CONNECTED - 7
20190198 왕현성
20190325 장다연
20191820 김형준
20200152 김규빈

Sketch Web application

DEMO PRESENTATION

왕현성

Design / AI

김형준

Frontend

장다연

Database

김규빈

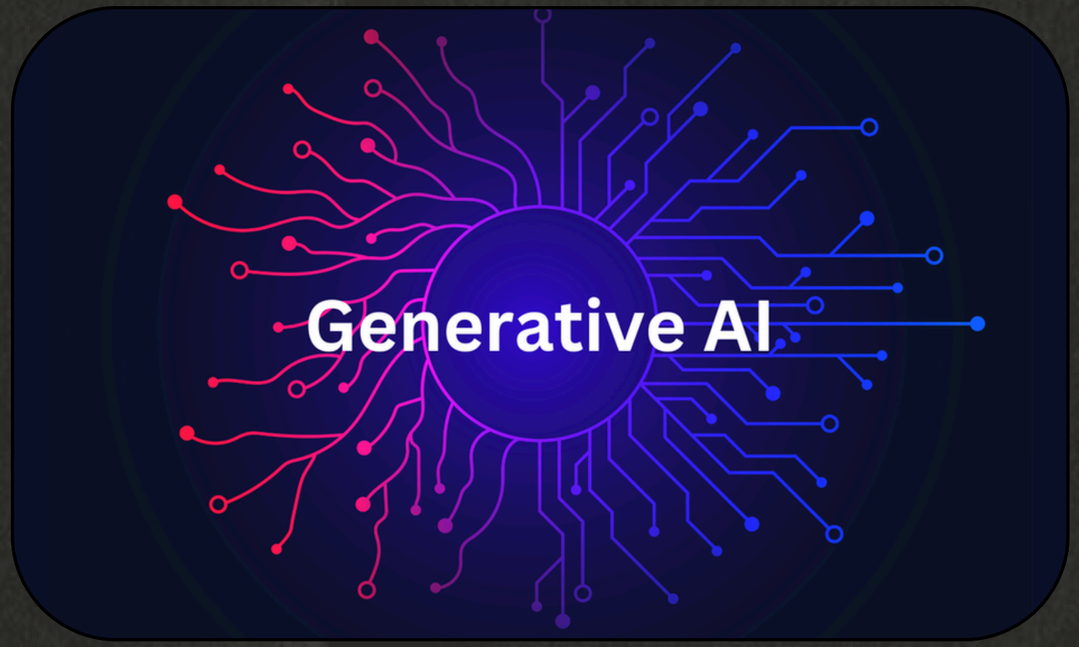
Backend

DEVELOPMENT GOAL

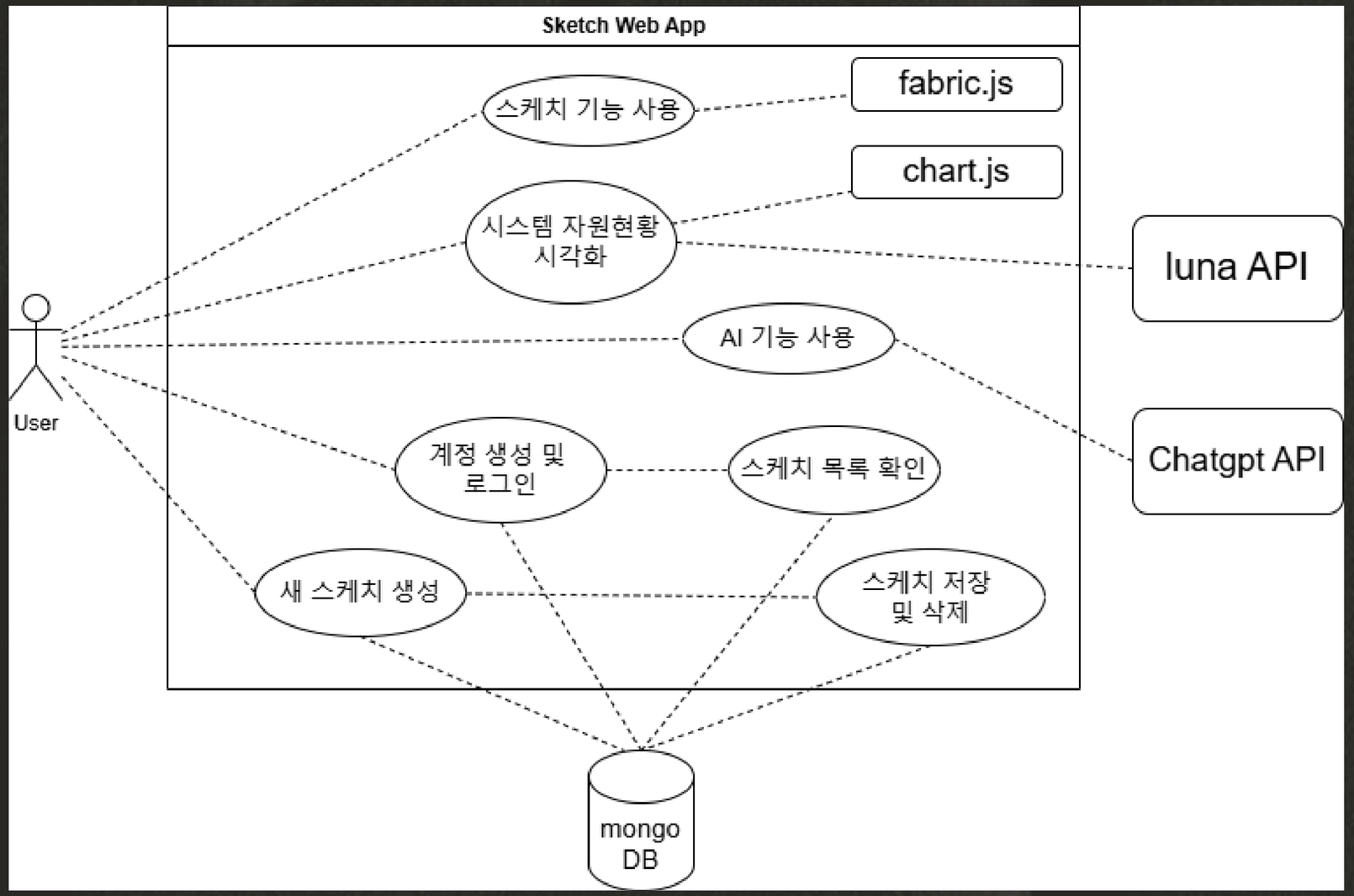
누가 언제 사용할까?

