



충북대 2024 하기 Web Full Stack AI 개발자 교육 보고서

충북대학교 컴퓨터공학과 2022040007 강지웅

개인프로젝트 설명 및 실행화면

1) 관리자 웹사이트

웹사이트를 사용하는 회원 및 회원들이 작성한 게시글을 수정관리하는 관리자 웹사이트입니다.

Node.js로 비동기방식의 라우팅 메소드를 구현하여 회원정보 관리 웹사이트를 제작하였습니다. RESTFul API를 이용하여 데이터를 관리하였으며 데이터 전달의 경우, Node View 엔진 유형 중 기본html문서에 자바스크립트 문법을 추가해 html 내용을 동적으로 조작하는 구조인 EJS View Engine을 이용하여 EJS 파일로 데이터를 전달합니다. 직접적인 데이터 관리는 MySQL에서 진행되었습니다. 이때, Node 기반의 MySQL ORM을 지원하는 노드 패키지 모듈인 Sequelize를 이용하여 DB를 연결하였습니다. 페이지의 동적관리를 위해 레이아웃을 이용하여 페이지 구조를 관리합니다. 회원을 관리하는 관리자 웹사이트이므로 관리자 신분의 로그인 정보가 필요하고 이 로그인 정보의 패스워드는 JWT 토큰을 이용합니다.

Example user
menu

로그인

회원가입

게시글목록

게시글작성

Search for something...

Log out

게시글정보관리

Home>게시글정보관리>게시글목록조회

글제목

아이피주소

게시여부

전체

조회

신규게시글

게시글번호	제목	아이피주소	조회수	게시여부	등록자번호	등록일시
1	게시글 제목1입니다.	111.111.111.111	10	1	1	1724566542764
2	게시글 제목2입니다.	222.111.111.111	11	0	2	1724566542764
3	게시글 제목3입니다.	333.111.111.111	30	1	3	1724566542764

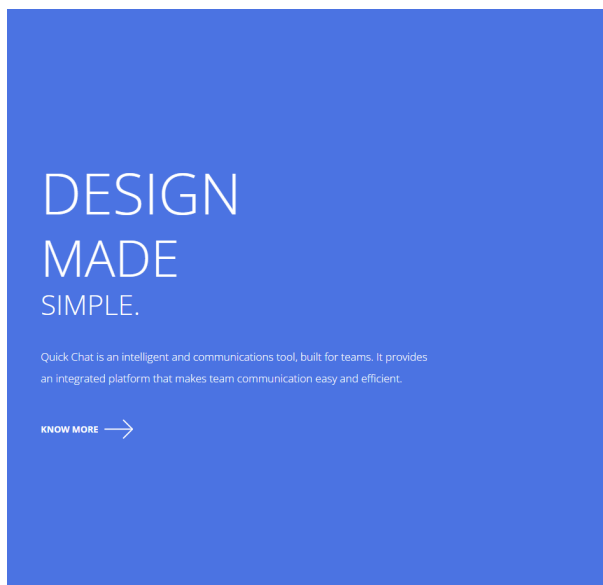
Copyright Example Company © 2014-2018

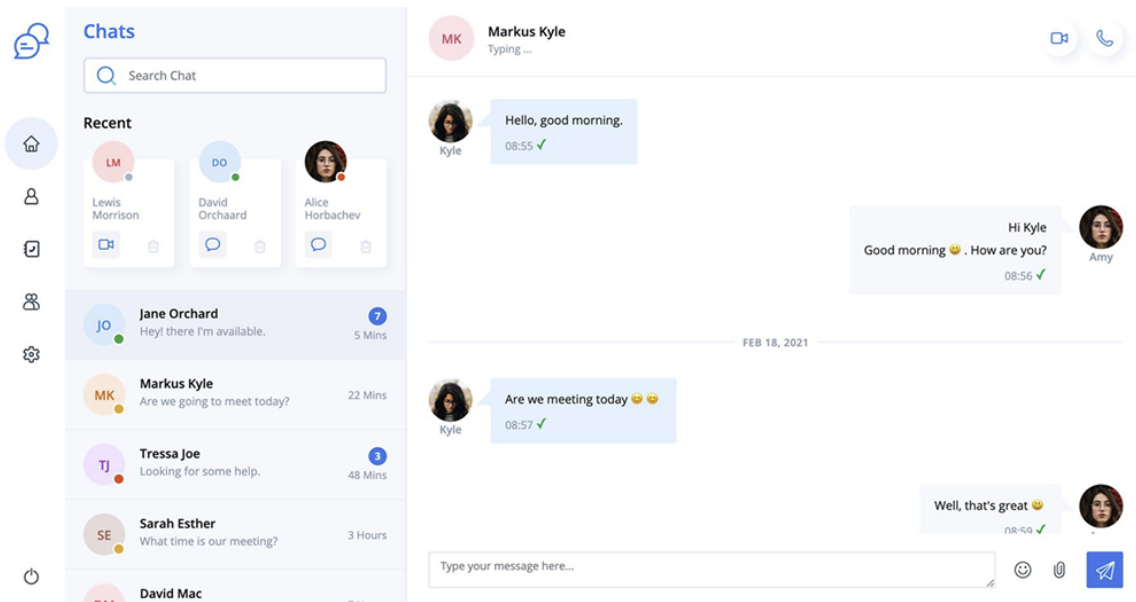
10GB of 250GB Free.

