KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

프론트엔드 준비

[KB] IT's Your Life

KB국민은행

프론트엔드 준비

프로젝트 만들기

09 Vue+Spring₩scoula> npm init vue frontend

Vue.js – The Progressive JavaScript Framework

✓ Add TypeScript? … No / Yes
✓ Add JSX Support? … No / Yes
✓ Add Vue Router for Single Page Application development? … No / Yes
✓ Add Pinia for state management? … No / Yes
✓ Add Vitest for Unit Testing? … No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? … No / Yes
Scaffolding project in …09 Vue+Spring₩scoula₩frontend…

Done. Now run:

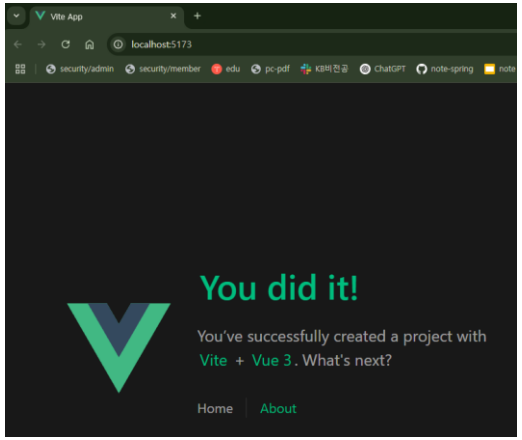
cd frontend
npm install
npm run dev

화살표로 이동
스페이스바로 선택

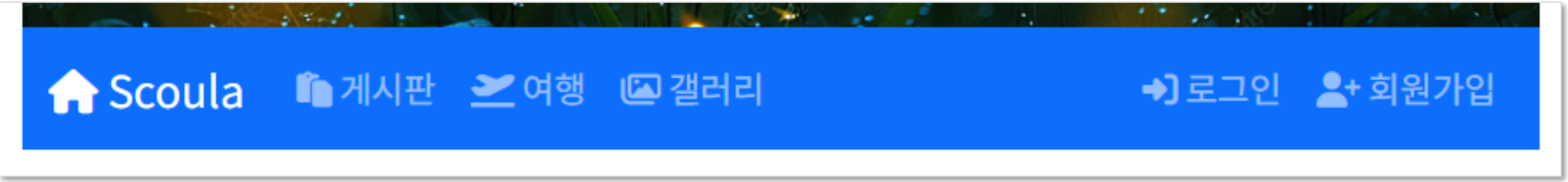
```
┌  Vue.js - The Progressive JavaScript Fr
│
◆  Select features to include in your pro
confirm)
│  ☐ TypeScript
│  ☐ JSX Support
│  ■ Router (SPA development)
│  ■ Pinia (state management)
│  ☐ Vitest (unit testing)
│  ☐ End-to-End Testing
│  ☐ ESLint (error prevention)
│  ☐ Prettier (code formatting)
```

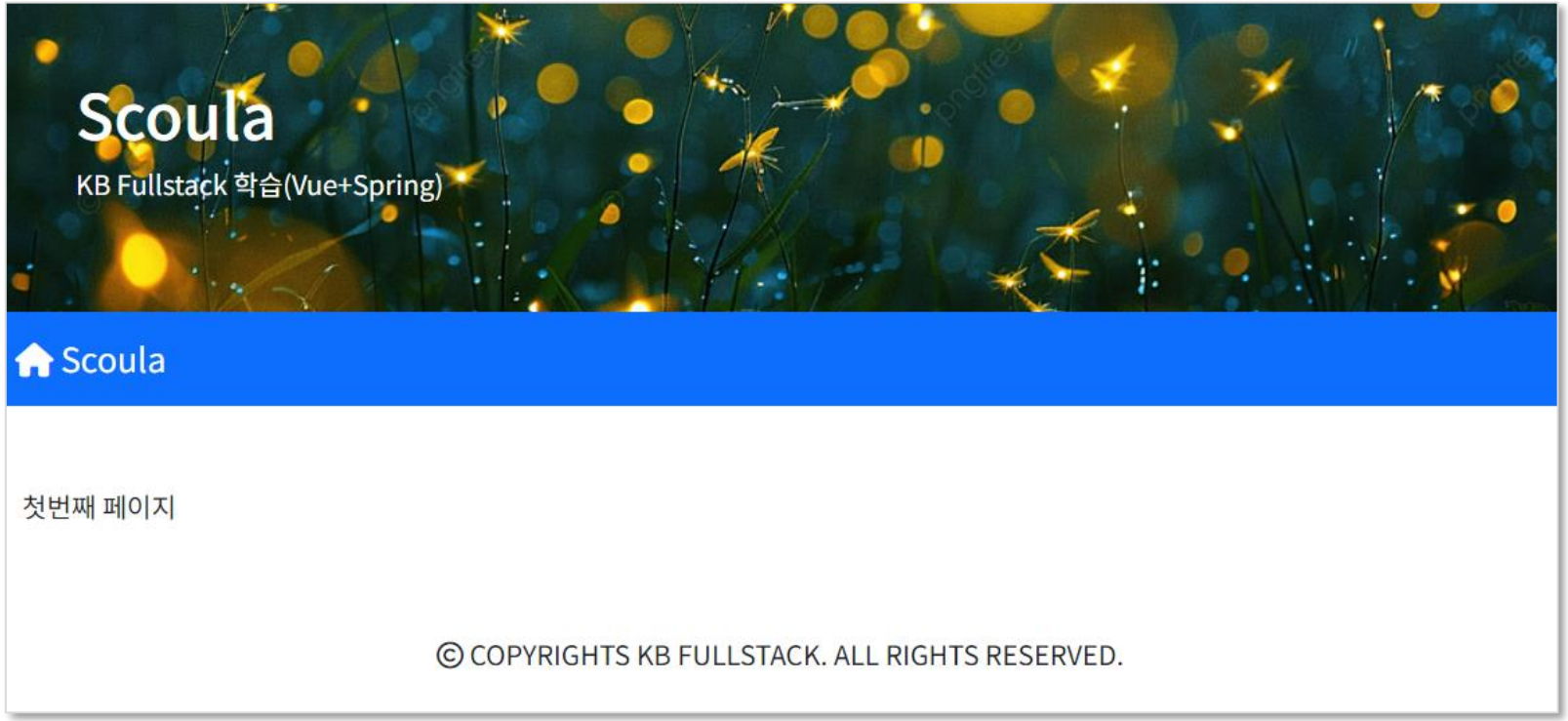
문제 출력 디버그 콘솔 터미널 포트

```
VITE v6.3.5  ready in 786 ms

➜ Local:   http://localhost:5173/
➜ Network: use --host to expose
➜ Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
➜ Vue DevTools: Press Alt(⌥)+Shift(⇧)+D in App to toggle the Vue DevTools
➜ press h + enter to show help
```



프론트엔드 준비


Scoula
게시판
여행
갤러리
로그인
회원가입


Scoula
KB Fullstack 학습(Vue+Spring)
Scoula
첫번째 페이지
© COPYRIGHTS KB FULLSTACK. ALL RIGHTS RESERVED.