DEVELOPMENT GOAL

어떤 기능들이 필요할까?



USECASE DIAGRAM



DATABASE

- 사용 기술:
  - DB: MongoDB
  - ODM: Mongoose
  - 클라우드 저장소: AWS S3
- 연결 방식:
  - .env 파일을 활용하여 MongoDB URI, AWS 키 등 환경 변수 관리
  - mongoose.connect()로 데이터베이스 연결 설정
- MongoDB를 선택한 이유
  - NoSQL
  - 빠른 개발 속도
  - 파일 저장 및 관리
  - 클라우드 친화적:

DATABASE

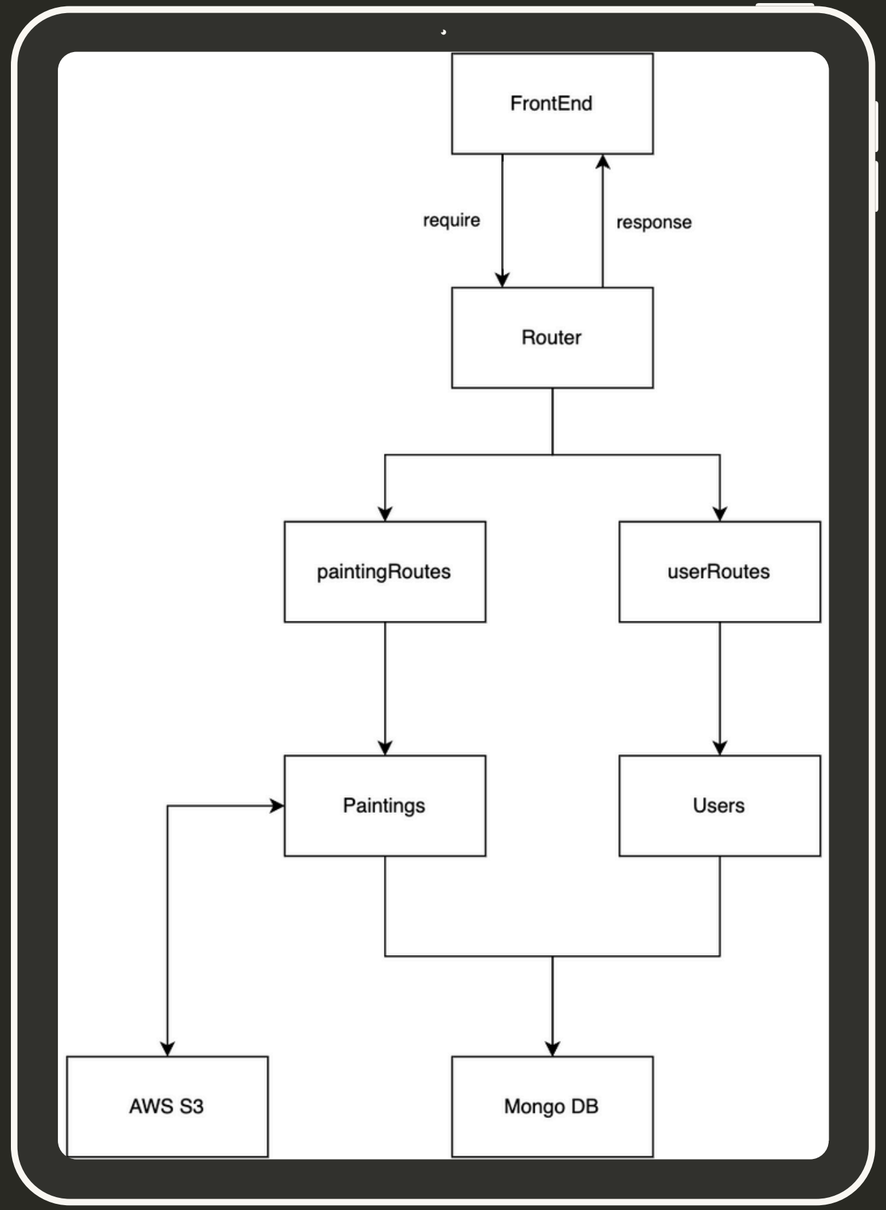
- 스키마 설계
  - Users Schema

```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true }, // 사용자 이름
  password: { type: String, required: true }, // 비밀번호 (해싱된 값)
  createdAt: { type: Date, default: Date.now }, // 계정 생성 날짜
  updatedAt: { type: Date, default: Date.now }, // 계정 수정 날짜
  mode: { type: String, default: 0 } // 키즈모드는 1
});
```

  - Paintings Schema

```
const paintingSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }, // 사용자 ID
  title: { type: String, required: true }, // 그림 제목
  description: { type: String }, // 그림 설명 (선택)
  summary: { type: String }, // 그림 요약 (선택)
  text: { type: String }, // 텍스트 추출 (선택)
  imageUrl: { type: String }, // URL로 이미지 경로 저장
  s3ImageUrl: {type: String }, // s3 URL 이미지 경로 저장
  preview: { type: String }, // URL로 미리보기 저장
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
});
```

BACKEND



ARCHITECTURE

1.Request 처리: Frontend에서 요청을 받아 Router가 paintingRoutes와 userRoutes로 분기 처리하며, 각각 Paintings 및 Users 데이터와 상호작용

2.데이터 저장 및 관리: Paintings와 Users는 MongoDB를 통해 사용자 및 그림 데이터를 관리
S3는 그림 및 텍스트 데이터를 저장하고 URL을 반환

3.Response 반환: 데이터를 AWS S3와 MongoDB에서 조회 및 처리한 후, Router를 통해 Frontend에 응답 전달

문제 해결

1. 구현 상황:

프론트엔드: 스크린샷을 png 파일로 저장하는 기능 구현

백엔드: png 파일을 S3에 업로드하고 URL을 반환하는 기능 구현

2. 문제점:

프론트엔드에서 생성된 png 파일을 서버로 전송 오류 발생

3. 해결 방안:

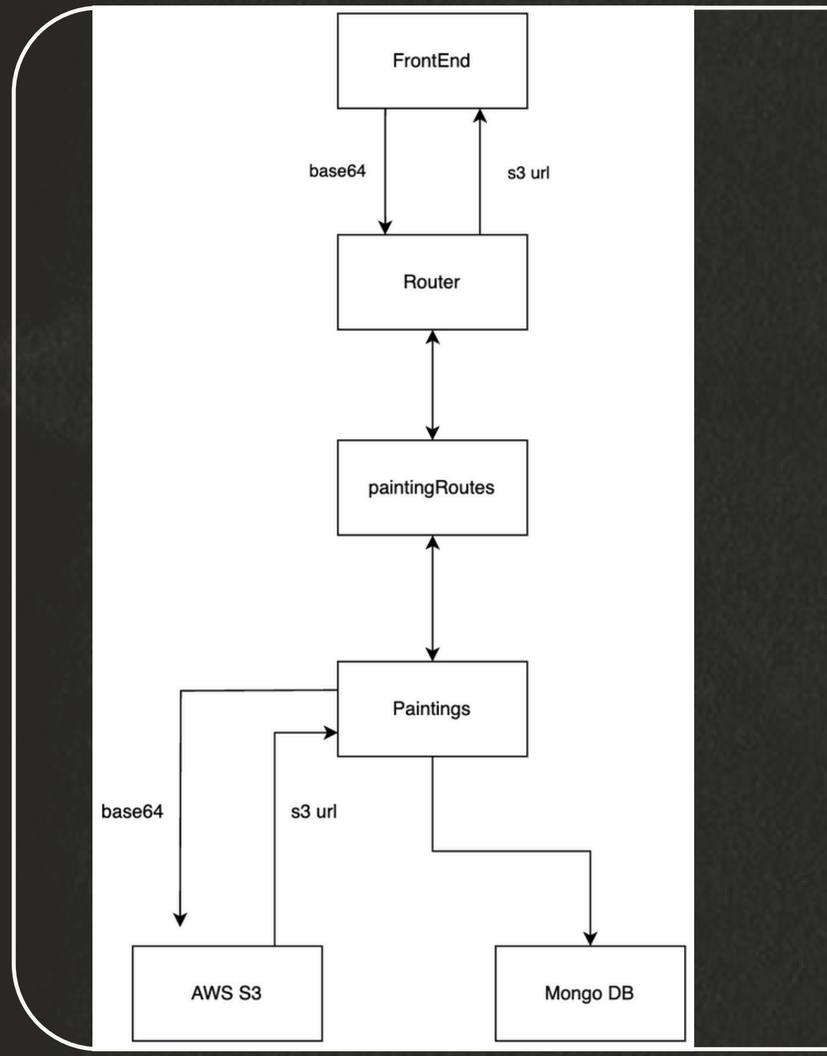
ToDataURL 사용

Base64 -> .txt 변환 및 S3 저장

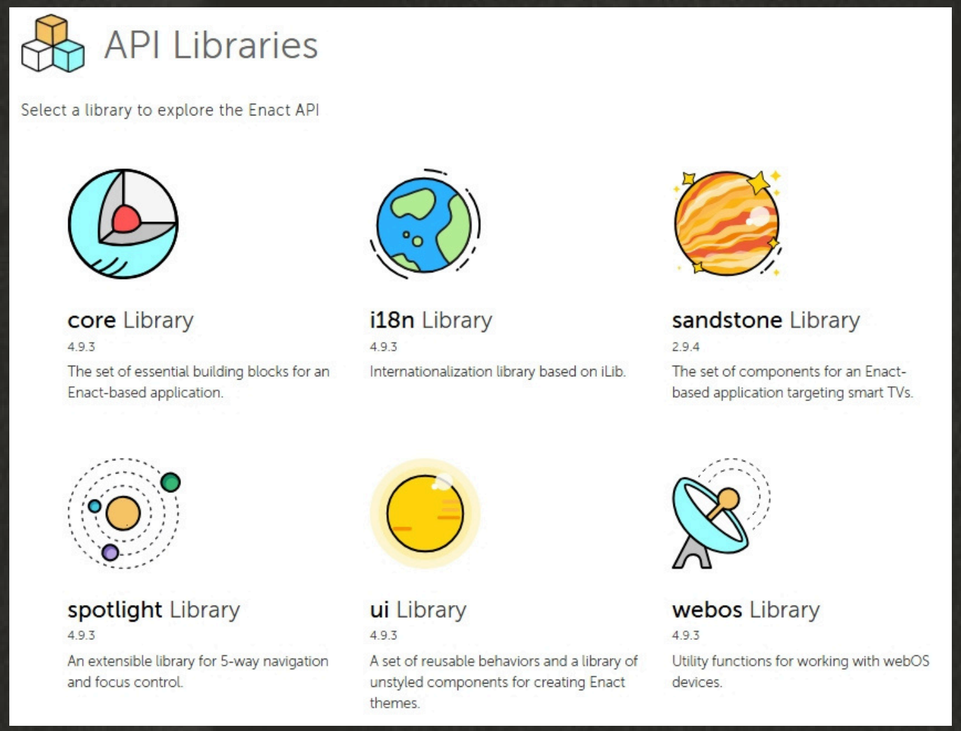
4. 적용:

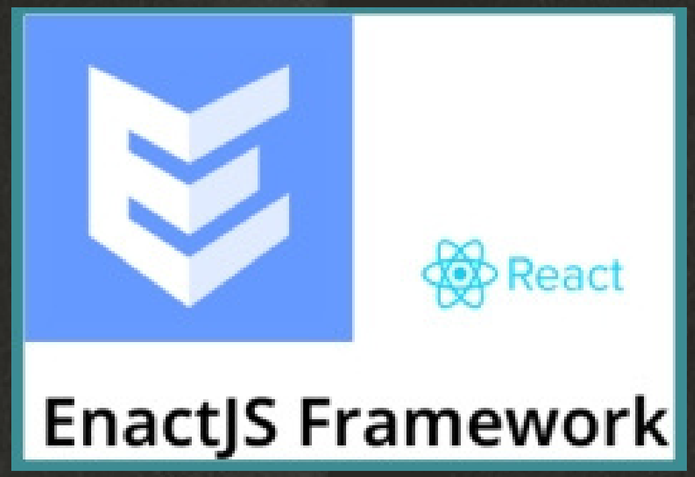
반환된 S3 URL을 AI 모델의 입력값으로 -> AI 기능 구현

URL을 통해 Base64 데이터 -> 미리보기로 활용



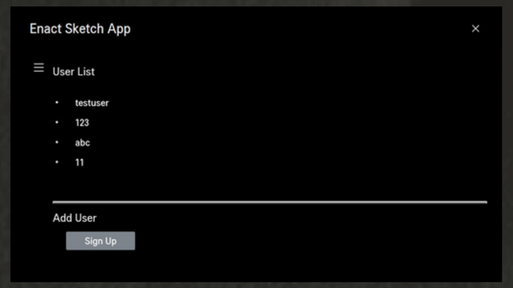
FRONTEND - FRAMEWORK



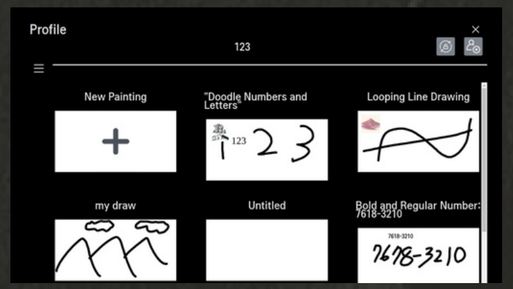


- 강의자료에 나온대로 Enact Framework와 Fabricjs Library를 사용함
- 각각의 UI는 EnactJs의 Api Library를 참고해 최대한 Enact 컴포넌트 위주로 구현함

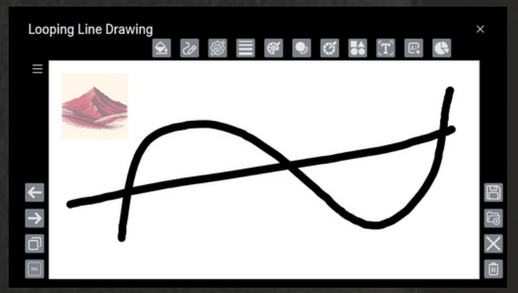
FRONTEND - APP 구조



- Account



- Profile

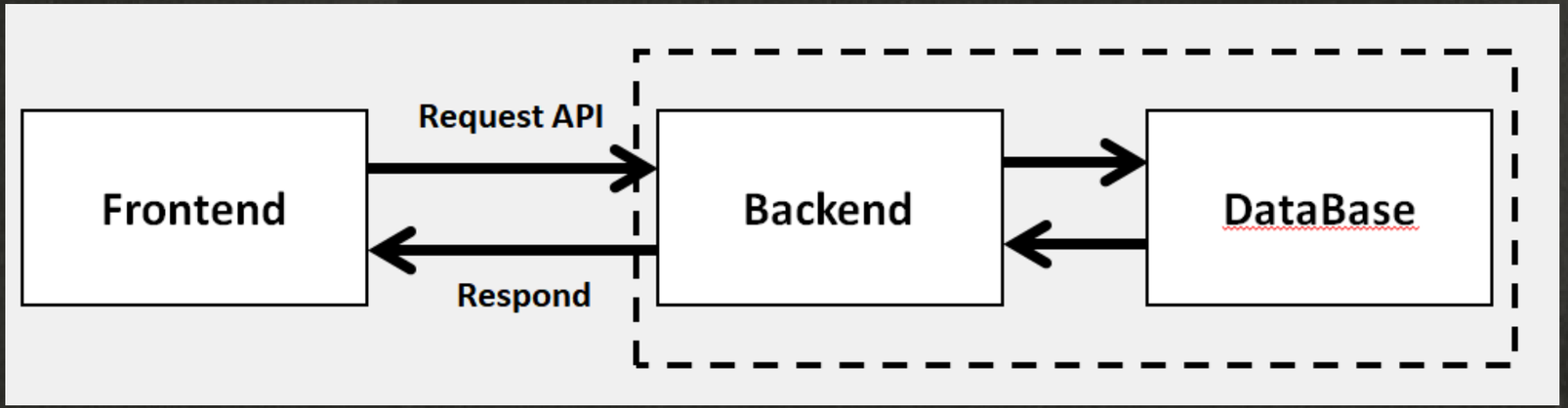


- Sketch

- System Status

- 전체적인 APP의 구조는 왼쪽과 같이 4개의 Page로 구성됨
- 각 Page에서는 정해진 버튼으로만 다른 페이지로 이동가능함
- 오른쪽 위의 X버튼을 눌러 이전 Page로 돌아갈 수 있음
- Main.js의 TapLayout 컴포넌트의 collapsed 속성을 true로 줘서 항상 Tap이 잠기게 설정함
- 사용자가 Tap버튼으로 임의의 Page로 이동하는 것을 차단하여 각각의 Tap을 분리된 Page로 제작함

FRONTEND - DB CONNECTION




```
const [curUser, setcurUser] = useState({
    _id: '',
    username: '',
    password: '',
    mode: '0'
});
```

