**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| **프로젝트 명** | 구름 USB |
| **팀 명** | 물을타지않은맥주클라우드USB 팀 |
| **문서 제목** | 구름 USB 계획서 |

|  |  |
| --- | --- |
| **Version** | 1.4 |
| **Date** | 2017-03-15 |

|  |  |
| --- | --- |
| **팀원** | 채 한울 (조장) |
| 이 창현 |
| 송 재영 |
| 진 희상 |
| 이 정화 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “구름 USB”를 수행하는 팀 “물을타지않은맥주클라우드USB”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “구름 USB”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 수행계획서-구름USB.doc |
| **원안작성자** | 전원 |
| **수정작업자** | 전원 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2017-03-07 | 전원 | 1.0 | 최초 작성 | 계획서 초안 작성 |
| 2017-03-08 | 전원 | 1.1 | 계획서 수정 | 계획서 내용 보완 |
| 2017-03-09 | 전원 | 1.2 | 계획서 수정 | 계획서 내용 수정 |
| 2017-03-12 | 이정화 | 1.3 | 계획서 보완 | 목차 수정 및 글 통일 |
| 2017-03-14 | 이정화 | 1.4 | 계획서 수정 | 계획서 내용 수정 |
| 2017-03-15 | 전원 | 1.6 | 계획서 보완 | 내용 검증 |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[**1**](#_gjdgxs) [개요](#개요) 4

[1.1 프로젝트 개요 4](#_30j0zll)

[1.2 추진 배경 및 필요성 4](#추진배경)

[1.2.1 저장매체 시장현황](#저장매체시장현황)

[1.2.2 기존 USB의 장점](#기존usb의장점)

[1.2.3 기존 USB의 단점](#기존usb의단점)

[1.2.4 기존 클라우드 서비스의 장점](#기존클라우드서비스의장점)

[1.2.5 기존 클라우드 서비스의 단점](#기존클라우드서비스의단점)

[1.3 유사 서비스 제공 제품](#유사서비스제공제품) 4

[**2**](#_3znysh7)**[개발 목표 및 내용](#개발목표및내용)**

[2.1 목표 5](#목표)

2.2 [구름 USB소개](#구름usb소개)

2.3 [구름 USB의 장점](#구름usb의장점)

[2.4 연구 / 개발 내용 7](#연구개발내용)

[2.4.1 작동 순서 개요 7](#작동순서개요)

[2.4.2 작동 순서 설명 7](#작동순서설명)

2.4.3 [요소별 세부내용](#요소별세부내용)

2.5 [개발결과](#개발결과)

2.5.1 [시스템기능 요구사항](#시스템기능요구사항)

[2.5.2 시스템 비기능(품질) 요구사항](#시스템비기능품질요구사항)

[2.6 기대효과 및 활용방안 7](#기대효과및활용방안)

[**3**](#_2s8eyo1) [**배경기술**](#배경기술)

[3.1 기술적 요구사항 8](#기술적요구사항)

[3.2 현실적 제한 요소 및 그 해결 방안 9](#현실적제한요소및그해결방안)

[3.2.1 하드웨어 9](#하드웨어)

[3.2.2 소프트웨어 9](#소프트웨어)

[3.2.3 추가기능 9](#추가기능)

[**4**](#_1ksv4uv) [**프로젝트 팀 구성 및 역할 분담**](#프로젝트팀구성및역할분담)

[**5**](#_3whwml4) [**프로젝트 비용**](#프로젝트비용)

[**6**](#_2bn6wsx) [**개발 일정 및 자원 관리**](#개발일정및자원관리)

[6.1 개발 일정 11](#개발일정)

[6.2 일정별 주요 산출물 12](#일정별주요산출물)

[6.3 인력자원 투입계획 13](#인력자원투입계획)

6.4 [비 인적자원 투입계획 14](#비인적자원투입계획)

[**7**](#_46r0co2) [**참고문헌 및 사이트** 30](#참고문헌)

**1****. 개요**

**1.1 프로젝트 개요**

세계는 지금 IT 기기 없이는 더 이상 살 수 없을 만큼 개개인의 삶과 IT기기간의 연관성이 높아지고 있다. 길거리를 지나다니면서 스마트폰 하나쯤 들지 않은 사람이 없고, 음식점에 가도, 영화관에 가도, 심지어 집에서 휴식을 취하면서도 우리는 크고 작은 IT기기들 사이에서 살아가고 있다. 이러한 사회에서 가장 눈에 띄고 체감하기 쉬운 여러 IT 서비스가 존재하지만 우리 팀은 그중 단순히 어플리케이션 개발을 하기보다는 하드웨어와 융합된 새로운 형태의 플랫폼을 제공하는 것이 어떨까 고민하던 끝에, 가장 오랫동안 흔하게 사용되는 저장매체인 USB와 최근 가장 유용하게 사용되고 있는 CLOUD서비스에서 장단점을 발견하였다. 둘의 장점을 융합한 구름USB를 개발해 보기로 결정했고 이에 대한 시장 조사를 진행했다.

**1.2추진 배경 및 필요성**

**1.2.1 저장매체 시장현황**

저장장치 시장은 지금까지 더 작게, 더 많이 라는 기술적 요소에 첨단을 달려왔다. 이에 따라 하드웨어와 USB의 크기는 더 이상 작아질 수 없을 만큼 작아졌고 용량은 꾸준히 증가하고 있다. 반도체 집적회로의 성능이 18개월마다 2배씩 증가한다는 무어의 법칙을 훨씬 뛰어넘어, 2002년 USB가 상용화 된 이후 15년동안 1만배 이상의 발전을 이루어 냈다. 하지만 아이러니하게도 빠르게 증가하고 있는 메모리 집적 기술이 제공하는 USB의 용량에도 불구하고 사람들이 실제로 사용하는 메모리는 언제나 한계를 맞이한다. 이러한 현실 속에서 우리는 USB가 발전할 새로운 방향을 제시해 보고자 하드웨어면서 용량의 제한에서 벗어나 가상의 온라인과 연결된 저장공간을 제공하는 USB를 만들어 보기로 결정하였다.

**1.2.2 기존USB의 장점**

1. USB는 현재 가장 널리 쓰이는 보편적인 저장장치이다.

- 마우스와 무선랜 장치 등 대부분의 수 많은 it기기가 USB단자를 사용하도록 하는 국제 협약이 체결 된 이후, USB는 현재 가장 널리 쓰이는 보편적인 저장장치다. 현재 대부분의 사람들은 자신이 소유한 파일을 다른 장소에서 사용하기위해 USB를 이용한다. 간혹 메일 또는 다른 방법을 이용하기도 하지만 용량이 GB 이상이라면 USB를 사용하지 않는 경우가 드물다.

2. 대부분의 데스크탑/노트북 뿐만 아니라 TV/자동차등에서도 USB를 사용 할 수 있다.

- USB는 사용자 OS가 설치 되어있지 않아도 이용 할 수 있다. 데스크탑 뿐 아니라 TV/가전기기 등 주변기기에서도 USB단자를 지원하기 때문에 USB에 동영상파일이 들어 있다면 TV에서 동영상을 실행할 수 있고, 음악파일이 들어 있다면 자동차를 운전하면서 음악을 들을 수 있다.

### **1.2.3 기존 USB의 단점**

1. 용량이 큰 파일들을 다루게 되면서 용량제한의 문제가 존재한다.

- USB는 용량이 보통 수 GB정도이다. 기존에 널리 쓰이던 저장매체(플로피디스크, CD)에 비하여 저장용량이 크지만 앞으로 사람들이 다루는 파일의 용량 또한 점점 더 커 질것이므로 USB가 사용자에게 충분한 크기의 저장공간을 계속해서 제공해줄 수 있을 것이라 장담할 수 없게 되었다. USB의 저장공간이 부족하면 용량을 확보하기위해 기존파일을 지워야 한다. 이러한 용량의 단점을 보완하기 위해 저장용량이 좀더 큰 TB단위의 USB/외장하드가 존재하지만, 저장공간이 부족해질 때마다 새로운 USB를 구매하고 필요한 파일을 옮기는데 들어가는 사용자의비용이 부담될 것이다.

