프로젝트 진행 시 어려웠던 점

2024-09-27

프로젝트를 진행하면서 회원 가입 기능을 구현하였고 그 다음으로 회원 가입 시 아이디가 중복되는지 확인하는 기능을 구현하고 있었습니다.

정확하게 fetch 를 작성하였고 민감한 정보인 아이디였기 때문에 POST 형식으로 전송하였습니다. 하지만 데이터가 전송이 되지 않았습니다.

400 즉 주소를 못찾는 문제가 발생하였고 alert으로 계속 확인했을 때는 .value 시에 값이 제대로 읽히는 것 까지 확인하였습니다. 하지만

버튼을 클릭 시 데이터는 읽지 못해 주소가 문제가 발생하는 것을 알게 되었습니다.

해결 방법은 자바스크립트가 블록 스코프인 것을 깜빡하고 있었고 이벤트리스너 안에서는 초기값을 가진다는 것을 생각해냈습니다.

그래서 주소로 읽을 값을 이벤트리스너 안에서 정의하였고 스프링 부트에서 데이터가 정확하게 전송되는 것을 확인할 수 있었습니다.

2024-09-28

fetch 문제를 해결하였으나 @PostMapping를 사용하는 함수에 레포지토리에서 id 값을 못찾는 문제가 발생하였습니다. fetch에서 중복을

찾기위해 유저 아이디를 전송했고 레포지토리는 String 타입으로 저장되었기 때문에 String 타입으로 받았으나 JSON 형식으로 전송되기

때문에 문제가 발생하였습니다. 그래서 찾아본 결과 JSON으로 오기 때문에 한번 파싱 작업을 해야한다는 것을 알게 되었습니다. 그래서

finduser(@RequestBody UserDTO users) 로 DTO 형식으로 변경하자 아이디 값을 찾을 수 있었습니다.

2024-10-01

스프링 시큐리티를 구현하는 과정에서 스프링부트를 공부하며 배운 내용으로 구현하였는데 스프링 시큐리티가 제대로 동작하지 않아서

이틀 간 수정하였고

문제가 발생하게된 이유는 sign.html 페이지를 만들고 form 태그에서 action="/login" 으로 했고 찾아본 결과로도 sign.html로 하고

action은 "/login"을 해도 문제가 없다 였습니다. 해결한 것은 action="/sign"으로 변경하고 아이디와 패스워드를 member\_id, member\_password

로 하였는데 username, password로 변경하였습니다. 그리고 데이터베이스 user 테이블을 member로 테이블명을 바꿨는데 나머지 외래키를

수정하지 않아서 발생한 문제였습니다.

2024-10-04



처음 게시판 페이지를 만들 때는 Board 엔티티 클래스에 Id는 board\_id 이런식으로 작성하였는데 이런 경우에는 JPA가 매핑을 하지 못해 오류가 발생하였습니다. 이를 해결하기 위해 각 테이블의 기본키는 id로 선언하고 @Column(name=”데이터베이스 테이블 기본키명”)으로 변경하여 해결하였습니다.

2024-10-07



게시판에 글 들 중에서 하나를 클릭하면 디테일 페이지로 넘어가는 기능을 만들었는데 이미지가 있을 경우에는 정상적으로 페이지가 화면에 출력되었지만 이미지가 없는 경우에는 화면이 정상 출력되지 않는 현상이 발생하였습니다. 이를 해결하고자 사진과 같이 작성하였는데 발상을 바꿔 board는 image 와 연관 관계인 것을 생각해 board 만 model로 전해줘도 되겠다는 생각을 하게 되었습니다. 그래서 디버그 기능으로 확인한 결과 board 객체만 호출해도 image도 함께 불러와져서 board만 전송해도 되는 것을 알게되어 디테일 페이지를 코드를 줄일 수 있게 되었습니다.



게시판 디테일뿐만 아니라 게시판 목록도 이미지 객체를 만들어 전송하는 것이 아닌 board 객체 하나를 전송하면 기존 기능을 수행하도록 변경하였습니다.



