

<PROJECT 중간보고서>

# 클라우드 컴퓨팅

2014103189 컴퓨터공학과 강형구

2015104160 컴퓨터공학과 김재연

2015104193 컴퓨터공학과 윤준현

2015104219 컴퓨터공학과 정재혁

2014104158 컴퓨터공학과 최지우

**김재홍 교수님**

## 1. Abstract

이 프로젝트는 클라우드 시스템을 활용하여 사용자가 파일을 관리할 수 있는 클라우드 저장소를 제공하는 것을 목표로 한다. 우리가 구축할 이 서비스는 인터넷이 연결된 환경이라면 어떤 디바이스라도 웹 브라우저를 통하여 쉽게 저장소에 접근할 수 있으며, 저장소라면 필수적인 기능인 파일 시스템 관리, 즐겨찾기 및 유저 편의를 위하여 공유 등의 기능을 제공한다.

이 서비스는 대표적인 클라우드 서비스, 드롭박스의 기능과 디자인을 참고하여 이 서비스에 필요한 기능, 갖추어야 할 요소들을 설계하였으며, 해당 서비스를 개발하기 위해 필요한 기술 스택들을 장단점과 더불어 사용한 이유를 정리하여 Background Study에 서술하였다. 또한, 이 기술들을 활용하여 구체적으로 어떤 방식으로 서비스의 시스템을 설계할 것인지 Project Architecture에 이 서비스의 Architecture Diagram과 함께 설계한 모듈에 대하여 자세하게 설명하였다.

그리고 각 모듈의 API통신을 기능, 접근 url, 필요한 parameter 및 header와 Request를 성공적으로 수행하였을 때 생기는 Response들로 정리하여 Implementation Spec에 기재하였다. 이후에 프로젝트의 현재 진행상황과 함께 앞으로 진행해야 할 업무, 그리고 팀원별 업무 분담과 프로젝트가 어떻게 진행될 것인지에 대한 스케줄을 작성하여 서술하였다.

## 2. Introduction

5G의 시대가 도래한 현재, 인터넷 통신 속도는 이전과는 비교도 할 수 없을 정도로 빨라졌고, 이에 따라 사람들은 자신의 파일들을 로컬 저장소에 저장하지 않고 클라우드에 저장하여 필요할 때마다 다운로드, 미디어의 경우는 스트리밍을 통하여 이용하고 있다. 우리가 인터넷으로 접할 수 있는 거의 모든 것은 클라우드를 통해 만들어졌고 이 구조를 제대로 알기 위해서 클라우드 시스템을 깊게 이해할 필요성을 느꼈다. 이를 위하여 우리들은 클라우드 저장소를 만드는 프로젝트를 시작하게 되었다.

이 프로젝트 기본적인 클라우드 저장소에 필요한 파일 공유 및 업로드, 다운로드의 기능을 제공하는 서비스이다. 클라우드를 활용한 소프트웨어는 여러 문제점이 존재하는데, 이 서비스는 컴퓨팅 환경을 AWS의 EC2로 구성하여 소프트웨어의 Availability를, 그리고 저장소를 AWS의 S3로 설정하여 소프트웨어의 Consistency와 Durability 문제를 해결할 것이다.

## 3. Background study

### A. 관련 접근방법/기술 장단점 분석

#### (1) Vue.js

Vue.js는 복잡해져 가는 프론트엔트 개발을 돕기 위해 나온 프레임워크이자 UI 라이브러리다.

Vue.js는 기존의 라이브러리를 사용하면서도 자신만의 고유한(독립적인) 라이브러리를 사용하여 기존 프로젝트에 더하는 식으로 작업이 가능하고, 그렇기 때문에 기존에 만들어 놓았던 프로젝트와의 통합이 매우 쉽다. 이런 방식으로 Vue를 사용하여 웹 페이지 응용 프로그램을 만들 수 있다.

기본적으로 Vue에서도 HTML을 사용하여 웹 페이지를 만든다. 거기에 추가해서 Vue는 템플릿 형식을 사용하게 되는데, 템플릿을 사용하면 전통적인 웹 개발 개념과 비슷하므로 개발자가 이해하기 쉽고 기여하기에도 좋으며, 기존 어플리케이션을 Vue로 옮기는 데에도 도움이 된다. 템플릿을 사용하면 유용한 전처리기도 사용할 수 있게 된다.

Vue에서 만든 프로젝트는 특별한 변환작업이 없이도 브라우저에서 Vue 프로젝트를 바로 볼 수 있다. 또한, Vue에서는 데이터를 쉽고 빠르게 변환시킬 수 있으며, 설정된 값을 계산하여 화면 이미지를 만들어내는 렌더링 시스템이 매우 빠르고 효율적인 것이 장점이다. Vue는 UI를 렌더링 할 때 가상 DOM을 사용하며, DOM을 조작하는 것에 가능한 가장 적은 오버헤드를 가하며 조작 수를 최소화시킨다. 게다가 Vue는 상태를 항상 모니터링하여 변경이 일어날 때마다 자동으로 렌더링을 실행하여 반응성이 매우 좋다. 전체적으로 Vue는 빠르고 용량이 가벼우며 라이브러리 구조도 간단하다.

단점은 Vue가 간단한 만큼 큰 규모의 어플리케이션을 만드는 데에는 그렇게 효율적이지 않다는 것이다. 만드는 어플리케이션의 규모가 더욱 커질 때, 이런 가볍고 빠른 특성을 가진 Vue의 기술적 부채가 늘어나 한계가 발생하게 된다. 그리고 템플릿 구조는 런타임 에러가 나오기 쉽고, 테스트하기가 어려우며, 재사용성이 떨어진다.

## (2) mongoDB

mongoDB는 NoSQL 기반의 데이터베이스로 특징과 장점을 나열하면, 아래와 같다.

### 1) 데이터를 유동적으로 저장할 수 있다.

Document 라는, Json 형태로, 데이터를 저장하기 때문에, 언제나 데이터 구조를 변경할 수 있기 때문에, 데이터베이스 설계의 부담이 비교적 적다. 또한 Document 는 어플리케이션 코드에 오브젝트로 자연스럽게 mapping 되기 때문에, 훨씬 쉽게 배우고 사용할 수 있다.

### 2) Availability, horizontal scaling, geographic distribution 을 쉽게 구현하고 사용할 수 있다.

분산 데이터베이스 기반으로, Availability, horizontal scaling, geographic distribution 이 이미 구현되어 있으며, 쉽게 사용 할 수 있다. monogDB 는 자동으로 추가적인 노드에 데이터를 저장하기 때문에 높은 Availability 를 기대할 수 있다 이론상 시스템에 문제가 발생하면 평균 5 초 이내에 복구한다. 또한 분산, 샤딩 등을 기본적으로 지원한다. 따라서 Scale 을 위해서 따로 어플리케이션에 추가적인 작업이 필요하지 않다.

### 3) 무료로 사용할 수 있다.

2018 년 8 월 16 일 전의 버전은 모두 AGPL 라이선스로 배포되었으며, 2018 년 8 월 16 일 이후의 버전은 SSPL 라이선스하에 배포되었다.

단점은 아래와 같다.

- join 연산을 지원하지 않는다.

복잡한 관계의 데이터를 저장하고 처리하기에는 적합하지 않으며, 구현 한다고 하더라도 상황에 따라 성능 저하와 코드 복잡도가 증가할 수 있다.

- 서로 다른 document 사이에서는 트랜잭션을 지원하지 않는다.

은행 업무와 같은 높은 일관성을 요구하는 경우 일관성을 보장하는 부분을 구현해줘야 하며 이는 굉장히 어려운 작업이다.

요약하자면 mongoDB는 NoSQL과 분산 데이터베이스 기반으로 높은 Availability와 Scalability를 지원하지만, Consistency 부분에서는 어느 정도 보완해야 할 이슈가 있기 때문에, 빠르고 쉽게 어플리케이션을 구축하고 Scalability와 Availability를 쉽게 구현할 때 사용하기 적합한 데이터베이스라고 할 수 있다.

### (3) Amazon EC2. Elastic Compute Cloud

Amazon Elastic Compute Cloud는 사용자가 가상 컴퓨터를 임대 받아 그 위에 자신만의 컴퓨터 애플리케이션들을 실행할 수 있게 하는 Amazon의 Cloud Service로 IaaS(Infrastructure as a Service)이다. 하나의 Virtual Machine을 “인스턴스”라고 하며, 고객으로 하여금 원하는 가격 내에, 원하는 성능을 발휘할 수 있는 인스턴스를, 원하는 운영체제와 소프트웨어와 함께 대여하여 사용할 수 있도록 해준다. 예시로는, 베어 메탈 인스턴스, GPU 컴퓨팅 인스턴스, GPU 그래픽 인스턴스, 높은 I/O 인스턴스, 고밀도 HDD 스토리지 인스턴스, 클러스터 인스턴스가 있다.

사용 용도에 따라서 인스턴스를 다르게 선택하면 보다 쉽게 최적화를 진행할 수 있다. Storage에 최적화 된 I3, I3en, D2, H1 중 하나로, I3와 I3en은 SSD, D2와 H1는 HDD를 Storage로 사용하며, 네트워킹 성능으로는 I3보다 I3en가, D2보다는 H1가 더 좋다.

다양한 Amazon EC2 워크로드마다 스토리지 요구 사항이 상당히 다를 수 있는 것을 고려하여 내장된 인스턴스 스토리지 외에 Amazon Elastic Block Store(Amazon EBS) 및 Amazon Elastic File System(Amazon EFS)와도 연동할 수 있다.

