

전자정부 표준프레임워크

# MSA 템플릿 프론트엔드



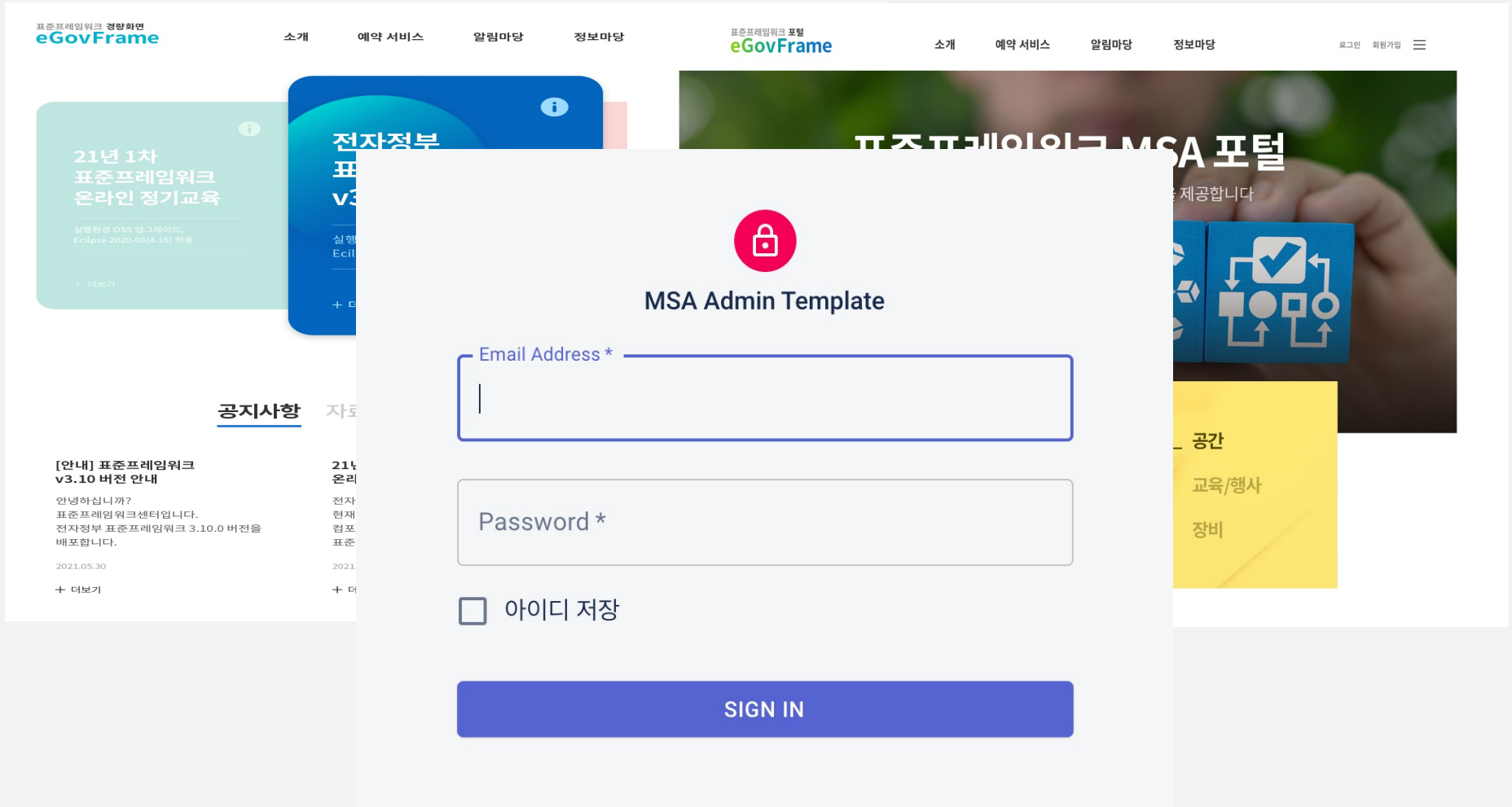
# Contents

1. \_ Frontend App 구성
2. \_ 프로젝트 설정
3. \_ 개발환경 설치
4. \_ 프로젝트 실행

# 1. Frontend Application 구성

MSA 템플릿

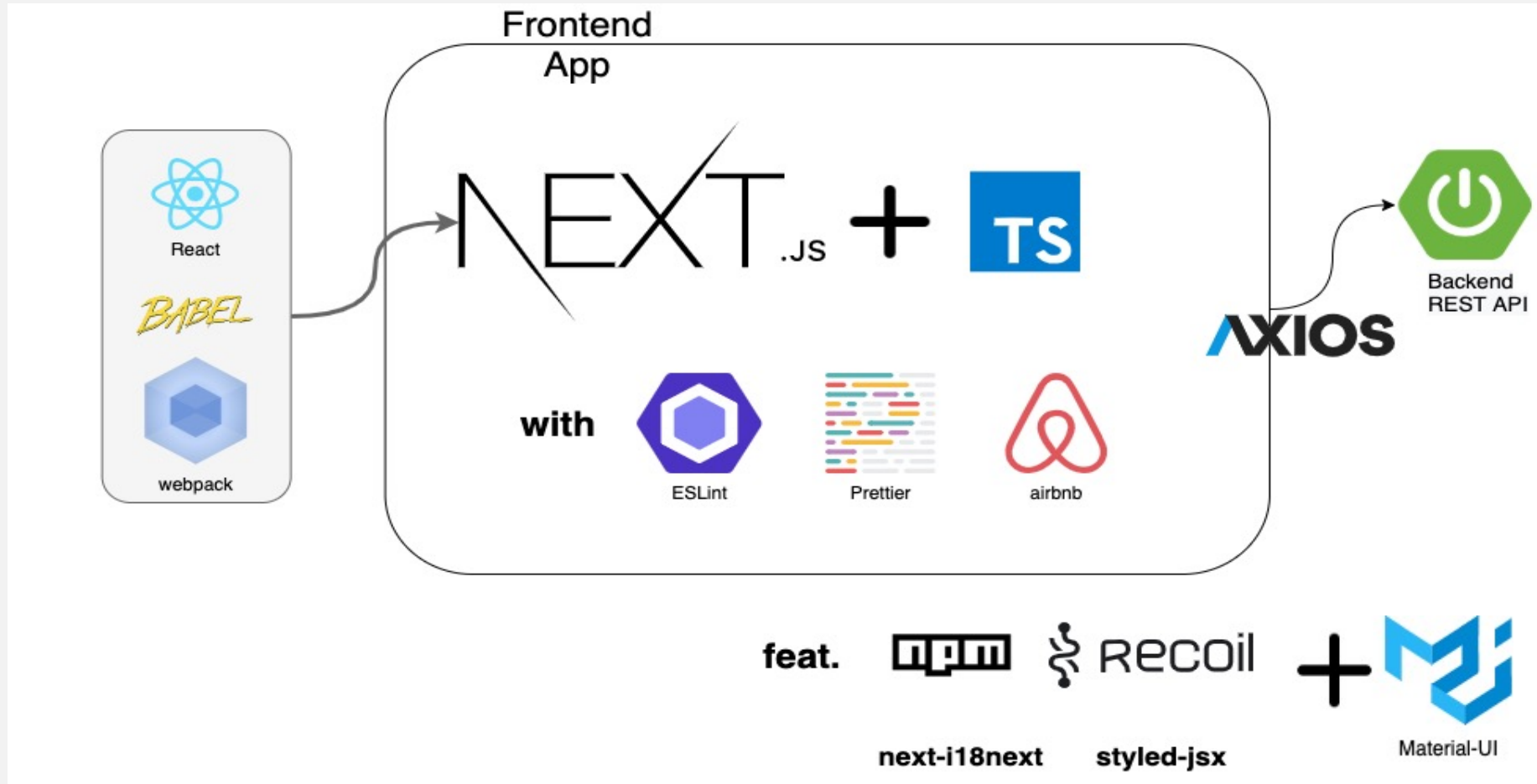
## MSA 템플릿 구성



# 1. Frontend Application 구성

MSA 템플릿

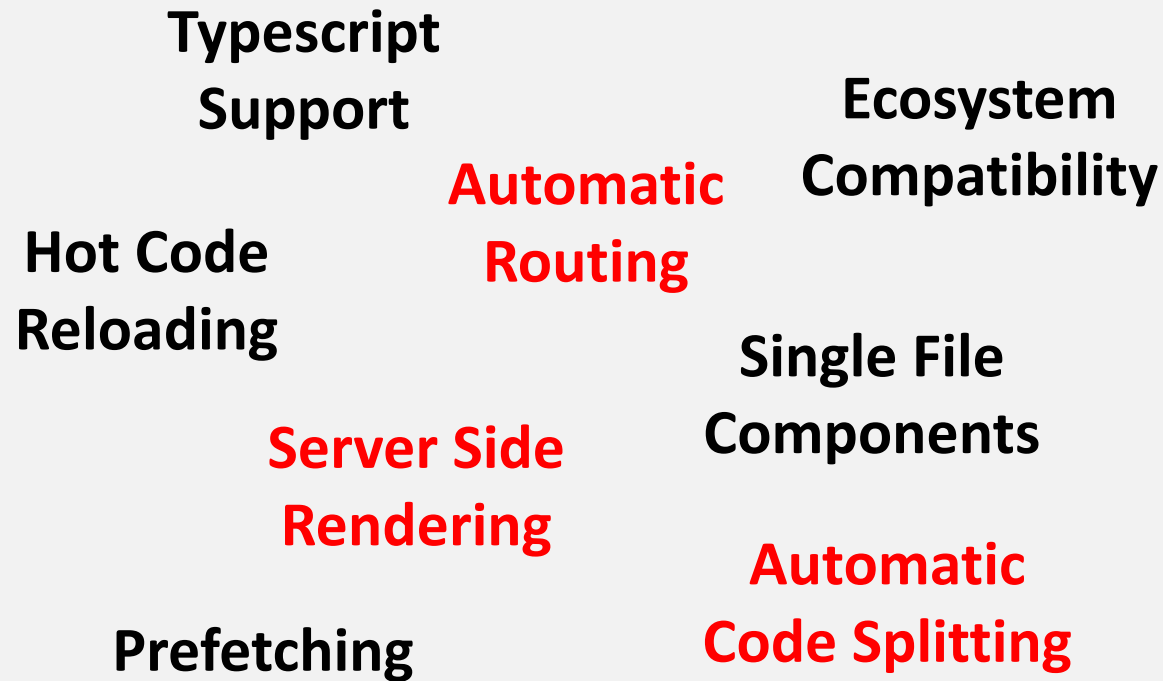
## ● NextJS 프레임워크기반으로 구성





## NextJS ?

별도의 설정 없이 SSR, SEO 등 많은 기능들을 제공하는 React 프레임 워크



# 1. Frontend Application 구성

## NextJS 구조

pages/\_app : 요청 시 가장먼저 실행되는 전체 컴포넌트의 레이아웃

pages/\_document : SPA에서 시작점이 되는 index.html

pages/... : 라우팅 대상이 되는 페이지

pages/api : api 라우팅

## NextJS Resolution Order

On the Server	On the Server with Error	On the Client
1. _app.getInitialProps 2. page.getInitialProps 3. _document.getInitialProps 4. _app.render 5. page.render 6. _document.render	1. _document.getInitialProps 2. _app.render 3. page.render 4. _document.render	1. _app.getInitialProps 2. page.getInitialProps 3. app.render 4. page.render

## 2. 프로젝트 설정

### 프로젝트 폴더 구조

```
-public                                # static resource root
  | locales                            # 다국어 message.json
  | styles                            # css + images
-
-server                               # custom server
  | index.ts
-
-src                                  # source root
  | @types                            # type declaration
  | components                        # components
  | constants                        # 상수
  | hooks                            # custom hooks
  | libs                             # deps library custom
  | pages                            # next.js page routing
  |   | api                          # next.js api routing
  |   | auth                         # 로그인 관련
  | service                          # API 호출
  | stores                           # recoil 상태관리
  | styles                           # material ui theme 관리
  | utils                            # utils
-
-test                                # test 관련
-
.babelrc                             # babel config
.dockerignore                         # docker ignore
.env.local                           # environment variables
.eslintrc.js                         # eslint config
.gitignore                           # git ignore
.prettierrc.js                       # prettier config
Dockerfile                           # Docker 배포 시
jest.config.js                       # jest config
jest.setup.ts                         # jest에서 testing-library 사용하기 위한 설정(그외 jest에 필요한 외부 라이브러리 설정)
manifest.yml                          # cf 배포 시
next-env.d.ts                        # next.js type declare
next.config.js                       # next.js 설정
package.json
README.md
tsconfig.json                        # typescript config
tsconfig.server.json                 # custom server 사용 시 typescript config
```

### ● 환경 설정

CORS, 환경변수, 다국어, webpack 등의 환경 설정 파일

next.config.js

```
const { i18n } = require('./next-i18next.config')
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',})
const withPlugins = require('next-compose-plugins')
const plugins = [[withBundleAnalyzer]]

const nextConfig = {
  i18n,
  env: {SERVER_API_URL: process.env.SERVER_API_URL,
    PROXY_HOST: process.env.PROXY_HOST, ... 환경변수 정의},
  webpack: (config, { webpack }) => {... Webpack 설정},
  async rewrites() {
    return [{
      source: '/server/:path*',
      destination: `${serverApiUrl}/:path*`,},],
  },
}

module.exports = withPlugins(plugins, nextConfig)
```

## 2. 프로젝트 설정

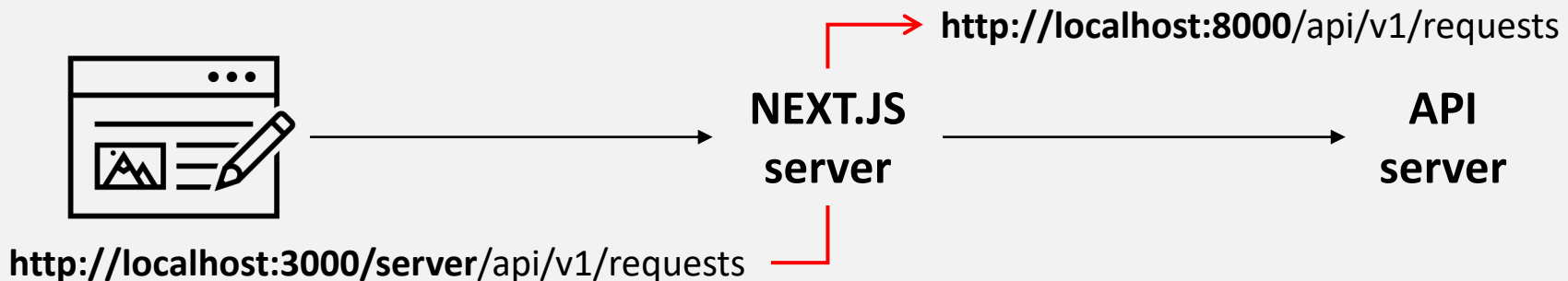
### CORS 설정

\_app

```
...
axios.defaults.baseURL = `${process.env.CLINET_URL}/server`
axios.defaults.withCredentials = true
...
```

next.config.js

```
...
async rewrites() {
  return [{
    source: '/server/:path*',
    destination: `${process.env.SERVER_API_URL}/:path*`,
  }]
},
...
```



### Routing

`Concept of Pages` 를 바탕으로 file-system 기반 Router

#### Index routes

- pages/index.tsx → /
- pages/user/index.tsx → /user

#### Nested routes

- pages/user/password/index.tsx → /user/password
- pages/user/leave/bye.tsx → /user/leave/bye

#### Dynamic route

- pages/board/[category]/view/index.tsx → /board/:category/view  
(e.g. /board/notice/view)
- pages/content/[id].tsx → /content/:id (e.g. /content/intro)
- pages/post/[...all].tsx → /post/\* (e.g. /post/2020/id/title)

## 2. 프로젝트 설정

### Styled-jsx (css-in-js)

#### GlobalStyles.tsx

```
import { ASSET_PATH } from '@constants/env'
import React from 'react'

export interface IGlobalStyleProps {
  children: React.ReactNode
}

const GlobalStyles = ({ children }: IGlobalStyleProps) => {
  return (
    <div>
      {children}
      <style jsx global>{@import `${ASSET_PATH}/layout.css`;}</style>
    </div>
  )
}

export default GlobalStyles
```

#### \_app

```
...
<GlobalStyles>
  <App component={Component} {...pageProps} />
</GlobalStyles>
...
```

## 2. 프로젝트 설정

### 🌀 API 호출 (with. **AXIOS**)

폴리필없이 구형 브라우저 지원

통신 시 필요한 기본값 설정

JSON 데이터 자동변환

CSRF 보호 기능 내장

\_app

```
...
axios.defaults.headers.common[CUSTOM_HEADER_SITE_ID_KEY] = SITE_ID
axios.defaults.baseURL = BASE_URL
axios.defaults.withCredentials = true
...
```

Login 후 token 세팅

```
const onSuccessLogin = (result: any) => {
  axios.defaults.headers.common[CLAIM_NAME] = result[ACCESS_TOKEN]
  axios.defaults.headers.common[AUTH_USER_ID] = result[AUTH_USER_ID]
}
```

Axios 사용시

```
try {
  const result = await axios.get(url)
  if (result) {
    setData(result.data)
  }
} catch (error) {
  setErrorState({ error })
}
```



### 🕒 Data Fetching (with. SWR)

SWR ? 원격 데이터 fetch를 위한 커스텀 혹은 npm 모듈

#### Data fetch

```
const fetcher = async (url: string, param: {}) => {  
  const res = await axios.get(url, {  
    params: param,  
  })  
  return res.data  
}  
  
const { data, mutate, error } = useSWR('/api/v1/fetch', fetcher)
```

1. 첫번째 인자로 원격 상태에 대한 key, 두번째 인자로 데이터 fetch 함수를 받습니다.
2. 첫번째 인자는 fetch 함수의 인자로 전달됩니다.
3. Fetch 함수가 데이터를 로드하면 해당 응답이 `data`로 세팅되고 오류 발생 시 해당 오류가 `error`에 세팅됩니다.
4. 컴포넌트에서는 `data`와 `error` 상태에 따라 알맞게 렌더링을 해주면 됩니다.
5. `mutate` 함수를 제공해 주는데 해당 함수를 통해 상태를 즉시 fetch하고 데이터를 갱신할 수 있습니다.