2) 사용자 채팅 웹사이트

사용자들간의 실시간 채팅이 가능한 채팅 웹사이트입니다.

관리자 웹사이트와 마찬가지로 비동기방식의 라우팅 메소드를 구현하였으며 EJS View Engine을 이용하여 EJS 파일로 데이터를 전달합니다. 프론트엔드없이 백엔드 단독으로 Visual Studio Code의 live server 확장 프로그램을 이용하여 로컬에서 웹페이지를 확인하며 구현하였습니다. Sequelize를 이용하여 DB를 연결하며 페이지의 동적 관리를 위해 레이아웃을 이용하여 페이지 구조를 관리합니다. 채팅에는 회원 정보가 필요하므로 로그인 정보가 필요하고 이 로그인 정보의 패스워드는 JWT 토큰을 이용합니다. 또한 사용자들간의 채팅에는 실시간으로 서버 연결 및 유지가 필요하므로 socket.io 클라이언트 라이브러리를 참조하여 즉시 수신할 수 있도록 합니다.





3) 커뮤니티 블로깅 웹사이트

블로깅, AI 생성 이미지 갤러리, 사용자간 채팅, 챗봇채팅, 마이페이지 등 여러 기능을 제공하는 커뮤니티 블로깅 웹사이트입니다.

백엔드 서버와 연동하여 개발한 프론트엔드 학습 위주의 웹사이트입니다. JavaScript 언어에 XML 표기문법을 추가한 자바스크립트 확장 문법인 JSX를 사용하였으며 컴포넌트 입력 관리의 방법으로 TypeScript와 Interface를 사용하였습니다. 컴포넌트 상태관리의 기술로 useState, useEffect, useRouter, useContext와 같은 Hook을 사용하고 있습니다. React Rendering 방식으로 CSR의 방식을 채택하여 최초 1회 화면이 Rendering(mounting)하는 시점에 백엔드 API로 데이터를 가져옵니다. 블로깅, 사용자간채팅, 마이페이지 등은 앞서 개발한 웹사이트들과 거의 동일한 기술을 사용하며, AI 생성에는 LLM을 사용합니다. OPENAI KEY를 이용하여 생성형 AI를 사용하고 원하는 이미지를 생성할 수 있습니다.



