|  |
| --- |
| 프로젝트 계획서 |

[치과 영상 SW 시스템 개발]

2022년 12월 26일

**목 차**

**1. 개요………………………………………………………………………………………………...2**

1.1. 프로젝트 명…………………………………………………………………………………………………………...…2

1.2. 프로젝트 기간………………………………………………………………………………………………..……...…2

1.3. 프로젝트 목적……………………………………………………………………………………………..………...…2

1.4. 프로젝트 기대효과………………………………………………………………………………………………...…2

**2. 프로젝트 범위…………………………………………………………………………………...3**

2.1. 작업명세서………………………………………………………………………………………………………….....…3

**3. 프로젝트 추진체계…………...…………....…………..…………....................................6**

3.1. 프로젝트 수행조직도…………………………………………………………………….....………………………6

3.2. 역할분담표……………………………...………………………………………………………………….……….....…6

**4. 시스템 구축 환경**

4.1. 시스템 아키텍처 구성도

4.2. UI 구성 예시

4.3. 데이터베이스 구성

**5. 프로젝트 관리**

5.1. 프로젝트 작업 계획

5.2. 코딩 가이드

5.3. 형상 관리 방안

5.4. 테스트 방안

5.5. 프로젝트 보고 계획

**1. 개요**

**1.1. 프로젝트 명**

본 프로젝트의 명칭은 “치과 영상 SW 시스템 개발”이라 한다.

**1.2. 프로젝트 기간**

- 프로젝트 기간: 2023년 01월 04일 ~ 2023년 02월 27일 (37일, 296시간)

- 프로젝트 발표: 2023년 02월 28일

**1.3. 프로젝트 목적**

치과산업 전반에 걸쳐 디지털 덴티스트리 환경을 구축하는 것이 중요해지면서, 진료 프로세스의 효율성을 위해 종합 솔루션 개발의 필요성이 대두되고 있다. 본 프로젝트는 진료과정 전체에 대한 토탈 솔루션을 개발함으로써, 의료진과 환자에게 종합적이고 체계적인 All-In-One 서비스를 제공하는 것을 그 목적으로 한다.

**1.4. 프로젝트 기대효과**

사용자 중심의 환자관리 시스템 개발을 통해 환자의 관리를 보다 편리하게 하고, 치과 영상 촬영부터 치료까지 손쉽게 계획하고 관리할 수 있는 서비스를 제공함으로써 다음과 같은 효과를 극대화한다.

* 키보드 사용이 어려운 진료환경에 맞춘 UI 개발로 의료진에게 높은 사용성 제공
* 환자정보와 영상 데이터를 통합된 환경에서 관리하여 진료의 신속성 제공
* DICOM 표준 기반의 시스템 개발로 타 의료기관과의 연동성 제공
* 영상처리 기술을 적용하여 진단과 치료의 정확성 제고
* 검사실 내 Small PACS를 구축하여 Main Server의 부하를 줄이고 시스템 안정성 확보

