



국민대학교
전자정보통신대학
컴퓨터공학부


캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	Git watcher
팀 명	GET(Github Evaluate & Test)
문서 제목	계획서

Version	1.5
Date	2018-MAR-09

팀원	전호현(조장)
	이근하
	유문상
	최원대

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인 I 수강 학생 중 프로젝트 "**Git Watcher**"를 수행하는 팀 "**GET**"의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 "**GET**"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	계획서-Git Watcher.doc
원안작성자	전호현, 이근하, 유문상, 최원대
수정작성자	전호현, 이근하, 유문상, 최원대

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2018-03-05	전호현	1.0	최초 작성	목차 별 초안 작성
2018-03-06	최원대	1.1	내용 추가	목차 3,5,6 추가
2018-03-06	전호현	1.2	내용 추가	목차 1,2,4,7 추가
2018-03-08	전호현	1.3	내용 수정	프로젝트 개요, 목표 수정, 아키텍처 수정 본 추가 및 내용수정
2018-03-08	최원대	1.4	내용 수정	목차 3의 내용 수정 및 추가
2018-03-09	전호현	1.5	내용 수정	목차 2의 내용 수정 및 추가

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

목 차

1	개요	4
1.1	프로젝트 개요	4
1.2	추진 배경 및 필요성	4
1.2.1	추진 배경	4
1.2.2	Git Inspector의 장점	5
1.2.3	Git Inspector의 문제점	5
1.2.4	Git Watcher의 필요성	6
2	개발 목표 및 내용	7
2.1	목표	7
2.2	연구/개발 내용	7
2.2.1	GitHub 프로젝트의 변동사항을 알려줄 API 탐색	7
2.2.2	Architecture 구축	7
2.2.3	Git Inspector	8
2.2.4	Amazon S3에 코드 저장, 유지	8
2.2.5	Client Side 구축	9
2.2.6	Git inspector의 기능 추가	9
2.3	개발 결과	9
2.3.1	시스템 기능 요구사항	9
2.3.2	시스템 비기능(품질) 요구사항	10
2.3.3	시스템 구조	11
2.3.4	결과물 목록 및 상세 사양	11
2.4	기대효과 및 활용방안	12
2.4.1	기대효과	12
2.4.2	활용방안	12
3	배경 기술	13
3.1	기술적 요구사항	13
3.2	현실적 제한 요소 및 그 해결 방안	14
3.2.1	하드웨어	14
3.2.2	소프트웨어	14
4	프로젝트 팀 구성 및 역할 분담	14
5	프로젝트 비용	15
6	개발 일정 및 자원 관리	15
6.1	개발 일정	15
6.2	일정별 주요 산출물	15
6.3	인력자원 투입계획	16

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

6.4	비 인적자원 투입계획	16
7	참고 문헌	17

1 개요

1.1 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, 기존에 버전 관리 시스템을 사용하여 만든 프로젝트를 분석하여, 그 정보를 제공함으로써 사용자가 버전 관리 시스템을 더 잘 사용할 수 있도록 만들 수는 없을까? 우리 팀은 버전 관리 시스템 중에서도 가장 보편적으로 쓰이는 GitHub를 활용한 프로젝트를 분석한 정보를 제공함으로써 사용자가 GitHub를 더 효율적으로 사용할 수 있도록 돕는 서비스를 만들고자 한다.

1.2 추진 배경 및 필요성

1.2.1 추진 배경



그림 1. Github

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

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 Github 저장소에 존재하는 프로젝트의 분석을 Contributor들에게 제공하여 그들이 실제 프로젝트 코드에 얼마만큼 기여했으며, Github를 얼마나 잘 사용했는지를 알 수 있도록 하는 서비스를 만들고자 했다. 그러던 중, 각 Contributor들의 깃허브 활용 정도를 보여주는 오픈소스 프로젝트, Git Inspector를 찾게 되었고, 이 시스템의 문제점을 보완하고 필요한 기능을 추가하여 평가에 적합한 서비스를 제공하고자 한다.

