[11주차 (5월 28일 ~ 6월 3일)]

주제 : 지능형 욕조(Inteligent Tub)
빅데이터와 IoT 기술을 융합하여 웹을 사용해 서비스하는 플랫폼

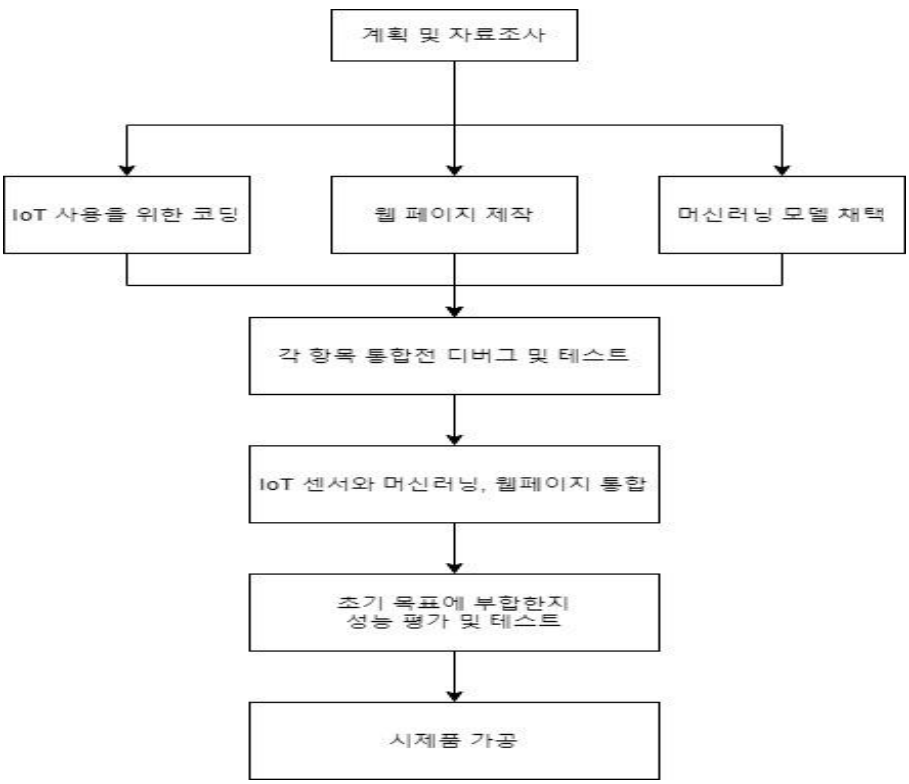
팀 인원 :
- 임대인 : 웹 백엔드, IoT 코딩, 데이터 분석, 시제품 제작
- 정해민 : 웹 프론트엔드
- 서정욱 : 웹 프론트엔드, 데이터 분석
- 조휘훈 : 데이터 수집 및 정제
- 박지수 : 데이터베이스 설계, 시제품 제작

개발 동기 및 목적 :

❑ 개발동기
- 샤워는 일주일에 적어도 한번하는 활동, 단순한 위생활동이 아닌 하루를 시작 또는 마무리할 때 하는 중요한 활동이라 생각하여 보다 편리하게 해주는 방법이 있지않을까 생각하다 개발하게 되었음
- 샤워에도 사람마다 취향이 있고 선호하는 것이 있는데 획일화 되는 것이 아쉬웠음, 더 나아가 사용자의 사용데이터, 사람들의 사용 데이터를 수집하여 데이터 분석을 통해 직업별, 날씨별과 같이 샤워라는 활동에 부가적인 서비스를 제공하는 플랫폼을 제작하고자함

❑ 개발목적
- 힘든 하루를 보낸 회사원, 학생이나 주부 등 여러 사람들이 자동으로 피로를 풀 수 있게 하기위해
- 개인의 기호를 시스템화 시켜서 가장 좋아하는 환경으로 샤워를 할 수 있게 하기위해
- 빅데이터와 결합시켜 여러 요인들을 이용해 예측해서 사용자의 편의성을 극대화하기 위해

개발 계획 및 일정 :



추진 내용	수행기간(월) (계획표시 : ■)												비고
	4 월				5 월				6 월				
	1	2	3	4	1	2	3	4	1	2	3	4	
프로젝트 회의 및 주제선정	■	■											
개발언어 및 디자인선정		■	■										
개발 언어, 프로그램 학습			■	■									
데이터 수집 및 정제		■	■	■	■								
개발진행				■	■	■	■	■	■				
오류최소화 및 리팩토링									■	■	■		
사용자 시연 및 배포											■	■	
프로젝트 최종시연 및 정리												■	

