과제 3 간단한 Q&A 사이트 개발하기

qna_site_Bootstrap 카드(Card) 스타일 사용

Django로 간단한 **질문-답변 사이트**를 만드는 코드 입니다. 주요 구성은 다음과 같으니, 따라서 해 보기 바랍니다. 본 과제는 따라하기 및 에러 함수 핸들링에 초점을 두고 있습니다.

✅ 구성 요소

- **User 모델** (기본 `auth.User` 사용)
- **Question 모델**: 제목, 내용, 작성자, 작성일
- **Answer 모델**: 내용, 질문, 작성자, 작성일
- 사용자 5명 자동 생성
- 질문과 답변도 일부 자동 생성

✅ 1. Django 프로젝트/앱 생성

```
django-admin startproject qna_site
cd qna_site
python manage.py startapp qa
```

✅ 2. `qa/models.py`

```
from django.db import models
from django.contrib.auth.models import User

class Question(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title
```

```
class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE, r
elated_name='answers')
    content = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Answer to {self.question.title} by {self.author.username}"
```

✅ 3. `qa/admin.py` (관리자에서 보기 편하게)

```
from django.contrib import admin
from .models import Question, Answer

admin.site.register(Question)
admin.site.register(Answer)
```

✅ 4. `qna_site/settings.py` 수정

```
INSTALLED_APPS = [
    ...
    'qa',
]
```

✅ 5. 마이그레이션 & 슈퍼유저 생성

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

✅ 6. `qa/management/commands/seed_data.py` 생성 (자동 생성용)

```
import random
from django.core.management.base import BaseCommand
from django.contrib.auth.models import User
from qa.models import Question, Answer
from faker import Faker

class Command(BaseCommand):
    help = 'Create test users, questions, and answers'

    def handle(self, *args, **kwargs):
        fake = Faker()

        # 사용자 생성
        for i in range(5):
            username = f'user{i+1}'
            if not User.objects.filter(username=username).exists():
                User.objects.create_user(username=username, password='test12
34')
                self.stdout.write(f"Created user: {username}")

        users = list(User.objects.all())

        # 질문 생성
        for _ in range(10):
            author = random.choice(users)
            q = Question.objects.create(
                title=fake.sentence(),
                content=fake.paragraph(),
                author=author
            )
            self.stdout.write(f"Created question: {q.title}")

            # 답변 생성
            for _ in range(random.randint(1, 3)):
                Answer.objects.create(
```

```
            question=q,
            content=fake.paragraph(),
            author=random.choice(users)
        )
        self.stdout.write(f"  → Added answer")
```

✅ 7. seed 실행

```
pip install faker
python manage.py seed_data
```

✅ 8. 기본 뷰 확인

이후엔 `views.py`, `templates`, `urls.py`를 추가해서 질문/답변 리스트, 상세보기, 작성 폼 등을 구현할 수 있어요. 필요하시면 그 부분도 추가로 도와드릴게요!

✅ 요약

- 기본 질문-답변 모델 구성
- 사용자 5명 자동 생성
- Faker로 질문/답변 자동 생성

이 프로젝트에 Django Rest Framework, React 연동, 검색, 좋아요 기능 등도 확장.

이제 `views.py`, `templates`, `urls.py`를 추가해서 **질문/답변 리스트**, **질문 상세 보기**, **질문 작성**, **답변 작성** 기능까지 구현해볼게요.

✅ 전체 목표 기능

- 질문 목록 페이지 (`/`)
- 질문 상세 페이지 + 답변 작성 (`/question/1/`)
- 질문 작성 페이지 (`/ask/`)

✅ 1. qa/views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Question, Answer
from django.contrib.auth.decorators import login_required
from django.utils import timezone
from django.contrib.auth.models import User

def question_list(request):
    questions = Question.objects.order_by('-created_at')
    return render(request, 'qa/question_list.html', {'questions': questions})

def question_detail(request, pk):
    question = get_object_or_404(Question, pk=pk)
    if request.method == 'POST':
        content = request.POST.get('content')
        if content and request.user.is_authenticated:
            Answer.objects.create(
                question=question,
                content=content,
                author=request.user,
                created_at=timezone.now()
            )
            return redirect('question_detail', pk=pk)
    return render(request, 'qa/question_detail.html', {'question': question})

