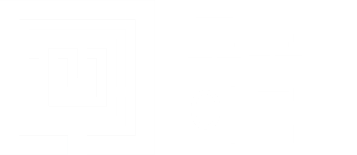


|  |
| --- |
| Live in 프로젝트 |
|  |
| 2020 멀티미디어처리 프로젝트  16 함진경  17 김지현  17 최찬영 |



|  |
| --- |
| 목차개요  * 1. 소개   2. 문제의식   3. 기대효과  기획 2-1. 주요기능  2-2. 유사 서비스와의 차별성  2-3. 추가확장가능성 요구사항 3-1. 사용자 요구사항  3-2. 기능적 요구사항  3-3. 비기능적 요구사항 |
| 개발환경설계구현 |
|  |

# 개요

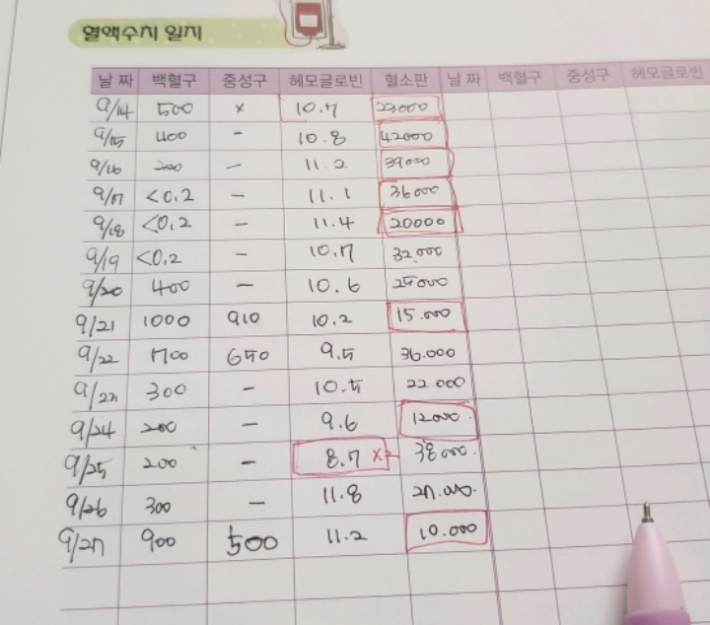
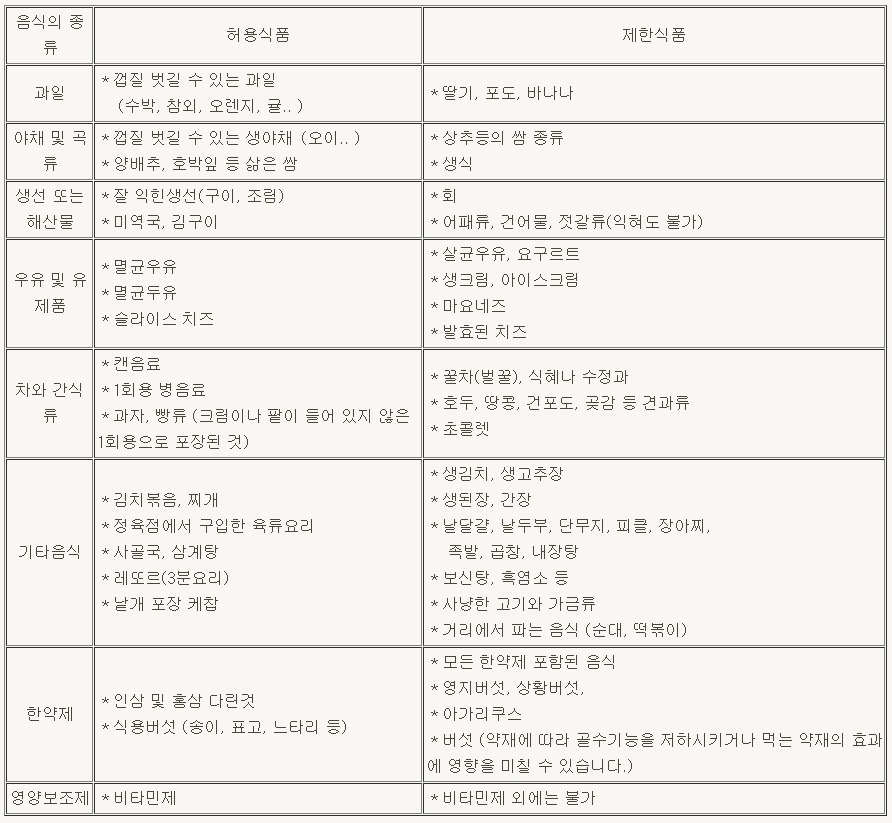
## 소개

Live in프로젝트는 혈액암, 그 중에서도 ‘재생불량성 빈혈’을 앓고 있는 사람들을 위한 웹서비스이다.

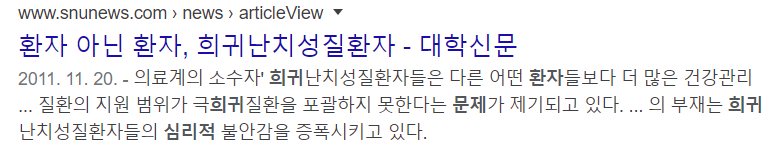
흔히 재생불량성 빈혈을 줄여서 ‘재빈’이라고 부른다. 우리는 재빈을 re:bin(리빈)으로 풀어서 이와 소리가 비슷한 단어인 Live in으로 프로젝트명을 정했다. Live in은 ‘희귀난치병인 재생불량성 빈혈로부터 살아남다’라는 뜻도 있지만, ‘같은 아픔을 지닌 사람들이 이곳에 공존하다’라는 의미도 담고 있다.

## 문제의식

1. 재생불량성 빈혈과 같은 혈액암 환자들은 매일같이 자신의 수치를 확인하고, 그 수치에 따른 여러 제약 조건을 따라야한다.

**** ****

1. 희귀난치병 환자와 가족들이 힘들어하는 것은 무엇보다 심리적인 문제가 크다.