EFA(Elastic Fabric Adapter)는 Amazon EC2 인스턴스를 위한 네트워크 인터페이스로, 고객이 전산 유체 역학, 기후 모델링, 저수지 시뮬레이션과 같이 높은 수준의 인스턴스 간 통신이 필요한 HPC 애플리케이션을 AWS에서 대규모로 실행할 수 있도록 지원한다. EFA를 사용하면 MPI(메시지 전달 인터페이스) 같은 주요 HPC 기술을 사용하는 HPC 애플리케이션을 수천 개의 CPU 코어로 확장할 수 있다.

Amazon EC2는 인스턴스를 여러 위치에 배치할 수 있는 기능을 제공한다. Amazon EC2 위치는 지역과 가용 영역으로 구성된다. 가용 영역은 다른 가용 영역에 장애가 발생할 경우 영향을 받지 않도록 구축된 개별 지점으로, 동일 지역 내의 다른 가용 영역에 저렴하고, 지연 시간이 짧은 네

트위크 연결을 제공한다.

Elastic IP 주소란 동적 클라우드 컴퓨팅을 위해 설계된 고정 IP 주소를 말한다. Elastic IP 주소는 특정 인스턴스가 아닌 사용자의 계정과 연결되며 사용자는 명시적으로 해제할 때까지 해당 주소를 제어한다.

Amazon EC2 Auto Scaling을 사용하면 정의한 조건에 따라 Amazon EC2 용량을 자동으로 확장하거나 축소할 수 있습니다. EC2 Auto Scaling은 용량에 대한 수요가 급증할 경우에는 사용 중인 Amazon EC2 인스턴스 수를 자동으로 늘려 성능을 유지할 수 있게 하고, 수요가 감소할 경우에는 인스턴스 수를 자동으로 줄여 비용을 최소화할 수 있게 한다.

#### **(4) Amazon S3, Simple Storage Service**

웹 인터페이스를 통해 Storage를 제공하는 파일 호스팅 서비스이다. 높은 확장성과 가용성, 낮은 지연시간을 제공하는 것을 목표로 Object storage 아키텍처를 사용하여 데이터를 관리한다. 기본 스토리지 단위는 버킷(Bucket)들로 이루어진 객체(Object)이며, 각 객체는 user-assigned key를 통해 식별된다. 객체는 S3에서 제공하는 AWS SDK 또는 Amazon S3 REST API를 사용하여 관리할 수 있으며, 최대 5TB 크기와 2KB의 메타데이터로 이루어진다. 또한 HTTP GET 인터페이스와 BitTorrent 프로토콜을 사용하여 객체를 다운로드 할 수 있다. Amazon S3는 무제한이나 다름없는 용량을 제공하며, 사용한 만큼만 비용을 지불하는 방식이다. 사용량은 GB단위로 계산되며, 기본 S3 Standard 스토리지가 GB당 0.023달러(약 30원)로 저렴하다. 이외에도 요청 및 데이터 검색, 데이터 전송, 관리 및 복제 비용이라는 추가적인 비용 요소를 고려해야 하지만, 그럼에도 매우 저렴한 편이다.

그러나 이 때문에 세분화된 가격 정책이 복잡해 써드파티 비용 관리 툴이 필요하다는 단점이 있다. 여러 시스템에 걸쳐 모든 S3 객체의 복사본을 자동으로 생성하고 저장하기 때문에 높은 내구성을 제공하고, 데이터 액세스의 패턴 변화에 따라 몇 가지 스토리지 클래스를 나누어 비용을 절약할 수 있다. 또한 S3는 S3 퍼블릭 액세스 차단을 통해 버킷 또는 계정 수준에서 모든 객체에 대한 퍼블릭 액세스를 차단할 수 있다. 여러 가지 규정 준수 프로그램을 적용하여 규제 요건을 준수할 수 있도록 지원하므로, 보안과 감사기능이 좋다고 할 수 있다.

Amazon S3는 개발자를 위한 여러 가지 API를 제공한다. AWS console이 파일을 관리하고 업로드 하는 기능을 제공하지만, 큰 버킷을 관리하거나 파일 수정은 불가능하다. Cloudberry Explorer, ForkLift와 같은 소프트웨어를 통해 S3에서 파일을 편집할 수 있다. 또한 S3Stat, Cloudytics 등을 사용하여 요청 콘텐츠에 대한 접근 시간, 사용 프로토콜, HTTP 코드, 처리 시간, HTTP 요청 메시지를 포함한 로그를 분석할 수 있다. S3는 HTTP형식으로 Storage에 접근하기 때문에, S3를 이용한 시스템은 일반 파일 시스템과 같은 형태로 접근을 하면 성능이 떨어진다.

## (5) Amazon Cognito

앱 및 모바일 앱에 대한 인증, 권한 부여 및 사용자 관리를 제공하는 Amazon 서비스. 사용자는 ID, Password를 통해 로그인 하거나, Facebook, Amazon, Google 또는 Apple 같은 타사를 통해 로그인 할 수 있다.

Cognito는 사용자 풀과 자격 증명 풀로 이루어져 있다. "사용자 풀"은 Cognito 사용자의 가입 및 로그인 옵션을 제공하는 사용자 모음이며, "자격 증명 풀"은 사용자들이 Amazon S3 및 DynamoDB 등과 같은 AWS 서비스를 액세스 할 권한을 관리하는 권한 모음이다. 두 가지를 별도로 혹은 함께 사용할 수 있다.

사용자 풀의 자세한 기능은 다음과 같은 것이 있다.

- 가입 및 로그인 서비스
- 사용자 로그인을 위한 내장 UI 제공
- Facebook, Google 등을 통한 소셜 로그인 및 사용자 풀의 SAML 및 OIDC 자격 증명 공급자를 통한 로그인 가능
- 멀티 팩터 인증(MFA), 이상 자격 증명 확인, 계정 탈취 보호, 전화 및 이메일 확인과 같은 보안 기능
- AWS Lambda 트리거를 통한 사용자 지정 워크플로우 및 사용자 마이그레이션

자격 증명 풀에서 지원하는 자격 증명 공급자로는 다음과 같은 것이 있다.

- Amazon Cognito 사용자 풀
- Facebook, Google, Login with Amazon 및 Sign in with Apple 을 통한 소셜 로그인
- OpenID Connect(OIDC) 공급자
- SAML 자격 증명 공급자
- 개발자 인증 자격 증명

## (6) Elastic Load Balancing

Elastic Load Balancing(이하 ELB)는 간단히 말하면 AWS가 관리하고 서비스 하는 로드 밸런서이다. 따라서 이를 이용하는 사용자(개발자) 입장에서는 ELB의 기능을 추상적인 수준에서만 이해하면 간편하게 AWS Console을 이용해서 로드밸런서를 설정할 수 있다.

ELB는 들어오는 어플리케이션 트래픽을 EC2, 컨테이너, IP주소, Lambda 함수 등 여러 대상으로 자동으로 분산 시키며, 단일/복수개의 가용 영역에서 어플리케이션의 부하를 처리 할 수 있으나, 단일 가용 영역을 사용하는 것은 구조적으로 좋은 설계는 아니다. 현재 ELB는 세가지 종류로 제공된다.

- Application Load Balancer
  - HTTP/HTTPS 에 가장 적합. 개별 요청 수준(계층 7)에서 작동.
  - 요청 콘텐츠 기반으로 VPC 내의 대상으로 트래픽을 분산(라우팅)

- Network Load Balancer
  - 매우 극한의 성능이 요구되는 TCP/TLC 등에 적합(OSI 4 계층)에서 작동
  - 갑작스러운 일시적 트래픽 패턴 처리에도 최적화
- Classic Load Balancer
  - Amazon EC2 인스턴스에서 기본적인 로드 밸런싱을 제공
  - 요청 수준 및 연결 수준에서 작동합니다.
  - EC2-Classic 네트워크 내에 구축된 애플리케이션을 대상

## B. 프로젝트 개발환경

Operating System	Ubuntu(Amazon AWS, Windows Subsystem for Linux)
Framework	Django,Vue.js
Database	MongoDB
Storage	S3

## 4. Goal/Problem & Requirements

한 기기에 저장된 파일을 모든 IT 기기 또는 다른 환경에서 접근하여 보기 좋고 쉽게 업로드, 다운로드 하여 언제 어디서든 파일에 access해 작업을 하고자하는 필요성을 느끼게 된 적이 있다. 이를 위한 방법으로 파일을 물리적인 저장소에 저장해 저장소를 직접 들고다니거나, 이메일 같은 환경에 파일을 자신의 메일에 업로드 하고 다운로드 하여 작업을 하는 방법 등이 있다. 하지만 이러한 방법들에는 제약조건이 많다. 물리적 저장소는 용량 제약이 있으며, 외부에서 가해지는 충격에 의해 데이터를 잃어버릴 수도 있고, 자연스레 지워지기도 한다. 또한 휴대용이기 때문에, 들고 다니면서 잃어버릴 위험부담이 있다. 이메일 환경은 일정 크기 이하 용량의 파일만 업로드가 가능하고, 그마저도 용량이 일정 범위를 초과하면 파일이 기간제로 전환되어, 기간 내에 다운로드 하지 않으면 사라지는 경우가 있다.

이를 해결하고자, 클라우드 기반의 파일 공유 웹 서비스를 만들어, 그곳에 파일을 업로드하여 다른 환경에서도 파일을 업로드 및 다운로드 하여 작업할 수 있도록 하고자 하며, 항상 접근이 가능하도록 지속적인 서비스를 제공하는 것이 목표이다.

