
    .connect('mongodb+srv://ljjshlove420:!abcd1234@cluster0.7kgnst.mongodb.net/
      ?retryWrites=true&w=majority')
    .then(() => console.log(`Server app listening on port ${port}`))
    .catch((err) => console.log(err));
});
```

# Firebase / MongoDB 02

## JS client / firebase.js

회원가입 정보 입력후 해당 정보값을 firebase로 인증정보 저장

```
import firebase from 'firebase/compat/app';
import 'firebase/compat/auth';

const firebaseConfig = {
  apiKey: 'AlzaSyAdqKCVxLLoOhFMuatIgPaGO6VcQl3ej4Y',
  authDomain: 'react-project-ba468.firebaseio.com',
  projectId: 'react-project-ba468',
  storageBucket: 'react-project-ba468.appspot.com',
  messagingSenderId: '1032476498509',
  appId: '1:1032476498509:web:0caa063e3da0f1ad215a04',
};

firebase.initializeApp(firebaseConfig);
```

## JS server / model / PostSchema.js

Schema : 데이터베이스에 저장될 자료형식이나 키값을 강제하는 시스템적인 틀  
게시글 저장시 post모델에 데이터 추가될때 해당 글을 작성하는 사용자 정보도  
userNum값을 이용해서 User 모델의 데이터 다큐먼트를 참조해서 가져오기 위함

```
const mongoose = require('mongoose');
const postSchema = new mongoose.Schema(
  {
    title: String,
    content: String,
    communityNum: Number,
    writer: {
      ref: 'User',
      type: mongoose.Schema.Types.ObjectId,
    },
  },
  { collection: 'Posts', timestamps: true }
);

const Post = mongoose.model('Post', postSchema);
module.exports = { Post };
```

# Firebase / MongoDB

## • 글 저장 라우터

```
const express = require('express');
const router = express.Router();
const { Post } = require('../model/postSchema.js');
const { Counter } = require('../model/counterSchema.js');
const { User } = require('../model/userSchema.js');

router.post('/create', (req, res) => {
  const temp = req.body;

  Counter.findOne({ name: 'counter' })
    .exec()
    .then((doc) => {
      temp.communityNum = doc.communityNum;
    });

```

**temp 객체에 있는 현재 로그인된 사용자의 아이디로 User 컬렉션으로 부터 해당 document를 찾음**

```
User.findOne({ uid: temp.uid })
```

```
  .exec()
  .then((doc) => {
    해당 다큐먼트의 ObjectId값을 body-parser로 전달받은 temp 객체의 writer 키값에 등록
    temp.writer = doc._id;
```

**위에서 최종적으로 만들어진 temp 객체로 PostModel 인스턴스 생성후 DB에 저장**

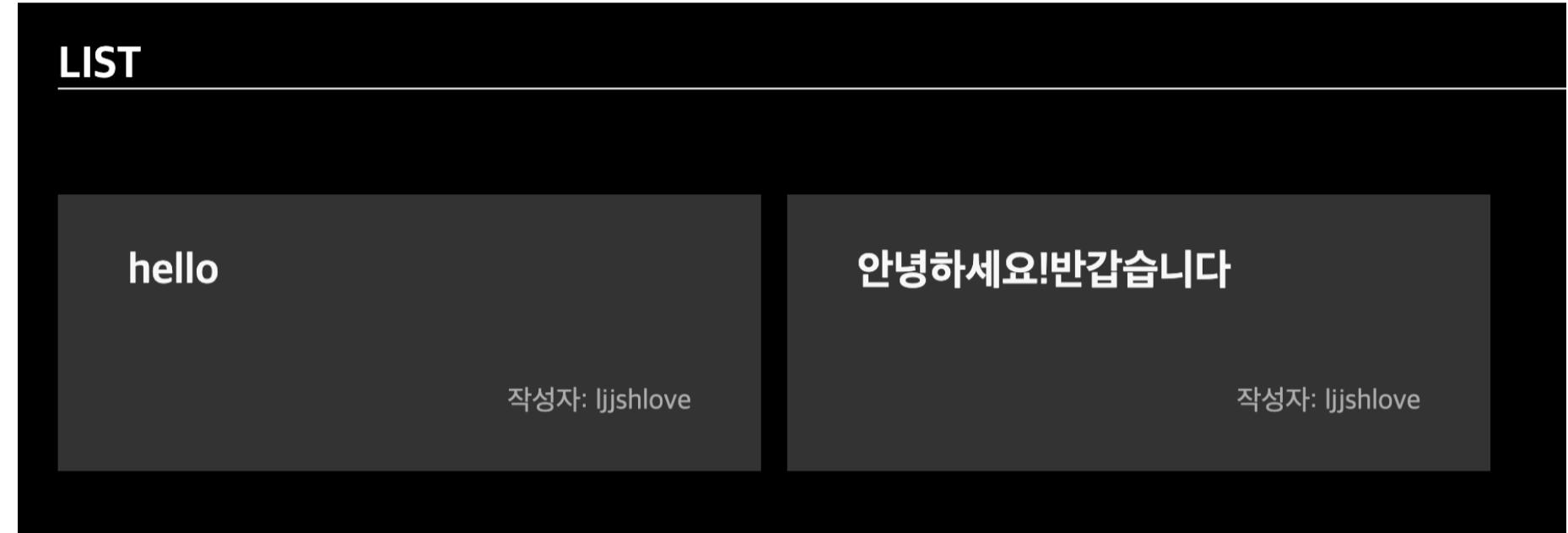
```
const PostModel = new Post(temp);
PostModel.save().then(() => {
  Counter.updateOne({ name: 'counter' }, { $inc: { communityNum: 1 } }).then(() => res.json({ success: true }));
});
}).catch((err) => console.log(err));
});
```

The image shows a dark-themed web application interface. At the top, it says "POST". Below that, there are two input fields: one for "Title" containing the value "hello" and another for "Content" containing the value "Hello world". At the bottom of the form is a grey "SEND" button.

# CommunityRouter.js 01

- 글 목록 요청 라우터