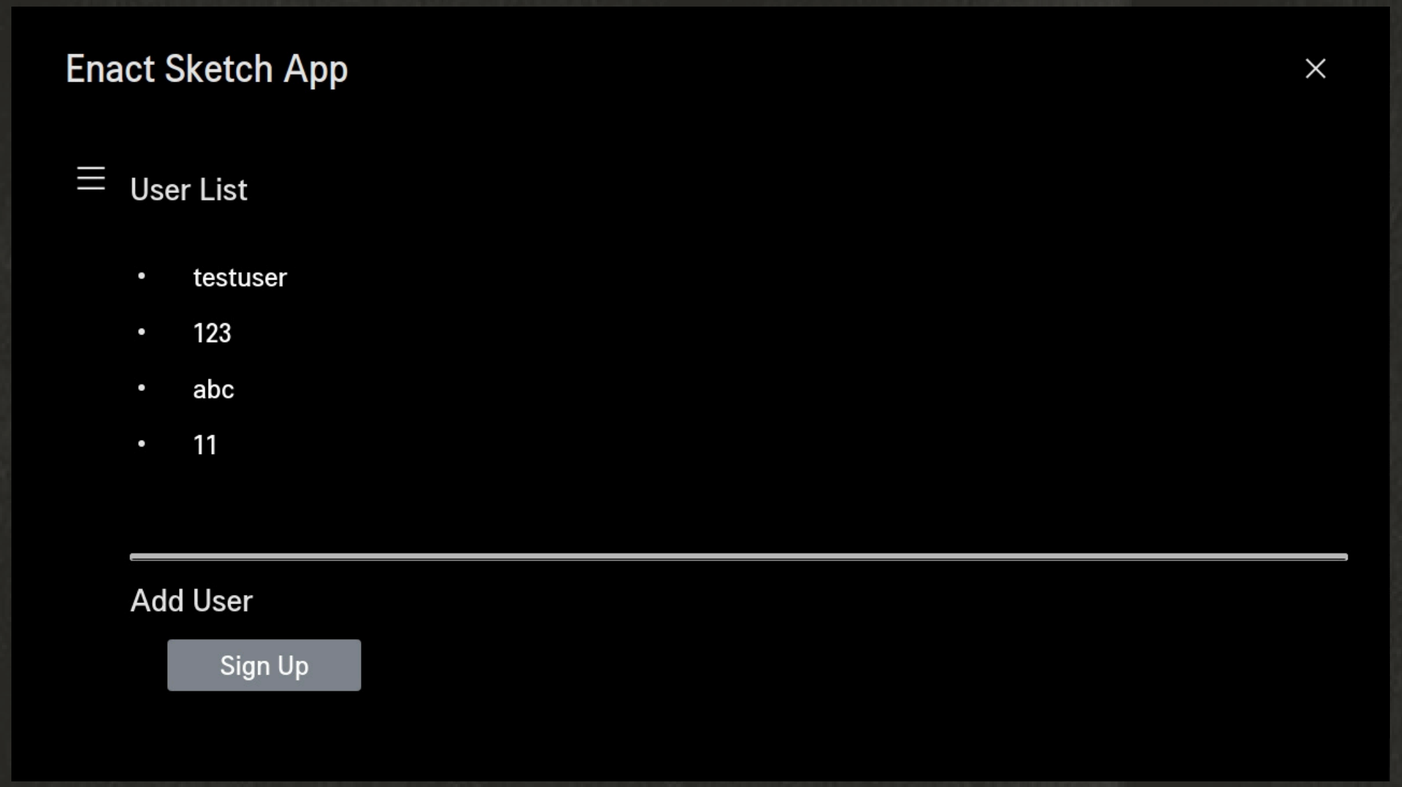
```
export const LoadPaintingbyUser = async (User) => {
    try {
        const response = await axios.get(process.env.REACT_APP_SERVER_URL+`/api/paintings/user/${User._id}`);
        console.log('>>>>>> RESPONSE: ', response.data);
        return response;
    } catch (error) {
        console.log(error);
    }
    return null;
};
```

```
const newPainting = await {
    userId: LoginUser._id,
    title: 'Untitled',
    description: "-",
    imageUrl: JSON.stringify(newCanvas),
    preview: newCanvas.toDataURL({
        format: 'jpg',
        quality: 0.8
    })
};
```

- 각 Page에서는 DB에서 정의한 Schema에 맞게 각각 User와 Painting Object를 정의해서 사용함

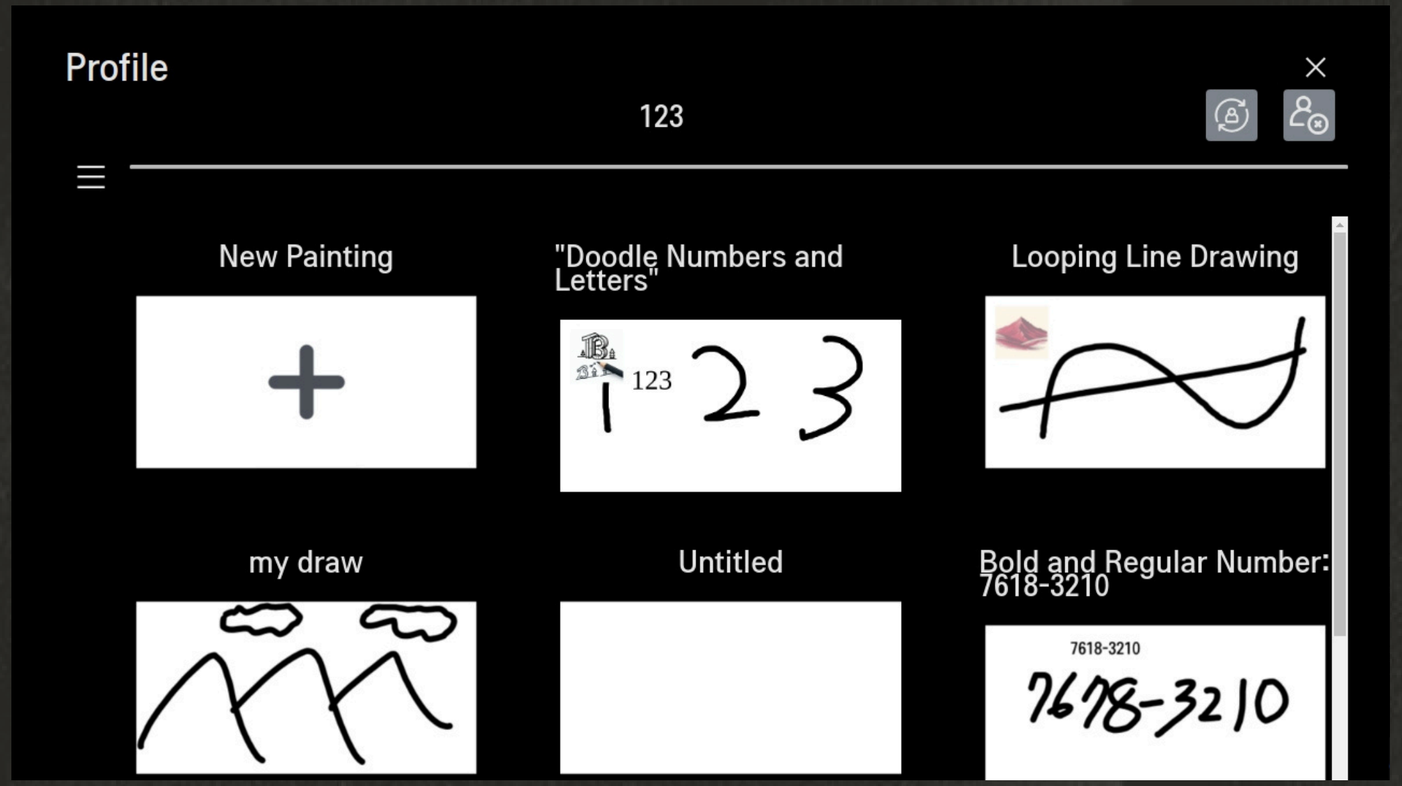
- Axios와 Async-await 문법으로 Backend의 API를 사용하여 DB의 CRUD작업을 수행함

