React 기본 프로젝트

사전 준비

1. Django Backend 설치

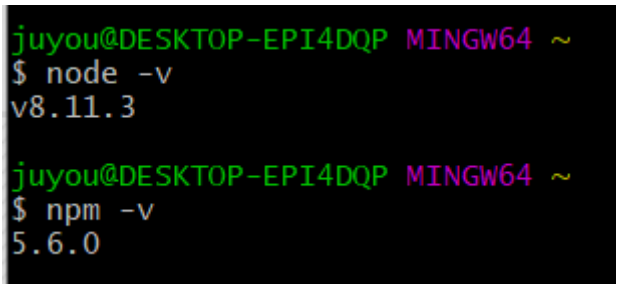
https://github.com/csy1204/djangobackend

2. React를 위한 환경 세팅

https://nodejs.org/ko/

node.js 설치


```
node -v
npm -v
```



이렇게 된다면 정상적으로 설치 완료

```
npm install -g yarn
```

npm 대신 yarn을 쓸거임.

3. yarn을 통한 create-react-app 패키지 설치

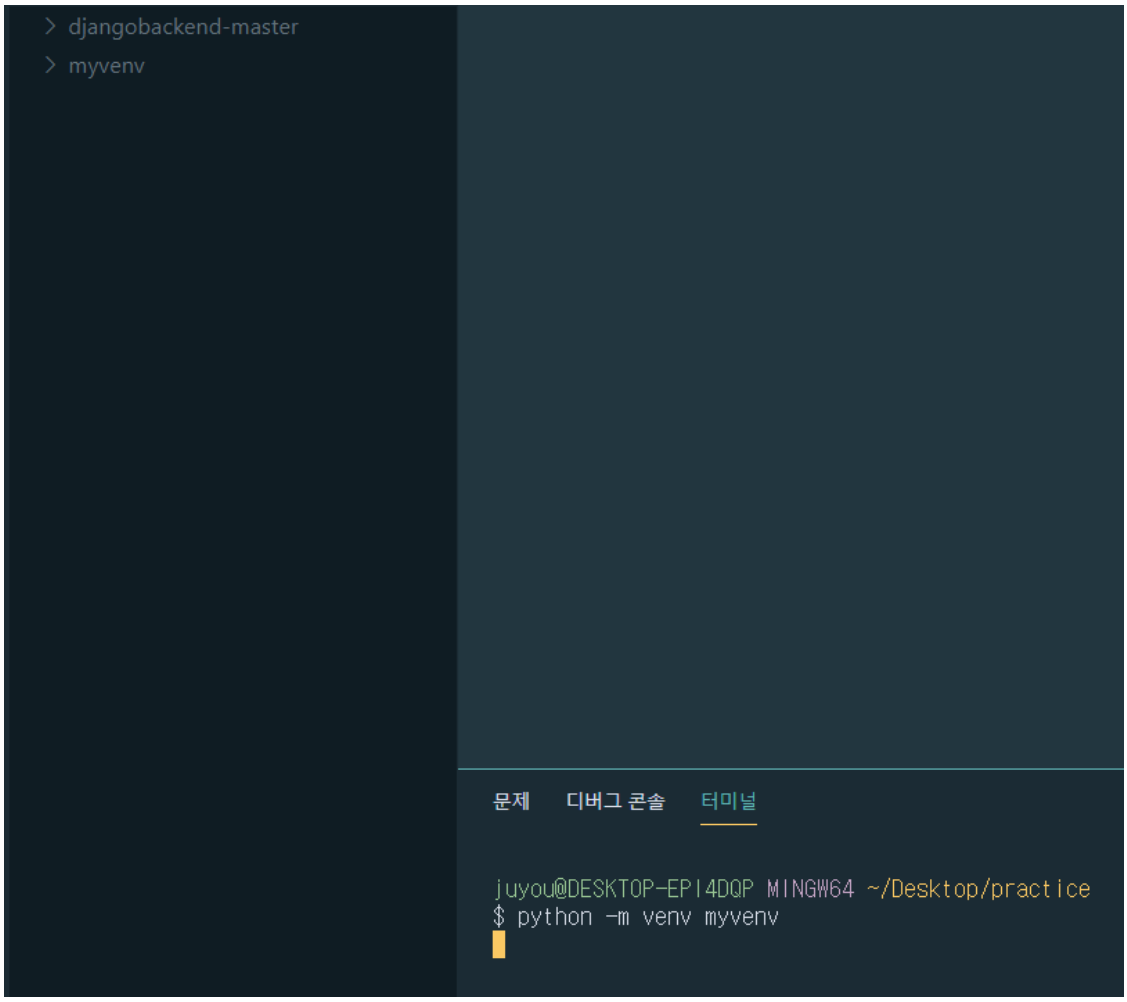
```
yarn global add create-react-app
```