### 🕒 Error Handling – custom error page (\_404, \_error)

pages 경로 아래 `\_error` 혹은 `\_404`, `\_500` 페이지 생성하면 각 에러 코드에 맞게 라우팅

/pages/\_error/index.tsx

```
import React from 'react'
import { NextPageContext } from 'next'
import CustomErrorPage from '@components/Errors'

const Error = ({ statusCode }) => {
  return <CustomErrorPage statusCode={statusCode} />
}

Error.getInitialProps = ({ res, err }: NextPageContext) => {
  const statusCode = res ? res.statusCode : err ? err.statusCode : 404

  return { statusCode }
}

export default Error
```

## ● Error Handling – GlobalErrorComponent (with. 상태관리)

Wrapper

suspense

Component



setErrorState({error})

Global Error Component



## 🌀 Error Handling – GlobalErrorComponent (with. 상태관리)

### component

```
try {
  const result = await axios.get(url)
  if (result) {
    setData(result.data)
  }
} catch (error) {
  setErrorState({ error })
}
```

### Global Error Component

```
const [errorState, setErrorState] = useRecoilState(errorStateAtom)
...
useEffect(() => {
  if (errorState.error) {
    ...
    setAlertState({
      open: true,
      errors,
    })
    ...
  }
}, [errorState])
...
return (
  <CustomAlert
    open={alertState.open} title={errorState.message}
    contentText={alertState.errors} />
)
```

## 2. 프로젝트 설정

### npm scripts

package.json 파일의 scripts 항목에 명령어를 설정해 두고 `npm run 명령어` 로 실행

package.json

```
...
"scripts": {
  "dev": "ts-node --project tsconfig.server.json server/index.ts",
  "build:server": "tsc --project tsconfig.server.json",
  "build:next": "next build",
  "prebuild": "rimraf ./build",
  "build": "NODE_ENV=production npm run build:next && npm run build:server",
  "start": "NODE_ENV=production node build/index.js",
},
...
```

대규모 포털 script

```
"dev:lg:win": "set SITE_ID=2&&npm run dev",
"dev:lg": "SITE_ID=2 npm run dev",
"build:prodlg": "env-cmd -f ./env.production-lg npm run build:next && npm run build:server",
"start:prodlg": "env-cmd -f ./env.production-lg node build/index.js",
```

소규모 포털 script

```
"dev:sm:win": "set SITE_ID=3&&npm run dev",
"dev:sm": "SITE_ID=3 npm run dev",
"build:prodsm": "env-cmd -f ./env.production-sm npm run build:next && npm run build:server",
"start:prodsm": "env-cmd -f ./env.production-sm node build/index.js",
```

# 3. 개발환경 설치

## ● 개발 환경

- node : 14.8.0
- npm : 6.14.7

## ● IDE

- Visual Studio Code : latest

# 3. 개발환경 설치


## Windows - Node

<https://nodejs.org/en/download/> > 공식 사이트에서 다운로드

LTS  
Recommended For Most Users

Current  
Latest Features

  
Windows Installer  
node-v14.18.1-x86.msi

  
macOS Installer  
node-v14.18.1.pkg

  
Source Code  
node-v14.18.1.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	

