소프트웨어 요구사항명세서(SRS)

**<제약회사 약품 수요예측 솔루션>**

**01-14조 조장** 이본규(2021041090) **조원** 김재훈(2021041092) **조원** 강민석(2021041088)

차례

**1. 개요**

1.1 개발목표

1.2 개발범위 및 내용

1.3 제품 가치

1.4 대상 고객

1.5 사용 시나리오

1.6 업무분석표(WBS)

1.7 용어 정의

**2. 요구사항 분석**

**2.1 기능적 요구사항**

2.1.1 사용사례 분석

2.1.2 데이터 처리 흐름도

2.1.3 시스템 작업내용

2.1.4 관리 기능

2.1.5 트랜잭션 처리 작업

2.1.6 모든 화면에 대해 수행되는 작업의 세부 사항

**2.2 비기능적 요구사항**

2.2.1 성능 요구사항

2.2.2 신뢰성 요구사항

2.2.3 부하 및 스트레스 요구사항

2.2.4 보안 요구사항

2.2.5 운영(용량) 요구사항

2.2.6 사용성 요구사항

2.2.7 호환성 요구사항

2.2.8 안정성 및 가용성 요구사항

2.2.9 확장성 요구사항

2.2.10 유지보수성 요구사항

2.2.11 설치 요구사항

2.2.12 복구 요구사항

2.2.13 법적 규정 및 환경 요구사항

2.2.14 위험분석 및 제약 요구사항

2.2.15 문서 산출물 요구사항

**3. 외부 인터페이스 요구사항**

3.2 하드웨어 인터페이스 요구사항

3.3 소프트웨어 인터페이스 요구사항

3.4 통신 인터페이스 요구사항

**4. 사용자 인터페이스 요구사항 및 설계**

4.1 사용자 인터페이스 요구사항

4.2 사용자 인터페이스 설계

4.2.1 제약회사 약품제조 관리 직원용 UI

4.2.2 시스템관리자용 UI

**5. 기능적 모델링**

5.1 사용사례 다이어그램

5.2 사용사례 명세(사용 시나리오)

