**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Git watcher* |
| 팀 명 | *GET* |
| 문서 제목 | 계획서 |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 04 |

|  |  |
| --- | --- |
| **팀원** | 전호현(조장) |
| 이근하 |
| 유문상 |
| 최원대 |
|  |

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

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 계획서-프로젝트명.doc |
| **원안작성자** | 전호현, 이근하, 유문상, 최원대 |
| **수정작업자** | 전호현, 이근하, 유문상, 최원대 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2018-03-05 | 전호현 | 1.0 | 최초 작성 | 목차 별 초안 작성 |
| 2009-03-15 | 홍길동 | 1.1 | 내용 수정 | 일정 및 역할분담 재조정 |
| 2009-03-20 | 홍길동 | 1.2 | 내용 수정 | 현실적 제한요소 추가 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[**1** **개요** 4](#_Toc347412182)

[1.1 프로젝트 개요 4](#_Toc347412183)

[1.2 추진 배경 및 필요성 4](#_Toc347412184)

[**2** **개발 목표 및 내용** 5](#_Toc347412185)

[2.1 목표 5](#_Toc347412186)

[2.2 연구/개발 내용 6](#_Toc347412187)

[2.3 개발 결과 7](#_Toc347412188)

[2.3.1 결과물 목록 및 상세 사양 7](#_Toc347412189)

[2.3.2 시스템 기능 및 구조 7](#_Toc347412190)

[2.4 기대효과 및 활용방안 7](#_Toc347412191)

[**3** **배경 기술** 8](#_Toc347412192)

[3.1 기술적 요구사항 8](#_Toc347412193)

[3.2 현실적 제한 요소 및 그 해결 방안 9](#_Toc347412194)

[3.2.1 하드웨어 9](#_Toc347412195)

[3.2.2 소프트웨어 9](#_Toc347412196)

[3.2.3 기타 9](#_Toc347412197)

[**4** **프로젝트 팀 구성 및 역할 분담** 10](#_Toc347412198)

[**5** **프로젝트 비용** 10](#_Toc347412199)

[**6** **개발 일정 및 자원 관리** 11](#_Toc347412200)

[6.1 개발 일정 11](#_Toc347412201)

[6.2 일정별 주요 산출물 12](#_Toc347412202)

[6.3 인력자원 투입계획 13](#_Toc347412203)

[6.4 비 인적자원 투입계획 14](#_Toc347412204)

[**7** **참고 문헌** 15](#_Toc347412205)

# **개요**

## 프로젝트 개요

규모가 있는 SW 개발을 함에 있어 동일한 프로젝트에 대해 여러 개발자가 동시에 개발할 수 있도록 돕는 버전 관리 시스템을 사용하는 것은 필수적이다. 따라서, 개발자라면 버전 관리 시스템을 효율적으로 사용할 수 있어야 한다.

그렇다면, 버전 관리 시스템을 사용하여 개발한 프로젝트에서 누가, 얼마나 개발에 기여했는지를 간략하게 보여줄 수는 없을까? 우리 팀은 버전 관리 시스템 중에서도 가장 많이 쓰이는 GitHub를 활용한 프로젝트의 기여도 평가를 돕는 웹 서비스를 만들고자 한다. 프로젝트 참여자 각각의 커밋 횟수, 삽입 라인, 삭제 라인, 코드의 안정성 등에 대한 정보를 제공하여, 각자의 기여도를 평가해 볼 수 있다.

## 추진 배경 및 필요성

### 추진 배경

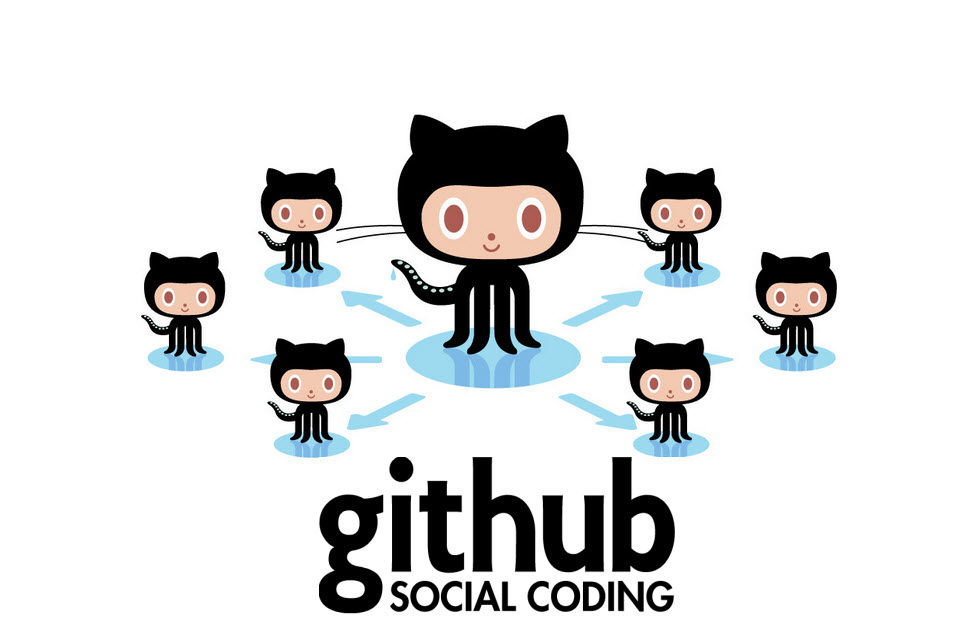
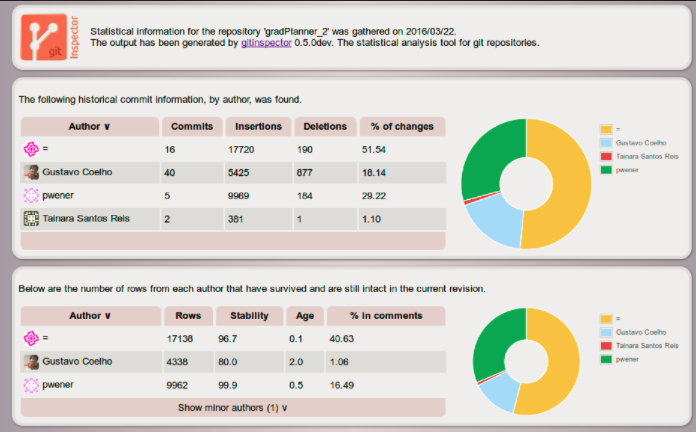


그림 1. Github

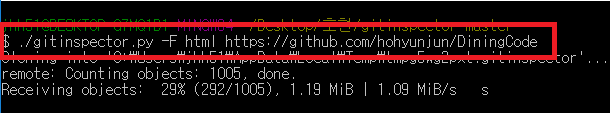
Github는 가장 보편적으로 사용되는 버전 관리 시스템 중 하나이다. 수많은 오픈소스 프로젝트들이 github의 저장소에 존재한다. 그리고 각각의 오픈소스 프로젝트들은 다수의 Contributor(기여자)들로부터 만들어진다. 우리 팀은 오픈소스 프로젝트 기여자들이 실제 코드에 얼마만큼 기여했는지를 평가할 수 있을만한 서비스를 만들고자 했다. 그러던 중, 각 기여자들의 깃허브 활용 정도를 보여주는 오픈소스 프로젝트, Git Inspector를 찾게 되었고, 이 시스템의 문제점을 보완하고 필요한 기능을 추가하여 평가에 적합한 서비스를 제공하고자 한다.

### Git Inspector의 장점

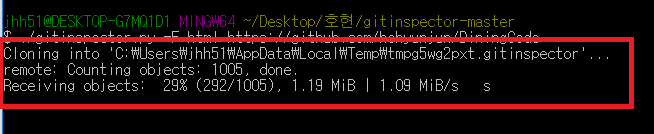


위 그림에서 볼 수 있듯이, Git inspector는 Github repository에 있는 오픈소스 프로젝트 기여자들의 Github 활동에 대한 다양한 정보를 제공한다. 각 기여자들의 Commit 수, 코드 삽입 라인 수, 코드 삭제 라인 수, 전체 코드 중에서 몇 퍼센트의 변화를 주었는지 등의 정보를 제공하는 것을 위 그림에서 볼 수 있다. 또한, 이러한 정보를 바탕으로 사용자에게 Pie 그래프를 제공하여 사용자는 누가 얼마나 많이 기여했는지를 한 눈에 파악할 수 있다.

### Git Inspector의 문제점



위 그림은 실제 Git inspector를 실행시키는 과정이다. 특정 repository에 대한 분석을 하려면 위와 같은 코드를 분석을 진행하고 싶을 때마다 터미널에 쳐야 한다. 사용하기가 매우 불편하다.



뿐만 아니라, Git inspector를 실행하면 실행할 때마다 사용자가 원하는 Github repository 로부터 Cloning을 받아와야 한다. 이 Cloning을 하는 시간이 오래 걸리며, 프로젝트가 크면 클수록 더 오래 걸린다.

### Git Watcher의 필요성

우리 팀은 위에서 본 것과 같은 Git inspector의 장점은 살리고, 단점을 보완하여 사용자 친화적인 웹 서비스를 만들고자 한다. Git inspector의 사용성을 개선하여, 사용자가 사용하기 쉬운 User Interface를 제공한다. 또한, 매 실행 시마다 Cloning이 이루어져서 생기는 성능적인 문제를 해결하기 위해, 데이터베이스에 Github 저장소의 코드를 저장하고 유지할 것이다. 이를 통해 사용자는 보다 쉽고 빠르게 Github 활용도에 대한 정보를 얻을 수 있다.

# **개발 목표 및 내용**

## 목표

본 프로젝트의 목표는 기존의 Git Inspector를 보완하여, GitHub를 활용해 프로젝트를 진행한 참여자들 각각의 기여도를 평가할 수 있도록 돕는다. 명령어 Command를 사용해야 하는 Git Inspector를 UI를 제공하여 편리하게 사용할 수 있으며, 데이터베이스에 정보를 미리 저장하여 더 빠른 분석결과를 제공한다. 더 나아가 Git Inspector에서는 제공하지 않는 기능들도 추가하여 제공한다.

이를 통해 기업에서 각 개개인의 기여도를 평가하거나, 대학에서 프로젝트 과목의 학점을 평가하는 데 도움이 될 수 있을 것이다. 개인이 혼자 진행한 Github 프로젝트에 대해서도, 프로젝트에 대한 커밋 횟수, 변화 수 등의 상세 정보를 제공함으로써 도움을 줄 수 있다.

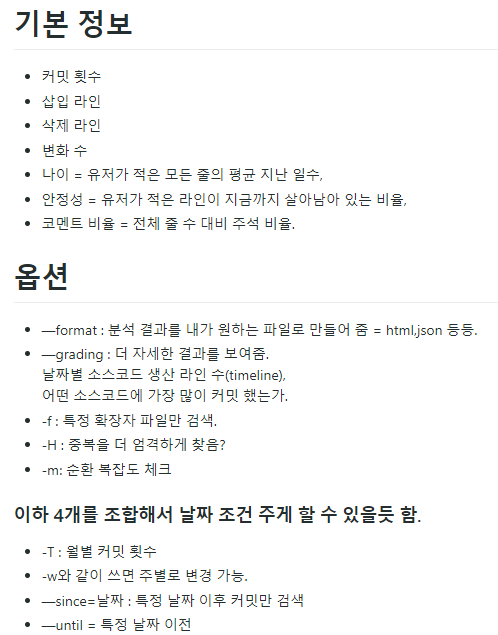
## 연구/개발 내용

### GitHub 프로젝트의 변동사항을 알려줄 API 탐색

지정된 Github 레포지토리의 활동을 계속 주시하여 새로운 push가 일어날 경우에, 데이터베이스에 있는 레포지토리 코드를 업데이트 할 수 있어야 한다. 이를 위해서는 Github 프로젝트의 변동사항을 알려줄 API가 필요하다. Github API 중에서, webhook 이라는 API를 활용하여, 특정 저장소에서 push가 일어나는지를 감시하도록 한다.

### Git Inspector

Git Inspector를 활용하여 코드를 분석할 것이므로, Git Inspector가 어떤 기능을 제공하는지를 완벽하게 파악한다. Git Inspector 저장소에 있는 Documentation을 활용하여 그 기능을 직접 수행하면서 기능을 파악한다.



### 데이터베이스에 코드 저장, 유지

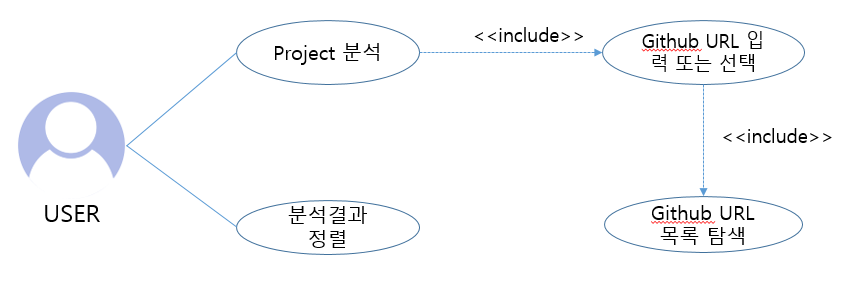
실행 시마다 Github repository의 Cloning이 일어나는 Git inspector의 문제점을 해결하기 위해, 특정 Github repository의 코드를 미리 데이터베이스에 저장하고 유지할 수 있도록 한다. 위에서 찾은 API를 AWS Lambda에 올려놓고, API로부터 신호가 올 때마다 AWS S3 데이터베이스에 있는 코드를 업데이트한다.

### Client Side 구축

Git Watcher는 사용자에게 사용하기 편리한 UI를 제공할 수 있어야 한다. 또한, 사용자의 요청에 따라 새롭게 분석을 진행하여 사용자에게 보여줄 수 있어야 한다.

## 개발 결과

### 시스템 기능 요구사항



1. 사용자가 분석하고자 하는 Github project URL을 입력 또는 선택한다.
2. Lambda에 올려져 있는 Git inspector가 Github URL에 해당하는 코드를 S3로부터 가지고 와서 분석한다.
3. 분석 결과를 사용자에게 제공한다. 사용자는 분석 결과를 분석 기준에 따라 정렬할 수 있다.

### 시스템 비기능(품질) 요구사항

1. 신뢰성과 이용가능성

특정 Github Project에 기여한 사람들의 정보를 가지고 그 기여도를 평가하는 것이 목적인 만큼, 정보를 정확하게 분석하는 것이 중요하다. 분석 결과에 오차가 없어야 한다. 또한, 사용자가 원할 때 언제든지 사용 가능하여야만 한다.

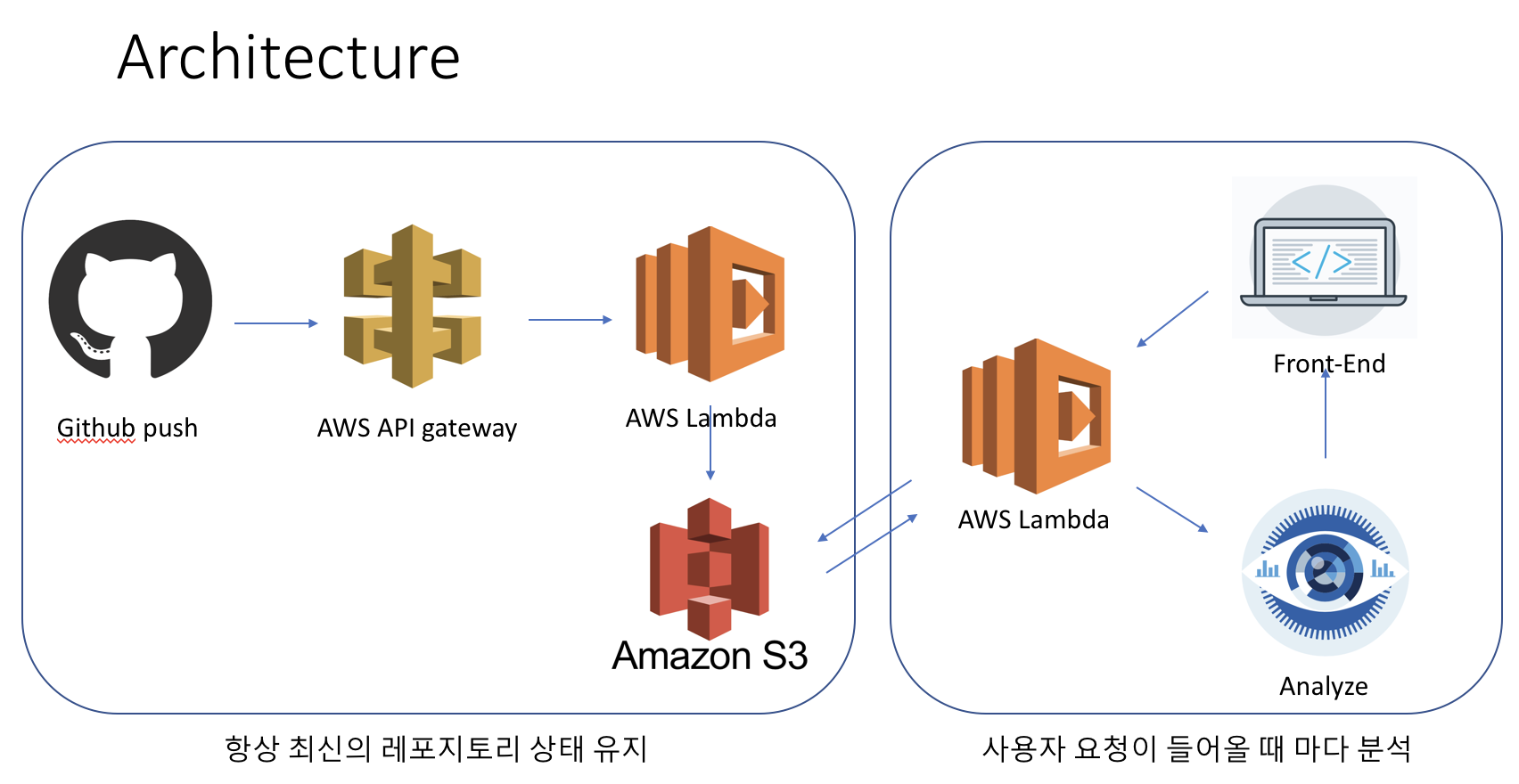
1. 성능/처리량

기존의 Git inspector 어플리케이션이 분석을 할 때마다 저장소를 clone 해와서 분석을 하였기 때문에, 분석 속도가 매우 느렸었다. 따라서 이 시스템의 성능은 최소한 기존의 Git inspector 보다 빨라야 한다.

1. 사용성

인터페이스 구성에 있어 사용자가 손쉽게 메뉴를 찾을 수 있도록 구성하여야 한다.

### 시스템 구조



### 결과물 목록 및 상세 사양

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 형식 | 비고 |
| 내부 모듈 | AWS Lambda1 | AWS API gateway로부터 Github project 에 push가 일어났다는 것을 인식받아 Amazon S3에 해당 github project 코드를 update한다. |  |  |
|  | AmazonS3 | Github project 들의 코드를 저장한다. |  |  |
|  | AWS Lambda2 | 사용자로부터 github project url 요청을 받아서 해당 url의 코드를 Amazon S3에서 찾는다. 찾은 코드를 git inspector를 활용하여 분석하고, 그 결과값을 사용자에게 제공한다. |  |  |
| UI | 레이아웃 | 사용자가 쉽게 사용할 수 있도록 User Interface를 제공한다. |  |  |
|  | 연동 | AWS Lambda2에서 전송해주는 결과값을 받아서 올바르게 사용자에게 제공한다. |  |  |

## 기대효과 및 활용방안

### 기대효과

GitHub는 대규모의 소프트웨어 프로젝트를 진행하는 산업체부터, 대학교의 팀 프로젝트, 개인 프로젝트 사용자까지 보편적으로 사용하는 버전 관리 시스템이다. 이렇게 GitHub 저장소에 올려진 프로젝트들을 분석하여 커밋 수, 살아남은 코드 줄 수, 작성한 코드가 차지하는 비율 등을 제공한다면, 프로젝트 기여자들 간의 기여도를 평가할 수 있다. 개인 프로젝트를 진행한 경우에도, 자신이 어떤 기간에 얼마나 프로젝트를 진행했으며, 얼마나 많은 코드들이 삭제되었고 추가되었는지 등의 정보를 얻을 수 있어 프로젝트를 진행하는 데 충분히 도움이 될 수 있다.

### 활용방안

* 개인 프로젝트를 진행하는 경우, 자신이 기간별로 얼마만큼의 작업을 했으며, 코드의 삽입과 삭제가 얼마나 많이 이루어졌는지 등을 확인하는 데 활용될 수 있다.
* 대규모 소프트웨어 프로젝트를 진행하는 대학 수업에서, 프로젝트에 기여한 학생들의 기여도를 평가할 수 있다. 누가 작성한 코드가 얼마나 코드에 실제로 반영되었는지, 기간 별로 얼마만큼의 작업이 이루어졌는지 등을 알 수 있다.

# **배경 기술**

## 기술적 요구사항

**프로젝트의 결과물의 기술적인 요구 사항을 모두 나열한다.**

**프로젝트를 개발하는 데 필요한 개발 환경과, 프로젝트 결과물을 확인할 수 있는 환경을 나누어 기술한다.**

**개발 환경은 개발에 필요한 운영체제 환경, 컴파일 환경, 개발 언어, 언어의 문법적 요구사항을 기술한다.**

**프로젝트 결과물 확인 환경은 동작시킬 수 있는 운영체제 환경, 미리 설치되어 있어야 하는 소프트웨어 및 라이브러리를 기술한다. 서버 환경의 경우 서버의 구성 방법에 대해서 기술해야 한다.**

## 현실적 제한 요소 및 그 해결 방안

**프로젝트를 수행하기 이전에 시스템 개발시 발생할 가능성이 있는 제한 요소를 미리 예측하여 나열한다. 또한 그 제한 요소를 피해갈 수 있는 해결 방안에 대해서도 나열한다. 예를 들어, GNU 라이선스가 있는 소프트웨어 라이브러리를 사용하는 경우에 이를 사용하는 소프트웨어의 소스를 공개하여야 한다. 만약 개발할 시스템이 상용화 제품일 경우에는 문제가 발생할 수 있다. 이를 어떻게 해결할 것인가? 하는 점 등이다. 또한 시스템의 성능(속도, 처리할 수 있는 데이터의 양 등등)이 어느 정도 이상이 되어야 한다든지 혹은 안정성을 어느 정도 확보를 하여야 하는 점도 현실적 제한 요소가 될 수 있다. 이를 하드웨어 측면 혹은 소프트웨어적인 측면에 대하여 기술한다.**

**이러한 현실적 제한요소를 팀원들과 토의한 내용과 지도 교수님과 토의한 내용은 반드시 회의록에 남기도록 한다.**

### 하드웨어

### 소프트웨어

### 기타

# **프로젝트 팀 구성 및 역할 분담**

**프로젝트에 참여하는 멤버의 역할을 구체적으로 명시한다.**

| 이름 | 역할 |
| --- | --- |
| 전호현 |  |
| 이근하 |  |
| 유문상 |  |
| 최원대 |  |

# **프로젝트 비용**

**프로젝트에 투입될 비용을 예상하여 기입한다. 여기서 Man-Days 라 함은 인적 비용을 계산하기 위한 단위로 한 사람이 하루 8시간 정도 일을 해야 하는 양을 1 MD라고 한다. 즉, 한 사람이 하루 4시간씩 일을 하게 되면 이틀 정도 일을 했을 때 1 MD 정도의 비용이 들어갔다고 한다.**

|  |  |
| --- | --- |
| **항목** | **예상치 (MD)** |
| 아이디어 구상 |  |
| 관련 정보 수집 |  |
| 연구 및 테스트 코드 작성 |  |
| 개발 환경 구축 |  |
| 내부 모듈 개발 |  |
| 웹 서비스 UI/UX 구성 |  |
| 내부 모듈과 웹 서비스 연동 |  |
| 프로젝트 테스트 및 유지보수 |  |
| 프로젝트 관련 문서작업 |  |
| 합 |  |

# **개발 일정 및 자원 관리**

## 개발 일정

**개발 일정을 계획한다.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **항목** | **세부내용** | **1월** | **2월** | **3월** | **4월** | **5월** | **6월** | **비고** |
| 요구사항분석 | 요구 분석 |  |  |  |  |  |  |  |
| SRS 작성 |  |  |  |  |  |  |  |
| 관련분야연구 | 주요 기술 연구 |  |  |  |  |  |  |  |
| 관련 시스템 분석 |  |  |  |  |  |  |  |
| 설계 | 시스템 설계 |  |  |  |  |  |  |  |
| 구현 | 코딩 및 모듈 테스트 |  |  |  |  |  |  |  |
| 테스트 | 시스템 테스트 |  |  |  |  |  |  |  |

## 일정별 주요 산출물

**일정별로 어떤 결과물을 도출할 지 상세하게 작성한다. 그래프의 형태로 작성하여도 좋다.**

|  |  |  |  |
| --- | --- | --- | --- |
| 마일스톤 | 개요 | 시작일 | 종료일 |
| 계획서 발표 | 프로젝트 아이디어 선정  프로젝트 아키텍처 구상  관련 정보 수집 및 연구  **산출물 :**   1. 프로젝트 수행 계획서 2. 프로젝트 소개 PowerPoint | 2018-01-10 | 2018-03-09 |
| 1차 중간 보고 | 프로젝트 아키텍처 구현 완료  프로젝트 프로토타입 구현 완료  **산출물 :**   1. 프로젝트 1차 중간 보고서 2. 프로젝트 진도 점검표 3. 1차분 구현 소스 코드 | 2018-03-09 | 2018-04-12 |
| 2차 중간 보고 | 웹 페이지 세부 구현 완료  기능 간 연동 완료  **산출물 :**   1. 프로젝트 2차 중간 보고서 | 2018-04-12 | 2018-05-01 |
| 구현 완료 | 시스템 구현 완료  **산출물:** | 2018-05-01 | 2018-05-11 |
| 테스트 | 시스템 통합 테스트  **산출물:** | 2018-05-11 | 2018-05-18 |
| 최종 보고서 | 최종 보고  **산출물:** | 2018-05-18 | 2018-05-28 |

## 인력자원 투입계획

**프로젝트 참여 인력이 언제부터 언제까지 어떤 일로 투입이 될 지 구체적으로 명시한다.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 이름 | 개발항목 | 시작일 | 종료일 | 총개발일(MD) |
| 홍길동 | *디바이스 드라이버* | 2009-03-01 | 2009-04-20 | 20 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 비 인적자원 투입계획

**개발 환경 등 비 인적 자원의 투입 계획을 명시한다.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 항목 | Provider | 시작일 | 종료일 | Required Options |
| 컴파일러 | Microsoft | 2009-03-01 | 2009-04-20 |  |
| 개발용 PC 4대 | Dell |  |  |  |
| 임베디드 보드 | 미정 |  |  | PXA270 |
|  |  |  |  |  |

# **참고 문헌**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 사이트 | Github Webhook | https://developer.github.com/webhooks/ |  |  |  |
| 2 | 사이트 | Git Inspector | https://github.com/ejwa/gitinspector |  |  |  |
| 3 | 사이트 | Amazon Lambda | https://aws.amazon.com/ko/lambda/features/ |  |  |  |
| 4 | 사이트 | Amazon S3 | https://aws.amazon.com/ko/s3/ |  |  |  |