2. 파손/분실 시 정보유실과 개인정보가 새나가는 등 보안문제가 발생한다.

- USB는 분실/파손의 위험이 있다. 사용자의 실수로 USB가 망가지면 파일이 유실됨은 물론이고, 분실했다면 그 USB를 타인이 취득하여 내부의 개인정보를 악의적으로 사용할 염려가 있다. 휴대성을 위해 크기가 작아진 근래의 USB는 그때문에 오히려 분실의 위험이 증가했다고 할 수 있다.

### **1.2.4 기존 클라우드 서비스의 장점**

1. 저장공간의 제약이 덜하다. (부족하면 추가신청으로 늘릴 수 있다.)

- 클라우드 서비스는 USB와 저장공간이라는 개념은 같지만 파일을 사용자기기의 내부에 저장하는 것이 아닌 외부서버에 저장한다. 클라우드의 장점은 저장공간이 비교적 덜 제한적이라는 것이다. 기존의 저장매체에서는 저장공간이 부족할 때 기존에 존재하던 다른 파일을 삭제하거나 USB자체를 좀더 큰 용량으로 바꾸면서 전체파일을 이동시켜야 했다. 그에 비해 클라우드 서비스는 현재 사용하던 저장공간이 부족하면 추가신청을 통해 용량을 동적으로 확장하는 것이 가능하다.

2. 정보의 파손/분실 위험이 없다.

- 클라우드 서비스는 파일을 외부에 저장하기 때문에 기존의 휴대용 저장매체와 달리 사용자의 부주의에 의한 분실/파손의 위험이 없다. 보안을 책임지는 정보관리주체가 클라우드 서비스 제공자로 이관되므로 사용자는 보안이슈로부터 자유로워질 수 있다

### 

### **1.2.5 기존 클라우드 서비스의 단점**

1. TV등 사용자OS가 깔려 있지 않은 제품에서는 이용 할 수 없다.

- 클라우드 서비스는 USB와 같은 저장공간이지만 서로 다른 특징을 갖고있다. USB단자만 있으면 사용가능한 USB와 달리, 클라우드 서비스는 TV등 사용자OS가 설치 되어 있지 않은 기기에서는 클라우드 서비스를 이용할 수 없다.

2. 로그인 등 이용하기 위한 기타 절차가 귀찮으며 ID/PW분실 우려가 있다.

- 클라우드 서비스를 이용하려면 브라우저 또는 전용 앱을 통한 로그인이 필요하다. 어디에서나 단자에 연결만 하면 파일이 보여지는 USB와 달리 로그인과정을 반드시 거쳐야한다. 이 로그인 절차가 사용자에게 귀찮게 느껴질 수 있으며 ID/PW를 분실하면 난감한 상황이 생길 수도 있다.

3. 타인의 컴퓨터에서 클라우드에 액세스하면 로그인 흔적이 남는다.

- 안전하지 않은 장소에서 클라우드 서비스에 로그인 해야할 수도 있다. 이때 로그인 흔적이 남아 클라우드에 저장된 개인정보가 새나 갈수 있다.

4. 컴퓨터 사용에 익숙하지 않은 IT취약계층은 사용이 어렵다.

- IT환경에 익숙하지 않은 IT취약계층에게는 클라우드 서비스라는 개념이 난해할 수 있다. 이러한 사용자들에게는 접근성이 비교적 떨어 질것이다.

**1.3 유사 서비스 제공 제품**

**1. 크롬 캐스트**

크롬 캐스트(Chromecast)는 구글에서 만든 멀티미디어 스트리밍 어댑터이다. 이 장치는 2인치(5.1cm) 동글이며 HDTV의 HDMI포트에 꽂아 오디오나 비디오를 와이파이를 통해 수신해 TV에서 스트리밍 재생한다. 크롬 캐스트는 HDMI 1.4+ 포트와 USB 포트, 전원 어댑터로부터 전원을 공급받으며 구글 크롬 OS의 심플 버전을 탑재하고 있다.

사용자는 PC나 모바일 장치에서 구글 크롬 인터넷 브라우저를 이용하거나, 모바일 장치에서 크롬 캐스트를 지원하는 모바일 애플리케이션을 이용할 경우 그 콘텐츠들을 크롬 캐스트가 장착된 TV로도 즐길 수 있게 된다. 크롬 캐스트를 지원하게 될 앱은 유튜브, 넷플릿스, 구글 플레이 뮤직, 구글 플레이 무비&TV이다.

크롬 캐스트는 다수의 플랫폼과 운영체제 상에서 동작한다. 자세하게는 안드로이드, IOS, 크롬 OS 등의 OS에서 동작하며, 그 밖에 윈도나 OS X상에서는 구글 크롬을 통해 동작한다.

**2. 이지 캐스트**

이지 캐스트(EZcast)는 TV, 프로젝터, LCD 장치의 HDMI 디스플레이에 사용하는 어댑터이다. 대표적인 크롬 캐스트 카피제품으로 확장성은 더 뛰어나다. 크롬 캐스트와는 다르게 화면 자체 미러링이 가능해서 스마트폰 게임이나 다운로드 받은 영상 재생도 전부 가능하다.

**3. WD 마이 클라우드(서버 역할 장치를 직접 구매, 관리)**

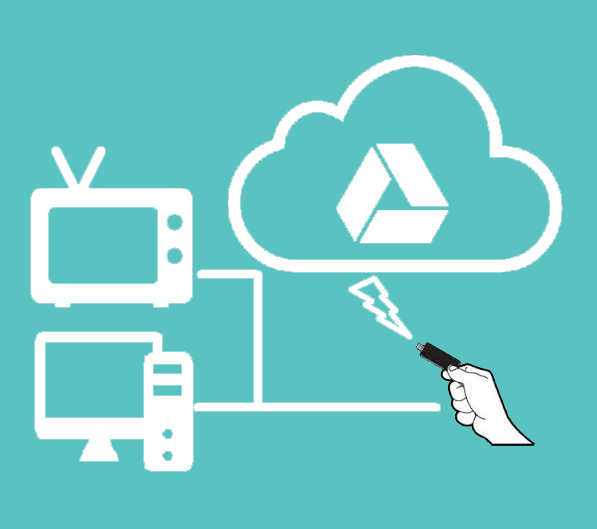
WD 마이 클라우드는 홈 네트워크에 연결되는 안정적인 듀얼 드라이브 퍼스널 스토리지를 제공한다. 원하는 모든 컨텐츠를 장치에 구애 받지 않고 인터넷 연결이 가능한 곳이라면 어디서나 액세스하고 공유할 수 있다. 두 개의 하드 드라이브가 장착되어 있어 첫 번째 드라이브에 저장된 대체 불가능한 컨텐츠가 두 번째 드라이브에 자동 복제되어 모든 컴퓨터에 대한 자동 백업 기능을 제공한다.

**2. 개발 목표 및 내용**

**2.1 목표**

기존 USB와 클라우드 서비스 각각의 장점을 합쳐 서로의 단점을 보완하는 새로운 방식의 저장매체를 제안한다.

**2.2 구름 USB 소개**



구름 USB는 구글 드라이브와 연동되는 새로운 방식의 USB이다. 사용자는 구글 드라이브에 파일을 넣어두면, 구름 USB를 이용하여 구글 드라이브 안에 있는 파일에 접근할 수 있다. 구글 드라이브에 있는 내용을 실행하면, 구름 USB는 구글 드라이브에 있는 파일을 실시간으로 다운로드 하여 실행시켜준다. 반대로 사용자가 구름 USB내의 파일을 바꾸면, 구글 드라이브에 있는 파일 역시 바뀌게 된다. 즉, 구름 USB와 구글 드라이브는 항상 같은 파일을 가지고 있음이 보장된다. 구름 USB는 두 플랫폼을 연동함으로써 사용자들이 자신의 파일을 관리, 사용하는데 있어 편의성을 제공해준다.