5.2.1 회원가입

5.2.2 로그인

5.2.3 약품데이터 업로드

5.2.4 알림확인

5.2.5 수요예측 결과확인

**6. 구조적 모델링**

6.1 클래스 다이어그램

6.2 CRC cards과 클래스 명세

**7. 데이터베이스 스키마 명세**

7.1 약품파일DB 스키마 테이블 명세서

7.2 사용자DB 스키마 테이블 명세서

7.3 수요예측결과DB 스키마 테이블 명세서

7.4 수요예측모델 스키마 테이블 명세서

**8. 행위 모델링: 사용사례에 대한 시퀀스 다이어그램**

8.1 회원가입(5.2.1)에 대한 시퀀스 다이어그램

8.2 로그인(5.2.2)에대한 시퀀스 다이어그램

8.3 약품데이터 업로드(5.2.3)에대한 시퀀스 다이어그램

8.4 알림확인(5.2.4)에대한 시퀀스 다이어그램

8.5 수요예측 결과확인(5.2.5)에대한 시퀀스 다이어그램

**9. 클래스의 메소드 알고리즘 명세**

9.1 클래스 사용자 메소드 알고리즘 명세

9.1.1 Class 사용자::addUser() 알고리즘 명세

9.1.2 Class 사용자::deleteUser() 알고리즘 명세

9.1.3 Class 사용자::updateUser() 알고리즘 명세

9.1.4 Class 사용자::searchUser() 알고리즘 명세

9.2 Class DB 메소드 알고리즘 명세

9.2.1 Class DB::addDatabase()알고리즘 명세

9.2.2 Class DB::deleteDatabase()알고리즘 명세

9.2.3 Class DB::searchDatabase()알고리즘 명세

9.2.4 Class DB::updateDatabase()알고리즘 명세

9.3 Class 데이터전처리 메소드 알고리즘 명세

9.3.1 Class 데이터전처리::upload()알고리즘 명세

9.3.2 Class 데이터전처리::delete()알고리즘 명세

9.3.4 Class 데이터전처리::update()알고리즘 명세

9.4 Class 로그인&회원가입 메소드 알고리즘 명세

9.4.1 Class 로그인&회원가입::login()알고리즘 명세

9.4.2 Class 로그인&회원가입::logout()알고리즘 명세

9.4.3 Class 로그인&회원가입::makeid()알고리즘 명세

9.4.4 Class 로그인&회원가입::makePassword()알고리즘 명세

9.5 Class 수요예측모델 메소드 알고리즘 명세

9.5.1 Class 수요예측모델::set\_range()알고리즘 명세

9.5.2 Class 수요예측모델::run\_model()알고리즘 명세

9.5.3 Class 수요예측모델::settarget()알고리즘 명세

9.5.4 Class 수요예측모델::addResult()알고리즘 명세

9.5.5 Class 수요예측모델::deleteResult()알고리즘 명세

9.5.6 Class 수요예측모델:: addfeature()알고리즘 명세

9.5.7 Class 수요예측모델::deletefeature()알고리즘 명세

9.5.8 Class 수요예측모델::setLearningrate()알고리즘 명세

9.6 Class 알림 메소드 알고리즘 명세

9.6.1 Class 알림::alarm\_ON()알고리즘 명세

9.6.2 Class 알림::alarm\_OFF()알고리즘 명세

9.6.3 Class 알림::sendflag()알고리즘 명세

9.7 Class 시각화시스템 메소드 알고리즘 명세

9.7.1 Class 시각화시스템::buildgraph()알고리즘 명세

**10. 요구사항 검토 결과**

**11. 결론**

**부록**

**참고문헌**

**1. 개요**

**1.1 개발목표**

제약회사의 약품 판매 데이터를 학습하여 분기별 약품 수요를 예측하는 웹 어플리케이션

**1.2 개발 범위 및 내용**

* 로그인 기능 : 홈페이지에 id와 pw를 입력하여 로그인 수행하는 기능
* 회원가입 기능 : 사용자를 인증하기 위한 id와 pw를 등록하는 기능
* 파일 변환 기능 : xls 파일을 csv 파일로 변환해 주는 기능
* 파일 업로드 기능 : 변환 된 csv 파일을 업로드 해주는 기능
* 수요 예측 모델 관리 기능 : 수요예측을 수행하기 위한 입력데이터를 관리하고 ,변수와 목표를 설정하는 기능
* 알림 관리 기능 : 수요 예측이 완료되면 사용자에게 알림을 보내주는 기능
* 데이터베이스 관리 기능 : 사용자DB , 약품 파일DB , 수요 예측 결과 DB 와 같은 DB에 접근하여 관리 할 수 있는 기능
* 사용자 관리 기능 : 사용자를 등록/수정/삭제/검색 할 수 있는 기능
* 유저 인터페이스 관리 기능 : 홈페이지에 나타나는 UI를 관리 할 수 있는 기능

**1.3 제품 가치**

제약회사의 약품제조 수요를 예측하여 , 분기별 다음 약품 생산량을 미리 생산 할 수 있다. 이로인해 약품 재고량을 부족하지 않게 관리 할 수 있게 해준다.

**1.4 대상 고객**

제약회사 약품제조 관리 직원

**1.5 사용 시나리오**

제약회사의 약품 제작과 납품 공급 판매 데이터를 클라우드 서버에 업로드 한다. 업로드 된 데이터들은 수요예측 모델로 전달되어 수요예측을 실시한다. 수요예측이 완료되면 도표로 결과를 표현해주고 사용자에게 완료 알림을 보내준다. 제약회사의 경우에는 완성된 도표를 확인하여 약품 종류별 생산 수치를 분기별로 예측할 수 있다.