@login_required
def ask_question(request):
    if request.method == 'POST':
        title = request.POST.get('title')
        content = request.POST.get('content')
        if title and content:
            Question.objects.create(
                title=title,
                content=content,
                author=request.user,
                created_at=timezone.now()
            )
```

```
            return redirect('question_list')
    return render(request, 'qa/ask_question.html')
```

✅ 2. qa/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.question_list, name='question_list'),
    path('question/<int:pk>/', views.question_detail, name='question_detail'),
    path('ask/', views.ask_question, name='ask_question'),
]
```

✅ 3. qna_site/urls.py 수정

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('qa.urls')),
]
```

✅ 4. 템플릿 디렉토리 구조 만들기

```
qa/
 └── templates/
     └── qa/
         ├── base.html
         ├── question_list.html
         ├── question_detail.html
         └── ask_question.html
```

✅ 5. base.html

```
<!DOCTYPE html>
<html>
<head>
  <title>QnA Site</title>
</head>
<body>
  <h1><a href="/">QnA</a></h1>
  {% if user.is_authenticated %}
    <p>Logged in as {{ user.username }} | <a href="/logout/">Logout</a>
</p>
  {% else %}
    <p><a href="/login/">Login</a></p>
  {% endif %}
  <hr>
  {% block content %}{% endblock %}
</body>
</html>
```

✅ 6. question_list.html

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>All Questions</h2>
<a href="{% url 'ask_question' %}">Ask a question</a>
<ul>
  {% for question in questions %}
    <li>
      <a href="{% url 'question_detail' question.pk %}">{{ question.title
}}</a> - {{ question.author.username }} ({{ question.created_at|date:"Y-m-
d H:i" }})
    </li>
  {% empty %}
    <li>No questions yet.</li>
  {% endfor %}
```

```
</ul>
{% endblock %}
```

✅ 7. question_detail.html

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>{{ question.title }}</h2>
<p>{{ question.content }}</p>
<p>By {{ question.author.username }} at {{ question.created_at }}</p>

<h3>Answers</h3>
<ul>
  {% for answer in question.answers.all %}
    <li>{{ answer.content }} - {{ answer.author.username }} ({{ answer.cre
ated_at|date:"Y-m-d H:i" }})</li>
  {% empty %}
    <li>No answers yet.</li>
  {% endfor %}
</ul>

{% if user.is_authenticated %}
  <h4>Write an Answer</h4>
  <form method="post">
    {% csrf_token %}
    <textarea name="content" rows="4" cols="50"></textarea><br>
    <button type="submit">Submit Answer</button>
  </form>
{% else %}
  <p><a href="/login/">Login</a> to answer.</p>
{% endif %}
{% endblock %}
```

✅ 8. ask_question.html

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>Ask a New Question</h2>
<form method="post">
  {% csrf_token %}
  <input type="text" name="title" placeholder="Title" size="50"><br><br>
  <textarea name="content" rows="6" cols="60" placeholder="Your question here..."></textarea><br>
  <button type="submit">Post Question</button>
</form>
{% endblock %}
```

✅ 9. 로그인 기능 (간단하게 기본 auth)

`qna_site/urls.py`에 다음도 추가:

```
from django.contrib.auth import views as auth_views

urlpatterns += [
  path('login/', auth_views.LoginView.as_view(template_name='qa/login.html'), name='login'),
  path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

`qa/templates/qa/login.html`:

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>Login</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Login</button>
</form>
{% endblock %}
```

✅ 마지막으로

```
python manage.py runserver 8020
```

- 접속: http://127.0.0.1:8020/
- 로그인: `/login/` (예: user1 / test1234)
- 질문 목록, 질문 상세, 질문 작성 가능!