## 기대효과

질병에 대한 치료와 완전한 재활은 우리의 영역 밖이다. 우리는 환자의 외로움과 홀로 준비해야하는 것들에 포커스를 맞추어 희귀난치병의 희소성을 지우고, 그 누구보다 서로의 감정을 잘 알고 이해할 수 있는 같은 처지의 사람들과 공감하며 함께 살아갈 수 있도록 도와준다.

여기저기 흩어져 있는 정보들을 한 곳에 모아 둠으로써 보다 쉽게 원하는 정보를 찾아볼 수 있다.

# 기획

## 2-1. 주요기능

[마이페이지]

* 수치 기록 및 확인
* 나만의 일기장
* 병원일정관리 캘린더

[투병일기]

* 같은 병을 앓고 있는 사람들의 이야기를 보며 공감하고 힘을 얻을 수 있는 공간

[병원, 의사 찾기]

* 거리/지역별 병원 검색
* 리뷰

## 2-2. 유사서비스와의 차별성

[커뮤니티] + [자기관리]

## 2-3. 추가확장가능성

[정보]

* 유용한 정보 공유
* 최신 뉴스 업데이트

[온라인 상담]

* 상담게시판에 글을 작성하면 전문가가 직접 답글을 달아줄 수 있는 온라인 상담

[쇼핑]

* 맞춤 용품이나 음식 등 관련 쇼핑몰 연결 또는 추천

# 요구사항

## 3-1. 사용자 요구사항

1. 로그인/회원가입을 할 수 있다.
2. 로그인을 하면 메인 페이지에서 가장 최근의 일정과 해당 일정의 내용을 확인할 수 있다.
3. 투병일기 게시판에 있는 게시글 중 가장 최근에 작성된 4개의 글의 제목과 내용 일부를 메인 페이지에서 미리 볼 수 있다.
4. 사용자가 찾고 싶은 병원/의사를 검색하여 병원의 사진과 이름, 위치를 알 수 있다.
5. 지역별 병원/의사를 검색할 수 있다.
6. 게시글을 작성/확인/수정/삭제할 수 있다.
7. 게시글에 대한 댓글을 작성/확인/삭제할 수 있다.
8. 호중구 수, 혈소판 수, 적혈구 수를 입력할 수 있다.
9. 입력한 수치에 따라 먹을 수 있는 음식과 먹으면 안되는 음식들의 목록을 볼 수 있다.
10. 캘린더에 병원일정을 작성/확인/수정/삭제할 수 있다.
11. 비밀일기를 작성/확인/수정/삭제할 수 있다.
12. 다이어리에 그 날의 수치와 기분(행복해요,보통이에요,별로에요,슬퍼요,화나요)을 입력할 수 있다.

# 개발환경

1. Client-Side

그리기, 표지판이(가) 표시된 사진

자동 생성된 설명그리기, 표지판이(가) 표시된 사진

자동 생성된 설명그리기, 음식이(가) 표시된 사진

자동 생성된 설명

Html , Sass : Syntatically css 디자인 저작도구 , Js : interactive web

2. Server-Side

그리기, 표지판이(가) 표시된 사진

자동 생성된 설명

파이썬 프레임워크 – Django Framework

3. 데이터베이스

팬, 테이블이(가) 표시된 사진

자동 생성된 설명

장고 내장 DB SQLite3 사용

4. 기타

개체, 시계이(가) 표시된 사진

자동 생성된 설명시계이(가) 표시된 사진

자동 생성된 설명

사용언어 : Python , 웹 배포: Heroku , 형상관리 도구: Git

4-1. Heroku 배포 URL

완성된 결과물은 <https://livein.herokuapp.com/> 에서 확인해보실 수 있습니다.

