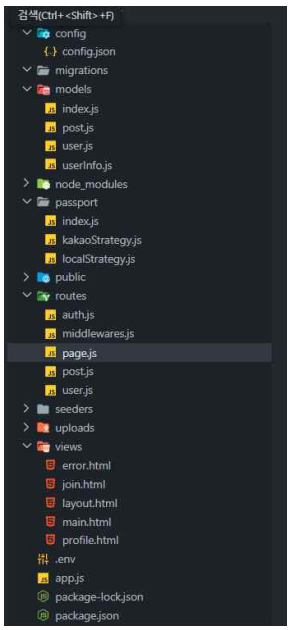
고급웹프로그래밍 기말 프로젝트

60175054 김준현

1. 전체 구성

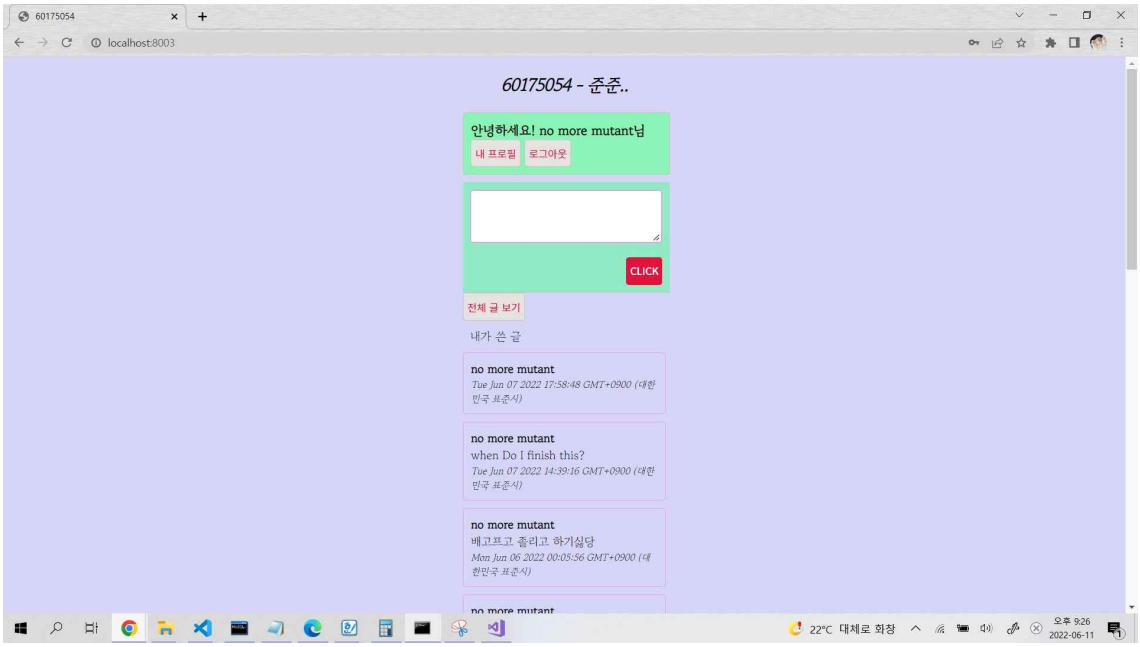


->이렇게 구성되어 있으며, 기존의 예제 코드와 크게 다르진 않지만, 세부적인 내용에서 변화를 줬다.

