프로젝트 최종 보고서

갓성비 팀

김영환 (201511795)
남승철 (201511809)
문정민 (201511812)
박은혜 (201713834)
심세은 (201713886)

목차

프로젝트 개요

✔ 프로젝트 선정 이유

💡 지인의 회사 홍보 사이트의 수정

-지인의 기존에 사용중인 회사 사이트가 제작이 된지 오래 되어 최신 트렌드에 맞게 새롭게 단장을 하고자 하였고. 추가로 기존의 기능에서 몇 가지 추가하고자 하는 기능들을 추가하고 이를 구현하는 것을 목표로 세웠습니다.

✔ 고객의 요구 사항

1) 제작된 지 오래된 홈페이지를 트렌드에 맞게 개선
2) 메인 페이지에서 가장 먼저 회사 전경. 소개를 보여주기
3) 회사 소개의 시각적 강조
4) 기성품 판매 기능
5) 주문 및 문의 시. 메일 알림 기능
6) 관리자 권한으로 사이트 내부정보 수정가능
7) 관리자가 다용도로 사용 가능한 게시판 기능 추가

✔ 기초자료 수집

- 고객이 요구한 사항중 제품의 디자인적인 면의 개선이 있었기 때문에 그러한 면에서의 개선점을 찾았고. 추가로 구현 가능한 기능이 무엇이 있을까 기존의 사이트 분석으로 시작.
- Main Page
    - 전부 이미지 UI로 구성됨 (사이트 최초 접속시 받는 데이터가 크다
    - 사이트에서 다음 사이트로의 이동시간이 길다
    - 개발자 없이 사이트에 대한 수정이 불가능 하다
    - 사용하지 않는 메뉴 버튼의 존재
    - 모바일 기능 미지원

- Product Page
    - 세분화 되지 않은 구조
    - 원본 사진 미리보기 (받아오는 데이터가 너무 크다)
    - 개발자 없이 사이트에 대한 수정이 불가능 하다
    - 상세보기 페이지 미지원
    - 모바일 기능 미지원

✔ 요구분석

1. 홈페이지들 트렌드에 맞게 개선
2. 기성품 판매
3. 관리자 권한으로 사이트 사진 및 글 작성
4. 주문 및 문의 시, 메일 알림 기능
5. 메인 페이지에서 가장 먼저 회사 전경, 소개글 보여주기
6. 회사소개의 시각적 강조
7. 다용도 게시판 기능 생성

진행과정

✔ 개발환경 설정

웹 PM 주도하에 GoormIDE 서비스를이용하여 초기상태를 구축시작.

- **사용 언어** : Python
    - 이유 : 팀원모두가 사용가능

- **사용 도구** : Django(Back), Bootstrap(Front)
    - 이유 : 단기간에 빠르고 효율적으로 웹프로젝트가 가능함

- **개발 환경** : GoormIDE
    - 이유 : 아마존 Cloud9과 같은 개발환경, 한국어 포멧지원, 리눅스환경, 무료로 상시 이용가능

- 팀원들 개인이 조사한 template중 고객의 요구사항에 가장 최적한 template를 선정 (상업적이용 가능, Bootstrap 이용)

이를 적용하여 분업환경을 구축

✔ 공학기법 선정

-웹 서비스 개발, 애자일 개발

웹 서비스 개발은 V모델과 같이 검증 절차와 문서화에 비용이 많이 드는 개발론을 사용하기가 어렵다고 판단하였고, 추가로 시간이 지남에 따라 변화되는 고객의 요구사항을 받아들이기엔 한계가 있다고 판단 하였습니다. 이러한 이유로 애자일 개발론과 재사용 컴퍼넌트 방법을 채택하였고, 계획기반 방법과 접증적 방법을 적절하게 사용하기 위해 스크럽 기법을 채택하였습니다. 고객의 요구사항을 달성시마다 고객의 평가를 받기위해 프로토타이핑 기법 채택하였습니다.

-스크럽 주기

- 전체 회의 매주 : 1회
    - 각 팀별 개발 진척상황 확인
        - 소스 병합
        - 요구조사 재반영
- 팀별 회의
    - 팀장 지시 아래 필요시 마다 온/오프라인 회의

-테스트주도 개발 (TDD)

Python 에서 지원되는 단위 테스트 모듈 사용

- Unittest 모듈을 사용
- Django에서 지원되는 Test.py 도구 사용
- Models.py 의 database Test
- Views.py 에 있는 function test

