



국민대학교
전자정보통신대학
컴퓨터공학부


캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	뷰리풀(Viewreful)
팀 명	코드네이터(Codenator)
문서 제목	결과보고서

Version	1.3
Date	2017-05-24

팀원	김남현 (조장)
	김대환
	김상훈
	성민경
	송인엽
	최은정
	무하림

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 "뷰리풀(Viewreful)"을 수행하는 팀 "코드네이터(Codenator)"의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 "코드네이터(Codenator)"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	12조-최종결과보고서-뷰리풀.doc
원안작성자	김상훈, 성민경
수정작업자	김남현, 김대환, 송인엽, 최은정

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2017-05-20	김상훈	1.0	최초 작성	초안 작성
2017-05-21	송인엽	1.1	내용 수정	연구내용 추가
2017-05-22	김대환	1.2	내용 수정	기대효과 추가 및 수정
2017-05-23	최은정	1.3	내용 수정	맞춤법 및 오타 수정

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

목차

1. 개요	4
1.1 프로젝트 개요	4
1.2 추진 배경 및 필요성	5
1.2.1 코드 리뷰의 중요성	5
1.2.2 교내 코드 리뷰의 문제점과 원인 파악	6
1.2.3 코드 리뷰 시스템의 개발 필요성	8
2. 개발 내용 및 결과물	9
2.1 목표	9
2.2 연구/개발 내용 및 결과물	10
2.2.1 연구/개발 내용	10
1) 코드 리뷰 정책	10
2) 코드 분석 셸 스크립트	17
3) 웹서버 개발	18
4) 깃 웹 후 기능	19
2.2.2 시스템 기능 요구사항	19
2.2.3 시스템 비기능(품질) 요구사항	21
2.2.4 시스템 구조 및 설계도	23
2.2.5 활용/개발된 기술	24
2.2.6 현실적 제한 요소 및 그 해결 방안	28
2.2.7 결과물 목록	29
2.3 기대 효과 및 활용 방안	31
2.3.1 교육적 측면	31
2.3.2 기술적 측면	31
2.3.3 활용 방안	31
3. 자기 평가	32
4. 참고 사이트	33
5. 부록	34
5.1 사용자 매뉴얼	34
5.2 테스트 케이스	38

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

1 개요

1.1 프로젝트 개요

현재 애플, 구글, 페이스북 등 굴지의 해외 IT 기업뿐만 아니라 국내 대기업에서도 유능한 개발자를 선별하기 위해 입사 시 코딩 테스트를 필수로 진행하고 있다. 코딩 테스트는 기업에서 제공한 문제를 해결할 수 있는지 확인하는 것을 넘어서 효율적인 코딩을 하였는지까지 확인한다. 따라서 유능하고 인정받는 개발자가 되기 위해서는 문제 해결력과 동시에 효율적인 코드를 작성하는 능력을 갖춰야 한다.

하지만 이러한 동향과는 다르게, 현재 교내에서 학생들에게 주어지는 실습과 과제 수행 형식은 컴파일이 정상적으로 이루어지는지 혹은 주어진 입력에 따라 적절한 출력도 출력되는지를 판단하는 등의 문제 해결력만 중시하는 형식이다. 이렇다 보니 학생들은 문제를 정답인지 아닌지를 확인할 뿐 조건문이 남발하거나 예외처리가 하나도 되지 않는 등, 코드에 대한 문제점은 전혀 해결하지 않고 있다. 그리고 계속해서 이런 식으로 문제를 풀다 보니, 고학년이 돼서도 이것이 습관으로 남아있어 쉽게 고쳐지지 않게 된다.

이러한 문제의 해결책 중 하나는 '코드 리뷰'이다. '코드 리뷰'를 통해 강사가 학생의 코드의 문제점을 지적해주고 학생은 자신의 코드 문제점을 파악하여 코드 품질을 향상한다. 그러나 보통 1시간 15분으로 이루어진 수업시간에는 한 명의 강사가 30~40명 정도의 학생들을 코드 리뷰를 해주는 것은 현실적으로 불가능하다. 또 수업 외의 시간에 코드 리뷰를 해주려 해도 현재의 인력으로는 부담되는 상황이다.

본 캡스톤 팀은 이러한 문제점들을 해결하기 위해 첫 번째로 효과적인 코드 리뷰 프로세스를 위해 코드 리뷰 단계를 정의할 것이다. 두 번째로 학생들이 과제로 제출한 코드를 분석해 학생들 눈높이에 맞게 가공된 코드 리뷰 결과를 학생들에게 보여줄 것이다. 마지막으로 강사들에게 학생들의 코드를 분석한 결과들을 바탕으로 분류한 리포트를 제공해 수업시간에 실시간으로 코드 리뷰가 가능하게 할 것이다.

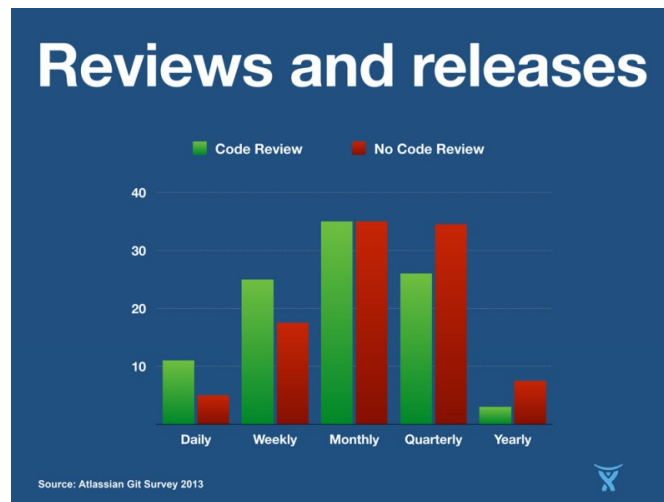
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

1.2 추진 배경 및 필요성

1.2.1 코드 리뷰의 중요성

프로그래밍을 시작하는 사람들 혹은 초보 개발자에게 빈번하게 일어나는 실수는 단지 빠르게 결과를 내는 것에 집중한다는 것이다. 이런 경우 코딩을 하면서 자신도 모르게 코드 내부의 구조, 스타일, 효율성 등을 생각하지 않게 되고 결국 질이 떨어지는 코드를 만들어 낸다.

이를 방지하기 위해 개발자는 스스로 주의를 기울여 코딩을 해야 한다. 하지만 무엇보다 효과적인 방법은 코드 리뷰를 받는 것이다. 코드 리뷰란 코드를 실행하지 않고 사람이 검토하는 과정을 통하여 코드에 숨어있는 잠재적인 결함을 찾아내고 개선하는 일련의 과정이다. 코드 리뷰를 통하여 개발자들은 스스로의 코딩 실력을 검토하고 수정, 다양한 기법들을 배울 수 있다. 코드 리뷰는 개발자의 코딩 실력을 높이는데 좋은 촉매제 역할을 하는 것이다.



[그림 1. Atlassian 코드 리뷰 유무에 따른 배포 주기 변화]

위 그래프는 Atlassian 에서 코드 리뷰 할 경우와 하지 않을 경우 배포 주기에 대한 통계자료이다. 그래프를 살펴보면 초반에는 코드 리뷰를 할 때 배포 주기가 더 길지만 시간이 더 지나면서 코드 리뷰를 할 때 배포 주기가 더 짧아지는 것을 볼 수 있다. 배포 주기가 짧아진다는 것은 코드 구조가 안정적이고, 코드의 수정이 용이함을 나타낸다. 이를 보았을 때 코드 리뷰를 함으로써 코드의 효율성이 높아지고, 코드의 구조적인 안정성을 높일 수 있다는 것을 알 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

현재 많은 IT기업들은 이러한 코드 리뷰의 중요성을 깨닫고 코드 리뷰 문화를 도입하고 있고 개발과정에서 코드 리뷰를 필수로 여기는 기업들도 많아지고 있는 추세이다. 또 이러한 시대적 흐름에 따라 코드 리뷰를 좀 더 편리하게 만들어 주기 위한 협업 툴들이 활성화되고 있다. 이렇듯 좋은 코드를 위한 코드 리뷰의 중요성이 부각되고 있다.