### **2.3 구름USB의 장점**

1. 일반 USB와 똑같은 방법으로 이용할 수 있다.

- 구름 USB의 겉모습과 사용법은 일반USB와 동일하다. 구름USB는 사용자에게 일반USB와같이 내부에 파일이 저장된 것처럼 보여진다.

2. 기존 USB에 비해 저장공간의 제약이 적다.

- 실제 파일은 구글 클라우드에 저장되기 때문에 기존 USB보다 저장공간의 제약이 적다. 기존 USB에서 저장공간이 부족하면 기존에 있던 다른 파일을 삭제하거나 크기가 더 큰 새로운 저장장치를 구매해서 전체파일을 옮겨야 한다. 하지만 구름USB를 사용하면 클라우드 서비스를 이용하듯이 추가적인 신청으로 저장공간을 늘릴 수 있다.

3. 컴퓨터, 태블릿 뿐 아니라 TV, 자동차 등에서도 클라우드 서비스를 이용할 수 있다.

- 기존 클라우드 서비스는 TV등 사용자OS가 깔려 있지 않은 기기에서 사용할 수 없다. 하지만 구름USB는 USB단자를 지원하는 TV/자동차등에서도 사용할 수 있다. 요즘은 클라우드 서비스를 이용할 수 있는 TV도 등장했지만 해당 클라우드 서비스를 지원하는 플랫폼에 의존적이다. 구름USB는 USB단자가 있는 기기에서는 어디에서나 이용가능한 좀더 범용적인 저장 매체라고 할 수 있다.

4. 분실/파손으로 정보가 유실되거나 보안사고가 일어날 위험이 없다.

- 구름USB는 내부에 파일이 저장되지 않기 때문에 기존 USB가 가지고 있던 파손, 분실의 위험이 없다. 구름USB가 파손 또는 분실 되더라도 새로운 USB단말에 인증하면 기존의 구름USB처럼 사용할 수 있기 때문이다. 또한 구름 USB는 스마트폰 앱으로 허가를 받아야 클라우드 계정에 접근할 수 있도록 설정함으로써 타인이 사용하는 것을 방지할 수 있다.

5. 남의 컴퓨터에서 사용하더라도 클라우드 로그인 흔적이 남지 않는다.

- 기존 클라우드 서비스는 로그인 과정이 PC에서 이뤄진다. 타인의 PC에서 클라우드 서비스에 로그인한다면 로그인기록이 남아 보안사고가 발생할 우려가 있다. 그에 비해 구름USB를 사용하여 클라우드 서비스에 로그인하는 과정은 구름USB내부에서 일어난다. 따라서 타인의 PC에서 클라우드 서비스에 접근할 때 보안문제 발생여지가 개선된 것 또한 구름USB의 장점이다.

6. 사용자가 보안기능을 활성화하면 스마트폰 앱으로 허가를 받아야 구름USB를 이용할 수 있다(보안기능 강화).

- 기존 클라우드 서비스는 유저가 직접 로그인하는 과정을 거쳐야 한다. 하지만 사용자가 로그인과정을 귀찮게 여길수도 있으며 ID/PW를 잊었다면 난감한 상황이 발생할 수 있다. 구름USB는 클라우드 저장공간을 이용하면서도 유저가 직접 로그인할 필요없이 USB를 꽂기만 하면 저장된 파일을 그대로 보여준다.

7. IT취약계층도 쉽게 사용할 수 있다.

- 구름USB는 USB의 PLUG&PLAY장점을 도입했다. USB를 장치에 꽂기만 하면 클라우드 서비스 로그인 후 파일목록을 불러오는 작업까지 사용자의 별다른 조작없이 이뤄진다. 이러한 단순한 사용법으로 클라우드 서비스에 접근하기 힘든 IT취약계층이 쉽게 사용할 수 있다.

**2.4 연구 / 개발 내용**

**2.4.1 작동 순서 개요**

1. 구름 USB는 기기에 연결되었을 때 일반 USB와 같은 저장장치로 인식된다.

2. 사용자가 구름 USB를 HOST(컴퓨터, TV, 자동차 등)에 연결하면, 구름 USB는 자동으로

구글 드라이브에 접속하여 로그인과정을 거친다.

3. 구글 드라이브에 저장되어 있던 최상위 디렉토리/파일 목록을 가져온다.

4. HOST에게는 구글드라이브에서 가져온 디렉토리/파일 목록이 구름 USB내부에 존재하는 것처럼 보여진다.

5. FAT32를 통해 upload / download 등의 사용자 요구사항에 따른 정보를 실제 구글드라이브에서 가져와 사용자에게 파일을 제공한다.

6. 사용자의 요구에 따라 5번을 반복한다.

**2.4.2 작동 순서 설명**

1. 구름 USB 연결&인식

- 사용자가 구름 USB를 호스트(컴퓨터/TV 와 같은 장치)에 연결하면, 호스트는 구름 USB를 일반적인 USB로 인식한다. 하지만 구름USB는 일반USB와 다르게 작동한다. 일반적인 USB는 내부에 저장된 디렉토리와 파일목록을 HOST에 전달하지만 구름 USB는 구글 드라이브에 저장된 정보를 HOST에 전달해야 한다. 이것은 일반적인 HOST끼리의 연결로는 구현될 수 없으며 가젯 드라이버를 이용해 구름 USB가 호스트에게 저장 장치로 인식되도록 해야한다. 이를 통해 구름 USB는 어떤 호스트 장치에 연결되어도 저장장치로 인식된다.

2. 구글 드라이브 인증

- 일반적인 저장장치는 연결시 사용자에게 저장장치 내부의 파일과 디렉토리 목록을 보여준다. 하지만, 구름 USB는 아직 이 정보가 존재하지 않는다. 구름 USB는 이 정보를 구글 드라이브로부터 다운로드 받아야 한다. 구름 USB의 가젯 드라이버는 먼저 구글 드라이브에 접근하기 위해 구글 API를 이용해 구글 드라이브에 대한 사용자의 권한을 요청한다. 권한요청시 필요한 로그인 정보는 구름 USB 내부에 json파일로 미리 저장하여 매 접속 시 권한 요청이 자동으로 이루어 질 수 있도록 한다.

3. 구글 드라이브의 파일 목록

- 구름 USB가 구글 드라이브에 인증을 성공하면, 구글 드라이브 API를 이용하여

구글 드라이브에 있는 사용자의 파일을 다운로드 받을 수 있다. 구름 USB는 우선

사용자에게 루트 디렉토리를 보여주어야 한다. 그렇기 때문에, 실제 파일을 다운로드

받는 것이 아니고 파일들의 경로, 이름, 용량 등과 같은 메타 데이터를 다운로드 받아야한다.

또, 파일 뿐만 아니라 디렉토리 정보도 함께 가져와야 한다. 하지만 해당 계정의

구글 드라이브의 모든 파일, 디렉토리 정보를 다 받아 오진 않는다. 먼저 루트 디렉토리에

존재하는 파일, 디렉토리 정보만 구글 드라이브로부터 다운로드 받아 온다. 그 후

동기화 프로그램은 이 정보들을 이용해 FAT32 이미지를 구현한다.

4. 구글 드라이브에서 받은 파일 목록을 구름 USB안에 실제 존재하는 것처럼 보여준다.