react를 사용하기 위한 환경을 편리하게 세팅을 해주는 패키지

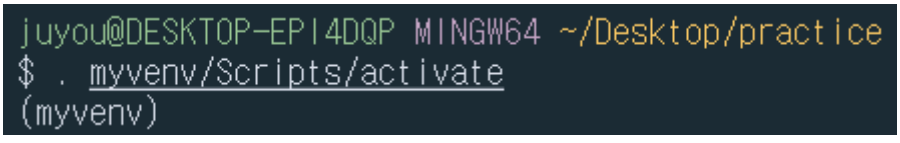
같이 해봅시다.

1. **가상환경 설치**

```
python -m venv myvenv
```



2.
```
. myvenv/Scripts/activate
```



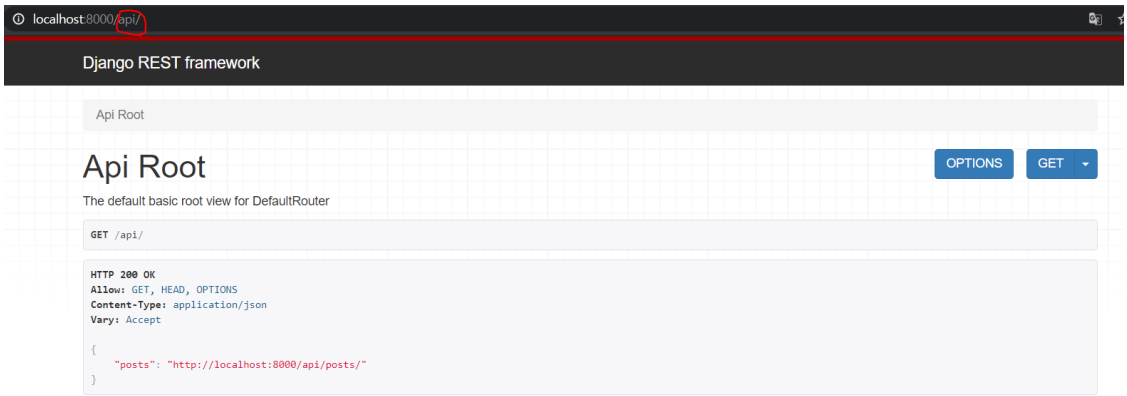
가상 환경 실행.

3. django app들 설치.

```
pip install django
pip install djangorestframework
pip install django-cors-headers
```

backend 프로젝트를 실행시키기 위한 필수 app

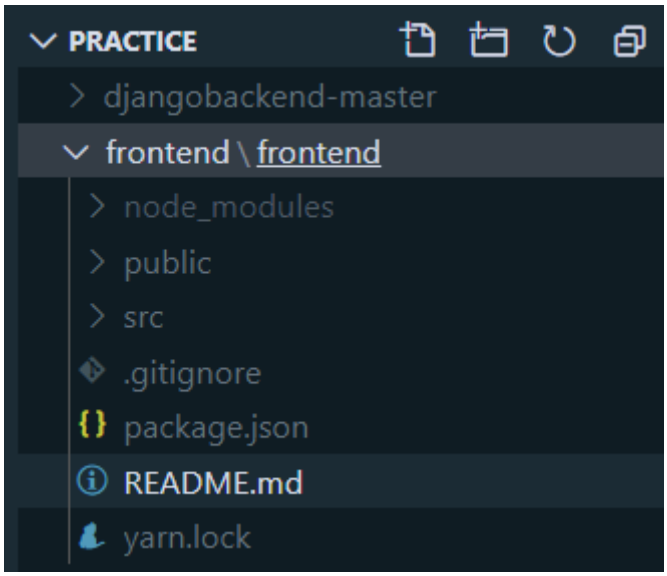
4.
```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```



제대로 잘 했으면 여기까지 완료.

5.
```
yarn create react-app frontend
```

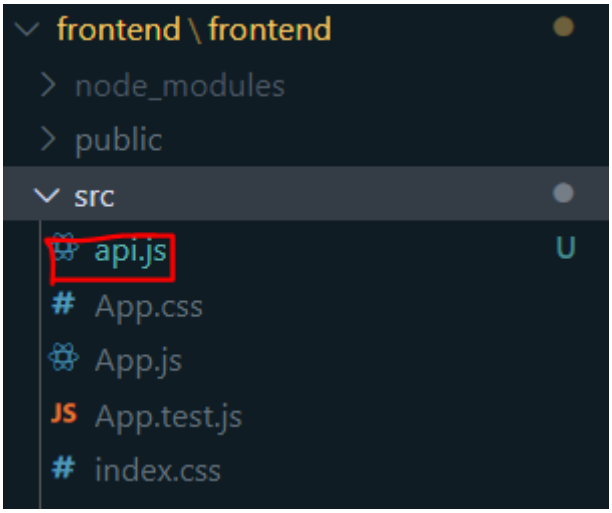
create-react-app 패키지를 통해 react 프로젝트 생성(yarn start 명령어가 안먹히시는 분들은 git bash창이 아닌 cmd창에서 실행하시길 권장)



create-react-app이 잘 설치가 되고 모든 과정이 잘 되면은 frontend 폴더가 생성됨.

6.
```
yarn add axios
```

frontend 폴더 내에서 실행



api.js 폴더를 만들고