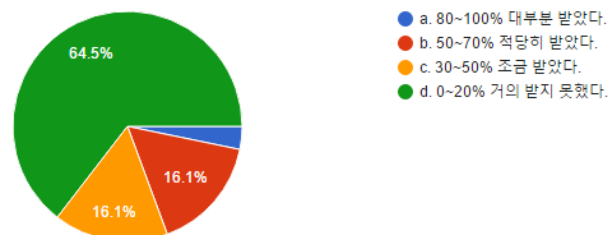
1.2.2 교내 코드 리뷰의 문제점과 원인 파악

본 캡스톤 팀은 현재 국민대학교 코드 리뷰 현황을 파악하기 위해 국민대학교 컴퓨터공학부 교수님들과 학생들 대상으로 코드 리뷰에 관해서 설문조사를 진행했다.

2. 자신의 코딩실력 향상을 위해 코드리뷰가 필요하다고 생각합니까? (응답 31개)



지금까지 프로그래밍 과목 수강 중 자신이 제출한 코드에 대한 리뷰를 받았습니까?
(응답 31개)



[그림 2. 교내 코드 리뷰 현황 조사(학생용)]

위 자료는 학생들 대상으로 실시한 설문 자료이다. 설문에 참여한 31명 학생 모두가 자신의 코딩 실력 향상을 위해 코드 리뷰가 필요하다고 생각했다. 하지만 그중 절반 이상의 학생들이 코드 리뷰를 거의 받지 못했다고 응답했다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

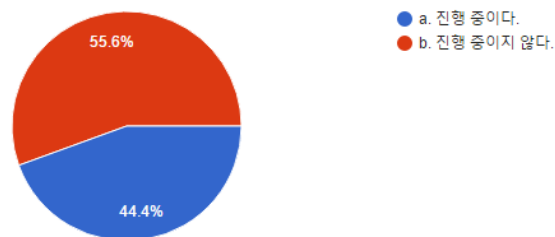
1. 코드리뷰가 학생들의 코딩실력 향상에 도움을 줄 수 있다고 생각하십니까?

(응답 9개)



1-a. 도움을 준다고 생각하신다면 학생들 과제에 코드리뷰를 진행 중이십니까?

(응답 9개)



[그림 3. 교내 코드 리뷰 현황 조사(교수용)]

위 자료는 교수용 설문지 자료이다. 교수용 설문도 마찬가지로 설문에 응답해주신 교수님 9명 모두 코드 리뷰가 학생들 코딩 실력 향상에 도움을 줄 수 있다고 응답했다. 하지만 그중 절반 이상의 교수님들이 코드 리뷰를 진행 중이지 않다고 응답했다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

코드 리뷰가 학생들 코딩 실력 향상에 큰 도움을 준다는 것을 알고 있음에도 코드 리뷰가 잘 이루어지지 못하는 원인을 심층적으로 알아보기 위해 프로그래밍 과목을 진행했던 교수님 네 분을 찾아가 코드 리뷰에 대한 인터뷰를 진행했다. 대부분의 교수님들은 시간적, 인적 자원 부족 등 현실적인 문제로 코드 리뷰를 진행하지 못하고 있다고 했다. 그중 유일하게 코드 리뷰를 진행한 윤성혜 교수님은 “학생들에게 코드 리뷰를 진행할 때 학생들의 코드에서 들여 쓰기, 무의미한 변수명, 코드 중복, 비효율적인 반복문 사용 등 기본적인 사항들이 지켜지지 않아 코드 리뷰를 할 때 적지 않은 시간이 소요된다. 그래서 정작 중요한 프로그램의 효율성, 구조, 알고리즘 등을 개선하기 위한 코드 리뷰를 해주는 것이 힘들어진다.”라고 조언했다. 또 지난 학기 3학년 전공과목인 고급 시스템 프로그래밍 과목을 강의했던 이민석 교수는 “작년에 3학년 전공과목 강의를 했는데, 프로그래밍 과제 채점 시 50% 이상의 학생들이 들여 쓰기, 주석 달기 여부, 무의미한 변수명 선언 등 기본적으로 프로그래밍할 때 지켜야 할 사항들을 지키지 않았다.”라고 조언했다.

설문조사와 교수님 면담 결과 현재 교내에서 코드 리뷰가 활발하게 이루어지지 못하고 있고, 그 원인은 학생들 개개인에게 코드 리뷰를 해주기 위해서는 많은 시간과 인적 자원이 필요하기 때문임을 알 수 있었다. 즉, 현재 학생들에게 코드 리뷰를 해주기 위한 시스템이나 절차가 정해져 있지 않아, 현재의 교육자원으로는 효율적인 코드 리뷰가 불가능하다.

1.2.3 코드 리뷰 시스템의 개발 필요성

현재 교내에서는 많은 프로그래밍 수업이 진행되고 있는데 코드 품질을 높이기 위해 코드 리뷰를 해주는 경우는 드물고 현실적으로 어렵다. 학생들의 수가 강사 수에 비해 많을 뿐 아니라 제대로 된 코드 리뷰를 해주려면 많은 시간이 걸리기 때문이다.

코드 리뷰가 효과적이고 효율적으로 이루어지기 위해서는 가장 먼저 효율적인 코드 리뷰 프로세스 정의가 필요하다. 또 시스템이 자동으로 지원할 수 있는 기본적인 코드 리뷰는 시스템에서 처리해 교수들의 일을 줄여줄 수 있는 서비스가 필요하다. 마지막으로 짧은 실습 시간에 코드 분석/분류 결과를 교수에게 제공해 실시간으로 코드 리뷰를 가능하게 해줄 수 있는 서비스가 필요하다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2 개발 내용 및 결과물

2.1 목표

본 프로젝트는 강사가 학생들에게 효과적인 코드 리뷰를 제공하고 학생들이 효율적으로 동작하는 코드를 만들 수 도록 도와주는 코드 리뷰 시스템 개발을 목표로 한다. 단순히 잘못된 점을 찾고 알려주는 시스템이 아닌 3단계로 나눈 효율적인 프로세스로 학생, 강사가 만족하는 코드 리뷰를 진행시키고, 학생과 강사가 원하는 코드 리뷰 자료를 바탕으로 Report 서비스를 제공한다.

1. 효과적인 코드 리뷰 프로세스 개발

3단계로 정의된 코드 리뷰 프로세스를 적용해 학생에게 효과적인 코드 리뷰를 해준다.

2. 학생 눈높이에 맞는 코드 분석 서비스 제공

정적 분석 툴들을 이용하여 코드를 분석하고 학생들 눈높이에 맞게 가공해서 Report를 제공한다.

3. 실시간 코드 분류 서비스 제공

실시간으로 학생들의 문제점 중 대표적인 문제를 포함하는 학생들의 코드를 제공해 짧은 시간 안에 다수의 학생에게 효과적인 코드 리뷰가 가능하도록 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2 연구/개발 내용 및 결과물

효과적인 코드 리뷰를 위해서 3 단계의 코드 리뷰 프로세스를 정의한다. 그리고 3 단계의 코드 리뷰 프로세스 중, 시스템이 자동으로 분석할 수 있는 1 단계의 코드 리뷰 프로그램을 개발한다. 이때, 본 캡스톤 팀만의 코드 분석 기준을 정의하고 그 기준에 따라 코드를 분석한다. 분석 결과는 학생 눈높이에 맞게 변환하여 학생에게 보고서를 웹으로 제공하고, 분석 결과를 모아, 본 캡스톤 팀만의 기준으로 분류하여 강사가 한눈에 학생들의 문제점을 파악할 수 있도록 코드 분류 보고서를 웹으로 제공한다.