✅ 다음 확장 아이디어

- 답변 수정/삭제
- 좋아요/추천 기능
- 검색 기능
- DRF + React 프론트엔드 연동

해당 오류는 다음과 같은 경우에 발생합니다:

❌ 문제: 로그아웃 요청이 GET 방식으로 이루어짐

Django의 기본 `LogoutView`는 **POST 방식만 허용**합니다. 그래서 단순히 `<a href="/logout/">Logout</a>` 링크를 클릭하면 **405 Method Not Allowed** 에러가 발생합니다.

✅ 해결 방법 2가지

🔶 방법 1: 로그아웃을 form POST 버튼으로 처리 (추천)

`base.html` 또는 로그아웃을 넣고 싶은 템플릿에서 다음처럼 처리:

```
<form action="{% url 'logout' %}" method="post" style="display:inline;">
  {% csrf_token %}
  <button type="submit">Logout</button>
</form>
```

이렇게 하면 Django가 허용하는 방식대로 **POST**로 로그아웃이 이루어집니다.

🔶 방법 2: `LogoutView`를 GET도 허용하도록 커스터마이징 (비추천)

```
from django.contrib.auth.views import LogoutView
from django.views.generic.base import RedirectView

class CustomLogoutView(LogoutView):
    def get(self, request, *args, **kwargs):
        return self.post(request, *args, **kwargs)
```

그리고 `urls.py`에서 교체:

```
from .views import CustomLogoutView

path('logout/', CustomLogoutView.as_view(), name='logout'),
```

하지만 이 방식은 보안상 추천되지 않습니다. 기본 동작대로 POST를 사용하는 게 안전하고 권장돼요.

✅ 정리

방법	설명	추천
form + POST 방식	Django 공식 방식. 오류 없음.	✅ 매우 추천
GET 허용하도록 뷰 수정	작동은 하지만 보안 우려 있음	❌ 비추천

로그인/로그아웃 상태에 따라 메뉴도 자동으로 바뀌게 만듭니다.

회원가입 기능도 필요합니다

여전히 `GET /logout/` 요청이 발생하고 있어서 **405 Method Not Allowed** 오류가 납니다.

이건 Django의 기본 `LogoutView`가 **POST 요청만 허용**하기 때문이에요.

✅ 확실한 해결 방법 요약

✔️ 1. 반드시 링크 대신 form으로 POST 요청해야 합니다!

`base.html`에서 아래처럼 수정했는지 확인해 주세요:

```
<form action="{% url 'logout' %}" method="post" style="display:inline;">
  {% csrf_token %}
  <button type="submit" style="background:none; border:none; color:blue; text-decoration:underline; cursor:pointer;">
    Logout
  </button>
</form>
```

> 🔷 절대 Logout 형태로 하지 마세요.
>
> 🔷 `<form>` + `POST` + `csrf_token` 필수입니다.

✔️ 2. `urls.py` 설정은 아래처럼 되어 있어야 합니다

```
from django.contrib.auth import views as auth_views

urlpatterns += [
  path('login/', auth_views.LoginView.as_view(template_name='qa/login.html'), name='login'),
  path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

❗ 여전히 안 된다면?

아직도 `<a href="/logout/">` 링크로 로그아웃 시도하고 있는 HTML이 남아 있는지 확인하세요.

오류 로그를 보면 Django는 여전히 `GET /logout/` 요청을 받고 있어요:

```
[23/Mar/2025 13:46:37] "GET /logout/ HTTP/1.1" 405 0
```

✅ 빠르게 체크리스트

항목	체크
🔲 `Logout`을 `<a>`가 아닌 `<form method="post">`로 바꿨나요?	✅
🔲 `csrf_token`을 폼 안에 넣었나요?	✅

🔲 `urls.py`에서 `LogoutView`를 제대로 연결했나요?	✅

필요하다면 로그아웃을 GET으로 강제로 허용하는 커스텀 뷰도 만들어드릴 수 있지만, 보안상 **POST 방식**을 사용하는 지금 방법이 가장 안전하고 권장됩니다.

아래는 `base.html` 전체 코드로, **로그인 상태에 따라 메뉴가 다르게 보이고, 로그아웃은 POST 방식으로 동작**하도록 구성했습니다. 디자인은 심플하게 유지하고 기능 중심으로 작성했습니다.

✅ qa/templates/qa/base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>QnA Site</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
        }
        header {
            margin-bottom: 20px;
        }
        nav {
            margin-bottom: 20px;
        }
        button.logout-btn {
            background: none;
            border: none;
            padding: 0;
            color: blue;
            cursor: pointer;
            text-decoration: underline;
            font-size: 1em;
        }
```

```
    </style>
</head>
<body>
    <header>
        <h1><a href="/">QnA</a></h1>
    </header>

    <nav>
        {% if user.is_authenticated %}
            <span>👤 {{ user.username }}</span> |
            <a href="{% url 'ask_question' %}">Ask Question</a> |
            <form action="{% url 'logout' %}" method="post" style="display:inli
ne;">
                {% csrf_token %}
                <button type="submit" class="logout-btn">Logout</button>
            </form>
        {% else %}
            <a href="{% url 'login' %}">Login</a>
        {% endif %}
    </nav>

    <hr>

    {% block content %}
    {% endblock %}
</body>
</html>
```

✅ 요약 기능

기능	설명
로그인 상태	사용자 이름, 질문 작성 버튼, 로그아웃 버튼 (POST) 표시
로그아웃	버튼 클릭 시 POST 요청으로 처리되며 405 오류 없음
비로그인 상태	로그인 링크만 표시됨
스타일	간단한 CSS 포함 (선택)

이제 위 템플릿을 적용하면 `/logout/`에서 405 오류는 더 이상 발생하지 않으며, 로그인/로그아웃 상태에 따라 UI도 깔끔하게 바뀝니다.

아래는 Bootstrap 5를 적용한 `base.html` 템플릿입니다.

로그인/로그아웃, 네비게이션 메뉴, 질문 작성 버튼 등을 모두 **Bootstrap 스타일**로 꾸몄습니다.

✅ Bootstrap 5 적용된 `qa/templates/qa/base.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>QnA Site</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light px-4">
    <a class="navbar-brand" href="/">QnA</a>
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav ms-auto">
        {% if user.is_authenticated %}
          <li class="nav-item">
            <span class="nav-link">👤 {{ user.username }}</span>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'ask_question' %}">Ask Question</a>
          </li>
          <li class="nav-item">
            <form action="{% url 'logout' %}" method="post" class="d-inline">
              {% csrf_token %}
              <button type="submit" class="btn btn-link nav-link" style
```

```
="display:inline; padding: 0;">Logout</button>
                </form>
              </li>
            {% else %}
              <li class="nav-item">
                <a class="nav-link" href="{% url 'login' %}">Login</a>
              </li>
            {% endif %}
          </ul>
        </div>
    </nav>

    <div class="container mt-4">
      {% block content %}
      {% endblock %}
    </div>

    <!-- Optional Bootstrap JS (for future interactive components) -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

✅ 특징

구성 요소	설명
✅ Bootstrap CDN	Bootstrap 5 적용 (CDN으로 간편하게)
✅ 네비게이션 바	`navbar` 으로 상단 메뉴 구성
✅ 로그인 상태	사용자 이름, 질문하기, 로그아웃 버튼 표시
✅ 로그아웃	`POST` 방식으로 버튼 처리 (405 오류 없음)
✅ 전체 콘텐츠	`container` 내부에서 정렬

추가 가능

- 질문 목록 카드 스타일로 정리
- 다크 모드 / 반응형 메뉴

- Bootstrap 폼 적용
- 회원가입 페이지
- 알림/메시지 UI

질문 작성 폼, 답변 작성 폼, 로그인 폼에 **Bootstrap 5 폼 스타일**을 적용합니다

HTML 폼 요소에 `form-control`, `btn`, `form-label`, `mb-3` 같은 클래스를 추가해서 예쁘게 꾸밈니다.