## 5. Approach

ThrowBox 는 파일의 업로드 및 다운로드, 조회 기능을 어디서든지 접근할 수 있도록 구현한 웹 서비스이다. 이를 원활하게 실행하고, 사용자가 불편함을 느끼게 하지 않기 위해서 고려해야 할 점은, 첫째로 데이터가 일관성을 가지도록 빠르게 업데이트 되어야 하며, 두번째로는 서비스에 항상 접근 가능하도록 해야 한다는 점이다. 그리고 만일을 대비해 scalable 하게 환경이 구성되어야 한다.

기본적으로 Amazon Web Service 의 클라우드 환경을 기반으로 서비스를 제공한다. AWS 는 전 세계의 유명 기업들과 단체가 사용하는 검증된 클라우드 컴퓨팅 서비스로, 사용자의 프로젝트 운영을 돕는 다양한 서비스 API 가 이미 만들어져 있으며, 그 서비스들을 바탕으로 IT 인프라를 구축할 수 있다. 또한 API 로 서비스들을 제어하고 자동화 할 수 있어 매우 효율적이다.

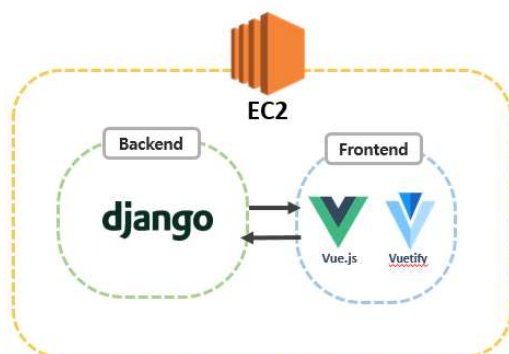
이런 편리함과 효율성을 바탕으로 AWS 의 컴퓨팅 서비스로 EC2 를 채택했다. EC2 는 서버 인스턴스를 생성하고 그 위에 사용자의 프로젝트를 올려 실행 시킬 수 있게 한다. 인스턴스를 원하는 만큼 늘리고, 줄이는 scaling 이 매우 쉽고 용이하며, 트래픽량을 실시간으로 모니터링해서 자동적으로 인스턴스를 scaling 하는 기능까지 가지고 있어서 서비스를 scalable 하게 만들어준다. 즉, 사용자 입장에서 서비스가 다운되거나 지체되는 문제를 해결시켜 주어 가용성을 확보하고, 개발자의 입장에서는 필요하지 않은 리소스 낭비를 방지하여 비용 문제를 해결해주는 이점을 가지고 있다.

또한 AWS 는 전 세계적으로 가용 영역과 데이터센터를 다중화 시켜, 데이터가 증발하는 문제를 해결했으며, 동기화가 빠르게 진행되어 일관성이 유지되고, 기본적인 인프라를 제공하여 사용자의 up-front 비용을 최소화 시키기에 부담없이 사용할 수 있는 서비스이다.

## 6. Project Architecture

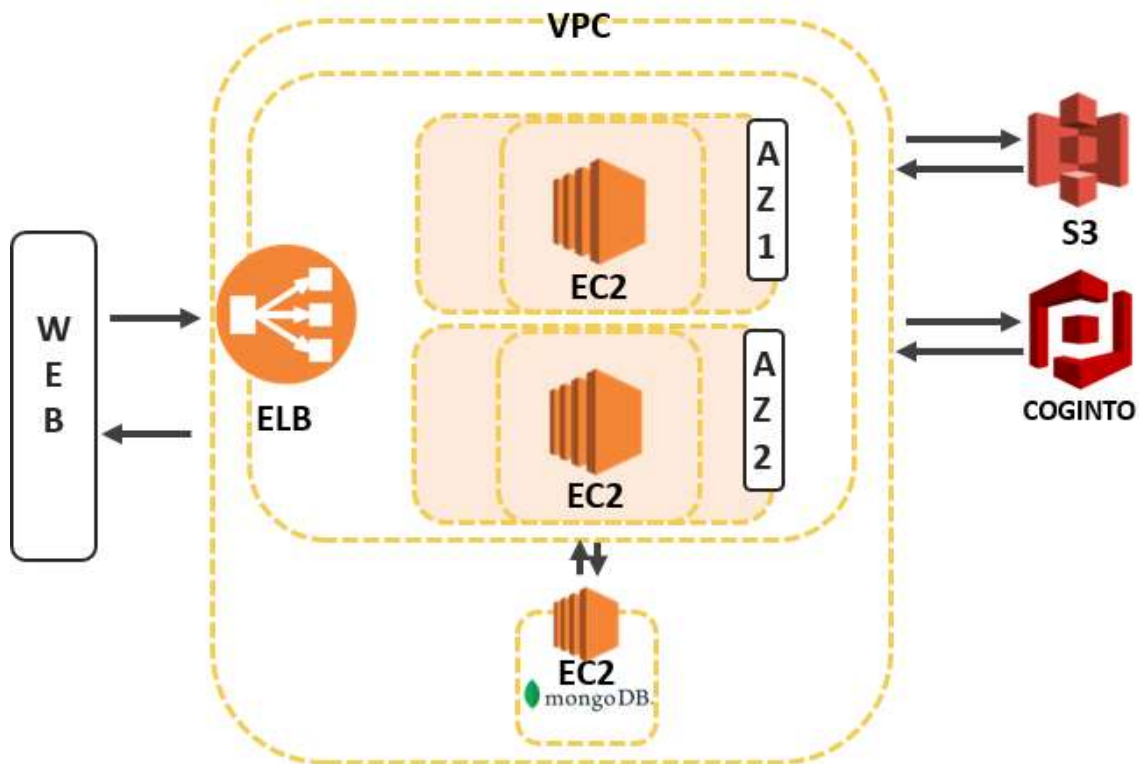
### A. Architecture Diagram

<EC2 구성>





## <전체적인 설계>



## B. Architecture Description

이 프로젝트의 시스템은 AWS의 EC2 상에서 구동되며 웹 서비스 형태로 제공된다. 웹의 Frontend는 자바스크립트 프레임워크 중 하나인 Vue.js와 Vue.js의 CSS 프레임워크인 Vuetify를 사용하여 일반 이용자들에게 친숙한 Material Design으로 UI를 구성하고 기존의 Drive 사용자들에게 친숙하고 직관적인 UX를 제공한다.

Backend는 웹 프레임워크인 Django와 NoSQL 데이터베이스인 MongoDB로 구성된다.

Django를 통해 Frontend에서 발생하는 다양한 요청(파일 조회, 파일 업로드 및 다운로드 등)을 받아들이고 요청에 맞는 응답을 보내주어 Frontend에 적절한 화면을 띄울 수 있게 한다. MongoDB에서는 사용자의 파일 시스템의 구조를 저장하고, 사용자 데이터(ID, Password, Email)은 Amazon Cognito를 활용하여 관리한다. 파일의 업로드와 다운로드는 AWS S3를 통하여 구현하고 Django와의 통신으로 필요한 정보를 교환한다.

과도한 트래픽이 올 경우를 대비하여 ELB를 사용하여 트래픽이 분산되게 설계하였으며, AZ을 2개로 설정하여, AZ 중 1개가 사용하지 못하게 되었을 때를 대비하였다.

## 7. Implementation Spec

### A. Input/Output Interface

#### 1. SignUp

- A. Describe : Cognito 에 회원가입을 요청
- B. URL : /api/signUp/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>username</b>	String	회원이가입을 요청할 사용자의 ID	No	-
<b>password</b>	String	회원이가입을 요청할 ID 의 비밀번호	No	-
<b>email</b>	String	확인 코드를 수신할 사용자의 Email	No	-

#### 2. SignUpConfirm

- A. Describe : 회원가입 요청 후, 이메일로 전송된 인증 코드 입력
- B. URL : /api/signUpConfirm/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>Username</b>	String	인증 코드를 수신한 사용자의 ID	No	-
<b>confirmationCode</b>	Integer	이메일로 수신된 확인 코드	No	-

#### 3. SignIn

- A. Describe : 사용자 로그인 요청
- B. URL : /api/signIn/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>username</b>	String	로그인 요청할 사용자의 ID	No	-
<b>password</b>	String	사용자 ID 의 Password	No	-

#### 4. UserDetails

- A. Describe : 사용자 정보(username, email) 요청
- B. URL : /api/userDetail/
- C. HTTP Method : GET
- D. Parameter :

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-

#### 5. UserModify

- A. Describe : 사용자 정보(password) 수정
- B. URL : /api/userModify/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-
attribute	String	변경할 사용자 정보의 속성	No	-
preValue	String	변경하기 전의 정보	No	-
postValue	String	새로운 정보	No	-

#### 6. UserDelete

- A. Describe : 사용자 탈퇴
- B. URL : /api/userDelete/
- C. HTTP Method : DELETE
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-

#### 7. FileList

- A. Describe : 해당 폴더의 파일 목록 요청
- B. URL : /api/fileList/
- C. HTTP Method : GET
- D. Parameter : application/json

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-
path	String	파일 목록을 요청할 폴더의 ID	Yes	-

## 8. FileUpload

- A. Describe : 파일 업로드 요청
- B. URL : /api/fileUpload/
- C. HTTP Method : POST
- D. Content-Type : Multipart/form-data
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>file</b>	FILE	업로드 할 파일	No	-
<b>path</b>	String	파일이 업로드 될 폴더의 ID	Yes	-
<b>isFile</b>	Boolean	파일인지 아닌지(폴더)를 나타내는 값	No	-
<b>fileSize</b>	Integer	파일의 크기	No	-