2024-10-13

10월 7일 이후로는 대부분의 기능을 구현하여 코드의 가독성과 중복되는 기능을 제거하면서 수정 작업을 진행하였습니다. 코드를 정리하면서 이전에 대충 구현했던 기능 중에서 논리적으로 이상한 부분을 오늘 수정하였습니다. 로그인된 상태에서만 게시판 글 삭제 수정 기능이 동작하게 만들었는데 삭제 버튼을 누르면 사용자가 누구든간에 삭제가 진행되는 이상함을 발견하였습니다. 이전에 기능 구현에 초점을 두어 기능들을 만들다 보니 삭제 기능을 간단하게 window.location.href=`/delete`; 로 작성하고 컨트롤러에서도 사용자 식별 없이 바로 삭제하는 과정을 거치기 때문에 사용자 식별 후 삭제 기능을 추가하였습니다.

2024-10-14

댓글 수정 기능을 구현하는 중에 고민한 생각으로 댓글을 후 수정 기능을 넣을지 아니면 다른 사이트처럼 삭제 기능만 구현하고 수정 기능을 없애버릴지 고민하였습니다.

화장품 추천 기능을 구현하기 위한 뼈대 작업으로 버튼을 클릭하면 이미지가 될 부분이 슬라이더 형식으로 동작하는 기능을 구현하였습니다.

2024-10-15

댓글 수정 기능에 대한 고민하여 내린 결론으로 댓글 수정 기능은 빼는게 낫다라고 생각하였습니다. 대부분의 사이트에서는 게시글에 대한 수정 기능은 제공하지만 댓글 수정 기능은 제공하지 않습니다. 그리고 댓글의 경우 사이트에 중요 기능이 아니고 로직 또한 복잡해져서 포트폴리오를 완성하고 나중에 구현하는게 낫다라고 판단하였습니다. 그래서 구현하고자하는 스마트 메이크업 사이트에서도 댓글에는 수정 기능을 잠시 빼고 구현하는 것으로 결정하였습니다.

게시판 삭제 기능에서 삭제 후에 이미지 링크는 데이터베이스에서 삭제되었는데 이미지는 스프링 부트 static 폴더에 계속 남는 현상을 수정하여 게시판을 삭제하면 이미지도 함께 삭제하게 변경하였습니다.

이미지를 삭제할 때 파일의 경로가 상대 경로일 경우 삭제가 되지 않아 절대 경로로 삭제하였습니다.

2024-10-16

화장품 추천 페이지를 제작하는데 맨 처음 생각으로는 이미지 5개가 출력되고 버튼을 클릭하면 이미지가 한칸씩 이동하고 이동되는 이미지가 끝이면 해당 이미지는 사라지고 비게된 이미지 부분은 새 이미지로 교체하는 방식 [링크](https://github.com/CJW00113/Smart_Makeup_Archive/blob/main/archive-html/slide.html) 를 생각하였습니다.





위에 예시 처럼 제작하려 했습니다.



그래서 위에 코드 처럼 이미지를 출력하고 필요한 부분만 디스플레이에 출력하고 나머지는 숨기는 기능을 사용하려 했습니다. 하지만 이러한 부분에 CSS와 Javascript를 사용해서 구현하려 할 때 오류가 발생하여 프로젝트 진행에 어려움을 겪게 되었습니다. 그러다가 고민하던 도중에 깨닳음을 얻어 이런 방식으로 구현하고자 합니다.

div 태그에 5개의 img 태그를 만들고 boarddetail.html에서 처럼 자바스크립트에서 컨트롤러로 전달된 모델 데이터를 받아서 [링크](https://hosunghyun.tistory.com/entry/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8%EC%97%90%EC%84%9C-%ED%83%80%EC%9E%84%EB%A6%AC%ED%94%84-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0) 에서처럼 데이터를 받고 각 이미지 태그에 이미지 링크를 삽입하면 어떻까? 라는 생각이 들었습니다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

위에 코드처럼 5개의 img 태그를 지정하고 자바스크립트로 이미지 링크를 받아서 상황에 맞게 이미지 링크를 적용하여 이전과 달리 오류가 발생하지 않게 되었습니다.