- 동기화 프로그램은 위에서 받아온 메타데이터 정보들을 이용해 FAT32 이미지를 구현 해야한다. FAT32 이미지 구현을 위한 동기화 프로그램은 C언어로 이루어져 있다. 하지만 구글 드라이브 API는 C언어로 제공하지 않기 때문에 우선 Python으로 그 목록을 받아 FAT32 파일 시스템을 관리하고 있는 동기화 프로그램으로 전달해야 한다. 동기화 프로그램이 이 목록을 받으면 파일, 디렉토리에 대한 메타데이터를 FAT32 파일시스템 형식에 맞게 구현한다. 메타 데이터들만 생성을 하면 실제 파일의 내용은 비어 있지만 사용자에게는 구글드라이브의 파일들이 실제로 구름 USB에 담겨있는 것처럼 보인다.

5. 파일의 실행, 생성, 삭제 등 사용자 요청을 분석하여 수행한다.

- 이제 사용자에게 보이는 구름 USB의 파일, 디렉토리 목록은 일반 저장 장치와 똑같이 보인다. 하지만, 사용자에게 보여지는 파일들은 실제로 존재하지 않는 파일이다. 그렇기 때문에 사용자가 파일을 실행, 생성, 삭제 요청을 할 때의 작업 또한 구름 USB에서 내부적으로 따로 처리해주어야 한다. 요청하는 파일이 어떤 것인가를 FAT32 파일 시스템에 맞게 분석하여 해당하는 파일을 구글드라이브에 요청한다. 구름 USB가 구글 드라이브로부터 실제 파일을 받아오면 이를 호스트 장치로 전달한다. 구름 USB는 이러한 과정으로 사용자의 파일에 관한 요청을 처리할 수 있다.

**2.4.3 요소별 세부 내용**

**1. 구글 드라이브**

- 사용자의 요구가 있을 경우 구글 드라이브에 접속하여 아이디와 비밀번호를 관리하고 구글 드라이브의 파일 목록을 보여준다. 또한 사용자에 따른 upload / download 등의 요청사항들을 구글 드라이브를 통해 제공한다.

1) 구글드라이브 접속

- 웹사이트로 연결하여 사용자의 아이디와 패스워드를 입력 받는다. 로그인이 되면

구글 API를 이용해 사용자의 구글 드라이브 목록 권한요청을 요구한다. 권한요청

수락 시 사용자의 구글 드라이브 목록을 가져온다. 로그인 시 입력된 정보는 json파일로 저장하여 다음 번 접속 시 아이디와 패스워드를 입력하는 과정을 생략하고 업데이트 된 정보를 바로 제공한다.

2) 구글 드라이브 파일 목록

- 구글 드라이브 로부터 받아온 폴더 경로, 파일 이름, 용량, 파일 정보에 대한 것을 출력한다.

3) 구글드라이브 파일 목록 FAT 32에 전달

- 구글드라이브를 통해 받은 파일 목록을 FIFO라는 NAMED PIPE를 사용하여 FAT32에 전달한다.

4) FAT32를 통해 upload / download 등의 사용자 요구사항에 따른 정보를 실제 구글드라이브에서 가져와 사용자에게 파일을 제공한다.

**2. 가젯 드라이버**

- 가젯 드라이버의 역할을 다음과 같다.

1) 구름 USB가 호스트 장치에 연결 돼있을 때 대용량 저장 장치로 인식 되어야한다. 사실 구름 USB는 일반적인 대용량 저장 장치가 아니다. 원래는 라즈베리파이와 같은 호스트 장치이며, 컴퓨터와 같은 호스트 장치에 연결되었을 때, 호스트 대 호스트 관계로 연결된다. 그러므로 먼저 구름 USB가 연결되면, 호스트 장치가 구름 USB를 대용량 저장 장치로 인식하도록 해야한다. 이러한 관계는 호스트장치에서 결정하는 것이 아니고 구름 USB에서 결정해야 한다. 왜냐하면 구름 USB는 컴퓨터, TV 등 어느 호스트장치에 연결되어도 대용량 저장 장치로 인식 되어야 하기 때문이다. 구름 USB는 리눅스 기반의 장치인데, 이 연결방식을 위해 리눅스 커널의 가젯 드라이버를 이용한다.

가젯 드라이버는 장치가 호스트 장치에 연결되었을 때 어떤 장치처럼 동작할 것인지를 결정해주는 드라이버이다. 핸드폰에 USB가 꽂히면 이를 저장 장치로 인식하는 호스트로서 동작하고, 핸드폰이 컴퓨터에 연결되었을 땐 반대로 자신이 저장 장치로 인식된다. 그 이유가 바로 핸드폰의 가젯 드라이버가 어떤 장치와 연결되었는지에 따라 핸드폰의 동작 방식을 다르게 하기 때문이다. 구름 USB는 대용량 저장 장치로 동작하도록 해야하기 때문에 가젯 드라이버의 여러 종류 중 하나인 g\_mass\_storage을 이용한다. g\_mass\_storage는 리눅스 커널 모듈 형태로 제공된다. 구름 USB가 호스트 장치에 연결되면 리눅스가 부팅되는데 이 때마다 스크립트를 이용하여 g\_mass\_storage 모듈을 적재 시킨다. 그러면 호스트 장치는 구름 USB를 대용량 저장 장치로 인식하게 된다.

2) 호스트로부터 요청된 파일 접근 요청을 파일 동기화 프로그램으로 넘겨주어야 한다. 파일 동기화 프로그램은 구글 드라이브에 있는 파일, 디렉토리 목록을 다운로드 받아 가젯 드라이버로 넘겨주는 프로그램이다. 가젯 드라이버는 이 파일, 디렉토리 목록을 받아 실제 존재하는 파일, 디렉토리처럼 호스트 장치에 보여준다. 이를 위해, 파일 동기화 프로그램은 어떤 파일, 디렉토리가 호스트 장치로부터 요청됐는지 알아야 한다. 일반적인 대용량 저장 장치는 하드 디스크 또는 플래시 메모리에 파일 시스템 구조를 만들어 여러 파일, 디렉토리를 저장한다. 하지만 가젯 드라이버는 큰 단일 파일에 파일 시스템 구조를 만들어 여러 파일, 디렉토리를 저장한다. 이 때, 이 단일 파일을 파일 시스템 이미지라고 한다. 다시 말해, 호스트 장치로부터 어느 파일, 디렉토리가 요청됐는지 알아야 한다는 뜻은 파일 시스템의 어느 위치가 호스트 장치로부터 요청됐는지 알아야 한다는 뜻이다. 위치 정보를 파일 동기화 프로그램으로 넘겨 주면, 파일 동기화 프로그램은 이 위치에 해당하는 파일, 디렉토리를 구글 드라이브로부터 다운로드 받는다.

3) 구글 드라이브에서 받아온 파일, 디렉토리를 호스트 장치에 넘겨준다.

3-1디렉토리 요청인 경우

* 파일 동기화 프로그램이 구글 드라이브의 디렉토리 정보를 받아와 이를 가젯 드라이버로 넘겨준다. 가젯 드라이버는 2에서와 같은 방식으로 이 디렉토리에 존재하는 파일, 디렉토리 목록을 실제 존재하는 것처럼 보여준다.

3-2 파일 요청인 경우

* 파일 동기화 프로그램이 구글 드라이브의 실제 파일을 받아와 가젯드라이버로 넘겨준다. 가젯드라이버는 이 파일을 넘겨받아 호스트 장치로 넘겨준다. 이를 넘겨 받은 호스트 장치를 파일을 실행하게 된다.