cmd

```
# version 확인  
node -v  
npm -v
```

# 3. 개발환경 설치

## MacOS

터미널에서 Homebrew(macOS용 패키지 관리자)로 설치

terminal

```
brew install node@14.8.0  
# version 확인  
node -v  
npm -v
```

Homebrew 가 없는 경우 Homebrew를 먼저 설치

terminal

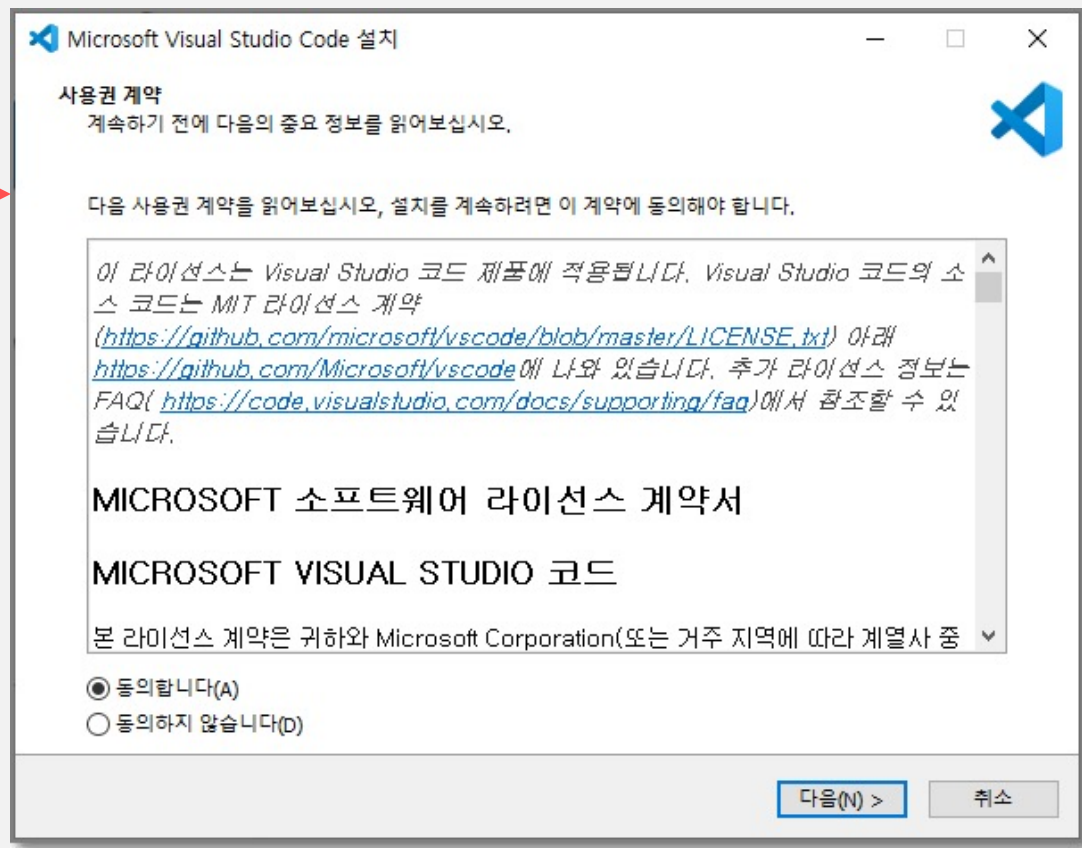
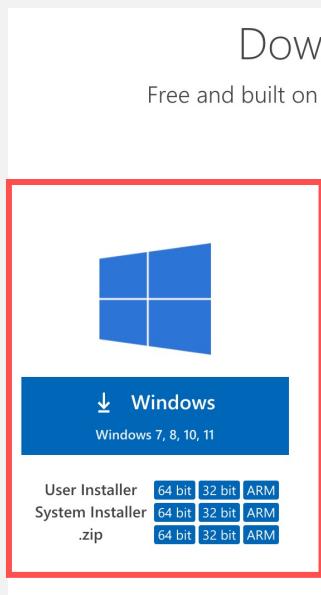
```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```



### 3. 개발환경 설치

#### Visual Studio Code - Windows

<https://code.visualstudio.com/download> > 공식홈페이지에서 다운로드

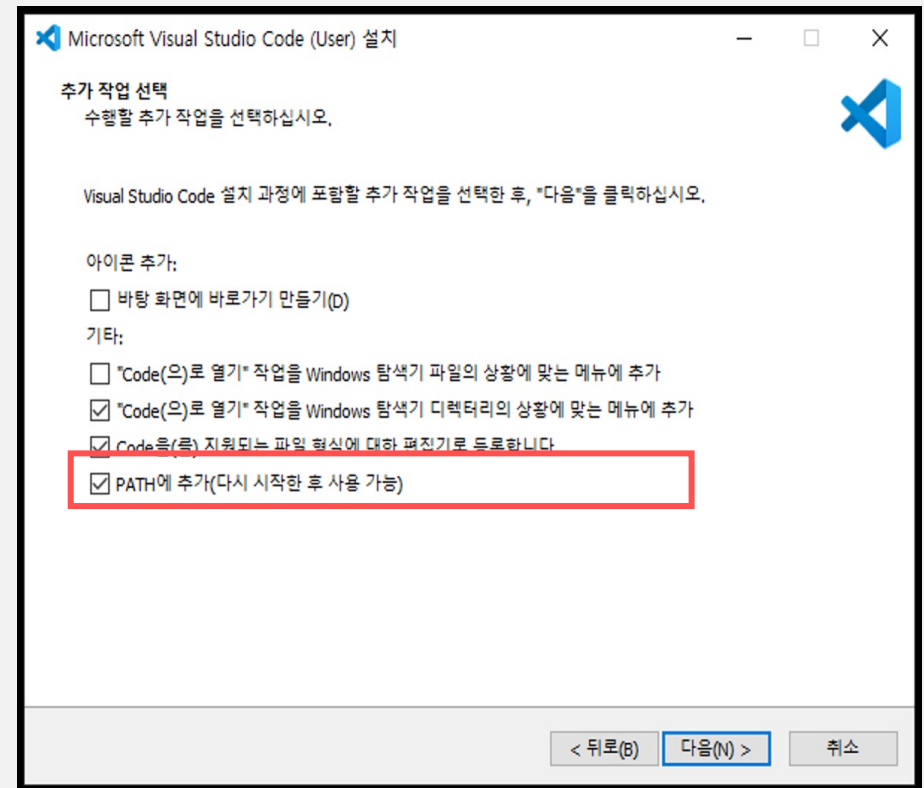
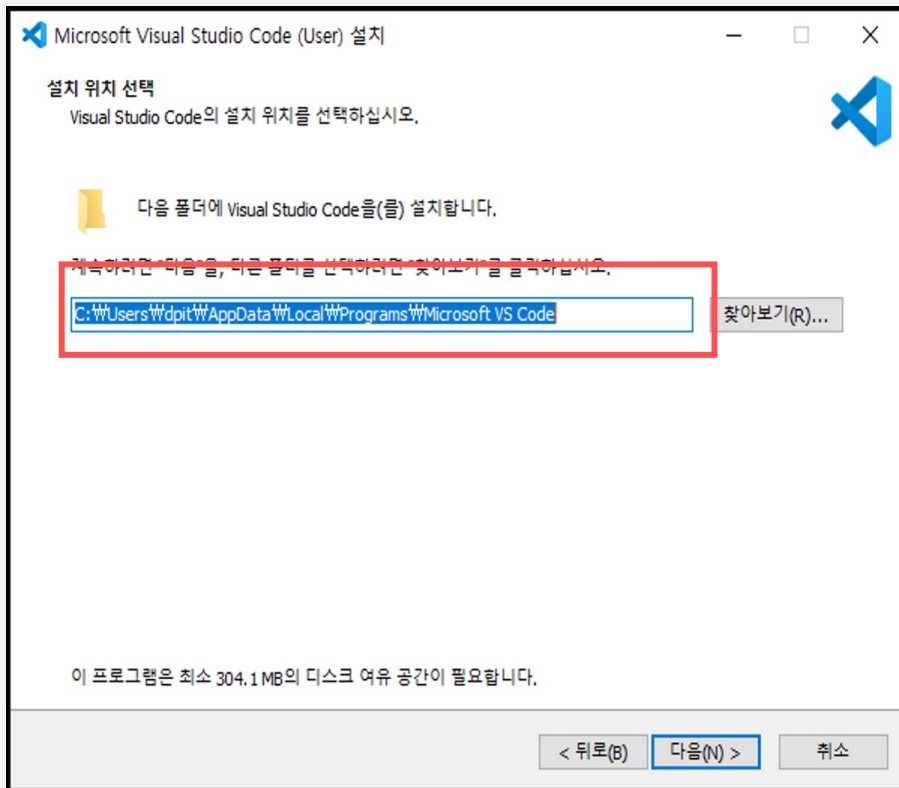


### 3. 개발환경 설치

#### Visual Studio Code - Windows

설치경로 > c:\Users\[USER]\AppData\Local\Programs\Microsoft VS Code