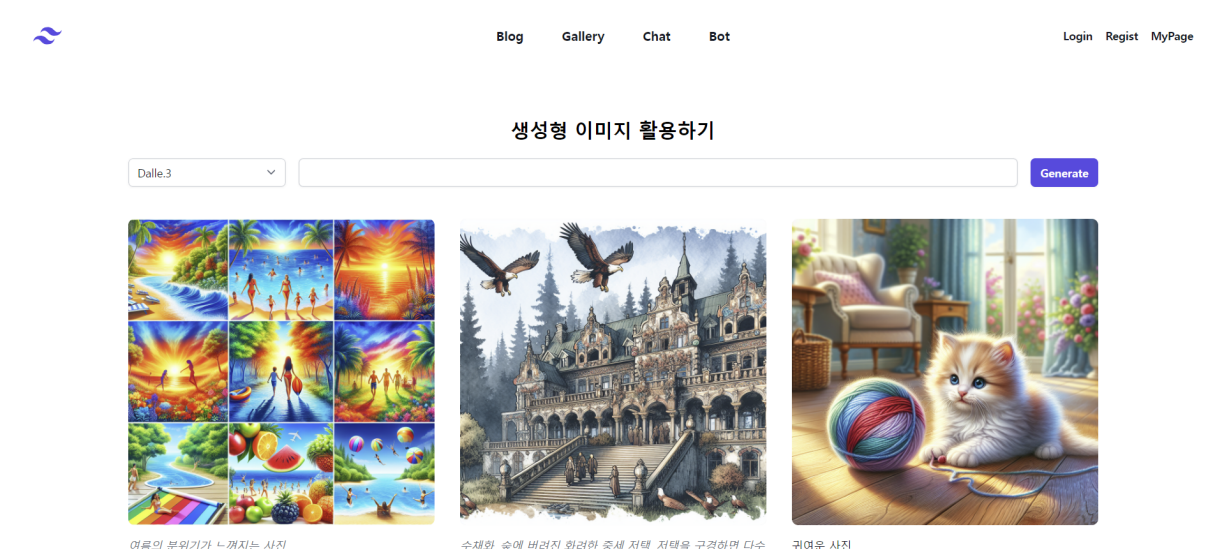
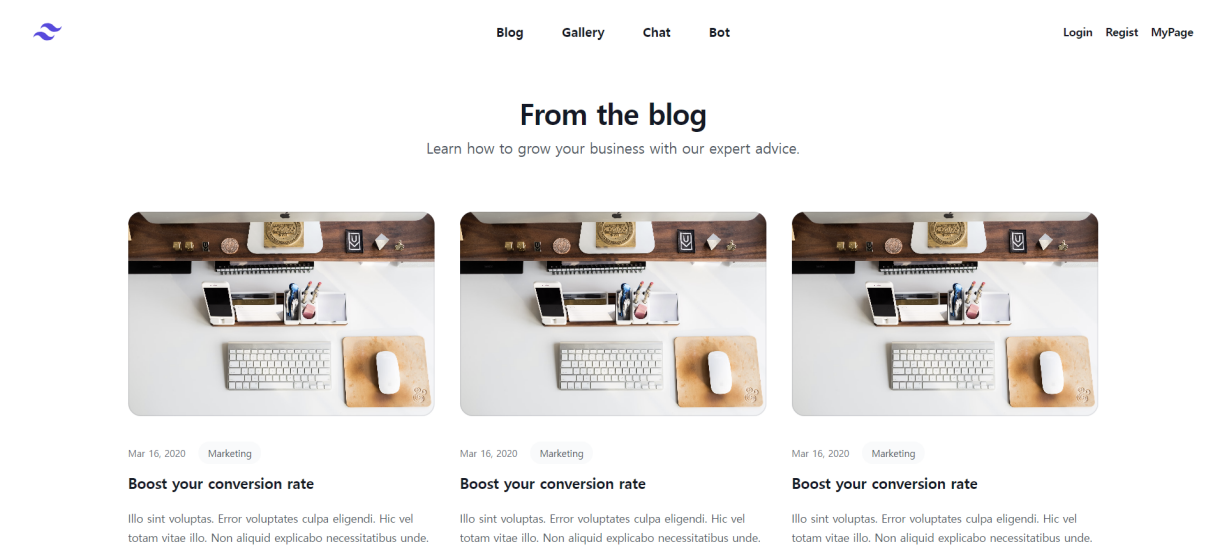
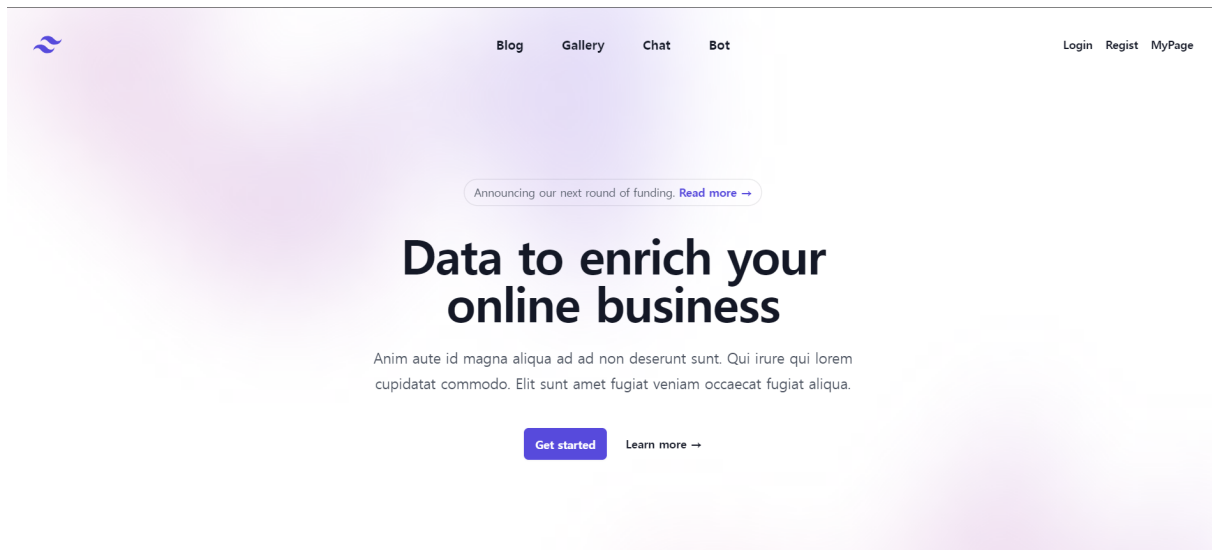
Sign in to your account