프론트엔드 준비

기본 라이브러리 추가

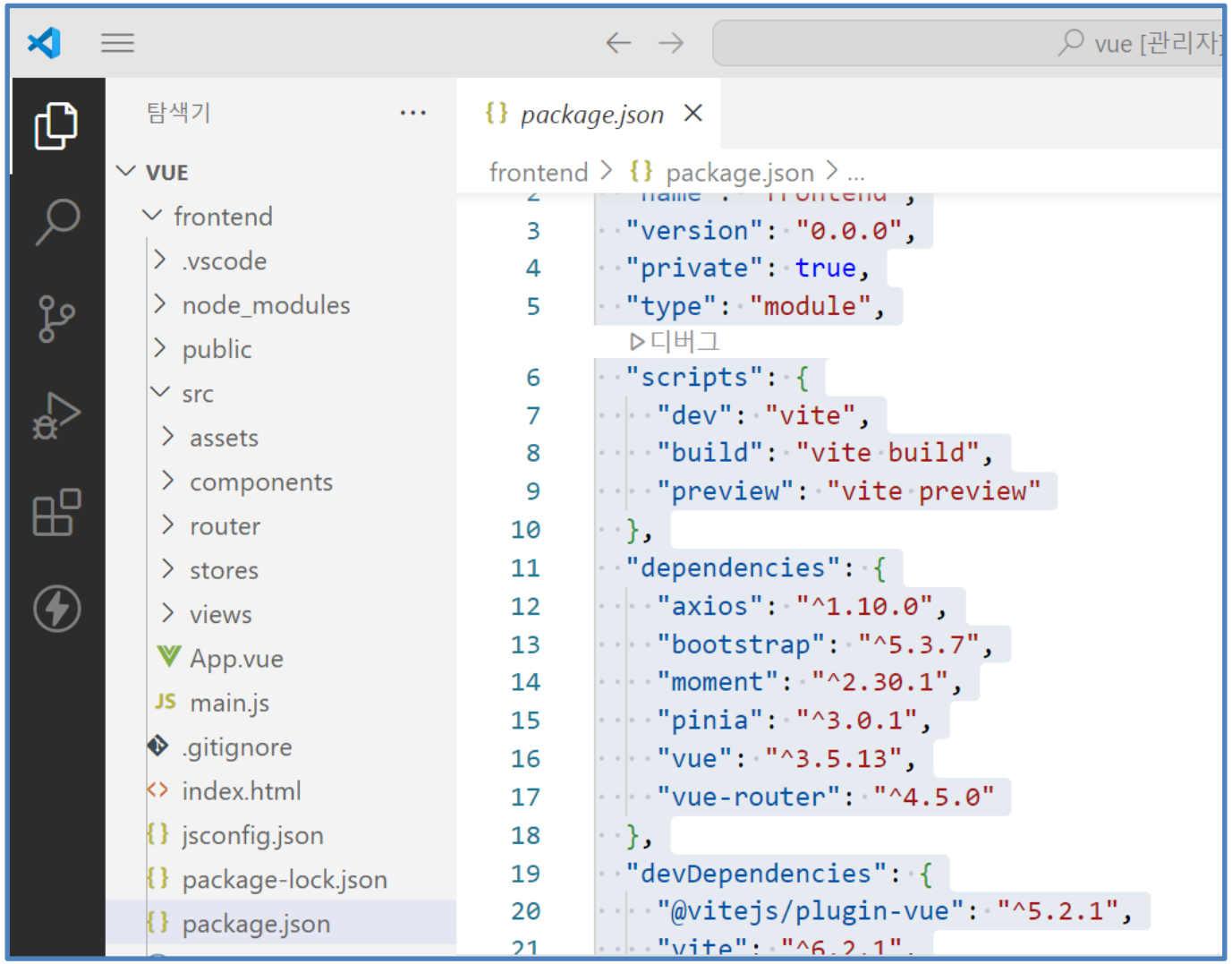
- npm install
- npm install bootstrap@5 axios moment

- bootstrap@5 : html + css + js 패키지
- axios : 비동기통신
- moment : JavaScript에서 날짜와 시간을 보다 쉽게 처리하고 포맷팅할 수 있게 도와주는 강력한 도구

index.html (fontawesom CDN 추가)

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css"
      integrity="sha512-
Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnYxTxve5UMg5GT6L4JJg=="
      crossorigin="anonymous"
      referrerpolicy="no-referrer"
    />
    <title>Scoula App</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

프론트엔드 준비

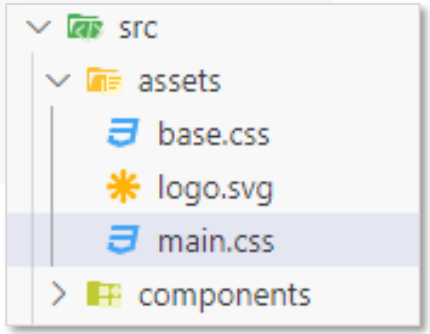



```
{
  "name": "frontend",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "axios": "^1.10.0",
    "bootstrap": "^5.3.7",
    "moment": "^2.30.1",
    "pinia": "^3.0.1",
    "vue": "^3.5.13",
    "vue-router": "^4.5.0"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^5.2.1",
    "vite": "^6.2.1",
    "vite-plugin-vue-devtools": "^7.7.2"
  }
}
```

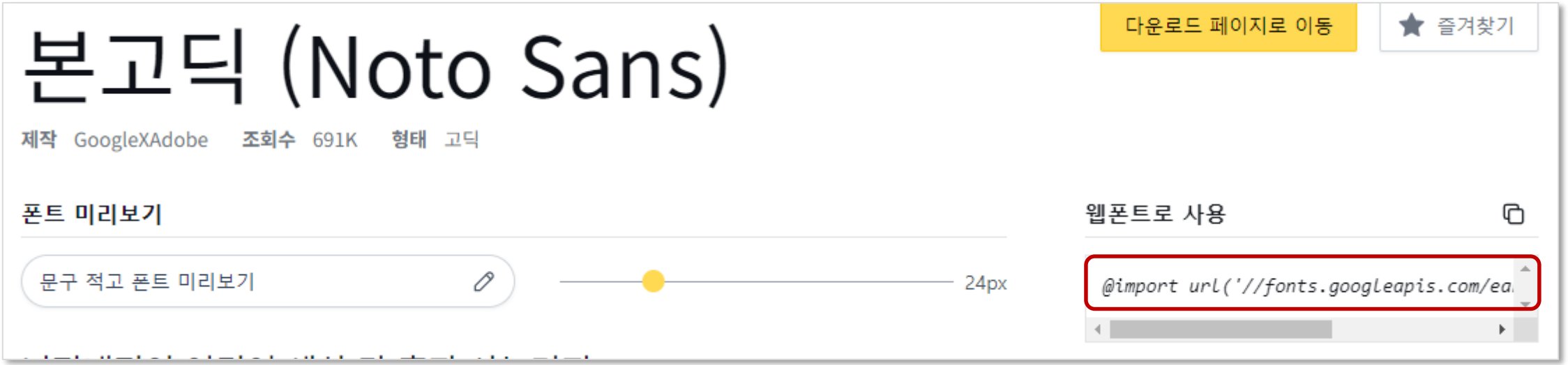
프론트엔드 준비

assets/main.css

```
@import
url('https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@100;200;300;400;500;600;700;800;900&display=swap');

* {
  font-family: 'Noto Sans KR';
}
```



- webfont 설정
  - https://noonnu.cc/ 에서 noto 검색



main.js

```
import './assets/main.css';
import 'bootstrap/dist/css/bootstrap.css';

import { createApp } from 'vue';
import { createPinia } from 'pinia';

import App from './App.vue';
import router from './router';

const app = createApp(App);
app.use(createPinia());
app.use(router);

app.mount('#app');
```