2. 과정 별 캡쳐 화면

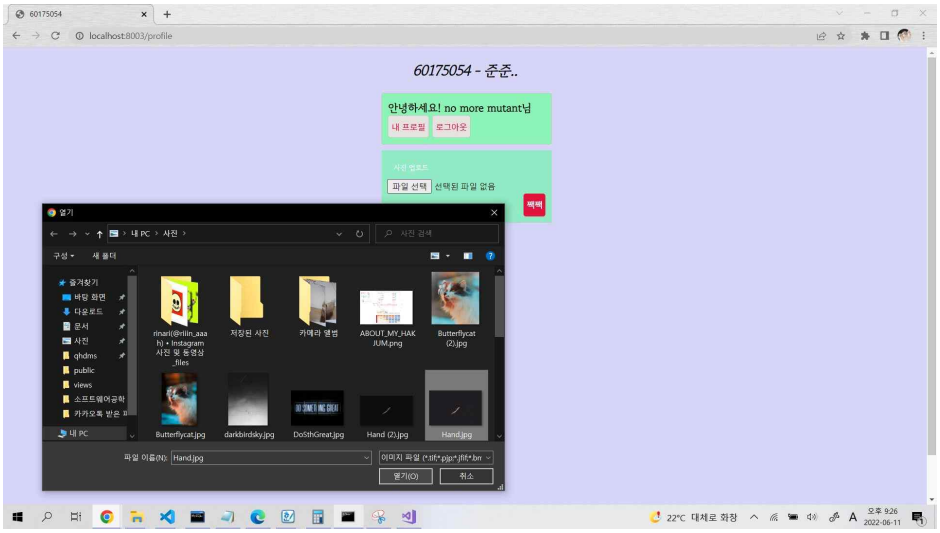
맨 처음 로그인 화면이다. 카카오톡 로그인에 조그만 변화를 주었고, 기존 이메일과 pw를 입력하면

이렇게 나온다.

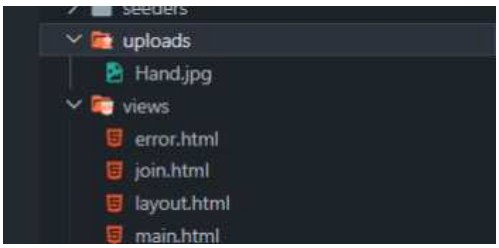


또한 내가 쓴글과 남이 쓴 글을 볼수가 있다. 위의 전체 글 보기를 누르면 다음과 같이 다른유저의 글도 볼 수 있다.

또한 내 프로필을 누르고 다음처럼 파일 선택을 하면
이렇게 사진을 골라서 upload할 수 있다.



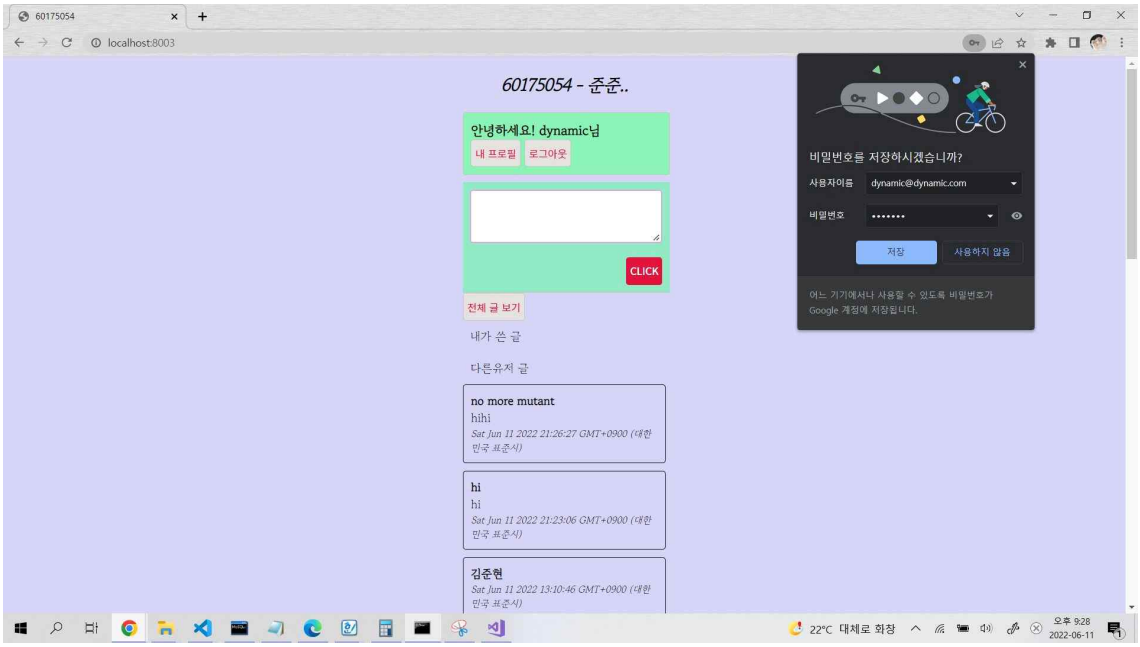
잘 된것같다.