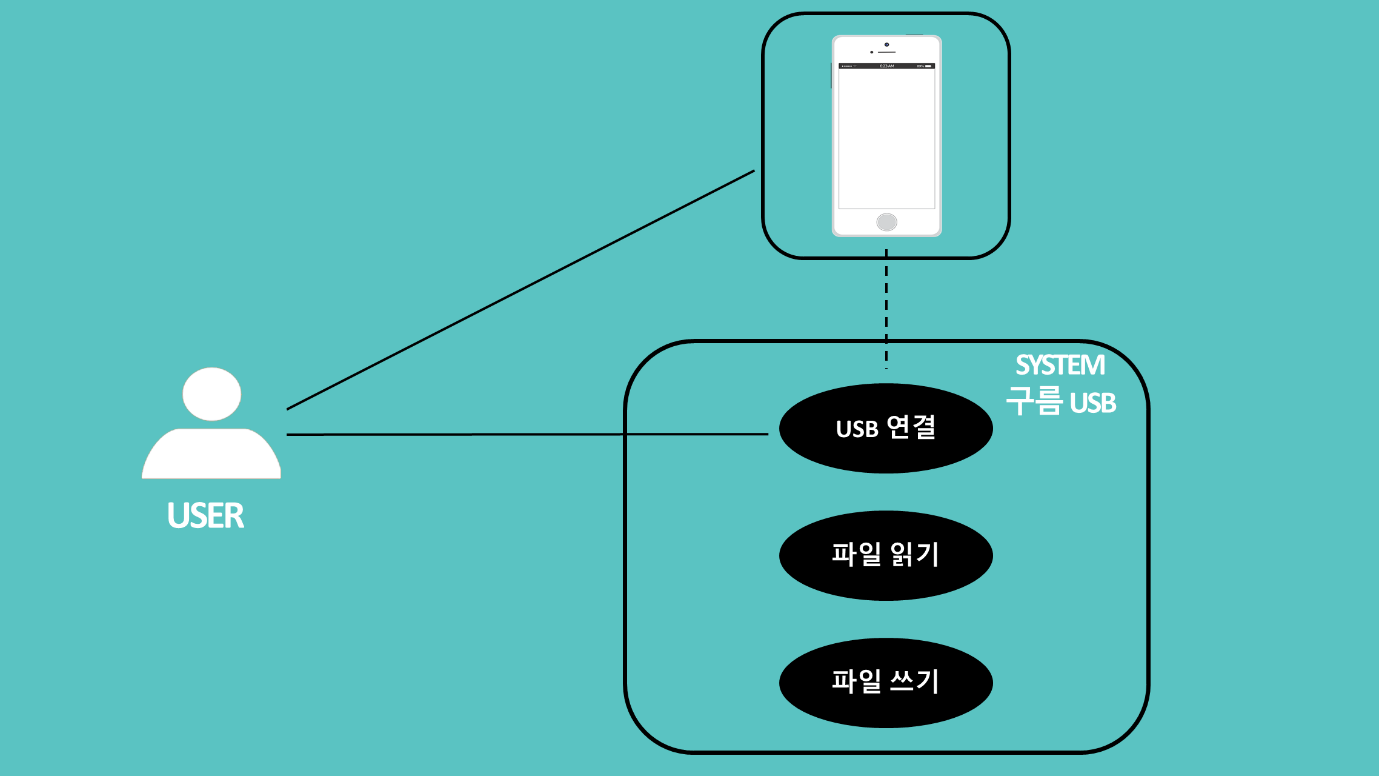
**3. FAT32 파일 시스템**

- FAT32, NTFS, EXT 등의 파일 시스템이 있지만 그 중 window, Linux, macOS 등 여러 종류의 OS에서 범용적으로 사용할 수 있고 구현이 쉬운 FAT32 파일 시스템을 선정했다. FAT32 파일 시스템은 한 파일당 약 4GB의 용량 제한이 있지만 클라우드 스토리지 파일들의 실제 내용은 FAT32 파일 시스템 이미지에 저장하지 않기 때문에 그 문제 또한 해결 가능하다.

FAT32 파일시스템의 역할은 가젯 드라이버가 라즈베리파이를 대용량 저장 장치로 보여줄 때에 사용되는 이미지를 만들어주고 클라우드 스토리지에 있는 파일들의 메타 데이터를 그 이미지에 써서 실제 파일이 USB에 존재하는 것처럼 보여준다. 보여 준 후에 사용자가 파일을 요청하면 그 신호(cluster 단위의 요청)를 분석하여 어떤 파일을 요청하는지 파악할 수 있도록 한다.

**2.5 개발결과**

**2.5.1 시스템 기능 요구사항**



1. 사용자는 구름 USB를 호스트장치(컴퓨터, TV)에 연결한다.

2. 구름 USB가 호스트장치에 연결되면, 구름 USB는 구글 드라이브로부터 파일 목록을

다운받아 이를 사용자에게 보여준다.

3. 사용자가 파일 목록 중 특정 파일에 대한 실행 요청을 하면, 구름 USB는 구글 드라이브에서

해당 파일을 다운 받아 실행한다.

4. 사용자가 파일 쓰기 요청을 하면, 해당 파일을 구글 드라이브에서 업로드 시킨다.

**2.5.2** **시스템 비 기능(품질) 요구사항**

1. 성능

USB를 꽂을 때 마다 Linux의 가젯 드라이버를 돌리기 위해 OS를 부팅 시켜야 하기 때문에 빠른 부팅이 필요하다.

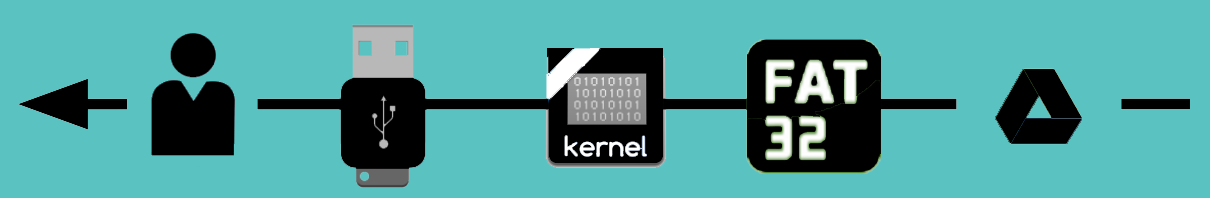
메모리 처리 속도에 비하여 상대적으로 느린 네트워크를 통해 파일을 주고 받기 때문에 효율적인 파일 관리가 필요하다. 이 시스템의 성능은 사용자의 읽기/쓰기/실행 요청에 대한 반응 속도가 기존의 USB와 큰 차이가 없어야 한다. 구름 USB와 구글 드라이브의 파일 동기화 속도는 오로지 네트워크 속도에 비례하며, 그 외의 내부적 요소에 관해서는 큰 영향이 없어야 한다.

본 프로젝트에서는 실질적인 성능을 가용성 보다 우선시 하였다.

2. 가용성

사용자가 시스템을 얼마나 편리하게 사용할 수 있는가에 대한 솔루션을 개발 해야 한다. 구글 드라이브에 접근하기까지의 과정을 축소시켜 사용자의 접근성을 편리하게 한다.

**2.5.3 시스템 구조**



**2.5.4 결과물 목록 및 상세 사양**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류** | **소분류** | **기능** | **형식** | **비고** |
| 내부 모듈 | 구글 드라이브 | 구글 드라이브 목록 및 파일 제공 |  |  |
|  | FAT 32 | 구글 드라이브 에서 받아온 목록 이미지 파일 생성,  구글 드라이브와 연동 |  |  |
|  | 가젯 드라이버 | 구름 USB의 저장장치 인식 기능, FAT 32와의 연동 |  |  |
| UI | 레이아웃 | User Interface 제공 |  |  |
|  | 연동 | 구글 드라이브와 연동 |  |  |
|  | 보안 | USB사용시 승인여부 |  |  |

## 

**2.6 기대효과 및 활용방안**

- 대부분의 사용자들이 각종 클라우드 서비스에 중요한 문서와 이미지를 저장하며 사용한다. 클라우드 서비스를 이용하기 위해서는 pc나 스마트폰이 필요하다는 단점이 있으며 IT취약 계층(미취학 아동, 노년층)의 경우 사용하기 어려운 점이 존재한다. 또한 대다수의 사람들은 USB를 사용하고 있다. 하지만 USB는 보안/분실에 취약하기 때문에 잃어 버릴 경우 백업하지 않는 이상 내용물을 찾기가 어렵고 개인정보가 담긴 문서라면 개인정보의 유출 가능성 또한 존재한다. 따라서 이러한 취약한 부분을 보완하고자 두가지를 결합 한 구름 USB를 생각하게 되었다. cloud와 USB의 단점은 보완하고 장점을 살린 구름 USB는 사용자들에게 이러한 점을 시사한다.