2024-10-17

같이 프로젝트를 진행하는 조원과 화장품 추천 페이지에서 이미지에 추가 기능을 넣을지에 대한 회의를 하였고 처음 계획서를 작성할 때 이미지 슬라이더 기능만 계획하였고 이미지 클릭시 판매 사이트로 가는 등의 기능은 계획에 없었고 화장 기능이 메인이라 사이트 링크까지 있는 것은 웹 쇼핑이 되는 것이라 추가하지 않는다라고 결정되었습니다

대부분의 기능을 구현하였고 디자인을 시작하기 위해 CDN 링크를 html 코드에 삽입하자 기존에 디자인했던 메뉴바의 디자인에 오류가 발생하여 오류가 발생하는 CDN 링크를 잠시 주석 처리하였습니다.

2024-10-18

화장품 추전에서 기존에 이미지와 버튼만 있던 디자인에서 부트스트랩을 이용해서 이미지를 카드 형태로 디자인하였고 아직 까지는 이동효과가 없어 부자연스럽지만 후에 자연스러운 이미지 변경 기능을 추가할 예정

로그인 페이지에서 로그인 폼과 아이디 찾기 폼을 부트스트랩으로 디자인



로그인 뿐만 아니라 프로필 그리고 게시판 글 수정하기 페이지도 부트스트랩으로 디자인을 수정하였습니다.

2024-10-19

부트스트랩으로 어느정도 디자인이 완료되어 구현하지 않았던 아이디 찾기 기능에 형태를 구현하였습니다.

아이디를 찾는 기능을 구현하였는데 이메일만 있어도 아이디가 찾기가 가능해서 이메일로 아이디 찾기 따로 전화번호로 아이디 찾기 따로 해서 서로 같으면 아이디를 출력하는 기능으로 구현할 지 고민했습니다. 조원과의 회의를 하고 난 뒤에 정해질 것 같습니다.

아이디 찾기 기능을 어느 정도 구현하여 비밀번호 찾기 기능의 뻐대를 작성하였습니다.

2024-10-20

비밀번호 찾기 기능을 구현하였습니다. 아이디와 이메일을 입력하면 비밀번호가 출력되는 로직으로 구현하였는데 데이터베이스에는 암호화된 상태로 존재하기 때문에 아이디와 이메일로 찾은 비밀번호는 암호화 상태라서 사용자가 비밀번호를 알 수 없는 상태였습니다. 그래서 복호화 방법을 찾았는데 스프링 시큐리티에서는 제공하지 않는 기능이고 보통은 새 비밀번호를 설정하는 방법이라 해서 비밀번호를 새로 설정하는 기능으로 변경하였습니다.

자바스크립트 코드를 유지보수가 쉽게 js 파일로 나누려 했는데 경로 오류가 발생해서 한번에 작성했었는데 타임리프를 사용해서 경로를 지정했더니 오류가 해결되어 기존에 하나로 작성된 코드들을 분리 작업을 수행하였습니다.

2024-10-21

게시판 작성 시 이미지가 제대로 저장되지 않는 오류를 수정했습니다. 원인으로는 상대 경로를 절대 경로로 변경하는 과정에서 잘못된 경로를 작성해서 이미지가 저장될 공간을 찾지 못해 오류가 발생하여 해당 부분을 수정하였습니다. 또한 게시판 삭제에서도 게시판의 이미지 경로를 찾지 못하던 오류를 수정하였습니다.

댓글 기능에서는 댓글을 삭제하면 홈 화면으로 이동하는 것을 다른 사이트와 마찬가지로 게시판에 페이지가 그대로 머물도록 수정하였습니다. 그리고 댓글 작성시 로그인되지 않은 사용자가 댓글 작성시 컨트롤러 부분에서 사용자를 찾지 못해 오류가 발생하는 것을 수정하여 로그인 되지 않으면 로그인 화면으로 이동하도록 수정하였습니다.

화장품 추천 페이지에서 CSS 파일 경로를 잘못 적어 발생한 오류를 수정하였습니다.

로그인 페이지에서 아이디와 비밀번호 찾기 기능에서 예외처리를 작성하였습니다.

비밀번호 찾기 기능을 작성하여 새 비밀번호로 변경이 가능하게 기능을 구현하였습니다.

2024-10-22

본격적인 디자인을 하기 위해서 index.html에 폰트를 적용하는 작업을 하였습니다.