개발 환경 :

- OS : Microsoft Windows 10 Education (version: 1903, build: 18362)
- DB : MySQL (version: 8.0)
- 웹 : HEROKU (웹 배포, version : 7.39.5), Node.js (웹 서버, version: 12.16.2 LST)
- 소스편집기 : VisualStudio Code (version: 1.44.2)

기대효과 및 활용방안 :

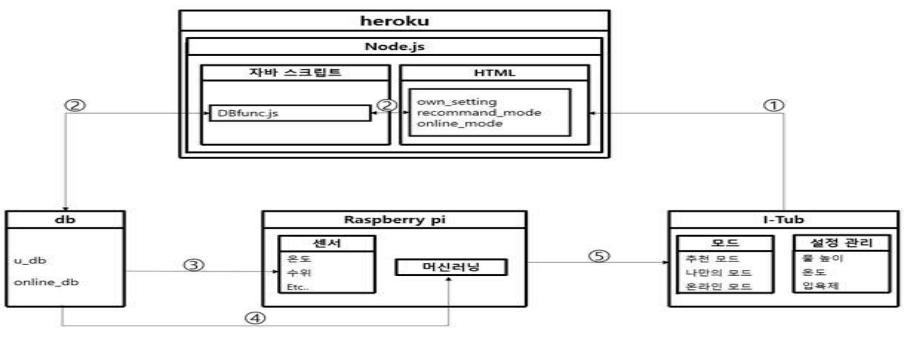
❑ 기대효과

- 이것저것 준비할 필요없이 터치 몇 번으로 환경이 갖추어짐
- 학교, 가사노동, 여러 활동의 피로를 풀 수 있는 환경을 쉽게 마련해 줌
- 편하고 쉽게 사용을 원하는 요즘 젊은 세대와 요구를 충족할 것으로 기대되고 노년층, 어린이들과 같이 여러 연령층이 쉽게 사용할 것으로 기대됨
- 욕조만 있다면 탈부착이 가능한 키트로 어디서든 활용할 수 있어 여러 사업, 집에서 쉽게 설치 가능할 것으로 기대

❑ 활용방안

- 샤워/목욕을 하나의 컨텐츠 화 시켜 찜질방, 사우나와는 다르게 사람의 기호별로 다른 방식의 경험을 할 수 있게 함
- 욕조가 없는 환경이더라도 키트의 모양을 약간 변경한다면 사용가능
  (부품의 기종을 변경, 내부 SW는 바꿀 필요가 없음)

시스템 흐름도 :



① MySQL을 사용한 DB, 웹서버와 사용자 데이터(닉네임, 나이, 성별 등)를 주고 받는다.
DBfunc.js가 MySQL과의 연결을 담당한다.
② 사용자에게 보여지는 웹으로 DB와 주고받은 데이터를 보여주는 역할을 한다.
③ DB 서버, 웹 페이지를 담당하는 네트워크 플랫폼 (프레임워크로 express를 사용)
④ Node.js로 만들어진 웹 서버를 사용자들이 사용할 수 있게 배포해주는 플랫폼
⑤ 각종 센서와 기능을 담당하기 위한 라즈베리파이 (온도, 수위 센서, 모니터 등)
또한 축적된 데이터로 머신러닝을 하여 모델을 생성
⑥ 웹과 라즈베리파이가 만든 서비스를 사용하는 사용자

[금주진행사항]

- 자바 스크립트 코드 변경 (웹 서버 통신을 담당하는 코드를 변경)

index.js	URL 컨트롤러 역할을 하는 자바 스크립트 ——————————————————————————————— 원래는 GET 방식의 요청으로 주소에 파일명이 노출이 되었지만 POST 방식의 요청으로 변경하여 파일명 노출 방지 및 잘못된 방법의 접근을 방지하는 코드를 추가
app.js	웹 서버에서 사용할 기능들을 선언한 자바 스크립트 ——————————————————————————————— 웹 서버에서 사용할 미들웨어들을 선언하고 오는 요청을 index.js를 통해 적절한 처리를 해주는 코드를 추가