✅ 1. 질문 작성 폼 (`ask_question.html`)

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>Ask a New Question</h2>
<form method="post" class="mt-4">
  {% csrf_token %}
  <div class="mb-3">
    <label for="title" class="form-label">Title</label>
    <input type="text" name="title" id="title" class="form-control" placeh
older="Enter your question title">
  </div>
  <div class="mb-3">
    <label for="content" class="form-label">Content</label>
    <textarea name="content" id="content" class="form-control" rows
="6" placeholder="Describe your question in detail..."></textarea>
  </div>
  <button type="submit" class="btn btn-primary">Post Question</button>
</form>
{% endblock %}
```

✅ 2. 답변 작성 폼 (질문 상세 보기 안에 포함 – `question_detail.html`)

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>{{ question.title }}</h2>
<p>{{ question.content }}</p>
<p><small>By {{ question.author.username }} at {{ question.created_at }}
</small></p>

<hr>
<h4>Answers</h4>
<ul class="list-group mb-4">
  {% for answer in question.answers.all %}
    <li class="list-group-item">
      {{ answer.content }}<br>
      <small class="text-muted">— {{ answer.author.username }} at {{ an
swer.created_at|date:"Y-m-d H:i" }}</small>
    </li>
  {% empty %}
    <li class="list-group-item">No answers yet.</li>
  {% endfor %}
</ul>

{% if user.is_authenticated %}
<h5>Write an Answer</h5>
<form method="post">
  {% csrf_token %}
  <div class="mb-3">
    <textarea name="content" class="form-control" rows="4" placeholder
="Write your answer here..."></textarea>
  </div>
  <button type="submit" class="btn btn-success">Submit Answer</butto
n>
</form>
{% else %}
<p><a href="{% url 'login' %}">Login</a> to write an answer.</p>
{% endif %}
{% endblock %}
```

✅ 3. 로그인 폼 (`login.html`)

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>Login</h2>
<form method="post" class="mt-4">
  {% csrf_token %}
  {% for field in form %}
    <div class="mb-3">
      <label class="form-label">{{ field.label }}</label>
      {{ field }}
      {% if field.errors %}
        <div class="text-danger small">{{ field.errors|striptags }}</div>
      {% endif %}
    </div>
  {% endfor %}
  <button type="submit" class="btn btn-primary">Login</button>
</form>
{% endblock %}
```

> ✔️ Django 기본 로그인 폼은 {% for field in form %}로 렌더링하면 Bootstrap과도 잘 어울립니다.

✅ 요약

적용 대상	결과
질문 작성 폼	Bootstrap 입력 UI
답변 작성 폼	카드식 + 버튼
로그인 폼	자동 반복 렌더링 + 스타일링

회원가입 폼, 질문/답변 수정 폼도 Bootstrap으로 만듭니다.

`TemplateDoesNotExist at /login/` 에러는 Django가 `qa/base.html` 템플릿을 찾지 못해서 발생한 오류예요. 이건 일반적으로 **템플릿 디렉토리 구조**가 잘못됐거나, **템플릿 경로 설정이 누락**된 경

우 발생합니다.

✅ 해결 순서

✅ 1. 템플릿 폴더 구조 확인

폴더 구조가 다음과 같은지 확인해 주세요:

```
your_project/
├── qa/
│   └── templates/
│       └── qa/
│           ├── base.html
│           ├── login.html
│           ├── ask_question.html
│           ├── question_list.html
│           └── question_detail.html
```

즉, `qa/templates/qa/base.html` 에 **정확히** 존재해야 해요.

✅ 2. `settings.py` 에 `TEMPLATES` 설정 확인

`DIRS` 부분이 **비어 있거나 잘못됐다면** Django가 템플릿을 못 찾습니다.

```
# settings.py

import os

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],  # 앱 내부 템플릿만 쓸 거면 이대로 둬도 됨
        'APP_DIRS': True,  # 이게 True여야 앱 내 templates 폴더 자동 인식!
        'OPTIONS': {
            'context_processors': [
                ...
            ],
        },
```

```
    },
]
```

✔️ `APP_DIRS: True` 반드시 확인하세요!

✔️ `qa` 앱이 `INSTALLED_APPS`에 등록되어 있어야 템플릿도 인식됩니다.

✅ 3. 템플릿 확장 태그 확인

`login.html`, `ask_question.html` 등에서 다음처럼 되어 있어야 합니다:

```
{% extends 'qa/base.html' %}
```

> ✔️ qa/base.html로 정확하게 일치해야 해요!

✅ 빠른 점검 리스트

항목	예시	확인 여부
템플릿 경로	`qa/templates/qa/base.html`	✅
`APP_DIRS: True`	`settings.py` 안에 있음	✅
`qa` 앱이 INSTALLED_APPS에 포함됨	`'qa',`	✅
`{% extends 'qa/base.html' %}` 정확하게 작성됨	✅	

❓ base.html이 없다면?

👉 `qa/templates/qa/base.html` 다시 생성

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>QnA Site</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
```

```
  <nav class="navbar navbar-expand-lg navbar-light bg-light px-4">
    <a class="navbar-brand" href="/">QnA</a>
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav ms-auto">
        {% if user.is_authenticated %}
          <li class="nav-item">
            <span class="nav-link">👤 {{ user.username }}</span>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'ask_question' %}">Ask Qu
estion</a>
          </li>
          <li class="nav-item">
            <form action="{% url 'logout' %}" method="post" class="d-i
nline">
              {% csrf_token %}
              <button type="submit" class="btn btn-link nav-link" style
="display:inline; padding: 0;">Logout</button>
            </form>
          </li>
        {% else %}
          <li class="nav-item">
            <a class="nav-link" href="{% url 'login' %}">Login</a>
          </li>
        {% endif %}
      </ul>
    </div>
  </nav>

  <div class="container mt-4">
    {% block content %}
    {% endblock %}
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstr
ap.bundle.min.js"></script>
</body>
</html>
```

질문 목록을 **Bootstrap 카드(Card) 스타일**로 정리합니다.

질문 하나하나를 카드 컴포넌트로 보여주면 가독성도 좋아지고 보기에도 훨씬 예뻐집니다.

✅ qa/templates/qa/question_list.html

```
{% extends 'qa/base.html' %}
{% block content %}
<h2>All Questions</h2>

<div class="d-flex justify-content-end mb-3">
  <a href="{% url 'ask_question' %}" class="btn btn-primary">Ask Questi
on</a>
</div>

<div class="row row-cols-1 row-cols-md-2 g-4">
  {% for question in questions %}
    <div class="col">
      <div class="card h-100">
        <div class="card-body">
          <h5 class="card-title">
            <a href="{% url 'question_detail' question.pk %}">{{ questio
n.title }}</a>
          </h5>
          <p class="card-text text-truncate">{{ question.content }}</p>
        </div>
        <div class="card-footer text-muted">
          By {{ question.author.username }} | {{ question.created_at|dat
e:"Y-m-d H:i" }}
        </div>
      </div>
    </div>
  {% empty %}
    <p>No questions yet.</p>
```

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

✅ 설명

항목	설명
`row row-cols-1 row-cols-md-2`	반응형 카드 1~2열 배치
`card`	Bootstrap 카드 컴포넌트
`text-truncate`	본문 내용 한 줄 요약
`card-footer`	작성자와 작성 시간 표시
`a href="{% url 'question_detail' %}"`	클릭 시 질문 상세 보기로 이동