@font-face {

font-family: 'MyCustomFont'; /\* 원하는 이름 \*/

src: url('fonts/mycustomfont.woff2') format('woff2'), /\* WOFF2 포맷 \*/

url('fonts/mycustomfont.woff') format('woff'), /\* WOFF 포맷 \*/

url('fonts/mycustomfont.ttf') format('truetype'); /\* TTF 포맷 \*/

font-weight: normal;

font-style: normal;

}

예시 처럼 무료로 다운 받은 폰트를 적용하려했는데 적용되지 않는 상황이 발생하였습니다. 원인은 폰트 파일의 이름에는 띄어쓰기가 들어갔는데 url에서 불러올 때는 띄어쓰기를 하지 않아 다운받은 파일을 읽어오지 못했던 것으로 띄어쓰기를 수정하자 해결되었습니다.

2024-10-23



검은색 바탕의 기존 메뉴바에서 파란색 계열의 메뉴바로 변경하였고 글꼴도 변경하였습니다. 그리고 페이지 가장 하단에는 기존에 존재하던 사이트들과 비슷하게 디자인하였습니다.

게시판에는 기존에는 구조만 짜기 위해 첫번째 페이지만 보이게 설계하였는데 게시글 10개씩 끊어서 페이지들을 전부 볼 수 있게 변경하였습니다

2024-10-24

이전에 디자인하면서 배경색과 글자색을 바꿨는데 일부 페이지에서는 바꾸지 않아 새 디자인과 색이 다르던 것을 수정하였습니다.

게시판 기능 중에서 글 수정하는 기능을 구현하였습니다. 이전에는 수정 버튼을 누르면 이미지가 전부 삭제되고 사용자가 다시 업로드할 것인지 선택해서 삭제할 것인지에서 선택해서 삭제하는 기능을 구현하기로 조원과 상의했고 구현하였습니다.

2024-10-25

메인에서 이미지 여러 개를 돌아가면서 출력하는 화면을 구현하였습니다.

게시판 글 수정 취소 기능은 메인 화면으로 이동했는데 이전 페이지로 이동하는 코드를 활용해서 이전 페이지로 이동하도록 수정하였습니다.

게시판 글 수정 기능을 누르면 게시판으로 이동하게 작성되었는데 오류로 인해 게시판 페이지 + 수정한 게시판 글의 ID 에 해당하는 페이지로 이동하는 오류가 존재하여 게시판 첫번째 페이지로 이동하도록 수정하였습니다.

게시판 글 수정 취소 버튼은 이미지 부분에서 막혀 현재는 이미지 옆에 삭제 버튼을 누르면 바로 삭제되는 기능으로 제작하였습니다. 나중에 기회가 되면 이미지가 바로 삭제 되는 것이 아닌 삭제 버튼을 누르면 화면에서만 사라지고 수정을 확인하는 버튼을 클릭시 삭제되는 기능으로 변경할 예정입니다.

메인에서 이미지는 점멸 형식으로 이미지가 변환되는데 이 부분 또한 나중에 슬라이드 형식으로 변경할 예정입니다.

2024-10-26

스마트메이크업 사이트에 대한 대부분의 기능 구현과 디자인 작업이 끝나 HTML 파일에 자바스크립트 코드를 따로 자바스크립트 파일로 분리하였습니다.

2024-10-27

템플릿 폴더에 쓸모가 없던 test.html을 삭제하였습니다.

앞으로 회의할 내용으로 몇몇 이미지(제품, 메인 등)의 이미지는 데이터베이스에 저장되어있지 않고 HTML에서 바로 불러와 사용하고 있습니다. 현재는 데이터베이스에 저장되는 이미지는 사용자가 업로드한 이미지와 제품 이미지 뿐입니다. 이에 대해서 앞으로 다른 종류의 이미지를 업로드하게 될 수 있는데 이를 데이터베이스에 저장해야 하는지 현재 방식처럼 폴더에 저장하고 HTML 코드에서 불러올 지 정하는 회의를 할 예정입니다.

2024-10-28

스프링부트로 사이트 구현이 대부분 완료되었고 조원이 만든 FastAPI 또한 어느 정도 구현되었기 때문에 두 기능을 합쳤습니다.

오류가 발생하여 두 기능을 합치면 FastAPI 기능이 동작하지 않아서 향후 프로젝트 계획은 두 기능을 합쳐서 동작 시켜 완성하는 방향으로 진행될 것 같습니다.