**1.6 업무분석표(WBS)**

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**1.7 용어 정의**

* **클라이언트** : 서버 업로드를 위해 End-User 단에서 사용되는 프로그램
* **서버** : 약품 데이터를 업로드 하기 위한 모든 서버들
* **데이터** : 이 문서에서는 약품 판매 데이터를 전부 데이터로 칭한다.

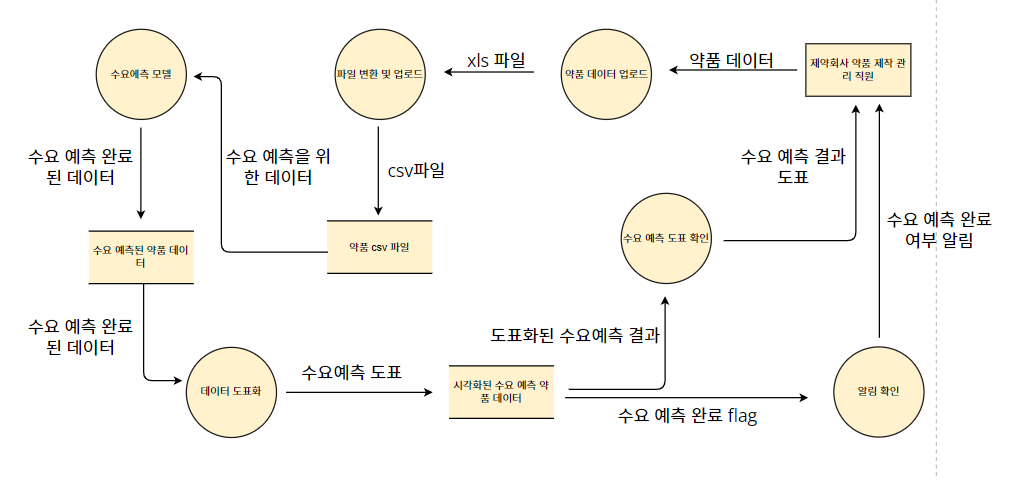
**2. 요구사항 분석**

**2.1 기능적 요구사항**

**2.1.1 사용사례 분석**

* 사용자는 ID를 생성한다.
* 사용자는 회원가입 시 생성한 ID와 비밀번호를 이용하여 로그인한다.
* 사용자는 비밀번호 변경 옵션을 선택하여 기존의 비밀번호를 새로운 것으로 변경할 수 있다.
* 사용자는 자신의 계정에서 회원 탈퇴 옵션을 선택하여 계정을 삭제한다.
* 사용자는 탈퇴 절차 중에 사용자의 ID와 PW 확인 과정을 진행한다.
* 클라이언트는 사용자의 ID와 PW 확인 후에만 계정 삭제가 진행된다.
* 사용자는 판매 데이터가 포함된 엑셀 파일을 선택하여 csv 파일로 변환하는 기능을 수행할 수 있다.
* 사용자는 엑셀 파일 내의 특정 열이나 행을 선택하여, 그 부분만을 대상으로 csv 변환을 실행할 수 있다.
* 클라이언트는 변환 과정에서 발생하는 모든 오류 또는 예외 상황에 대한 메시지를 사용자에게 명확하게 전달해야 한다.
* 클라이언트는 완성된 csv 파일은 원본 엑셀 데이터의 구조와 내용을 정확하게 유지되었는지 검증하는 기능이 필요하다.
* 사용자는 판매 데이터가 포함된 엑셀 파일을 선택하여 csv 파일로 변환하는 기능을 수행할 수 있다.
* 클라이언트는 변환된 csv 파일을 특정 컴퓨터 경로에 자동으로 저장한다.
* 클라이언트는 FileZilla를 이용하여 sFTP 로그인한다.
* 클라이언트는 해당 csv파일을 RIS 서버로 복사하여 클라우드에 자동으로 업로드한다.
* 서버는 복사 및 업로드 과정에서의 진행 상태 및 완료 여부가 실시간으로 클라이언트에게 전송해주어야 한다.
* 클라이언트는 서버에서 받은 진행 상태 및 완료 여부를 사용자에게 가시적으로 표시해준다.
* 서버는 모든 파일 전송이 완료되면, 클라이언트의 완료 플래그를 보낸다
* 클라이언트는 서버에서 알림 플래그를 받으면 사용자에게 완료 팝업 메시지를 띄운다..
* 사용자는 클라이언트로 시스템에 로그인 후, 시스템 관리 메뉴를 통해 최근 업로드된 파일 리스트를 확인할 수 있다.
* 서버는 자료들의 각각의 리스트 항목은 해당 파일의 이름, 크기, 업로드 일시 등의 정보를 포함해야 한다.