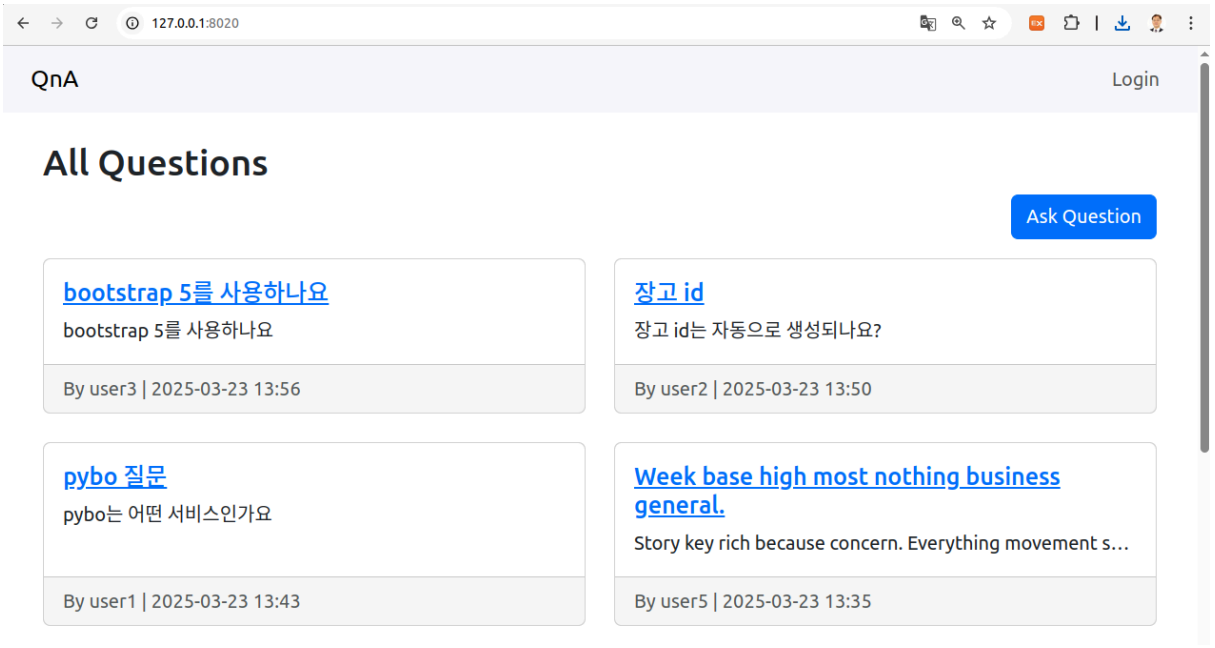
💡 더 발전시키고 싶다면?

- `답변 수` 뱃지 표시
- `좋아요`, `태그`, `조회수` 등 추가 메타 정보
- 검색창, 정렬 필터
- 카드 색상 구분 (ex. 인기 질문 강조)

