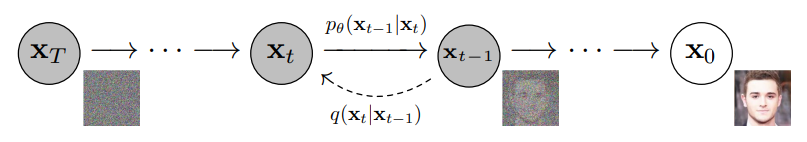
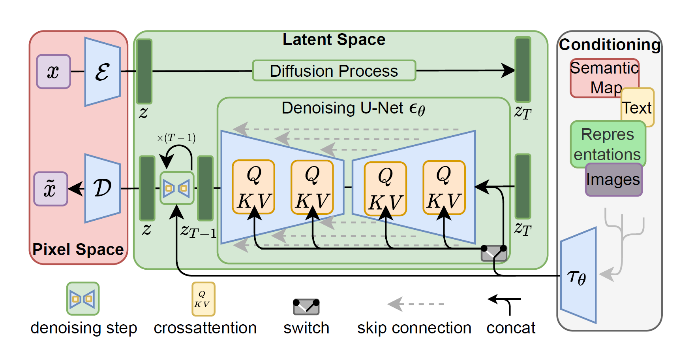
AI 설계

1. Model Architecture

Deep generative model 중 하나인 diffusion model을 사용한다. Data로부터 noise를 조금씩 더해가면서 data를 완전한 noise로 만드는 forward process와 이와 반대로 noise로부터 조금씩 복원해가면서 data를 만들어내는 reverse process를 활용한다.



하지만 diffusion model은 이미지 크기만큼의 차원에서 반복적인 denoising process를 진행하기 때문에 생성 속도가 느리고 cost가 많이 든다는 단점이 존재한다. 따라서 본 프로젝트에서는 이러한 단점들을 보안한 latent diffusion model을 사용한다. 

Latent diffusion model도 마찬가지로 noise에서 복원하는 과정은 똑같지만 image를 바로 복원하는 것이 아닌, VAE를 활용하여 image를 latent space에 mapping시킨 후 복원한다. 고해상도의 이미지를 빠르고 적은 cost로 생성하기에 프로젝트에 더 적합하다.

Latent diffusion model 중에서도, 오픈 소스로 공개된 stable diffusion(SD 1.5) model을 사용할 것이다.

Latent diffusion model은 다음의 구성을 가진다.

* Text\_encoder : prompt의 벡터화
* Tokenizer : prompt의 실수화 (토큰화)
* Unet : denoising architecture
* Scheduler : denoising method
* VAE : latent space에 mapping

본 프로젝트에서는 Text\_encoder, Tokenizer, Unet은 stable diffusion model이 제공하는 대로 사용하고, Scheduler는 k\_dpmpp\_2로, VAE는 kl-f8-anime2 & SD-vae로 고정한다.

1. 텍스트, 화면, 스크린샷, 다크이(가) 표시된 사진

   자동 생성된 설명Main process 1.

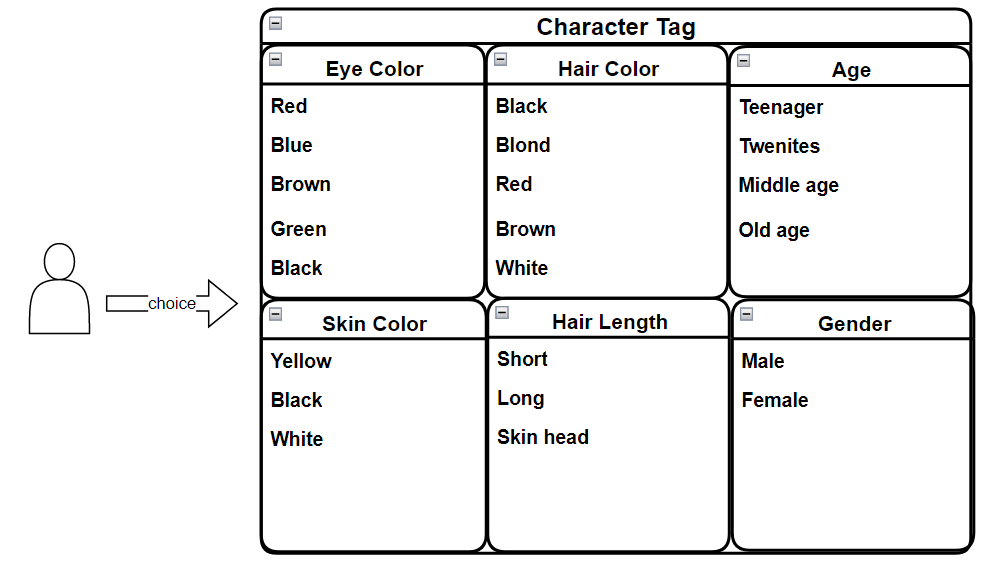
AI의 main process는 크게 두 가지이다.