1) 기대 효과

(1) 아이디와 비밀번호를 매번 입력하는 번거로움이 없다.

(2) 클라우드 용량을 활용하여 USB 용량 부족 현상을 해결 할 수 있다.

(3) USB가 파손 될 경우 대비가 가능하다.

(4) USB가 분실 될 경우 대비가 가능하다.

(5) 본인 계정을 알려주지 않아도 실시간 공유가 가능하다.

(6) pc나 스마트폰이 아니더라도 USB잭을 탑재한 IT 기기라면 클라우드 서비스를 사용 할 수 있다.

(7) 현재 캡스톤팀이 개발하는 구름 USB아이템을 이용하여 일반인도 쉽게 사용가능한 원격 서비스의 제품으로 발전할 수 있다.

2) 활용 방안

(1) 1회에 한해 로그인을 하면 다음에 사용 할 시 아이디와 패스워드를 입력하는

번거로움이 없어진다.

(2) USB용량이 부족한 사용자의 경우 CLOUD의 용량을 활용하여 대용량 스토리지

서비스를 통한 용량 부족 현상을 해결한다.

(3) USB 메모리 단자에 물리적 손상으로 인해 파일을 읽을 수 없는 경우 대안이 가능하다.

기존 USB의 경우 백업을 따로 하지 않은 이상 내용물을 파악하기 어렵다. 본 캡스톤 팀이

제안하는 제품의 경우 물리적인 손상으로 인해 USB를 사용하지 못할 때 새로운 USB를

구매한 뒤 인증절차를 거치면 구글 드라이브에 연동되기 때문에 기존 USB와 동일한

내용의 USB로 활용 가능하고 따로 백업을 할 필요성이 없다.

(4) USB를 분실할 경우 타인에 의한 정보유출을 막을 수 있다.

기존의 USB는 주운 사람이면 누구나 내용을 확인 할 수 있다. 그로 인해 개인정보가 담긴 문서나 기밀문서의 경우 정보유출이 될 가능성이 존재한다. 구름 USB의 경우 USB를 삽입하면 APP을 통해 원격제어가 가능함으로 외부인의 USB접근을 사전에 막을 수 있다.

(5) 실시간 공유서비스를 활용하기 위해서는 본인의 계정이 요구되는 때가 있다.

1회 인증을 거친 USB를 가지고 있다면 계정을 알려주지 않아도 다양한 IT기기에서

다중연결이 가능하며 개인 정보를 알리지 않아도 된다는 보안점을 가질 수 있다.

(6) 일반적으로 클라우드 서비스를 활용하기 위해서는 PC나 스마트폰이 요구된다.

PC, 스마트폰이 아니더라도 USB잭을 탑재한 IT기기라면 CLOUD서비스를 이용 할 수

있다. 무선인터넷 서비스가 제공되지 않는 곳에서도 스마트폰 데이터를 이용해 사용 할 수

있다.

시나리오 1)

TV를 통하여 동영상을 실행해야 할 때, 굳이 컴퓨터와 TV를 연결하고, 컴퓨터를 이용하여 클라우드나 USB에서 파일을 실행할 필요 없이, 구름 USB를 바로 TV에 연결하여 동영상을 시청할 수 있다.

시나리오 2)

구름 USB하나만 가지고 있으면, 지인들이 클라우드를 통하여 공유한 파일들을 구름 USB에서 바로 확인 가능하다. 회사나 동아리 등의 그룹에서 공용 클라우드 계정을 가진 여러개 의 구름 USB를 배포하면, 그룹 내에서 쉽게 각종 자료들을 관리할 수 있다.

- 본 프로젝트는 단순히 STORAGE서비스를 제공하는 것에 그치지 인터넷 연결이 불가능한 IT기기에도 구름 USB를 통한 인터넷 연결을 가능하게 하여 네트워크의 접근할 수 있다는 점에서 봤을 때, 인터넷을 이용한 서비스 제품에 새로운 패러다임을 제시할 것으로 예상한다.

**3 배경 기술**

**3.1기술적 요구사항**

1. FAT32 파일 시스템

- 구글 드라이브의 파일 전체를 USB로 복사하는 것은 현실적으로 불가능하기 때문에

우선 파일들의 이름, 용량 등의 메타 데이터들만 먼저 받아 가상의 임시 파일을 생성하여

보여준다. 이 임시 파일들은 실제 USB에 저장되어 있는 것처럼 보여야 하기 때문에 FAT32

파일 시스템에 맞춰 생성해야 한다.

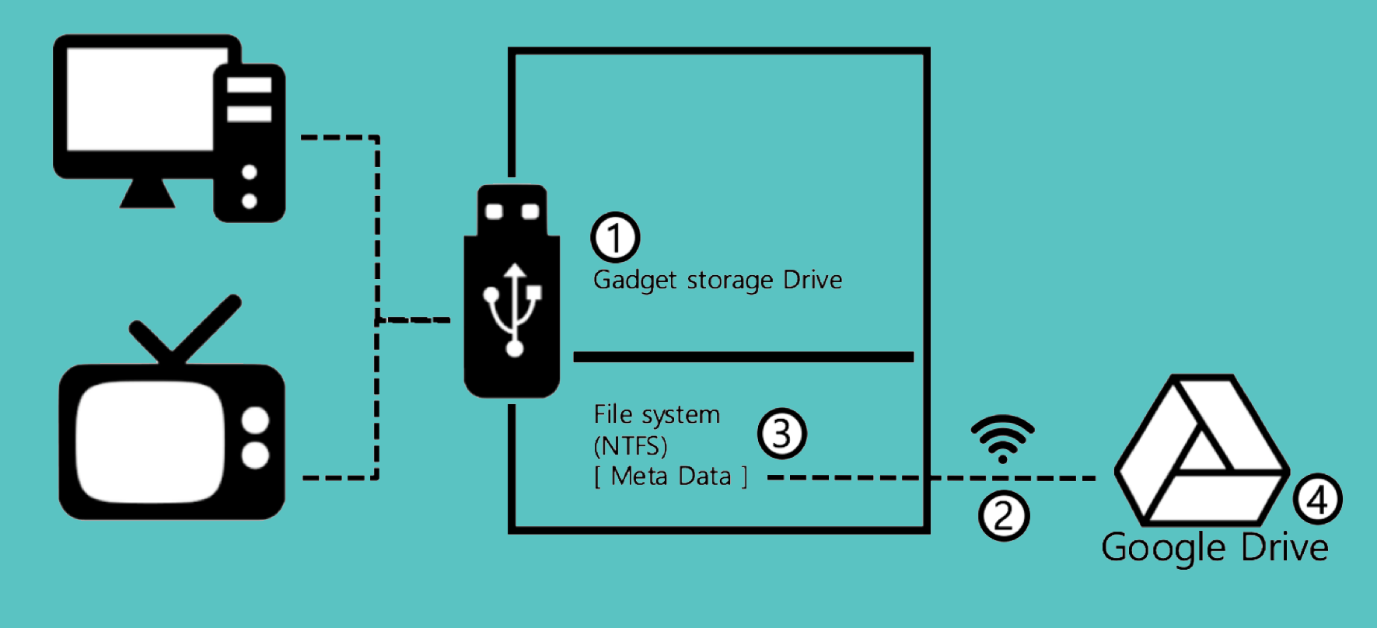
그 후에 Host PC에서 USB로 임의의 파일에 대해 접근을 시도하면 가젯 드라이버로 부터 신호를 받아 FAT32 파일 시스템에 맞게 분석하여 어떤 파일의 어떤 부분을 요청하는지 확인하고 해당하는 부분을 구글 드라이브에서 찾아 사용자에게 전달한다.