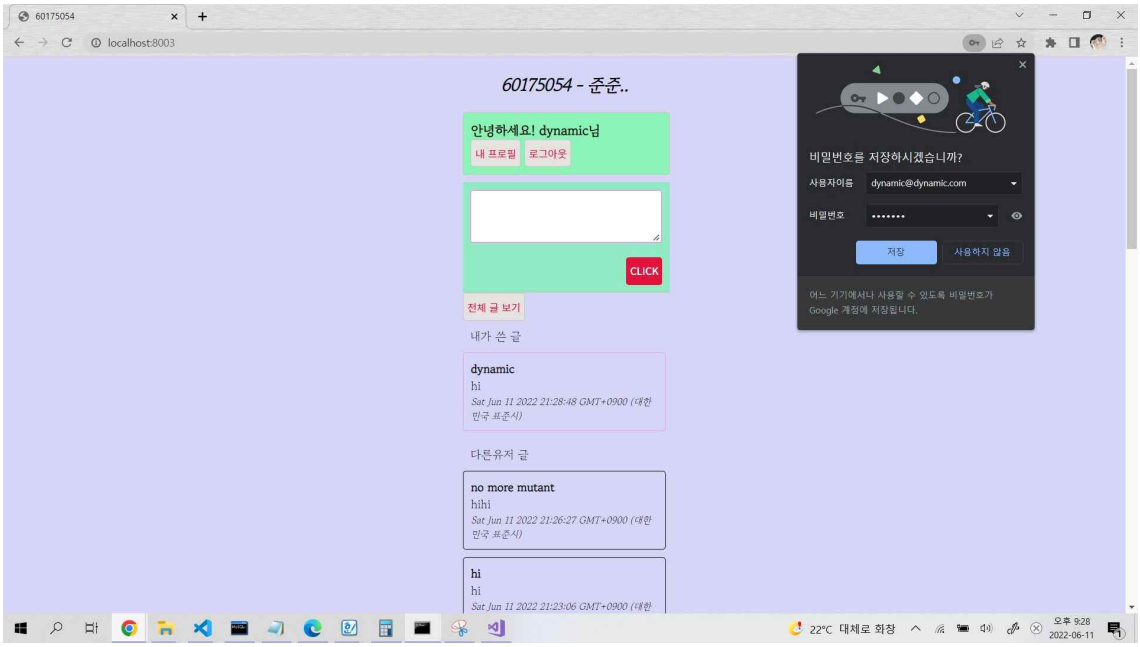
또한 맨처음 로그인 페이지에서 회원가입을 시도해보자

회원가입 완료가 되면

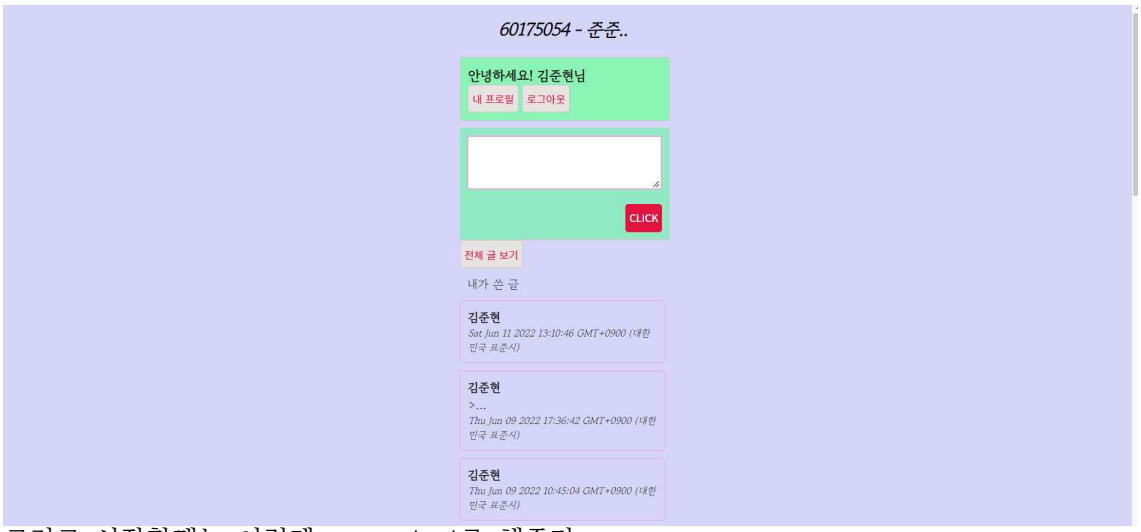
이렇게 새롭게 프로필이 나온다.