첫 번째, main process는 등장인물 sample을 뽑는 과정이다.

작가는 등장인물을 고르기 위해 다양한 태그로써

등장인물을 정의하고 해당하는 등장인물 sample 4장을

얻을 수 있다. Ai model은 civitai에서 제공하는 model을 사용하고 이때 이용하는 generation toolkit으로써는 Invokeai를 사용한다.

1. Character Tag

작가가 생각한 등장 인물

등장인물을 정의하는 태그이다. 생김새를 정의하는 부분이기 때문에 행동이나 배경 과 같이 동적이고 등장인물과 상관없는 태그는 제외한다. AI model의 input으로 들어간다.

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명Ex)

1. CivitAI

모델을 제공하는 사이트이다. Model type으로는

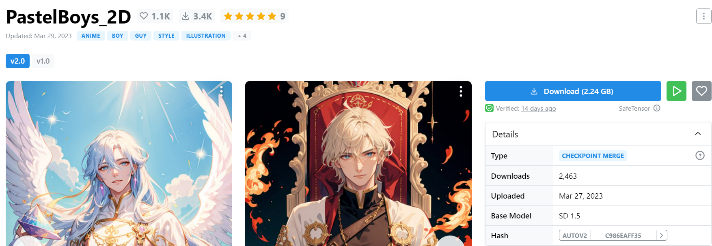
* 1. Checkpoint
  2. Textual Inversion
  3. Hypernetwork
  4. Aesthetic Gradient
  5. LoRA
  6. LyCORIS
  7. Controlnet
  8. Poses
  9. Wildcards
  10. Other

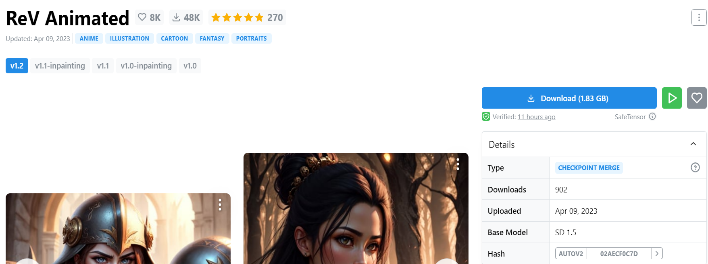
이 존재하고, 본 프로젝트에서는 checkpoint model만 사용하고 확장자는 .ckpt or .safetensor or .pt만 사용한다. (그 중에서도 base model이 SD1.5인 model만 사용.)

이유 – 추후에 Textual Inversion을 embedding으로 적용시키기 위함이다. SD1.5에만 적용이 가능.

등장인물의 다양성과 작화를 고려하여 2개의 모델을 적용시킨다.

* anime style의 model



* Midjourney style의 model

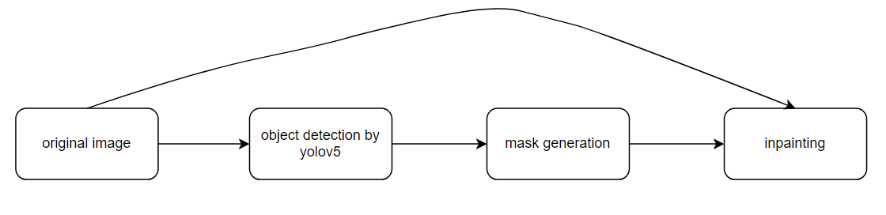
1. Textual Inversion

기본적으로 image generation model은 사람에 대해서 image를 생성 시, 손이나 얼굴의 뭉개짐이 존재한다. 이를 해결하기 위해 negative prompt로써 pretrained textual inversion token을 적용시킨다. Pre-trained embedding token은 현재 SD1.5에서만 적용 되기 때문에 CivitAI model 중

Base model이 SD1.5인 것만

사용.