# 5. 설계 및 구현

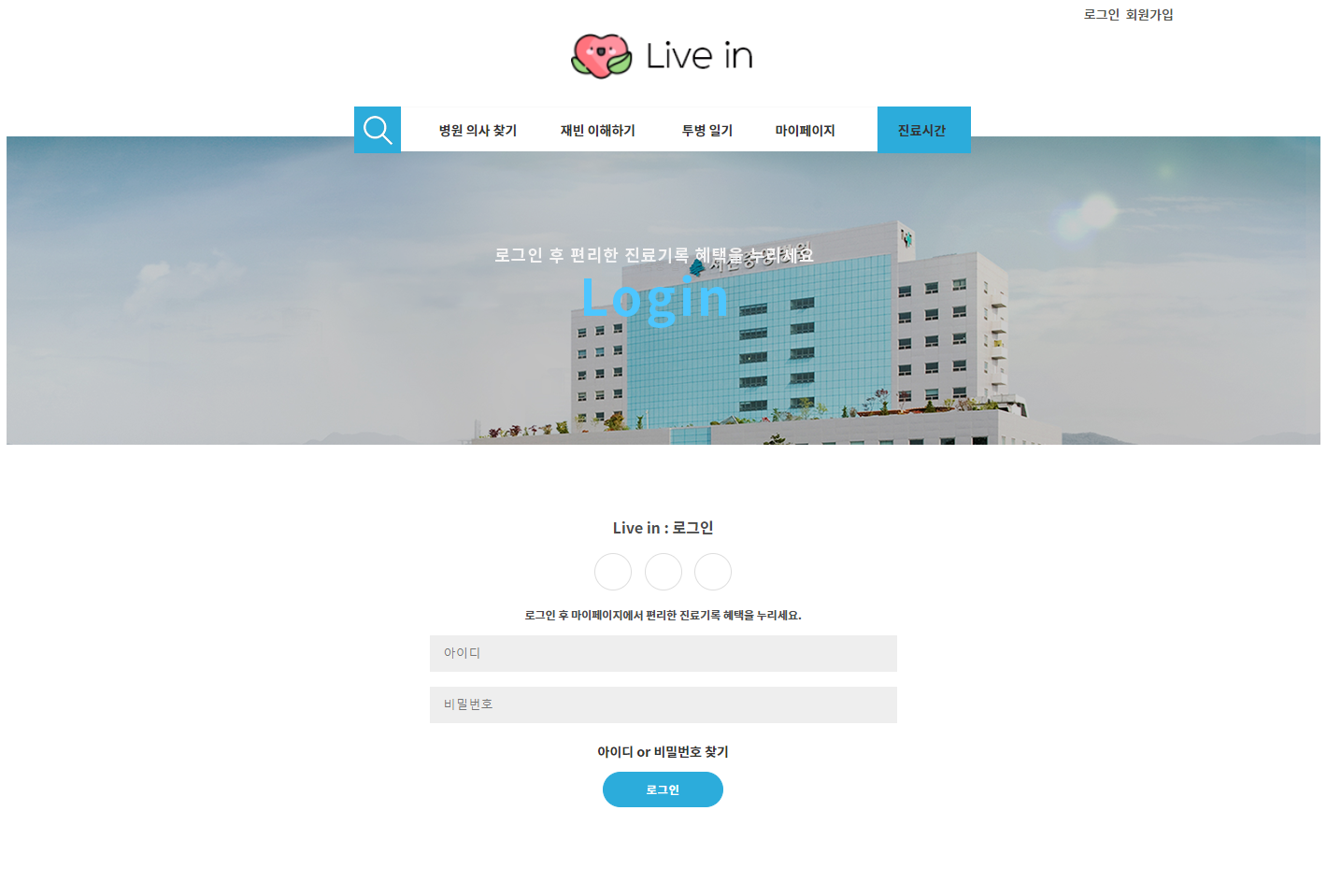
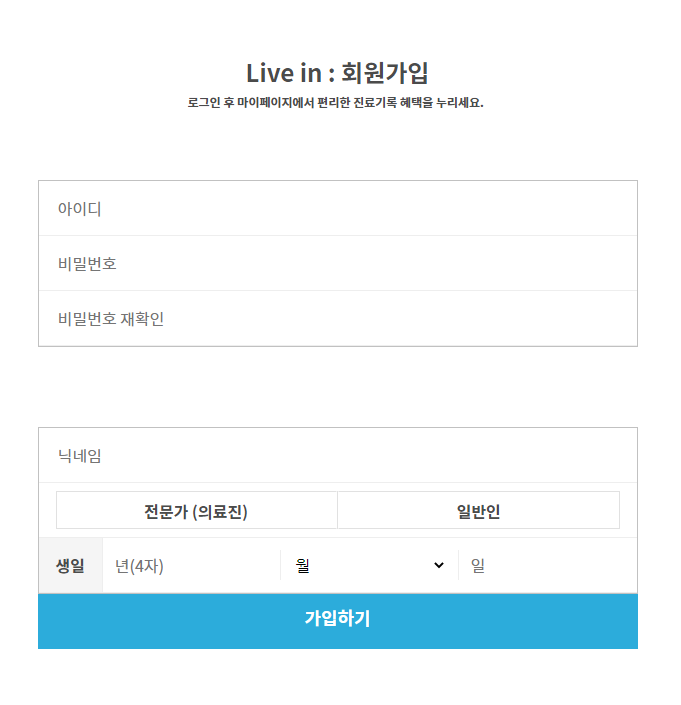
0. 메인페이지

스크린샷이(가) 표시된 사진

자동 생성된 설명

메인페이지는 관련된 내용 또는 게시판의 일부내용을 노출하여 사용자들이 해당정보를 빠르게 찾아보기 위해 연결하는 역할을 담당한다. 더불어 이 프로젝트의 주된 기능인 수치를 입력할 경우, 먹을 수 있는 식품과 제한적인 식품을 구별하여 보여주는 기능을 담당한다.

0-0. 로그인 및 회원가입

진행된 UI들의 형태입니다. 초기 모델 구축시에 지정된 아이디, 비밀번호, 이름, 전문가여부(Boolean), 생년월일을 input data로 만들었음

0-1. Modal창 – 수치에 따른 허용∙제한 식품



[메인페이지 노출 화면]



$('.food\_list button').mouseover(function(){

        $(this).css('background-color','#fff');

        $(this).css('color','#8dc4d8');

    });

    $('.food\_list button').mouseleave(function(){

        $(this).css('background-color','transparent');

        $(this).css('color','#fff');

    });

    $('#submit').click(function(){

        var rbc = $('input#rbc').val();

        var pt = $('input#pt').val();

        var wbc = $('input#wbc').val();

        $('.modal\_back').css('display','block');

        if(rbc<500||pt<20000||wbc<20000){

            $('.modal3').css('display','block');

        }else if(rbc<1000||pt<50000||wbc<60000){

            $('.modal2').css('display','block');

        }else{

            $('.modal1').css('display','block');

        }

    });

해당 Modal창은 순수 JQuery만을 이용하여 제작되었다. val() 메소드를 통해 가지고 온 input태그의 데이터를 기반으로 수치를 계산하여 중증도를 표현하였다. 적혈구 또는 백혈구 또는 그물적혈구 수 중 하나의 수치라도 기준 수치에서 떨어질 경우, 중증에 가깝게 표시되도록 구현되었다.

∴ 중증에 가까울수록 제한되는 음식이 표에 추가된다.