FRONTEND - ACCOUNT



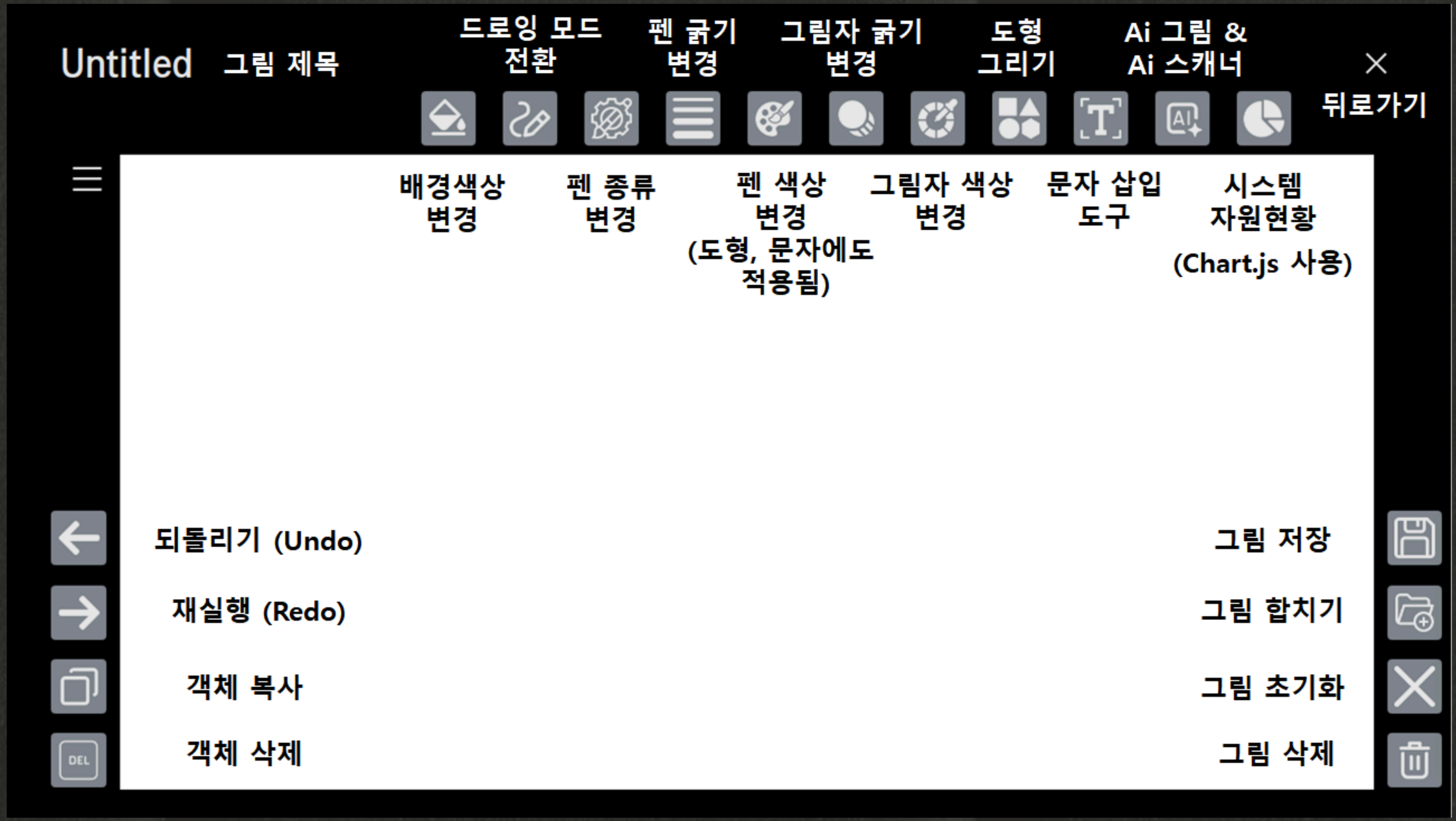
- Account Page에서는 현재 DB에 있는 모든 유저 목록을 보여주며, 클릭시 로그인창을 띄움
- 로그인 시, Profile Page로 이동해서 로그인한 유저의 Painting 목록을 미리보기와 함께 보여줌