1. Auto – inpainting

Object의 손이나 얼굴 뭉개짐을 해결하기 위한 두 번째 수단으로, image를 생성할 때 inpainting을 자동으로 적용시키는 것이다.

Process는 다음과 같다.

Original image를 yolo model을 이용하여 얼굴이나 손을 탐지 -> 해당 부분을 masking -> inpainting

이때 기존의 pre-trained yolo model은 CivitAI에서 제공하는 model이 생성하는 image를 탐지하기가 어려운 문제가 존재한다. (yolo는 실사 이미지에 대해서 학습) 따라서 추가로 학습을 진행해준다.

<training>

Object detection model : yolov5

Training image : midjourney style의 image 100장

Labeling : roboflow를 이용

input :

1. Image size : 512
2. Batch size : 16
3. Epoch : 50
4. Image : 100장

<inference>

input :

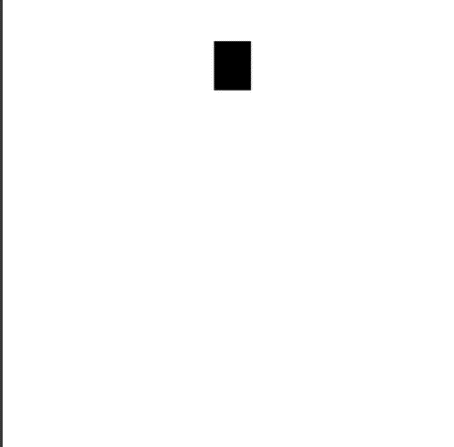
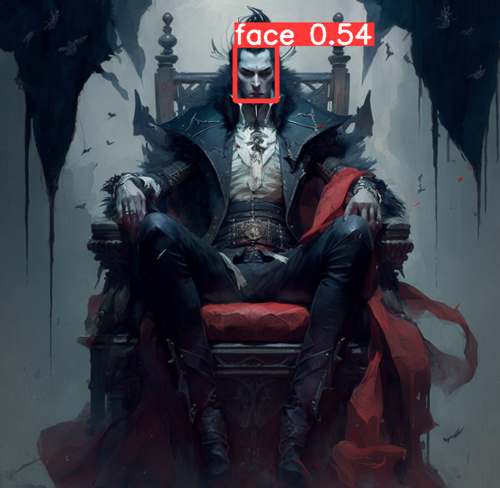
1. Image size : 512
2. Confidence : 0.3
3. Detection 하고자 하는 Image 한 장

<inpainting>

input :

1. prompt : 원본 이미지의 prompt
2. negative\_prompt : 원본 이미지의 negative prompt
3. num\_images\_per\_prompt : 4 (recommend)
4. num\_inference\_steps : 50 (recommend)
5. guidance\_scale : 7~8 (recommend)
6. image : 원본 이미지
7. mask\_image : masking image
8. generator : random seed value
9. strength : 3.0 ~ 8.0 (recommend)
10. model : runwayml/stable-diffusion-inpainting (huggingface)

<inpainting example>



- face detection 후 masking 과정

 -masking을 통해 angry로 변환

1. Invokeai

Generation toolkit이다. civitai에서 제공하는 model을 저장하여 간편하게 image를 생성할 수 있다.

* GUI를 command로 실행시키기 위해

자체적인 python code 필요

Python script로 제공, prompt가 들어오면 InvokeAI를 실행시켜 image를 생성.

1. Image processing

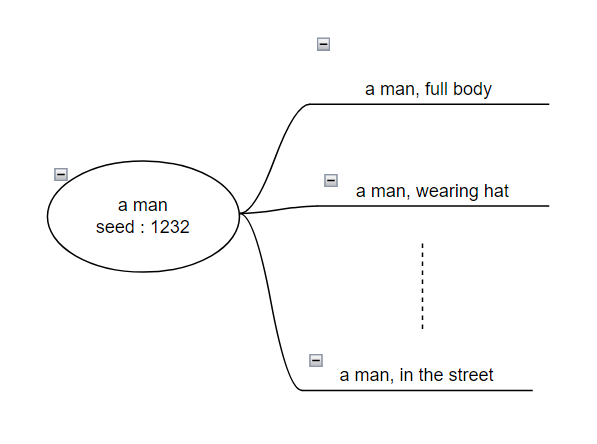
등장인물 sample을 뽑기 위한 image는 user id와 동일한 directory에 저장 후, 웹 플랫폼에 response를 마치면 삭제한다.