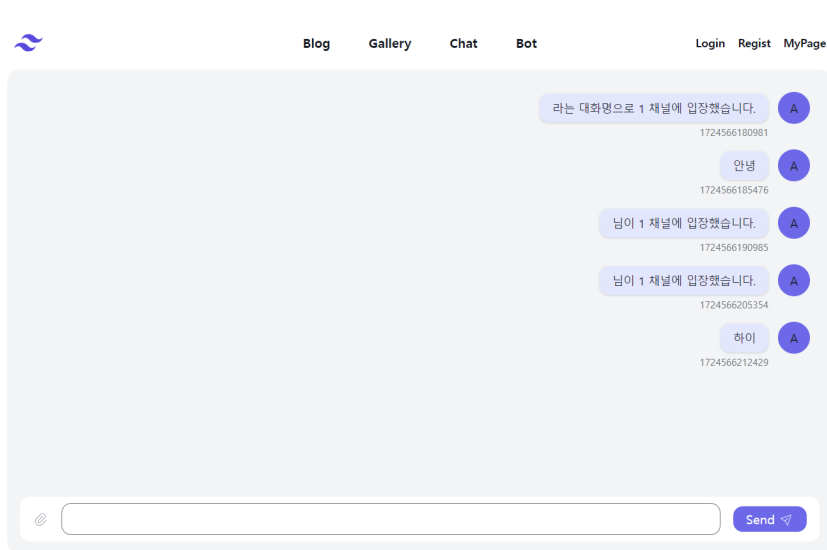
Email address


Password [Forgot password?](#)

[Log in](#)

Not a member? [Start a 14 day free trial](#)








Register your account

Email address

Password

Name

Already a member? [Login](#)



[Blog](#)
[Gallery](#)
[Chat](#)
[Bot](#)

[Login](#)
[Register](#)
[MyPage](#)

채팅방 목록

일대일 사용자 채팅방 목록

채팅방명	동참자 제한수	
집가고싶은사람들의모임 집사모	1500	<input type="button" value="참여하기"/>
그룹채팅1	800	<input type="button" value="참여하기"/>
요를레이후	100	<input type="button" value="참여하기"/>



Home

Profile

Blogs

Profile

This information will be displayed publicly so be careful what you share.

Photo



Change

Username

janesmith

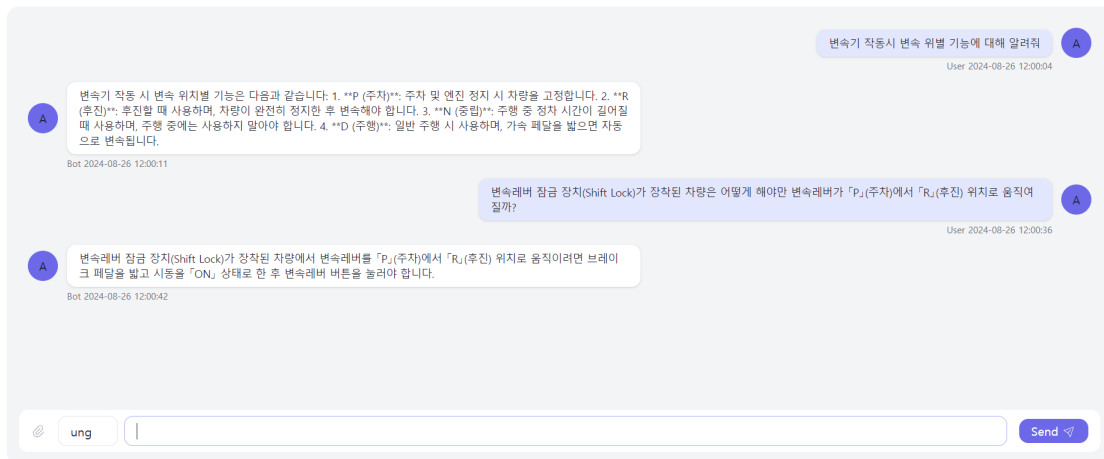
About

Write a few sentences about yourself.

4) LLM AI Chatbot

사용자가 질문을 입력하면, 서버는 특정 PDF 문서에서 관련된 정보를 검색하고, 검색된 내용을 기반으로 답변을 생성하는 RAG 챗봇을 구현한 웹사이트입니다.