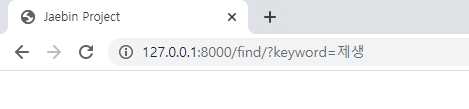
1. 병원∙의사 찾기 페이지

스크린샷이(가) 표시된 사진

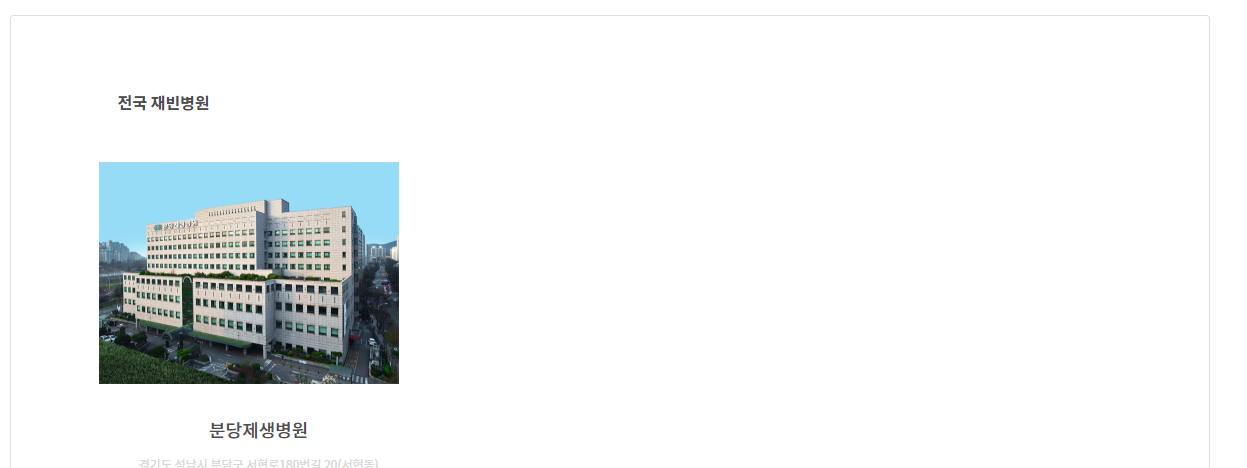
자동 생성된 설명

병원∙의사찾기 페이지는 재생불량성 빈혈에 관한 병원 및 병동이 제한적이다는 것을 감안하여 전국에 존재하는 혈액병원 및 재빈병원의 위치와 정보를 보여주는 페이지이다.

1.1 Keyword를 통한 검색

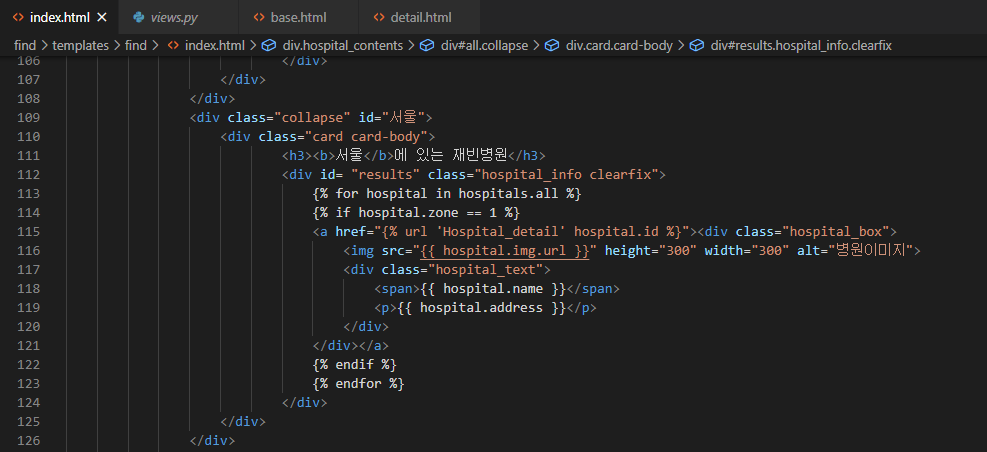


검색어 입력을 통해 keyword 값을 입력받으면,

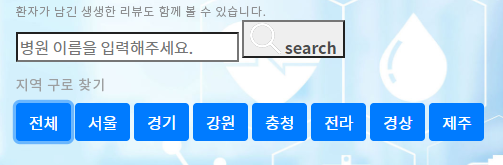


키워드와 관련된 병원의 리스트만 반환받을 수 있다.

1.2 전국 지역 버튼 click시 나타나는 지역에 해당되는 병원 리스트



hospital.zone에 값을 부여하였고 데이터베이스에 해당 병원의 정보를 저장할 때 지역정보를 저장해주었다.

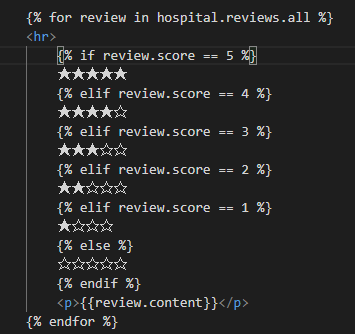


아래 버튼은 부트스트랩에서 가져온 버튼이다. 해당 버튼을 클릭할 시 해당 지역에 있는 재빈병원에 대한 리스트를 볼 수 있다.