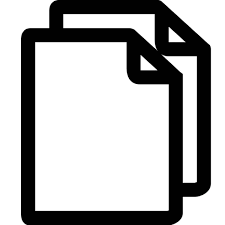
1. Main process 2

두 번째 main process이다. 작가가 선택한 sample을 dreambooth 기법을 적용시켜 개인화 시켜주는 과정이다. Base model은 openjourney를 사용. 한 model에 대해서 다양한 instance prompt를 학습한다. User가 선택한 등장인물의 prompt와 seed값을 사용.

1. Sample 복제

퀄리티가 좋은 image를 뽑기 위해서, sample을 다양한 각도, 표정, 배경에 대하여 복제한다. 복제 시, 사용하는 model은 CivitAI에서 선정한 model을 사용. Request로 받은 해당 sample의 seed value, prompt를 바탕으로 20장을 복제한다. 이때 복제용 prompt는 기존의 이미지 prompt + seed + 다양한 상황 묘사를 기반으로 제공한다. 복제한 20장의 사진은 Dreambooth instance image path에 저장.

Ex) a man, (seed : 123241)



다양한 각도, 표정, 배경에 대해서 복제

1. 복제한 이미지를 dreambooth 기법 적용

<dreambooth>



Dreambooth는 fine-tuning 기법이다. 몇 장의 간단한 사진만으로도 object의 시각적 특징에 대한 높은 충실도를 유지하면서 새로운 맥락의 사진을 제공한다. dreambooth에도 base model이 필요한데 이 model은 stable diffusion model을 재학습한 openjourney model을 사용.

<dreambooth training>

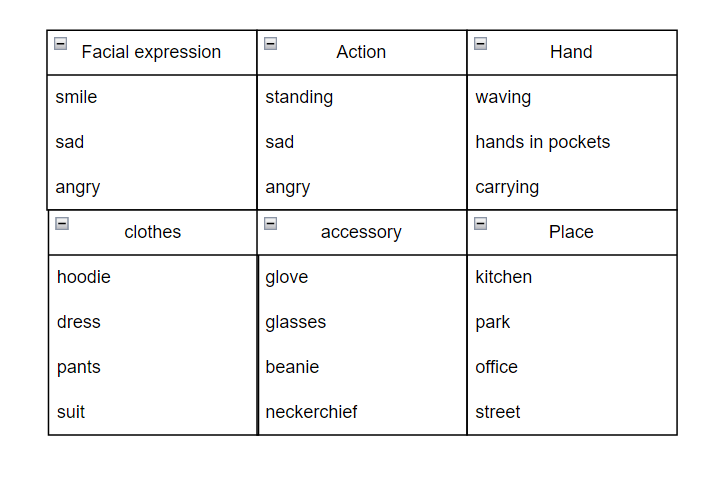
Input :

1. Scheduler: k\_dpmpp\_2
2. VAE : stabilityai/sd-vae-ft-mse or kl-f8-anime2
3. Instance\_prompt : zwx woman
4. Class\_prompt: woman
5. prior\_loss\_weight=1.0
6. learning\_rate=1e-6
7. lr\_scheduler="constant”
8. num\_class\_images=49 (mid journey style의 얼굴 사진 49장)
9. max\_train\_steps= instance image X 100
10. save\_sample\_prompt="zwx woman"

<dreambooth inference>

1. 이미지는 4장씩 출력
2. 이미지 사이즈: 512\*512 (default)
3. Num\_Inference\_step : 50
4. Prompt : 작가가 선택한 Tag
5. Seed : random value
6. Guidance\_scale : 6~8 (recommend)
7. Illustration Tag

Character tag와는 다르게 등장인물을 선정한 뒤, 그 등장인물을 삽화로써 사용하고자 할 때 쓰는 tag이다. 따라서 동적인 태그가 들어간다.



1. Instance prompt

Instance prompt는 특정 등장인물을 한 단어로 정의한 것과 동일하다. 따라서 unique한 prompt를 제공해야 한다.