TypeScript를 사용하여 개발되었으며, JSX 문법을 적용하여 컴포넌트 기반의 구조를 갖추고 있습니다. 사용자 인터페이스 및 상태 관리를 위해 `useState`, `useEffect`와 같은 React Hooks가 사용되었으며, 프론트엔드와 백엔드 간의 통신은 API를 통해 이루어집니다. PDF 문서 내 텍스트를 벡터화하여 메모리 내에 저장하고, 사용자의 질문과 관련된 정보를 검색하여 정확한 답변을 제공합니다. CSR 방식을 채택하여 사용자의 질문에 실시간으로 응답합니다.



주요 코드

```

var express = require("express");
var router = express.Router();

/* 메인 페이지 요청과 응답처리 라우팅 메소드 */
router.get("/", function (req, res, next) {
  res.render("index", { title: "Express" });
});

/*
-샘플용 프론트엔드 채팅 웹페이지
-채팅서버와 연결된 모든사용자들간 채팅하는 웹페이지 요청과 응답처리 라우팅메소드
-요청주소: http://localhost:5000/chat
-요청방식: Get
-응답결과: 단순 채팅 웹페이지 뷰 반환
*/
router.get("/chat", async (req, res) => {
  res.render("chat.ejs");
});

/*
-샘플용 프론트엔드 채팅 웹페이지
-특정 채팅방에 입장한 그룹 사용자들간 채팅하는 웹페이지 요청과 응답처리 라우팅메소드
-요청주소: http://localhost:5000/groupchat
-요청방식: Get
-응답결과: 단순 그룹 채팅 웹페이지 뷰 반환
*/
router.get("/groupchat", async (req, res) => {
  res.render("groupchat.ejs");
});

module.exports = router;

```

```

/*
- 신규 게시물 등록 웹페이지에서 보내준 사용자가 입력/선택한 신규 게시물 데이터를 등록 처리 요청과 응답처리 라우팅 메서드 구현
- 호출 주소: http://localhost:3000/article/create
- 호출 방식: POST 방식
- 응답 결과: 신규 게시물 DB 등록 처리 후 특정 페이지로 이동 또는 특정 뉴파일 제공
- ** 라우팅 주소와 요청 방식 2가지가 동일해야 해당 라우팅 메서드가 호출되고 실행된다. **
*/
router.post("/create", async (req, res) => {
  // Step1: 사용자가 입력한 폼태그 내 입력/선택 데이터 추출하기
  // req.body.전달되는 폼태그 내 html 입력/선택 요소의 name 속성명
  const title = req.body.title;
  const contents = req.body.contents;
  const display = req.body.display;

  // Step2: DB 게시물 테이블에 저장할 JSON 데이터 생성하기
  // 객체 속성명과 속성의 데이터값 변수/상수명이 같으면 상수/변수명은 생략가능하다.
  var article = {
    title: title,
    contents: contents,
    display: display,
    ip_address: "111.111.111.111",
    view_cnt: 0,
    regist_id: 1,
    regist_date: Date.now(),
  };
});

```

```

// 데이터 베이스 객체
const db = {};

// DB연결정보로 시퀄라이즈 ORM 객체 생성
const sequelize = new Sequelize(
  config.database,
  config.username,
  config.password,
  config
);

// DB 처리 객체에 시퀄라이즈 정보 맵핑처리
// 이후 DB객체를 통해 데이터 관리가능해짐
db.sequelize = sequelize; // DB연결정보를 포함한 DB제어 객체속성(CRUD)
db.Sequelize = Sequelize; // Sequelize 패키지에서 제공하는 각종 데이터 타입 및 관련 객체정보를 제공함

// 회원모델 모듈파일 참조하고 db속성정의하기
// 6개 테이블에 대한 모델 맵핑처리
db.Member = require("./member.js")(sequelize, Sequelize);

db.Channel = require("./channel.js")(sequelize, Sequelize);
db.ChannelMember = require("./channel_member.js")(sequelize, Sequelize);
db.ChannelMsg = require("./channel_msg.js")(sequelize, Sequelize);

db.Article = require("./article.js")(sequelize, Sequelize);
db.ArticleFile = require("./article_file.js")(sequelize, Sequelize);