추가 작업 선택 > 'PATH에 추가'는 필수적으로 선택하는 것을 추천

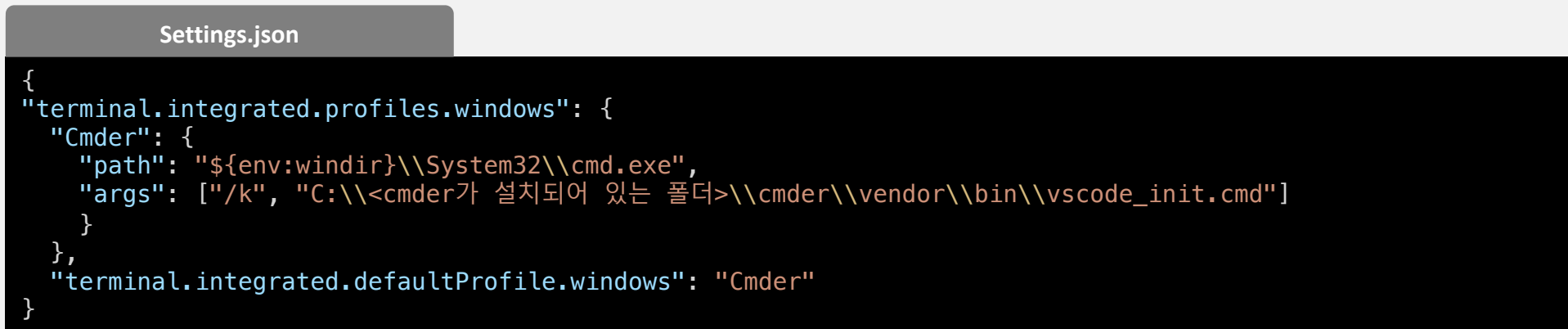
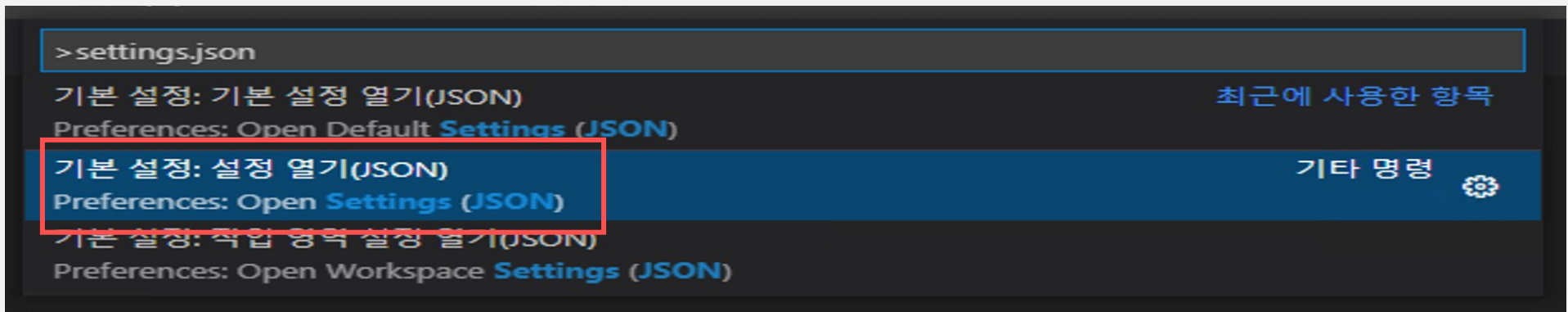


### 3. 개발환경 설치

#### Visual Studio Code - Windows

VSCoide 실행 후 Ctrl + P > settings.json

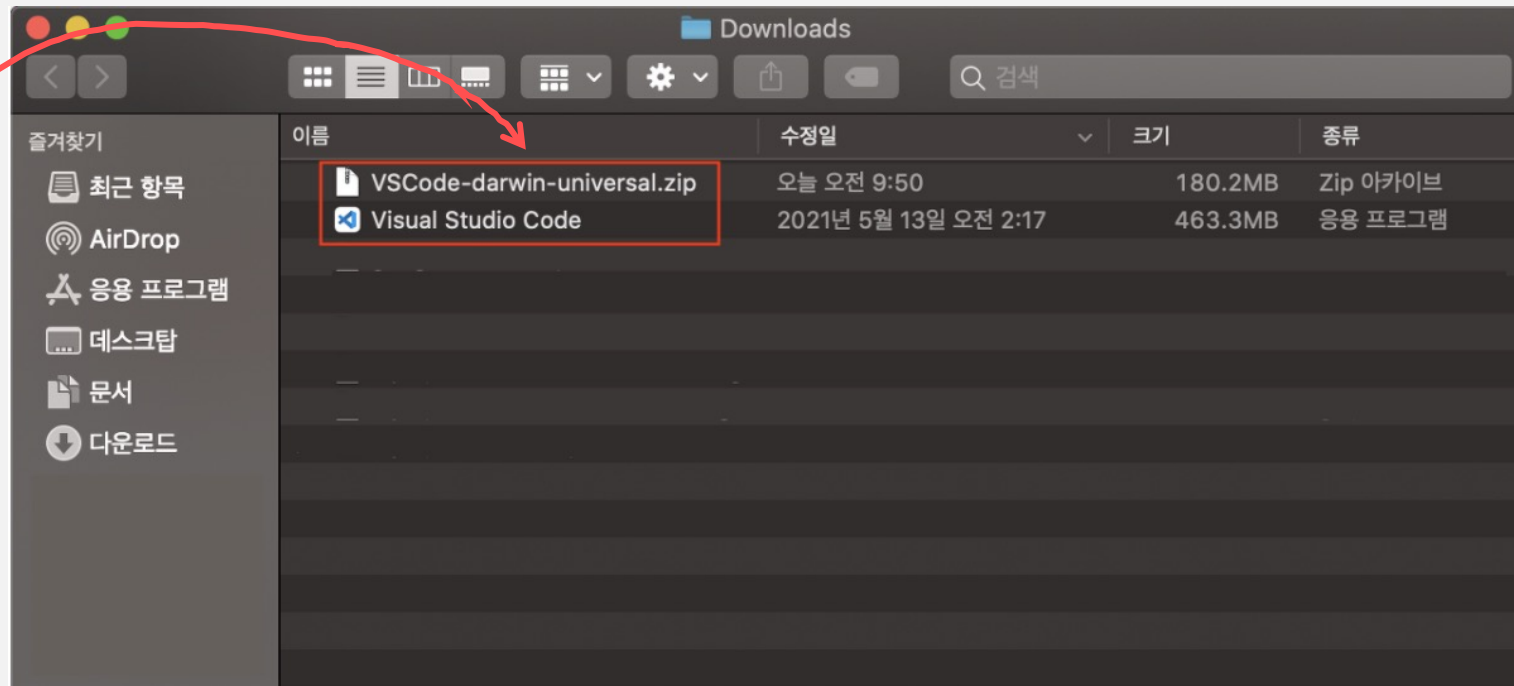
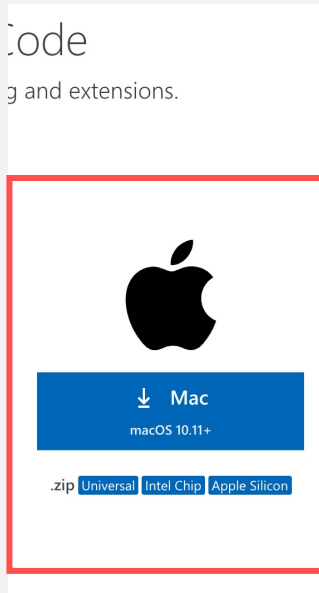
VSCoide 터미널 Cmder로 설정



### 3. 개발환경 설치

#### Visual Studio Code - MacOS

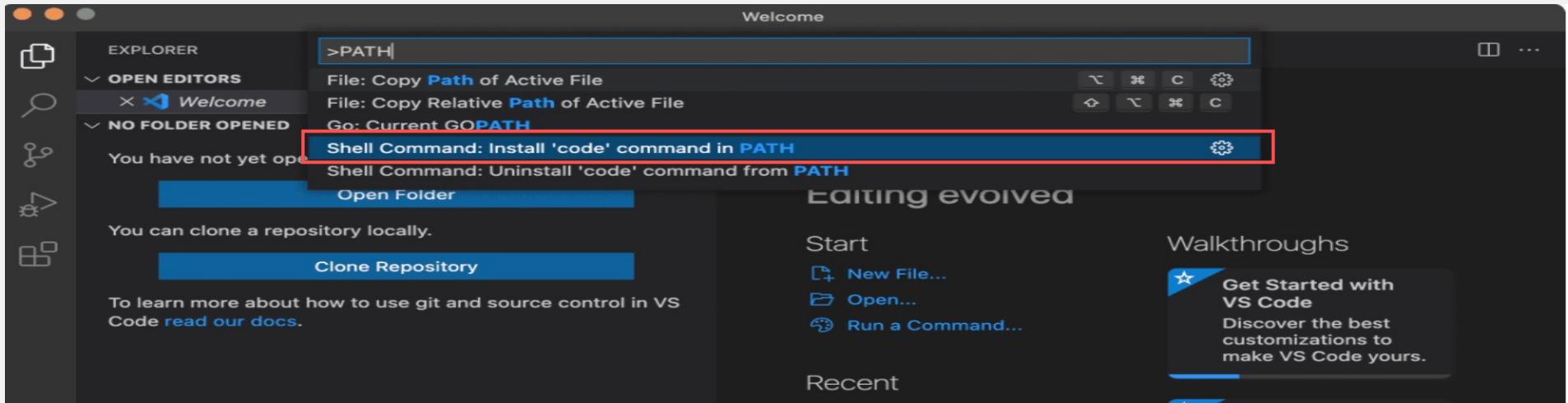
<https://code.visualstudio.com/download> > 공식홈페이지에서 다운로드



### 3. 개발환경 설치

#### Visual Studio Code - MacOS

- VSCode를 열고 command+shift+p 를 입력해 커맨드 팔레트를 엽니다.
- PATH로 검색해서 다음 명령을 실행합니다.



- 터미널에서 'code' 명령어를 통해 VSCode를 실행할 수 있습니다.