글을 쓰면 다시 이렇게 나온다.



카카오로 로그인 할때는 이렇게 본명이 나오게 된다.



그리고 시작할때는 이렇게 npm start로 해준다.

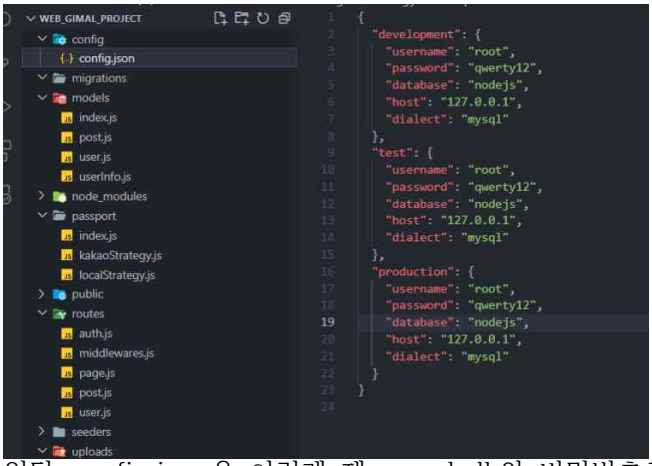
```
C:₩Users₩qhdms₩Downloads₩WEB_GIMAL_Project>npm start

> nodebird@0.0.3 start
> nodemon app

[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
8003 번 포트에서 대기중
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCH
users' AND TABLE_SCHEMA = 'nodejs'
Executing (default): SHOW INDEX FROM `users` FROM `nodejs`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCH
posts' AND TABLE_SCHEMA = 'nodejs'
Executing (default): SHOW INDEX FROM `posts` FROM `nodejs`
데이터베이스 연결 성공
```

3. 소스코드 구성

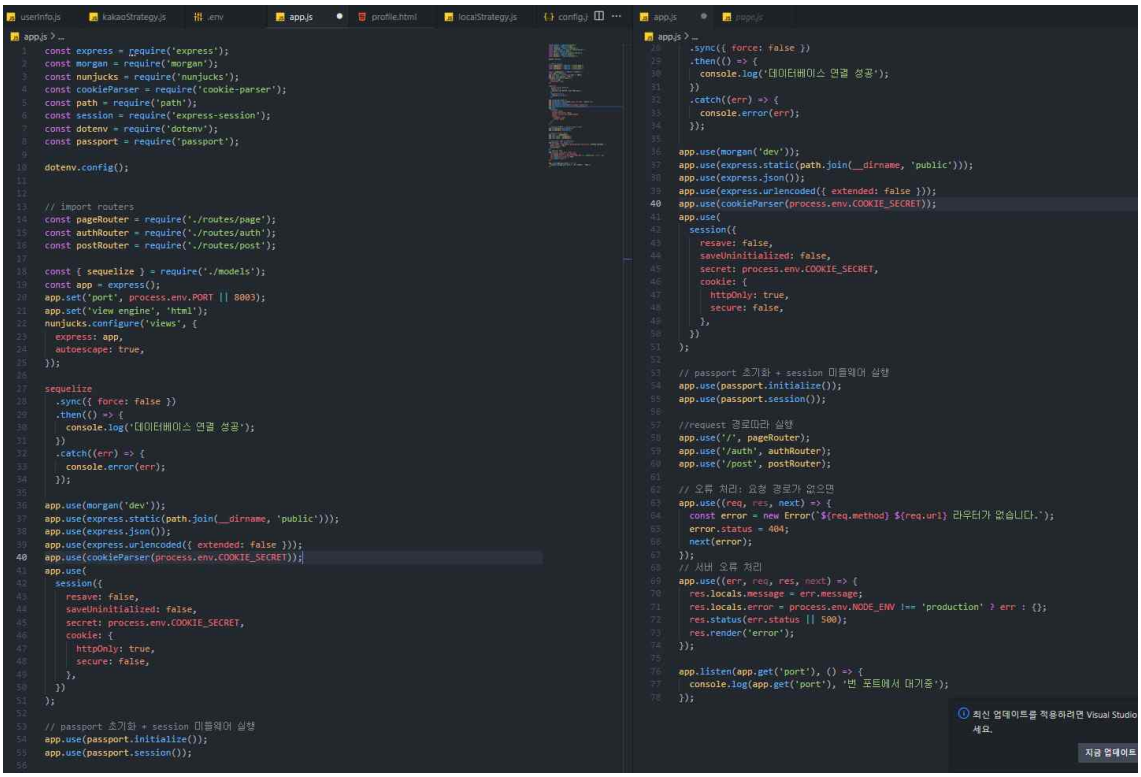
달라진 부분 중점으로 설명하겠습니다.