2.2.1 연구/개발 내용

1) 코드 리뷰 정책

① 코드 리뷰 3단계 프로세스 정의

1 단계 >

학생이 제출한 코드에 대해서 변수의 이름, 함수의 이름, 주석, 공백 사용 등 코드 가독성을 높여주는 코드를 만들 수 있게 도와준다. 더 나아가 코드의 잠재적 문제, 복잡도 등을 알려주어, 학생이 미처 파악하지 못한 문제를 알고 수정할 수 있게 한다.

- 1 단계 코드 리뷰 단계는 시스템이 자동으로 코드 분석할 수 있도록 개발한 프로그램을 사용하여 진행한다. 코드 분석 프로그램은 현재 나와있는 코드 자동 분석 툴과 본 캡스톤 팀이 가지고 있는 코드(백준 알고리즘 사이트에 있는 코드, 비이공계 컴퓨터 프로그래밍 수업 때 제출된 실습 코드)를 분석한 데이터를 바탕으로 얻은 분석 기준을 이용한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2 단계 >

정답뿐만 아니라 효율적인 코드를 작성하는 방법에 대해 알게 한다. 중복된 코드의 함수화, 문제보다 적합한 문법, 자료구조 사용 유도 등을 알려주어 좀 더 문제에 맞는 코딩을 할 수 있게 만든다.

3 단계 >

주어졌던 문제와 비슷한 문제를 제시하고 2 단계에서 배운 효율적 코드 작성 방법을 사용할 수 있게 한다. 이 단계에서는 학생이 배운 코드 작성법을 자신의 것으로 체득할 수 있게 만드는 역할을 한다.

- 정의한 2,3 단계 모두 효율적인 코드를 작성하고 체득하기 위해서 필수적인 단계이다. 현재 코드 리뷰 시스템 조사 결과, 2,3 단계 역할을 해주는 코드 리뷰 시스템은 존재하지 않아, 현재 코드 리뷰 서비스들과 같이 강사 혹은 헬퍼가 학생들과 소통하며 코드 리뷰를 할 수 있도록 한다.

② 코드 분석 기준 정의

- Coding Convention (코드 가독성 높이기)

코드 가독성을 높이면 다른 프로그래머가 코드를 읽을 때 쉽게 이해할 수 있고, 코딩한 사람 또한 자신의 코드를 쉽게 설명할 수 있으므로, 프로그래머 간 소통이 원활하게 될 수 있다. 따라서 코드의 가독성은 매우 중요하며 가독성을 유지하기 위해서는 코드에 일관성을 부여하여야 한다. 이런 일관성은 초보 개발자일 때부터 습관을 기르는 것이 중요하다. 아래는 초보 코더들이 많이 하는 가독성 및 일관성 방해 요소들을 크게 5가지로 분류한 것이다.

A. Indentation Check

python은 들여 쓰기를 사용하여 코드들을 묶는다. 불필요한 들여 쓰기 혹은 문법에 맞지 않는 들여 쓰기, 탭과 스페이스를 같이 사용하여 다른 환경에서 열었을 때, 맞지 않는 들여 쓰기가 나오는 실수들을 체크한다.

B. Naming Check

코드 가독성을 높이는 데 가장 중요한 것은 Naming이다. 가독성을 매우 떨어뜨리는 변수명, 함수명을 체크한다. 예를 들면 x, xx, xxx, abc 등이 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

C. Comment Check

적절한 주석문은 자신 혹은 타인이 코드의 기능 등을 빠르게 이해할 수 있도록 도와준다. 인라인 주석 문, 블록 주석 문 등의 유무를 보고, 주석을 읽기 쉬운 형식으로 쓰고 있는지 확인한다.

D. WhiteSpace Check

공백 문자와 빈 줄을 잘 작성하는 것은 코드를 보는 사람의 가독성을 증가시킬 수 있다. 최상위 함수들 사이 혹은 클래스들 사이에는 두 줄을 띄워서 구분, 클래스 내의 함수들 사이는 한 줄을 띄워서 구분, 수식을 보기 쉽게 white space를 사용 등을 체크한다.

E. Code Format Check

위에서 언급한 whitespace, indentation을 제외하고도 코드의 가독성을 올리기 위해 중요한 format이 있다. 한 줄에 두 문장을 쓴다거나 여러 모듈을 한 번에 import 하는 등 가능하기는 하지만 좀 더 깔끔한 코드(clean code)를 만들기 위한 규정들을 체크한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

● Code Problem (코드의 잠재적 문제 해결하기)

아무리 간단한 코드를 작성하더라도 한 번에 완벽한 코드를 짜기는 거의 불가능하며 실행되는 코드이더라도 잠재적 문제점들과 다양한 경고 메시지를 무시하는 경우가 비일비재하다. 아래는 학생들에게 효율적인 리뷰를 하기 위해 코드 문제점이 발생하는 위치 기준으로 4가지를 제시한다.

A. Module

프로그래밍을 할 때 다양한 모듈을 사용하여 코드를 작성한다. 학생들이 모듈을 사용하면서 발생하는 error와 warning을 분석하여 사용자에게 제공한다. 또한 가능한 부분의 refactor를 제안한다.

B. Function

코드의 재사용성을 높이기 위해 함수를 사용한 코딩은 필수적이다. 학생들이 함수를 사용하면서 발생하는 error와 warning을 분석하여 사용자에게 제공한다. 또한 가능한 부분의 refactor를 제안한다.

C. Class

클래스는 객체지향 언어 사용에 기본적인 것이지만, 처음 배울 때 있어서 많은 실수와 문제를 일으킨다. 클래스를 사용하며 생기는 error와 warning을 분석하여 사용자에게 제공한다. 또한 가능한 부분의 refactor를 제안한다.

D. Statement

코드에는 위에서 제시한 module, function, class를 제외한 평범한 statements들도 존재하며 이런 평문에서 발생하는 error와 warning을 분석하여 사용자에게 제공한다. 또한 가능한 부분의 refactor를 제안한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

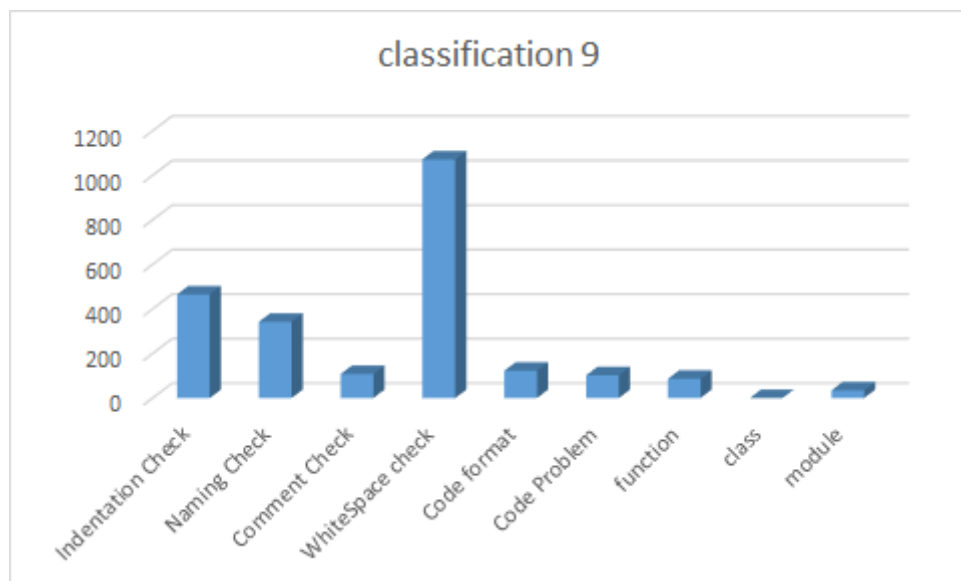
④ 카테고리별 가중치 부여

본 캡스톤 팀은 python 코드 리뷰의 테스트 자료로 백준 알고리즘 사이트(<https://www.acmicpc.net/>)에서 많은 사람들이 풀었던 10 문제의 Python 코드와 국민대학교 비이공계 학생을 대상으로 진행 중인 Python 수업에서 나온 코드, 약 1000개의 코드를 이용하였다.