App.vue

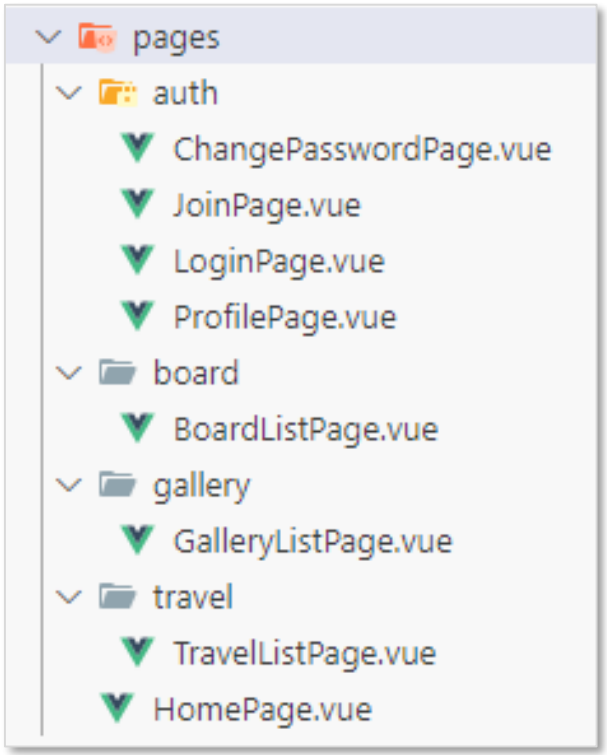
```
<!-- <script setup>은 Vue 3에서 Composition API를 더 간단하게 쓸 수 있게 해주는 문법임 -->
<script setup>
// RouterView는 vue-router에서 제공하는 컴포넌트임
// 현재 URL 경로에 따라 맞는 컴포넌트를 화면에 보여줌
import { RouterView } from 'vue-router';
</script>

<!-- 화면에 보여줄 HTML 구조 정의하는 부분임 -->
<template>
  <!-- RouterView는 라우터 설정에 따라 해당 컴포넌트를 이 위치에 표시함 -->
  <RouterView />
</template>

<!-- scoped는 이 CSS가 이 컴포넌트에만 적용되게 제한해 줌 -->
<!– scoped가 없으면 다른 컴포넌트를 합했을 때 모두 영향을 미치게 됨. 전역 설정이 됨. →
<style scoped>
/* 아직 스타일 없음 */
</style>
```

<script setup>

```
export default {
  name: 'App', // 컴포넌트 이름 (선택 사항, 디버깅이나 devtools에서 유용함)
  setup() {
    // Composition API 방식의 setup() 함수
    // 현재 이 컴포넌트에서 별도의 반응형 변수, computed, watch 등이 없기 때문에 비어 있음
    return {}; // 템플릿에서 사용할 변수나 함수를 넘겨줄 수 있음
  }
};
```

프론트엔드 준비

기본 페이지 컴포넌트

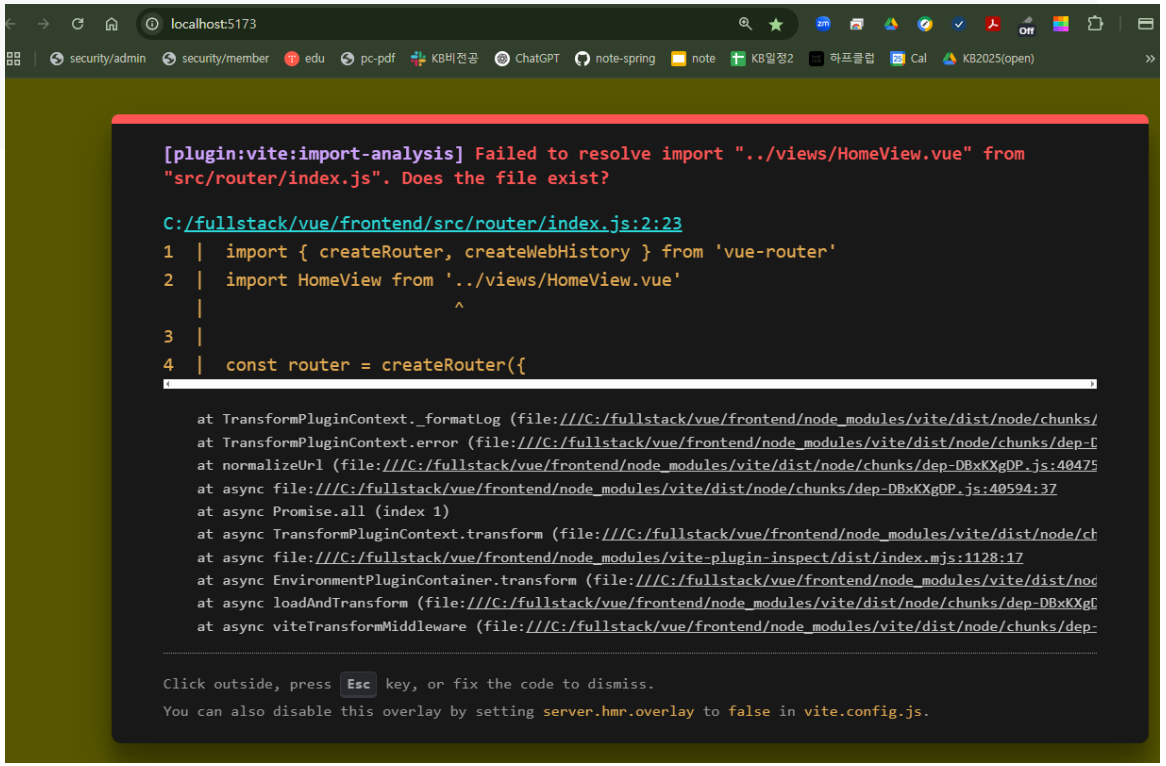


- 기존 views 폴더는 삭제

pages/HomePage.vue

```
<script setup>
</script>

<template>
  <h1>첫 번째 페이지</h1>
</template>
```

라우터설정 변경해야함.

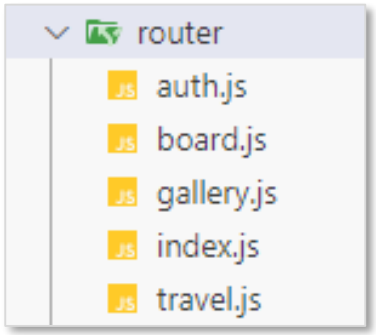


○ 나머지 모든 페이지 → 타이틀만 구성

나머지 pages내 페이지는 비어있는 상태로 구성

프론트엔드 준비

라우팅 준비



- 기능별로 라우팅 설정을 모듈화
  - index: 기본 라우팅
  - auth: login, logout, join 라우팅
  - board: 게시글 관련 라우팅
  - travel: 추천 여행지 관련 라우팅
  - gallery: 갤러리 관련 라우팅

index.html | main.js | index.js

frontend > src > router > index.js > router > routes > component

```
1  import { createRouter, createWebHistory } from 'vue-router'
2  import HomeView from '../views/HomeView.vue'
3
4  const router = createRouter({
5    history: createWebHistory(import.meta.env.BASE_URL),
6    routes: [
7      {
8        path: '/',
9        name: 'home',
10       component: HomeView,
11     },
12     {
13       path: '/about',
14       name: 'about',
15       // route level code-splitting
16       // this generates a separate chunk (About.[hash].js) for this route
17       // which is lazy-loaded when the route is visited.
18       component: () => import('../views/AboutView.vue'),
19     },
20   ],
21 })
22
23 export default router
```

router/auth.js

```
export default [
  {
    path: '/auth/login',
    name: 'login',
    component: () => import('../pages/auth/LoginPage.vue'),
  },
  {
    path: '/auth/join',
    name: 'join',
    component: () => import('../pages/auth/JoinPage.vue'),
  },
  {
    path: '/auth/profile',
    name: 'profile',
    component: () => import('../pages/auth/ProfilePage.vue'),
  },
  {
    path: '/auth/changepassword',
    name: 'changepassword',
    component: () => import('../pages/auth/ChangePasswordPage.vue'),
  },
];
```

router/board.js

```
export default [
  {
    path: '/board/list',
    name: 'board/list',
    component: () => import('../pages/board/BoardListPage.vue'),
  },
];
```