```
terminal
code ./egovframe-msa-edu/frontend/portal
# 또는
cd /egovframe-msa-edu/frontend/portal
code .
```

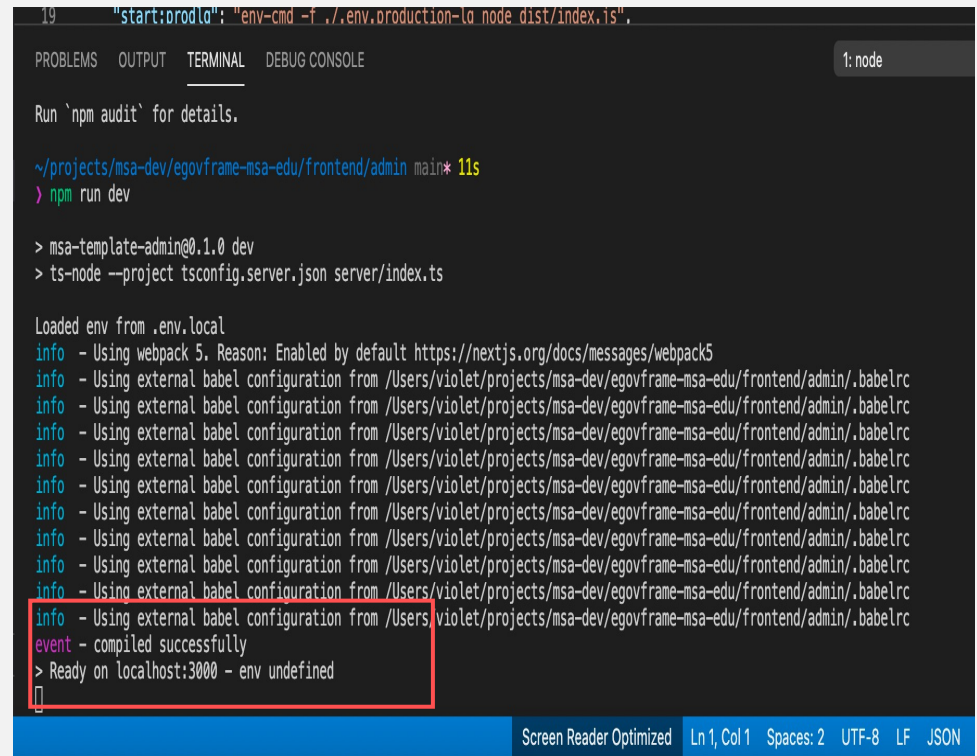
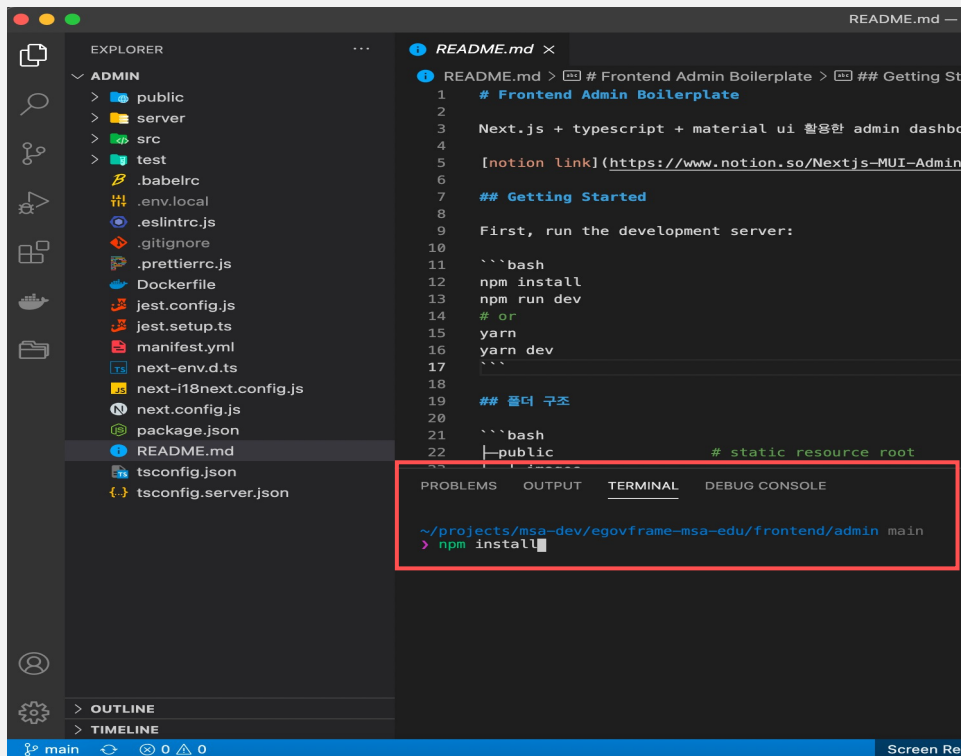
# 4. 프로젝트 실행

MSA 템플릿

## 관리자 시스템 실행

cmdr