1.3 병원의 정보를 자세히 볼 수 있는 디테일 페이지

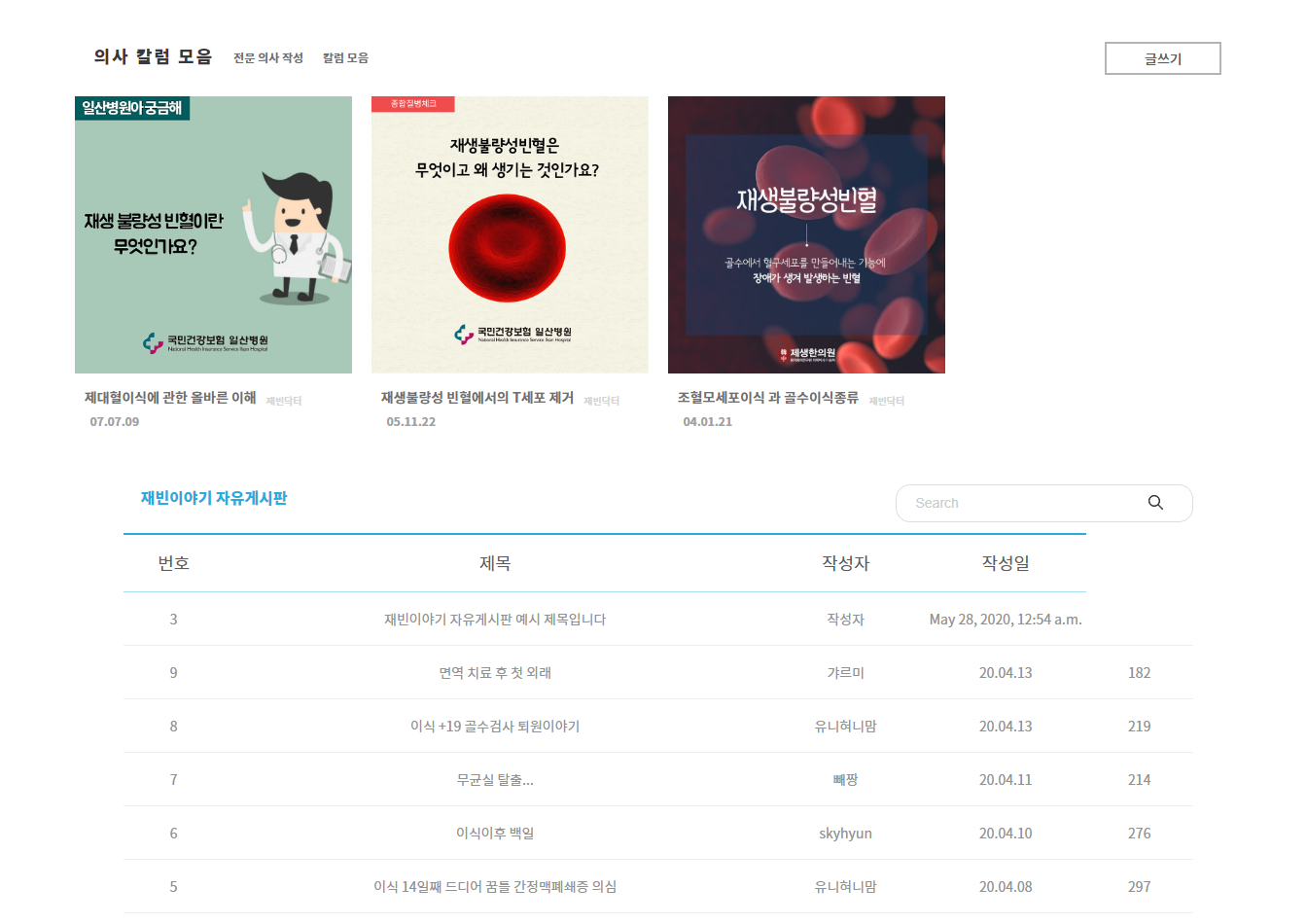


병원의 리스트에서 해당 병원영역을 클릭하면, 해당병원에 대한 자세한 정보 및 사이트 그리고 평점과 댓글을 남길 수 있는 페이지로 이동한다.



Review의 모델을 새로 정의하여 해당 점수를 부여할시 별점으로 나타날 수 있게 하였다. 병원을 다녀간 사람들의 별점과 의견이 댓글로 담겨 병원에 대한 평가로 직접적으로 이어지게 하였다.

2. 재빈이해하기 페이지



재빈이해하기 페이지는 이 웹사이트를 이용하는 사용자들이 자신이 투병하면서 알게 되었던 또는 검색을 통해 알게 되었던 재빈에 대한 정보들을 자유롭게 공유하는 페이지이다.

2-1. 일반인 자유게시판

class BoardForm(forms.ModelForm):

    class Meta:

        model = Board

        fields = ['title', 'content']

        widgets = {

            'title' : forms.TextInput(attrs={'class':'notice-title', 'placeholder':'제목을 입력하세요.'}),

            'content' : forms.TextInput(attrs={'class':'notice-content', 'placeholder':'내용을 입력하세요.'}),

        }

        labels = {

            'title' : '',

            'content' : '',

        }

django에서 제공되는 ModelForm 클래스를 상속받아 모델의 title 및 content에 접근한다. 사용자들에게 보여지는 페이지에서는 attrs메소드에 객체를 주어 input 태그를 갈음한다. as\_table을 통해서 전체 테이블을 한꺼번에 띄울 수도 있다.

def boardform(request, board=None):

    if request.method == 'POST':

        form = BoardForm(request.POST, instance=board)

        if form.is\_valid():

            board = form.save(commit=False)

            board.pub\_date = timezone.now()

            board.save()

            return redirect('notice')

    else:

        form = BoardForm(instance=board)

        return render(request, 'notice/new.html', {'form': form})

def edit(request, pk):

    board = get\_object\_or\_404(Board, pk=pk)

    return boardform(request, board)

def remove(request, pk):

    board = get\_object\_or\_404(Board, pk=pk)

    board.delete()

    return redirect('notice')

def detail(request, board\_id):

    board = get\_object\_or\_404(Board, pk=board\_id)

    if request.method == 'POST':

        form = CommentForm(request.POST)

        if form.is\_valid():

            comment = form.save(commit=False)

            comment.Board\_id = board

            comment.content = form.cleaned\_data['content']

            comment.pub\_date = timezone.now()

            comment.save()

            return redirect('detail', board\_id)

    else:

        form = CommentForm()

        return render(request, 'notice/detail.html', {'board': board, 'form': form})

이후 views.py에서 CRUD기능을 구현한다. save() 및 delete() 메소드를 통해 데이터베이스에 직접 접근하여 데이터를 저장하거나 또는 삭제한다.

2-2. 전문가(전문의료진)용 칼럼



전문가용 게시판은 일반인 게시판과 다르게 사용자가 처음 회원가입을 할 때, 사용자가 pro (전문가인지 여부 Boolean Field)의 값이 True일 경우 detail페이지의 정보가 자동으로 전문가용 칼럼에 업로드된다.

class Photo(models.Model):

    caption = models.CharField(max\_length=80)

    file = models.ImageField(upload\_to="board\_photos")

    def \_\_str\_\_(self):

        return self.caption

사용자들이 작성하는 게시판의 글 중 첫번째로 업로드한 이미지의 정보를 담기 위해 모델에서는 Photo 클래스를 만들어주었다.

settings.py에서는 이미지가 업로드되는 MEDIA URL의 root diretory의 세팅을 설정해주었다.

MEDIA\_ROOT = os.path.join(BASE\_DIR, "uploads")

MEDIA\_URL = "/media/"

기본 BASE\_DIR는 django가 기본적으로 제공하는 해당 프로젝트의 디렉토리를 저장한 메타데이터가 담겨있다. 여기에 uploads를 조인하여 업로드 파일들이 해당 디렉토리에 담기도록 url를 설정해주었다.

from django.utils.html import mark\_safe

@admin.register(Photo)

class PhotoAdmin(admin.ModelAdmin):

    list\_display = ('\_\_str\_\_', 'get\_thumbnail')

    def get\_thumbnail(self, obj):

        return mark\_safe(f'<img src="{obj.file.url}" width="50px"/>')

    get\_thumbnail.short\_description = "Thumbnail"

이후 Admin 페이지에서는 파라미터로 담은 obj에 담긴 메타데이터를 이용하여 업로드된 파일의 url정보를 이용하여 썸네일을 띄울 수 있다. mark\_safe() 메소드는 django의 utils 라이브러리 안에 있는 함수로 사용자가 직접적으로 장고내에서 데이터를 조작할 경우, 발생할 수 있는 보안이슈를 해결하기 위해 장고에게 미리 안전할 사용임을 알려줄 때 사용하는 메소드이다.