**2. 프로젝트 범위**

**2.1. 작업명세서**

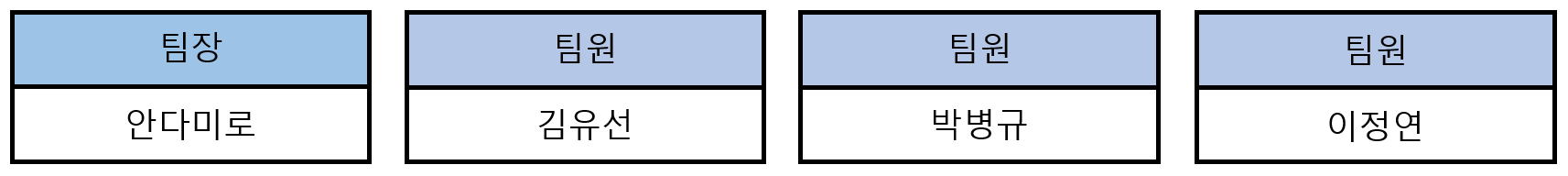
2.1.1. 치과 영상 SW 시스템 모듈 개발

|  |  |  |
| --- | --- | --- |
| **구분** | **업무 범위** | **업무 상세** |
| **환자관리 SW** | 환자정보 확인 | ∙ DB에서 환자의 정보 로드 및 UI에 출력 |
| 환자 특이사항 메모 | ∙ 위젯을 통해 메모를 작성하고 저장할 수 있도록 구현 |
| 진료 기록 확인 | ∙ 정보(날짜 및 담당의 등)와 진료 기록 매핑  ∙ 클릭을 통해 진료 기록 및 시간 확인 |
| 진료 예약 | ∙ 편리한 날짜선택을 위해 캘린더 형식으로 구성  ∙ 진료예약 추가, 수정, 삭제 |
| 환자 검색 | ∙ DB에서 환자 정보를 통해 검색이 가능하도록 구성 |
| 환자정보 추가, 수정, 삭제 | ∙ 환자 추가 시 즉시 DB에 저장되는 구조 설계  ∙ 수정 가능한 정보 : 이름, 사진, 주소, 이메일, 나이, 환자 특이사항  ∙ 수정이 불가능한 정보 : 주민번호, 성별, 진료기록 |
| 환자 영상 데이터 출력 | ∙ 최근순으로 미리보기에 출력하고 하단에 촬영날짜 명시 |
| **촬영 SW** | 환자정보 연동 | ∙ 영상촬영 전, 피검사자의 정보를 화면에 출력 |
| 장비 제어 명령 | ∙ 장비 초기화, 촬영 준비, 촬영 시작, 촬영 종료 명령 |
| 영상 재구성 | ∙ Raw Frame 데이터를 파노라마 영상으로 재구성  ∙ 스티칭 및 병렬 처리 프로그래밍 기법 적용 |
| **CT Simulation SW** | 촬영 신호 전송 | ∙ 촬영 및 Raw Frame 데이터 전송 |
| 장비 제어 명령 | ∙ 동작 시작 및 종료 신호를 받아 기기 시스템 작동  ∙ 촬영데이터만 서버로 전송, 동작 데이터는 종료 시 삭제 |
| 촬영 제어 명령 | ∙ 촬영 시작 및 종료 신호를 받아 촬영 기능 작동  ∙ 촬영 및 장비 일시 정지 |
| 장비 재시작 및 초기화 | ∙ 초기화 신호를 받아 장비 시스템 리셋 |
| **영상 뷰어**  **SW** | 줌 인 / 줌 아웃 | ∙ 마우스를 통해 이미지를 확대, 축소 |
| 이미지 밝기 조절 | ∙ 이미지의 밝기를 조절 |
| 브러시를 이용한 그리기 | ∙ 펜을 활용하여 이미지에 원형, 사각형 등의 도형을 그리거나 자유롭게 작성할 수 있도록 구현 |
| 이미지 회전, 반전 | ∙ 이미지의 회전, 상하좌우 반전을 통해 환자에게 해당 치아를 더 잘 보여줄 수 있도록 설계 |
| 이미지 자르기 | ∙ 마우스를 활용해 특정 관심영역 캡쳐 |
| 이미지 불러오기 | ∙ 과거 진료내역(사진) 출력을 통해 수술 전후의 결과 비교 |
| 이미지 노이즈 제거 | ∙ 노이즈 제거를 제거하여 선명한 영상 데이터 획득 |
| 이미지 보정(샤프닝, 감마) | ∙ 영상의 선명도를 높이고, 최적의 밝기로 보정  ∙ 픽셀단위의 영상처리 기법 적용 |

2.1.2. 시스템 구축

|  |  |
| --- | --- |
| **업무** | **업무 범위** |
| **시스템 구축** |  |
| **구현** |  |
| **테스트 및 안정화** | □ 최종 테스트 및 배포  - 단위 테스트  - 통합 테스트  - 기능 테스트  - 배포 |

**3. 프로젝트 추진체계**

**3.1. 프로젝트 수행조직도**

**3.2. 역할분담표**

|  |  |
| --- | --- |
| **구분** | **역할 및 책임** |
| **안다미로(팀장)** | ∙ 프로젝트 개발일정 및 이슈 관리  ∙ 촬영 SW 개발  ∙ 영상 재구성 알고리즘 개발  ∙ 네트워크 개발: 서버와 촬영 SW 연동 |
| **김유선** | ∙ 회의록 등 문서산출물 관리  ∙ 환자관리 SW 개발  ∙ 메인 서버 및 프로토콜 개발  ∙ DB 구축 및 관리  ∙ 네트워크 개발: 서버와 환자관리 SW 연동 |
| **박병규** | ∙ CT Simulation SW 개발(3D)  ∙ 테스트 프로세스 관리  ∙ 산출물 및 최적화 검증 프로세스 설계  ∙ 네트워크 개발: 서버와 CT Simulation SW 연동 |
| **이정연** | ∙ 자료 저장 및 공유 페이지(Notion) 관리  ∙ 영상 뷰어 SW 개발  ∙ 사용자 친화적 통합 UI/UX 개발  ∙ 2D 영상처리 알고리즘 개발  ∙ 네트워크 개발: 서버와 뷰어 SW 연동 |