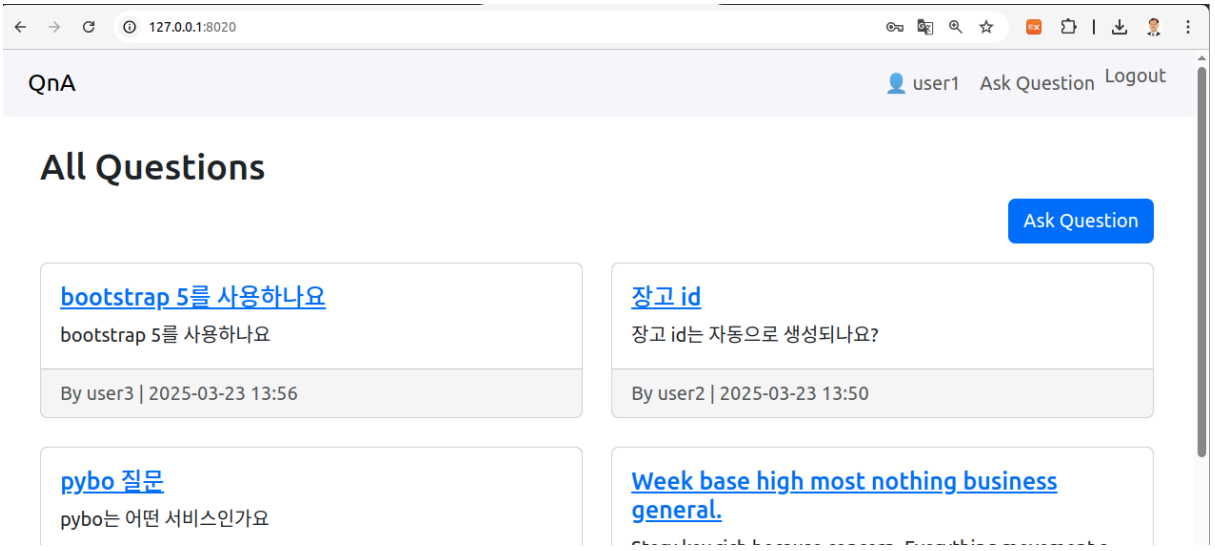
이 디자인을 기반으로 검색창, 필터 버튼, 페이징 등도 가능

과제 3 간단한 Q&A 사이트 개발하기

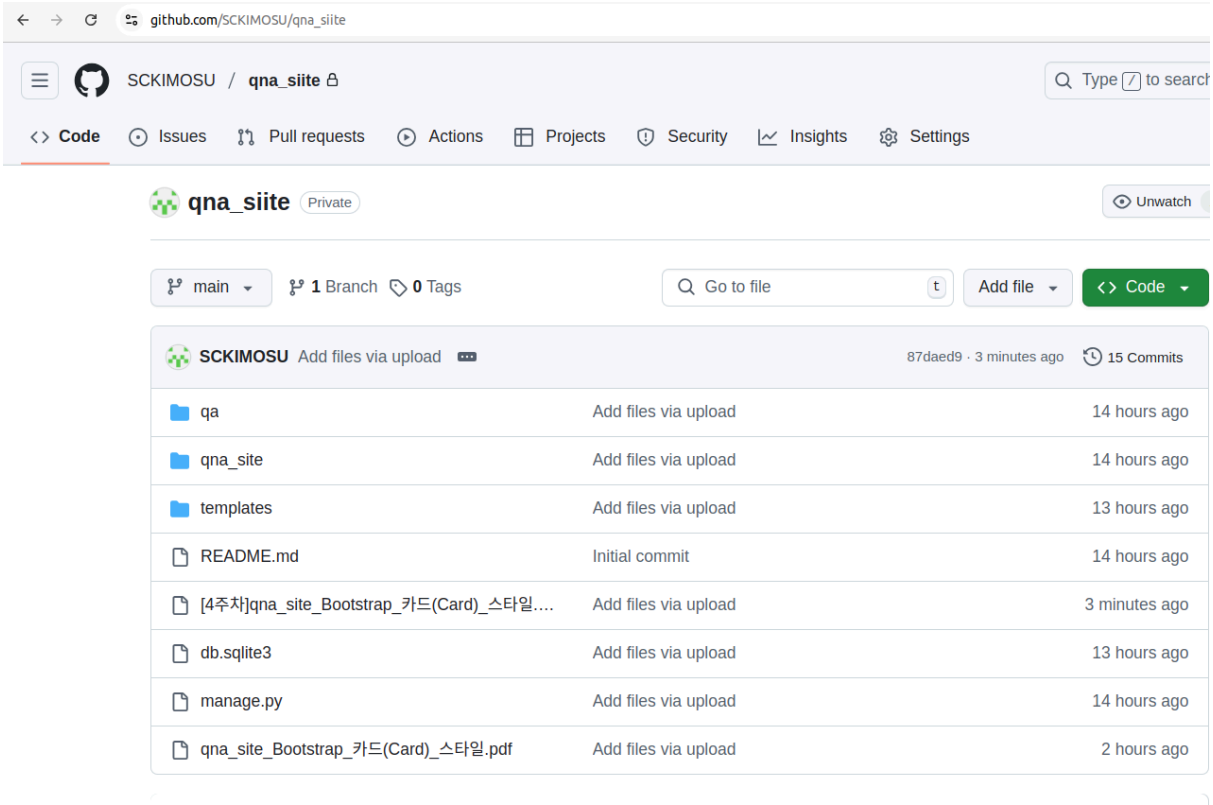
1. **제출일정**
    a. **4월 2일 오후 1:30분**
    b. **이후 늦게 제출된 과제 접수는 받지 않습니다**
    c. 가상강의실 제출
2. 제출내용
    a. 127.0.0.1:8020 으로 접속해서 아래와 같은 화면이 생성되는 지 확인하고 캡쳐해서 제출한다 (3점)



b. user1/test1234 로 로그인을 시도해보고 결과 화면을 캡쳐해서 제출한다 (3점)



c. 자신의 깃허브 사이트에 작성한 코드를 업로드하고 화면을 캡쳐해서 제출한다(2점)



d. 노션(notion)에 자신이 만든 코드를 입력하고 화면을 캡쳐해서 제출한다(2점)