router/travel.js

```
export default [
  {
    path: '/travel/list',
    name: 'travel/list',
    component: () => import('../pages/travel/TravelListPage.vue'),
  },
];
```

router/gallery.js

```
export default [
  {
    path: '/gallery/list',
    name: 'gallery/list',
    component: () => import('../pages/gallery/GalleryListPage.vue'),
  },
];
```

router/index.js

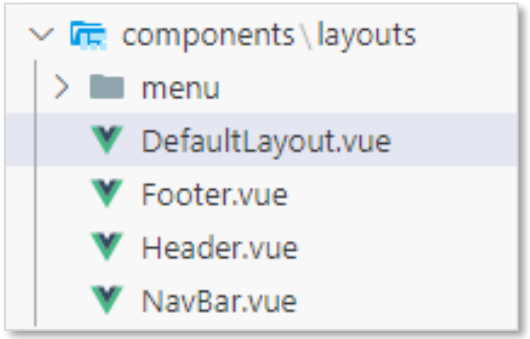
```
import { createRouter, createWebHistory } from 'vue-router';
import HomePage from '../pages/HomePage.vue';
import authRoutes from './auth';
import boardRoutes from './board';
import travelRoutes from './travel';
import galleryRoutes from './gallery';

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    { path: '/', name: 'home', component: HomePage },
    ...authRoutes,
    ...boardRoutes,
    ...travelRoutes,
    ...galleryRoutes,
  ],
});

export default router;
```

```
// ...authRoutes
// ...은 JavaScript의 전개 연산자(spread operator)
// 아래의 코드를 전개해서 넣어주는 의미

// authRoutes는 객체 배열임

// const authRoutes = [
//   { path: '/login', component: LoginPage },
//   { path: '/register', component: RegisterPage },
// ];
// {
//   path: '/auth/login',
//   name: 'login',
//   component: () => import('../pages/auth/LoginPage.vue'),
// },
// {
//   path: '/auth/join',
//   name: 'join',
//   component: () => import('../pages/auth/JoinPage.vue'),
// },
// {
//   path: '/auth/profile',
//   name: 'profile',
//   component: () => import('../pages/auth/ProfilePage.vue'),
// },
// {
//   path: '/auth/changepassword',
//   name: 'changepassword',
//   component: () => import('../pages/auth/ChangePasswordPage.vue'),
// },
```

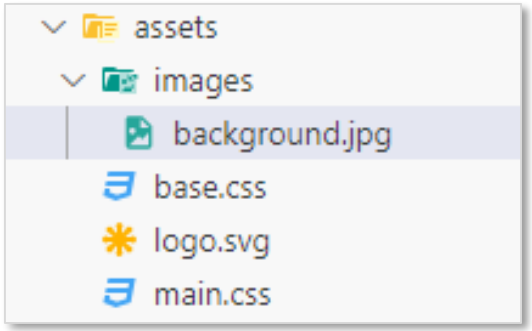
프론트엔드 준비

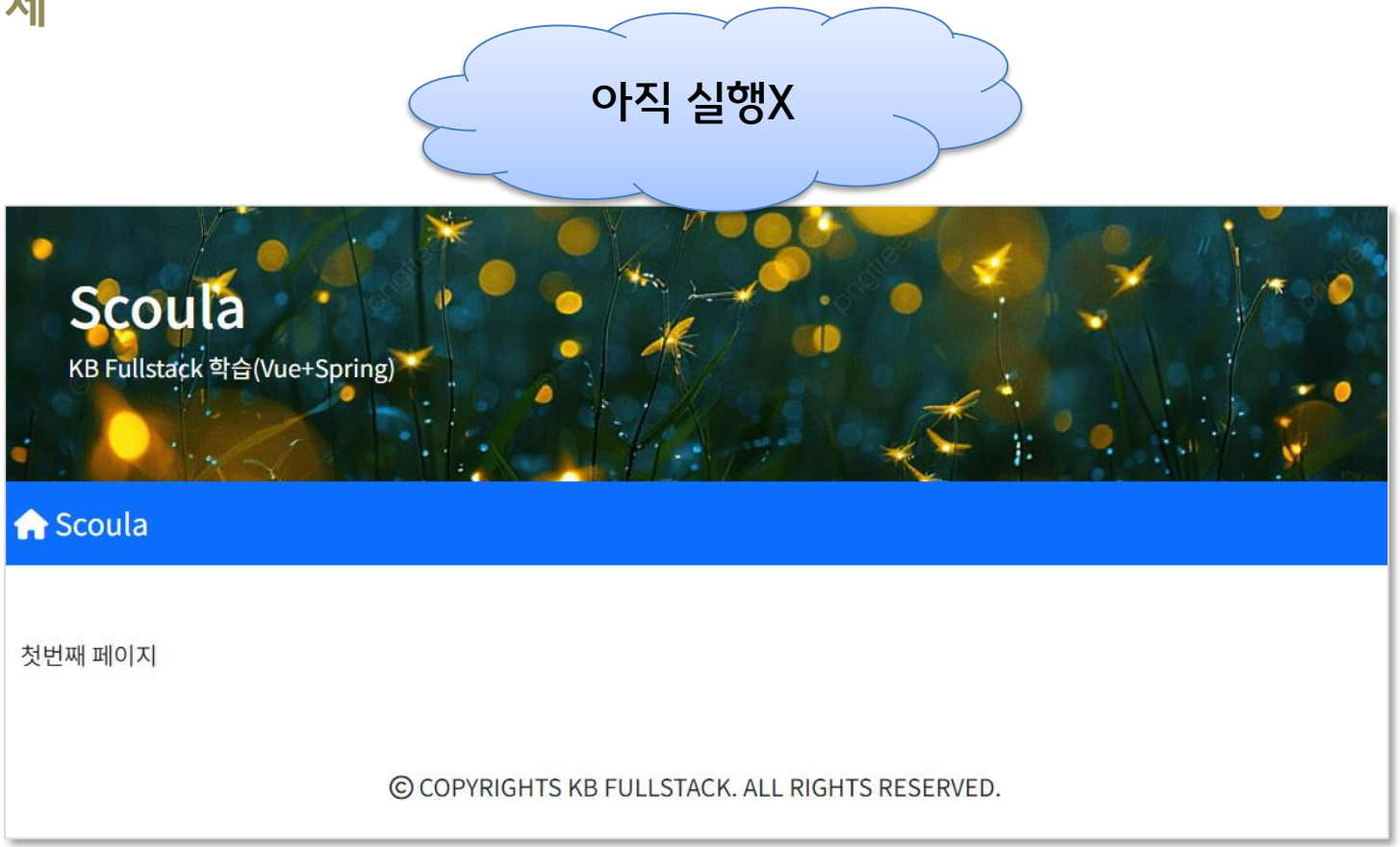
기본 레이아웃

- 기존 components/의 내용 모두 삭제
- 다음 구조로 컴포넌트 작성



- 헤더의 배경 이미지 준비





config/index.js

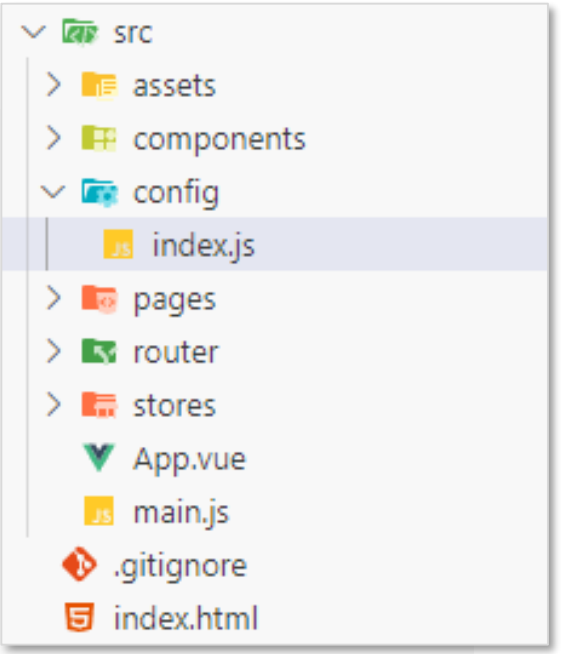
```
export default {
  title: 'Scoula',          // 메인 타이틀
  subtitle: 'KB Fullstack 학습(Vue+Spring)', // 서브 타이틀
  menus: [  // 메인 메뉴 구성 정보
    {
      title: '게시판',
      url: '/board/list',
      icon: 'fa-solid fa-paste',
    },
    {
      title: '여행',
      url: '/travel/list',
      icon: 'fa-solid fa-plane-departure',
    },
    {
      title: '갤러리',
      url: '/gallery/list',
      icon: 'fa-regular fa-images',
    },
  ],
```

vue 컴포넌트 코딩시
여기에 설정한 값을 가져다가 설정 가능함. → 설정 값을 외부 파일에 넣는 의미

Spring의 properties파일과 유사

읽기전용

→ 읽기/쓰기 가능 중앙저장소는??
Pinia(store폴더) 사용



config/index.js

```
  accoutMenus: {          // 인증 관련 메뉴 정보
    login: {
      url: '/auth/login',
      title: '로그인',
      icon: 'fa-solid fa-right-to-bracket',
    },

    join: {
      url: '/auth/join',
      title: '회원가입',
      icon: 'fa-solid fa-user-plus',
    },
  },
};
```

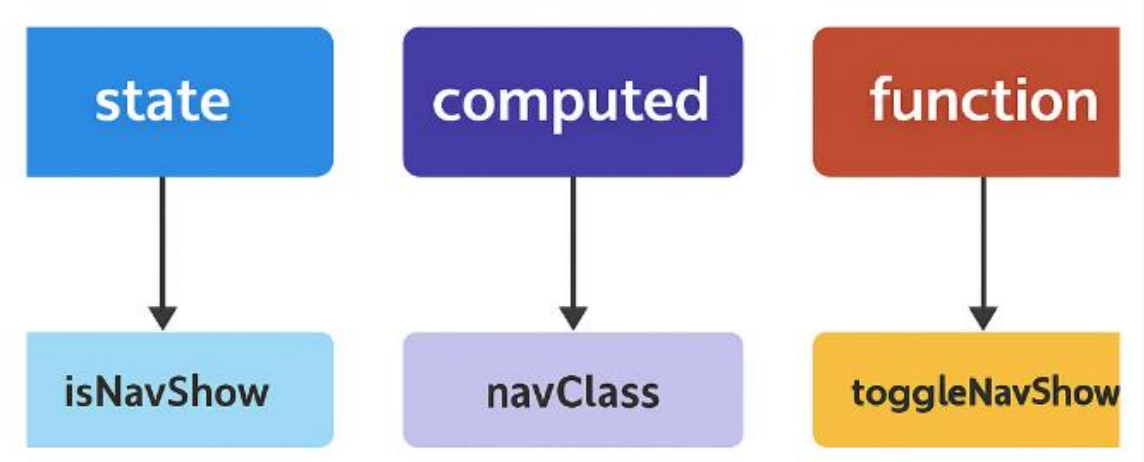
components/layouts/Header.vue

컴포넌트를 만들어보자.

```
<script setup>
import config from '@/config';
</script>
<template>
  <div class="jumbotron p-5 bg-primary text-white">
    <h1>{{ config.title }}</h1>
    <p>{{ config.subtitle }}</p>
  </div>
</template>
<style scoped>
.jumbotron {
  background-image: url('@/assets/images/background.jpg');
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  color: white;
  padding: 2rem;
}
</style>
```

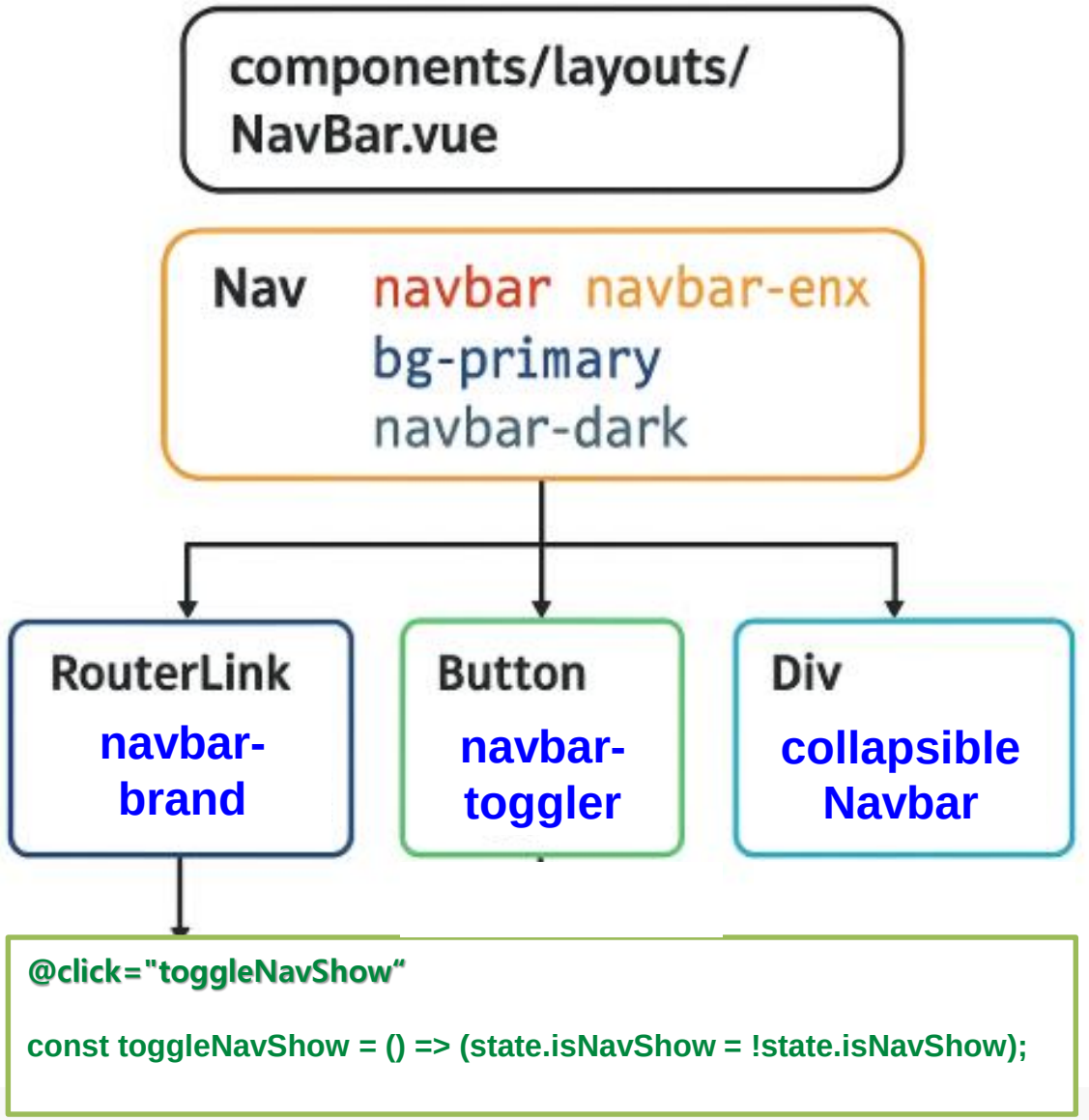
components/layouts/NavBar.vue

```
<script setup>
// Vue의 Composition API에서 반응형 객체와 계산 속성을 불러옴
import { reactive, computed } from 'vue';

// 전역 설정 값을 담고 있는 config 파일을 불러옴 (현재 코드에서는 사용되지 않음)
import config from '@/config';

// 반응형 객체 생성. isNavShow는 네비게이션 메뉴의 표시 여부를 나타냄
// 객체 형태의 반응형 상태 선언
let state = reactive({ isNavShow: false });

// state.isNavShow 값에 따라 적용할 CSS 클래스명을 동적으로 계산
// 상태에 따라 자동 계산되는 속성 정의
let navClass = computed(() =>
  state.isNavShow
    ? 'collapse navbar-collapse show'   // 메뉴가 열릴 때 클래스
    : 'collapse navbar-collapse'        // 메뉴가 닫힐 때 클래스
);

// 버튼 클릭 등으로 isNavShow 값을 true/false로 토글하는 함수
const toggleNavShow = () => (state.isNavShow = !state.isNavShow);
</script>
```



components/layouts/NavBar.vue

```
<template>
  <!-- 부트스트랩 스타일이 적용된 최상위 네비게이션 바 -->
  <nav class="navbar navbar-expand-sm bg-primary navbar-dark">
    <!-- 반응형 레이아웃을 위한 컨테이너 -->
    <div class="container-fluid">
      <!-- 홈으로 이동하는 라우터 링크, 브랜드 영역 -->
      <router-link class="navbar-brand" to="/">
        <i class="fa-solid fa-house"></i>  <!-- 집 모양 아이콘 -->
        Scoula                          <!-- 브랜드명 -->
      </router-link>
      <!-- 모바일 화면에서 메뉴 버튼 역할을 하는 햄버거 버튼 -->
      <button class="navbar-toggler" type="button"
              data-bs-toggle="collapse"
              data-bs-target="#collapsibleNavbar"
              @click="toggleNavShow" >
        <!-- 햄버거 아이콘 -->
        <span class="navbar-toggler-icon"></span>
      </button>
      <!-- 메뉴 영역: isNavShow 값에 따라 동적으로 class가 변경됨 -->
      <div :class="navClass" id="collapsibleNavbar">
        <!-- 추후 네비게이션 메뉴 항목들이 들어갈 자리 -->
      </div>
    </div>
  </nav>
</template>
```

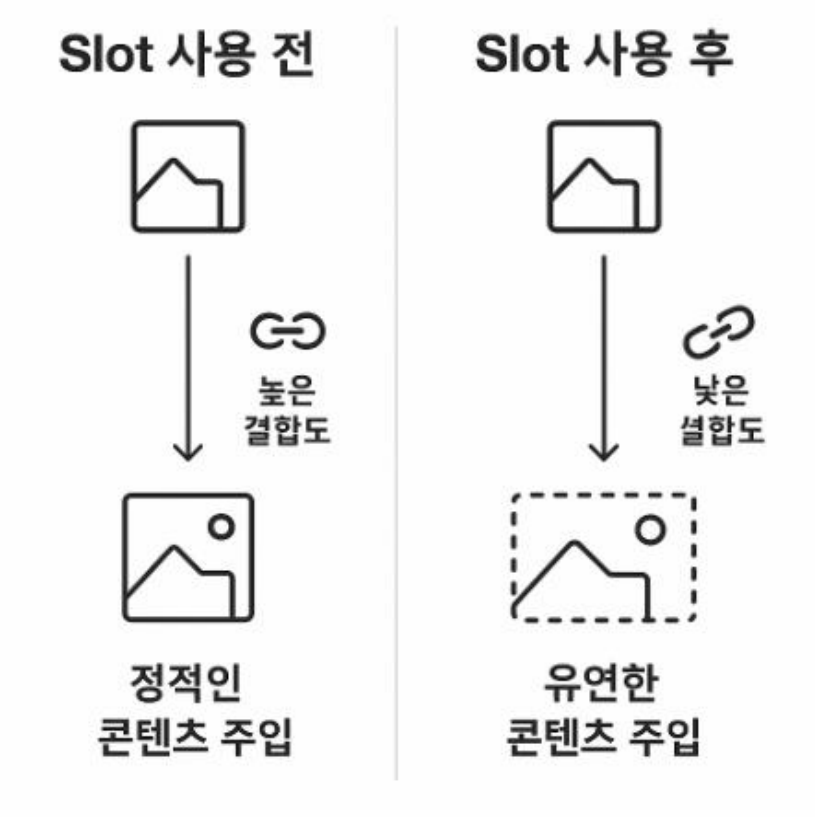


components/layouts/Footer.vue

```
<script></script>
<template>
  <div class="my-5 p-3 text-center">
      <i class="fa-regular fa-copyright"></i>
      COPYRIGHTS KB FULLSTACK. ALL RIGHTS RESERVED.
  </div>
</template>
```

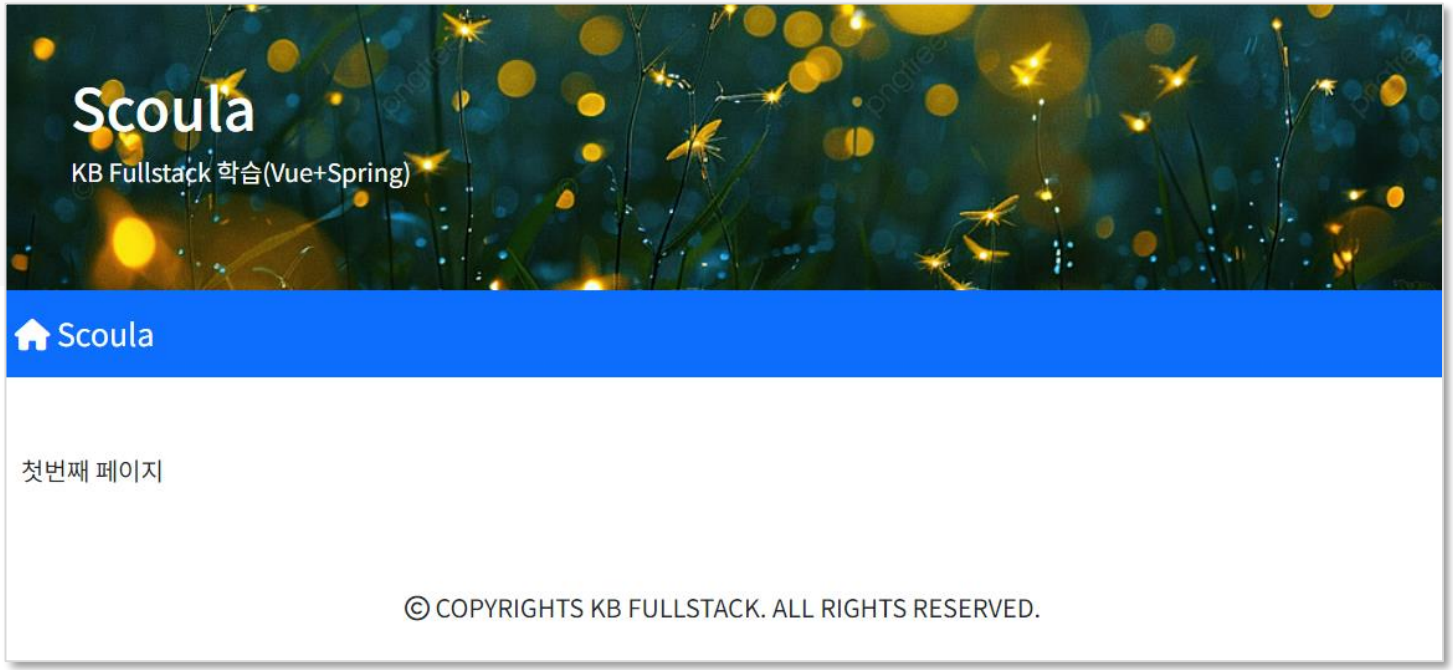
components/layouts/DefaultLayout.vue

```
<script setup>
import Header from './Header.vue';
import NavBar from './NavBar.vue';
import Footer from './Footer.vue';
</script>

<template>
  <div class="container">
    <Header />
    <NavBar />
    <div class="content my-5 px-3">
      <slot></slot>
    </div>
    <Footer />
  </div>
</template>
```

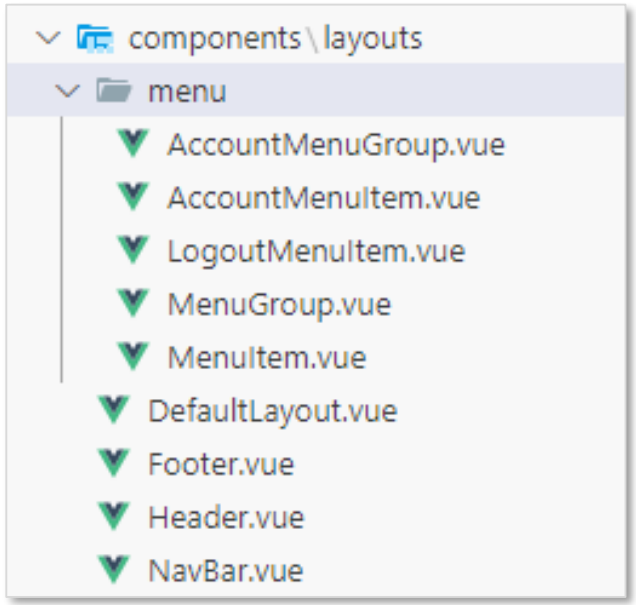


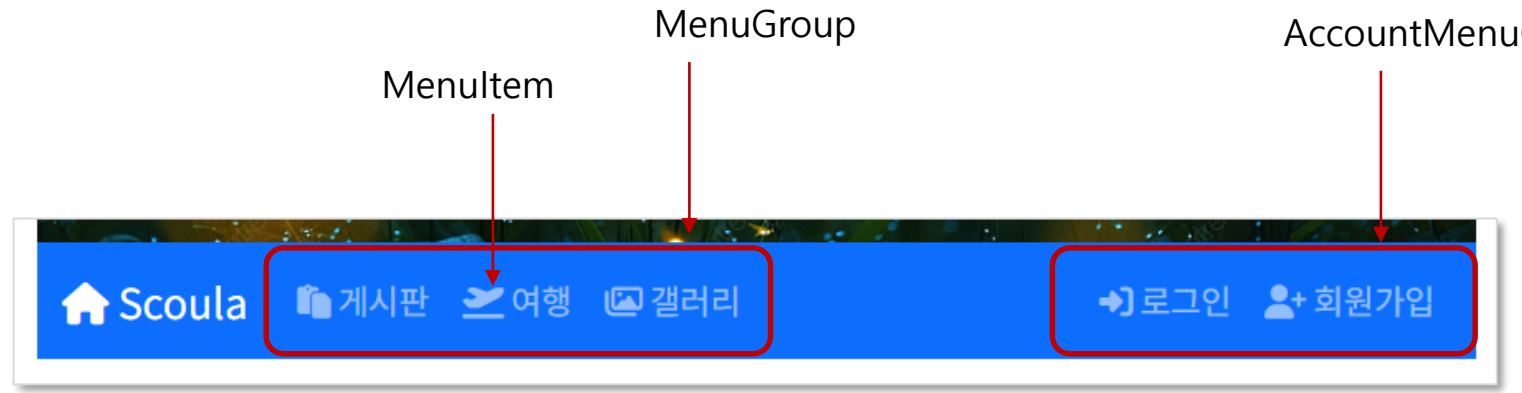
App.vue

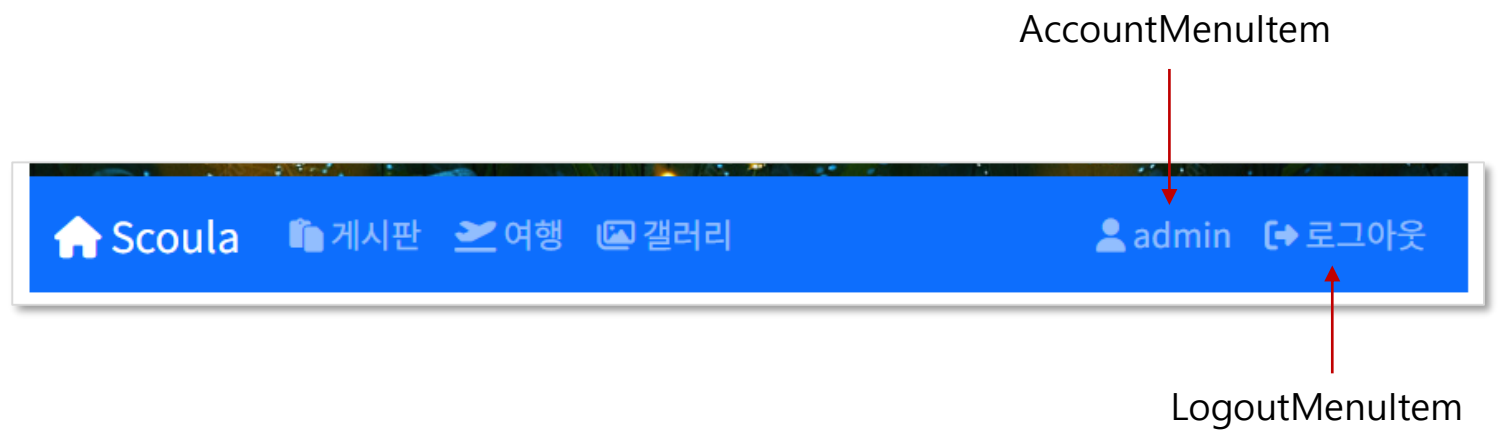
```
<script setup>
import { RouterView } from 'vue-router';
import DefaultLayout from './components/layouts/DefaultLayout.vue';
</script>

<template>
  <DefaultLayout>
    <RouterView />
  </DefaultLayout>
</template>

<style scoped></style>
```



메뉴의 구성


components \ layouts
menu
AccountMenuGroup.vue
AccountMenuItem.vue
LogoutMenuItem.vue
MenuGroup.vue
MenuItem.vue
DefaultLayout.vue
Footer.vue
Header.vue
NavBar.vue