* 클라이언트는 서버에서 선택한 항목에 대한 상세 정보 조회 및 다운로드 옵션도 제공되어야 한다.
* 사용자는 클라이언트를 통해 업로드된 파일의 무결성을 검사 요청을 할 수 있다.:
* 클라이언트는 사용자에게 무결성 검사 요청을 받으면 서버에 검사 요청 파일을 요청한다.
* 서버는 요청된 파일과 MAC 파일을 클라이언트에게 보낸다.
* 클라이언트는 서버를 통해 받은 파일과 MAC을 통해 무결성 검사를 진행한다.
* 클라이언트는 두 MAC 값이 일치하지 않을 때 사용자에게 경고 메시지를 표시한다.
* 클라이언트는 두 MAC 값이 일치 할 때 사용자에게 확인 완료 메시지를 표시한다.
* 사용자는 클라이언트를 통해 최근 업로드된 파일 리스트에서 각각의 항목을 선택하여 상세 정보를 조회할 수 있다.
* 사용자는 문제가 발견된 파일에 대해 클라이언트를 통해 서버에 수정을 요청한다.
* 서버는 수정 요청을 받은 파일을 클라이언트로 보내준다.
* 클라이언트는 서버가 보낸 수정 요청 파일을 실행하여 수정할 수 있다.
* 사용자는 수정이 완료된 파일을 저장할 수 있다.
* 클라이언트는 사용자에게 수정 완료 플래그를 받을 시 서버에 수정된 파일을 업로드한다.
* 서버는 클라이언트가 보낸 수정 완료 파일을 원본 파일을 삭제하고 업로드 할 수 있어야 한다.
* 서버는 정해진 스케줄 혹은 트리거에 따라 클라우드에서 데이터를 자동으로 가져온다.
* 서버는 가져온 데이터는 중복, 누락 등의 기본적인 검사를 거친 후, 사용 가능한 형태로 전처리한다.
* 서버는 전처리 과정을 거친 데이터는 학습에 사용될 수 있도록 적절한 저장소에 저장한다.
* 사용자는 학습할 데이터의 범위나 기간을 설정한 뒤 학습 모델을 통해 수요 학습을 진행할 수 있다.
* 사용자는 원하는 범위나 기간을 클라이언트를 통해 지정한다.
* 클라이언트는 지정된 범위나 기간에 맞는 선택된 데이터를 서버에 보낸다.
* 서버는 AI 모델에 선택된 입력 데이터를 학습한다.
* AI 모델은 학습 완료 후 학습 모델을 서버로 보내준다.
* 서버 내에서는 학습 모델을 성능 평가 메트릭스로 확인한다.
* 서버는 성능 평가가 완료되면 AI 모델에게 최적화 완료 플래그를 보낸다.
* AI 모델은 최적화가 완료되면 새로운 수요 예측 모델을 기본으로 설정한다.
* AI 모델은 생성된 수요 예측 모델을 사용하여 향후 1년 동안의 약품 주문 목록 및 수량을 예측한다.
* AI 모델은 예측 결과를 서버를 통해 클라이언트에게 전송한다.
* 클라이언트는 받은 예측 결과를 그래프로 변환 후 사용자에게 제공한다. 이 때 제공
* 클라이언트는 수요 예측 결과에 따라 약품의 종류와 수량을 결정한다.