1.2.2 Git Inspector의 장점

```

+--[1mAuthor          Commits    Insertions    Deletions    % of changes+--[0;0m
DESKTOP-8DP0IN2#one1    10         3239         195         36.28
DESKTOP-G7MQ1D1#jhh5    2           122           3           1.32
HoHyun Jun             10        2788        1171        41.82
SANG YUN LEE           2           43           43           0.91
SangYunLEE             1          258          72           3.49
Unknown                2        1522          10          16.18

```

Below are the number of rows from each author that have survived and are still intact in the current revision:

```

+--[1mAuthor          Rows      Stability      Age      % in comments+--[0;0m
DESKTOP-8DP0IN2#one1  2895      89.4          0.5      5.08
HoHyun Jun           1615      57.9          0.0      4.33
SangYunLEE           992       384.5         0.0      0.91

```

위 그림은 Git inspector를 실행한 결과화면이다. 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.


1.2.3 Git Inspector의 문제점

```

C:\Users\H51\Desktop> cd Desktop\호현\gitinspector-master
$ ./gitinspector.py -F html https://github.com/hohyunjun/DiningCode
Opening into 'C:\Users\H51\AppData\Local\Temp\tmp55ng2xt\gitinspector'...
remote: Counting objects: 1005, done.
Receiving objects: 29% (292/1005), 1.19 MiB | 1.09 MiB/s   s

```

위 그림은 실제 Git inspector를 실행시키는 과정이다. 특정 repository에 대한 분석을 하려면 위와 같은 코드를 분석을 진행하고 싶을 때마다 터미널에 명령어에 입력해야 한다. 사용자가 사용하기에 불편하고, 코드 오타로 인해 오류가 생길 가능성도 크다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

```
jhh51@DESKTOP-G7M01D1 MINGWB4 ~/Desktop/호현/gitinspector-master
$ ./gitinspector.py -F http://github.com/hojunjun/DiningCode
Cloning into 'C:\Users\jhh51\AppData\Local\Temp\tmpg5wg2pxt.gitinspector'...
remote: Counting objects: 1005, done.
Receiving objects: 29% (292/1005), 1.19 MiB | 1.09 MiB/s  s
```


뿐만 아니라, Git inspector를 실행하면 실행할 때마다 사용자가 원하는 Github repository 로 부터 Cloning을 받아와야 한다. Repository를 Cloning을 하는 시간은 Repository에 있는 코드 정보 전체를 받아와야 하므로, 시간이 매우 오래걸린다. 따라서 사용자는 분석을 위해 오랫동안 기다려야 한다.

1.2.4 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. 쉽게 레포지토리에 대한 분석을 할 수 있도록, 기존에 저장된 레포지토리 목록을 보여주고, 어떤 옵션을 사용할 것인지를 간단히 버튼으로 만들 것이다.

또한, 매 실행 시마다 Cloning이 이루어져서 생기는 성능적인 문제를 해결하기 위해, Amazon S3에 Github 저장소의 코드를 저장하고 유지할 것이다. 이에 따라 사용자의 분석 요청이 들어올 때, 새로 코드를 Cloning 해올 필요성이 사라지고, 저장된 코드를 분석만 하면 되기 때문에 시간이 크게 단축될 것이다. 그러므로 사용자는 보다 쉽고 빠르게 Github 활용도에 대한 정보를 얻을 수 있다.

그리고 Git inspector에는 없는 분석 항목을 추가하여 사용자에게 프로젝트에 대해 더 자세한 정보를 제공하고자 한다. 예를 들면, 레포지토리에 올려진 Issue에 대한 정보나 branch에 대한 정보도 제공할 수 있다. 이러한 분석 항목을 통해 사용자는 GitHub에 있는 더 많은 기능들을 알게 되며, 이를 더 효율적으로 사용할 수 있을 것이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

2 개발 목표 및 내용

2.1 목표

본 프로젝트의 목표는 기존의 Git Inspector를 보완하여, GitHub를 활용해 프로젝트를 진행한 참여자가 GitHub를 더 효율적으로 사용할 수 있도록 분석 정보를 제공하는 것이다. 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 사용자는 보다 쉽게 자신이 GitHub를 얼마나 잘 활용하고 있는지에 대한 정보를 제공받을 수 있다. 프로젝트에 대한 커밋 횟수, 변화 수 등의 상세 분석 결과를 바탕으로 더 효율적으로 GitHub를 사용할 수 있다.