FRONTEND - PROFILE

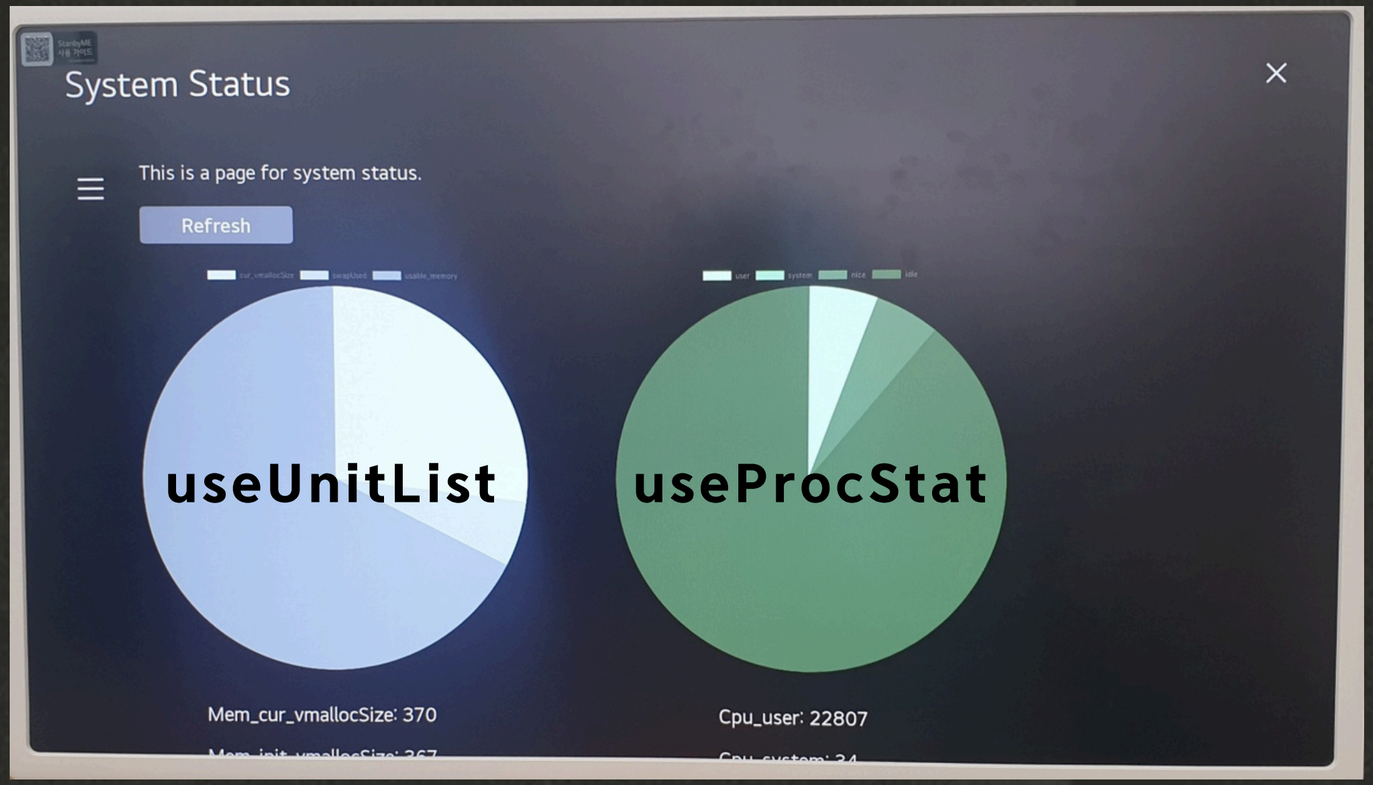


- 목록의 그림을 클릭하면 Sketch Page로 이동해서 해당 그림을 불러오고, New Painting을 누르면 새로운 빈 그림을 생성함
- css style의 overflow:’auto’를 추가해 Page의 목록이 화면을 초과하면 자동으로 스크롤바가 추가되게 설정함

FRONTEND - SKETCH



FRONTEND - CPU/MEM VISUALIZE



- LunaAPI의 getUnitList와 getProcStat를 사용해 각각 메모리 사용정보와 CPU 사용정보를 가져옴
- chart.js 라이브러리의 pie chart를 사용해서 시각화를 구현함
- 상단의 Refresh 버튼을 눌러 CPU 및 메모리 사용정보를 갱신할 수 있음

FRONTEND - 해결한 문제들

Copy 

- Canvas에서 선택한 Object들은 activeSelection 타입 Object로 묶임
- 선택된 Object들을 하나씩 .clone()으로 복사해서 Canvas에 추가함
- 그러나 Image타입 Object는 .clone()이 없어 따로 추출하고, 나머지 Object들의 복사가 끝난 후, 새로운 Image Object를 생성하고 속성값을 복사해서 Canvas에 추가하는 방식으로 문제를 해결함

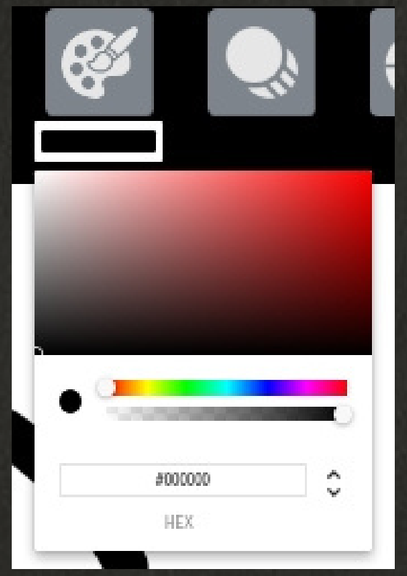
FRONTEND - 해결한 문제들

Eraser

- Fabric API에는 기본 지우개 기능이 존재하지 않음
- Canvas의 기본 Brush인 BaseBrush class를 상속한 EraseBrush class를 생성
- Class 내부에 현재 마우스 포인터 위치에서 Object를 삭제하는 이벤트 핸들러(onMouse, onMove)를 등록해 문제를 해결함