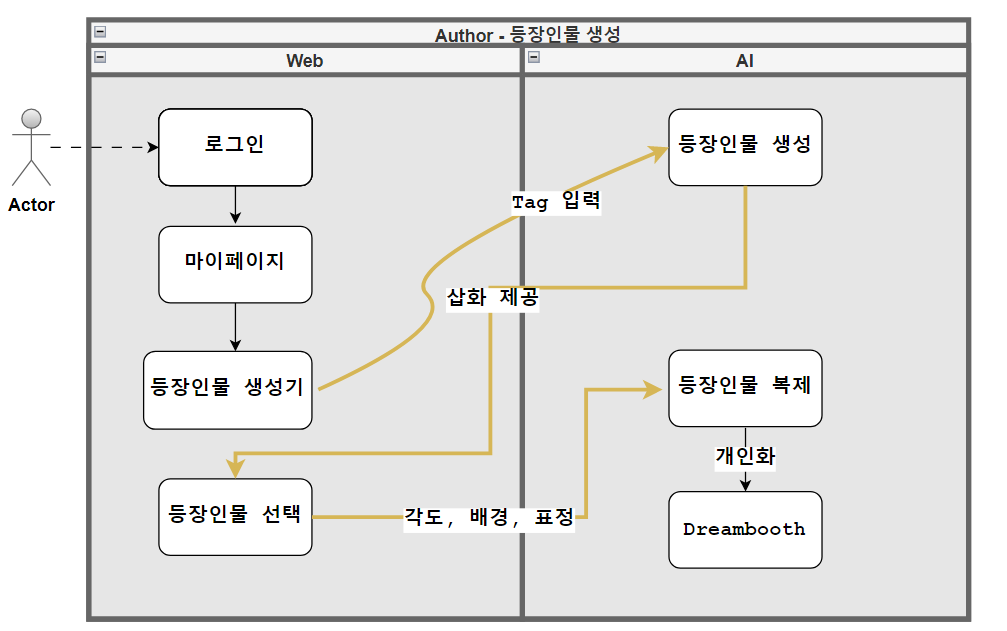
Ex) zwx, ckj, abs (단어의 길이는 상관없지만 본 프로젝트에서는 length = 3으로 제한한다)

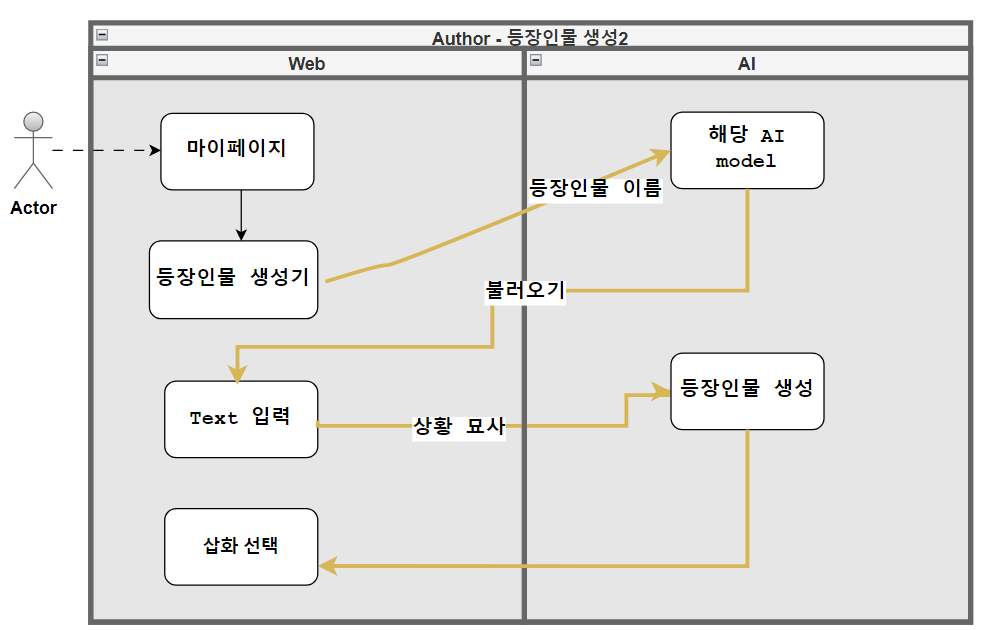
1. Image processing

Dreambooth instance image path : /instance\_image/{instance\_name}

Dreambooth class image path : /class\_image (pre-setting한 49장)

* Main process를 종합적으로 보면 다음과 같다.