추후 InlineAdmin에 admin.TabularInline 또는 StackedInline에 Photo 모델을 지정하고 inlines = (PhotoInline)코드를 추가하여 해당 Photo를 해당 모델에 연결할 수 있다.

3. 투병일기 게시판

스크린샷이(가) 표시된 사진

자동 생성된 설명

투병일기 페이지에서는 재빈에 대한 정보가 아닌 투병하며 느낀 점들을 간략한 일기형태로 글을 적어나갈 수 있는 게시판이다. 다른 사람들의 대략적인 투병시기 및 검사내용과 그에따른 기분들을 파악할 수 있어 커뮤니티 기능의 주요한 역할을 담당하고 있다.

def postform(request, post=None):

    if request.method == 'POST':

        form = PostForm(request.POST, instance=post)

        if form.is\_valid():

            post = form.save(commit=False)

            post.pub\_date = timezone.now()

            post.save()

            return redirect('health\_diary')

    else:

        form = PostForm(instance=post)

        return render(request, 'health\_diary/new.html', {'form': form})

def edit(request, pk):

    post = get\_object\_or\_404(Post, pk=pk)

    return postform(request, post)

def remove(request, pk):

    post = get\_object\_or\_404(Post, pk=pk)

    post.delete()

    return redirect('health\_diary')

def detail(request, post\_id):

        post\_detail = get\_object\_or\_404(Post, pk=post\_id)

        return render(request, 'health\_diary/detail.html', {'post': post\_detail})

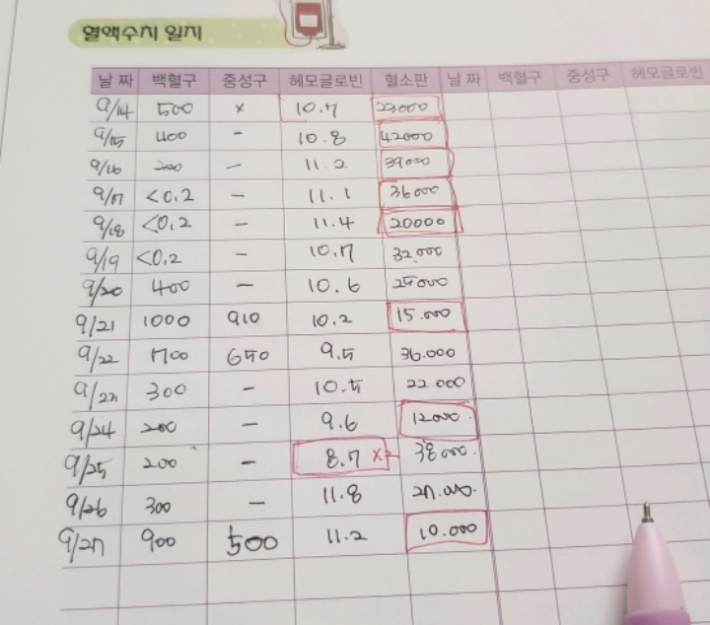
투병일기 페이지는 위의 재빈이해하기의 일반인 게시판의 CRUD기능과 유사하게 구현되었다.