```

```

<script>
  // 클라이언트(웹브라우저) 서버 연결 소켓 객체 정의
  // 서버소켓과 연결을 시도하고 연결이 완료되면 연결이 지속됩니다.
  // io.connect("/")를 통해 서버소켓과 연결을 완료하고 서버로 메시지를 보낼 socket객체가 반환됩니다.
  var socket = io.connect("http://localhost:5000/");

  // jquery에서 html요소의 id값으로 html요소를 찾습니다. $("#html요소의id값")
  // 전송버튼이 클릭되면 서버로 사용자 메시지를 전송합니다.
  $("#btnSend").click(function () {
    // 사용자 닉네임과 메시지 입력값을 추출합니다.
    var nickName = $("#nickname").val();
    var message = $("#message").val();
    var msgData = `{${nickName}: ${message}}`;

    // socket.emit("서버이벤트수신기명", 서버로보낼전송데이터);
    socket.emit("broadcast", msgData);
  });

  // 서버에서 보내준 메시지를 수신하는 수신기 정의하기
  // 클라이언트 이벤트 메시지 수신기 정의하기
  socket.on("receiveAll", function (serverMsg) {
    // 서버에서 보내준 메시지 문자열을 포함한 li태그를 하나 만들고 ul태그에 li태그를 추가(append())합니다.
    $("#chatHistory").append("<li>${serverMsg}</li>");
  });
</script>

```



```

// 로그인 사용자 정보 상태관리 데이터 초기화
const [member, setMember] = useState({ email: "", password: "" });

// 로그인 UI요소(메일주소/암호) 사용자 입력시 데이터 동기화 처리 함수
const memberChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  setMember({ ...member, [e.target.name]: e.target.value });
};

// 로그인 버튼 클릭시 로그인 정보 백엔드 API에 전달하여 JWT 토큰 정보를 받아온다.
const loginSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();

  // 백엔드 login RESTful API를 호출한다.

  // case1: 웹브라우저 자바스크립트엔진에 탑재되어있는 fetch함수를 통해 백엔드 RESTful Login API를 호출한다.
  try {
    const response = await fetch("http://localhost:5000/api/member/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(member),
    });

    // 통신 결과에서 로그인 api에서 반환한 JSON 데이터값 추출하기
    const result = await response.json();
    console.log("login api에서 반환한 요청 결과값:", result);

    if (result.code == 200) {
      console.log("정상적으로 로그인 완료");

      // step1: 백엔드에서 제공한 JWT 토큰값을 웹브라우저의 localStorage 저장소에 저장
      localStorage.setItem("token", result.data.token);

      // 로그인한 사용자 정보를 전역상태의 member 속성값으로 변경저장하기
      setGlobalData(result.data.member);
    }
  } catch (error) {
    console.log(error);
  }
};

```

```

//최초 1회 화면이 렌더링되는 시점(마운팅되는시점)에 실행되는 useEffect함수
//프로젝트 루트에 next.config.mjs파일내 reactStrictMode(엄격모드)값을 false로 변경해야 정확히 1회만 실행됨
//채팅서버와 연결되는 클라이언트 채팅 소켓 객체 생성 및 각종 채팅 이벤트 기능 구현영역
useEffect(() => {
  //웹브라우저 저장소에 저장된 서버에서 발급해준 JWT사용자인증정보 토근추출하기
  const token = localStorage.getItem("token");
  if (token == undefined) {
    router.push("/login");
  }

  console.log("전역 데이터 정보 확인하기:", globalData);
  setMemberToken(token as string);

  //최초 화면이 렌더링되는 시점(최초1회)에 서버소켓 연결하기
  socket.connect();

  //서버소켓과 연결이 완료되면 실행되는 이벤트처리함수
  //서버 소켓과 연결이 완료되면 자동으로 client 소켓에서connect이벤트가 실행되고
  //connect이벤트가 실행되면 처리할 이벤트 처리할 기능 구현
  //소켓 시스템 이벤트
  socket.on("connect", () => {
    console.log("정상적으로 서버소켓과 연결이 완료되었습니다.");
    console.log("전역 데이터 정보 확인하기:", globalData);
  });

  //disconnect 이벤트는 서버소켓이 끊어진경우 발생하는 이벤트
  //서버와의 연결소켓이 끊어진경우 처리할 기능을 핸들러함수에서처리합니다.
  //소켓 시스템 이벤트
  socket.on("disconnect", () => {
    console.log("서버소켓 연결이 종료되었습니다.");
  });

  //개발자 정의 클라이언트 소켓 이벤트 수신기 정의하기
  //socket.on('클라이언트 이벤트 수신기명',서버에서 전달해준 데이터를 받는함수정의);
  socket.on("receiveAll", function (msg: IMessage) {