1. Extension
2. Upscaling

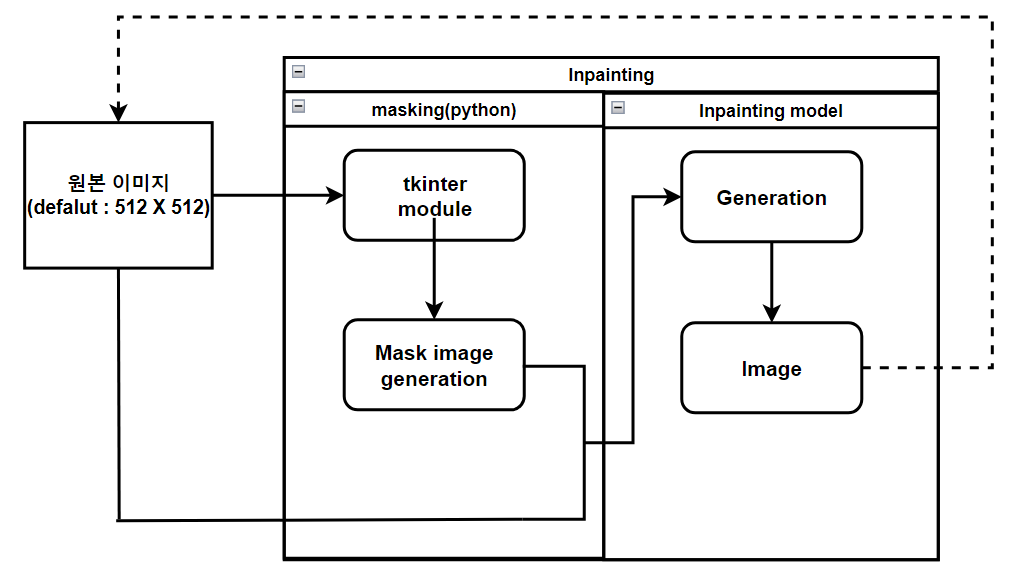
Dreambooth model은 기본적으로 512x512 image size가 default이다. 하지만 목적에 따라 작가가 원하는 image size가 다를 수 있기 때문에 다른 image size를 제공받을 수 있는 기능을 제공

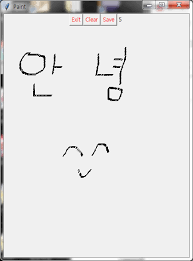
1. prompt : illustration tag
2. negative\_prompt : base로 제공
3. num\_images\_per\_prompt : 4 (recommend)
4. num\_inference\_steps : 50 (recommend)
5. guidance\_scale : 7~8 (recommend)
6. generator : random seed value
7. model : dreambooth model
8. Inpainting

자동으로 inpainting 되는 것이 아닌, 작가가 원하는 부분에 직접 mask를 씌워서 inpainting을 시킬 수 있는 기능.

1. prompt : 원본 이미지의 prompt
2. negative\_prompt : 원본 이미지의 negative prompt
3. num\_images\_per\_prompt : 4 (recommend)
4. num\_inference\_steps : 50 (recommend)
5. guidance\_scale : 7~8 (recommend)
6. image : 원본 이미지
7. mask\_image : masking image
8. generator : random seed value
9. strength : 3.0 ~ 8.0 (recommend)
10. model : runwayml/stable-diffusion-inpainting (huggingface)

**<Tkinter을 사용한 그림판>**

* process



1. Interface

본 프로젝트에서 AI 구동과 Web platform 구동은 같은 곳에서 작동하지 않는다. AI model을 돌리는데 드는 cost가 비싸서 GPU 사양을 맞추기 위함이다.

또한 ai model은 python으로 돌아가고 web platform은 java로 돌아가기 때문에 독립적인 시스템을 연결해야 한다.

* **Protocol : HTTP method [ POST, GET ]**
* **Language : AI -> python | Web server -> Java**
* **Framework : python -> flask + ngrok**

**Java -> spring**

* **Data : JSON file (image, text etc.)**

**JSON format으로 보낼 데이터는 image와 text이다. Image는 바이트 형태로 읽어서 전송, text는 string형태로 전송한다.**

* **Library : python -> OS (path), flask(web), torch(model), gc(memory), subprocess(script), requests(interface), multiprocessing(thread)**

**Flask의 경우 들어오는 경로에 따라 제공하는 기능을 다르게 준다.**

**/txt2img : prompt를 제공받으면 image 제공**

**/upscaling : 1024 x 1024 image 제공**

**/16\_9 : 512 x 900 image 제공**

**/inpainting : original image와 mask image를 받아서 inpainting**

**또한, 생성된 순서에 상관없이 user가 요청한 사진을 제공해야 한다**

User의 이름은 unique하므로 user의 이름으로 된 directory 생성 후 그 안에 image를 생성하여 byte 형태로 JSON file에 담아서 response.