2. Pipe(FIFO)

- 구글 드라이브 API는 C언어로 제공하지 않기 때문에 FIFO라는 named pipe를 사용하여

Python(구글 드라이브)과 C 프로그램 사이의 통신을 지원한다.

**3.2 구성요소**



- 구름USB는 그 자체로 운영체제가 하나의 HOST로써 작동하지만 구름USB는 데스크탑, TV등에 연결될 때 일반적USB와 같은 device모드로 인식된다. Device모드 연결을 위하여 이번 프로젝트에서는 라즈베리파이 시리즈중 device모드 연결을 지원하는 라즈베리파이 zero를 사용하였다.

HOST(데스크탑)는 USB에 특정 정보를 블록 단위로 요청한다. 이때 특정 블록을 요청하는 로직은 USB가 어떤 파일시스템을 사용하고있는가에 따라 나뉜다. 구름USB는 FAT파일시스템을 사용한다. 단, 구름USB의 FAT파일시스템은 구글드라이브와 연동되기위해 변형되어 있다. 이 때문에 FAT파일시스템을 이해해야 한다.

구름USB내부의 device driver는 데스크탑/TV로부터 블록요청을 전달받는다. device driver가 커널 단의 프로그램이기 때문에 이 요청을 유저 프로그램으로 넘겨주려면 특별한 방법을 사용해야 한다. 커널과 유저프그램이 통신하려면 시스템콜을 이용해야 한다. 하나의 OS가 유지할 수 있는 시스템콜의 숫자는 한계가 있기 때문에 디바이스 드라이버와 유저프로그램의 통신은 IOCtl이라는 하나의 시스템콜을 공유해 이뤄진다. 이 유저프로그램은 커널에서 전달받은 블록 요청을 FAT파일시스템에 근거한 분석으로 데스크탑/TV가 원하는 파일을 알아낸다.

IOCtl: 커널 단의 device driver와 유저 프로그램간의 통신을 위해 필요한 시스템콜.

FAT 파일시스템: 구름USB는 데스크탑/TV등에 FAT파일시스템을 이용하는 USB로 보여진다. 구글 드라이브를 연동하기 위해 기본은 FAT파일시스템 관리프로그램을 변형하여 이용한다.

**3.3 현실적 제한 요소 및 그 해결 방안**

**3.3.1하드웨어**

1. 부팅속도

- 구름 USB는 Linux OS가 돌아가는 하나의 시스템이다. Linux OS가 돌아가기 위해서는 전원이 공급이 필요하다. 하지만 휴대성이 필요한 USB의 특성상 항상 전원을 공급할 수는 없고 사용시에만 가능하다. 그렇기 때문에 매번 사용하기 전에 OS를 부팅하는 과정이 필요하다. Linux의 fast boot mode, Linux 커널 축소 등의 해결책이 있다.

**3.3.2 소프트웨어**

1. 보안

- 분실 시

기존 USB의 문제점은 USB를 잃어 버렸을 경우, 안에 있는 파일이 분실 됨과 동시에 개인 정보 유출 문제가 있다. 구름 USB도 마찬가지로 이러한 문제점을 가지고 있다. 구름 USB에 실제 파일은 존재하지는 않지만, 구글 드라이브에 접속 할 수 있는 계정 정보가 들어있기 때문에 결국 같은 문제가 존재한다. 이를 해결하기 위해 모바일 앱을 이용한다. 모바일 앱은 구름 USB와 연동되어 해당 기기를 유효성을 선택할 수 있다. 만약 사용자가 구름 USB를 잃어버리면 모바일 앱에서 해당 기기의 인증을 해제시키면 된다. 그러면 그 구름 USB는 인증이 되지 않았기 때문에 구글 드라이브에 접속할 수 없다. 이걸 구현하기 위해서는 자체서버를 운영하여 구름 USB의 정보와 각 기기의 구글 드라이브 로그인 정보를 관리해야 한다. 또 이를 응용하면 하나의 유저가 여러 개의 구름 USB를 인증 하여 사용할 수도 있다.

2. 속도

- 스트리밍

구름 USB는 파일을 실행 요청 시 구글 드라이브에서 파일을 다운 받기 때문에 만약 파일의 크기가 크다면 파일이 실행이 지연 될 수 있다. 그런데 대부분의 사용자들이 사용하는 파일 중 크기가 큰 파일은 동영상, 음악 파일인 경우가 많다. 이 두 파일의 특징은 랜덤 액세스의 빈도가 매우 적고 대부분 시퀀셜하게 액세스 된다. 이러한 특징을 이용한 것이 스트리밍 서비스이다. 파일을 다 다운 받지 않고 다운로드와 실행을 동시에 할 수 있다. 구름 USB도 서비스 특성상 동영상, 음악 파일의 사용이 많을 것이므로 스트리밍 서비스가 가능하도록 한다. 이 부분은 구글 드라이브 API가 파일을 분할 다운로드를 지원하는지에 따라 바뀔 수 있다.

- 캐싱

동영상, 음악 파일이 아니더라도 사용자는 다른 종류의 크기가 큰 파일 또는 비슷한 종류의 여러개 파일을 사용할 경우가 많다. 캐싱 기법은 이러한 특징을 이용하여 여러 시스템에서 사용되는 방식이다. 예를 들어, 컴퓨터에서 디스크에 존재하는 데이터를 실행하기 위해서는 이 데이터를 메모리에 복사해야 한다. 특정 데이터에 접근한다면 가까운 미래에 그 데이터가 한번 더 요청되거나 그 데이터의 주변 데이터가 요청 될 확률이 높다. 그래서 요청된 데이터를 복사할 때 그 주변의 데이터도 같이 가져오며, 그 데이터들을 캐시에 저장한다. 그러면 가까운 미래에서 그 데이터들이 요청될 경우 디스크가 아닌 속도가 빠른 캐시에서 데이터를 가져올 수 있다. 이 특징을 이용하면 시스템의 파일 접근 속도가 매우 빨라질 수 있다. 구름 USB에서도 이 특징을 그대로 이용한다. 특히, 구름 USB는 파일을 네트워크로부터 가져오기 때문에 그 속도가 매우 느릴 것이다. 파일 다운로드 속도가 유저의 파일 요청 속도를 따라가지 못할 확률이 매우 높기 때문에, 캐싱을 이용하여 구름 USB의 성능을 개선하여야 한다.

- 파일 동기화 작업

구름 USB가 호스트 장치에 연결되면 구름 USB는 항상 구글 드라이브에 있는 최신 파일로 동기화 될 것임을 보장한다. 하지만 파일이 항상 동기화 된다는 것이 항상 새 파일을 받아온다는 뜻은 아니다. 현재 구름 USB 로컬 영역(또는 캐시)에 있는 실제 파일과 구글 드라이브에 존재하는 파일의 정확히 일치한다면 파일을 받을 필요가 없다. 그러므로 두 파일의 날짜, 용량, 해시 값 등을 확인하여 일치한다면 파일을 새롭게 다운받지 않고 바로 구름 USB가 해당 파일을 실행시킬 수 있다. 물론 구름 USB의 용량은 제한되어 있기 때문에 어떤 파일들을 로컬 영역에 저장해 놓고 있을 것인지는 여러 접근 방식을 적용하여 결정해야 한다.