2.2 연구/개발 내용


2.2.1 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.

2.2.2 Architecture 구축

회의를 통해 구상한 Pipeline 아키텍처를 구축한다. 우리 팀이 만들고자 하는 프로젝트의 아키텍처는 크게 두 가지로 나뉜다. 첫번째는 항상 최신의 레포지토리 상태를 유지시킬 수 있는 아키텍처이다. 사전에 주어진 GitHub repository에서 push가 일어났을 경우부터 시작하여, 이것을 API gateway와 연결하고, AWS Lambda와 연동시켜 Amazon S3에 업데이트된 코드를 저장시킬 수 있어야 한다.

두 번째 아키텍처는 사용자의 요청이 들어올 때마다 분석을 수행하는 아키텍처이다. AWS Lambda에 Git inspector를 올리고, 웹페이지로부터 사용자 요청을 받고, 그 결과를 보내줄 수 있어야 한다. 또한, 최신의 레포지토리 상태를 유지하고 있는 첫번째 아키텍처의 Amazon S3와도 연결되어야 한다. 저장되어 있지 않은 repository를 사용자가 요청할 경우에 대비하여, Git inspector가 바로 Github repository에서 Cloning 하여 분석을 수행할 수 있도록 해야한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

2.2.3 Git Inspector

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

기본 정보

- 커밋 횟수
- 삽입 라인
- 삭제 라인
- 변화 수
- 나이 = 유저가 적은 모든 줄의 평균 지난 일수,
- 안정성 = 유저가 적은 라인이 지금까지 살아남아 있는 비율,
- 코멘트 비율 = 전체 줄 수 대비 주석 비율.

옵션


- `--format`: 분석 결과를 내가 원하는 파일로 만들어 줌 = `html,json` 등등.
- `--grading`: 더 자세한 결과를 보여줌.
날짜별 소스코드 생산 라인 수(`timeline`),
어떤 소스코드에 가장 많이 커밋 했는가.
- `-f`: 특정 확장자 파일만 검색.
- `-H`: 중복을 더 엄격하게 찾음?
- `-m`: 순환 복잡도 체크

이하 4개를 조합해서 날짜 조건 주게 할 수 있을듯 함.

- `-T`: 월별 커밋 횟수
- `-w`와 같이 쓰면 주별로 변경 가능.
- `--since=날짜`: 특정 날짜 이후 커밋만 검색
- `--until`: 특정 날짜 이전

2.2.4 Amazon S3에 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 GitHub repository의 코드로 업데이트한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

2.2.5 Client Side 구축

Git Watcher는 사용자에게 사용하기 편리한 User Interface를 제공할 수 있어야 한다. 사용자에게 Amazon S3에 저장된 저장소의 목록을 보여줄 수 있도록 User Interface를 구성한다. 또한, 분석 결과로 보여지는 기본 정보인 커밋 횟수, 삭제라인, 변화 수 등을 기준으로 Contributor들을 정렬하여 보여줄 수 있어야 한다. 분석 결과를 원하는 파일로 만들어 주는 format 이나 특정 확장자 파일만 검색하도록 도와주는 옵션에 대한 User Interface 또한 존재해야만 한다.