```
import axios from 'axios'

axios.defaults.baseURL = 'http://127.0.0.1:8000/api'

export default {
  getAllPosts() {
    return axios.get('/posts/')
  },

  createPost(data) {
    return axios.post('/posts/', data)
  },
}
```

서버와의 api 통신을 위한 js파일.

7. app.js 파일 수정하기

```
import React, { Component } from 'react'
import './App.css'

export default class App extends Component {
  render() {
    return <div className="App"></div>
  }
}
```

8.

```
<div className="App">
    <div className="PostingSection">
        <form className="" onSubmit={this.handleSubmit}>
            <input type="text" name="title"/>
            <textarea name="content"></textarea>
            <button type="submit">제출하기</button>
        </form>
    </div>
    <div className="ViewSection"></div>
</div>
```

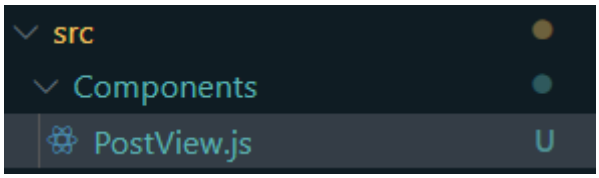
PostingSection과 ViewSection을 추가.

form태그, input, textarea, button 태그를 추가해서 post를 입력할 수 있게 함.

```
export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      title: '',
      content: '',
    }
  }

  handleChange = evt => {
    this.setState({
      [evt.target.name]: evt.target.value,
    })
  }

  handleSubmit = evt => {
    evt.preventDefault()
    api.createPost({
      title: this.state.title,
      content: this.state.content,
    })
  }

  render() {
    return (
      <div className="App">
        <div className="PostingSection">
          <form className="" onSubmit={this.handleSubmit}>
            <input
              type="text"
              name="title"
              onChange={this.handleChange}
              value={this.state.title}
            />
            <textarea
              name="content"
              onChange={this.handleChange}
              value={this.state.content}></textarea>
            <button type="submit">제출하기</button>
          </form>
        </div>
```

```
        <div className="ViewSection"></div>
      </div>
    )
  }
}
```

Posting을 위한 form태그를 위한 hanldeSubmit과 input, textarea의 value의 변화를 위한 handleChange를 생성 후 적용.

9. 

글들을 보여주기 위한 Component를 넣을 Components 폴더 생성 후 PostView.js 파일 생성.

```
import React, { Component } from 'react'

const dummy_prop = {
  title: '테스트용 제목',
  content: '테스트용 내용',
}

export default class PostView extends Component {
  render() {
    const { title, content } = dummy_prop
    return (
      <div>
        <h3>{title}</h3>
        <p>{content}</p>
      </div>
    )
  }
}
```

App.js의 ViewSection에 넣을 PostView Component 선언.

10.
```
import React, { Component } from 'react'
import './App.css'
import api from './api'

// PostView Import
import PostView from './Components/PostView'

export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      title: '',
      content: '',
    }
  }
```

```
  handleChange = evt => {
    this.setState({
      [evt.target.name]: evt.target.value,
    })
  }

  handleSubmit = evt => {
    evt.preventDefault()
    api.createPost({
      title: this.state.title,
      content: this.state.content,
    })
  }

  render() {
    return (
      <div className="App">
        <div className="PostingSection">
            {/* 이전에 있던 내용을 지우라는 것이 아닌 코드가 길어져 생략한 것임. */}
        </div>
        <div className="ViewSection">
          {/*PostView 추가.*/}
          <PostView />
        </div>
      </div>
    )
  }
}
```

App.js 에 PostView Import 후 ViewSection에 PostView 추가.

11.

