|  |  |  |
| --- | --- | --- |
| 담당 교수님 |  | 최윤자 교수님 |
| 팀장 |  | 유동규 2011105062 |
| 팀원 |  | 박동원 2011105038 |
| 팀원 |  | 윤준호 2011105065 |
| 팀원 |  | 이승진 2011105072 |

**INDEX**

1. MRD (Market Requirements Definitions)

2. SRS (Software Requirements Specification)

3. Test Case

4. Context Diagram

5. Architecture, MVC

**1. MRD (Market Requirements Definitions)**

|  |  |  |
| --- | --- | --- |
| Functional |  | * 교수, 조교, 학생이 시스템을 사용한다. (윤준호) * 교수, 조교는 과제를 등록한다. (유동규) * 학생은 교수, 조교가 올린 과제를 확인하고, 제출한다. (유동규) * 학생이 과제를 올리면 교수, 조교에게 알림이 간다. (박동원) * 학생이 과제를 제출하면 제출 되었다고 알림이 뜬다. (박동원) * 교수, 조교는 학생이 제출한 과제를 확인하고 평가 및 코멘트를 단다. (유동규) * 교수, 조교가 과제를 평가하면 학생에게 알림이 간다. (박동원) * 학생은 교수, 조교에게 질문이 가능하다. (이승진) * 교수, 조교는 학생의 질문에 답변이 가능하다. (이승진) |
| Non -Functional |  | * 학생은 자기 자신의 과제 및 채점 결과만 볼 수 있다. (윤준호) |

**2. SRS (Software Requirements Specification)**

**Functional**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 과제 제출 |  | * 각 강의마다 과제 게시판이 따로 있다. (박동원)   2. 교수와 조교는 자신이 담당 교수나 조교인 강의 게시판에 과제를 올리고 학생들의 과제를 열람 가능하다. (유동규)  2.1. 교수, 조교는 게시판에 자신이 지금 강의중인 과목의 과제게시판만 들어 갈 수 있다. (이승진)  3. 학생들은 수강 신청한 과목의 과제를 해당 과목 게시판에 올릴 수 있고, 게시자가 자신의 이름으로 된 자신의 과제만 열람과 수정이 가능하다. (유동규)  3.1. 학생들은 게시판에 자신이 수강중인 과목의 과제게시판만 들어 갈 수 있다. (박동원)  3.2. 학생들이 과제를 제출한 시간이 포함된 알림이 학생이 제출하자마자 바로 전송된다. (윤준호)  3.3. 학생들이 과제를 올리면 교수와 조교에게 실시간 알림 창이 뜬다. (박동원)  3.3.1. 만약 조교 및 교수가 시스템에 접속하고 있지 않으면 알림이 가지 않는다. (유동규)  3.3.2. 후에 시스템에 접속할 시 밀려있던 알림이 한번에 전송된다. (유동규) |
| 과제 채점 |  | 1. 교수와 조교는 학생들의 과제를 채점하고 코멘트를 단 다음 학생의 과제에 답글 형태로 올린다. (유동규)  1.1. 채점 된 과제들의 점수 및 코멘트는 모두 다 서버에 저장된다. (이승진)  1.2. 채점 결과가 수치 화 가능하면 평균점수를 낸다. (윤준호)  1.2.1. 교수 및 조교에게는 각 점수대 별 인원과 과제의 평균점수를 보여준다. (윤준호)  1.3.2. 학생에게는 평균점수 및 자기점수를 보여준다. (이승진)  1.3.3. 만약 수치화 불가능 할 시 과제 제출유무를 막대그래프로 표시해준다. (유동규)  1.3. 학생에게 실시간 알림 창이 뜬다. (박동원)  1.3.1. 만약 해당하는 학생이 시스템에 접속해 있지 않으면 알림이 가지 않는다. (유동규)  1.3.2. 후에 시스템에 학생이 접속할 시 밀려있던 알림이 한번에 전송된다. (이승진) |
| 질의 응답 |  | 1. 학생들은 교수나 조교에게 질문 글을 올릴 수 있으며, 질문을 비공개나 공개 글로 설정 가능하다. (이승진)  1.1. 공개 / 비공개 설정을 안하고 업로드하면 자동으로 공개 처리된다. (윤준호)  2. 교수나 조교는 학생들의 질문에 답글 형태로 의견을 달 수 있다. (이승진) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 보안 |  | | * 학생은 자신의 게시물만 볼 수 있다. (이승진) * 학생에게 통계를 보여줄 때는 자신의 점수와 해당 과목의 평균점수 만을 보여준다. (유동규) | |
| 운영 |  | | 1. 한 학기가 끝나면 한 학기간 저장된 과제와 코멘트를 지운다. (박동원)  2. 사용자 요구 게시판을 만들어 주말과 공휴일을 제외한 날의 13시, 17시에 모니터링 한다. (윤준호)  3. 현재 시스템이 돌아가기 어려울 정도의 오류가 있을 시 즉시 업데이트 하며, 그 이외에는 일주일에 한 번 주기적으로 업데이트 한다. (유동규) | |
| 퍼포먼스 | |  | | 1. 모든 알림은 최소 10초 안에 교수와 조교, 학생에게 알림이 간다. (박동원) |
| 문화 | |  | | 1. 외국인 학생을 배려해 영어로도 시스템을 제공한다. (박동원) |