## 9. FolderUpload

- A. Describe : 폴더 생성 요청
- B. URL : /api/folderUpload/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>path</b>	String	폴더를 생성할 위치의 폴더 ID	Yes	-
<b>name</b>	String	폴더의 이름	No	-
<b>isFile</b>	Boolean	파일인지 아닌지(폴더)를 나타내는 값	No	-
<b>fileSize</b>	Integer	파일의 크기	No	-

## 10. FileDownload

- A. Describe : 파일 다운로드 URL 요청
- B. URL : /api/fileDownload/
- C. HTTP Method : GET
- D. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>fid</b>	String	다운로드를 요청할 file 의 ID	No	-

## 11. FileRename

- A. Describe : 파일의 이름 변경
- B. URL : /api/fileRename/
- C. HTTP Method : PUT
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>path</b>	String	파일이 속한 폴더의 ID	No	-
<b>fid</b>	String	이름을 변경할 파일의 ID	No	-
<b>name</b>	String	파일의 새로운 이름	No	-

## 12. FileStarred

- A. Describe : 즐겨 찾기 목록에 추가
- B. URL : /api/fileStarred/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>fid</b>	String	즐거 찾기에 추가할 파일의 ID	No	-

## 13. FileStarred

- A. Describe : 즐겨 찾기 목록 요청
- B. URL : /api/fileStarred/
- C. HTTP Method : GET
- D. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-

#### 14. FileMove

- A. Describe : 파일을 다른 위치로 이동
- B. URL : /api/fileMove/
- C. HTTP Method : PUT
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>fid</b>	String	이동시킬 파일의 ID	No	-
<b>Path</b>	String	파일의 새로운 위치		

#### 15. FileRecent

- A. Describe : 가장 최근에 추가한 파일의 목록 요청
- B. URL : /api/fileRecent/
- C. HTTP Method : GET
- D. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-

#### 16. FileTrash

- A. Describe : 파일을 휴지통으로 이동
- B. URL : /api/fileTrash/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-
<b>fid</b>	String	휴지통으로 이동시킬 파일의 ID	No	-

#### 17. FileTrashList

- A. Describe : 휴지통의 파일 목록 요청
- B. URL : /api/fileTrashList/
- C. HTTP Method : GET
- D. Parameter :

Field	Type	Describe	Null	Default
<b>AccessToken</b>	String	사용자 인증 Token	No	-

## 18. FileRecovery

- A. Describe : 휴지통에 있는 파일 복구
- B. URL : /api/fileRecovery/
- C. HTTP Method : POST
- D. Content-Type : application/json
- E. Parameter :

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-
fid	String	복구를 요청할 파일의 ID	No	-

## B. Inter Module Communication Interface

### 1. FileErase

- A. Describe : 휴지통에 30 일 이상 있었던 파일 삭제
- B. URL : /api/fileErase/
- C. HTTP Method : POST
- D. Communicated Module name : SignIn(/api/signIn/), POST
- E. Content-Type : application/json
- F. Parameter :

Field	Type	Describe	Null	Default
AccessToken	String	사용자 인증 Token	No	-

## C. Modules

### - 사용자 관련 기능

- SignUp API : 회원가입 관련 모듈
- SignUpConfirm API : 이메일 인증 코드 확인 모듈
- SignIn API : 로그인 관련 모듈
- UserDetail API : 사용자 정보 요청 모듈
- UserModify API : 사용자 정보 수정 모듈
- UserDelete API : 사용자 삭제 요청 모듈

### - 파일 관련 기능

- FileList API : 파일 목록 출력 모듈
- FileUpload API : 파일을 업로드, 데이터를 생성 모듈
- FolderUpload API : 폴더를 업로드, 데이터를 생성 모듈
- FileDownload : 파일을 로컬 저장소에 저장하기 위한 다운로드 URL 요청 모듈
- FileRename : 파일의 이름을 변경하기 위한 모듈
- FileStarred API : 즐겨찾기 관련 모듈

- FileMove : 파일의 경로 수정 모듈
- FileRecent API : 최근 추가한 파일 목록을 출력해주는 모듈
- FileErase : 30 일이 지난 휴지통 내의 파일 자동 삭제 모듈
- FileTrash : 휴지통으로 파일 이동 모듈
- FileTrashList : 휴지통 내의 파일 목록 요청 모듈
- FileRecovery : 휴지통에 있었던 파일 복구 모듈

## 8. Solution

### A. Implementations Details

- 사용자 관련 기능
  - 회원가입/이메일인증
  - 로그인/로그아웃
  - 회원정보조회
  - 비밀번호 변경
  - 회원탈퇴
- 파일 관련 기능
  - 파일 업로드/파일 다운로드/폴더 생성
  - Drag&Drop 으로 업로드 가능
  - 다중 파일 업로드 가능
  - 파일 이동/이름 변경/삭제
  - 파일 검색/정렬
  - ThrowBox 사용 용량 표시
  - 즐겨찾기 설정/취소
  - 최근에 추가한 파일 목록
  - 휴지통/오래된 휴지통 파일 자동 삭제

### B. Implementations Issues

- AWS Educate 계정의 한계로 IAM 의 Full Access 권한이 없어서 팀원 각자 만든 Cognito, S3, EC2 에 IAM 을 통한 접근할 수 없는 문제 발생. 특히 boto3 를 사용하여 접근해야 할 Cognito 와 S3 의 Access key 생성이 불가능하여 프리티어 계정을 생성하여서 진행하였다.
- MongoDB 가 실행되고 있는 EC2 와의 네트워크 속도가 매우 느린 편이었다. AWS Educate 계정은 버지니아 Region 밖에 선택할 수 없어 개선하지 못하였다. 도쿄 Region 에 있는 EC2 와의 속도를 실험해보았을 때에 매우 빠름을 확인하였다.
- 단일 EC2 를 사용할 경우, 과도한 트래픽을 처리하지 못 할 가능성이 있으므로, ELB 를 통해 트래픽을 관리하도록 진행하였다. 또한 AZ 을 단일로 구성하면 가용성을 유지할 수 없으므로 AZ 는 2 로 설계하였다



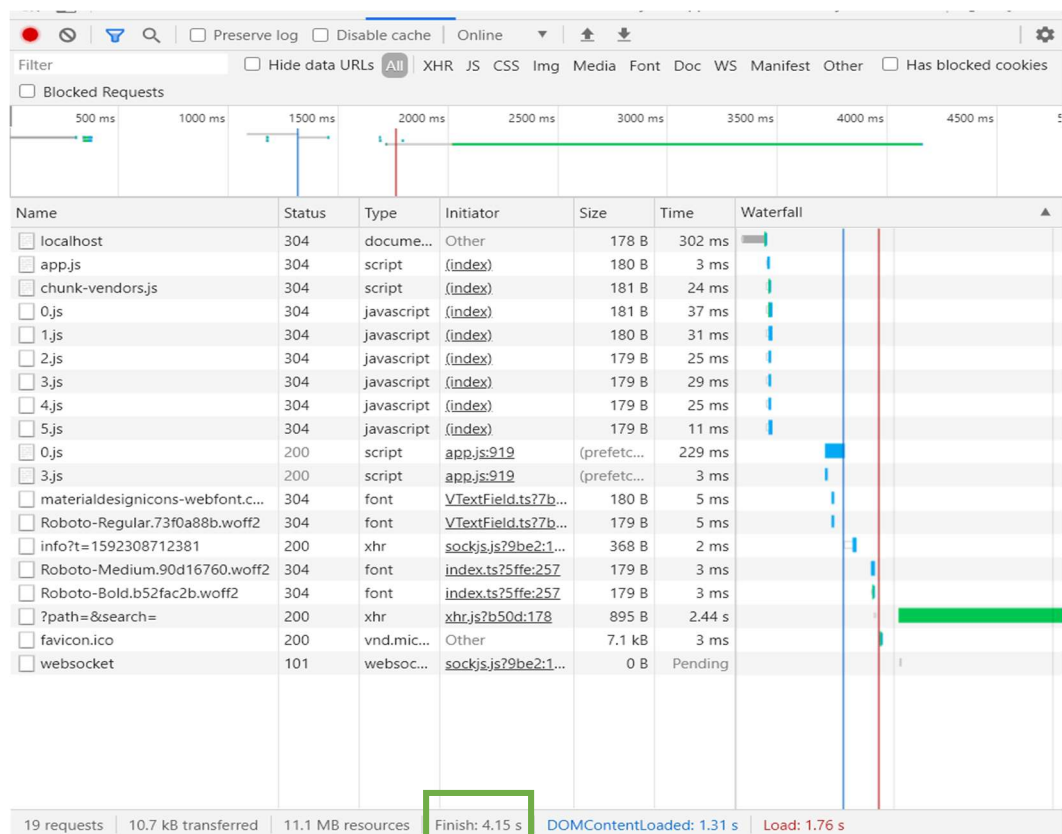
## 9. Results

### A. Experiments

- 간단한 웹 기반 클라우드 서비스를 제공하는 어플리케이션이기 때문에 속도는 곧 성능이고, 사용자의 서비스에 대한 만족도를 좌우하는 중요한 요소이다. 하지만 AWS의 Educate 계정으로는 Region 선택에 한계가 있기 때문에 Region 차이로 인한 속도차이가 있다는 것이 파악되었다.
- 구글 크롬의 개발자도구 Network 탭을 통해 속도 및 시간을 측정하였다.
- 실험에서 측정한 시간은 다음과 같다.
  1. 로그인 후 첫 파일 조회에 걸리는 시간
  2. 파일 업로드에 걸리는 시간
  3. 파일 다운로드에 걸리는 시간

### B. Result Analysis and Discussion

1. 로그인 후 첫 파일조회에 걸리는 시간



ID와 비밀번호를 입력한 후 로그인 버튼을 처음 눌렀을 때를 기점으로 파일이 조회되어 리턴 되기까지 걸리는 시간을 측정하였다. 첫 시도에는 4.15 초, 두번째

시도에서는 3.78 초, 세번째에서는 3.84 초, 네번째에서 3.81 초가 소요되었다.  
 평균적으로 3.89 초가 걸렸다.