```
code ~/workspace.edu/egovframe-msa-edu/frontend/admin
# vscode terminal 에서 실행
npm install
npm run dev
```



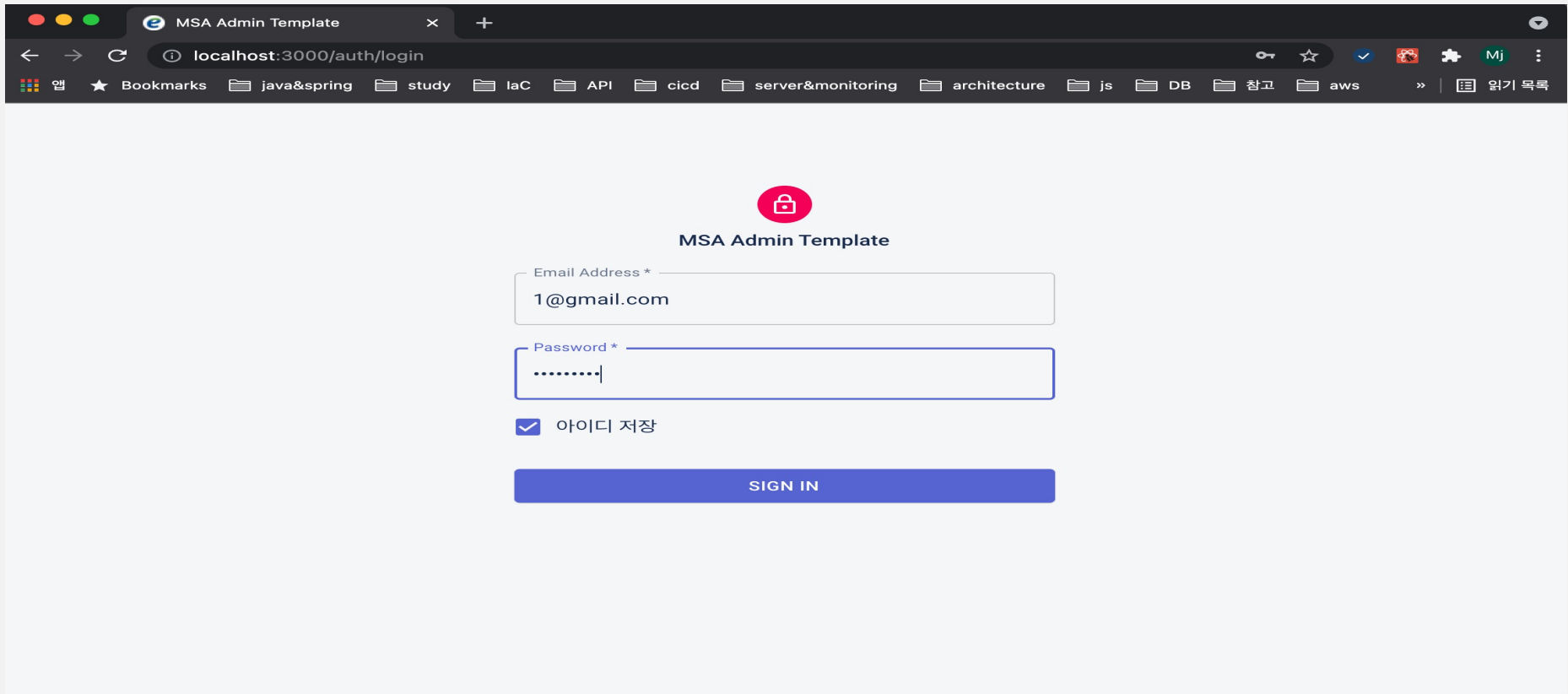
## 4. 프로젝트 실행

MSA 템플릿

### 관리자 시스템 실행

Email : 1@gmail.com

Password: test1234!



The screenshot shows a web browser window with the title "MSA Admin Template". The address bar displays "localhost:3000/auth/login". The browser's bookmark bar includes "앱", "Bookmarks", "java&spring", "study", "laC", "API", "cicd", "server&monitoring", "architecture", "js", "DB", "참고", and "aws". The login page features a red lock icon at the top, followed by the text "MSA Admin Template". Below this, there are two input fields: "Email Address \*" containing "1@gmail.com" and "Password \*" containing ".....". A checkbox labeled "아이디 저장" (Save ID) is checked. At the bottom, there is a blue "SIGN IN" button.

## 4. 프로젝트 실행

MSA 템플릿

### ● 배너 추가 해보기

서비스 관리 > 배너 관리 > + 버튼 클릭

MSA Admin <

서비스 관리 ^

컨텐츠 관리

게시판 관리

배너관리

첨부파일 관리

설정관리 v

통계 관리 v

예약 관리 v

배너관리

shinmj@gmail.com v

Home > 서비스 관리 > 배너관리

전체 v

배너 제목 v

Search...

Q

+

번호	사...	배너 구분	배너 제목	사용 여부	등록 일시	관리
1	대규모 ...	메인배너	표준 프레임워크 ...	<input checked="" type="checkbox"/>	2021-10-12 16:41:51	<a href="#">수정</a> <a href="#">삭제</a>
2	대규모 ...	메인배너	표준프레임워크 ...	<input checked="" type="checkbox"/>	2021-10-12 16:43:00	<a href="#">수정</a> <a href="#">삭제</a>
3	소규모 ...	메인배너	'21년 9차 표준프...	<input checked="" type="checkbox"/>	2021-08-20 18:22:33	<a href="#">수정</a> <a href="#">삭제</a>
4	소규모 ...	메인배너	2021 표준프레임...	<input checked="" type="checkbox"/>	2021-08-20 18:22:58	<a href="#">수정</a> <a href="#">삭제</a>
5	소규모 ...	메인배너	'21년 8차 표준프...	<input checked="" type="checkbox"/>	2021-09-06 10:32:34	<a href="#">수정</a> <a href="#">삭제</a>

[<](#) [<](#) [1](#) [>](#) [>](#)



## 4. 프로젝트 실행

MSA 템플릿

### ● 배너 추가 해보기

정보 입력 > 파일 입력 > 저장

- 사이트 : 대규모 포털
- 배너 구분: 메인배너
- 배너 제목: 표준프레임워크 MSA 포털
- 파일 선택
- URL :  
<https://www.egovframe.go.kr/home/ntt/nttRead.do?menuNo=74&bbsId=6&nttId=1829>
- 배너 내용 : MSA(MicroService Architecture) 템플릿을 제공합니다

배너관리

shinmj@gmail.com

Home > 서비스 관리 > 배너관리

사이트 \*

대규모 포털

배너 구분 \*

메인배너

배너 제목 \*

표준 프레임워크 MSA 포털

1 개의 파일이 선택되었습니다.

파일찾기

intro\_img02.png  
7.32 KB

URL

https://www.egovframe.go.kr/home/ntt/nttRead.do?

새 창 여부

배너 내용

MSA(MicroService Architecture) 템플릿을 제공합니다

정렬 순서 \*

1

목록 저장

## 4. 프로젝트 실행

### 🕒 사용자 시스템 실행

cmdr