많은 테스트 결과를 쉽게 확인하기 위하여, 엑셀을 이용하였다. 엑셀을 이용하여, 각 코드 리뷰 카테고리 별로 리뷰 결과로 나온 이슈들을 표로 정리하고, 그래프로 표현하였다.

1 2		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	
	1	여러메세지	합	평균	105552.py	1089808.py	1118412.py	122713.py	1284160.py	1491975.py	1492534.py	1498125.py	1506788.py	2263122.py	2290273.py	3032633.py	3032683.py	3032735.py	344012.py	3450083.py	3667020.py	3703957.py	3928478.py	4407980.py	442060.py	4490201.py	451
	2	Indentation Check	465	9.69	0	36	0	50	0	0	0	0	0	0	42	0	0	0	0	0	0	0	76	0	12	11	
+	13	Naming Check	343	7.15	14	15	0	10	4	13	9	0	0	0	12	0	0	0	12	13	12	7	11	0	12	0	
+	19	Comment Check	108	2.25	1	3	0	3	6	1	5	0	0	0	3	0	0	0	3	4	2	2	3	0	2	1	
+	31	WhiteSpace check	1072	22.3	34	9	54	56	0	8	10	6	31	29	37	6	8	7	70	35	3	2	29	8	48	17	
+	69	Code format	123	2.56	4	5	0	10	1	8	4	0	0	4	1	1	2	0	2	3	0	0	2	1	19	2	
+	104	Code Problem	102	2.13	0	3	0	8	1	0	2	0	0	0	7	0	0	0	6	9	5	3	7	0	4	0	
+	180	function	87	1.81	0	6	0	2	4	6	4	0	0	0	2	0	0	0	3	2	0	4	2	0	5	0	
+	195	class	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
+	233	module	35	0.73	0	5	1	0	0	0	2	1	1	1	0	1	1	1	0	1	0	0	1	0	0	2	
+	247	error	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
+	278	Total	2335	48.6	53	82	55	139	16	36	36	7	32	34	104	8	11	8	96	67	22	18	130	10	102	33	

[그림4. 분석 결과를 이용하여 만든 분류표 (분류기준 : 왼쪽 세로축, 코드 : 상단 가로축)]



[그림5. 분석 기준에 따른 발생 빈도 막대 그래프]

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

위는 백준 알고리즘 사이트에서 가져온 'DFS와 BFS'의 문제를 풀었던 사람들의 python 코드를 코드 리뷰 시스템으로 분석한 결과이다.

(<https://www.acmicpc.net/problem/1260>)

여러 문제의 코드 분석 결과, 대부분의 코드에서 Indentation, Naming, WhiteSpace 분석 기준에서 제일 많은 이슈가 나왔다. 그러나 Indentation의 경우, python에서 강제적으로 Indentation을 적용하여야 했기에, Indentation 분석 기준의 이슈가 가독성을 크게 떨어지는 이슈가 아니었다. 또한, WhiteSpace 분석 기준은 함수 간, 클래스 간 등의 공백을 규칙적으로 두어 가독성을 높이는 기준이지만 문제 풀이자들이 시스템이 정한 기준에는 맞추지 않을 뿐, 어느 정도 자신만의 기준으로 whiteSpace를 두고 있었다. (예를 들어, 클래스 간 공백 2 줄의 space가 필요하나, 풀이자는 1 줄의 space를 두는 경우) 본 캡스톤 팀은 이런 경우 WhiteSpace 분류 기준이 엄청나게 가독성이 떨어지는 이슈를 발생시킨다고 생각하지 않았다. 하지만 Naming 분류 기준은 'aa', 'funcresult' 등 가독성을 매우 떨어뜨리는 변수, 함수, 클래스 명의 문제를 지적하는 이슈이었다. 따라서 이런 이유에 있어서, 본 캡스톤 팀은 Indentation 분석 기준과 WhiteSpace 분석 기준에서 나오는 이슈에 대한 가중치를 줄였다. (이슈 1개 → 이슈 0.8 개) 또한, Naming 분석 기준에서 나오는 이슈에 대한 가중치를 높였다. (이슈 1개 → 이슈 1.3 개)

⑤ 코드 리뷰 이슈별 가중치 부여

1	2	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	1	에러메세지	합	평균	105552.py	1089808.py	1118412.py	122713.py	1284160.py	1491975.py	1492534.py	1498125.py	1506788.py	2263122.py	2290273.py	3032633.py	3032683.py	3032735.py	344012.p	3450083.py	3667020.py	3703957.py	3828478.py	4407980.py	442060.p	4490201.py
	2	Indentation Check	465	9.69	0	36	0	50	0	0	0	0	0	0	42	0	0	0	0	0	0	0	76	0	12	11
+	13	Naming Check	343	7.15	14	15	0	10	4	13	9	0	0	0	12	0	0	0	12	13	12	7	11	0	12	0
+	19	Comment Check	108	2.25	1	3	0	3	6	1	5	0	0	0	3	0	0	0	3	4	2	2	3	0	2	1
	20	E114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	21	E115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	E116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	E261	2	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	24	E262	2	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	E265	26	0.54	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
	26	E266	1	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	27	c0111	77	1.6	1	3	0	3	6	1	3	0	0	0	3	0	0	0	3	3	2	2	3	0	2	0
	28	c0112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	29	w0410	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	30	w0511	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-	31	WhiteSpace check	1072	22.3	34	9	54	56	0	8	10	6	31	29	37	6	8	7	70	35	3	2	29	8	48	17
+	69	Code format	123	2.56	4	5	0	10	1	8	4	0	0	4	1	1	2	0	2	3	0	0	2	1	19	2
+	104	Code Problem	102	2.13	0	3	0	8	1	0	2	0	0	0	7	0	0	0	6	9	5	3	7	0	4	0
+	180	function	87	1.81	0	6	0	2	4	6	4	0	0	0	2	0	0	0	3	2	0	4	2	0	5	0
+	195	class	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
+	233	module	35	0.73	0	5	1	0	0	0	2	1	1	1	1	1	1	1	0	1	0	0	0	1	0	2
+	247	error	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
+	278	Total	2335	48.6	53	82	55	139	16	36	36	7	32	34	104	8	11	8	96	67	22	18	130	10	102	33

[그림6. 분류 기준별 세부 이슈 정보 확인]

분석 기준으로 코드 이슈에 가중치를 주는 것은 엄청나게 포괄적인 가중치 부여 방법이라 생각하여 약 400개의 각 코드 분석 이슈에 본 캡스톤 팀의 기준에 따라 1~5 만큼의 가중치를 주었다. 기준은 아래와 같다.

가중치	내용
1	큰 의미가 없다.
2	큰 의미는 없지만 많이 발생한다.
3	코드 개선을 위해 참고할 만 하다.
4	코드 개선을 위해 필수적이다.
5	코드 개선을 위해 필수이고 많이 발생한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

3) 웹 서버 개발

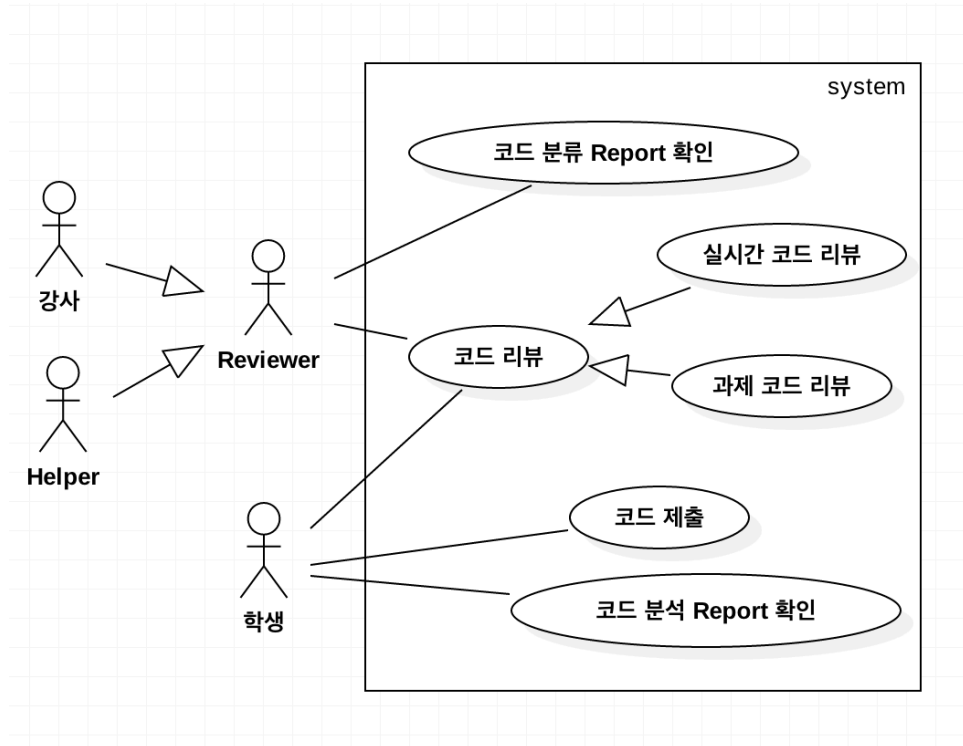
웹 서버는 Nodejs로 개발되었으며, express 모듈을 이용하여 구현하였다. DB는 NoSql 인 MongoDB를 사용하였다. 서버의 기능은 학생들이 깃에 올린 코드를 가져와 분석 모듈인 쉘 스크립트를 구동한다. 이에 구동된 쉘 스크립트는 분석부터 MongoDB에 분석 결과를 저장까지 하며, 서버는 DB에서 필요한 데이터를 가져와 client에게 가공해주는 역할을 한다.

4) 깃 웹 훅(Git Web Hook) 기능

웹 훅(Webhook)을 사용하면 Github에서 특정 이벤트를 구독하는 통합 기능을 구축하거나 설정할 수 있다. 이러한 이벤트 중 하나가 확인되면 Github에서는 웹 훅(Webhook)에 등록된 URL(코드 리뷰 시스템의 서버)에 HTTP POST payload(모든 이벤트에 대한 정보)를 보낸다. 이때, HTTP POST payload는 현재 Github에 일어난 모든 이벤트 정보 및 정보에 대한 구체적인 내용을 말하며, 등록된 URL이란 정보를 보낼 서버이다. 따라서 본 캡스톤 팀은 이 기능을 사용하여 학생들이 payload url 에 Viewreful 서버 주소를 등록하면 학생이 자신의 Github repository에 소스를 push 할 때마다 코드를 가져와 분석/분류를 진행하도록 개발하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.2 시스템 기능 요구사항



[그림 8. 유즈 케이스 다이어그램]

학생 >

A. 코드 제출 – Github < 완료 >


학생은 강사가 제시한 데이터 표준에 맞춰 코드를 제출한다. (학생은 코드 리뷰 웹 사이트와 연결된 GitHub Repository에 자신의 코드를 제출한다.)

B. 코드 분석 Report 확인 < 완료 >

학생은 제출한 코드에 대해서 실시간으로 코드 분석 Report(Web Page)를 받을 수 있다.

C. 코드 리뷰 < 완료 >

학생은 웹 페이지에서 자신의 가독성이 떨어지는 코드 부분을 찾고, 강사들의 코드 리뷰를 웹 사이트 혹은 Github에서 받아 볼 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

Reviewer(강사, 조교) >

A. 코드 분류 Report 확인 < 완료 >

강사는 전체 학생들의 코드 분석 결과를 정리해놓은 Report(Web page)를 받을 수 있다.

B. 코드 리뷰 < 완료 >

i. 실시간 코드 리뷰

실습 시간, 강사는 실시간으로 학생 전체의 코드와 분류된 코드 분석 결과를 볼 수 있다. 분류된 결과에서는 강사가 학생들의 전체적인 코드 문제점을 카테고리 별로 확인할 수 있고, 대표적인 문제, 추천 코드 등을 확인할 수 있다.

ii. 과제 코드 리뷰

실습 시간 이외에도, 강사는 수시로 코딩 과제에 대한 코드 리뷰가 가능하다. 강사가 직접적인 코멘트를 달고 싶을 때는 Github url에 바로 접속할 수 있도록 하여 코멘트를 달수 있도록 하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.3 시스템 비기능(품질) 요구사항

제품 요구 사항	사용성 요구 사항 <달성>	<ul style="list-style-type: none"> - 사용자가 다른 매뉴얼을 배우지 않아도 코드 리뷰를 받을 수 있도록 기존에 존재하는 서비스로부터 데이터를 받아올 수 있도록 한다. - 학생이 코드 분석 보고서를 받았을 때, 라인 번호와 정확한 위치를 제시하여 어느 부분에서 자신의 문제가 일어났는지 알 수 있도록 한다. - 강사가 학생 코드 분류 보고서를 받았을 때, 빈도수를 사용하여 학생들에게 가장 많이 발생한 문제점을 제시한다.
	신뢰성 요구 사항	<ul style="list-style-type: none"> - 여러 플러그인을 합쳐서 만든 본 캡스톤 팀의 코드 리뷰 프로그램을 사용했을 때의 결과와 플러그인들을 단독으로 사용했을 때의 결과를 비교했을 때, 같은 이슈가 나올 확률이 98% 이상 되도록 한다. <달성> - 해당 소프트웨어가 제시하는 코드 수정 예시를 적용했을 때 정확하게 이슈가 수정되도록 한다. <미달성>
	효율성 요구 사항 <달성>	<ul style="list-style-type: none"> - 동시에 코드를 제출한 사용자가 100명 이하일 경우 1분 이내, 1000명 이하일 경우 5분 이내에 분석을 완료할 수 있도록 한다.
조직 요구 사항	배포 요구 사항 <달성>	<ul style="list-style-type: none"> - 최종 서비스는 웹 서비스로 배포할 수 있도록 한다.
	구현 요구 사항 <달성>	<ul style="list-style-type: none"> - 제품의 구현에 있어 개발팀은 Incremental 기법을 활용하도록 한다. - 개발팀을 개발 환경을 통일시키도록 하며 협업 및 코드 관리 도구로서 GitHub를 사용하도록 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

	표준 요구 사항 <달성>	- 개발 환경은 해당 문서의 3.1.1 의 첫 번째 항목인 개발한 환경을 표준으로 한다.
--	---------------------	---

<미달성> 신뢰성 요구 사항

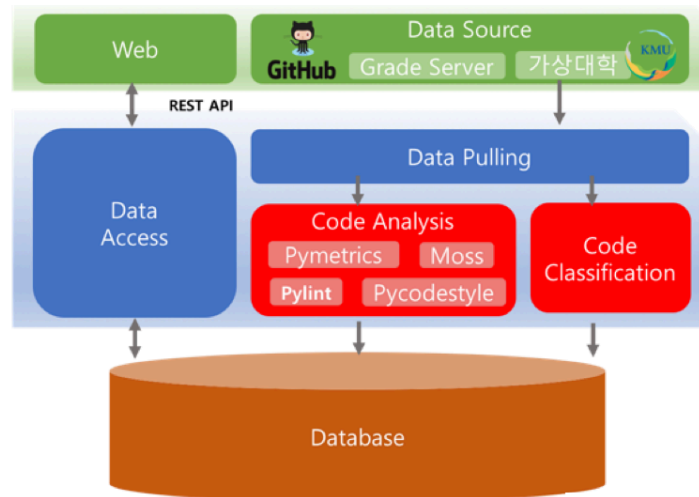
해당 소프트웨어가 제시하는 코드 수정 예시를 적용했을 때 정확하게 이슈가 수정되도록 한다.

문제 분석>

본 캡스톤 팀의 코드 리뷰 시스템에서 사용되는 코드 리뷰 툴인 pylint 의 refactoring 기능이 모든 코드 이슈에 대해서 수정 예시를 제시해주는 것이라고 착각하였다. 그러나 좀 더 조사해 보니, 모든 코드 이슈가 아닌 매우 일부의 코드 리뷰 이슈에 대해서 수정 예시를 제공해 주거나 refactoring 에 대한 설명을 제공하는 것뿐이었다. 본 캡스톤 팀은 사실상 상당한 양의 코드 리뷰 이슈 하나하나에 대한 수정 예시를 만드는 것은 비효율적인 작업이라 생각하여, 이 부분을 포기하고, 그 대신 코드 리뷰 이슈를 학생, 강사들이 편리하게 확인할 수 있도록 하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.4 시스템 구조 및 설계도



[그림 9. 시스템 구조도]

< Data Source >

학생이 GitHub 에 올린 코드를 가져온다.

< Data Pulling>

학생이 시스템이 제시하는 형식대로 코드를 GitHub 에 올렸을 때, GitHub API 중 Web Hook 기능을 이용하여 코드를 분석 시스템에 전달한다.

< Code Analysis >

코드 분석 플러그 인들을 모아 코드 분석을 한다. 분석 결과 나온 이슈들을 본 캡스톤 팀만의 분석 기준에 맞게 나눈 뒤, json 형태로 Database 에 저장한다.

< Code Classification >

Code Analysis 모듈에서 얻은 분석 결과를 이용하여 강사가 보기 쉽게 데이터를 분석 기준에 따라, 분석 결과 본 캡스톤 팀의 이슈 중요도에 따라 재 정렬한다.

< Web UI >

학생이 제출한 코드에 대한 코드 분석, 분류 결과를 웹 페이지로 제공한다.

< Data Access >

학생, 강사들이 웹 페이지에 접속했을 때, Database 에서 코드 리뷰 결과 분석, 분류된 이슈들을 가져와 웹 페이지에 전달한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.5 활용/개발된 기술



- pyLint(파이선 정적 분석 도구)

pylint는 python코드의 오류를 검사하고 코딩 표준을 적용하여 코드 smell을 찾는 파이션 정적 분석 도구다. Pylint는 Basic checker, Typecheck checker, Classes checker 등등을 사용하여 코딩 컨벤션부터 문법 오류까지 확인하여 메시지를 준다.

- Pymetrics(파이선 사이클로 매틱 복잡도 분석 도구)

Pymetrics는 python 코드의 사이클로 매틱 복잡도(Cyclomatic complexity)를 측정하는 도구이다. 각 함수 별 복잡도와 전체적인 코드의 복잡도를 계산하여 준다.

Moss

- Moss (Measure Of Software Similarity)

Moss는 Stanford에서 만든 코드 유사성의 척도를 검사하는 도구이다. Moss를 사용하여 학생들의 코드의 유사성을 비교하여 url를 제공하여 강사들은 각 과제별 학생들의 코드 Copy Check을 간편하게 실시할 수 있도록 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

- pycodestyle(파이선 코드 분석 도구)

pycodestyle은 파이선 코드 스타일 가이드인 pep8에 기반한 파이선 코드 정적 분석 도구이다. pycodestyle은 runtime error를 비롯한 Indentation, Whitespace, Blank line , Line length 등 기본적인 코드 스타일에 관한 메시지를 준다



- GCP(구글 클라우드 플랫폼)

GCP(Google Cloud Platform)는 안전한 클라우드 서비스 플랫폼으로서, 컴퓨팅 파워, 데이터베이스 스토리지, 콘텐츠 전송 및 기타 기능을 제공한다. 그중 GCE(Google Compute Engine)을 통해서 VM을 생성하고 웹서버를 구축하여 코드 분석 및 분석 결과를 보여준다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24



- NodeJs & express (웹 서버 및 웹 프레임워크)

NodeJs는 확장성 있는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하여 Nonblocking I/O 와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다. NodeJs는 V8(자바스크립트 엔진) 위에서 동작하고, 웹 서버와 같이 확장성 있는 네트워크 프로그램 제작을 위해 고안되었다.

express는 웹 및 모바일 애플리케이션을 위한 일련의 강력한 기능을 제공하는 간결하고 유연한 Node.js 웹 애플리케이션 프레임워크이다.



- MongoDB

몽고DB는 크로스 플랫폼 도큐먼트 지향 데이터베이스 시스템이다. NoSQL 데이터베이스로 분류되는 몽고DB는 JSON과 같은 동적 스키마형 문서들을 선호함에 따라 전통적인 테이블 기반 관계형 데이터베이스 구조의 사용을 삼간다. 이로써 특정한 종류의 애플리케이션을 더 쉽고 더 빠르게 데이터 통합을 가능케 한다. 서버단에서 분석 결과, 분석된 이슈들의 정보 등을 저장하기 위해 사용한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24



- Git & GitHub

GitHub는 분산 버전 관리 툴인 깃(Git)을 사용하는 프로젝트를 지원하는 웹호스팅 서비스이다. Github의 웹 훅(Webhook)을 사용하면 Github에서 특정 이벤트를 구독하는 통합 기능을 구축하거나 설정할 수 있다. 이러한 이벤트 중 하나가 확인되면 Github에서는 웹 훅(Webhook)에 등록된 URL(코드 리뷰 시스템의 서버)에 HTTP POST payload(모든 이벤트에 대한 정보)를 보낸다. 이때, HTTP POST payload는 현재 Github에 일어난 모든 이벤트 정보 및 정보에 대한 구체적인 내용을 말하며, 등록된 URL이란 정보를 보낼 서버이다. 따라서 본 캡스톤 팀은 이 기능을 사용하여 학생들이 payload url 에 Viewreful 서버 주소를 등록하면 학생이 자신의 Github repository에 소스를 push 할 때마다 코드를 가져와 분석/분류를 진행하도록 개발하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.6 현실적 제한 요소 및 그 해결 방안

제한 요소	설명	해결 방안
Viewreful 웹 사이트 사용 방법의 익숙하지 않음	이 소프트웨어를 직관적으로 사용하긴 어렵고 스스로 사용방법을 알 수 없다.	각 사용자 타입 별로 프로그램을 사용하는 방법, 해야 할 행동을 매뉴얼로 제작하여 체계적으로 알려준다.
시스템이 할 수 있는 수준 이상의 분석을 필요로 하는 경우	코드 분석을 시스템이 자동으로 하는 데는 한계가 있다. 코딩 컨벤션과 예측 가능한 예러들은 분석이 가능하지만 사용자가 복잡한 알고리즘이 적용된 코드의 분석을 원하고 더 효과적인 알고리즘을 제시하는 것은 불가능하다.	본 시스템이 할 수 있는 범위를 넘어서는 문제들은 사람이 직접 확인하고 코드리뷰를 진행하도록 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.2.7 결과물 목록

(1) 자동 빌드 및 분석 시스템

대분류	소분류	기능	형식	비고
소프트웨어	Data Pulling	학생들의 코드를 서버로 가져오는 pulling 기능	Api, 모듈	
	Code Analysis	코드 분석 기능	모듈	
	Code Classification	분석결과를 이용한 코드 분류 기능	모듈	
	Data Access	코드 분석 및 분류 결과 전달 기능 분석 및 분류 결과 Database 저장 및 전송 기능,	모듈	
	Web	학생/강사용 보고서 제공 기능	웹	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

(2) 문서

대분류	소분류	기능	형식	비고
기술문서	문서	데이터 형식 표준화	Pdf	
	문서	코드 리뷰 단계 정의	Pdf	
	문서	코드 분류 기준 정의	Pdf	
	문서	실사례 적용 결과	excel	
	문서	강사/학생용 Report 형식	pdf	
	문서	코드 이슈 한글 해석	pdf	
	문서	코드 이슈 분류 결과 분석	excel	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

2.3 기대효과 및 활용방안

2.3.1 교육적 측면

● 강사 측면

1, 2, 3 단계로 나뉜 코드 리뷰 시스템으로 학생에게 단순히 '동작 가능한' 코드를 작성하게 하는 것이 아닌, '효율적으로 동작 가능한' 코드를 작성하도록 지도하는 데 간편해진다. 그리고 원래 강사가 하던 1 단계 코드 리뷰를 SW 로 대체하게 되어 강사의 일을 덜어주고, 그로 인해 강사가 2, 3 단계 코드 리뷰를 더욱더 세세하게 지도해 줄 수 있다.