✔ 기초 문서작업

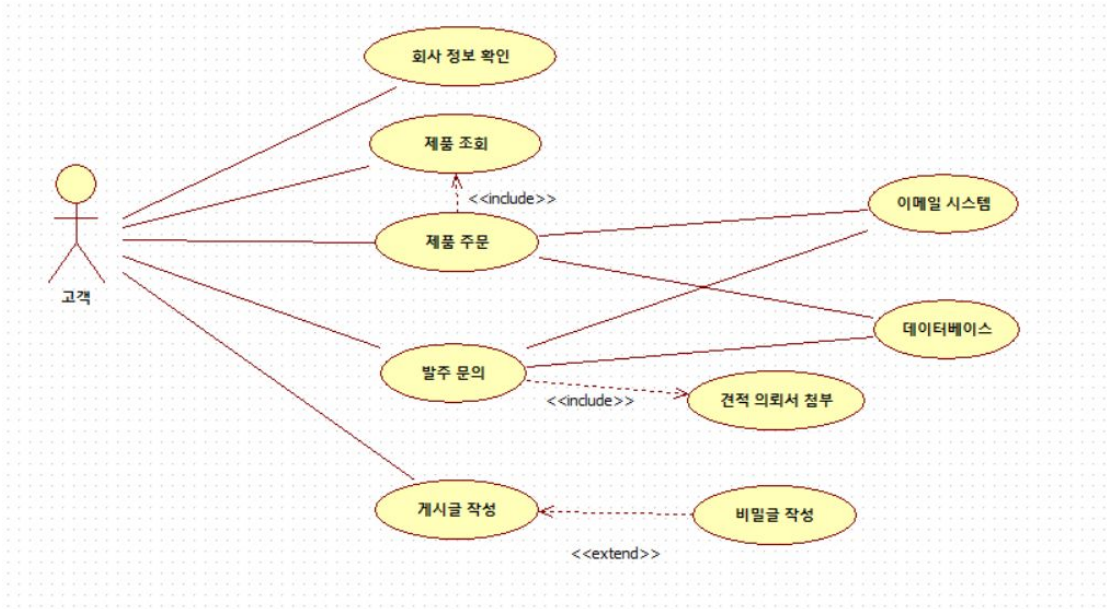
-온/오프라인 활동에서 도출된 결과 들

- **HangOut 서비스** 사용

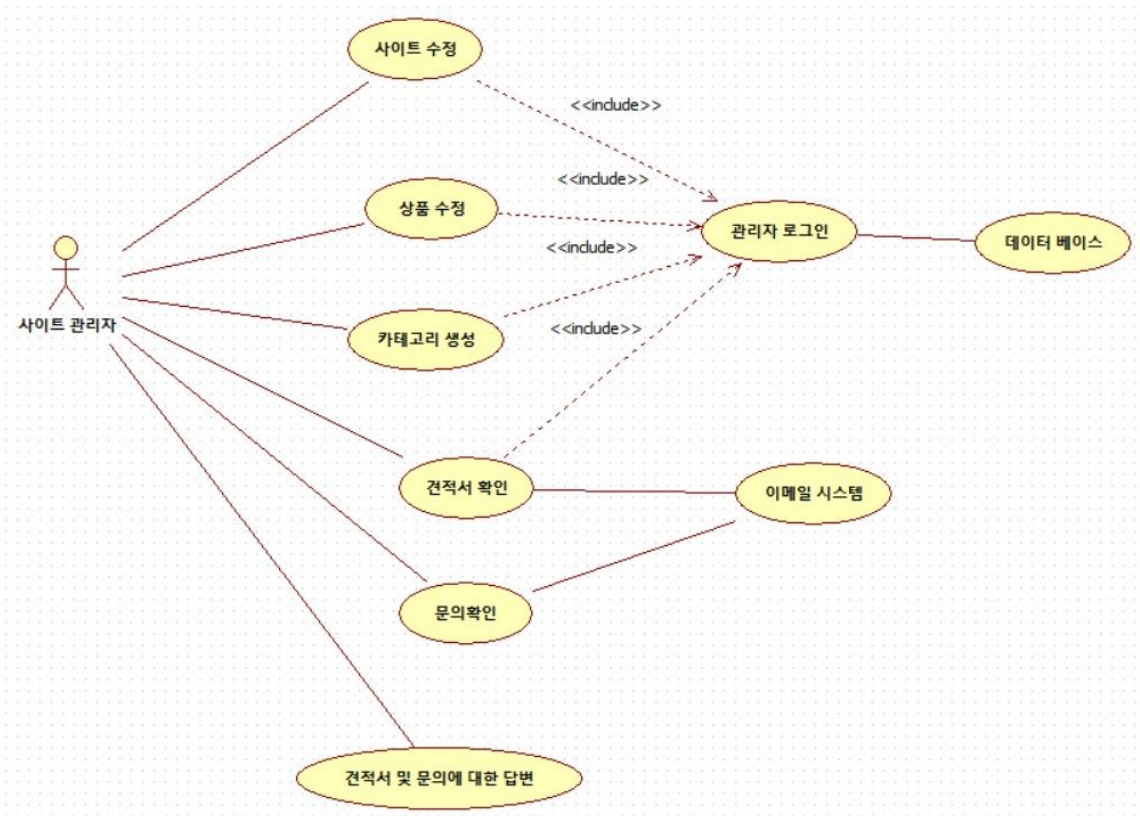
최대한 팀원들의 프로젝트 참여의 효율성을 증대시키기 위해서 오프라인 회의만 고수하지 않았습니다. 오프라인 회의를 최대한 활용하여 회의 결과나 각자에게 부여한 조사항목들을 Cloud 문서에 기록하였습니다.추가적으로 DB설계와 같이 문서작업이 필요하겠다고 판단된 부분에 대해서는 UML등을 활용해 문서화 작업을 실시하였습니다.