```

```

// 백엔드에서 이미지 목록 데이터를 가져오는 비동기 함수 기능 정의
async function getBlogFiles() {
  // fetch함수를 통해 백엔드 이미지 목록 호출하기
  const response = await fetch("http://localhost:5000/api/openai/all", {
    method: "GET",
    headers: { "Content-Type": "application/json" },
  });

  const resultData = await response.json();
  console.log("백엔드에서 전달해준 결과값 확인:", resultData);
  setFileList(resultData.data);
}

// 이미지 생성 요청 함수
const generateSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();

  // fetch함수를 통해 백엔드 DALL-E API 호출하기
  const response = await fetch("http://localhost:5000/api/openai/dalle", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ model, prompt }),
  });

  const resultData = await response.json();
  console.log("백엔드에서 전달해준 결과값 확인:", resultData);

  await getBlogFiles();

  // 이미지 생성 후 프롬프트 및 모델 선택값 초기화처리
  setPrompt("");
  setModel("dall-e-3");
};

```

학습한 중요 기술 정리

1. AJAX(Asynchronous Javascript And XML)

AJAX는 웹 개발에서 사용되는 중요한 기법으로, JavaScript와 여러 웹 기술(HTML, CSS, DOM, XML 등)을 함께 사용하여 전체 페이지를 새로 고치지 않고도 필요한 데이터만을 서버로부터 비동기적으로 받아와 웹 페이지를 동적으로 업데이트할 수 있게 합니다. AJAX의 핵심은 XMLHttpRequest 객체를 이용한 비동기 통신이며, 오늘날에는 JSON 포맷이 XML을 대신하여 주로 사용되고 있습니다.

2. 미들웨어(Middleware)

미들웨어는 웹 애플리케이션에서 요청과 응답 사이에 위치하여 특정 로직을 수행하는 기능입니다. 라우팅 메소드 호출 시, 호출 주소와 콜백 함수 사이에 미들웨어를 삽입하여, 예를 들어 사용자의 로그인 여부를 체크하거나 요청 데이터를 검증하는 등의 작업을 수행할 수 있습니다.

3. 시맨틱 태그(Semantic Tag)

시맨틱 태그는 HTML에서 문서의 구조와 내용을 명확하게 표현하기 위한 태그들입니다. 예를 들어

<header>, <footer>, <article>, <nav> 등의 태그는 각 부분이 어떤 역할을 하는지 쉽게 유추할 수 있게 합니다. 시맨틱 태그를 사용하면 검색 엔진 최적화(SEO)와 접근성 향상에 유리하며, 코드의 가독성을 높이는 데도 도움이 됩니다.

4. 관계형 데이터베이스 관리 시스템(RDBMS)

RDBMS는 데이터를 안정적으로 영구히 관리하는 데 중점을 둔 데이터베이스 시스템입니다. RDBMS는 무결성을 보장하기 위해 조건을 사용하여 결함이 있는 데이터를 걸러내고, 데이터 수집의 신뢰성을 높입니다. 그러나 데이터의 양이 많아질수록 속도 저하가 발생할 수 있으며, DB 서버 복제 과정이 오래 걸릴 수 있다는 단점도 있습니다. RDBMS는 데이터 무결성에 중점을 두기 때문에, 속도보다는 데이터의 정확성과 일관성을 중요하게 생각하는 시스템에서 사용됩니다.

5. Socket.io 모듈

Socket.io는 실시간 양방향 통신을 가능하게 하는 Node.js 기반의 웹 소켓 라이브러리입니다. 클라이언트와 서버 간의 메시지 전송을 효율적으로 처리하며, 실시간 채팅, 알림, 실시간 데이터 업데이트 등의 기능을 구현할 때 유용하게 사용됩니다.

6. CORS(Cross-Origin Resource Sharing)

CORS는 웹 애플리케이션에서 다른 도메인에 있는 리소스에 접근할 때 발생하는 보안 이슈를 다루는 메커니즘입니다. 웹 브라우저는 보안 상의 이유로 동일 출처 원칙(Same-Origin Policy)을 적용하는데, 이 원칙은 다른 도메인, 프로토콜 또는 포트에서 리소스를 가져오는 것을 기본적으로 차단합니다. 이를 해결하기 위해 서버에서 특정 도메인에 대해 CORS를 설정하여 해당 도메인에서의 접근을 허용할 수 있습니다.

7. JWT(JSON Web Token)

JWT는 JSON 형식의 데이터를 기반으로 하며, 이를 암호화한 토큰을 통해 사용자 인증 및 권한 부여를 처리하는 방법입니다. JWT는 세션을 서버에 저장할 필요 없이 클라이언트 측에서 인증 정보를 유지할 수 있습니다.

8. RESTful API

RESTful API는 REST 원칙을 준수하여 설계된 API를 말합니다. REST(Representational State Transfer)는 자원을 URI로 표현하고, HTTP 메소드를 통해 자원을 처리하는 방식입니다. RESTful API는 그 설계 규칙을 엄격히 따름으로써, 일관되고 예측 가능한 API를 제공하며, 클라이언트와 서버 간의 상호작용을 효율적으로 만듭니다.