현재까지 도전한 방식은 @ComponentScan("test.fastAPI")을 붙여서 스프링 시큐리티를 동작하는 것으로 이 방식은 스프링 시큐리티에 오류가 발생해 기존에 작성한 시큐리티 코드가 동작하지 않게 됩니다.

2024-10-29

기존에는 게시판 수정에서 이미지에 관한 기능으로 삭제 버튼을 누르면 바로 데이터베이스에 삭제가 반영되서 취소 버튼을 눌러도 복구가 불가능 했는데 변경 후에는 삭제 버튼 클릭시 리스트에 저장한 후에 화면에 안보이게 하고 수정 확인을 누르면 그 때 데이터베이스에서 삭제되는 로직으로 변경하였습니다.

FastAPI는 많은 오류로 인해 변경 사항을 취소하였고 조원이 완전히 구현하면 사이트에 반영할 예정입니다.

2024-10-30

Edit.html에서 자바스크립트 코드를 static 폴더에 자바스크립트 파일을 만들고 코드를 옮겼습니다.

스프링부트로 만든 smartmakeupweb 사이트와 조원이 제작한 FastAPI 파일을 합쳤습니다. Makeup.html 파일이 동작하게 되었지만 어떤 컴퓨터에서는 정상 작동하지만 어느 컴퓨터에서는 동작하지 않는 문제가 발생하였습니다. 문제의 원인으로 생각되는 것은 PythonRunner.java 파일에서 프로세스를 실행시키는 builder.start()라는 구문이 존재하는데 디버깅 하면서 예상한 바로는 해당 부분이 동작하지 않아 파이썬 서버가 실행되지 않는 문제가 발생하는 것 같습니다.

2024-10-31

opencv-contrib-python을 설치하고 나서 코드를 실행해보니 동작하는 것을 확인할 수 있었습니다. 하지만 중간에 강제종료 되는 것으로 보아 컴퓨터 성능에 따라 동작하는 것을 알 수 있었습니다. 이를 통해 계속 동작되지 않던 오류를 예상하면 현재 프로젝트를 진행하는 컴퓨터의 성능이 좋지 못해 FastAPI 코드 부분을 동작하지 못한 것으로 예상됩니다.

2024-11-01

Power shell 에서 python 프로그램을 동작 명령을 내렸는데 파이썬 파일이 동작하지 않는 것을 보고 시스템 환경 변수를 확인하였습니다. 확인 결과 경로의 오타가 있었고 이 부분을 수정하자 이전에 스프링부트 서버에서 FastAPI가 동작하지 않던 문제가 해결되었습니다.

2024-11-02

조원이 작성한 코드를 수정하여 가독성은 높아졌지만 수정하는 과정에서 주요 기능의 로직을 변경하면서 오류가 발생하였습니다. 해당 부분은 조원과 회의를 통해 수정할 예정입니다.

2024-11-03

|  |
| --- |
| !pip install pymysql  import pymysql  # 데이터베이스에 연결  connection = pymysql.connect(      host='localhost',  # MySQL 서버 호스트      user='root',  # MySQL 사용자명      password='1234',  # MySQL 비밀번호      database='smart\_makeup',  # 데이터베이스 이름  )  try:      with connection.cursor() as cursor:          # 데이터 조회 쿼리 실행          sql = 'SELECT \* FROM member'          cursor.execute(sql)          # 결과 가져오기          rows = cursor.fetchall()          # 결과 출력          for row in rows:              print(row)  finally:      # 연결 종료      connection.close() |

조원이 작성한 FastAPI 코드는 자바스크립트->스프링부트 컨트롤러->스프링부트 서비스->파이썬 FastAPI->메모장으로 저장되는 복잡함을 가지고 있습니다. 메모장으로는 후에 구현할 동작들을 구현할 때 문제가 발생할 것이라 예상되어 조원과 상의해서 로직을 변경하고자 합니다. 페이지에서 FastAPI에 필요한 데이터를 사용자에게 받아 데이터베이스에 저장하고 난 뒤에 파이썬에서 필요한 데이터를 위에 코드를 활용해 데이터베이스에 접근하는 방식을 활용하고자 합니다.