MenuItem
MenuGroup
AccountMenuGroup
Scoula
게시판
여행
갤러리
로그인
회원가입


AccountMenuItem
Scoula
게시판
여행
갤러리
admin
로그아웃
LogoutMenuItem

components/layouts/menu/MenuItem.vue

```
<script setup>
// menu라는 props를 객체 타입으로 받아옴 (필수)
const props = defineProps({
  menu: { Type: Object, required: true },
});
</script>

<template>
  <!-- 메뉴 항목 하나 -->
  <li class="nav-item">
    <!-- 링크로 이동 (url로) -->
    <router-link class="nav-link" :to="menu.url“ >
      <!-- 아이콘 표시 -->
      <i :class="menu.icon"></i>
      <!-- 메뉴 이름 표시 -->
      {{ menu.title }}
    </router-link>
  </li>
</template>
<style></style>
```

components/layouts/menu/MenuGroup.vue

```
<script setup>

// 메뉴 항목 컴포넌트 import
import MenuItem from './MenuItem.vue';

// menus 배열 props 받음
const props = defineProps({
  menus: { Type: Array, required: true },
});
</script>

<template>
 <!-- 네비게이션 리스트 -->
  <ul class="navbar-nav">
 <!-- 메뉴 배열 반복 렌더링 -->
    <MenuItem v-for="menu in menus" :menu="menu"  />
  </ul>
</template>
```

components/layouts/menu/AccountMenuItem.vue

```
<script setup>
// username 문자열 props 받음
defineProps({ username: String });
</script>

<template>
 <!-- 네비게이션 항목 -->
  <li class="nav-item">
 <!-- 프로필 페이지로 이동하는 링크 -->
    <router-link class="nav-link" to="/auth/profile">
     <!-- 사용자 아이콘 -->
      <i class="fa-solid fa-user"></i>
      <!-- 사용자 이름 표시 -->
      {{ username }}
    </router-link>
  </li>
</template>
```

components/layouts/menu/LogoutMenuItem.vue

```
<script setup>
// 라우터 인스턴스 가져옴
import { useRouter } from 'vue-router';

// 라우터 객체 생성
const router = useRouter();

// 로그아웃 시 홈으로 이동
const logout = (e) => {
  // 로그아웃
  router.push('/');
};
</script>

<template>
 <!-- 로그아웃 링크 -->
  <a href="#" class="nav-link" @click.prevent="logout">
    <!-- 로그아웃 아이콘 -->
    <i class="fa-solid fa-right-from-bracket"></i>
    로그아웃
  </a>
</template>
```

components/layouts/menu/AccountMenuGroup.vue

```
<script setup>
// 계산 속성 사용
import { computed } from 'vue';
// 메뉴 컴포넌트 불러오기
import MenuItem from './MenuItem.vue';
import AccountMenuItem from './AccountMenuItem.vue';
import LogoutMenuItem from './LogoutMenuItem.vue';
import config from '@/config';

// 설정 파일에서 계정 메뉴 가져오기
const { login, join } = config.accoutMenus;

// 로그인 상태 (임시: false)
const islogin = computed(() => false);   // 임시: 로그인하지 않음
// 사용자 이름 (임시: 없음)
const username = computed(() => '');  // 임시: 사용자명 없음
</script>
```

components/layouts/menu/AccountMenuGroup.vue

```
<template>
  <ul class="navbar-nav ms-auto">

    <!-- 로그인 상태일 때 -->
    <template v-if="islogin">
      <!-- 사용자 이름 표시 메뉴 -->
      <AccountMenuItem :username="username" />
      <!-- 로그아웃 메뉴 -->
      <LogoutMenuItem />
    </template>

  <!-- 비로그인 상태일 때 -->
    <template v-else>
      <!-- 로그인 메뉴 -->
      <MenuItem :menu="login" />
      <!-- 회원가입 메뉴 -->
      <MenuItem :menu="join" />
    </template>
  </ul>
</template>
```

components/layouts/NavBar.vue

```
<script setup>
import { reactive, computed } from 'vue';
import config from '@/config';
import MenuGroup from './menu/MenuGroup.vue';
import AccountMenuGroup from './menu/AccountMenuGroup.vue';
…
</script>
<template>
  <nav class="navbar navbar-expand-sm bg-primary navbar-dark">
    <div class="container-fluid">
      …
      <div :class="navClass" id="collapsibleNavbar">
        <!-- 추후 작업 예정 -->
        <MenuGroup :menus="config.menus" />
        <AccountMenuGroup />
      </div>
    </div>
  </nav>
</template>
<style></style>
```

프론트엔드 준비

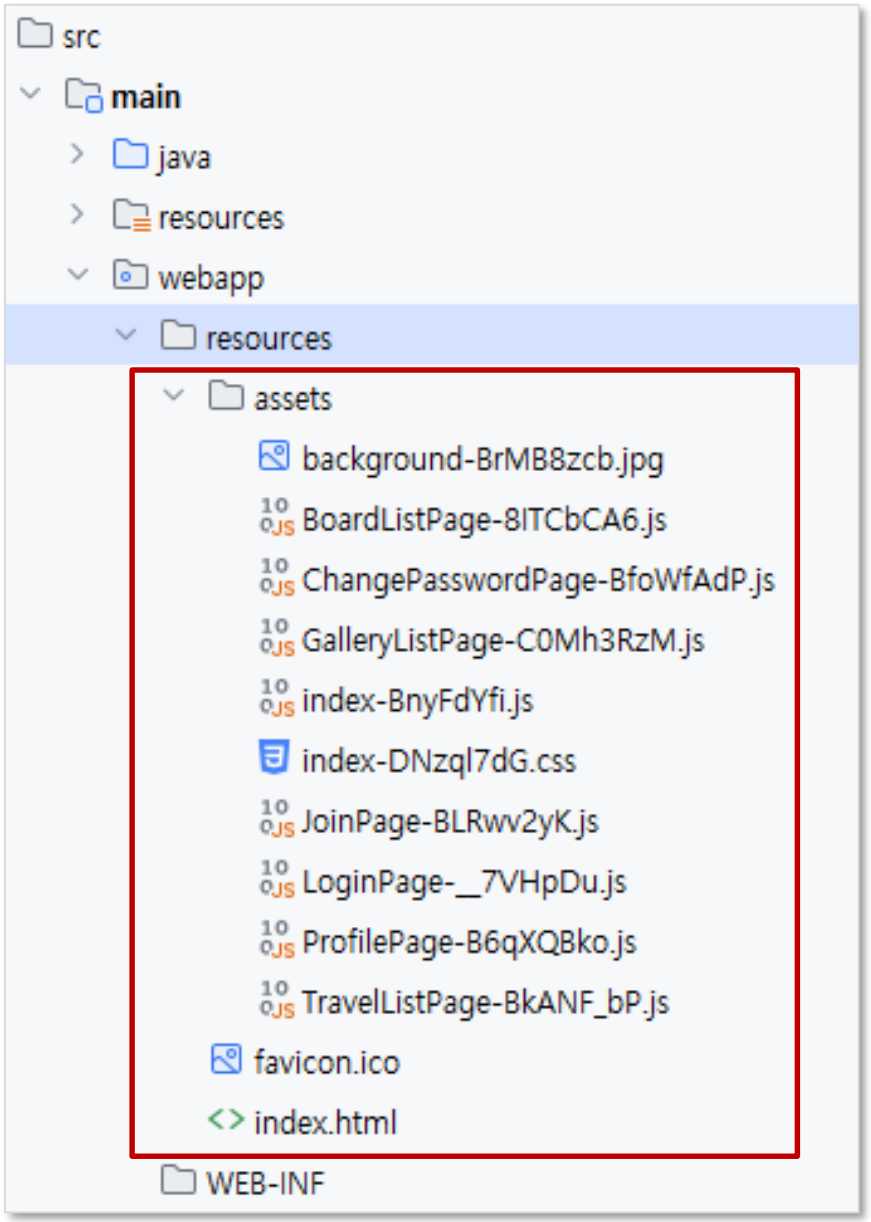
- **api 연결을 위한 proxy 서버 설정 및 배포 설정**
  - 백엔드 서버로 요청을 포워딩할 proxy 설정
  - npm run build 명령시 결과 생성 폴더를 백엔드의 resources 폴더로 지정

vite.config.js

```
import { fileURLToPath, URL } from 'node:url';
…
export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url)),
    },
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:8080',
      },
    },
  },
  build: {
   outDir: '../backend/src/main/webapp/resources',
  },
});
```

프론트엔드 준비

- **npm run build**
  - backend의 resources 폴더에 결과물 생김



백엔드 기동

http://localhost:8080/