다수의 user request를 고려 : multi-threading을 통해 동시에 image를 제공한다.

* Process – character sampling

1. Author가 생각하는 등장인물에 맞는 Tag를 선택하여 request ( Tag & user id)
2. User id의 directory를 만든다.
3. Tag를 바탕으로 CivitAI model을 통해 image 4장 생성 (request한 경로에 따라 image size가 다르게 생성)
4. 해당 directory에 저장
5. Image는 byte 형식으로, seed 값과 함께 response

* Process – training

1. Author가 등장인물 한 명 선택
2. Prompt + seed + user id와 함께 request
3. Dreambooth training
4. 학습 완료 후, 완료됐다는 response

* Process – Illustration

Author가 등장인물의 Tag 선택

Tag + prompt + user id + 등장인물 정보와 함께 request

등장인물 정보 + user id로 instance prompt 유추

Tag를 통한 image 생성

Byte 형태로 담아 4장의 Image response

1. 윤리적 고려 사항

SD(stable diffusion) 1.5 뿐만 아니라 CivitAI에서 제공하는 model들은 일부분 선정적인 image로 학습되었기 때문에, 목적에 맞지 않는 사진이 나올 가능성이 존재한다. 따라서 model을 선정하는 부분에 있어서 목적에 맞게 고려해야 한다.

1. Nsfw checker

* InvokeAI에서 제공하는 검열 tool을 사용

1. negative prompt

* negative prompt를 기본 prompt로 제공하여 미연에 방지

1. model

* 선정적인 이미지가 적은 data set으로 학습된 model 사용

1. 제약 사항
2. Time-out

서버 간의 time-out이 1분으로 설정되어 있다. 일반적으로 이미지 생성의 시간은 1분내외이기 때문에 전송이 실패할 수 있다. 서버 자체의 제한이기 때문에, 자유도가 높은 서버를 구축하면 해결이 가능하다.

1. Auto-Inpainting

Auto-inpainting은 image generation과 동시에 적용되는 기능이다. 필연적으로 서버 응답시간이 길어질 수밖에 없는데, 이로 인해 time-out이 일어날 수 있고 사용자가 시간을 더 기다려야 한다. 자유도가 높은 서버 구축과 기다리는 시간 동안 다른 작업을 유지할 수 있다면 해결.

1. License

Stable diffusion 자체는 오픈된 model이지만, CivitAI에서 제공하는 model은 license가 걸려있는 model이 대부분이다. 따라서 상업적으로 이용하는 것에 대한 고려사항이

있지만 시간이 충분했다면 직접 model을 제작함으로써 해결할 수 있다.

1. 손, 발, 얼굴

기술의 발전으로 인해, 고화질의 image를 뽑을 수 있지만 여전히 손, 발, 얼굴의 생성이 힘들다. 이를 해결하기 위한 방법으로 textual inversion, LoRA, multi prompt 등이 존재하지만 시간 부족으로 인해, 이 모든 걸 해볼 순 없다.

1. 명명 규칙

<Html,Jsp,Css>

명명 규칙

○ 각 요소의 Id는 중복되지 않도록 설정한다.

○ 각 요소의 Id는 camel case를 원칙으로 하되 의미를 구분해야하는 경우에는 언더바( \_ )를 사용하여 구분하도록 권장한다.

○ css class명은 하이픈 ( - )을 사용하여 구분하는 것을 원칙으로 한다.

○ jsp 파일의 기본 경로는 [web-root]/[프로젝트명]을 기본으로 하며 기능별로 구분하여 디렉토리를 생성, 관리한다.

○ jsp 파일명은 camel case로 작성하는 것을 원칙으로 한다.

○ css이 위치하는 논리적 위치는 [web-root]/css를 기본으로 한다.

Coding Style

○ 스타일은 css를 사용한다.

Java script

명명 규칙

○ js가 위치하는 논리적 위치는 [web-root]/[프로젝트명]이며 jsp경로와 동일하게 사용할 것을 원칙으로 한다.