2024-11-04

회의 결과 POST로 전송하는 것이 데이터베이스에 저장해서 불러오는것보다 빨라 일단은 해당 기능은 유지하기로 했습니다.

기존에 카메라 동작 등의 코드 오류를 수정하였습니다. 카메라가 작동하고 HTTP 전송이 정상 작동합니다.

Makeup.html 파일을 수정하여 기존에는 하나의 색상 팔레트를 사용했지만 팔레트를 다르게 만들었습니다. 그리고 이에 맞게 makeup.js 파일도 수정하였습니다.

메이크업 테이블에 데이터를 저장하는 기능을 추가하였습니다.

2024-11-05

화장품 종류별로 화장 테이블과 제품 테이블에서 외래키로 역할하도록 변경하였습니다. 그리고 makeup.js 에서 그에 맞게 로직을 변경하여 원래는 화장품 종류 하나에 저장 한번을 눌러서 따로 저장을 해야 했는데 화장품 종류별로 함수 호출해서 데이터를 전송해 저장 버튼을 한번만 눌러도 전부 저장되도록 변경하였습니다. 그리고 컨트롤러와 서비스도 그에 맞게 로직을 변경하여 화장품 종류가 같으면 기존 저장을 수정하도록 변경하였습니다.

2024-11-06

저장된 데이터를 불러오면 데이터에 맞게 화면에 적용되도록 코드를 변경하였습니다. 그리고 데이터를 불러올 때 필요한 버튼은 로그인된 사용자만이 볼 수 있게 변경하였습니다.

메이크업 서비스의 로직 상의 오류로 원치 않은 데이터가 갱신되던 현상을 수정하였습니다. 기존에는 화장품 종류를 비교해서 저장하였지만 사용자가 저장하고자 하는 버튼까지 비교하여 저장하도록 변경하여 원치 않은 데이터가 갱신되는 현상을 수정하였습니다.

2024-11-07

회원 가입시 아이디 중복을 체크하는 기능만 존재하고 중복확인을 안했을 경우에 대한 기능이 존재하지 않는 모양만 잡아둔 기능을 최종 구현하여 앞으로 회원 가입 시 아이디 중복 체크를 하지 않으면 회원 가입할 수 없으며 중복 체크 이후에 아이디를 변경하면 다시 중복 체크하도록 기능을 구현하였습니다.

화장품 추천 기능을 수정하였습니다. 이전에는 사진만 보이게 구현하였지만 변경된 기능은 화면에 이미지뿐만 아니라 제품명과 제품가격 그리고 제품의 구매 페이지 이동을 구현하였습니다.

스프링부트 제품 추천 관련 컨트롤러를 구현하는 과정에서 model.addAttribute() 구현할 때 문제가 발생하였습니다. 해당 문제는 모델로 전송하려는 product 테이블이 image와 productcategory 같이 여러 테이블과 연관 관계에 있어 보내려는 데이터가 너무 크고 복잡해서 무한 루프에 빠져 보낼 수 없던 문제였습니다. 그래서 해결책으로 원하는 데이터 이미지링크, 제품명, 제품가격, 제품구매처링크를 각각 리스트로 만들어 자바스크립트에서 받도록 기능을 구현하였습니다.

2024-11-08

기능 추가보다는 코드의 가독성을 증가시키기 위해 코드를 수정하였습니다. 그리고 다른 사람이 이해하기 편하게 하기위해 주석문을 추가, 수정하였습니다.

2024-11-09

메이크업 페이지에서 카메라 연결 동작이 안되는 오류를 수정하였습니다. 원인을 찾지는 못했지만 makeupvariable.js 파일을 찾지 못해서 발생하는 오류로 예상되며 해결방법은 새로운 파일을 생성하였습니다.

2024-11-10

Makeup.js에서 작성한 코드로 웹 페이지에서 카메라 연결을 담당하는 코드인 cunnectWebsocket();의 동작은 알 수 없는 이유로 한 번만 실행 시 완전히 실행되지 않고 다시 한번 더 실행해야 정상 작동하는 오류가 있어 페이지 실행 시 함수가 한 번 동작하게 한 뒤 필요할 때 호출되도록 하였습니다.