4. 마이페이지

스크린샷이(가) 표시된 사진

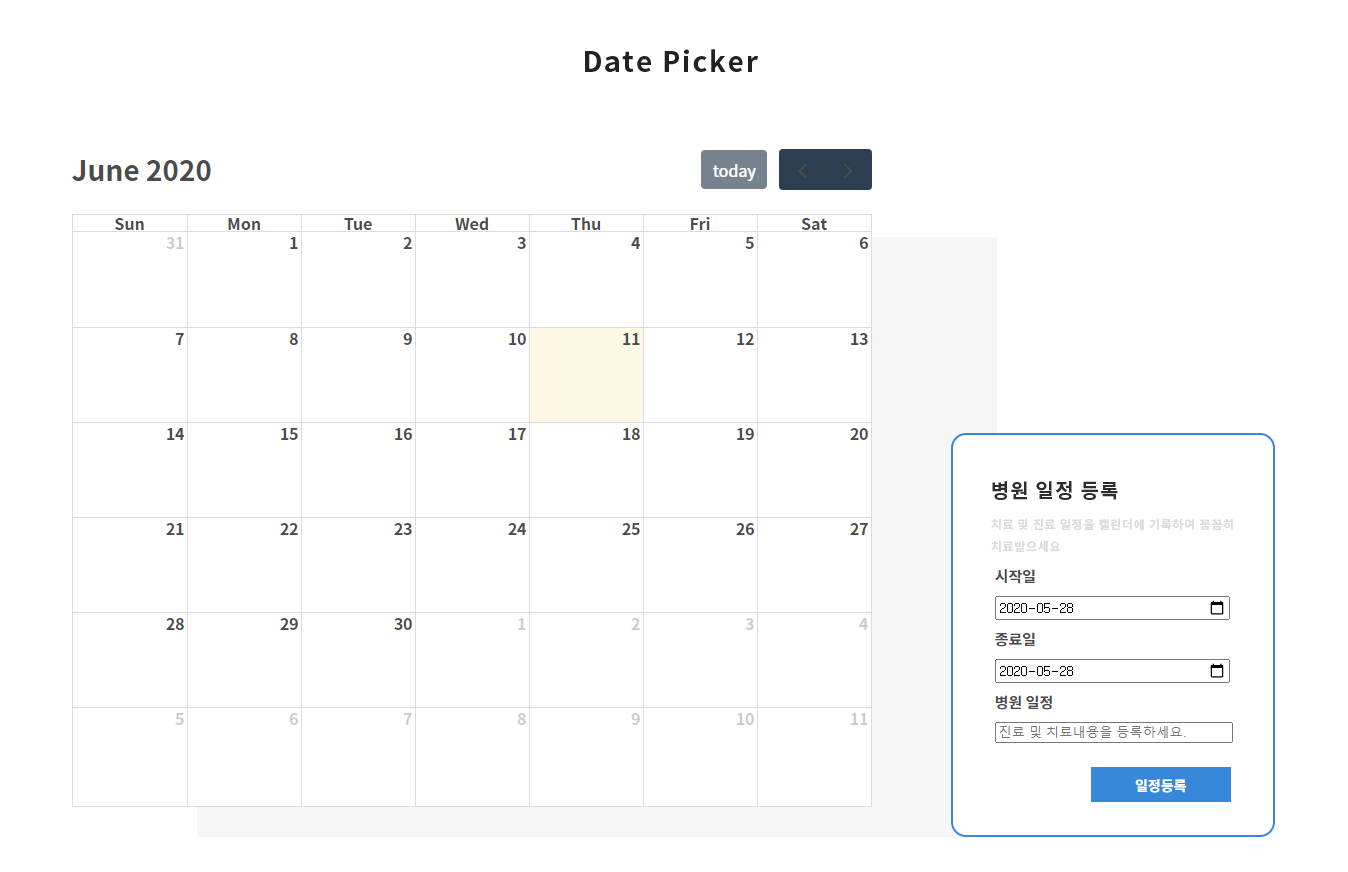
자동 생성된 설명

마이페이지에서는 사용자가 로그인 되었을 때 들어올 수 있는 페이지이다. 병원의 일정을 저장하여 한눈에 볼 수 있거나 또는

****

다음과 같이 수기로 작성하는 자신의 혈액일지를 웹상에서 작성하고 그날의 기분을 이모티콘으로 표현할 수 있다. 반응형웹으로 구현하면 모바일화면에서도 작성할 수 있기 때문에 편리하게 이용할 수 있다.

4-1. 병원일정 캘린더



Date Picker에서는 시작일과, 종료일 그리고 병원일정을 작성하고 일정등록을 click하게 되면 위의 val()메소드를 이용하여 데이터를 받아온 것처럼 데이터를 js의 객체로 넘겨주게 된다.

var calendar = new FullCalendar.Calendar(calendarEl, {

            plugins: [ 'interaction', 'dayGrid' ],

            defaultDate: `${year}-${month}-${date}`,

            editable: true,

            eventLimit: true, // allow "more" link when too many events

            events: [

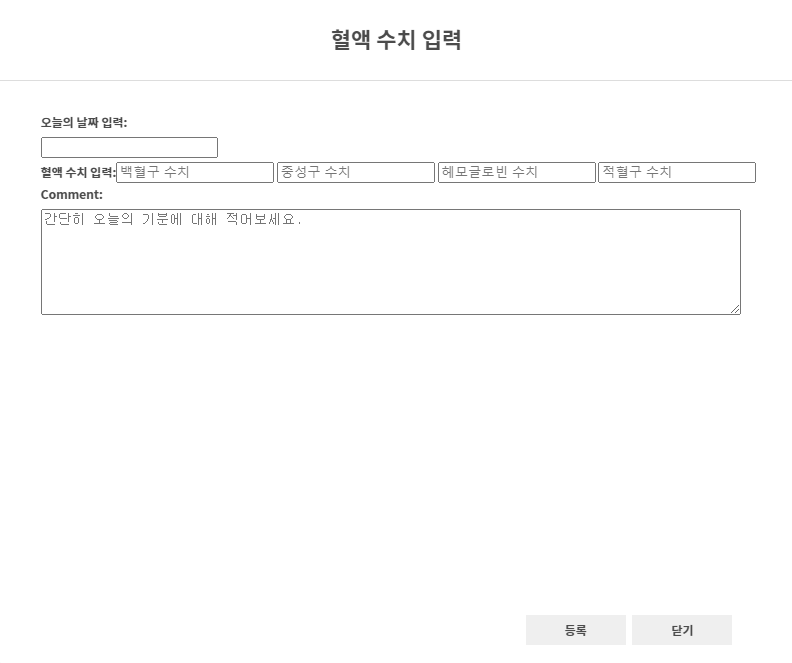
            ]

다음과 같은 캘린더 객체에 events에 받아온 input 데이터를 객체로 받아서 넣게 된다. 그럼 해당 날짜에 해당 일정이 등록되게 되며 여러 개의 일정이 등록된 경우 layout의 깨짐을 방지하기 위해 +more가 나와 작은 창에서 볼 수 있도록 한다.

4-2. 혈액수치 일지



[혈액수치 일지]



[글쓰기 버튼을 눌렀을 때 나오는 모달창]

def index(request):

    op\_list = Diary.objects.order\_by('id')

    form = DiaryForm()

    context = {'op\_list':op\_list, 'form':form,}

    return render(request, 'mypage/index.html', context)

@require\_POST

def addResult(request):

    form = DiaryForm(request.POST)

    if form.is\_valid():

        post = Diary(date = request.POST['date'],WBC = request.POST['wbc'],neutrophil = request.POST['neutrophil'],RBC = request.POST['rbc'],PT = request.POST['pt'],content = request.POST['comment'])

        post.save()

    return redirect('mypage')

form 데이터 처리를 POST방식을 이용하여 데이터 처리해준 것은 위의 게시판의 형태와 동일하다.

EMOTION = [

        (1, '😍행복해요'),

        (2, '🙃보통이에요'),

        (3, '😑별로에요'),

        (4, '😭슬퍼요'),

        (5, '😡화나요'),

    ]

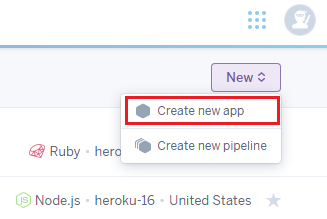
추가적인 이모티콘을 추가하여 추후에 매일 수치의 변화로 인해 변화되는 기분을 직관적으로 알아볼 수 있도록 기능을 구현해나갈 예정이다.

5. Heroku app 배포

헤로쿠는 간단하게 무료로 호스팅 할 수 있도록 해주는 서비스이다. 무료인 이유는 해당 도메인으로 몇시간 동안 요청이 없을 경우, 수면상태로 전환하므로 초기 접속이 늦어지기 때문이다. 때문에 개인 블로그 서버라든지, 포트폴리오의 결과물을 배포하는 용도로 가볍게 활용하기에 좋다. 물론 AWS EC2처럼 현금을 지불하게 되면 원활한 서비스를 이용할 수 있다.



헤로쿠 계정을 만들고 로그인 후 메인페이지 우상단의 메뉴바 – dashboard를 클릭한다.

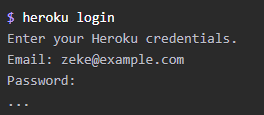


Dashboard 페이지에서 우상단의 create new app을 클릭한다.



App name 항목에 URL에 표기할 이름을 영어로 작성한다. 예를 들어, App name으로 live in이라고 작성했다면, 배포할 사이트의 주소는 다음과 같다.

<https://livein.herokuapp.com/> 이다.



Heroku cli를 통해 heroku login 명령어를 입력하고 헤로쿠 계정의 이메일과 비밀번호를 입력한다. git에서 현재상태까지의 커밋을 완료한다.

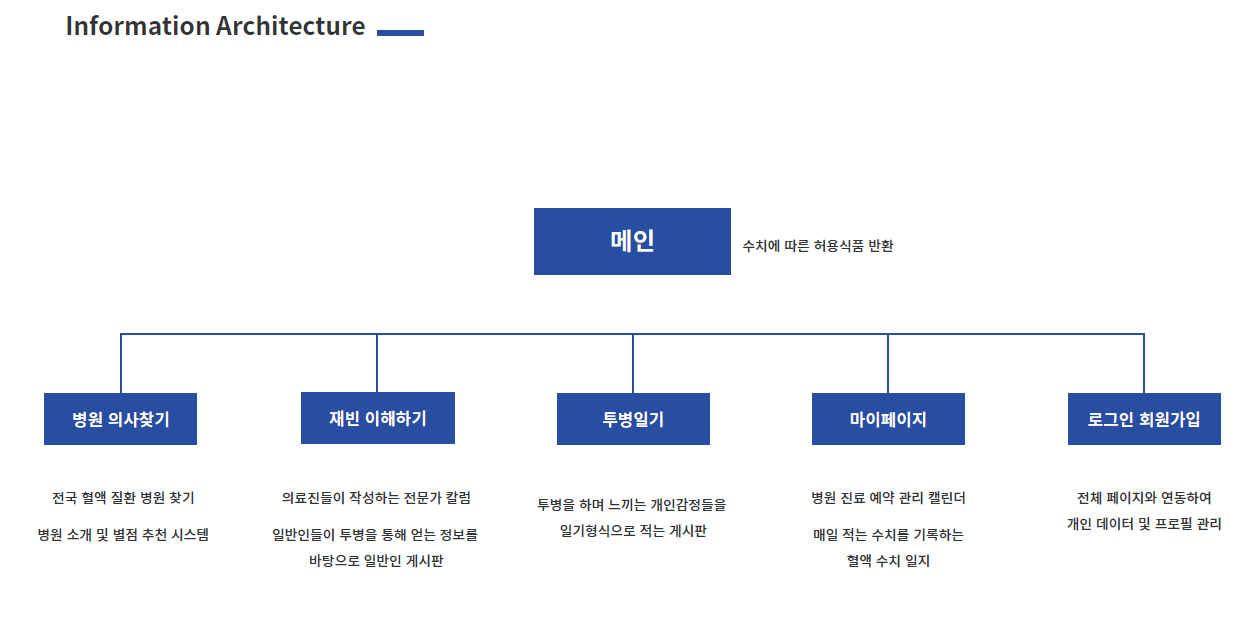


커밋한 app의 git을 remote로 등록한다. 해당 app name은 livein이기 때문에 Heroku git:remote -a livein라고 작성했다. 다음으로 만든 remote repository로 프로젝트 소스코드를 push한다. 마지막으로 Heroku ps:scale web=1 명령어를 입력하고 배포가 제대로 되었는지 url을 통해 확인해본다.

해당 프로젝트는 <https://livein.herokuapp.com/> 사이트에서 확인해 볼 수 있다.

# 6. 기타

1. Information Architecture



Live in 프로젝트는 메인페이지 1개와 로그인∙회원가입 페이지 그리고 4개의 서브페이지 그리고 하위의 디테일 페이지들로 구성되어 있다.



해당 프로젝트는 암닥터 페이지를 레퍼런스로 하여 희귀난치병 환우들의 커뮤니티 형성을 통한 정보 교환 기능과 혈액 수치를 자가 측정 해볼 수 있는 자가진단 기능을 하나의 서비스에서 시행해볼 수 있다는 특징을 갖는다.

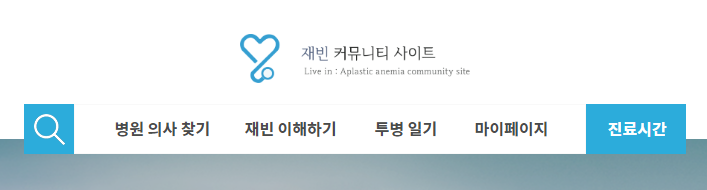
그러므로, 주축이 되는 기능인 수치 측정 기능을 메인페이지에 두었으며, 추가적으로 자가측정 기능을 서포트해줄 수 있는 병원 및 병동 찾기 기능을 병원∙의사 찾기 서브페이지로 구성하였으며, 커뮤니티 기능을 보여주는 재빈 이해하기와 투병일기 페이지로 하위의 서브페이지로 구성하였다. 마이페이지는 추가적인 기능으로 병원 일정 및 혈액수치 일지 작성 등의 기능을 할 수 있게 구성하였다.

2. Logo variation

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

혈액 질환이라는 것에 포커스를 맞춘 모양과 가로로 긴 사이즈의 로고 제작을 기획 방향으로 삼았다.



심장의 치료라는 느낌이 들 수 있도록 하트에 청진기를 합친 형태의 로고로 채택하였다.