● 학생 측면

학생이 문제 해결에만 목적을 둔 코딩이 아닌 효율적이고 의미가 명확한 코딩을 하게끔 만들 수 있다. 또 학생이 단계별 코드 문제점을 알게 하여 자신의 능력을 향상할 수 있는 방향을 알게 할 수 있으며 좋은 코딩 습관을 형성할 수 있다.

2.3.2 기술적 측면

- 현재로는 기계만으로 불가능한 2, 3 단계의 코드 리뷰를 강사들의 코드 리뷰 데이터를 수집 및 축적하여 인공지능 기술과 결합해 모든 코드 리뷰를 SW 만으로 가능하게 한다.
- 학생들의 코드 분석으로 나오는 데이터를 수집 및 축적하여 자체적인 코드 분석 플러그인을 개발할 수 있다.

2.3.3 활용 방안


- 코딩 교육 중요성의 증가 추세와 더불어 현재 우리 학부에 맞춰진 시스템뿐만 아니라 다른 학교 및 교외 등 다방면에서 활용 가능하게 일반화하여 사용할 수 있다.
- 기계적으로 가능한 1 단계 코드 리뷰를 제외하더라도 2,3 단계에서 강사가 해야 하는 일이 많기 때문에 학부에서 운영 중인 헬퍼를 활용하여 학생들 간의 코드 리뷰를 활성화한다.
- 실습 및 과제뿐만 아니라 개인적으로 코드 리뷰를 받을 수 있도록 시스템을 확장하여 사용 가능하다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

3 자기평가

본 캡스톤 팀이 진행한 코드 리뷰 시스템 개발 프로젝트에는 개발 관련 이슈보다 코드 리뷰 정책 설립에 대한 이슈가 매우 중요했다. 코드 리뷰에 대한 정책 설립을 위해서 많은 회의를 하고, 본격적인 시스템 제작에 앞서, 간단히 만든 분석 프로그램에 테스트 데이터를 넣어 나온 결과를 분석하고, 그것을 다시 정책으로 만드는 과정에서 생각보다 꽤 많은 시간이 걸렸다.

그 결과 정책적인 부분은 어느 정도 확립하였으나, 개발 시간이 매우 부족하였고 코드 리뷰 가능 언어는 python 밖에 없었다. 추후 지속적으로 개발로 C, C++, Java 등의 언어를 추가하고, 더 많은 데이터를 현 코드 리뷰 시스템에 적용시켜 정책적으로 더욱더 명확하고, 어느 상황에도 적합한 코드 리뷰를 할 수 있도록 코드 리뷰 시스템을 운영할 것이다. 또한 사용자인 학생과 강사들의 피드백을 지속적으로 시스템에 반영하여, 사용자에게 맞는 시스템이 되도록 할 것이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

4 참고 사이트

번호	제목	출처 및 사이트
1	PYLINT	https://www.pylint.org/
2	PYCODESTYLE	https://pypi.python.org/pypi/pycodestyle
3	Moss	https://theory.stanford.edu/~aiken/moss/
4	PEP8	https://www.python.org/dev/peps/pep-0008/
5	데이터 표준화	http://www.dbguide.net/db.db?cmd=view&boardUid=12803&boardConfigUid=9&categoryUid=216&boardIdx=30&boardStep=1
6	안좋은 코드의 예	https://wikidocs.net/597
7	Cyclomatic complexity	http://story.wisedog.net/sw-metric-cyclomatic-complexity-%EB%B3%B5%EC%9E%A1%EC%84%B1-%EC%A7%80%ED%91%9C/
8	코드리뷰의 중요성	http://egloos.zum.com/swprocess/v/2462137
9	Complexity check tool	http://www.whiteboxtest.com/Code-Complexity-Tools.php
10	백준 알고리즘	https://www.acmicpc.net/

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

5 부록

5.1 사용자 매뉴얼

- Filtering & Sorting

- 학생들의 코드 리뷰 결과로 나온 이슈들 중에서 가장 많은 횟수로 나온 이슈와 그 이슈에 대한 설명 제공

Filtering & Sorting

#	Issue	Count	Description
1	C0103	661	유효하지 않은 "%s" 이름 "%s"
2	C0326	564	% s 공간 % s % s % s % s
3	W0312	356	%ss 대신에 %ss로 들여쓰기를 했습니다.
4	W0141	170	사용된 내장 함수 %r
5	C0111	150	누락 된 %s의 docstring
6	W0621	128	외부 범위 (% s 출)에서 이름 % r를 재정의합니다.
7	W0311	90	부정적하 들여쓰기 %c %c 변경되 예상 %c

- Copy Check

- 학생들의 코드를 비교하여 카피 체크 결과 제공

CopyCheck < refresh

<http://moss.stanford.edu/results/15813576>

/20120019/ (14%)	/20120075/ (17%)
25-32	8-14

```

/20120019/
>>> file: 20120019.py
def DFS(v, z, chk):
    chk[v] = 1
    ans = str(v) + ' '
    for i in z[v]:
        if(chk[i]==0):
            ans = ans + DFS(i, z, chk)
    return ans

s = input()
s = s.split()
n = int(s[0])
m = int(s[1])
v = int(s[2])

z = []
for i in range(n+1):
    z.append([])

for i in range(m):
    s = input()
    s = s.split()
    a = int(s[0])
    b = int(s[1])

    z[a].append(b)
    z[b].append(a)

for i in range(n+1):
    z[i].sort()

chk = []
for i in range(n+1):
    chk.append(0)

print(DFS(v, z, chk))

for i in range(n+1):
    chk[i] = 0
    ans = ''
    x = [v]
    chk[v] = 1
    while(len(x)>0):
        v = x[0]
        x = x[1:]
        ans = ans + str(v) + ' '
    for i in z[v]:

```

```

/20120075/
>>> file: 20120075.py
n, m, start = map(int, input().split())
a = []
for i in range(1, n+1):
    a.append([])
for i in range(m):
    u, v = map(int, input().split())
    a[u].append(v)
    a[v].append(u)
for i in range(n+1):
    a[i].sort()

check = []
for i in range(n+1):
    check.append(False)


def dfs(a, check, now):
    if not check[now]:
        check[now] = True
    else:
        return
    print(now, end=' ')
    for k in a[now]:
        dfs(a, check, k)

dfs(a, check, start)
print()

check = []
for i in range(n+1):
    check.append(False)

def bfs(a, check, now):
    queue = []
    queue.append(now)
    # print(now, end=' ') #프린트 여기서 하는게 마님
    check[now] = True
    while queue:
        now = queue[0]
        print(now, end=' ')
        queue = queue[1:]
        for i in a[now]:
            if not check[i]:
                queue.append(i)
                check[i] = True

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codinator)	
	Confidential Restricted	Version 1.3	2017-05-24

- Recommend Code

- 각 분석 기준 별로 코드 리뷰가 필요한 대표적인 학생 코드(Github url) 제공

Recommend Code

```

Indentation : https://github.com/Viewreful1/20120091/blob/master/20120091.py
Naming : https://github.com/Viewreful1/20120091/blob/master/20120091.py
Comment : https://github.com/Viewreful1/20120091/blob/master/20120091.py
WhiteSpace : https://github.com/Viewreful1/20120064/blob/master/20120064.py
CodeFormat : https://github.com/Viewreful1/20120073/blob/master/20120073.py
Statement : https://github.com/Viewreful1/20120052/blob/master/20120052.py
Function : https://github.com/Viewreful1/20120098/blob/master/20120098.py
Class : https://github.com/Viewreful1/20120067/blob/master/20120067.py
Module : https://github.com/Viewreful1/20120071/blob/master/20120071.py

```

- Issues

- 분석 기준 별로 나온 이슈 개수 제공

Issues

Indentation : 446

Naming : 661

Comment : 150

WhiteSpace : 686

CodeFormat : 114

Statement : 201

Function : 170

Class : 2

Module : 69

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

● Code Issues 확인

- 학생들의 리스트와 각 학생의 코드를 확인하고, 코드에 대한 정보 제공

81	20120018	홍길동 81	1	https://github.com/Viewreful1/20120018/blob/master/20120018.py
82	20120083	홍길동 82	1	https://github.com/Viewreful1/20120083/blob/master/20120083.py
83	20120085	홍길동 83	1	https://github.com/Viewreful1/20120085/blob/master/20120085.py
84	20120086	홍길동 84	1	https://github.com/Viewreful1/20120086/blob/master/20120086.py
85	20120087	홍길동 85	1	https://github.com/Viewreful1/20120087/blob/master/20120087.py
86	20120088	홍길동 86	1	https://github.com/Viewreful1/20120088/blob/master/20120088.py
87	20120068	홍길동 87	1	https://github.com/Viewreful1/20120068/blob/master/20120068.py
88	20120090	홍길동 88	1	https://github.com/Viewreful1/20120090/blob/master/20120090.py
89	20120046	홍길동 89	1	https://github.com/Viewreful1/20120046/blob/master/20120046.py
90	20120092	홍길동 90	1	https://github.com/Viewreful1/20120092/blob/master/20120092.py
91	20120093	홍길동 91	1	https://github.com/Viewreful1/20120093/blob/master/20120093.py
92	20120045	홍길동 92	1	https://github.com/Viewreful1/20120045/blob/master/20120045.py
93	20120094	홍길동 93	1	https://github.com/Viewreful1/20120094/blob/master/20120094.py
94	20120095	홍길동 94	1	https://github.com/Viewreful1/20120095/blob/master/20120095.py
95	20120044	홍길동 95	1	https://github.com/Viewreful1/20120044/blob/master/20120044.py
96	20120096	홍길동 96	1	https://github.com/Viewreful1/20120096/blob/master/20120096.py
97	20120028	홍길동 97	1	https://github.com/Viewreful1/20120028/blob/master/20120028.py

3	20120013	홍길동 3	80	https://github.com/Viewreful1/20120013/blob/master/20120013.py
Static Information		Issues Count		Complexity
Block 길이 : 0 최대 Block 길이 : 0 Block 개수 : 10 글자수 : 797 Class 개수 : 0 주석 개수 : 0 Inline 주석 개수 : 0		Indentation : 25 Naming : 10 Comment : 3 WhiteSpace : 27 CodeFormat : 5 Statement : 8 Function : 2		5
				평균 복잡도 : 6.536082474226804

```

1 def DFS(graph, start):
2     visited=set()
3     stack=[start]
4     while stack:
5         u=stack.pop()
6         if u in visited:continue
7         yield u
8         visited.add(u)
9         for v in reversed(graph[u]):
10             if v in visited:continue
11             stack.append(v)
12 def BFS(graph, start):
13     visited=set()
14     queue=[start]
15     while queue:
16         u=queue.pop()
17         if u in visited:continue
18         yield u
19         visited.add(u)
20         for v in graph[u]:
21             if v in visited:continue
22             queue.insert(0,v)
23 _edges, start=map(int, raw_input().split())
24 graph=dict()
25 for _ in range(edges):
26     u,v=map(int, raw_input().split())
27     graph.setdefault(u, []).append(v)
28     graph.setdefault(v, []).append(u)
29 for u in graph:graph[u].sort()
30 for u in DFS(graph, start):
31     print u,
32 print
33 for u in BFS(graph, start):
34     print u,

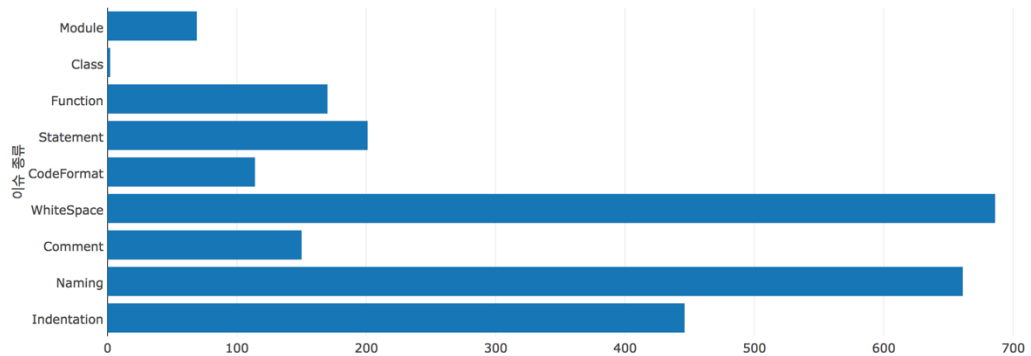
```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codinator)	
	Confidential Restricted	Version 1.3	2017-05-24

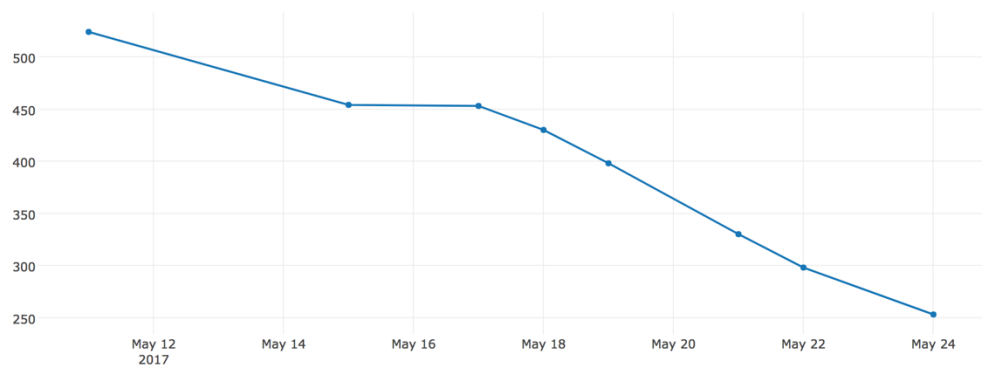
● Graph 확인


- 이슈 종류별 발생 횟수와 학생들의 변화 추이 그래프로 제공

이슈 종류별 발생 횟수



학생들 날짜별 평균 Issue수



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	뷰리풀(Viewreful)	
	팀 명	코드네이터(Codenator)	
	Confidential Restricted	Version 1.3	2017-05-24

5.2 테스트 케이스

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
Github 연동	<i>Data pulling</i>	Github 과 서버를 연동한다	Github 의 웹 후 기능 서버와 연동하고 소스코드를 push 한다.	해당 파일이 서버에 저장된다.	성공
분석 결과 확인	<i>Result check</i>	서버에서 제공되는 분석결과 를 확인한다	제공된 웹 url 로 접속하여 분석 결과가 정상적으로 나오는지 확인한다.	분석 결과가 정상적으로 출력된다.	성공
	<i>Copy Check</i>	서버에서 copy check 결과를 url 로 제공한다..	돋보기 버튼 클릭시 Copy Check 결과 URL 이 정상적으로 제공되는지 확인한다.	Copy check 결과 url 이 정상적으로 출력된다.	성공