**2.1.2 데이터 처리 흐름도**

****

**2.1.3 시스템 작업내용**

* 시스템은 데이터를 업로드 할 수 있어야 한다.
* 시스템은 데이터를 보관할 수 있어야한다.
* 시스템은 데이터를 통해 수요 학습을 할 수 있어야 한다.
* 시스템은 수요 예측 데이터를 통해 예측 결과를 그래프로 보여줄 수 있어야 한다.
* 시스템은 수요 학습 완료 시 완료 Flag를 전송할 수 있어야 한다.

**2.1.4 관리 기능**

* 시스템관리자는 다양한 사용자에 대한 권한을 관리할 수 있어야 한다.
* 사용자는 데이터 업로드 관리를 할 수 있어야 한다.
* 사용자는 데이터를 삭제 할 수 있어야 한다.
* 시스템 관리자는 클라이언트의 기능 추가를 관리하여야 한다.

**2.1.5 트랜잭션 처리 작업**

* 시스템관리자는 파일이 업로드되고 파일 변환을 처리할 때 트랜잭션으로 묶어 처리해야한다.
* 시스템관리자는 업로드과정이나 파일변환 과정에 오류가 발생하면 롤백하여 이전 상태로 복원 해야한다.
* 시스템은 AI 모델에 학습 파일을 트랜잭션으로 묶어 처리해야 한다.
* 시스템은 학습 데이터 추가 시 새로운 학습 데이터를 반영하여 새로운 학습용 트랜잭션을 생성한다.
* AI 모델은 수요예측 결과 데이터 트랜잭션을 도표로 처리할 수 있어야 한다.