```
import React, { Component } from 'react'
import './App.css'
import api from './api'
import PostView from './Components/PostView'

export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      title: '',
      content: '',
      // 모든 Post들을 저장할 results 배열.
      results: [],
    }
  }

  // Posts 불러오기.
  componentDidMount() {
    this.getPosts()
  }

  // getPosts()를 선언.
  async getPosts() {
    let _results = await api.getAllPosts()
```

```
    this.setState({ results: _results.data })
  }

  render() {
    return (
      <div className="App">
        <div className="PostingSection">
         {/* 이전에 있던 내용을 지우라는 것이 아닌 코드가 길어져 생략한 것임. */}
        </div>
        <div className="ViewSection">
          {/* map을 이용해서 PostView를 생성 */}
          {this.state.results.map(post => (
            <PostView />
          ))}
        </div>
      </div>
    )
  }
}
```

12.
```
render() {
    return (
      <div className="App">
        <div className="PostingSection">
        </div>
        <div className="ViewSection">
          {/* PostView props로 전달 */}
          {this.state.results.map(post => (
            <PostView
              key={post.id}
              id={post.id}
              title={post.title}
              content={post.content}
            />
          ))}
        </div>
      </div>
    )
  }
```

그리고 PostView.js에서 props를 수정

```
import React, { Component } from 'react'

// dummy_prop를 주석처리하거나 제거
// const dummy_prop = {
//   title: '테스트용 제목',
//   content: '테스트용 내용',

export default class PostView extends Component {
  render() {
    // dummy_props에서 this.props로 수정.
    const { id, title, content } = this.props
    return (
```

```
      <div>
        {id}
        <h3>{title}</h3>
        <p>{content}</p>
      </div>
    )
  }
}
```

13.

```
import React, { Component } from 'react'
import './App.css'
import api from './api'
import PostView from './Components/PostView'

export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      title: '',
      content: '',
      results: [],
    }
  }

  componentDidMount() {
    this.getPosts()
  }

  async getPosts() {
    let _results = await api.getAllPosts()
    this.setState({ results: _results.data })
  }

  handleChange = evt => {
    this.setState({
      [evt.target.name]: evt.target.value,
    })
  }

  handleSubmit = async evt => {
    // async, await을 추가해야 정상 작동.
    evt.preventDefault()
    await api.createPost({
      title: this.state.title,
      content: this.state.content,
    })
    // Submit 후에 input과 textarea를 비우고,
    this.setState({
      title: '',
      content: '',
    })
    // Post를 새로 불러와 PostSection 업데이트
    this.getPosts()
  }
```

```
  render() {
    return (
      <div className="App">
        <div className="PostingSection">
        </div>
        <div className="ViewSection">
        </div>
      </div>
    )
  }
}
```

14.
```
import axios from 'axios'

axios.defaults.baseURL = 'http://127.0.0.1:8000/api'

export default {
  getAllPosts() {
    return axios.get('/posts/')
  },

  createPost(data) {
    return axios.post('/posts/', data)
  },

  deletePost(id) {
    return axios.delete('/posts/' + String(id))
  },
}
```

api.js에 delete를 위한 api 설정.

15. App.js 수정. handle delete와 button tag 추가.

```
import React, { Component } from 'react'
import './App.css'
import api from './api'
import PostView from './Components/PostView'

export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      title: '',
      content: '',
      results: [],
    }
  }

  componentDidMount() {
    this.getPosts()
  }
```

```
async getPosts() {
  let _results = await api.getAllPosts()
  this.setState({ results: _results.data })
}

handleChange = evt => {
  this.setState({
    [evt.target.name]: evt.target.value,
  })
}

handleSubmit = async evt => {
  evt.preventDefault()
  await api.createPost({
    title: this.state.title,
    content: this.state.content,
  })
  this.setState({
    title: '',
    content: '',
  })
  this.getPosts()
}

// handleDelete 함수 추가.
// 여기도 async, await
handleDelete = async id => {
  await api.deletePost(id)
  this.getPosts()
}

render() {
  return (
    <div className="App">
      <div className="PostingSection">
      {/* 이전에 있던 내용을 지우라는 것이 아닌 코드가 길어져 생략한 것임. */}
      </div>
      <div className="ViewSection">
        {this.state.results.map(post => (
          <div>
            <PostView
              key={post.id}
              id={post.id}
              title={post.title}
              content={post.content}
            />
            {/* 삭제를 위한 button 추가 */}
            <button
              type="submit"
              onClick={event => this.handleDelete(post.id)}>
              삭제하기
            </button>
          </div>
        ))}
      </div>
    </div>
  )
```

```
  }
}
```

Material UI를 이용해 코드를 이쁘게 꾸며보세요.(강의를 들으시는 것을 강추!)