**4. 시스템 구축 환경**

**4.1. 시스템 아키텍처 구성도**

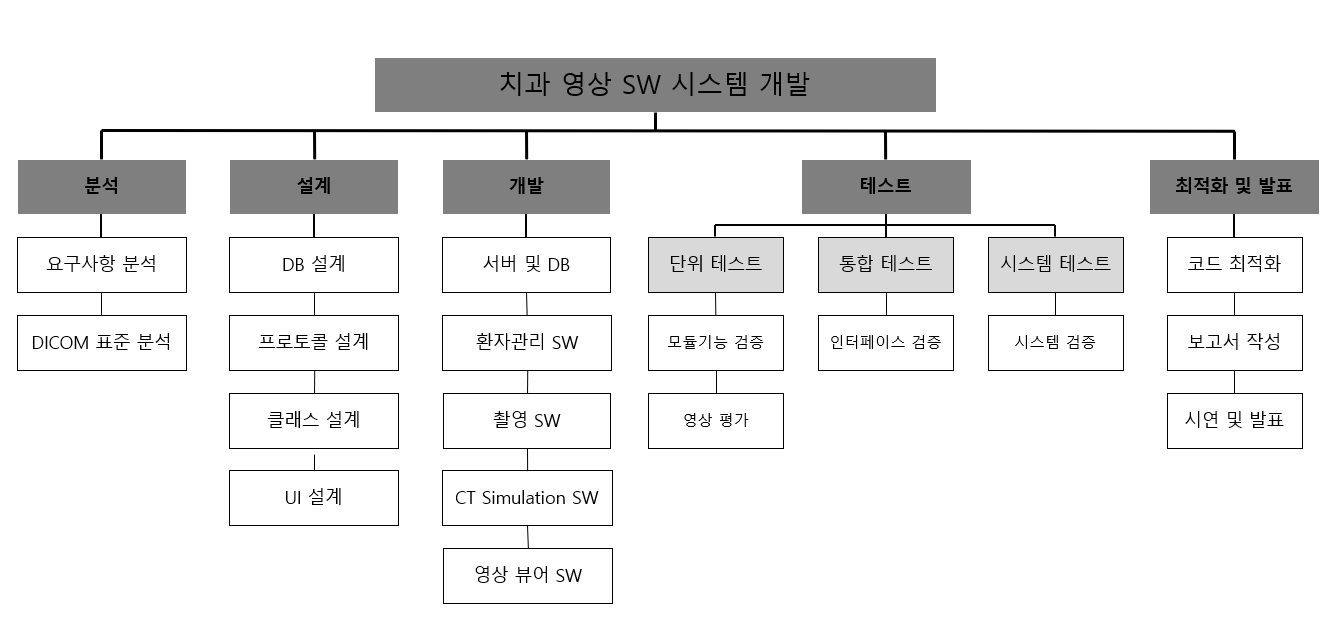
**4.2. UI 구성 예시**

**4.3. 데이터베이스 구성**

**5. 프로젝트 관리**

**5.1. 프로젝트 작업 계획**

5.1.1. WBS

****

5.1.2. 세부일정 추진 계획 (멘토링 일정에 따라 추후 변동 예정)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **단계** | **세부 단계** | | **기간(일)** | **2023년 01월** | | | | **2023년 02월** | | | |
| **1주** | **2주** | **3주** | **4주** | **1주** | **2주** | **3주** | **4주** |
| **분석** | 요구사항 분석 | | 3 |  |  |  |  |  |  |  |  |
| DICOM 표준 분석 | | 10 |  |  |  |  |  |  |  |  |
| **설계** | DB 설계 | | 4 |  |  |  |  |  |  |  |  |
| 프로토콜 설계 | | 4 |  |  |  |  |  |  |  |  |
| 클래스 설계 | | 4 |  |  |  |  |  |  |  |  |
| UI 설계 | | 4 |  |  |  |  |  |  |  |  |
| **개발** | 서버 및 DB 구현 | | 12 |  |  |  |  |  |  |  |  |
| 환자관리 SW 구현 | | 20 |  |  |  |  |  |  |  |  |
| 촬영 SW 구현 | | 20 |  |  |  |  |  |  |  |  |
| 영상장비 SW 구현 | | 20 |  |  |  |  |  |  |  |  |
| 뷰어 SW 구현 | | 20 |  |  |  |  |  |  |  |  |
| **테스트** | 단위 테스트 | 모듈기능 검증 | 4 |  |  |  |  |  |  |  |  |
| 영상 평가 | 4 |  |  |  |  |  |  |  |  |
| 통합 테스트 | 인터페이스 검증 | 4 |  |  |  |  |  |  |  |  |
| 시스템 테스트 | 시스템 검증 | 4 |  |  |  |  |  |  |  |  |
| **최적화 ∙ 발표** | 코드 최적화 | | 4 |  |  |  |  |  |  |  |  |
| 보고서 작성 | | 10 |  |  |  |  |  |  |  |  |
| 시연 및 발표 | | 1 |  |  |  |  |  |  |  |  |

**5.2. 코딩 가이드**

5.2.1. Naming Convention

명명 규칙은 크게 카멜 표기법과 파스칼 표기법을 따른다.

* camelCase: 지역 변수, 매개 변수, 클래스 멤버 변수, 클래스 멤버 함수

(예: int patientNumber, void memFunc())

* PascalCase: 타입 이름(클래스명, namespace, enum 클래스)

매크로 이름은 모두 대문자로 작성하는 것을 원칙으로 한다. 두 단어의 연결이 필요한 경우는 ‘\_(언더바)’를 사용한다. (예: #define MAX\_NUM 3.0)

5.2.2. Code Block

함수와 클래스 등 타입은 BSD 스타일로 Code Block을 묶는다. 조건문과 반복문 등 제어문은 K&R 스타일을 사용한다. 다음은 두 스타일의 예시이다.

|  |  |
| --- | --- |
| **BSD** | **K&R** |
| class BsdClass : public parent  {  (…)  }  void BsdClass::bsdFunction()  {  (…)  } | for (int I = 0; I < 10; I++) {  if (I == 0) {  (…)  } else {  (…)  }  } |

5.2.3. 주석 스타일

**5.3. 형상 관리 방안**

5.3.1. 형상관리 대상

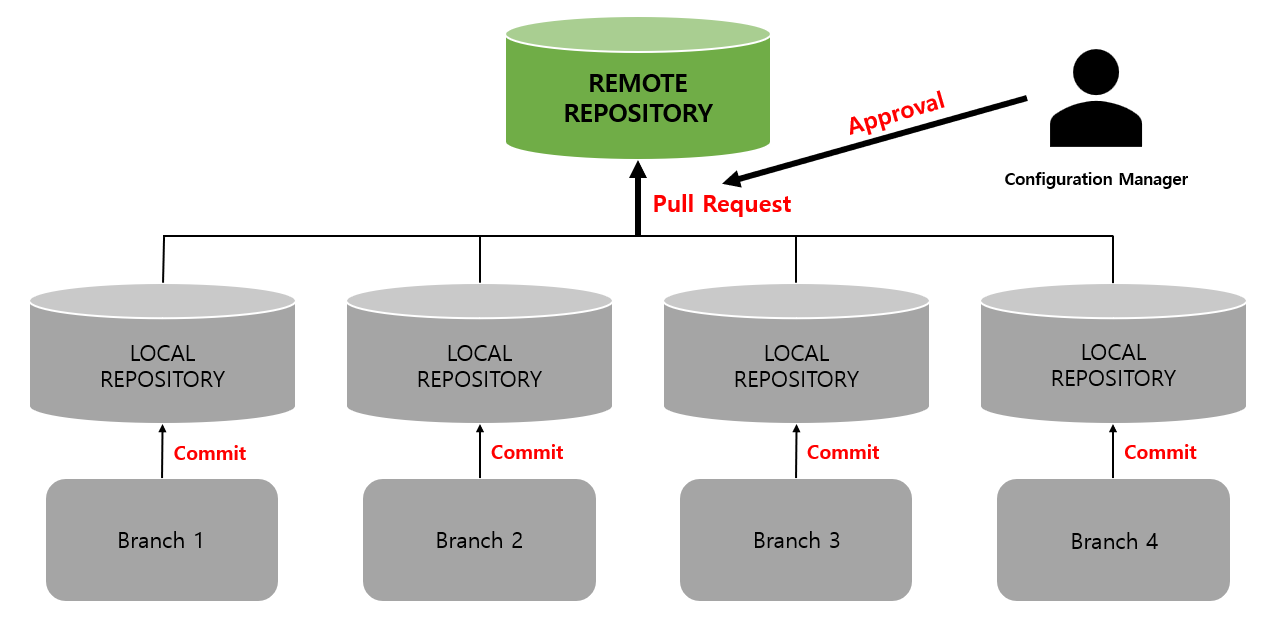
프로젝트에서 공식적으로 정의되고 관리가 필요한 모든 대상을 그 범위로 한다.