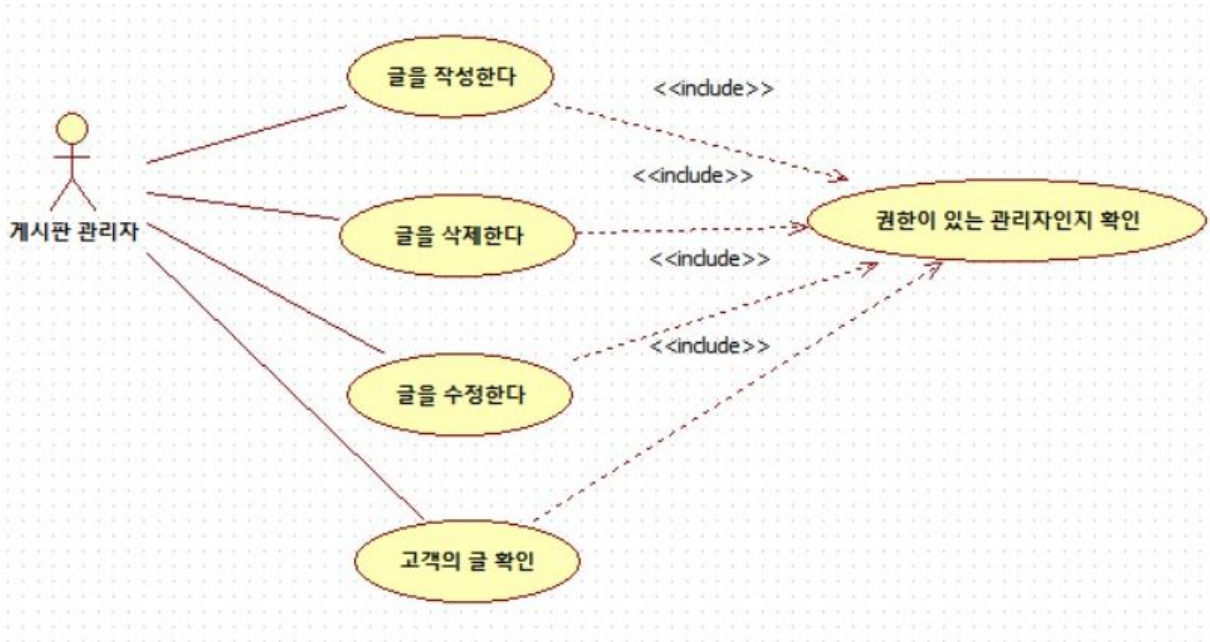
-공동문서 작업

- **Google Doc 서비스** 사용

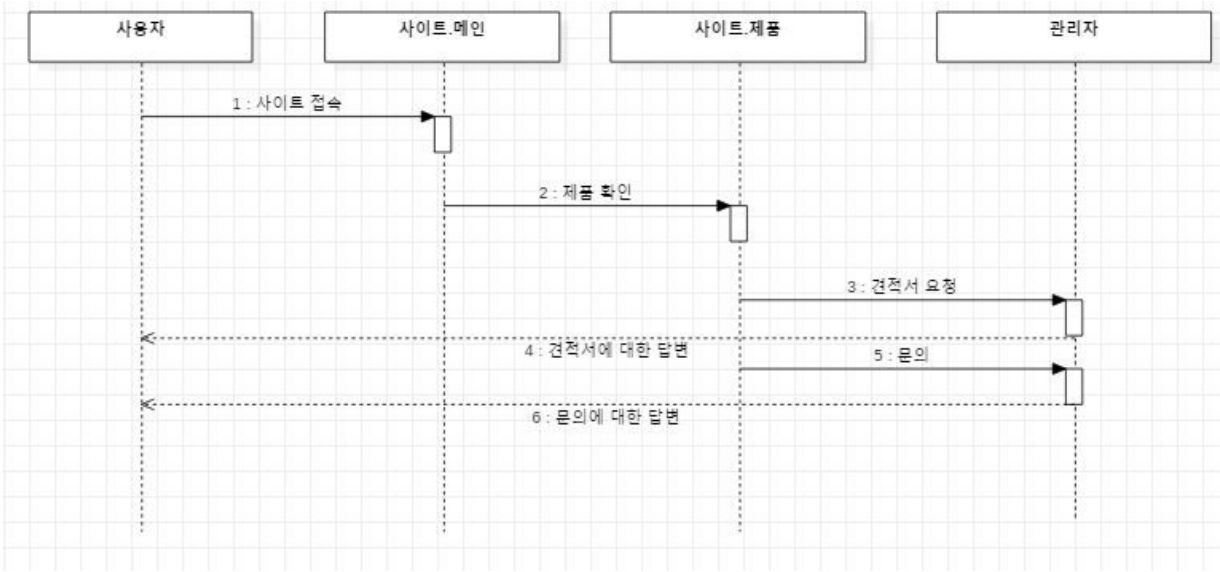
UseCase Diagram -User


UseCase Diagram -Admin

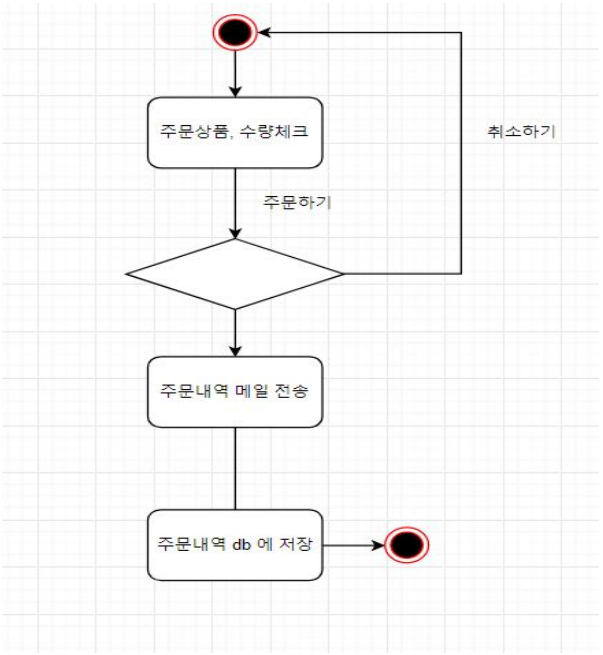


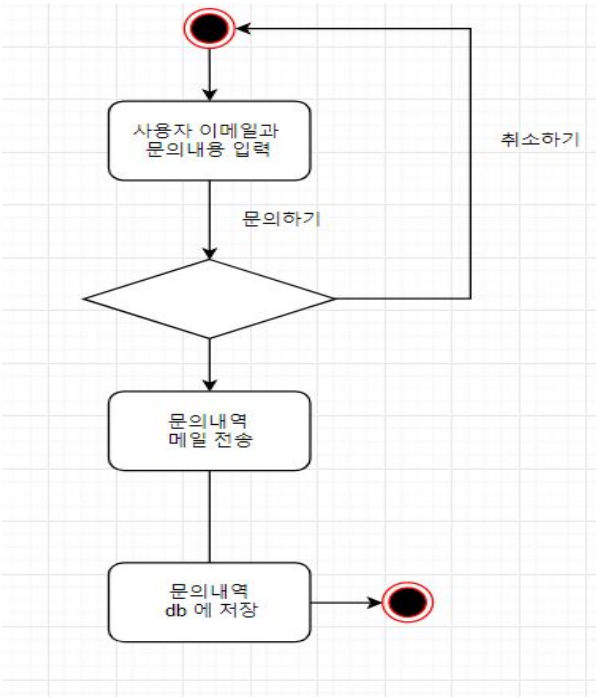


Sequence Diagram



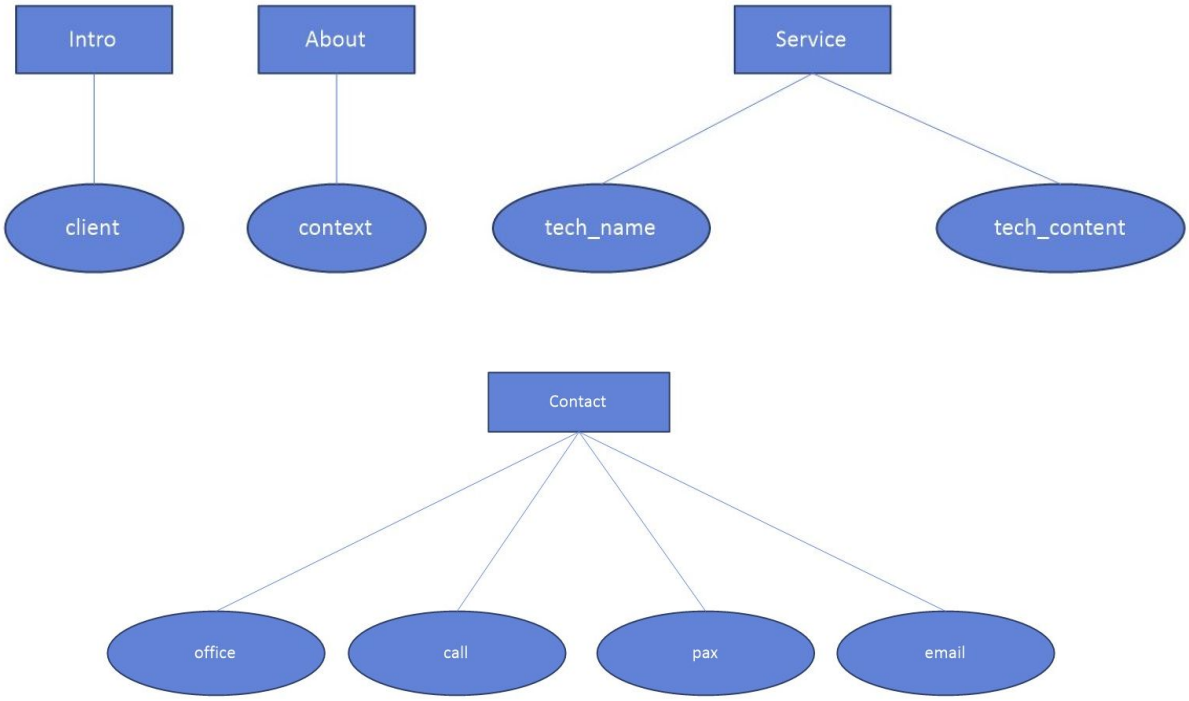
Activity Diagram



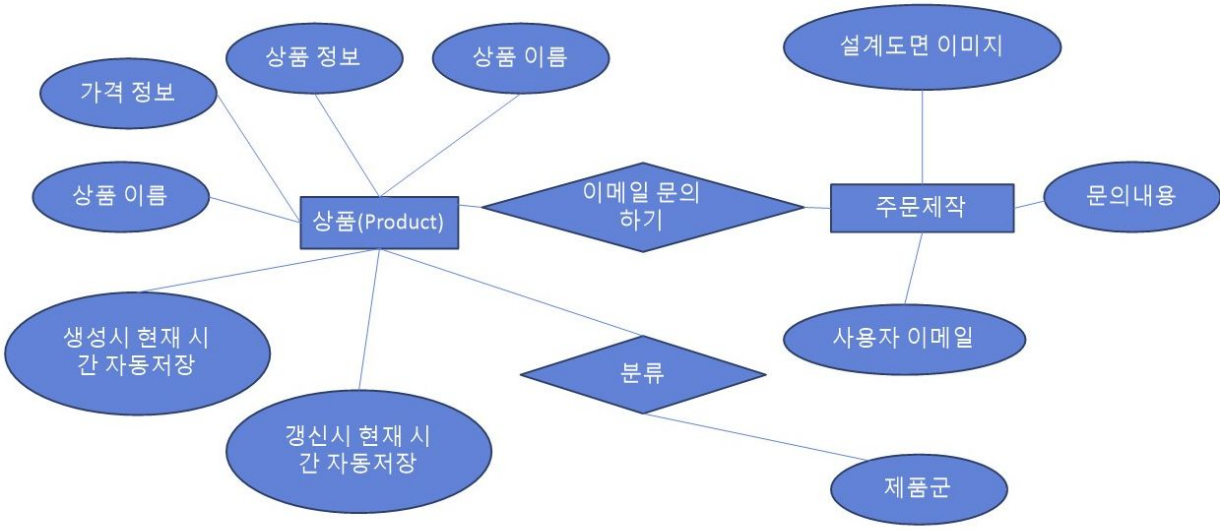


Class Diagram

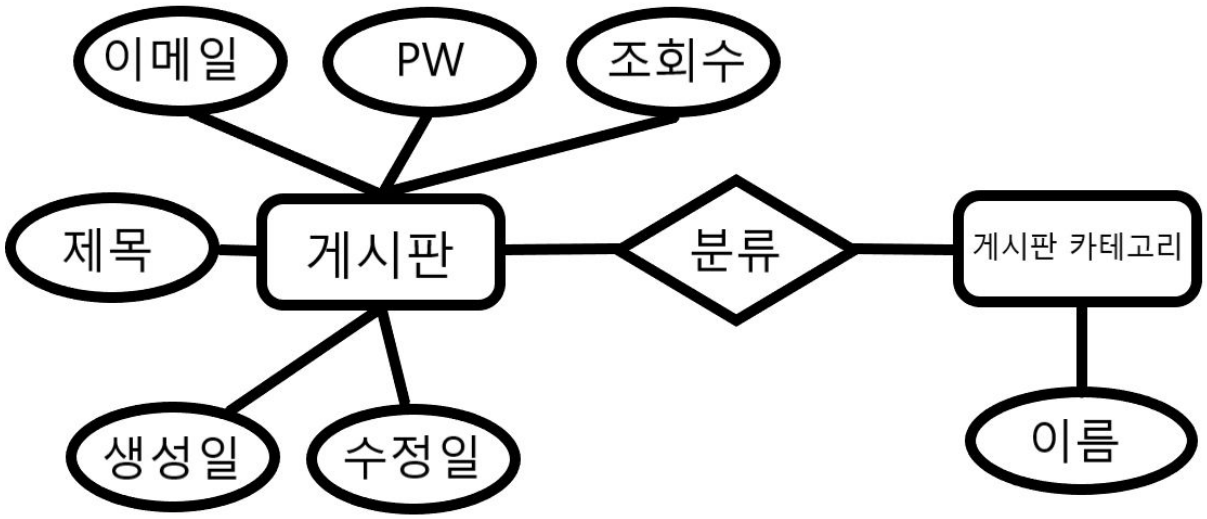
<메인 사이트 DB>



<Product DB>



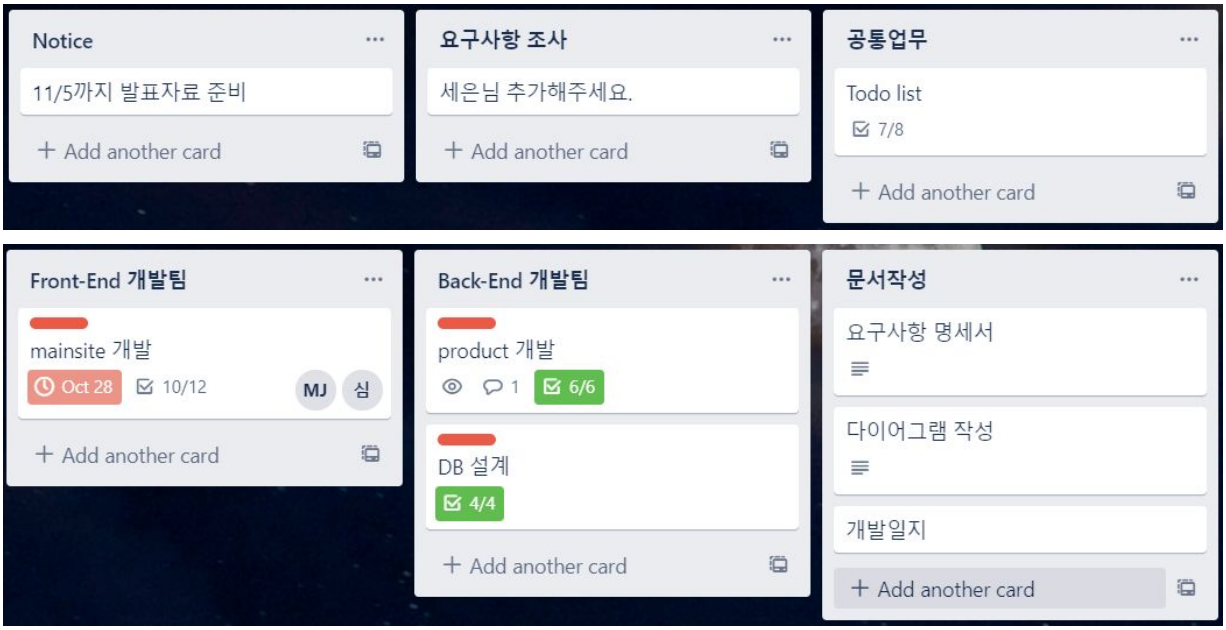
<Board DB>