그러다가 sendConnectPython(); 함수가 먼저 실행되고 cunnectWebsocket(); 함수가 실행되면 오류가 수정될 것이라고 생각하였습니다. FastAPI 서버가 동작하지 않은 상태에서 웹 소켓 함수를 동작하므로 웹 소켓 함수는 서버를 실행하는 동작도 함께 진행되어 완전히 연결이 진행되지 않고 이후 작성된 함수로 완전 동작하는 것이라 예상하여 두 함수의 순서를 변경하여 서버를 먼저 동작하고 웹 소켓을 연결을 진행하였습니다.

수정하기 전에는 정상 작동하지 않던 카메라 동작 기능이 정상 작동하게 되었습니다.

2024-11-11

FastAPI 서버를 실행하는데 상대 경로로 지정되었기 때문에 폴더 열기 위치가 달라지면 실행되지 않는 문제가 발생했습니다.

System.getProperty("user.dir")를 사용해서 상대 경로를 절대 경로로 변경하여 어느 위치에서 파일을 실행 시켜도 동작할 수 있도록 변경하였습니다.

2024-11-13

여러 시도한 끝에 알아낸 문제로는 스프링부트에서 process.start(); 부분이 동작하지만 이후 자바스크립트로 반환할 때는 파이썬 코드가 종료되었기 때문에 자바스크립트에서 웹 캠을 사용할 수 없는 문제가 발생했습니다.

이에 대한 해결 방법으로는 스프링부트 서버를 실행할 때처럼 통합개발환경에서 따로 실행하는 방식으로 해결해야 하는것으로 예상됩니다.

2024-11-14

컴퓨터 성능 문제가 발생할 수 있어 FastAPI 서버를 makeup.html에서 웹 캠 연결할 때만 동작하도록 설계하였습니다. 하지만 Process 와 Builder를 사용해서 FastAPI 서버 실행 동작을 하면 서버가 정상적으로 작동하지 않아 사이트 기능이 정상 작동하지 않는 문제가 발생하였습니다. 그래서 컴퓨터 성능을 상관하지 않고 FastAPI 서버가 동작하도록 변경하기로 회의 하였습니다.

2024-11-15

프로젝트가 대부분 완료되었기 때문에 전체적으로 모든 기능을 오류점검을 하였고 오류가 발생한 부분(게시판 이미지 업로드, 회원 찾기, 로그인 페이지에서 회원 찾기) 등에 대한 오류 수정을 하였습니다.

추가한 기능으로는 사용자가 화장 정보를 저장하면 저장된 화장 정보에 맞게 화장 추천 페이지에서 같은 색상의 화장품을 추천하도록 변경하였습니다.

2024-11-16

메인 화면을 디자인하기 위해서 test.py에 미리 준비된 영상을 화장하도록 작성된 코드를 붙여놓았습니다.

2024-11-17

Test.py 에 작성된 링크를 사용자 카메라가 아닌 미리 준비해둔 영상으로 메이크업 기능을 적용하였습니다.

2024-11-18

메이크업 과정이 녹화된 영상으로 메인 화면에 사이트에 대한 설명을 작성하였습니다. 제작한 사이트는 네이버, 구글 같이 모두가 아는 사이트가 아닌 새로운 기능을 갖춘 사이트이기 때문에 사용자들에게 사용 방법을 알려주는 설명글을 작성하였습니다.

2024-11-19

메인 페이지는 사용자에게 정보를 주기 위한 페이지이므로 다음과 같이 제작하였습니다.

맨 위에는 메뉴바를 작성하여 사용자가 웹 페이지를 이동할 수 있게 작성하였고, 그 밑은 이미지와 로고를 작성하여 사용자로 하여금 어떤 페이지 인지 한눈에 알 수 있게 작성하였습니다. 그러고 나서 사이트의 기능을 소개하고 해당 기능으로 이동할 수 있는 부분과 화장 미리보기 기능은 사용자에게 생소하기 때문에 해당 기능을 설명하는 부분으로 작성하였습니다.

2024-11-20

프로젝트 완료 전 마지막 점검을 하였고 발생된 오류들을 수정하였습니다. 그리고 메인 페이지 비디오 크기 조절과 설명글을 수정하였습니다.

2024-11-21

프로젝트가 최종 완료되었고 포트폴리오 제작과 문서 수정 위주로 깃허브 수정할 예정입니다.