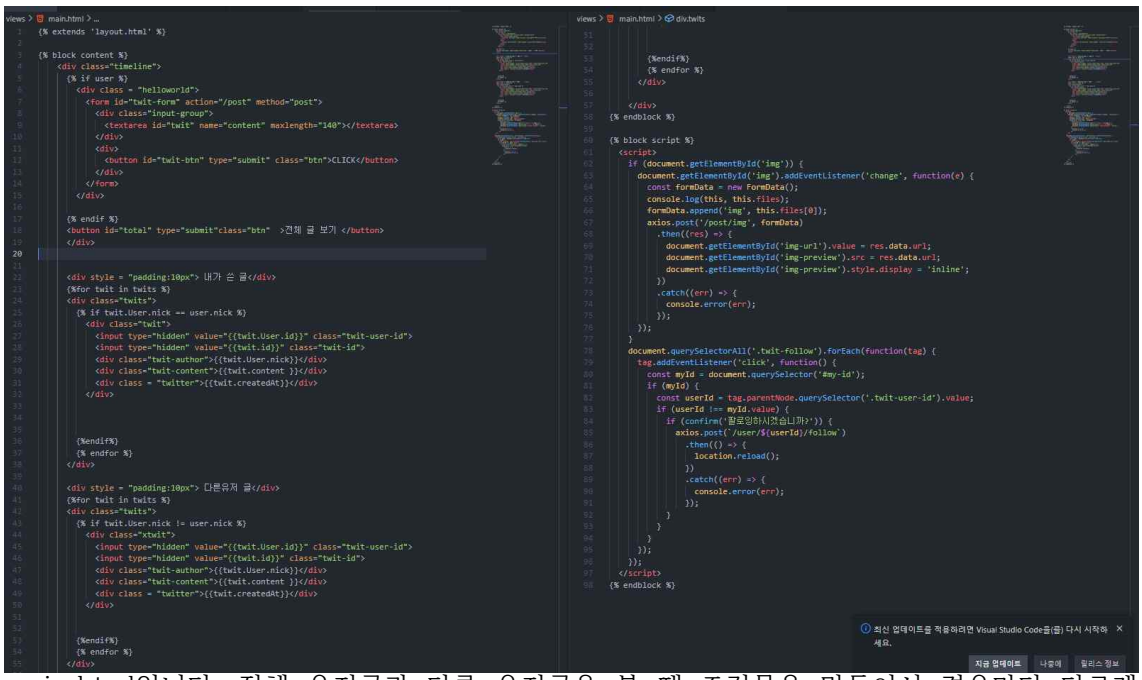
일단 config.json을 이렇게 제 mysql db와 비밀번호로 바꿨습니다.

또한 css에서

카카오톡의 그림을 넣기 위한 작업도 했습니다.