```
router.get('/read/:count', (req, res) => {
  Post.find()
    .populate('writer')
    .sort({ createdAt: -1 })
    .limit(req.params.count)
    .exec()
    .then((doc) => {
      res.json({ success: true, communityList: doc });
    })
    .catch((err) => {
      console.log(err);
      res.json({ success: false });
    });
});
```

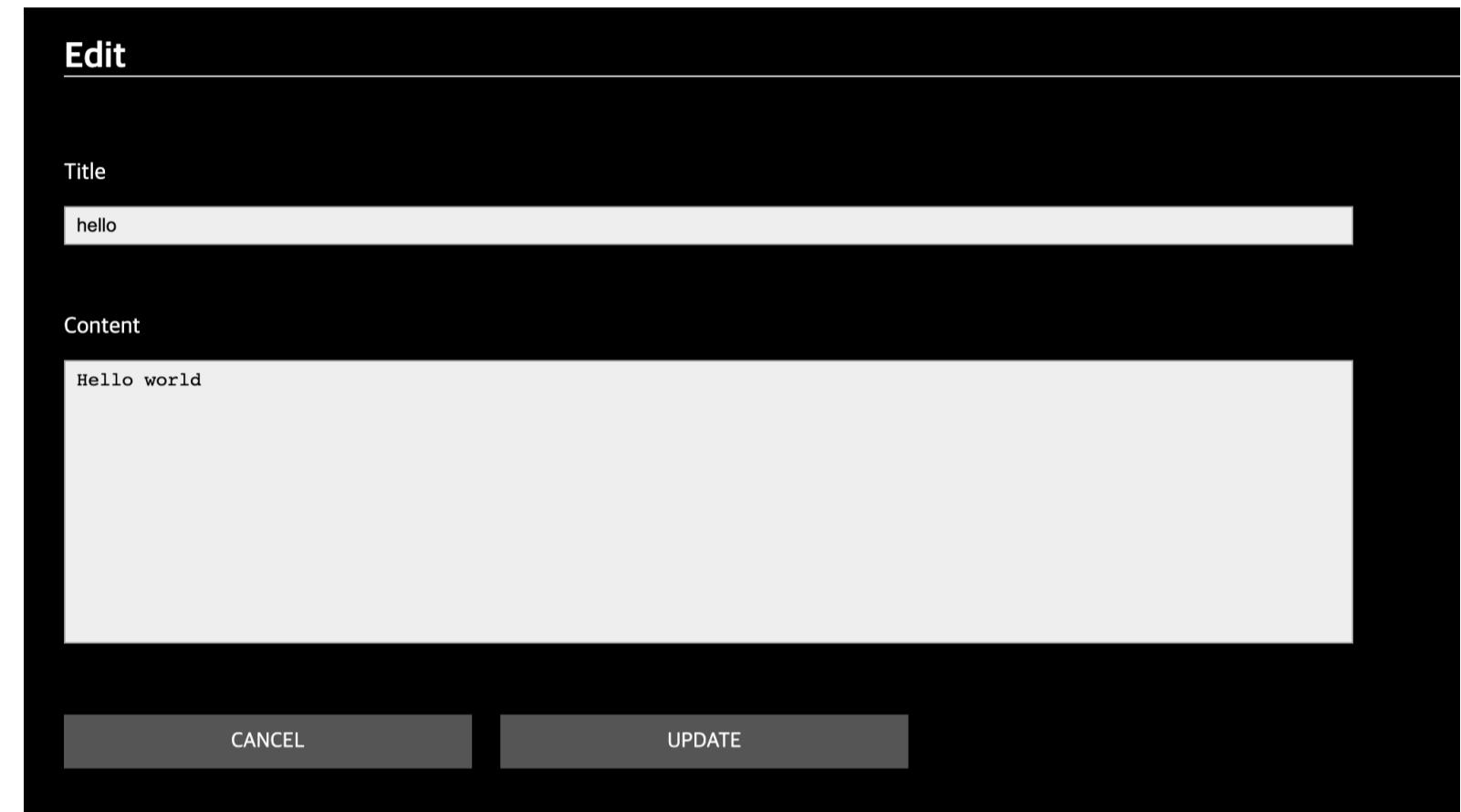


# CommunityRouter.js 02

- 글 수정 요청 라우터

```
router.put('/edit', (req, res) => {
  const temp = {
    title: req.body.title,
    content: req.body.content,
  };

  Post.updateOne({ communityNum: req.body.num }, { $set: temp })
    .exec()
    .then((doc) => {
      console.log(doc);
      res.json({ success: true });
    })
    .catch((err) => res.json({ success: false }));
});
```



- 글 삭제 요청 라우터

```
router.delete('/delete/:num', (req, res) => {
  Post.deleteOne({ communityNum: req.params.num })
    .exec()
    .then(() => res.json({ success: true }))
    .catch(() => res.json({ success: false }));
});
```