**2.1.6 모든 화면에 대해 수행되는 작업의 세부 사항**

* 사용자는 중복되지 않는 고유한 ID를 생성해야 한다.
* 클라이언트는 비밀번호는 특정 조건 (예: 최소 길이, 대/소문자, 숫자 또는 특수 문자 포함 등)에 부합한지 확인해야 한다.
* 서버는 회원가입 시 관리자 인증을 위한 키를 입력한다.
* 서버는 키를 통해 시스템 관리자와 일반 사용자를 구분한다.
* 사용자는 회원가입 시 생성한 ID와 비밀번호를 이용하여 로그인할 수 있다.
* 사용자는 비밀번호 변경 옵션을 선택하여 기존의 비밀번호를 새로운 것으로 변경할 수 있다.
* 클라이언트는 새로운 비밀번호 역시 회원가입 시 설정된 비밀번호 조건에 부합한지 확인한다.
* 사용자는 자신의 계정에서 회원 탈퇴 옵션을 선택하여 계정을 삭제할 수 있다.
* 사용자는 탈퇴 절차 중에 사용자의 ID와 PW 확인 과정을 진행한다.
* 클라이언트는 사용자의 ID와 PW 확인 후에만 계정 삭제가 진행된다.
* 사용자는 판매 데이터가 포함된 엑셀 파일을 선택하여 csv 파일로 변환하는 기능을 수행할 수 있다.
* 사용자는 엑셀 파일 내의 특정 열이나 행을 선택하여, 그 부분만을 대상으로 csv 변환을 실행할 수 있다.
* 클라이언트는 변환 과정에서 발생하는 모든 오류 또는 예외 상황에 대한 메시지를 사용자에게 명확하게 전달해야 한다.
* 클라이언트는 완성된 csv 파일은 원본 엑셀 데이터의 구조와 내용을 정확하게 유지되었는지 검증하는 기능이 필요하다.
* 사용자는 판매 데이터가 포함된 엑셀 파일을 선택하여 csv 파일로 변환하는 기능을 수행할 수 있다.
* 클라이언트는 변환된 csv 파일을 특정 컴퓨터 경로에 자동으로 저장한다.
* 클라이언트는 FileZilla를 이용하여 sFTP 로그인한다.
* 클라이언트는 해당 csv파일을 RIS 서버로 복사하여 클라우드에 자동으로 업로드한다.
* 서버는 복사 및 업로드 과정에서의 진행 상태 및 완료 여부가 실시간으로 클라이언트에게 전송해주어야 한다.
* 클라이언트는 서버에서 받은 진행 상태 및 완료 여부를 사용자에게 가시적으로 표시해준다.
* 서버는 모든 파일 전송이 완료되면, 클라이언트의 완료 플래그를 보낸다
* 클라이언트는 서버에서 알림 플래그를 받으면 사용자에게 완료 알림을 보낸다.
* 사용자는 클라이언트로 시스템에 로그인 후, 시스템 관리 메뉴를 통해 최근 업로드된 파일 리스트를 확인할 수 있다.
* 서버는 자료들의 각각의 리스트 항목은 해당 파일의 이름, 크기, 업로드 일시 등의 정보를 포함해야 한다.
* 클라이언트는 서버에서 선택한 항목에 대한 상세 정보 조회 및 다운로드 옵션도 제공되어야 한다.
* 서버는 클라이언트가 보낸 수정 완료 파일을 원본 파일을 삭제하고 업로드 할 수 있어야 한다.
* 서버는 정해진 스케줄 혹은 트리거에 따라 클라우드에서 데이터를 자동으로 가져온다.
* 서버는 가져온 데이터는 중복, 누락 등의 기본적인 검사를 거친 후, 사용 가능한 형태로 전처리한다.
* 서버는 전처리 과정을 거친 데이터는 학습에 사용될 수 있도록 적절한 저장소에 저장한다.
* 사용자는 학습할 데이터의 범위나 기간을 설정한 뒤 학습 모델을 통해 수요 학습을 진행할 수 있다.
* 사용자는 원하는 범위나 기간을 클라이언트를 통해 지정한다.
* 클라이언트는 지정된 범위나 기간에 맞는 선택된 데이터를 서버에 보낸다.
* 서버는 AI 모델에 선택된 입력 데이터를 학습한다.
* AI 모델은 학습 완료 후 학습 모델을 서버로 보내준다.
* 서버 내에서는 학습 모델을 성능 평가 메트릭스로 확인한다.
* 서버는 성능 평가가 완료되면 AI 모델에게 최적화 완료 플래그를 보낸다.
* AI 모델은 최적화가 완료되면 새로운 수요 예측 모델을 기본으로 설정한다.
* AI 모델은 생성된 수요 예측 모델을 사용하여 향후 1년 동안의 분기별 약품 주문 목록 및 수량을 예측한다.
* AI 모델은 예측 결과를 서버를 통해 클라이언트에게 전송한다.
* 클라이언트는 받은 예측 결과를 그래프나 도표로 변환 후 사용자에게 제공한다.
* 클라이언트는 모든 설정에 접근할 수 있는 메뉴와 판매 예측 추이를 확인할 수 있는 대시보드를 UI로 제공하여야 한다.

**2.2 비기능적 요구사항**

**2.2.1 성능 요구사항**

* 서버는 일주일 치 거래 데이터 약 300건이 1분 내외로 서버에 업로드가 되어야 한다.
* 서버는 업로드 시 1초 안에 클라이언트에게 성공/실패 여부를 알려줘야 한다.
* 서버에 존재하는 모든 데이터는 업로드 후 10분 안에 사용자가 클라이언트를 통해 접근 가능하여야 한다.
* 서버는 데이터 저장공간으로는 기업 당 1TB 공간을 활용한다.
* 서버는 저장공간을 전부 사용 시 자동 저장소 관리가 들어 갈 수 있어야 한다.

**2.2.2 신뢰성 요구사항**

* 서버는 업로드 된 자료의 신뢰율을 90% 이상을 유지한다.
* 서버는 업로드 완료 시 데이터를 100% 일관되게 유지한다.

**2.2.3 부하 및 스트레스 요구사항**

* 서버는 최소 MTBF를 6개월 이상 유지할 수 있어야한다.
* 서버는 최소 23시간동안 가동되어도 이상없이 운영되어야한다.
* 클라이언트는 대기 시간동안 컴퓨터 부하를 40% 이하로 유지될 수 있도록 한다.
* 업로드 컴퓨터는 아이들 상태에서 13시간 이상 버틸 수 있어야한다.

**2.2.4 보안 요구사항**

* 서버는 Microsoft 계정 관리 서버를 통해 계정을 관리한다.
* 서버는 Microsoft Azure 클라우드 환경 안에서 사용한다.

**2.2.5 운영(용량) 요구사항**

* 클라이언트의 용량은 최대 5GB를 넘어가지 않도록 한다.
* 클라이언트의 기능 및 최적화업데이트 패치 후 수정까지 최대 3일을 넘기지 않도록 한다.
* 서버는 최소 일주일에 한 번 점검 후 재가동을 시행할 수 있어야한다.
* AI알고리즘은 한달에 한 번 학습 후 새로운 모델을 적용할 수 있어야한다.
* MTTR은 2일이 넘어가지 않도록 한다.

**2.2.6 사용성 요구사항**

* 클라이언트는 모든 기능에 쉽게 접근할 수 있어야한다.
* 클라이언트는 3클릭 법칙을 준수하여아한다.
* 클라이언트는 메뉴에 대한 설명을 넣어 사용자가 이해할 수 있게 한다.
* 클라이언트의 설명서에는 모든 기능에 대한 상세 설명과 문제 해결 가이드, 문제 해결 고객센터 연락처를 포함하는 메뉴얼이 필요하다.
* 클라이언트는 문제 발생 시 항상 원격 관리자가 배치되어 직접적인 관리를 할 수 있어야 한다.
* 원격 관리자는 문제 발생시 즉각 대응하여 지원을 제공해야한다.
* 서버는  클라이언트를 통해 접근 가능하여야한다.

**2.2.7 호환성 요구사항**

* 서버는 데이터베이스 시스템에 SQL 언어를 통해 접근할 수 있어야 한다.
* 클라이언트는 윈도우 7 이상이 돌아가는 1.5Ghz 이상 CPU와 16GB 이상의 램을 가진 네트워크가 연결된 PC에서 작동 가능하여야한다.

**2.2.8 안정성 및 가용성 요구사항**

* 클라이언트는 데이터 업로드 PC에서 23시간 이상 작동되었을 때 컴퓨터의 총 자원의 15%를 사용하지 않도록 한다.
* 클라이언트는 최소 48시간 이상 가동이 되어도 이상없이 제 기능을 수행할 수 있어야한다.
* 서버는 가동 시간 내에서는 파일에 대한 접근 및 수정이 실시간으로 진행되어야한다.

**2.2.9 확장성 요구사항**

* 클라이언트는 모든 기능을 모듈화 시켜서 각 분야의 기능 추가 시 모듈 수정으로 기능 확장이 이루어 질 수 있도록 한다.
* 서버는 저장공간 부족이 발생할 시 자동으로 클라우드 시스템에 요청하여 용량 확장을 이루어질 수 있도록 한다.