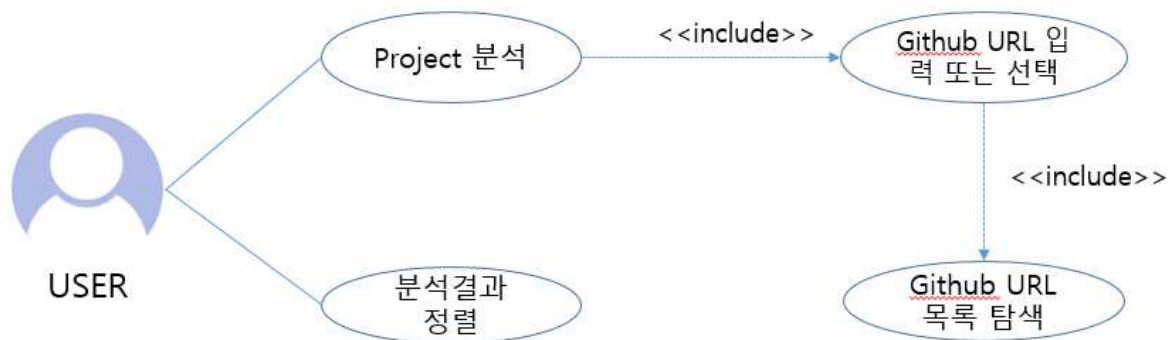
또한, 사용자의 요청에 따라 새롭게 분석을 진행하여 사용자에게 보여줄 수 있어야 한다. 기존 Amazon S3에 저장되어 있지 않은 repository에 대해서도 분석이 가능하도록, 사용자로부터 repository의 URL을 받을 수 있도록 하는 User Interface가 만들어져야 한다. 사용자의 실시간 분석 요청에 대해서도 반응할 수 있도록 User Interface가 필요하다.

2.2.6 Git inspector의 기능 추가


기존 Git inspector의 기능에 더하여, 사용자의 GitHub 활용성에 도움을 줄 수 있을 만한 분석 요소를 찾아서 분석기능을 추가하도록 한다.

2.3 개발 결과

2.3.1 시스템 기능 요구사항



1. 사용자가 분석하고자 하는 Github project URL을 입력 또는 선택한다. 사용자는 자신이 분석하고 싶은 Github repository가 데이터베이스에 유지되고 있는지 그 목록을 살펴볼 수 있다. 목록에 분석하고자 하는 Github repository가 없는 경우, 사용자는 직접 분석하고 싶은 Github URL을 입력한다.
2. GitHub repository 목록에 사용자가 분석하고자 하는 repository가 존재할 경우, Lambda에 올려져 있는 Git inspector가 Github URL에 해당하는 코드를 S3로부터 가지고 와서 분석한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

3. GitHub repository 목록에 사용자가 분석하고자 하는 repository가 존재하지 않을 경우, 사용자는 직접 Github URL을 입력한다. 이 경우 Git inspector는 Github로부터 바로 코드를 Cloning 해 와서 분석을 수행한다.
4. 분석 결과를 사용자에게 제공한다. 사용자는 분석 결과를 바탕으로 Contributor들을 분석 기준에 따라 정렬할 수 있다.

2.3.2 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.

2. 성능/처리량

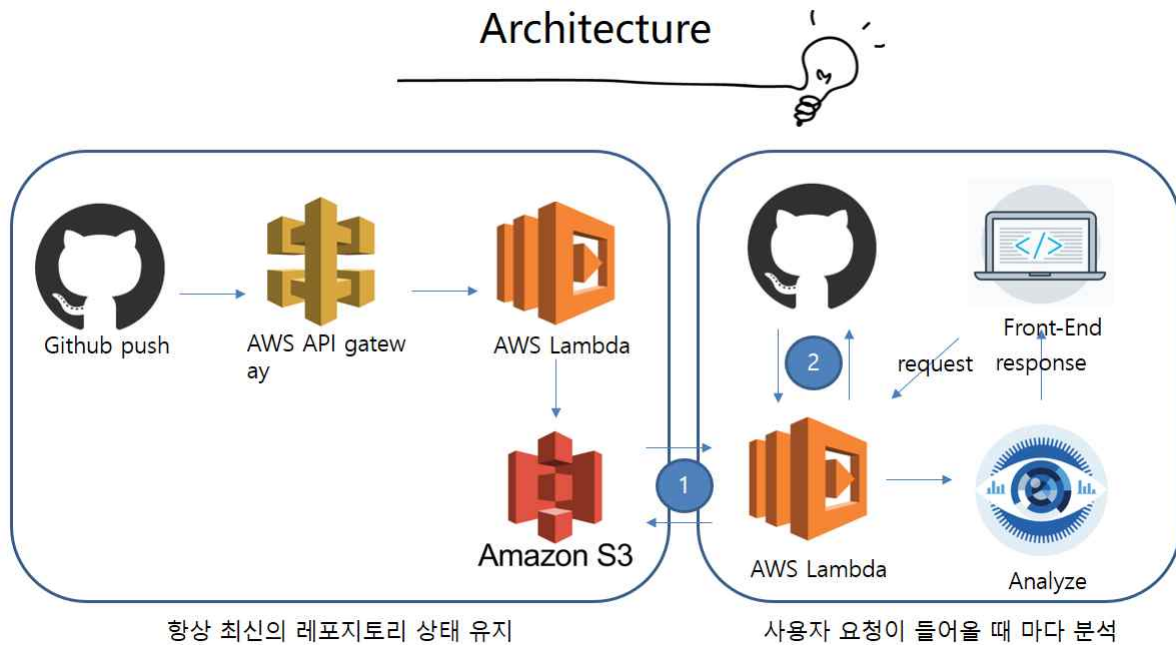
기존의 Git inspector 어플리케이션이 분석을 할 때마다 저장소를 cloning 하여 분석을 하였기 때문에, 분석 속도가 매우 느렸었다. 분석속도는 프로젝트의 크기가 크면 클수록 더 늘어났었다. 따라서 이 시스템의 성능은 최소한 기존의 Git inspector 보다 빨라야 하며, GitHub Repository를 유지하고 있는 경우에 프로젝트의 크기에 따른 분석 시간의 차이가 최소화되어야 한다.

3. 사용성

인터페이스 구성에 있어 사용자가 손쉽게 메뉴를 찾을 수 있도록 구성하여야 한다. 사용자는 구성된 Interface를 보고도 이것이 어떠한 기능을 수행할 것인지를 인지할 수 있어야 하며, 수행하고자 하는 기능을 불편함 없이 사용할 수 있어야 한다.


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

2.3.3 시스템 구조



2.3.4 결과물 목록 및 상세 사양

대분류	소분류	기능	형식	비고
내부 모듈	AWS Lambda1	AWS API gateway 로부터 Github project 에 push 가 일어났다는 것을 인식받아 Amazon S3 에 해당 github project 코드를 update 한다.		
	AmazonS3	Github project 들의 코드를 저장한다.		
	AWS Lambda2	1. 사용자로부터 github project url 요청을 받아서 해당 url 의 코드를 Amazon S3 에서 찾는다. 찾은 코드를 git inspector 를 활용하여 분석하고, 그 결과값을 사용자에게 제공한다. 2. 미리 저장되어있지 않는 레포지토리 정보를 사용자로부터 받았을 경우, 바로 GitHub 로부터 Cloning 받아 분석을 수행한다.		
UI	레이아웃	사용자가 쉽게 사용할 수 있도록 User Interface 를 제공한다.		
	연동	AWS Lambda2 에서 전송해주는 결과값을 받아서 올바르게 사용자에게 제공한다.		

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

2.4 기대효과 및 활용방안


2.4.1 기대효과

GitHub는 대규모의 소프트웨어 프로젝트를 진행하는 산업체부터, 대학교의 팀 프로젝트, 개인 프로젝트 사용자까지 보편적으로 사용하는 버전 관리 시스템이다. 이렇게 GitHub 저장소에 올려진 프로젝트들을 분석하여 자신이 어떤 기간에 얼마나 프로젝트를 진행했으며, 얼마나 많은 코드들이 삭제되었고 추가되었는지 등의 정보를 제공하면, 사용자는 자신이 GitHub를 얼마나 효율적으로 사용했는지에 대한 정보를 얻을 수 있다.

또한, GitHub 프로젝트에 참여한 기여자들 간의 기여도를 평가할 수 있다. 사용자가 GitHub 프로젝트에 대해 커밋을 한 횟수, 살아남은 코드 줄 수, 작성한 코드가 차지하는 비율 등을 제공 받음으로써 프로젝트에 얼마나 기여했는지를 평가할 수 있다.

2.4.2 활용방안


- 개인 프로젝트를 진행하는 경우, 자신이 기간별로 얼마만큼의 작업을 했으며, 코드의 삽입과 삭제가 얼마나 많이 이루어졌는지 등을 확인하는 데 활용되어, 더 효율적으로 GitHub를 사용할 수 있다.
- 대규모 소프트웨어 프로젝트를 진행하는 대학 수업에서, 프로젝트에 기여한 학생들의 기여도를 평가할 수 있다. 누가 작성한 코드가 얼마나 코드에 실제로 반영되었는지, 기간별로 얼마만큼의 작업이 이루어졌는지 등을 알 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

3 배경 기술

3.1 기술적 요구사항

- Github Webhook : Github의 repository에 특정 이벤트가 발생하는지를 탐지하는 API. Webhook을 이용해서 이벤트가 발생하면 HTTP Post 페이로드를 AWS API Gateway로 보낸다.
- AWS API Gateway : AWS Lambda와 함께 서버리스 서비스를 가능하게 해주는 REST 기반 HTTP 서비스를 제공한다. Github Webhook으로 받은 데이터를 AWS Lambda로 보낸다.
- AWS Lambda : EC2 없이 컴퓨팅 작업용 함수를 수행하게 해주는 서버리스 아키텍처 기반의 서비스이다. 본 프로젝트에서는 두 번의 AWS Lambda를 이용하는데, 첫 번째는 AWS API Gateway에서 받은 Github repository의 정보를 clone 해서 Amazon S3에 저장하기 위해 사용한다. Git inspector를 사용할 때 항상 repository를 clone하기 때문에 다소 시간이 걸린다. 이를 Amazon S3에 미리 clone 시켜 정보를 불러오는데 시간이 줄이기 위함이다. 두 번째 AWS Lambda는 사용자가 웹 어플리케이션 상에서 정보를 요구하면 Git inspector를 통해서 Amazon S3에 저장되어 있는 repository의 정보를 pull 해서 결과를 보여주기 위해 사용된다. Java, Node.js , C# 및 Python을 지원한다.
- Amazon S3 : 아마존에서 제공하는 객체 스토리지로, 원하는 양의 데이터를 저장하고 검색할 수 있도록 구축 되어있다. 위에서도 말했듯이 S3에는 Github repository의 clone을 저장하기 위해 사용한다.
- Git inspector : Github repository를 분석해주는 오픈소스이다. repository의 주소를 입력하면 commit 횟수, 삽입한 line 수 , 삭제한 line 수 등등에 대한 정보를 나타내준다.
- AWS Route 53 : 아마존에서 제공하는 것으로 웹 애플리케이션이 원활하게 작동하도록 도와준다. Route 53을 사용하여 향후 제작할 웹에 도메인을 부여한다.
- 웹 어플리케이션 : Javascript, HTML, CSS 그리고 Ajax 를 이용해서 유저가 사용하는 웹 어플리케이션을 제공한다. 유저가 Github의 주소와 조건을 입력하면 웹을 통해서 해당 주소의 repository를 분석해서 조건에 맞는 정보를 제공한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

3.2 현실적 제한 요소 및 그 해결 방안

3.2.1 하드웨어


- 별도의 하드웨어 서버 세팅이 필요 없이 클라우드 서비스에서 제공해 주는 소프트웨어 서비스들을 활용해서 웹 서버를 구축하는 프로젝트이다. 크게 사양에 구애 받지 않고 네트워크만 연결되어 있다면 문제 없이 사용할 수 있기 때문에 하드웨어 적으로 제한 되는 요소는 크게 없다고 보여진다.

3.2.2 소프트웨어

1. Git inspector가 제공하는 정보만으로는 Git을 잘 사용하고 있는지를 알 수 없을 수도 있다. Git inspector는 현재 commit 횟수, 삽입 line, 삭제 line, 변화 비율, 유저가 적은 모든 줄의 평균 지난 일 수(age), 유저가 적은 라인이 지금까지 살아남아 있는 비율(stability) 그리고 전체 줄 수 대비 주석 비율을 보여주고 있다. 이는 Git에 대해 잘 알고 있다면 이러한 정보를 통해 repository를 충분히 분석 할 수 있겠지만 잘 모른다면 Git을 얼마나 잘 사용하고 있는지에 대한 정보가 부족할 수 있다. 이를 해결하기 위해 Git inspector에 기능을 추가해 평가 요소를 제공하는 것을 고려할 수 있다.
2. repository를 pull 하는 것이 어느 정도의 시간이 걸릴지 체크해 봐야 한다. 현재는 따로 DB를 두지 않고 S3에 저장되어 있는 정보를 요청이 들어왔을 때 AWS Lambda를 통해서 pull을 해온 후에 유저가 지정해 준 조건에 맞추어 웹에 나타내주는 방식으로 설정했다. 하지만 만약 pull 역시 시간이 오래 걸린다면 따로 DB를 두어서 거기에 pull 정보를 담아 놓고 유저의 요청이 들어왔을 때 단순히 DB에서 정보를 꺼내어 웹으로 나타내 주는 방식도 있을 것이다. 하지만, 이렇게 되면 유저의 조건을 받아들이기에 어려움이 있기 때문에 가능하면 현재 방식이 좋을 것이라고 생각한다.

4 프로젝트 팀 구성 및 역할 분담

이름	역할
전호현	- Git inspector 기능 개선과 웹 제작
이근하	- AWS lambda, Amazon S3 등을 이용한 Back-end 구성
유문상	- AWS lambda, Amazon S3 등을 이용한 Back-end 구성
최원대	- Git inspector 기능 개선과 웹 제작

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

5 프로젝트 비용

항목	예상치 (MD)
아이디어 구상	3
관련 정보 수집	10
연구 및 테스트 코드 작성	5
내부 모듈 개발	30
웹 서비스 UI/UX 구성	15
내부 모듈과 웹 서비스 연동	10
프로젝트 테스트 및 유지보수	4
프로젝트 관련 문서작업	5
합	90


6 개발 일정 및 자원 관리

6.1 개발 일정

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

6.2 일정별 주요 산출물

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 ppt	2018-03-02	2018-03-09

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09


중간 보고	Back-end 구축 완료 Git inspector 를 통한 간단한 결과 페이지 산출물 : 1. 프로젝트 중간 보고서 2. 중간 소스 코드	2018-03-10	2018-04-12
구현 완료	Front-end 구축 완료 산출물: 1. 완전한 웹 어플리케이션 2. 최종 소스 코드	2018-04-13	2018-05-22
테스트	시스템 통합 테스트 산출물: 1. 버그 조사서	2018-05-23	2018-05-28
최종 보고서	최종 보고 산출물: 1. 최종 보고서 2. 웹 어플리케이션	2018-05-29	2018-05-29

6.3 인력자원 투입계획

이름	개발항목	시작일	종료일	총개발일(MD)
전호현	Git inspector 기능 개선과 웹 제작	2018-03-02	2018-05-22	30
유문상	AWS lambda, Amazon S3 등을 이용한 Back-end 구성	2018-03-02	2018-05-22	30
이근하	AWS lambda, Amazon S3 등을 이용한 Back-end 구성	2018-03-02	2018-05-22	30
최원대	Git inspector 기능 개선과 웹 제작	2018-03-02	2018-05-22	30

6.4 비 인적자원 투입계획

항목	Provider	시작일	종료일	Required Options
개발용 노트북	개별 지참	2018-03-02	2018-05-29	
Amazon S3	Amazon	2018-03-02	2018-05-29	
AWS lambda	Amazon	2018-03-02	2018-05-29	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	계획서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.5	2018-MAR-09

7 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/			