**3.3.3 추가기능**

**1. 네트워크 속도**

- 네트워크의 속도는 메모리의 속도보다 느리기 때문에 클라우드 스토리지에서 제공하는

기능을 그대로 사용하면 속도가 일반 USB를 사용하는 것에 비해 매우 느리다. 그렇기

때문에 음성, 영상 파일에 대해서는 스트리밍과 같은 기술이 필요하고 그 외의 파일에

대해서는 캐싱과 같은 기술들을 채용하여 사용자가 그저 매우 느린 USB로 인식하지 않도록

해야한다.

**2. 여러 클라우드 스토리지 통합**

- 현재는 구글 드라이브만 이용 가능하지만 이 외의 클라우드 스토리지, 예를 들어

네이버의 N드라이브, Microsoft의 Dropbox 등을 통합 지원하여 기본적으로

제공하는 무료 용량을 확장하고 접근성을 높일 수 있도록 한다.

**3. Mobile App 개발**

- Mobile App을 이용하여 AP에 연결할 수 있도록 하고 클라우드 스토리지에 처음

연결하거나 계정을 추가, 삭제시에 활용하여 분실시에도 정보가 유출되지 않도록 한다.

**4. 성능 개선**

- 처음 설계 시부터 성능을 고려하지 않고 원하는 기능의 동작을 목표로 했기 때문에

실제로는 사용할 수 없는 느린 결과물이 나올 것으로 예상된다. 실제로 사용될 만큼의

성능을 갖추기 위해서 구현된 코드를 개선하고 Cache, Pre-process 등의 기술을 추가해야

한다.

**4**. **프로젝트 팀 구성 및 역할 분담**

|  |  |
| --- | --- |
| **이름** | **역할** |
| 채한울 | - FAT32 file system분석  - Google 드라이브 동기화 |
| 이창현 | - FAT32 file system 분석  - 디바이스 드라이버 연동 |
| 송재영 | - Raspberry Pi 부팅 시간 단축  - ppt 제작 |
| 진희상 | - Raspberry Pi slave 모드 활성화  - 디스크 블록 요청 분석 |
| 이정화 | - Google API 연결  - 문서수정 |

**5. 프로젝트 비용**

|  |  |
| --- | --- |
| **항목** | **예상치 (MD)** |
| 아이디어 구상 | 10 |
| 관련 자료 조사 | 15 |
| 연구 및 테스트 코드 작성 | 15 |
| 어플리케이션 UI구성 | 20 |
| 개발 환경 구축 | 15 |
| 내부 모듈 개발 | 45 |
| 내부 모듈과 UI연동 | 20 |
| 프로젝트 테스트 및 유지보수 | 20 |
| 프로젝트 평가 및 보고서 작성 | 15 |
| 합 | 175 |

**6 개발일정 및 자원관리**

**6.1개발 일정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **항목** | **세부내용** | **1월** | **2월** | **3월** | **4월** | **5월** | **비고** |
| 요구사항분석 | 아이디어 구상 |  |  |  |  |  |  |
| 정보 수집 |  |  |  |  |  |  |
| 관련분야연구 | 구글 드라이브 API 연구 |  |  |  |  |  |  |
| FAT파일시스템 분석 및 문서화 |  |  |  |  |  |  |
| 가젯 드라이버 분석 및 연구 |  |  |  |  |  |  |
| 설계 | 시스템 설계 |  |  |  |  |  |  |
| 구현 | UI 구현 |  |  |  |  |  |  |
| 내부 모듈 구현 |  |  |  |  |  |  |
| UI와 내부 모듈 연동 |  |  |  |  |  |  |
| 테스트 | 시스템 테스트 |  |  |  |  |  |  |
| 최종 발표 | 발표 준비 및 발표 |  |  |  |  |  |  |

**6.2일정별 주요 산출물**

|  |  |  |  |
| --- | --- | --- | --- |
| **마일스톤** | **개요** | **시작일** | **종료일** |
| 계획서 발표 | 프로젝트 아이디어 선정  프로젝트 자료 수집  개발 환경 구축  프로그램 설계  **산출물 :**   1. 프로젝트 수행 계획서 2. 프로젝트 기능 일람표 | 2016-01-04 | 2017-03-17 |
| 설계 완료 | 구글 드라이브와 FAT 32 연동  **산출물 :**   1. 프로젝트 1차 중간 보고서 2. 1차분 구현 소스 코드 | 2017-02-01 | 2017-03-17 |
| 1차 중간 보고 | 개발환경 및 전체 모듈 연동  **산출물 :**   1. 프로젝트 1차 중간 보고서 2. 1차분 구현 소스 코드 | 2017-03-18 | 2017-04-13 |
| 2차 중간 보고 | 사용자 인터페이스  **산출물 :**   1. 프로젝트 2차 중간 보고서 2. 2차분 구현 소스 코드 | 2017-03-20 | 2017-04-20 |
| 구현 완료 | 내부 모듈 및 UI 연동  **산출물: 구름USB** | 2017-04-21 | 2017-05-14 |
| 테스트 | 시스템 통합 테스트  **산출물: 구름 USB 최종본** | 2017-05-15 | 2017-05-20 |
| 최종 보고서 | 최종 보고  **산출물: 최종보고서** | 2017-05-21 | 2017-05-24 |

**6.3 인력자원 투입계획**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **이름** | **개발항목** | **시작일** | **종료일** | **총개발일(MD)** |
| 채한울 | FAT32 file system | 2017-01-04 | 2017-05-24 |  |
| 이정화 | Google Drive API | 2017-01-04 | 2017-05-24 |  |
| 이창현 | FAT32 file system | 2017-01-04 | 2017-05-24 |  |
| 진희상 | Gadget Driver | 2017-01-04 | 2017-05-24 |  |
| 송재영 | Linux Fastboot | 2017-02-06 | 2017-05-24 |  |

**6.4 비 인적자원 투입계획**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **항목** | **Provider** | **시작일** | **종료일** | **Required Options** |
| Raspberry Pi Zero | Raspberry | 2017-01-16 | 2017-05-24 | 1+1 |
| Raspberry Pi 3 | Raspberry | 2017-01-16 | 2017-05-24 |  |
| 개발용 노트북 5대 | Lenovo, Samsung,  Mac | 2017-01-04 | 2017-05-20 |  |

**7** **참고문헌 및 참고 사이트**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **번호** | **종류** | **제목** | **출처** | **발행년도** | **저자** | **기타** |
| 1 | 사이트 | 구글 API python | https://developers.google.com/api-client-library/python/ |  |  |  |
| 2 | 사이트 | Using FIFOs in Python | http://timmurphy.org/2013/11/11/using-fifos-in-python/ |  |  |  |
| 3 | 사이트 | Named Pipe in Linux with a Python Example | http://www.roman10.net/2011/04/21/named-pipe-in-linux-with-a-python-example/ |  |  |  |
| 4 | 사이트 | Embedded python | http://www.clien.net/cs2/bbs/board.php?bo\_table=lecture&wr\_id=212575 |  |  |  |
| 5 | 사이트 | FAT file system C code | http://ultra-embedded.com/fat\_filelib/ |  |  |  |
| 6 | 사이트 | Design of the FAT file system | https://en.wikipedia.org/wiki/Design\_of\_the\_FAT\_file\_system |  |  |  |
| 7 | 사이트 | Fuse(Filesystem in Userspace) | https://github.com/libfuse/libfuse |  |  |  |
| 8 | 책 | 리눅스 커널 모듈 프로그래밍 | 리눅스 커널 심층 분석 개정3판 501p 17장 장치와 모듈 | 2013.01.20 | 로버트 러브 |  |
| 9 | 책 | 파일 시스템 마운팅 개념 | Operating System Concepts 8th Edition | 2013.02.15 | Abraham Silberschatz , Peter B. Galvin, Greg Gagne |  |
| 10 | 사이트 | g\_mass\_storage | <http://www.linux-usb.org/gadget/file_storage.html> |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |