**SE 2024 봄학기 텀프로젝트:**

**이슈관리 시스템 개발**

**1. 주요 제출일 및 평가 기준**

* **텀프로젝트 배점: 금번학기 총점 중 30~40%**
* **팀규모:** 한 팀당 **최대** **5명**
* **Final Due: 6월 3일 월요일 오후 9시 (자정이 아닌점에 유의할 것)**
* 최종 제출물 및 제출 방법은 Section 3 참조
* **Presentation: 6월 4일/11일 오후 6시 ~ 9시 수업시간 (팀별 발표 및 Q&A 진행 예정)**
* **평가:** 제출물(발표자료, 프로젝트 문서, 소스코드/테스트코드, 동영상 등) + 발표/Q&A
* **Mid Due: 5월 13일 월요일 오후 9시**
* GitHub 프로젝트 페이지 생성
* e-class의 텀 프로젝트에 팀원 추가:

e-class 왼쪽 패널 매뉴의 과목 선택 -> “사용자 및 그룹” 선택 -> “TermProject Team” 탭 선택 -> 적절한 TermProject 팀을 골라서 멤버 추가:

* e-class의 텀 프로젝트 게시판에 해당 페이지 주소 공지

e-class 왼쪽 패널 매뉴의 과목 선택 -> “사용자 및 그룹” 선택 -> “TermProject Team” 탭 선택 -> 해당 텀프로젝트 번호를 골라서 점 3개 있는 부분 클릭 -> “그룹 홈페이지방문” 클릭 -> “+공지사항” 클릭하여 github 주소 작성

* **수행시 주의 사항:** 본 프로젝트는 필요한 기능을 정확히 구현하는 것 뿐 아니라 좋은 설계, 충실한 문서화 측면에서도 평가가 됨을 주목해야 함. 또한 주어진 요건 외에도 보다 나은 설계(적합한 아키텍처 설계, 확장성과 유지 보수성 좋은 설계, 단위 테스트 용이한 설계 등) 및 그의 구현을 위해 노력한 점이 있다면, 프로젝트 문서를 통해 텍스트, 다이어그램, 코드 등을 이용해 명확히 설명하고, 문서 및 비디오 클립에서 해당 사항을 강조할 것.

**2. 프로젝트 개요**

**2.1 이슈 관리 시스템 개발 프로젝트**

* 이슈 관리 시스템은 소프트웨어 개발 과정에서 조직/단체의 필요에 의해 이슈 목록과 내용 등을 체계적으로 관리하는 소프트웨어의 일종으로 트러블 티켓 시스템(trouble ticket system)이라고도 한함
* JIRA, TRAC, Bugzilla, Backlog 등 다양한 이슈 관리 시스템이 이미 존재함
* 본 과제에서는 이슈를 등록하고 관리하는 이슈 관리 시스템의 전형적인 기능을 포함함
* 객체지향 분석과 설계 개념을 적극적으로 활용하며, 어떤 부분에 왜 그러한 설계를 하였는지를 문서화해야 함
* 기능 확장과 설계 개선 활동이 잘 되었다는 것을 JUnit을 이용해 테스트케이스를 만들고 적절히 문서화해야 함
* 시스템에 저장된 데이터는 영속하는(persistent) 저장소에 보관되어, 시스템을 재시작하여도 기존 데이터를 재사용할 수 있어야 함. 영속하는 저장소로 반드시 DBMS를 사용할 필요는 없으며 File System을 활용할 수도 있음.
* 기본적 요건은 GUI(Graphical User Interface) 혹은 Web 인터페이스를 통해 SW가 동작하도록 만드는 것임
* 설계/구현 시 UI와 응용 로직(Logic)을 분리하여 추후 UI가 바뀌더라도 로직 이하의 계층은 변경없이 재사용할 수 있도록 설계 및 구현해야 함. 이와 같은 설계 내용이 프로젝트 문서에도 명확히 나타나야 함
* 다중의 인터페이스 (예를 들어 Swing과 Web 인터페이스) 방식을 구현하였으며, 모델이 잘 분리되어 다중의 인터페이스를 지원하기 위해 적절한 설계가 되어 있다고, 문서상으로 충분히 설명되었고, 잘 동작한다는 것을 보이면 평가시 가산점을 부여함 (설계만 하는 것이 아니라 실제로 2개 이상의 인터페이스를 구현하고 시연해야 하며, 설계 면에서 인터페이스를 교교 했을 때, 인터페이스와 직접적으로 관련 없는 모듈은 수정없이 공유되는 것을 잘 설명해야 함)
* **테이블이(가) 표시된 사진

  자동 생성된 설명**기존에 해결된 이슈 데이터 이력을 활용하여, 새로운 이슈가 등록되었을 때 적절한 담당자를 자동적으로 추천하는 확장된 기능을 추가하면 평가시 가산점을 부여함

**2.1 MUST 포함 기능**

하기에 충분히 설명되지 않은 조건 및 세부 명세 등에 대해서는 각자 적절한 가정을 세우고 프로젝트 문서에서 해당 가정을 명시할 것

* 계정 추가
* 이슈 브라우즈 및 검색 (예를 들어 assignee 값, 이슈 상태, reporter 값 등)
* 이슈 등록 (꼭 채워야하는 필드는 Title, Description. reporter 필드는 이슈를 등록한 계정으로, reported date는 등록한 날짜와 시간으로 자동으로 채워짐)
* 이슈 코멘트 추가
* 이슈 상세 정보 확인 (이슈의각 필드와 코멘트 history 확인 가능)
* 이슈 배정을 포함한 이슈 상태 변경
* 이슈 통계 분석(일/월 별 이슈 발생 횟수 및 트랜드 등 표시)
* ~~해결된(closed/resolved) 이슈 이력을 활용한 assignee 자동 추천 기능~~ (extra 점수 부여)
* 상기 언급된 기능이 아니지만, Trac이나 Bugzilla와 같은 실제 이슈 관리 시스템의 기능을 참고하여 해당 시스템과 유사하게 만들수록 완성도 면에서는 더 우수한 점수를 받을 수 있음. 단, 이러한 추가 기능이 있다면, 해당 부분에 대한 언급을 보고서에서 명시하고, 발표에서도 강조하기 바람.

**2.2 예제 시나리오**

구현 편의성을 위해 모든 프로젝트를 Admin 하나의 계정으로 공유하여 시스템 관리자 역할을 한다고 가정하기 바랍니다. 그리고 admin 계정은 시스템 최초 run 전에 이미 생성되어 있다고 가정하면 됩니다. admin은 소스 코드상에 미리 정해진 admin 패스워드를 사용합니다.

그리고 PL, dev, tester 등 계정 종류별로 역할을 달리하는 것도 가능하고, 굳이 구분하지 않고 누구나 이슈를 생성하고, 검색하고, 배정하고, 코멘트를 달 수 있게 만들어도 됩니다. 다만 역할 구분을 포함하여, 모든 기능 측면에서 Trac이나 Bugzilla와 같은 실제 시스템과 유사하게 만들수록 완성도 면에서는 더 우수한 점수를 받을 수 있습니다

(데모를 위해 최소 admin, PL1, PL2, dev1~10, tester1~5 계정을 만들어 둘 것)

다음과 같은 시나리오가 동작할 수 있어야 함.

Admin이 project1을 추가함. 또한 PL계정, 개발자 계정, 테스터 계정을 추가함.

* 다음은 모두 project1에 대한 활동임.
* tester1이 이슈를 하나 생성하고, 해당 이슈의 코멘트를 하나 추가함. 이후 시스템은 해당 이슈의 상태를 new로 변경하고, reporter의 값은 tester1으로 저장함.
* PL1은 모든 이슈들을 브라우즈 함. 이후 new 상태의 이슈들을 검색하고, 위에서 tester1이 reporter로 되어 있는 이슈에 대해 dev1을 담당자(assignee)로 지정하며, 코멘트에 적절한 메시지를 추가함. 이후 시스템은 해당 이슈의 상태를 assigned로 변경함
* dev1은 이슈 검색 기능으로 자신에게 assign 되어 있는 이슈들을 브라우즈 함. 이슈들에 대해 남겨진 코멘트를 포함하여 다양한 정보를 확인하고, 코드를 고친 후 (ITS에는 없는 기능임. 타 시스템을 이용해서 한다고 가정), 코멘트를 추가한 후 해당 이슈를 fixed로 바꿈. 이렇게 되면 해당 이슈의 fixer는 dev1으로 등록됨.
* tester1은 자신이 report하였던 이슈 중에 fixed된 이슈들을 검색하고, 적절히 고쳐진 이슈들을 선택하여 “resolved” 상태로 바꿈.
* PL1은 resolved 상태인 이슈들을 선택해서 closed로 바꿈.
* optional 기능 (추가 배점): PL2는 현재 new 상태인 이슈들을 브라우즈하고, 상세 정보 보기 기능을 통해 특정 이슈의 상세한 내용과 기존 코멘트들을 살펴봄. 이때 시스템은 fixed된 이슈들의 이력을 이용해서 가정 적절한 개발자를 추천해줌. (예를 들어 “best candidate: dev2, dev5, dev1 등으로”, 가장 가능성이 높은 후보 3명을 순서대로 추천함)

**2.3 이슈 필드 및 optional 기능 관련**

* 이슈 필드 (최소이며, 필요에 따라 필드를 추가할 수 있고, 프로젝트 문서에서 설명할 것)
* Issue Title
* Issue Description
* Reporter
* Reported Date
* Fixer
* Assignee
* 우선순위: blocker/critical/major/minor/trivial (기본값은 major)
* 이슈상태: new/assigned/resolved/closed/reopened
* Comments (코멘트는 추가되는 순서대로 누적됨. 코멘트 날짜 포함)
* 해결된 이슈들을 활용하여 적절한 assignee를 추천하는 방법은 팀별로 자유롭게 제안할 수 있음. 적절한 주기에 따라 데이터를 한꺼번에 모아서 기계학습 등을 통해 학습모델을 구축하여 추천할 수도 있고, Information Retrieval 방법 등을 이용해서 가장 유사한 내용의 기존 해결된(resolved/closed) 이슈의 fixer를 추천할 수도 있음. 단, 적절한 주기에 따라 동작하는 시스템의 경우, 평가 데모를 위해 수동으로 특정 시점에 학습할 수 있는 기능 (예를 들어 “learn now”) 등을 추가할 것.

**3. 최종 제출물 및 제출 방법**

**3.1 제출 방법**

* 제출방법: eclass 과제 및 평가 탭의 과제 “TermProject”에 업로드 (링크 생성 예정)
* 소프트웨어 구현 언어: Java (서버 측만 Java 언어 위주로 개발하면 됨. 인터페이스 및 부분은 javascript, HTML/CSS, 혹은 다른 언어로 개발해도 무방함)
* Github 이용: GitHub에 프로젝트 페이지를 만들고 GitHub에서 프로젝트를 수행해야 함. 개발 과정이 commit 히스토리 등을 통해 잘 나타나도록 GitHub을 이용해서 과제를 수행해야 함

**3.2 제출물**

* 제출물 포맷: “팀번호\_팀원이름1\_팀원이름2\_팀원이름3\_팀원이름4\_팀원이름5.zip” 형식의 화일명을 갖는 zip 화일 1개를 제출하며, 해당 zip 화일은 다음의 산출물을 포함해야 함
* README.txt 파일
* 제출된 모든 산출물에 대한 목록 및 요약 및 github 주소
  + 소스코드 및 실행화일
  + 프로그램 사용 방법
* 발표 슬라이드 (최종 발표용 슬라이드)
* 프로젝트 문서 (하기 설명 참조)
* 모든 소스코드 화일, 실행 화일, JUnit 테스트 코드, 데이터 등
* 프로젝트 문서(PDF 형식 파일로 제출, 문서에 반드시 포함되어야 하는 내용은 하기 설명을 참고하고, 추가로 필요하다고 생각되는 내용을 자유롭게 포함시키면 됨)
* 표지에 팀원들의 학번, 이름을 기재
* 프로젝트 내용 요약. 특히 MUST 요건 중 충족하지 못한 것이 있다면 이 부분에서 해당 내용에 대해 명시해야 함.
* 요구 정의 및 분석
  + 유스케이스 다이어그램 (전체 유스케이스 모두, 단 includes 관계 1개 이상 포함, extends 관계 1개 이상 포함)
  + 유스케이스 명세 (10개 이상의 유스케이스에 대해)
  + 도메인 모델 (Domain Model)
  + SSD (System Sequence Diagram), Operation Contract 등을 이용하여 요구 정의 및 분석 결과를 표현할 것. 텍스트 형태의 설명도 포함해야 함
* 설계
  + 클래스 다이어그램과 시퀀스 다이어그램은 해당 그림에 나타난 주요 구성요소들 예를 들어 클래스, 오브젝트, 메시지 등에 대한 설명이 포함되어야 함
  + OOAD/GRASP 패턴/설계 원칙의 적용 부분 및 이유에 대해 설명
  + 이외에도 적절한 UML 다이어그램들을 이용하여 다양한 설계 결과를 표현할 것
* 구현
  + 주요 기능들을 스크린 샷 등을 통해 표현
  + 위의 내용을 설명하기 위해 다양한 UML 다이어그램의 활용이 추천됨 (클래스 다이어그램, 시퀀스 다이어그램 등)
* 테스트 수행 내역
  + 각 테스트 케이스에 대해 테스트 케이스를 만든 목적 등에 대한 설명이 포함되어야 함. 기존 기능들이 잘 동작하는 것도 보여야 하며, 특히 새롭게 추가한 기능의 경우 각 추가 기능들이 잘 동작한다는 것을 효과적으로 보여주어야 함.
  + JUnit을 활용했다면 테스트 코드도 함께 제시
  + 테스트 케이스 적용 결과 (스크린 샷)
* GitHub 프로젝트 활용 요약
  + 팀의 github 프로젝트 주소
  + 프로젝트 progress history 스크린샷
  + 팀원별 기여를 잘 나타낼 수 있는 각종 자료
* 비디오 클립: 프로젝트 문서의 내용 바탕으로 설계 및 구현, 그리고 테스트 활동에 대한 설명을 해야함. 더불어 주요 기능들이 잘 작동함을 보여줄 것. 특히 팀이 프로젝트 수행에서 가졌던 설계 고민과 결과들이 잘 드러나도록 해야함. 또한, 테스트 케이스를 만든 목적 등에 대한 설명이 포함되어야 함. (30분 내의 분량)

**4. 이슈 관리 시스템 관련 자료**

**4.1 Trac**

<https://www.youtube.com/watch?v=b9O8RhdIukI>

<https://www.youtube.com/watch?v=11RymVx74QM>

<https://www.youtube.com/watch?v=nUAKXtLlTM0>

<https://www.youtube.com/watch?v=B8xHRzg7aZA>

**4.2 Bugzilla**

<https://www.youtube.com/watch?v=zXeiGetQmHg>

<https://www.youtube.com/watch?v=QmA3hNVB6Vw>

<https://www.youtube.com/watch?v=XKZs_RkeBJQ>

<https://www.youtube.com/watch?v=gnG58qEzg7k>

<https://www.youtube.com/watch?v=jK2pQQsDKds>

**4.3 기타 SE 도구 정보**

[https://www.swbank.kr/helper/tool/toolMain.do#](https://www.swbank.kr/helper/tool/toolMain.do)