- 설문조사 데이터 추가 수집

진행사항	통계, 분석 간 표본이 부족한 연령, 시간대 데이터가있어 추가적으로 데이터를 지속적으로 수집

- 설문한 데이터를 바탕으로 시각화

시각화	파이썬 언어를 사용하여 수집한 데이터들을 시각화, 올바른 데이터인지 확인하는 과정을 진행

[금주구현코드]

- index.js
(GET 방식의 요청에서 POST 방식의 요청으로 방식을 변경)

```
var  express  = require ('express');
var  path  = require ('path');
var  cookieParser  = require ('cookie-parser');
var  bodyParser  = require ('body-parser');
var  router  = express .Router ();
var  db  = require ('./db/DBfunc');
app  = express ();
app .use (cookieParser ());
app .use (bodyParser .json ()) // json의 형태로 받을때
app .use (bodyParser .urlencoded ({ extended : true  })) // 인코딩된 url 형태로 받을때
router .use ('/db', db );
// url routing
router .get ('/', function  (req , res ) {
    console .log ('접속 성공적')
    res .sendFile (path .join (__dirname  + "/../public/main_page.html")) // html 파일을 보내는 것
});
router .post ('/profile', function  (req , res ) {
    res .sendFile (path .join (__dirname  + "/../public/profile.html")) // html 파일을 보내는 것
});
router .post ('/profile_cookie', function  (req , res ) {
    res .cookie ('cookie_name', req .body .cookie_name );
    res .cookie ('cookie_age', req .body .cookie_age );
    res .cookie ('cookie_gender', req .body .cookie_gender );
    res .cookie ('cookie_job', req .body .cookie_job );
    res .redirect (307 , '/user');
});
router .post ('/online', function  (req , res ) {

    res .sendFile (path .join (__dirname  + "/../public/online_test_mode.html")) // html 파일을 보내
```

```
는 것
});
router .post ('/recommend', function  (req , res ) {
    res .sendFile (path .join (__dirname  + "/../public/recommend_mode.html")) // html 파일을 보내는
것
});
router .post ('/user', function  (req , res ) {
    res .sendFile (path .join (__dirname  + "/../public/user_choice.html")) // html 파일을 보내는 것
});
router .post ('/own', function  (req , res ) {
    res .sendFile (path .join (__dirname  + "/../public/own_setting.html")) // html 파일을 보내는 것
});
router .post ('/register', function  (req , res ) {
    res .sendFile (path .join (__dirname  + "/../public/register.html")) // html 파일을 보내는 것
});
// URL 직접 접근 막기
router .get ('/*.html', function  (req , res ) {
    console .log ('직접 접근하지 마라 ㅡㅡ')
    res .sendFile (path .join (__dirname  + "/../public/main_page.html")) // html 파일을 보내는 것
});
module .exports  = router ;
// 이렇게 index.js가 컨트롤러 역할을 해준다.
```

- app.js

(POST 요청이 오면 index.js를 사용해 적절한 처리를 할 수 있도록 코드 변경)

```
var  express  = require ('express')
var  app  = express ()
var  bodyParser  = require ('body-parser')
var  index  = require ('./router/index')
var  db  = require ('./router/db/DBfunc')
port  = process .env .PORT  || 8000 ;
app .listen (port , function (req , res ) {
    console .log ("Open I-Tub server!! congratulation");
});
// 이부분을 middleware 라고 한다.
app .use (express .static ('public'))
app .use (bodyParser .json ()) // json의 형태로 받을때
app .use (bodyParser .urlencoded ({extended :true })) // 인코딩된 url 형태로 받을때
app .use ('/db', db )
app .use ('/', index );
app .use ('/profile', index );
app .use ('/user', index );
app .use ('/own', index );
app .use ('/recommand', index );
app .use ('/online', index );
app .use ('/register', index );
app .use ('/profile_cookie', index );
app .set ('view engine', 'ejs') // view engine으로 ejs를 사용한다는 의미
```

[차주예정사항]

머신 러닝	수집한 데이터를 바탕으로 머신 러닝 학습을 하여 회원에게 알맞은 모드를 추천해주는 기능을 추가 예정 (10주차에 진행하려 했으나 데이터가 부족하여 원하는 성능이 나오지 않아 추가 데이터 수집후 진행 예정)
데이터 추가	설문 데이터 추가 (표본이 적은 연령대나 시간대가 있어 추가적으로 데이터를 수집하고 있음)

[팀내건의사항]

없음