## 2. 파일 업로드에 걸리는 시간

3.js	200	script	app.js:919	(prefetc...	2 ms
materialdesignicons-webfont.c...	304	font	VTextField.ts77b...	180 B	4 ms
Roboto-Regular.73f0a88b.woff2	304	font	VTextField.ts77b...	179 B	5 ms
info?t=1592310837639	200	xhr	sockjs.js79be2:1...	428 B	6 ms
Roboto-Medium.90d16760.woff2	304	font	index.ts75ffe:257	179 B	3 ms
Roboto-Bold.b52fac2b.woff2	304	font	index.ts75ffe:257	179 B	4 ms
?path=&search=	200	xhr	xhr.js?b50d:178	895 B	2.77 s
websocket	101	websock...	sockjs.js79be2:1...	0 B	Pending
favicon.ico	200	vnd.mic...	Other	7.1 kB	4 ms
fileUpload/	201	xhr	xhr.js?b50d:178	507 B	4.34 s
?path=&search=	200	xhr	xhr.js?b50d:178	1.1 kB	2.75 s

파일을 드래그해서 홈페이지에 올려놓는 순간부터 파일의 업로드 완료에 걸리는 시간까지 측정하였다. 1 바이트의 파일을 업로드하는데 걸리는 시간은 4.34 초였다. 그 후 3 번 더 시도한 결과 각각 4.50 초, 4.75 초, 5.19 초 의 시간이 걸렸다. 업로드에는 평균적으로 4.69 초가 걸렸다 같은 방법을 사용하여 1 KB 의 파일을 업로드 하는데 걸리는 시간은 3.99 초, 4.19 초, 4.52 초, 4.84 초, 평균적으로 4.38 초가 소요되었다.

favicon.ico	200	vnd.mic...	Other	7.1 kB	4 ms
?path=&search=	200	xhr	xhr.js?b50d:178	2.3 kB	3.05 s
fileUpload/	201	xhr	xhr.js?b50d:178	503 B	5.34 s
?path=&search=	200	xhr	xhr.js?b50d:178	2.5 kB	2.81 s
fileUpload/	201	xhr	xhr.js?b50d:178	507 B	5.24 s
?path=&search=	200	xhr	xhr.js?b50d:178	2.7 kB	2.84 s
fileUpload/	201	xhr	xhr.js?b50d:178	507 B	5.55 s
?path=&search=	200	xhr	xhr.js?b50d:178	2.9 kB	2.74 s
fileUpload/	201	xhr	xhr.js?b50d:178	507 B	7.31 s
?path=&search=	200	xhr	xhr.js?b50d:178	3.0 kB	2.78 s

28 requests | 25.3 kB transferred | 11.1 MB resources | Finish: 1.3 min | DOMContentLoaded: 1.36 s | Load: 1.82 s

이번에는 1 MB 크기의 파일을 총 4 번 업로드했다. 각각 5.34 초, 5.24 초, 5.55 초, 7.31 초가 걸렸다. 평균적으로 5.86 초가 걸렸다.

## 3. 파일 다운로드에 걸리는 시간

favicon.ico	200	vnd.mic...	Other	7.1 kB	4 ms
?fid=5ee8c442d0293318b009e...	200	xhr	xhr.js?b50d:178	448 B	2.37 s
5ee8c442d0293318b009e016	200	xhr	xhr.js?b50d:178	(disk ca...	2 ms
?fid=5ee8c442d0293318b009e...	200	xhr	xhr.js?b50d:178	448 B	2.37 s
5ee8c442d0293318b009e016	200	xhr	xhr.js?b50d:178	(disk ca...	1 ms
?fid=5ee8c442d0293318b009e...	200	xhr	xhr.js?b50d:178	448 B	2.46 s
5ee8c442d0293318b009e016	200	xhr	xhr.js?b50d:178	(disk ca...	2 ms
?fid=5ee8c442d0293318b009e...	200	xhr	xhr.js?b50d:178	448 B	2.34 s
5ee8c442d0293318b009e016	200	xhr	xhr.js?b50d:178	(disk ca...	2 ms

원하는 파일을 더블클릭 한 후 다운로드 버튼을 누르는 것부터 시작하여 PC의 다운로드 폴더에 다운로드가 완료될 때 까지의 시간을 측정하였다.

1바이트의 파일을 다운로드 받았을 때 걸리는 시간은 각각 2.37초, 2.37초, 2.46초, 2.34초였다. 평균적으로 2.38초가 걸렸다. 1 KB의 파일을 다운로드 받을 때는 2.55초, 2.37초, 2.40초, 2.22초가 걸렸으며 평균적으로 2.38초가 걸렸다. 1 MB의 파일을 다운로드 받을 시에 2.47초, 2.35초, 2.52초, 2.37초가 걸렸다. 평균은 2.42초이다.

측정 항목	1 Byte	1 KB	1 MB
파일 업로드	4.69초	4.38초	5.86초
파일 다운로드	2.38초	2.38초	2.42초
로그인 및 파일조회	3.89초		

위의 측정결과를 보면 파일을 조회하는 속도가 약간 느리고, 파일을 업로드 할 수록 응답 시간이 길어진다는 문제가 있다는 걸 알 수 있었다. 하지만 AWS Educate 계정에 서 서비스 Region을 버지니아로 선택할 수 밖에 없는 한계로 인해 응답에 어느정도 지연이 생기는 것을 감수해야한다.

대체적으로 파일을 업로드 하는 것보다는 다운로드 하는데 걸리는 시간이 더 짧음을 알 수 있었다. 또한 1Byte 파일 업로드의 평균 속도와 1KB 파일 업로드의 평균 속도를 비교해 봤을 때, 1Byte가 더 느리게 나오는 것으로 보아 네트워크의 상태 또한 응답 시간에 많은 영향을 끼치고 있다는 것을 알 수 있었다.

## 10. Division & Assignment of Work

항목		담당자
Front-End (Html, CSS, Vue.js)	전체적인 UI/UX 구성	윤준현
	Storage, Favorite 화면 구성	
	Recent, Recycle Bin 화면 구성	
	로그인, 회원가입 화면 구성	김재연
Back-End (EC2, Django, MongoDB, S3, Cognito)	Cognito 및 사용자 인증 설정	강형구
	로그인, 회원가입 API 개발	
	파일 조회, 회원 조회 API 개발	
	파일 업로드&다운로드 API 개발	
	EC2 및 Load Balancer 설정	정재혁
	MongoDB 설정 및 삭제파일 조회 API 개발	
	파일 삭제, 파일 복구 API 개발	
	S3 설정 및 Django 와 연동	최지우
	즐거찾기 추가&제거 API 개발	
	즐거찾기 조회, 파일 영구 삭제 API 개발	
	파일경로 변경, 파일이름 변경 API 개발	김재연
	EC2 의 Request 부하 테스트	

## ◆ [Appendix] User Manual

### 1. 회원가입 및 로그인 화면

#### A. 로그인 화면

ThrowBox LOGIN

Log In to Throw Box

1 ID  
This field is required

Password  
This field is required

2 LOGIN

3 SIGN UP

ThrowBox 웹사이트의 로그인 화면이다. 1 번에서 보이는 폼에 ID, Password 를 입력 시 2 번에서 보이는 버튼이 활성화되어 로그인할 수 있다. 잘못된 ID 나 Password 를 입력 시 Dialog 로 에러 메시지가 출력된다. 3 번에서 보이는 탭은 로그인 화면, 회원가입 화면을 전환하는 기능을 한다.

#### B. 회원가입 화면

ThrowBox LOG IN

Sign Up to Throw Box

1 ID  
This field is required 0 / 15

Password 0 / 30

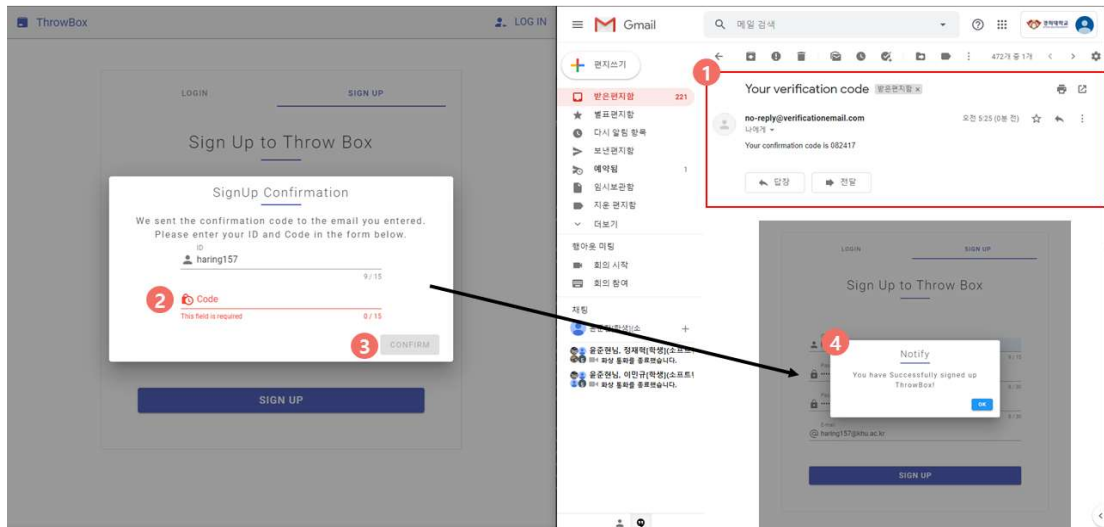
PasswordConfirm 0 / 30

E-mail 0 / 30

2 SIGN UP

ThrowBox 의 회원가입 화면이다. 1 번에서 보이는 폼의 각 항목에 조건에 맞는 올바른 정보를 입력 시 2 번 버튼이 활성화되어 회원가입이 가능해진다. 각 항목에 잘못된 정보를 입력 시, 버튼이 비활성화되고 Textfield 아래에 에러메시지가 등장한다. 중복된 아이디나 Email 을 입력 시, Dialog 에 에러메시지가 출력된다.