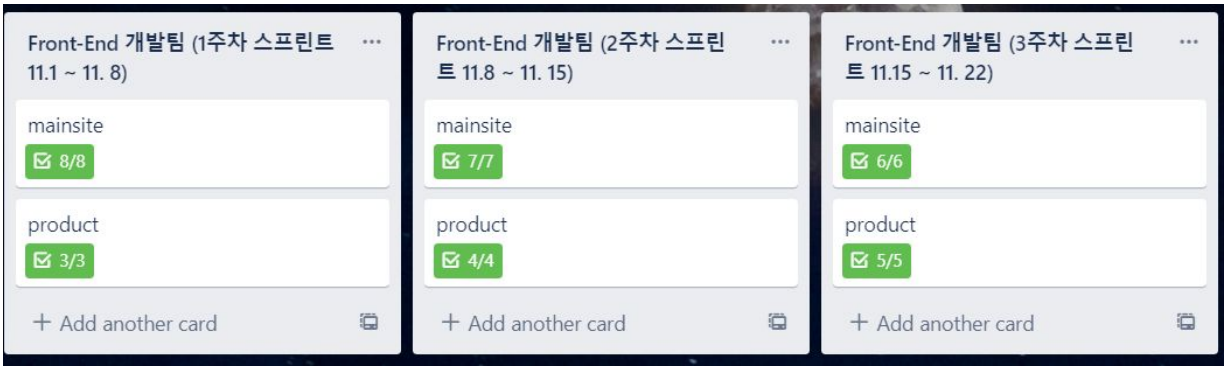
-협업지원 도구 사용

- **Trello 사용**
  - **제품 백로그 작성**
  - **스프린트 백로그 작성**
  - **팀별 활동로그 기록**
  - **팀별 일정관리**

필요시마다 사용되는 공용 카드공간



Front-End 개발팀 스프린트 백로그



1week	
Mainsite 관련	**Product 관련**
	

2week

Mainsite 관련

카테고리(이하 우측 상단바) - 문정민 Hide completed items Delete

100%

- [x] ~~메인에서 잘 보이게 폰트 색상 수정~~
- [x] ~~회사에 맞게 카테고리명 변경~~

Add an item

intro 부분 수정 - 심세은 Hide completed items Delete

100%

- [x] ~~텍스트 입력효과 추가~~
- [x] ~~FIND OUT MORE 버튼의 Grey 계열 색상으로 변경~~
- [x] ~~FIND OUT MORE 버튼의 클릭시 색상 변경 및 테두리 변화 변경~~