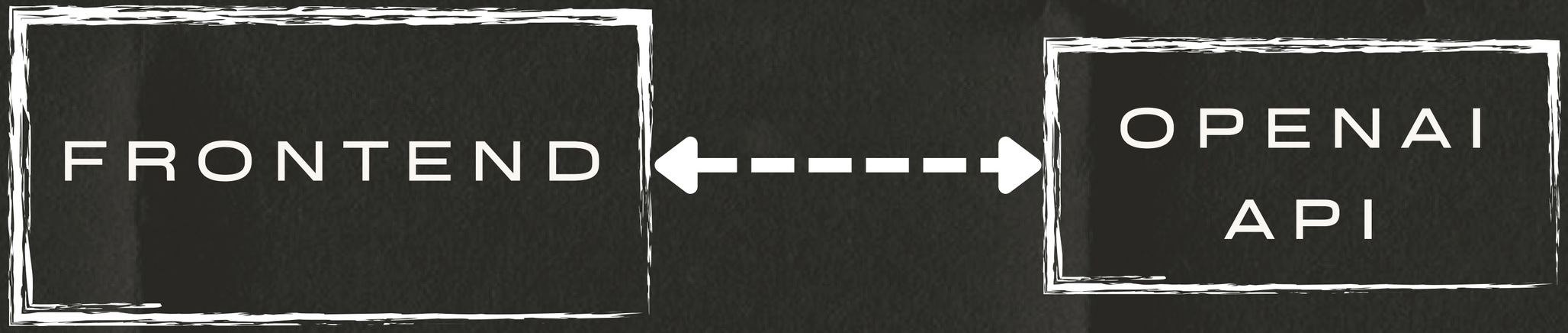
FRONTEND - 해결한 문제들

react-color

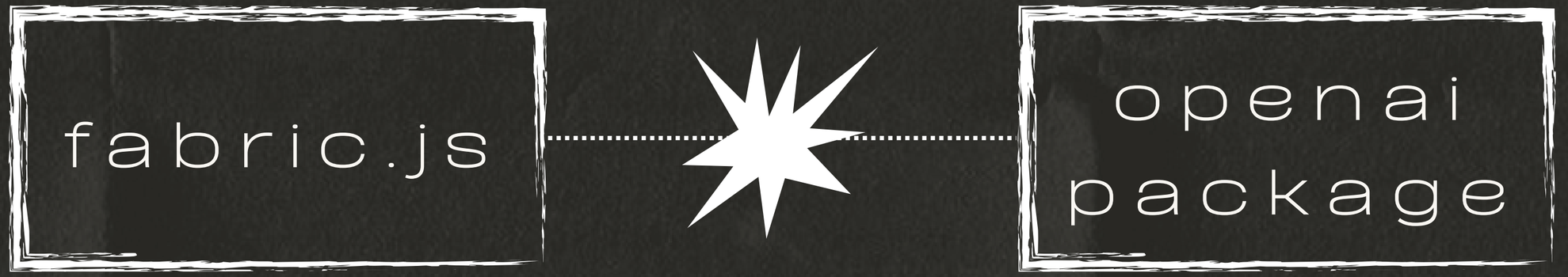


- 기본 프로젝트 템플릿에서도 <input type='color'/> Tag가 작동되지 않음
- 외부 library (react-color) 사용으로 색상 입력을 대체함
- 공식 홈페이지의 API 설명을 참고해 원래 input처럼 버튼을 누르면 색상 선택창이 나오게 제작함

AI SERVICES



AI SERVICES



AI SERVICES

HTTP 요청 라이브러리인 axios를 사용하여
API 엔드포인트에 직접 요청.

AI SERVICES

Image generation	Image Vision
DALL-E-3	GPT-4o-mini
1024x1024	1024x1024
image_url	image_url

AI SERVICES

Image generation	Image Vision
DALL-E-3	GPT-4o-mini
1024x1024	1024x1024
image_url	image_url

AI SERVICES

CORS - Cross-Origin Resource Sharing

CORS를 통하지 않고, 다른 origin으로 부터 가져온 데이터들은 canvas 에 그려지는 즉시 canvas는 오염
오염된 canvas는 더 이상 안전하지 않은 것으로 여겨지고, canvas 이미지 에서 데이터를 가져오려는 어떤 시도든 exception이 발생

```
canvas.toDataURL();
```

AI SERVICES

Image generation	Image Vision
DALL-E-3	GPT-4o-mini
1024x1024	1024x1024
image_url	image_url

AI SERVICES

Image generation	Image Vision
DALL-E-3	GPT-4o-mini
1024x1024	1024x1024
base64_json	base64_json

AI SERVICES

- Ai Image Generator : 텍스트를 입력받아 스케치 풍으로 이미지를 생성함

- Ai Canvas Text Scanner : 그림을 스캔해서 문자를 변환함

- Ai Canvas Title Generator : 그림의 제목을 자동으로 생성함

- 모든 기능은 사용시 대기 알림창(5~10초)을 표시하고, 결과값을 대기하는 도중에도 다른 기능을 사용가능함

AI SERVICES

response
delay

BALANCE

quality

token price

TEST PLAN

Unit test

Integration test

System test

TEST PLAN

Unit test

Integration test

System test

TEST PLAN

Unit test

Integration test

System test

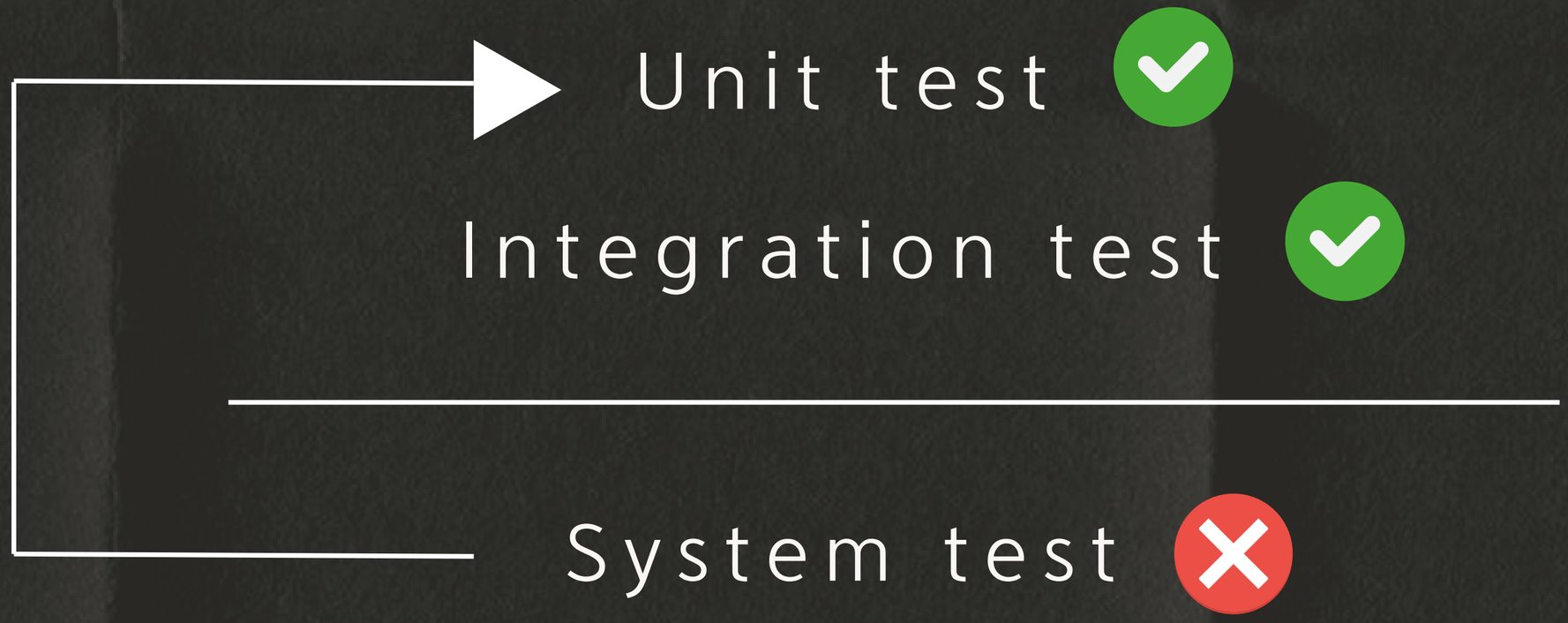
TEST PLAN

Unit test 

Integration test 

System test 

TEST PLAN



REFLECTION

Design

Optimization

PIP/Widget

DEMO / Q&A