경로따라 실행하고, 오류처리도 따로 했습니다.



```
views > main.html > ...
{% extends 'layout.html' %}

{% block content %}
  <div class="timeline">
    {% if user %}
      <div class = "helloworld">
        <form id="twit-form" action="/post" method="post">
          <div class="input-group">
            <textarea id="twit" name="content" maxlength="140"></textarea>
          </div>
          <div>
            <button id="twit-btn" type="submit" class="btn">CLICK</button>
          </div>
        </form>
      </div>

    {% endif %}
    <button id="total" type="submit"class="btn"  >전체 글 보기 </button>
    </div>


    <div style = "padding:10px"> 내가 쓴 글</div>
    {%for twit in twits %}
    <div class="twits">
      {% if twit.User.nick == user.nick %}
        <div class="twit">
          <input type="hidden" value="{{twit.User.id}}" class="twit-user-id">
          <input type="hidden" value="{{twit.id}}" class="twit-id">
          <div class="twit-author">{{twit.User.nick}}</div>
          <div class="twit-content">{{twit.content }}</div>
          <div class = "twitter">{{twit.createdAt}}</div>
        </div>



      {%endif%}
      {% endfor %}
    </div>

    <div style = "padding:10px"> 다른유저 글</div>
    {%for twit in twits %}
    <div class="twits">
      {% if twit.User.nick != user.nick %}
        <div class="xtwit">
          <input type="hidden" value="{{twit.User.id}}" class="twit-user-id">
          <input type="hidden" value="{{twit.id}}" class="twit-id">
          <div class="twit-author">{{twit.User.nick}}</div>
          <div class="twit-content">{{twit.content }}</div>
          <div class = "twitter">{{twit.createdAt}}</div>
        </div>


      {%endif%}
      {% endfor %}
    </div>
```

```
views > main.html > div.twits


      {%endif%}
      {% endfor %}
    </div>

  </div>
{% endblock %}

{% block script %}
  <script>
    if (document.getElementById('img')) {
      document.getElementById('img').addEventListener('change', function(e) {
        const formData = new FormData();
        console.log(this, this.files);
        formData.append('img', this.files[0]);
        axios.post('/post/img', formData)
          .then((res) => {
            document.getElementById('img-url').value = res.data.url;
            document.getElementById('img-preview').src = res.data.url;
            document.getElementById('img-preview').style.display = 'inline';
          })
          .catch((err) => {
            console.error(err);
          });
      });
    }
    document.querySelectorAll('.twit-follow').forEach(function(tag) {
      tag.addEventListener('click', function() {
        const myId = document.querySelector('#my-id');
        if (myId) {
          const userId = tag.parentNode.querySelector('.twit-user-id').value;
          if (userId !== myId.value) {
            if (confirm('팔로잉하시겠습니까?')) {
              axios.post(`/user/${userId}/follow`)
                .then(() => {
                  location.reload();
                })
                .catch((err) => {
                  console.error(err);
                });
            }
          }
        }
      });
    });
  </script>
{% endblock %}
```

main.html입니다. 전체 유저글과 다른 유저글을 볼 때 조건문을 만들어서 경우마다 다르게 설정했습니다. 팔로우 기능은 실현시키진 못했습니다.

```
routes > post.js > ...
const express = require('express');
const Post = require('../models/post');
const { isLoggedIn } = require('./middlewares');
const router = express.Router();

// post 생성
router
.post('/', isLoggedIn, async (req, res, next) => {
  try {
    const post = await Post.create({
      content: req.body.content,
      UserId: req.user.id,
    });
    res.redirect('/');
  } catch (error) {
    console.error(error);
    next(error);
  }
})
// 수정
.put('/:id', isLoggedIn, async (req, res, next) => {
  try {
    const post = await Post.update({
      where: {id: req.path.id},
      content : req.body.content,
    });
    res.redirect('/');
  } catch (error) {
    console.error(error);
    next(error);
  }
})
// 삭제
.delete('/:id', isLoggedIn, async (req, res, next) => {
  try {
    await Post.destroy({
      where: {id: req.path.id}
    })
  } catch (error) {
    console.error(error);
    next(error);
  }
});

module.exports = router;
```

routes/ post.js입니다. 수정 삭제 할때 필요한 router의 집합입니다.

이상입니다.