Add an item

보유기계 이미지 추가 - 문정민 Hide completed items Delete

100%

- [x] ~~보유 기계를 Admin페이지에서 추가시에 특정 이미지와 함께 기계에 대한 설명 추가 기능~~
- [x] ~~Grey 계열 색상으로 폰트 색상 변경~~

Add an item

Product 관련

사이드바 (좌측) - 심세은 Hide completed items Delete

100%

- [x] ~~카테고리 부분 박스 색상 Grey 계열 색상으로 변경~~
- [x] ~~폰트 색상 White 로 동일화~~

Add an item

메인화면 수정 - 문정민 Hide completed items Delete

100%

- [x] ~~카드 색상은 White로 나머지 부분은 Grey 계열 색상으로 변경~~
- [x] ~~카드의 경우 길게 터치 혹은 클릭시 제품의 원본 사진을 불러오도록 기능~~

Add an item

3week

Mainsite 관련

CEO 인사말 - 문정민 Hide completed items Delete

100%

- ~~인사말 배경 Grey 계열 색상으로 변경~~
- ~~Grey 계열 색상에서 글이 잘 보이게 검은색으로 폰트 색 수정~~
- ~~Get STARTED! 버튼 및 줄바꿈 색상 White로 변경 및 버튼 중앙으로 이동~~
- ~~Get STARTED! 버튼의 클릭시 색상 변경 및 테두리 변화 변경~~

Add an item

페이지네이션 - 심세은 Hide completed items Delete

100%

- ~~메인사이트 페이지네이션 디자인~~

Add an item

모바일 버그 체크 Hide completed items Delete

100%

- ~~버그 수정 완료~~

Add an item

Product 관련

사이드바 모바일 뷰 수정 - 문정민 Hide completed items Delete

100%

- ~~네브바 접었을 때 튀어나오는 글자 크기 수정~~
- ~~flex 형태를 grid로 수정~~
- ~~주문 버튼 가격 표시 크기 위치 수정~~

Add an item

페이지네이션 추가 - 심세은 Hide completed items Delete

100%

- ~~제품 사이트 페이지네이션 디자인~~

Add an item

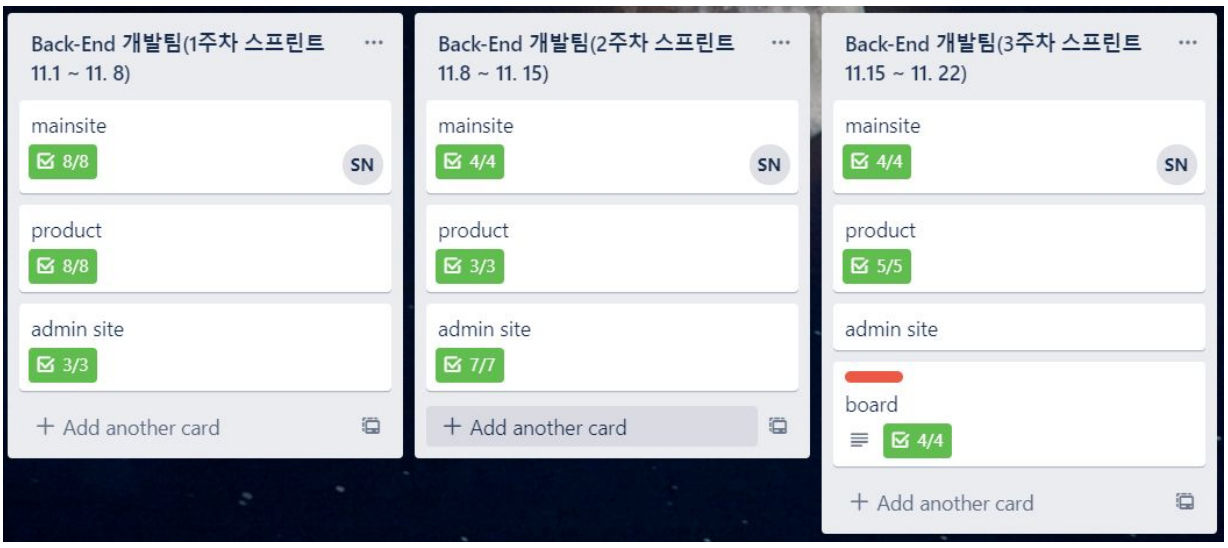
모바일 버그 체크 Hide completed items Delete

100%

- ~~버그 수정 완료~~

Add an item

Back-End 개발팀 스프린트 백로그



1week

Mainsite 관련

ceo소개란 - 박은혜 - 2019.11.1까지 Hide completed items Delete

100%

- ~~about db 생성~~
- ~~ceo소개 column생성~~
- ~~intro db 생성~~
- ~~db에서 내용을 불러와 프론트엔드로 넘겨주기~~

Add an item

회사의 주소서비스 소개 - 남승철 - 2019.11.1까지 Hide completed items Delete

100%

- ~~service db 생성~~
- ~~contact db 생성~~
- ~~유에서 내용을 불러와 프론트엔드로 넘겨주기~~

Add an item

단위 테스팅 -남승철, 박은혜 Hide completed items Delete

100%

- ~~테스팅 통과~~

Admin 관련

사용자 주문제작 발주기능 메일알림 - 남승철- 2019.11.5까지 Hide completed items Delete

100%

- ~~동일한 메일 전송 함수 사용~~

Add an item

단위 테스팅 Hide completed items Delete

100%

- ~~테스팅 통과~~

Add an item

Product 관련

navbar 제품 종류별 분류 - 박은혜 - 2019.11.6까지 Hide completed items Delete

100%

- ~~product_category db생성~~

Add an item

관련상품 진열 -김영환 - 2019.11.5 까지 Hide completed items Delete

100%

- ~~product db 생성~~
- ~~media 폴더에 이미지파일 모아서 관리~~
- ~~이미지처리 pillow 라이브러리 추가~~

Add an item

기성상품별 발주기능 메일 알림 -남승철 -2019.11.5까지 Hide completed items Delete

100%

- ~~order db 생성~~
- ~~메일전송 api 추가~~

Add an item

사용자 주문제작 발주기능 메일알림 - 남승철- 2019.11.5까지 Hide completed items Delete

100%

- ~~동일한 메일 전송 함수 사용~~

Add an item

단위 테스팅 Hide completed items Delete

100%

- ~~테스팅 통과~~

Add an item

2week

Mainsite 관련

db 수정 - 남승철 - | Hide completed items | Delete

100%

- [x] ~~sendEmail 함수와 order 함수를 통일~~
- [x] ~~pw 칼럼 추가~~

Add an item

버그 체크 - 남승철, 박은혜 | Hide completed items | Delete

100%

- [x] ~~버그 수정 완료~~

Add an item

exception handler 작성 | Hide completed items | Delete

100%

- [x] ~~에러 메시지 정의 및 출력하는 부분 작성~~

Add an item

Admin 관련

admin site | Hide completed items | Delete

100%

- [x] ~~admin template - bootstrap 사용~~
- [x] ~~update.html~~
- [x] ~~create.html~~
- [x] ~~index.html~~
- [x] ~~admin site 디비 접근 권한~~
- [x] ~~admin 최근 기록~~
- [x] ~~admin 기능개선~~