**Non Functional**

**3. TEST CASE**

**Functional**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 과제 제출 |  | 1. 각 강의마다 과제 게시판이 따로 있고 자신이 강의 중인 과제 게시판에만 접속이 가능하다. (박동원)  **Input**  - A교수의 계정으로 로그인  - A교수가 이번학기 강의중인 AAA강의 과제게시판을 클릭  **Test procedure**  - 서버에서 AAA강의가 A교수가 강의중인 강의임을 확인하고 AAA강의의 과제게시판을 불러옴.  **Expected output**  - AAA강의 과제게시판에 접속  - 학생들이 올린 과제가 있을 시 과제목록이 출력  - 학생들이 올린 과제가 없을 시 빈 화면이 출력  2. 교수와 조교는 자신이 담당 교수나 조교인 강의 게시판에 과제를 올리고 학생들의 과제를 열람 가능하다. (유동규)  2-1. 교수와 조교는 자신이 담당하는 강의의 과제게시판에 과제를 올린다.  **Input**  - 과제제목 : MRD SRS 제출  과제내용 : 4월 16일까지 MRD, SRS를 작성하여 문서화한 후 GIT에 올릴것  **Test procedure**  - 입력된 내용과 첨부파일을 서버에 저장함  **Expected output**  - AAA강의 과제게시판에서 'MRD SRS 제출' 이란 제목의 글이 올라옴  2-2. 교수나 조교가 학생들의 과제를 열람한다.  **Input**  - 현재 AAA강의 게시판에서 학생이 올린 과제 중 '2015111111 B학생 MRD SRS 제출' 이란 글을 클릭함  **Test procedure**  - 서버에서 '2015111111 B학생 MRD SRS 제출' 이란 제목을 가진 글을 찾아옴  **Expected output**  - 과제제목 : 2015111111 B학생 MRD SRD 제출  과제내용 : 1. MRD는 Market Requirement Definition  2. SRS는 Software Requirement Specification이다.  이런 내용의 게시글이 출력  3. 교수, 조교가 게시판에 자신이 지금 강의중인 과목이 아닌 과제 게시판을 클릭하면 접속이 불가능하다. (이승진)  **Input**  - A교수의 계정으로 자신이 강의하는 과목이 아닌 CCC강의 과제게시판을 클릭  **Test procedure**  - 서버에서 CCC강의가 A교수가 강의하는 강의가 아님을 알림  **Expected output**  - 'CCC강의는 A교수가 이번학기에 강의하는 강의가 아닙니다.' 라는 경고문구 출력  - 과제게시판 선택 화면 출력  3. 학생들은 수강 신청한 과목의 과제를 해당 과목 게시판에 올릴 수 있고, 게시자가 자신의 이름으로 된 자신의 과제만 열람과 수정이 가능하다. (유동규)  3.1. 학생들은 게시판에 자신이 수강중이지 않은 강의의 과제게시판을 선택하면 접속이 불가능하다. (박동원)  **Input**  - B학생이 현재 자신이 수강하고 있지 않는 CCC강의 과제게시판을 클릭  **Test procedure**  - 서버에서 CCC강의가 B학생이 수강중인 강의가 아님을 알림  **Expected output**  - 'CCC강의는 B학생이 이번학기에 수강중인 강의가 아닙니다.' 라는 경고문구 출력  - 과제게시판 선택 화면 출력  3.2. 학생들은 게시판에 자신이 수강중인 강의의 과제게시판을 선택하면 접속이 가능하다. (박동원)  **Input**  - B학생이 현재 자신이 수강중인 AAA강의 과제게시판을 클릭  **Test procedure**  - 서버에서 AAA강의가 B학생이 수강중인 강의임을 확인하고 AAA강의의 과제게시판을 불러옴.  **Expected output**  - AAA강의 과제게시판에 접속  - 이전에 올라온 과제가 있을 시 과제목록이 출력  - 이전에 올라온 과제가 없을 시 빈 화면이 출력  3.3. 학생들이 과제를 제출한 시간이 포함된 알림이 학생이 제출하자마자 바로 전송된다. (윤준호)  **Input**  - B학생이 AAA강의 과제게시판에 'MRD SRS 제출' 이란 과제에 9:00 PM에 과제를 제출함.  - 과제제목 : 2015111111 B학생 MRD SRD 제출  과제내용 : 1. MRD는 Market Requirement Definition  2. SRS는 Software Requirement Specification이다.  과제 제출 시각 : 9:00 PM  **Test procedure**  - 서버에서 제출된 과제를 저장하고 과제 제출 시각을 표현한 알림문구를 A교수의 알림시스템으로 보냄.  **Expected output**  - 1. A교수가 현재 시스템에 접속중일 때  알림제목: B학생 9:00 PM 과제 제출  - 2. A교수가 현재 시스템이 접속중이지 않을 때  서버에 계속해서 저장  3.4. 알림이 쌓여있는 상태에서 교수나 조교가 접속을 함. (유동규)  **Input**  - 현재 D교수에게 가야 할 알림이 총 2가지가 있는 상태에서 D교수가 시스템에 접속함  - 알림1 : E학생 10:00 PM 과제 제출  - 알림2 : F학생 10:30 PM 과제 제출  **Test procedure**  - 서버에서 D교수가 시스템에 접속했음을 인지함.  - 현제 저장되어 있는 알림을 먼저 온 순서대로 D교수의 알림시스템으로 보냄  **Expected output**  - 알림제목 : E학생 10:00 PM 과제 제출  - 먼저 들어온 알림1 출력 후  알림제목 : F학생 10:30 PM 과제 제출 출력 |
| 과제 채점 |  | 1. 교수와 조교는 학생들의 과제를 채점하고 코멘트를 단 다음 학생의 과제에 답글 형태로 올린다. (유동규)  1.1. 채점 된 과제들의 점수 및 코멘트는 모두 다 서버에 저장된다. (이승진)  **Input**  - 교수나 조교가 코멘트를 작성하고 채점결과를 입력한 후 업로드 버튼을 누른다.  **Test procedure**  - 업로드된 정보들이 서버에 저장된다.  **Expected output**  - X  1.2. 채점 결과가 수치 화 가능하면 평균점수를 낸다. (윤준호)  1.2.1. 교수 및 조교에게는 각 점수대 별 인원과 과제의 평균점수를 보여준다. (윤준호)  **Input**  - 교수나 조교가 과제게시판에서 점수확인 버튼을 누른다.  **Test procedure**  - 해당 과제의 수치화된 정보를 서버에서 불러온다.  **Expected output**  - 불러온 수치화된 정보를 출력한다.  1.2.2. 학생에게는 평균점수 및 자기점수를 보여준다. (이승진)  **Input**  - 학생이 과제게시판에서 점수확인 버튼을 누른다.  **Test procedure**  - 해당 과제의 수치화된 정보를 서버에서 불러온다.  **Expected output**  - 불러온 수치 중 평균점수와 자기점수를 출력한다.  1.2.3. 만약 수치화 불가능 할 시 과제 제출유무를 막대그래프로 표시해준다. (유동규)  **Input**  - 교수나 조교가 과제게시판에서 점수확인 버튼을 누른다.  **Test procedure**  - 과제 제출유무 정보를 서버에서 불러온다.  **Expected output**  - 과제 제출유무를 막대그래프로 만들어 화면에 출력한다.  1.3. 학생에게 실시간 알림 창이 뜬다. (박동원)  1.3.1. 만약 해당하는 학생이 시스템에 접속해 있지 않으면 알림이 가지 않는다. (유동규)  **Input**  - 학생이 접속 상태가 아니다.  **Test procedure**  - 학생 접속 상태를 서버에서 확인한다.  **Expected output**  - X  1.3.2. 후에 시스템에 학생이 접속할 시 밀려있던 알림이 한번에 전송된다. (이승진)  **Input**  - 알림창이 밀려있는 상태에서 학생이 시스템에 접속을 한다.  **Test procedure**  - 서버에서 접속신호를 받고 저장되어 있던 알림들을 학생에게 전송한다.  **Expected output**  - 학생의 알림창에 알림들이 뜬다. |
| 질의 응답 |  | 1. 학생들은 교수나 조교에게 질문 글을 올릴 수 있으며, 질문을 비공개나 공개 글로 설정 가능하다. (이승진)  1.1. 공개 / 비공개 설정을 안하고 업로드하면 자동으로 공개 처리된다. (윤준호)  **Input**  - 학생이 질문을 올린다.  **Test procedure**  - 질문의 공개설정을 서버에서 확인하고 업로드한다.  **Expected output**  -질문게시판에 질문이 올라간다.  2. 교수나 조교는 학생들의 질문에 답글 형태로 의견을 달 수 있다. (이승진)  **Input**  - 교수나 조교가 답글을 입력한 후 등록한다.  **Test procedure**  - 답글이 서버에 저장된다.  **Expected output**  - 게시글에 답글이 올라간다. |

**Non Functional**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 보안 |  | | 1. 학생은 자신의 게시물만 볼 수 있다. (이승진)  **Input**  - 학생이 게시물을 클릭하여 접근한다.  **Test procedure**  - 게시자와 게시물 접근자를 비교한다.  **Expected output**  - 게시자와 접근자가 같으면 게시글을 읽고, 아니라면 접근이 거부된다.  2. 학생에게 통계를 보여줄 때는 자신의 점수와 해당 과목의 평균점수 만을 보여준다. (유동규)  **Input**  - 학생이 과제게시판에서 점수확인 버튼을 누른다.  **Test procedure**  - 해당 과제에서 본인의 점수 및 학생들의 평균 점수를 서버에서 불러온다.  **Expected output**  - 불러온 점수를 보여준다. | |
| 운영 |  | | 1. 한 학기가 끝나면 한 학기간 저장된 과제와 코멘트를 지운다. (박동원)  **Input**  - 교수나 조교가 학기 종료 버튼을 누른다.  **Test procedure**  - 버튼이 눌러진 시점에서 입력된 날짜와 시스템에 지정해놓은 학기 종료 날짜를 비교한다.  **Expected output**  - 현재 날짜가 정해진 학기 종료 날짜를 지났으면 서버에서 데이터를 모두 삭제한다. 그렇지 않으면 그대로 둔다 | |
| 퍼포먼스 | |  | | 1. 모든 알림은 최소 10초 안에 교수와 조교, 학생에게 알림이 간다. (박동원)  **Input**  - 알림이 올라올 상태를 만든다. (과제 제출, 과제 채점)  **Test procedure**  - 접속자에게 알림을 보낸다.  **Expected output**  - 10초 이내에 알림이 뜬다. |
| 문화 | |  | | 1. 외국인 학생을 배려해 영어로도 시스템을 제공한다. (박동원)  **Input**  - “English” 버튼을 클릭한다.  **Test procedure**  - 번역된 모든 텍스트 데이터를 서버에서 불러온다.  **Expected output**  - 영어로 된 시스템을 새로이 시작한다. |

**4. Context Diagram**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MVC 선택 |  | | * UI 와 Application Logic 이 분리되어 서로의 수정이 용이함. * MVC 의 경우 시스템이 비동기적 인데, 우리 프로그램에서는 동기식일 필요가 없으므로. * 현재 프로그램 자체도 UI 와 Logic 이 분리되어 있음. |
| Client Server |  | * 데이터의 분산이 굳이 필요하지 않음. | | | |
| Layered |  | * 우리 프로그램에서는 Layer를 나눌만한 요소가 적음. | | |
| Repository |  | * 우리 프로그램은 단일 서버로 구축 예정이고 실제로 Sub-System 들 간의 데이터 공유가 없다고 봐도 되기 때문. | | |
|

**4. Architecture, MVC(Model View Controller)**