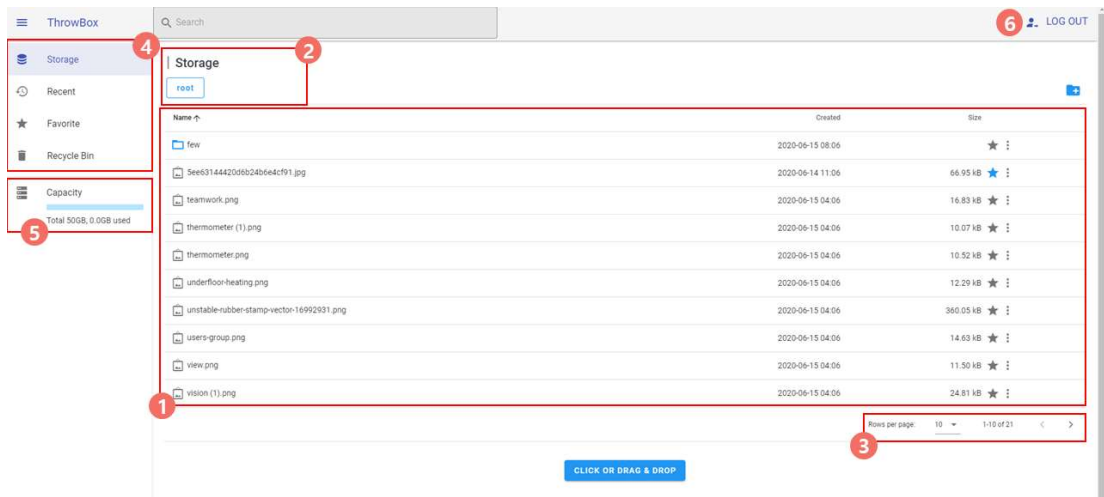
## C. 메일인증 화면



회원가입 화면에서 SIGN UP 버튼을 누르면 메일인증 화면으로 이동한다. 회원가입 폼에서 입력한 Email 로 1 번에서 보이듯이 인증메일이 발송되며, 2 번의 Textfield 에 해당인증 코드를 입력하면 3 번의 버튼이 활성화되며 올바른 인증코드를 입력할 시 4 번의 Dialog 와 같이 안내 메시지가 출력된다. 잘못된 인증코드를 입력하면 안내메시지 대신 에러메시지가 출력된다.

## 2. Storage Page 및 Data Table 공통 화면

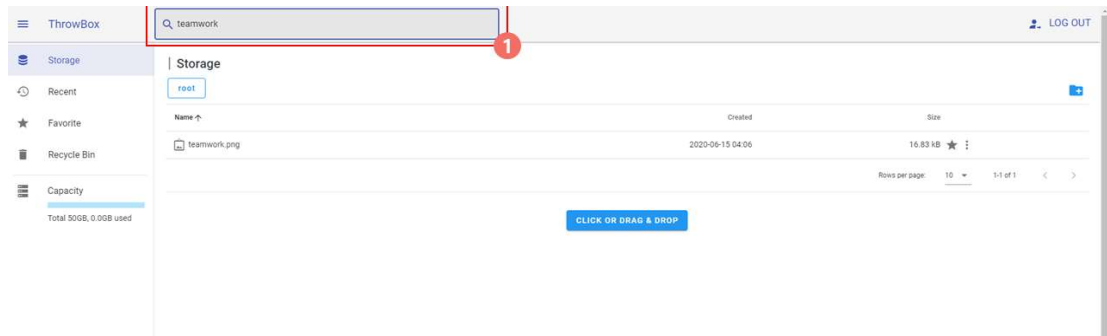
### A. Storage Page 및 Data Table 공통 화면



ThrowBox 의 Storage Page 화면이다. 1 번 박스에서 보이는 Data Table 에서 현재 접속한 회원이 ThrowBox 에서 업로드한 파일 및 폴더 목록을 볼 수 있다. Data Table 상단에 보이는 Name, Created, Size 를 누르면 이름, 생성시간, 크기별로 파일을 정렬할 수 있다. 2 번 박스에서는 현재 접속한 페이지 이름과 현재 ThrowBox 의 경로를 확인할 수 있다. 폴더를 타고 들어가면 경로가 변경된다. 3 번 박스에서는 페이지네이션 기능을 구현한 것을 볼 수 있고 현재 경로의 파일 수와 페이지 수 등을 확인 할 수 있다. 4 번

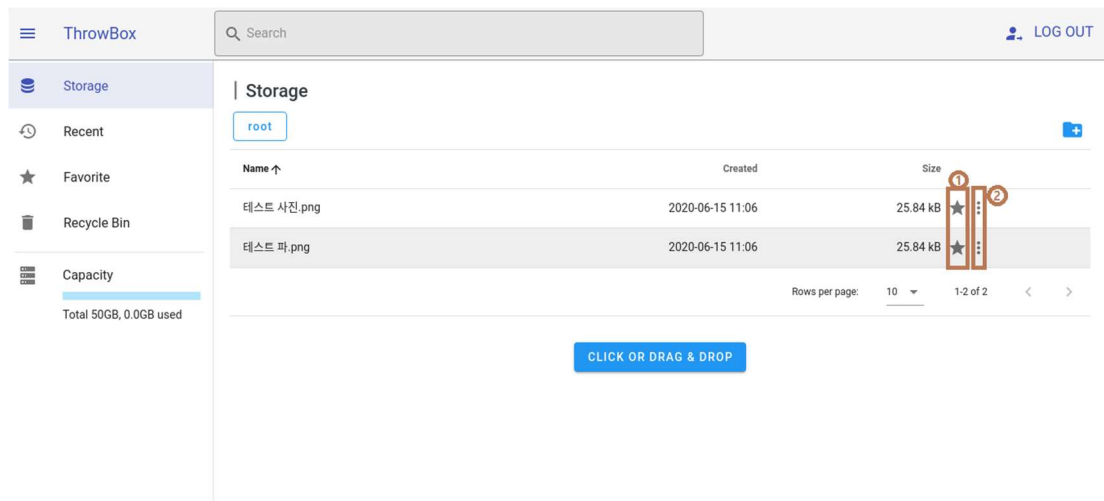
박스에서는 이동 가능한 페이지를 볼 수 있으며, 5 번 박스에서는 현재 접속한 회원이 ThrowBox 에 업로드한 전체 파일의 용량을 확인할 수 있다. 6 번의 LOGOUT 버튼을 클릭 시 로그아웃이 진행된다.

## B. Search 실행 화면

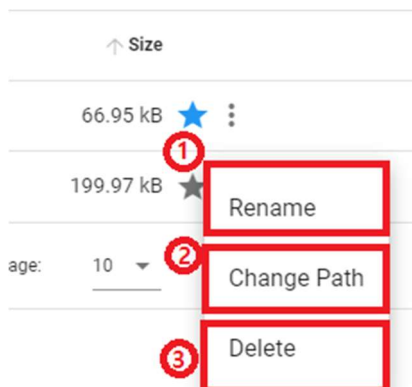


1 번의 Textfield 에 검색하고자 하는 파일의 이름을 입력 시, 해당하는 파일만이 Data Table 에 출력된다. 파일 형식을 입력 시, 해당하는 파일 형식만이 Data Table 에 출력된다.

## C. 즐겨찾기 버튼 및 파일 수정 Menu

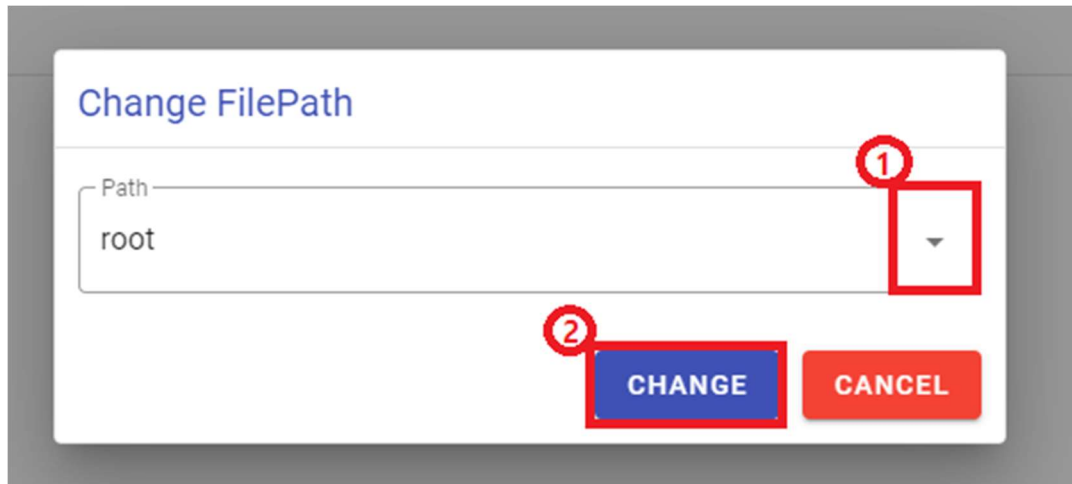


파일 조회 화면에서 1번 파일의 별표 표시를 클릭하여 즐겨찾기를 설정/해제할 수 있다. 별표 표시가 된 파일들은 즐겨찾기 조회 화면에서 빠르게 접근할 수 있다. 2번의 각 파일의 세로 말 줄임표 버튼을 클릭하면 아래와 같은 선택창을 띄울 수 있다.