- 문서: \*.ppt, \*.md, \*.docx, \*.pdf 등

- 개발소스: 디자인 파일(\*.png, \*.bmp, \*.jpg), 소스 파일(\*.pro, \*.h, \*.cpp, \*.ui, \*.obj, \*.lib)

5.3.2. 형상관리 방법

분산저장소 타입의 Git을 통해 관리한다. 구성은 다음과 같다.

****

개발자는 각각의 Branch를 통해 작업하고, Local Repository에서 작업한다. Remote Repository에 대한 Merge작업은 형상관리자의 승인을 통해 진행한다. 자세한 관리 규칙은 다음과 같다.

* Default Branch는 ‘main’으로 한다.
* Branch명은 다음을 기본으로 한다.

∙ 김유선: kim

∙ 박병규: park

∙ 안다미로: an

∙ 이정연: lee

* 원활한 관리를 위해 Merge 요청에 대한 승인은 형상관리자(팀장)가 진행한다.
* 형상관리자는 Merge를 진행한 후, 즉시 팀원에게 git pull 요청을 한다.
* 충돌이 없도록, git pull 요청을 받은 팀원은 가능한 즉시 git pull을 진행한다.
* 효과적인 진행상황 관리와 과거 코드 추적을 위해 다음과 같이 Commit Message 규칙을 정한다.

∙ ‘type : contents’ 형식으로 Commit Message를 구성한다. (예: feat : Add login widget)

∙ type은 아래 정의된 내용을 기반으로 작성한다.

1. feat: 새로운 기능에 대한 Commit
2. fix: 버그 수정에 대한 Commit
3. docs: 문서 수정
4. style: 코드 포맷팅, 세미콜론(;) 누락, 코드 변경이 없는 경우
5. refactor: 코드 리팩토링
6. test: 테스트 코드, 리팩토링 테스트 코드 추가(프로덕션 코드 변경 X)
7. chore: 빌드 업무 수정, 패키지 매니저 수정(프로덕션 코드 변경 X)
8. design: 사용자 UI 디자인 변경
9. comment: 필요한 주석 추가 및 변경
10. rename: 파일 혹은 폴더명을 수정하거나 옮기는 작업만인 경우

∙ contents의 내용은 대문자로 시작하고, 명령형으로 작성한다.

∙ type과 :(콜론), contents 사이는 한 칸의 공백을 준다.

∙ type뒤에 괄호를 통해 파일의 이름을 작성할 수 있다. (예: docs(README.md) : Change layout)

**5.4. 테스트 방안**

테스트 기간 중에는 전체 시스템에 영향을 미치는 변경은 자제하고, 결과의 정확성에 집중한다. 테스트에서 오류가 발생한 경우에는 각 테스트별 계획에 따라 검증과 수정 후 테스트를 다시 진행한다.

모든 테스트는 다음과 같은 목표를 가지고 진행한다.

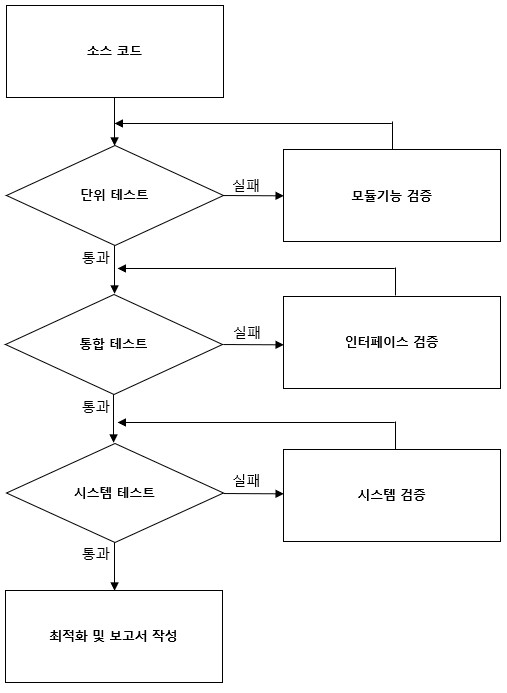
- 작업명세서 내 항목에 대한 이행여부 확인

- 코딩 가이드 준수 여부

- 프로그램 사용성에 대한 평가

- 시스템 및 네트워크의 성능 및 효율 검증

- 영상 품질 검증

5.4.1. SW 테스트 모형

5.4.2. 단위 테스트 계획

- 테스트 범위: 각각의 모듈내에서 메서드 또는 SW 수준으로 진행

- 대상 프로그램: 환자관리 SW, 촬영 SW, CT Simulation SW, 뷰어 SW

|  |  |
| --- | --- |
| **테스트 항목** | **평가 기준** |
| ∙ UI 기본 기능 테스트 – 정보 표시 | ∙ 가독성 – 자체 상호 평가  ∙ 정확성 |
| ∙ UI 기본 기능 테스트 – 위젯 동작 | ∙ 에러 발생 여부  ∙ 작업명세서 이행여부 |
| ∙ UI 기본 기능 테스트 – 종료 | ∙ 에러 발생 여부 |
| ∙ 동작 테스트 – 작업명세서 내 항목 | ∙ 동작 성공/실패 |
| ∙ (일부) 영상 평가 – 평가모델 추후결정 | ∙ 평가 모델에 따른 정량화 |

5.4.3. 통합 테스트 계획

단위 테스트가 완료된 모듈을 대상으로 의도대로 협력하는지 테스트한다. 통합 테스트에서는 네트워크와 DB의 작동 및 연동성 평가를 주 목적으로 한다.

- 테스트 범위: 상호 운용이 필요한 모듈들을 2개 단위로 묶어서 진행

|  |  |
| --- | --- |
| **테스트 항목** | **평가 기준** |
| ∙ 환자관리 SW – DB 연동 | ∙ 데이터 전송 성공/실패  ∙ 데이터 손실 여부 |
| ∙ 환자관리 SW – 촬영 SW 연동 | ∙ 데이터 전송 성공/실패  ∙ 데이터 손실 여부 |
| ∙ 환자관리 SW – 영상 뷰어 SW 연동 | ∙ 데이터 전송 성공/실패  ∙ 데이터 손실 여부 |
| ∙ 촬영 SW – CT Simulation SW 연동 | ∙ 장비제어 명령의 전송 성공/실패  ∙ 영상 데이터 손실 여부  ∙ 영상 전송 속도 확인 |

5.4.4. 시스템 테스트 계획

시스템의 기능적인 측면과 비기능적인 요구사항에 대해서 검증한다.

|  |  |  |
| --- | --- | --- |
| **테스트 항목** | | **평가 기준** |
| ∙ 견고성 테스트 | ∙ 네거티브 테스트  ∙ 기대하지 않은 입력 데이터를 사용 | ∙ 예상하지 못한 오류 발생 여부 |
| ∙ 성능 테스트 | ∙ 서버의 통신 부하 테스트  ∙ 어플리케이션의 데이터 부하 테스트  ∙ 메모리 누수 검증  ∙ NPE(Null Point Exception) 체크 | ∙ 항목별 통과 여부 |
| ∙ 신뢰성 테스트 | ∙ 오류발생 시 데이터의 백업/복구 | ∙ 오류 발생 시 데이터 보존 여부 |

**5.5. 프로젝트 보고 계획**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **보고내용** | **보고 주기** | **보고자** |
| 기획보고 | ∙ 프로젝트 범위  ∙ 프로젝트 계획 설명 | 계획서 작성 완료 시점 | 팀 전원 |
| 멘토링보고 | ∙ 주간 회의록 보고  ∙ 팀원 상호 코드리뷰 내역  ∙ 작업 현황  ∙ 계획 변동 사항 | 멘토링 회차별 진행 | 팀 전원 |
| 완료보고 | ∙ 프로젝트 결과보고서 제출 | 프로젝트 종료 시점 | 팀 전원 |