Add an item

Product 관련

파일 업로드 부분 수정 - 박은혜 | Hide completed items | Delete

100%

- [x] ~~업로드 형식이 pdf 확장자만 가능하게~~

Add an item

파일업로드 부분 예외 처리 - 남승철 | Hide completed items | Delete

100%

- [x] ~~업로드를 안했을 때 오류 처리~~

Add an item

단위 테스팅 | Hide completed items | Delete

100%

- [x] ~~테스팅 통과~~

Add an item

3week

Mainsite 관련

썸네일 관리 - 남승철 Hide completed items Delete

100%

- [x] ~~썸네일 크기 문제로 인해 썸네일을 지정해 주는 api install 및 적용~~

Add an item

인수테스팅 Hide completed items Delete

100%

- [x] ~~테스팅 통과~~

Add an item

알파 테스팅 Hide completed items Delete

100%

- [x] ~~테스팅 통과~~

Add an item

단위 테스팅 Hide completed items Delete

100%

- [x] ~~테스팅 통과~~

Add an item

Product 관련

메일 전송기능 수정 - 박은혜 Hide completed items Delete

100%

- [x] ~~메일 보낼 때 첨부파일이 없을 때 첨부 오류 수정~~
- [x] ~~메일로 받아 봤을 때 보이는 형식 수정~~

Add an item

알파 테스팅 Hide completed items Delete

100%

- [x] ~~테스팅 통과~~

Add an item

인수 테스팅 Hide completed items Delete

100%

- [x] ~~테스트 통과~~

Add an item

단위 테스팅 Hide completed items Delete

100%

- [x] ~~테스팅 통과~~

Add an item

Admin 관련

admin custom Hide completed items Delete

100%

- [x] ~~admin template 리팩토링~~
- [x] ~~admin navbar 수정~~
- [x] ~~admin logo 수정~~
- [x] ~~admin 모바일뷰 개선~~
- [x] ~~admin admin.py 기능개선~~
- [x] ~~admin page DB접근관련 기능 개선~~

Add an item

Board 관련

LABELS

긴급 +

Description Edit

게시판 요구사항 추가
사람들이 문의에 대해 이메일로만 처리를 하면 번거로울 것 같음
제품 백로그:
고객이 언제든지 홈페이지를 통해 제품을 보고 문으를 남길 수 있는 게시판이 있으면 좋겠습니다. 글쓰기 버튼 클릭 하나로 글을 작성 할 수 있었으면 좋겠습니다. 그리고 수정 삭제 할 수 있는 권한이 있고 비밀번호를 통해 그 작업이 가능하게 했으면 좋겠습니다. 게시글을 읽을 때는 누구나 읽을 수 있게 개방되어도 상관 없습니다.

게시판 만들기 - 김영환 Hide completed items Delete

100%

- [x] ~~게시판 메인 페이지 제작~~
- [x] ~~게시판 수정 제작~~
- [x] ~~게시판 삭제 제작~~
- [x] ~~게시판 등록 제작~~

Add an item

✔ 현장조사

고객의 요구사항		- 제작된 지 오래된 홈페이지를 트렌드에 맞게 개선 - 메인 페이지에서 가장 먼저 회사 전경, 소개를 보여주기 - 회사 소개의 시각적 강조 - CEO 인사말, 주요 거래처 리스트 추가 - 기성품 판매 기능 - 핵심 기계 사진 추가하여 대표 기술 소개 - 주문 및 문의 시, 메일 알림 기능 - 관리자 권한으로 사이트 사진, CEO 인사말, 소개글 수정
기존 사이트의 기능분석	Front	- Vanilla html, JS, css - 대부분의 로고를 img로 사용
	Back	- 서버사이드 (Php언어), - 제품소개 기능 (단조로움) - 게시판 기능 (활용성이 낮음) - 채용게시판 기능 (활용성이 낮음) - 페이지 관리자 기능 (없음)
기존 사이트의 개선이 필요한 부분	기능적	- UI 디자인 개선 - 온라인 견적 문의 기능 개선 - 제품 문의 기능 추가 - 모바일 뷰, 페이지 관리자 뷰 기능 추가 - 제품소개란에 자세한 설명 추가
	비기능적	- 관리자 모드가 복잡하지 않아 직원이 모든 기능 쉽게 사용 가능 - 주문 및 문의 시 발송되는 메일에 대한 오류 최소화 (미발송, 내용 누락 등)

✔ 기초자료 수집

Project 형태	Web 기반 프로젝트
사용가능한 SE모델	MVC. MTV. 계층구조. Mediator. Client-Server. Facade

사용가능한 디자인 패턴	MVC. MTV. 계층구조. Mediator. Client-Server. Facade
채택한 패턴. 모델의 특성	MTV 패턴. ● MVC와 비슷한 패턴으로 웹에서 많이 사용. ● 계층구조로 views.py 와 template. models.py를 독립적인 컴포넌트로 나눔으로써 디자이너. 개발자. DB관리자의 분업이 가능해진다. ● MVC와 차이점은 거의 없다.

채택SE Model	환경 및 통합 (재 사용 가능한 컴포넌트)
모델의 특성	● 단기간에 높은 수준의 소프트웨어를 구현이 가능하다.

사용할 개발언어	Python
언어 특징	● 동적 언어 ○ Code inspector 사용 불가 ● 정적 테스팅 ○ Unit 테스트

사용할 프레임워크	Django
재 사용 가능한 컴포넌트	Bootstrap. Materials.
사용할 디자인 패턴	MTV 패턴

✔ 스토리카드

일단 전반적으로 현재의 사이트보다 디자인적으로 개선이 되었으면 좋겠습니다.

메인 페이지에 들어왔을 때 가장먼저 회사의 전경과 회사 소개가 나왔으면 좋겠습니다. 그리고 회사 소개부분이 시각적으로 강조가 되었으면 좋을것 같습니다.

그 다음 다른 타사 사이트와 같이 메인페이지에 CEO 인사말이 들어가 있으면 좋겠습니다. 추가로 협력사에대한 리스트가 화면에 나왔으면 좋겠어요.

그리고 중간에는 회사의 기술을 대표할 수 있는 공간을 마련해주었으면 합니다. 가령 핵심 기계와 같은 사진을 넣어주셨으면 합니다.

추가로 기존에는 기성품 판매없이 사용자가 직접 제품을 설계한거에만 주문을 받았는데. 기성품에 대한 주문이 가능한 형식이 있었으면 합니다. 뿐만 아니라 주문시 DB에 저장은 물론. 그때그때 메일로도 받아보고 싶습니다.

기존의 페이지에서는 사이트에 올라가 있는 사진들을 수정한다거나 하기가 힘들었는데. 수정이 가능했으면 합니다.

또 메인사이트 내부의 CEO인사말이나 소개글도 수정이 가능했으면 해요.

+ **제안사항**
    - Board App
    - 게시판 기능을 만들고 카테고리를 관리자가 생성함으로써 원하는 게시판을 만들고 때에따라 여러가지 용도로 사용하자.
        - ex) 이벤트용 게시판. 설문. 입사지원 자소서 등...

📜 메인사이트

	요구사항	기능적내용
header	배경을 전경사진으로 회사 소개를 시각적으로 강조	배경이미지 ● Css로 처리 ○ Opacity : 배경을 어둡게처리 ● 회사 소개를 배경위에 얹기
Section : about	CEO 소개란	About DB 생성 ● CEO 소개 Column 생성 Intro DB 생성 ORM을 이용해 디비에서 불러와 html에 추가
Section: services	회사의 주요 서비스 소개	Service DB 생성 ● Service Column 생성 Contact DB 생성 ● Pax ● Office ● Call ● Email ORM을 이용해 디비에서 불러와 html에 추가
Section: machine	회사의 주요 기계	향후에 바뀌지 않을것을 감안하여 고정된 사진으로 사용

📜 프로덕트 사이트

	요구사항	기능적내용
navbar	서랍형 형식으로 좌측에 배치	**상품의 정보를 유저가 보기쉽게 디자인** ● 서랍형 navBar 사용
	제품 종류 별	**product_Category DB 생성** ● Category : 상품 군 이름
Section : products	관련상품 진열	**Product DB생성** ● Name : 이름 ● Description : 설명 ● Price : 가격 ● Photo : 사진이름 **관련 Image 파일 관리** ● Media 폴더에 따로 모아서 관리 ● 이미지처리 : Pillow 라이브러리 사용
	기성 상품별 발주기능 메일알림 추가	**Order DB 생성** ● Email : 사용자 이메일 ● Subject : 요청 제목 ● Order_count : 주문량 ● Description : 요청내용 ● Created_at : DB 생성시간 메일전송 함수 및 DB를 동일한 것으로 사용 BackEnd 에서 하나의 이메일 계정으로 일괄처리 전송
	사용자 주문제작 발주기능 메일알림 추가	**Order DB 사용** 메일전송 함수 및 DB를 동일한 것으로 사용 BackEnd 에서 하나의 이메일 계정으로 일괄처리 전송

📜 게시판 사이트

	추가사항	기능적내용
Board App	게시판 카테고리 생성	Product template을 재사용 (Extends 이용) Adaptor 패턴 이용 (interface 를 이용하여 다른 뷰를 보여줌)
	관리자만 보기 기능	
	게시글 수정 및 삭제	

📜 관리자 사이트

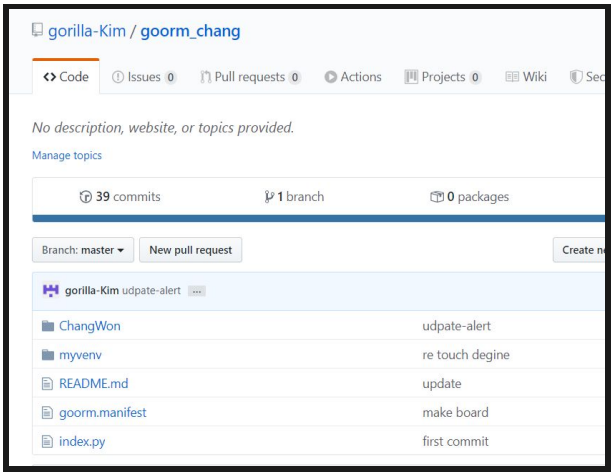
	요구사항	기능적내용
Admin App	관리자 로그인	Django Admin app을 재사용컴포넌트로 사용 Interface를 이용해 Customizing
	관리자 사이트	
	사이트 관련 DB 접근	
	관리자 계정 생성	
	관리자 Template ● 깔끔한 디자인. ● mobile, desktop 호환성	

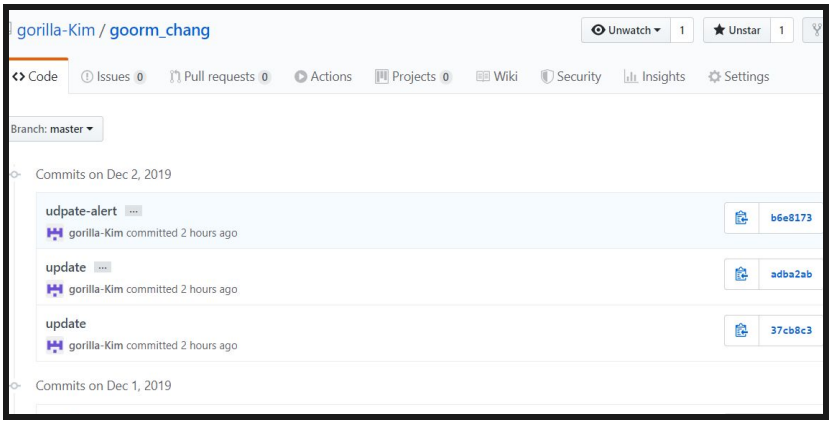
형상관리 및 테스트진행

✔ 형상관리

-형상관리 도구 사용 (Github)

- 사용 서비스 : Github
  - 이유 : 가장 보편적이고. 한 계정당 하나의 Private repository를 지원해줌
  - 팀 내 Github 서비스 유경험자 2명이상
- 사용방법
  - 각자 repository를 만들고 수정사항이 발생시 PM의 repository에 병합
  - 병합 이후 각자 pm repository를 clone후 작업 재실시 (매 스프린트/회의 시 수행)
  - Brunch 미사용 이유
    - 경험부족
    - 팀원 모두가 Git을 배우고 시작하기에 시간적 여유가 없었음





https://github.com/gorilla-Kim/goorm_chang

✔ 테스트진행

테스트 케이스 작성

Mainsite db 와 views.py 테스트

 # model test

```
from django.test import TestCase
from django.db import models
from . models import Intro, About, Service, Contact, Portfolio
from . import views
from django.urls import reverse
# Create your tests here.
class test_Mainsite(TestCase):
    @classmethod
    def setUpClass(cls):
        super(test_Mainsite, cls).setUpClass()

        intro = Intro(
            client = 'exampleperson'
        )
        intro.save()

        service = Service(
            tech_name = 'metal',
            tech_content = 'metalslug'
        )
        service.save()

        about = About(
            context = '123123123'
        )
        about.save()

        contact = Contact(
            office = 'seoul',
            call = '01071883841',
            pax = '123123',
            email = 'nexus2493@gmail.com'
        )
        contact.save()

        portfolio = Portfolio(
            name  = "imgname",
            description = "imgdes",
```

```
            image = "C:/Users/13053/OneDrive/바탕 화면/주석 2019-12-02 193930.jpg",

        )
        portfolio.save()

class Board_Category_ModelTestCase(test_Mainsite):
    def test_Category(self):
        print('Mainsite App test')
        intro = Intro.objects.get(client = 'exampleperson')
        about = About.objects.get(context = '123123123')
        service = Service.objects.get(tech_name = 'metal')
        contact = Contact.objects.get(office = 'seoul')
        portfolio = Portfolio.objects.get(name = "imgname")

        self.assertEqual(intro.client, 'exampleperson')
        self.assertEqual(service.tech_name, 'metal')
        self.assertEqual(service.tech_content, 'metalslug')
        self.assertEqual(about.context, '123123123')
        self.assertEqual(contact.office, 'seoul')
        self.assertEqual(contact.call, '01071883841')
        self.assertEqual(contact.pax, '123123')
        self.assertEqual(contact.email, 'nexus2493@gmail.com')
        self.assertEqual(portfolio.name, 'imgname')
        self.assertEqual(portfolio.description,'imgdes')
        self.assertEqual(portfolio.image,"C:/Users/13053/OneDrive/바탕 화면/주석 2019-12-02
193930.jpg")
        print('Mainsite App test success')


#views.py test
class HomePageTests(TestCase):
    print('\n')
    print("Mainsite App view.py test")
    def test_home_page_status_code(self):
        response = self.client.get('/')
        self.assertEquals(response.status_code, 200)


    def test_view_uses_correct_template(self):
        response = self.client.get(reverse('main'))
        self.assertEquals(response.status_code, 200)
        self.assertTemplateUsed(response, 'mainsite/index.html')

    def test_home_page_contains_correct_html(self):
        response = self.client.get('/')
        self.assertContains(response, 'Chang-Won')
```

```
    def test_home_page_does_not_contain_incorrect_html(self):
        response = self.client.get('/')
        self.assertNotContains(
            response, 'Hi there! I should not be on the page.')

    print("Mainsite App view.py test success")
```

Product db 와 views.py 테스트

 # model test

```
from django.test import TestCase
from django.db import models
from . models import KindOfProduct, Product, Order
from django.urls import reverse

# Create your tests here.
class test_Product(TestCase):
    @classmethod
    def setUpClass(cls):
        super(test_Product, cls).setUpClass()

        kindofproduct = KindOfProduct(
            name = "kind"
        )
        kindofproduct.save()

        product = Product(
            kindOf = KindOfProduct.objects.get(name = 'kind'),
            name = "1",
            description = "1",
            price = 1,
            photo = "C:/Users/13053/OneDrive/바탕 화면/주석 2019-12-02 193930.jpg",
        )
        product.save()

        order = Order(
            email = "1@naver.com",
            pwd = "1",
            subject = "1",
            order_count = 1,
            description = "1"
        )
        order.save()
```

```
class Board_Category_ModelTestCase(test_Product):
    def test_Category(self):
        print('Product App test')
        kindofproduct = KindOfProduct.objects.get(name = 'kind')
        product = Product.objects.get(name = '1')
        order = Order.objects.get(email = '1@naver.com')

        self.assertEqual(kindofproduct.name, 'kind')

        self.assertEqual(str(product.kindOf), 'kind')
        self.assertEqual(product.name, '1')
        self.assertEqual(product.description, '1')
        self.assertEqual(product.price, 1)
        self.assertEqual(product.photo, "C:/Users/13053/OneDrive/바탕 화면/주석
2019-12-02 193930.jpg",)

        self.assertEqual(order.email, '1@naver.com')
        self.assertEqual(order.pwd, '1')
        self.assertEqual(order.subject, '1')
        self.assertEqual(order.order_count, 1)
        self.assertEqual(order.description, '1')
        print('Product App test success')


#views.py test
class HomePageTests(TestCase):
    print('\n')
    print("Product App view.py test")
    def test_home_page_status_code(self):
        response = self.client.get('/products/')
        self.assertEquals(response.status_code, 200)

    def test_view_uses_correct_template2(self):
        kindofproduct = KindOfProduct(
            name = "kind"
        )
        kindofproduct.save()

        product = Product(
            kindOf = KindOfProduct.objects.get(name = 'kind'),
            name = "1",
            description = "1",
            price = 1,
```

```
            photo = "C:/Users/13053/OneDrive/바탕 화면/주석 2019-12-02 193930.jpg",
        )
        product.save()

        order = Order(
            email = "1@naver.com",
            pwd = "1",
            subject = "1",
            order_count = 1,
            description = "1"
        )
        order.save()

        self.client.login(username='user', password='test')
        response = self.client.get('/products/%d' %product.id)
        self.assertEquals(response.status_code, 301)
        self.assertTemplateUsed('products/product.html')

    print("Product App view.py test success")
```

Board db 와 views.py 테스트

```
# model test
from django.test import TestCase
from django.db import models
from . models import Board, Board_Category
from django.urls import reverse
from django.shortcuts import redirect

# Create your tests here.
class test_Board_Category(TestCase):
    @classmethod
    def setUpClass(cls):
        super(test_Board_Category, cls).setUpClass()
        board = Board_Category(
            name="공지사항",
            admin_option = False,
            created_at=1,
            updated_at=1
        )
        board.save()

        boards = Board(
```

```
            board_category = Board_Category.objects.get(name = '공지사항'),
            subject = 'testsubject',
            email = 'test@gmail.com',
            content = 'test',
            hit = 1,
            admin_option = True,
            created_at = 1,
            updated_at = 1,
            ifmodify = "n"
        )

        boards.save()

class Board_Category_ModelTestCase(test_Board_Category):
    def test_Category(self):
        print('Board App test')
        ob = Board_Category.objects.get(name = '공지사항')
        self.assertEqual(ob.name, '공지사항')
        self.assertEqual(ob.admin_option, False)
        self.assertEqual(str(ob.created_at)[:10], '2019-12-03')
        self.assertEqual(str(ob.updated_at)[:10], '2019-12-03')
        obs = Board.objects.get(subject= 'testsubject')
        self.assertEqual(obs.subject, 'testsubject')
        self.assertEqual(obs.email, 'test@gmail.com')
        self.assertEqual(obs.content, 'test')
        self.assertEqual(obs.hit, 1)
        self.assertEqual(obs.admin_option, True)
        self.assertEqual(obs.ifmodify,"n")
        print('Board App test success')


# views.py test
class HomePageTests(TestCase):
    print('\n')
    print("Board App view.py test")

    def test_view_uses_correct_template1(self):
        board = Board_Category(
            name="공지사항",
            admin_option = False,
            created_at=1,
            updated_at=1
        )
```

```
        board.save()
        response = self.client.get('/board/')
        self.assertEquals(response.status_code, 200)

        self.assertTemplateUsed(response, 'board/index.html')

    def test_view_uses_correct_template2(self):
        self.client.login(username='user', password='test')
        response = self.client.get('/board/read/18')

        self.assertEquals(response.status_code, 302)
        self.assertTemplateUsed(redirect('main'))

    def test_view_uses_correct_template2(self):
        self.client.login(username='user', password='test')
        response = self.client.get('/board/update_board/18')
        self.assertEquals(response.status_code, 302)
        self.assertTemplateUsed(redirect('board_main'))

    def test_view_uses_correct_template2(self):
        boards = Board_Category(
            name="공지사항",
            admin_option = False,
            created_at=1,
            updated_at=1
        )
        boards.save()
        ob = Board.objects.create(board_category = Board_Category.objects.get(name
= '공지사항'),
            subject = 'testsubject',
            email = 'test@gmail.com',
            content = 'test',
            hit = 1,
            admin_option = True,
            created_at = 1,
            updated_at = 1,
            ifmodify = "n")
        ob.save()
        print(ob)
        self.client.login(username='user', password='test')
        response = self.client.get('/board/updatepage/%d' %ob.id)
        self.assertEquals(response.status_code, 200)
        self.assertTemplateUsed(response, 'board/update.html')
```

```
def test_view_uses_correct_template2(self):
    boards = Board_Category(
        name="공지사항",
        admin_option = False,
        created_at=1,
        updated_at=1
    )
    boards.save()
    ob = Board.objects.create(board_category = Board_Category.objects.get(name =
'공지사항'),
        subject = 'testsubject',
        email = 'test@gmail.com',
        content = 'test',
        hit = 1,
        admin_option = True,
        created_at = 1,
        updated_at = 1,
        ifmodify = "n")
    ob.save()
    print(ob)
    self.client.login(username='user', password='test')
    response = self.client.get('/board/delete_board/%d' %ob.id)
    self.assertEquals(response.status_code, 200)
    self.assertTemplateUsed(redirect('board_main'))

    print("Board App view.py test success")
```

테스트 진행

```
root@goorm:/workspace/sw_project/ChangWon(master)# python manage.py test


Board App view.py test


Mainsite App view.py test
Mainsite App view.py test success


Product App view.py test
Product App view.py test success
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Board App test
Board App test success
.==========공지사항 접속==========
==========메인페이지 접속==========
.testsubject
.Mainsite App test
Mainsite App test success
.....Product App test
Product App test success
./workspace/sw_project/ChangWon/products/views.py:47: UnorderedObjectListWarning: Pagination may yiel
dered object_list: <class 'products.models.Product'> QuerySet.
  paginator = Paginator(products, 3)
..
----------------------------------------------------------------------
Ran 11 tests in 0.366s

OK
Destroying test database for alias 'default'...
```