○ js 파일의 명명규칙은 [용어사전].js로 한다

○ 용어사전의 단어를 사용하는 경우 camel case에 의거해 작성한다. (ex. novCheckValid.js)

○ 변수명은 소문자로 시작하고 새로운 단어의 시작은 대문자를 사용한다. (ex. reqMsg)

Coding Style

○ 여러 sub-system에서 include 하는 js는 [web-root]/js에 위치한다.

JAVA

공통

java class file 은 필요 시 사업을 위한 wrapper를 생성하여 사용함으로써 객체에 대한 추상화와 구현 시의 유연성을 확보한다. Java class 작성 시에 해당 package, file 및 Method의 명명 규칙과 coding style을 준수하여 일관성 있는 개발이 되도록 한다.

명명 규칙

Java package

○ 기본 패키지 명명규칙은 다음과 같다.

○ [도메인].[회사명].[프로젝트]명을 기본으로 한다.

○ 소문자 사용을 원칙으로 한다.

○ 업무별로 나누어서 package 생성할 수 있도록 한다.

JAVA class

○ java file의 명명규칙은 Pascal Case를 따른다.

○class 사용시 약어 사용을 지향한다.

Interface class

○ Interface는 java 클래스명과 동일한 방법으로 부여한다.

Java method

○ java method의 명명규칙은 [동사][용어사전]로 한다.

※ 클래스 명명은 필수로 하되 메소드 명명규칙은 용어사전을 최대한 준용하도록 한다.

○ java method는 camel case를 따른다.

○ java method는 숫자 및 특수문자는 사용하지 않는다.

○ method에서 사용하는 동사는 다음 표와 같이 명명한다.

|  |  |
| --- | --- |
| 구분 | 규칙 |
| 화면로드 | loadpage |
| 조회 | select \* |
| 등록 | insert \* |
| 수정 | update \* |
| 삭제 | delete \* |
| 통합 | save \* |
| 가져오기 | get \* |
| 설정 | set \* |
| 다운로드 | download \* |
| 업로드 | upload \* |

Database

Table

명명규칙

* table명은 snake case를 따른다.
* table명은 소문자를 사용한다.
* table명은 영문자와 숫자만을 사용하여 명명한다.
* table명은 약어 사용을 지양한다.
* table명은 데이터 타입을 이름으로 작성하지 않는다
* 최초 3자리는 업무 관련된 자리로 사용한다.

Column

명명규칙

* Column명은 예약어 사용을 지양한다.
* Column명은 테이블명 사용을 지양한다.
* Column명은 용어사전에 있는 단어 사용할 것을 권장한다.

기타

명명 규칙

○뷰- \*\_v

○트리거 – tg\_\*

○프로시저 – sp\_\*

○패키지 – pkg\_\*

○제약조건 – ct\_\*

○외래키명 – fk\_\*

SQL Query

Coding Style

○ SQL 예약어는 소문자를 사용하는 것을 원칙으로 한다.

○ SELECT, UPDATE시 컬럼 나열은 1줄당 1개씩 소문자로 작성을 기본으로 하며 컬럼이 많은 경우에 한하여서 1줄에 3개 컬럼까지 쓰는 것을 예외로 허용한다.

○ SELECT의 들여쓰기는 SELECT를 기준으로 FROM, WHERE, INTO 등의 시작점을 맞춥니다 단 GROUP BY나 ORDER BY는 SELECT(6) 자 보다 더 긴 관계로 BY 이후 스페이스 1칸을 띄우고 관련내용을 기술한다.

○ FROM 절의 테이블명은 1줄당 1개씩 소문자로 작성하는 것을 기본으로 하며 테이블이 많은 경우에 한하여서 1줄당 2개의 테이블까지 적는 것을 예외로 허용한다.

○ WHERE절의 조건은 1줄당 1개의 조건씩 작성한다

○ INSERT문장의 경우 전체 컬럼에 대해서 insert를 하더라도 컬럼명을 반드시 적어준다.

○ DML의 들여쓰기는 UPDATE, DELETE, INSERT를 기준으로 SET, FROM, WHERE 등의 시작점을 맞추고, 컬럼 구분은 가장 길이가 긴 컬럼의 쉼표 나 등호 등을 기준으로 다른 컬럼을 맞춘다.

○ 괄호를 사용할 경우 즉 인라인 뷰 스칼라 서브쿼리 등을 사용할 경우 괄호가 존재하는, 라인은 라인을 변경하지 않는다.