```
code ~/workspace.edu/egovframe-msa-edu/frontend/portal  
# vscode terminal 에서 실행  
npm install  
npm run dev
```

포트, 사이트 id 환경변수 추가

(대규모 포털 : SITE\_ID=2, 소규모 포털 : SITE\_ID=3)

📁 .env.local

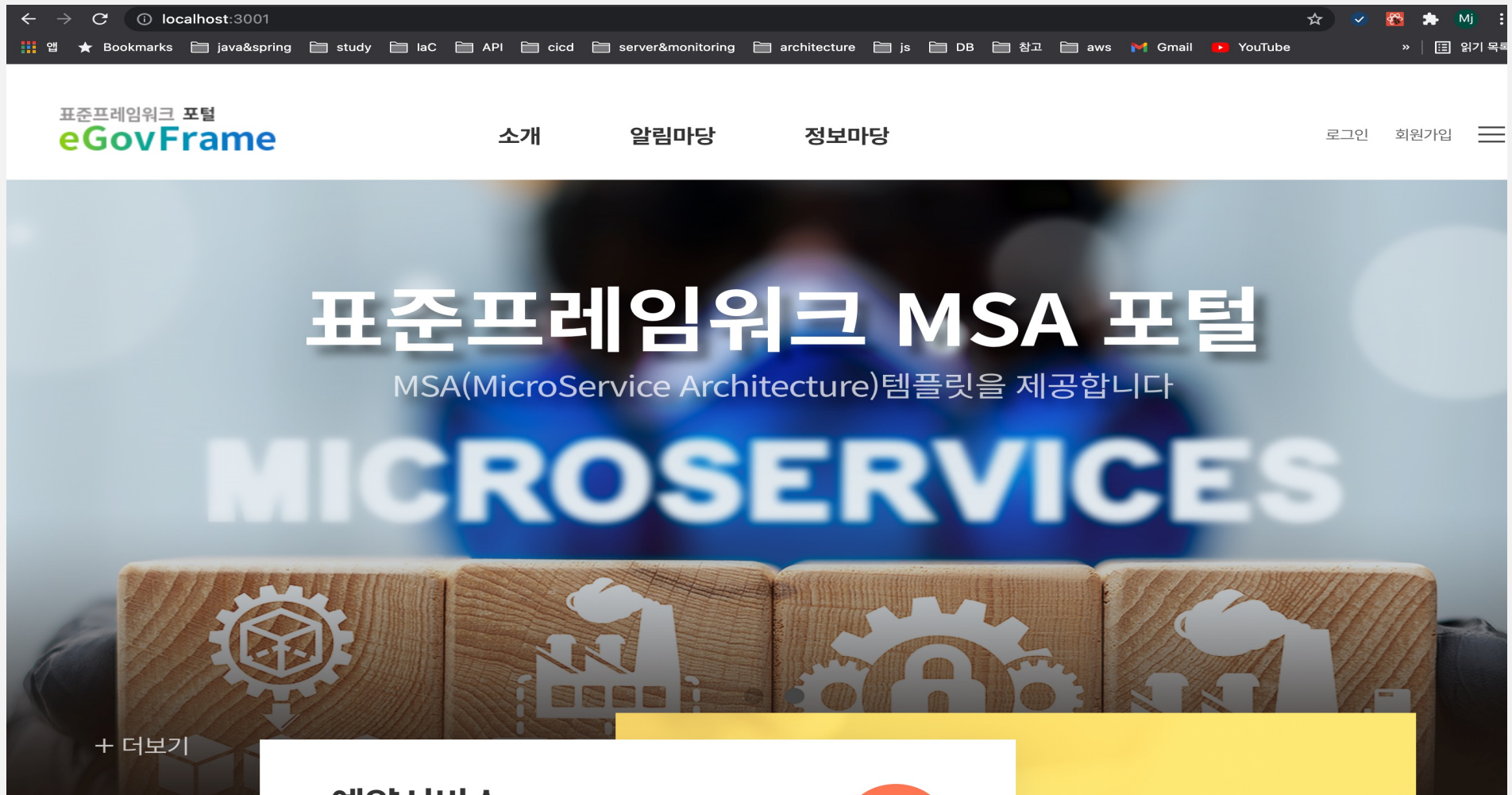
```
1  NEXT_PUBLIC_TEST="DEV"  
2  
3  PORT=3001  
4  SITE_ID=2  
5  
6  
7
```

```
> npm run dev:lg  
  
> msa-template-portal@0.1.0 dev:lg  
> SITE_ID=2 npm run dev  
  
> msa-template-portal@0.1.0 dev  
> ts-node --project tsconfig.server.json server/index.ts  
  
Loaded env from .env.local  
info - Using webpack 5. Reason: Enabled by default https://nextjs.org/docs/messages/webpack5  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
info - Using external babel configuration from /Users/violet/projects/msa-dev/egovframe-msa-edu/frontend/portal/.babelrc  
event - compiled successfully  
> Ready on localhost:3001 - env undefined
```

## 4. 프로젝트 실행

MSA 템플릿

### 🕒 사용자 시스템 배너확인



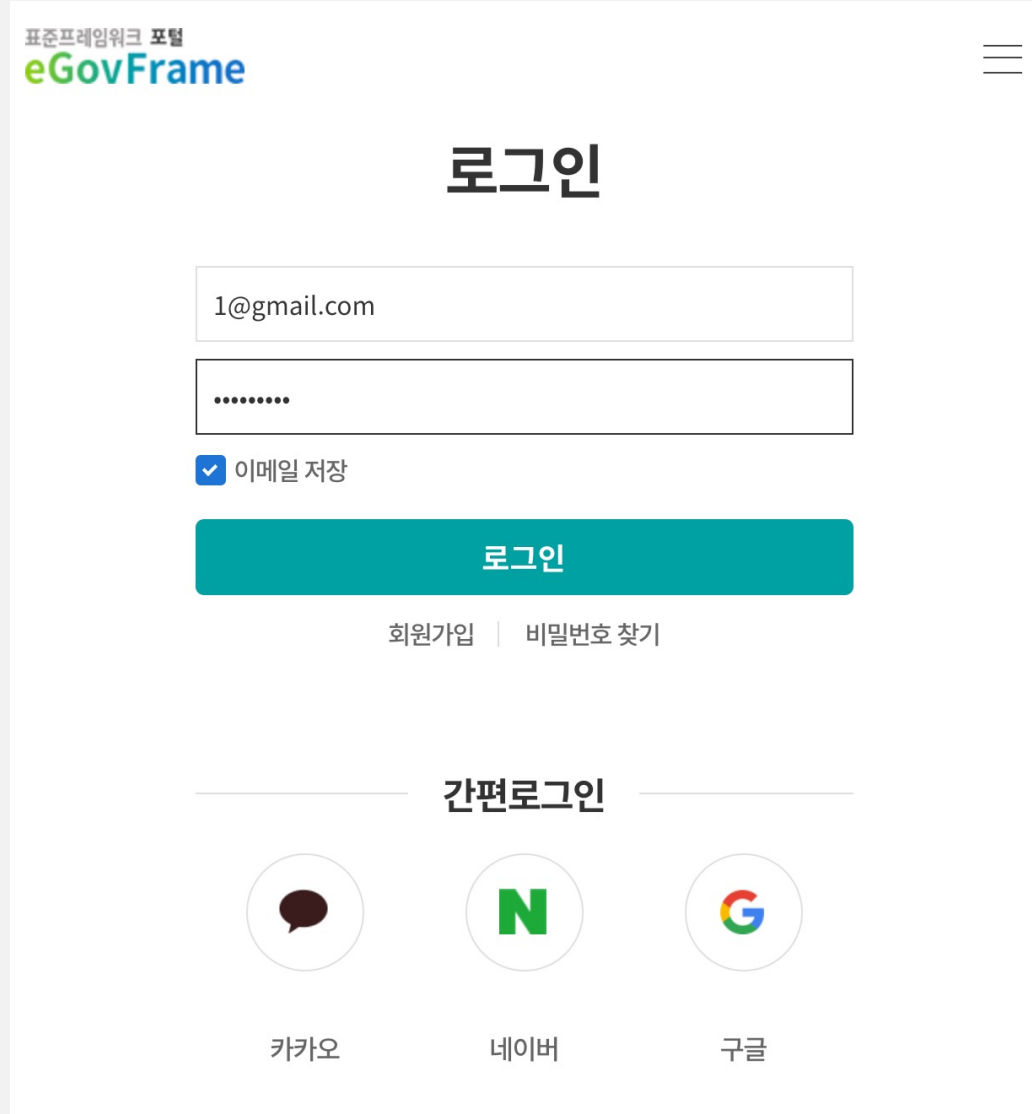
## 4. 프로젝트 실행

MSA 템플릿

### ● 사용자 시스템 로그인

Email : 1@gmail.com

Password: test1234!



표준프레임워크 포털  
eGovFrame

# 로그인

1@gmail.com

.....

☒ 이메일 저장

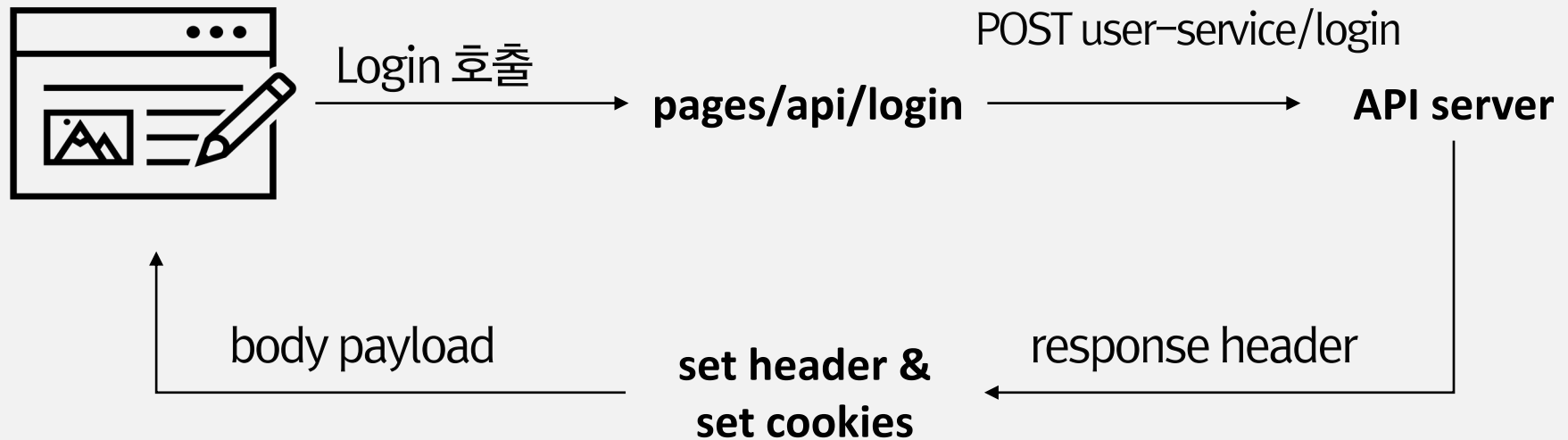
로그인

회원가입 | 비밀번호 찾기

### 간편로그인

카카오 네이버 구글

## ● 사용자 시스템 로그인



service/Login.ts