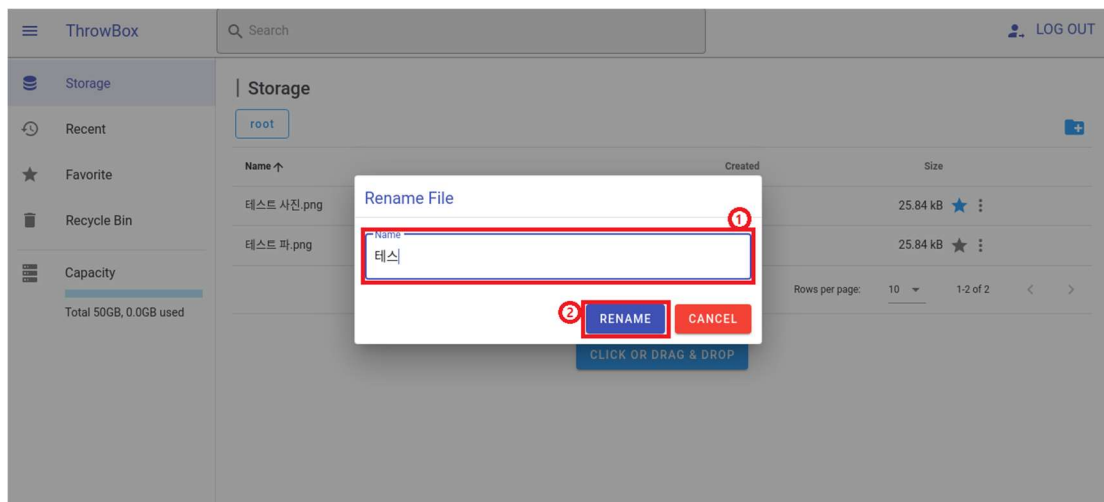
1번 Rename 버튼을 클릭하여 파일의 이름을 변경할 수 있으며, 파일 이름 수정 화면으로 연결된다. 2번 Chanage Path 버튼을 클릭하여 파일의 경로를 변경할 수 있으며, 파일 경로 수정 화면으로 연결된다. 3번 Delete 버튼을 클릭하여 파일을 휴지통으로 이동시킬 수 있다.

#### D. 파일 경로 수정 화면



파일의 세로 말줄임표 클릭 후 Change Path를 선택하면, 파일의 경로를 변경할 수 있다. 1번 버튼을 클릭하면 변경할 수 있는 디렉토리의 목록이 펼쳐지며, 위치할 디렉토리를 선택한 후 2번 CHANGE 버튼을 클릭하면 파일의 경로가 변경된다. CANCEL 버튼을 클릭하면 작업이 취소된다.

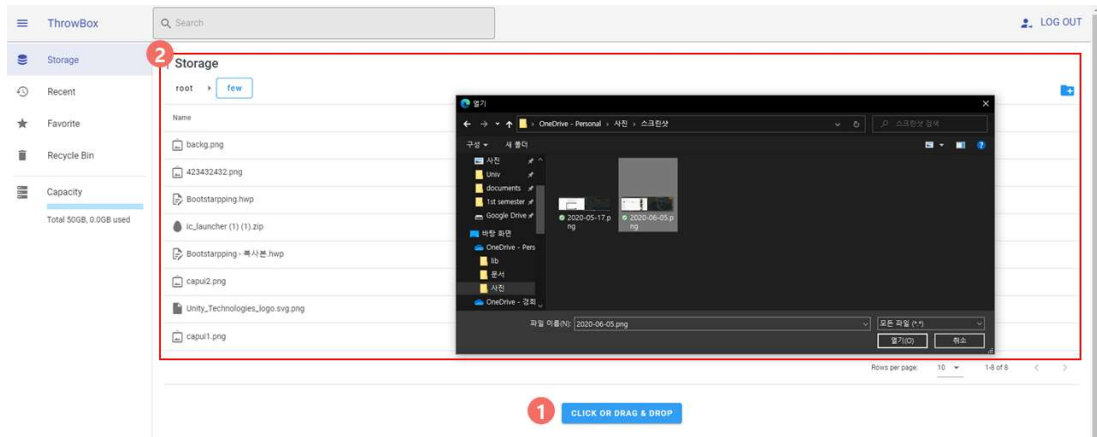
#### E. 파일 이름 수정 화면



파일의 세로 말 줄임표 클릭 후 Rename을 선택하면, 파일의 이름을 변경할 수 있다. 1번 필드에 변경할 파일의 이름을 입력하고, 2번의 RENAME 버튼을 클릭하면 파일의 이름이 변경된다. CANCEL 버튼을 클릭하면 작업이 취소된다.

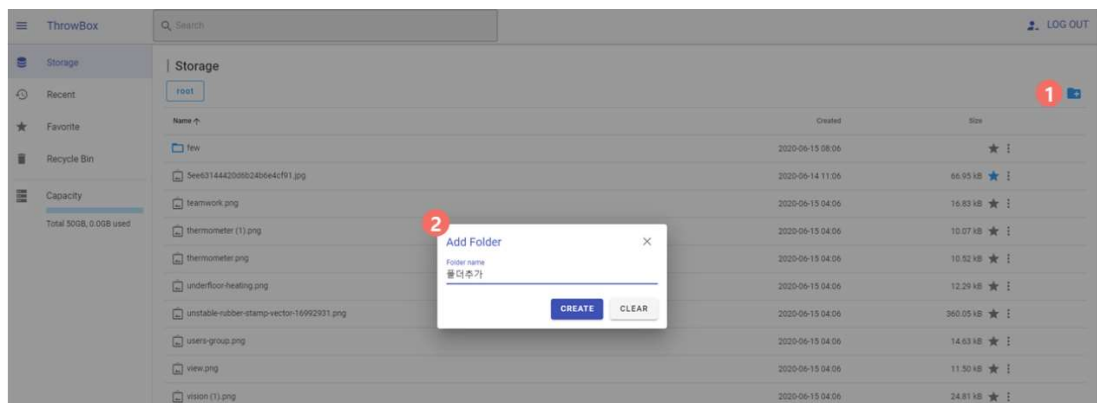


## F. 파일 업로드 화면



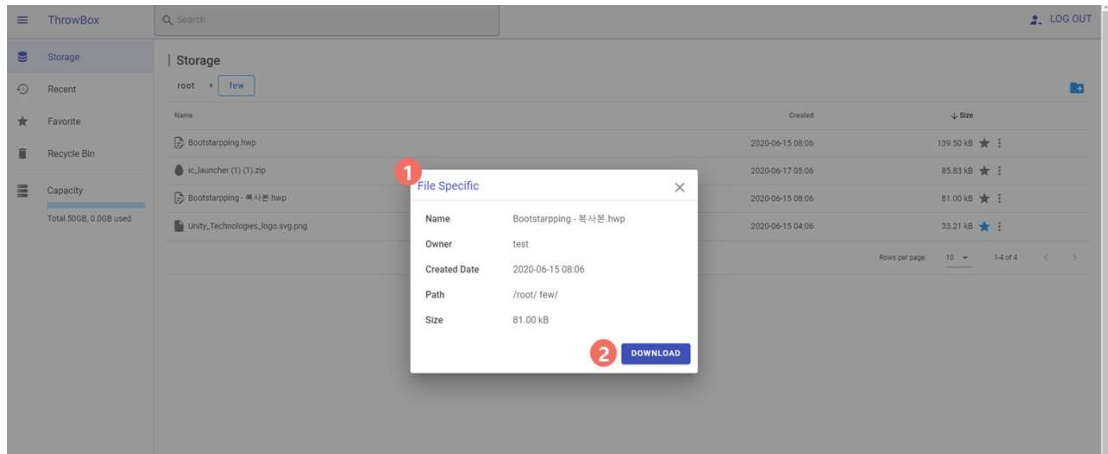
1 번의 파일 업로드 버튼을 클릭 시, 위의 사진과 같이 파일 업로드 창이 등장하며 파일 업로드 시, 로딩을 거쳐 로딩이 완료되면 Data Table 에 업로드한 파일이 표시된다. OS의 FileSystem에서 파일을 Drag 하여 2 번 영역에 Drop 할 때에도 파일이 업로드 된다. 멀티 파일 업로드를 지원하기 때문에, 여러 파일들을 한번에 업로드 또한 가능하다.

## G. 폴더 업로드 화면



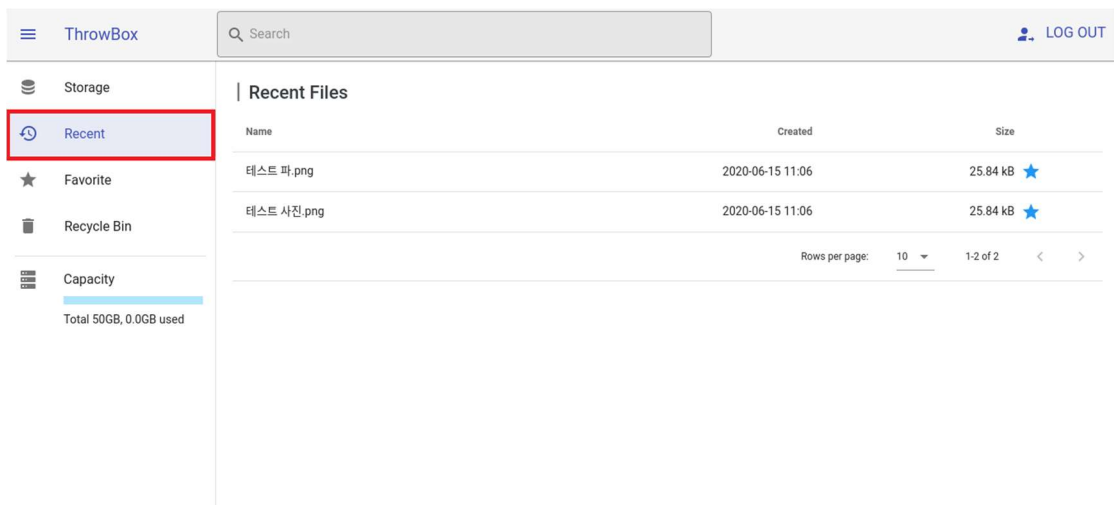
화면에 보이는 1 번의 폴더모양 버튼을 클릭 시, 2 번의 Dialog 가 등장한다. Textfield 에 업로드할 폴더 이름을 지정하고 Create 버튼을 클릭 시, 해당 폴더가 ThrowBox 에 업로드 되어 Data Table 에 추가된다.

## H. 파일 상세보기 화면



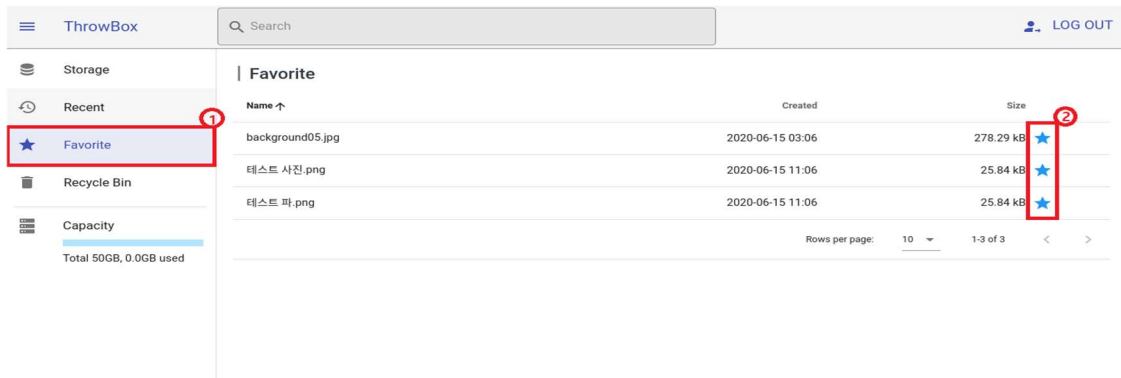
Data Table 에서 각 파일을 더블 클릭하면 위의 화면과 같이 파일의 상세정보가 담긴 Dialog 가 등장한다. 이 화면에서 2 번 Download 버튼을 클릭 시, 해당 파일이 Local Filesystem 에 저장하게 된다. 폴더를 더블 클릭시에는 상세화면이 아닌 해당 경로로 이동하게 된다.

## 3. Recent Page 화면



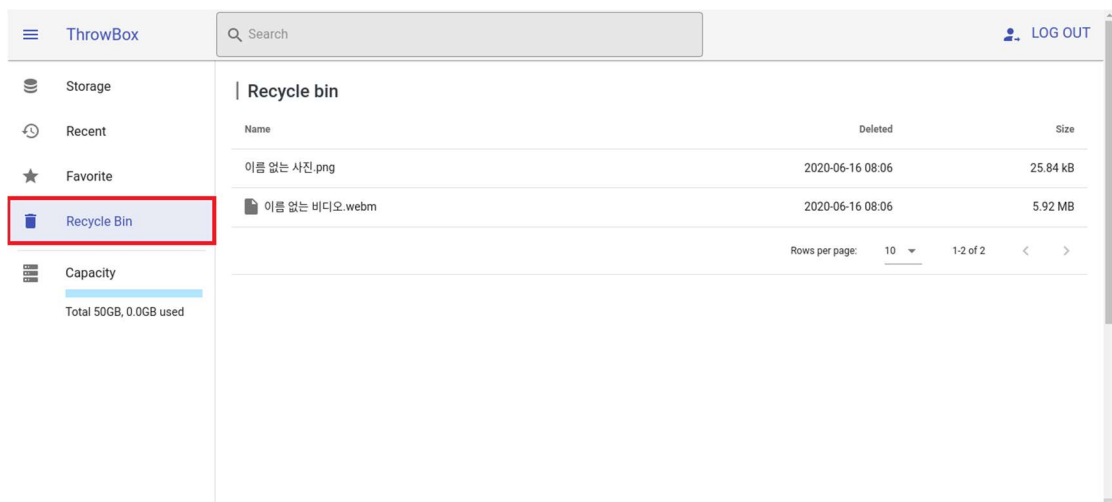
• 메뉴의 Recent 버튼을 클릭하면 최근에 업로드한 파일들을 조회할 수 있다.

#### 4. Favorite Page 화면



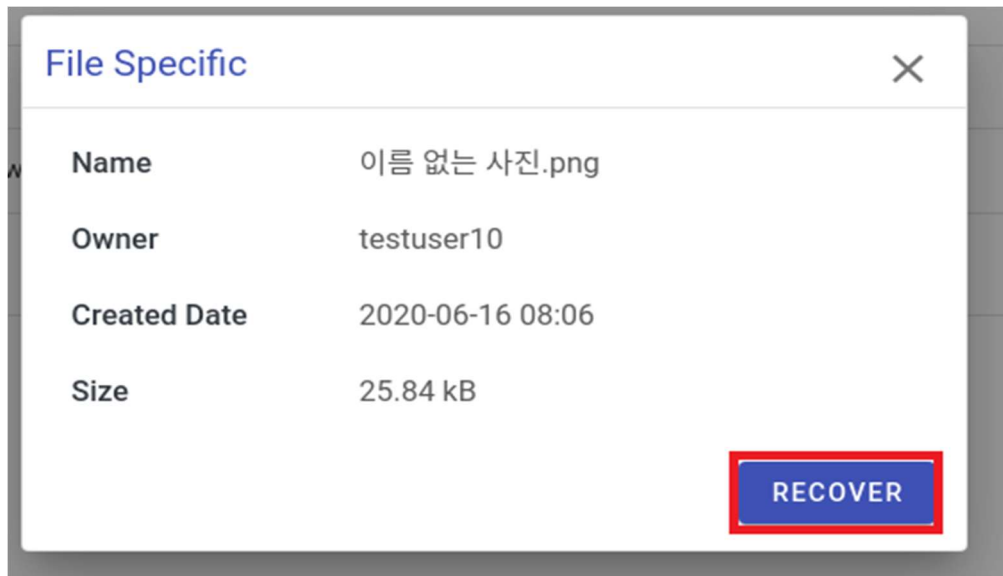
메뉴의 1번 Favorite 버튼을 클릭하면 즐겨찾기를 설정한 파일들을 모아서 조회할 수 있다. 이 화면 내에서는 2번 별표 표시 클릭을 통해 즐겨찾기 해제가 가능하다.

#### 5. Recycle Bin 화면






메뉴의 Recycle Bin 버튼을 클릭하면 삭제했던 파일들을 조회할 수 있다. 파일을 클릭하여 복원 기능에 접근할 수 있다. 삭제한 파일은 휴지통에서 30일간 유지되며, 이후에는 자동적으로 영구 삭제되며, 휴지통에서 조회할 수 없다.

## 6. File Recover 화면







휴지통에 있는 파일을 클릭하여 복원 기능에 접근할 수 있다. RECOVER 버튼을 클릭하면 기존에 파일이 위치하던 경로로 복원된다.

## ◆ [Appendix] 팀원 별 남은 Credit 및 ThrowBox URL


 <b>Active</b> full access ( krkdgud9@khu.ac.kr )
 <b>\$89.05</b> remaining credits (estimated)
 <b>2:60</b> session time
<a href="#">Account Details</a> <a href="#">AWS Console</a>


 <b>Active</b> full access ( kimgeni1@khu.ac.kr )
 <b>\$86.96</b> remaining credits (estimated)
 <b>2:60</b> session time
<a href="#">Account Details</a> <a href="#">AWS Console</a>



**Active**  
 full access ( eraser1121@khu.ac.kr )




**\$100**  
 remaining credits (estimated)




**2:60**  
 session time

Account Details


AWS Console



**Active**  
 full access ( haring157@khu.ac.kr )




**\$100**  
 remaining credits (estimated)




**2:60**  
 session time

Account Details


AWS Console



**Active**  
 full access ( yomapi@khu.ac.kr )



**\$84.86**  
 remaining credits (estimated)



**2:60**  
 session time

Account Details

AWS Console

	강형구	김재연	윤준현	정재혁	최지우	사용량
Credit	\$ 89.05	\$ 86.96	\$ 100	\$ 84.86	\$ 100	\$ 39.13

ThrowBox URL

<http://throwbox-elb-332262368.us-east-1.elb.amazonaws.com:8080/signin>