9. 클라우드 기반 서비스(IaaS)

IaaS(Infrastructure as a Service)는 클라우드 서비스 제공자가 가상화된 컴퓨팅 자원(서버, 스토리지, 네트워크 등)을 인터넷을 통해 제공하는 서비스 모델입니다. 수업에서는 AWS(Amazon Web Services)와 네이버 클라우드를 사용했습니다.

10. TypeScript

TypeScript는 JavaScript의 상위 호환성(Superset)을 제공하는 프로그래밍 언어로, 정적 타입 검사 기능을 추가하여 코드의 오류를 미리 방지할 수 있습니다. TypeScript는 컴파일 과정을 거쳐 JavaScript 코드로 변환되며, 순수 JavaScript와 혼용하여 사용할 수 있습니다.

11. JSX(JavaScript XML)

JSX는 React에서 사용하는 문법으로, JavaScript 코드 안에서 HTML과 유사한 구문을 작성할 수 있게 해줍니다. JSX는 실제로 HTML이 아닌 JavaScript 문법의 일환이며, Babel과 같은 트랜스파일러를 통해 일반 JavaScript로 변환됩니다.

12. 컴포넌트(Component)

컴포넌트는 React에서 UI를 구성하는 기본 단위로, 재사용 가능한 코드 블록입니다. 컴포넌트는 props(속성 값)를 통해 부모 컴포넌트로부터 데이터를 전달받고, 내부에서 상태(state)를 관리하며, JSX를 반환하여 UI를 렌더링합니다.

13. 컴포넌트 생명주기(Lifecycle)

컴포넌트는 생성, 업데이트, 제거의 생명주기 동안 다양한 메서드를 통해 상태를 관리합니다.

14. 마운팅(Mounting): 컴포넌트가 처음으로 생성되어 DOM에 삽입되는 과정입니다. 이 과정에서 컴포넌트는 초기 props와 state를 설정하고, 화면에 최초로 렌더링됩니다.

업데이팅(Updating): props 또는 state가 변경되어 컴포넌트가 다시 렌더링되는 과정입니다. 이 과정에서는 컴포넌트의 일부 UI만 재렌더링되므로 성능을 최적화할 수 있습니다.

언마운팅(Unmounting): 컴포넌트가 DOM에서 제거되는 과정입니다. 이때 컴포넌트는 정리 작업을 수행하여 메모리 누수를 방지합니다.

15. Fetch와 Axios

Fetch와 Axios는 JavaScript에서 비동기 HTTP 요청을 보내기 위해 사용하는 라이브러리입니다. Fetch는 브라우저 내장 함수로, 상대적으로 가볍고 최신 표준을 따릅니다. 반면 Axios는 추가 기능(예: 요청 취소, JSON 자동 변환)을 제공하는 외부 라이브러리로, 많은 개발자들 사이에서 인기가 높습니다.

16. MVC(Model-View-Controller) 아키텍처

- **Model(모델):** 애플리케이션의 데이터와 비즈니스 로직을 담당하는 부분입니다. 데이터베이스와 상호작용하거나, 데이터를 가공하는 역할을 합니다. Model은 사용자가 요청한 데이터를 조회, 수정, 삭제하는 등의 작업을 처리합니다.
- **View(뷰):** 사용자에게 보여지는 화면을 담당합니다. View는 Model에서 가져온 데이터를 사용자에게 보여주기 위해 렌더링하며, HTML, CSS, JavaScript를 사용하여 UI를 구성합니다. View는 데이터를 표시하는 역할만 하며, 그 자체로는 데이터를 변경할 수 없습니다.
- **Controller(컨트롤러):** Model과 View를 연결하는 중간 역할을 합니다. 사용자의 입력(예: HTTP 요청)을 받아 처리하고, 필요한 데이터를 Model에서 가져오거나 수정하며, 이 데이터를 View에 전달하여 최종적으로 화면에 표시합니다. Controller는 비즈니스 로직을 실행하고, 사용자 요청에 따라 적절한 View를 선택해 반환하는 역할을 합니다.

17. 비동기 라우팅 메소드(Asynchronous Routing Methods)

비동기 라우팅 메소드는 웹 애플리케이션에서 사용자의 요청을 비동기적으로 처리하는 방법입니다. 일반적으로 서버로부터 데이터를 가져오거나, 데이터를 저장할 때 비동기 처리를 사용하며, 수업에서는 `async/await`의 방식으로 구현했습니다.

18. RAG: 검색 기술을 이용해 LLM의 취약점을 보완하는 기술입니다.

GitHub 화면 캡처

링크: https://github.com/KangJiUng/2024_Summer_FullStackEdu