```
...
axios.defaults.headers.common[CLAIM_NAME] = result[ACCESS_TOKEN]
axios.defaults.headers.common[AUTH_USER_ID] = result[AUTH_USER_ID]
...
```

## 4. 프로젝트 실행

### 소셜 로그인 - 구글

- 구글 Developers Console > <https://console.cloud.google.com/>
- 새 프로젝트 추가 > API 및 서비스 > OAuth 동의 화면 추가, 범위(email, profile) 선택
- API 및 서비스 > 사용자 인증 정보 > OAuth 2.0 클라이언트 ID 추가, App Key 생성

앱 등록 수정

✓ OAuth 동의 화면 — ② 범위 — ③ 테스트 사용자 — ④ 요약

범위는 사용자에게 앱 승인을 요청하는 권한을 나타내며 프로젝트에서 사용자의 Google 계정에 있는 특정 유형의 비공개 사용자 데이터에 액세스하도록 허용합니다.  
[자세히 알아보기](#)

범위 추가 또는 삭제

민감하지 않은 범위

API ↑	범위	사용자에게 표시되는 설명	
	.../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인	🗑️
	.../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기	🗑️

Google Cloud Platform msa-template

API API 및 서비스

사용자 인증 정보 + 사용자 인증 정보 만들기 삭제

대시보드 라이브러리 사용자 인증 정보 OAuth 동의 화면 도메인 확인 페이지 사용 동의

사용자 인증 정보

사용 설정한 API에 액세스하려면 사용자 인증 정보를 만드세요. [자세히 알아보기](#)

API 키

이름	생성일 ↓	제한사항	키	작업
표시할 API 키가 없습니다.				

OAuth 2.0 클라이언트 ID

이름	생성일 ↓	유형	클라이언트 ID	작업
msa-template	2021. 10. 22.	웹 애플리케이션		🗑️ ⚙️

서비스 계정 [서비스 계정 관리](#)

이메일	이름 ↑	작업
표시할 서비스 계정이 없습니다.		

## 4. 프로젝트 실행

### 소셜 로그인 - 구글

- 도메인 및 Redirect URL 입력 > 확인
- 클라이언트 ID, 클라이언트 보안 비밀 번호 확인

← 웹 애플리케이션의 클라이언트 ID    JSON 다운로드    보안 비

승인된 자바스크립트 원본 ?

브라우저 요청에 사용

URI \*

http://localhost:3001

+ URI 추가

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

URI \*

http://localhost:3001/auth/login

+ URI 추가

← 웹 애플리케이션의 클라이언트 ID    JSON 다운로드    보안 비밀 재설정    삭제

이름 \*

mss-template

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.

클라이언트 ID

클라이언트 보안 비밀

생성일

0 아래에 추가한 URI의 도메인이 승인된 도메인으로 OAuth 등의 화면에 자동으로 추가됩니다.

## 4. 프로젝트 실행

### 소셜 로그인 - 구글

#### 프론트엔드 설정

.env.local

```
...  
NEXT_PUBLIC_GOOGLE_CLIENT_ID=<google-client-id>  
...
```

#### 백엔드 설정

application-oauth.yml

```
spring  
  security  
    oauth2  
      client  
        registration  
          google  
            client-id <google-client-id>  
            client-secret <google-client-secret-key>  
            scope profile,email
```



## 4. 프로젝트 실행

### 소셜 로그인 - 카카오

- 카카오 Developers > <https://developers.kakao.com/>
- 애플리케이션을 추가하여 app-key 생성
- Web 플랫폼을 추가하고 사이트 도메인 설정

APP

**msa-template**

ID 655421 OWNER

**앱 키**

네이티브 앱 키	
REST API 키	
JavaScript 키	
Admin 키	

**Web**

사이트 도메인	http://localhost:3001
---------	-----------------------

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

## 4. 프로젝트 실행

MSA 템플릿

### 소셜 로그인 - 카카오

- 카카오 로그인 활성화 > Redirect URI 설정
- 카카오 로그인 > 동의 항목 중 닉네임과 이메일을 동의 받도록 설정
- 카카오 로그인 > 보안 > Client Secret 코드 생성, 활성화 사용 설정

**카카오 로그인** ON

**활성화 설정**

상태 ON

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.  
상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

**Redirect URI**

Redirect URI	http://localhost:3001/auth/login
--------------	----------------------------------

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

**카카오 로그인** ON

**개인정보**

항목 이름	ID	상태
닉네임	profile_nickname	<span>필수 동의</span> <span>설정</span>
프로필 사진	profile_image	<span>사용 안함</span> <span>설정</span>
카카오톡 채널 추가 상태 및 내역	plusfriends	<span>권한 없음</span>
카카오계정(이메일)	account_email	<span>필수 동의</span> <span>설정</span>
성별	gender	<span>사용 안함</span> <span>설정</span>

**Client Secret**

토큰 발급 시, 보안을 강화하기 위해 Client Secret을 사용할 수 있습니다. (REST API인 경우에 해당)

코드	<span>재발급</span>
활성화 상태	<span>사용함</span> <span>설정</span>

## 소셜 로그인 - 카카오

프론트엔드 설정 – Javascript key

.env.local

```
...  
NEXT_PUBLIC_KAKAO_JAVASCRIPT_KEY=<kakao-javascript-app-key>  
...
```

백엔드 설정 – REST API Key, Client Secret

application-oauth.yml

```
spring  
  security  
    oauth2  
      client  
        registration  
          kakao  
            client-id <kakao-rest-api-app-key>  
            client-secret <kakao-client-secret-key>  
            redirect-uri "{baseUrl}/{action}/oauth2/code/{registrationId}"  
            client-authentication-method POST  
            authorization-grant-type authorization  
            scope profile_nickname, account_email  
            client-name Kakao
```

## 4. 프로젝트 실행

### ● 소셜 로그인 - 네이버

- 네이버 Developers > <https://developers.naver.com/apps/>
- 애플리케이션 > 애플리케이션 등록 > 사용 API(네아로) 선택 > 회원이름, 이메일 주소 필수 선택

사용 API ⇄

네아로 (네이버 아이디로 로그인)

제공 정보 선택(이용자 식별자는 기본 정보로 제공) ?

권한	필수	추가
회원이름	<input checked="" type="checkbox"/>	<input type="checkbox"/>
이메일 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>
별명	<input type="checkbox"/>	<input checked="" type="checkbox"/>
프로필 사진	<input type="checkbox"/>	<input type="checkbox"/>
성별	<input type="checkbox"/>	<input type="checkbox"/>
생일	<input type="checkbox"/>	<input type="checkbox"/>
연령대	<input type="checkbox"/>	<input type="checkbox"/>
출생연도	<input type="checkbox"/>	<input type="checkbox"/>
휴대전화번호	<input type="checkbox"/>	<input type="checkbox"/>

[알림] 추가권한에 대한 [네이버 로그인 공지사항](#)을 확인하세요.



## 소셜 로그인 - 네이버

- 로그인 오픈 API 서비스 환경 > PC 웹 환경 추가 > 서비스 URL, Callback URL 추가 > 등록 > Client Id, Client Secret 확인

환경 추가

PC 웹

서비스 URL

http://localhost:3001

서비스 URL 예시: (O) http://naver.com (X) http://www.naver.com  
서비스 URL 값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.  
불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.  
서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 아이디로 로그인** 배지가 노출됩니다.

네이버아이디로그인  
Callback URL (최대 5개)

http://localhost:3001/auth/login/naver

로그인 오픈 API  
서비스 환경 ①

## msa-template

개요	API 설정	네아로 검수상태	멤버
----	--------	-------------	----

## 애플리케이션 정보

Client ID

Client Secret

재발급

## 소셜 로그인 - 네이버

### 프론트엔드 설정

.env.local

```
...  
NEXT_PUBLIC_NAVER_CLIENT_ID=<naver-client-id>  
NEXT_PUBLIC_NAVER_CALLBACK_URL=http://localhost:3001/auth/login/naver  
...
```

### 백엔드 설정

application-oauth.yml

```
spring  
  security  
    oauth2  
      client  
        registration  
          naver  
            client-id <naver-client-id>  
            client-secret <naver-client-secret>  
            redirect-uri "{baseUrl}/{action}/oauth2/code/{registrationId}"  
            authorization_grant_type authorization  
            scope name,email,profile_image  
            client-name Naver
```

실 습