**2.2.10 유지보수성 요구사항**

* 클라이언트는 시스템에서 발생하는 모든 버그 및 결함을 식별하여 유지보수 팀에 리포트를 전송할 수 있도록 한다.
* 유지보수 팀은 클라이언트를 통해 전송된 데이터를 2주 이내에 수정해야 한다.
* 서버는 업로드 데이터 검증 기능을 일주일에 한 번씩 실행하여 정확성과 무결성을 유지해야한다.
* 서버는 검증 과정에서 오류가 발견될 경우 클라이언트 알림을 통해 즉시 관리자에게 알림이 전송해야 한다.
* 서버는 대량의 데이터 처리나 복잡한 계산 수행 시 시스템 응답 시간은 5초를 초과하지 않아야 한다.
* 서버는 처리 데이터를 분당 최소 1000건의 트랜잭션을 처리할 수 있도록 최적화해야한다.
* 클라이언트는 새로운 데이터 형식이나 환경 변화가 생길 경우 1개월 이내에 호환성 업데이트를 진행해야한다.

**2.2.11 설치 요구사항**

* 클라이언트는 제약회사 컴퓨터에 웹 앱으로 설치되어 있어야한다.
* 클라이언트는 OS에 상관없이 웹 앱으로 설치될 수 있어야한다.

**2.2.12 복구 요구사항**

* 서버는 이전 버전의 데이터베이스를 새 데이터가 업데이트 후 1시간 이내에 백업 서버를 통해 백업한다.
* 서버는 현재 데이터의 해시값의 불일치를 발견하면 백업 서버를 통해 롤백 과정을 시행한다.

**2.2.13 법적 규정 및 환경 요구사항**

* 클라이언트는 개인정보 보호법을 준수하기 위해 회원가입 시 개인정보 제공 방침을 안내할 수 있어야한다.
* 클라이언트는 사용자의 개인정보 사용 권한을 6개월에 한 번씩 재동의를 받을 수 있어야한다.
* 서버는 정보통신망법 준수를 위해 데이터 전송과 보존 시 Triple DES등의 보안 시스템을 활용할 수 있어야한다.

**2.2.14 위험분석 및 제약 요구사항**

* 클라이언트는 업로드 데이터의 형식이 바뀔 수 있음을 대비하여야한다.
* 서버는 Azure 환경에서 허용되지 않은 부분에 대한 제약적 기능을 고려하여야한다.
* 서버는 급작스러운 사용자 확대 가능성이 존재한다.

**2.2.15 문서 산출물 요구사항**

* 클라이언트의 설명서에는 시스템의 모든 기능에 대한 명확한 설명과 일반적인 문제 해결 방법에 대한 정보, 문제 해결 고객센터 연락처를 포함하는 메뉴얼이 필요하다.
* 개발자는 시스템 아키텍처, 요구사항분석서, 프로젝트 요약서, 코드 및 클래스 다이어그램을 포함한 개발 문서를 작성해야한다.
* 개발자는 이 소프트웨어 제작 시 사용한 오픈 소스의 코드와 서비스의 API에 대한 상세한 문서를 작성한다.
* API 문서는 각 API 엔드포인트, 요청/응답 형식, 에러 코드 등에 대한 명확한 정보를 제공한다.
* 개발자는 버그 리포팅 플로우 차트, 성능 검사 방법, 처리 가이드, 데이터 백업 및 복구, 클래스간 Coupling 현황의 내용을 포함한 시스템 유지보수 문서를 작성하여야한다.
* 개발자는 작업 분류, 우선 순위, 진척도, 리소스 활용 목록 등을 포함한 프로젝트 관리문서가 필요하다

**3. 외부 인터페이스 요구사항**

**3.1 하드웨어 인터페이스 요구사항**

* 클라이언트 가동 컴퓨터는 윈도우 7 이상이 돌아가는 1.5Ghz 이상 CPU와 16GB 이상의 램을 가진 네트워크가 연결된 PC에서 구동될 수 있어야 한다.
* 서버는 TPM 2.0 이상의 하드웨어 보안 환경이 구축되어있는 클라우드 서버 단말을 활용한다.

**3.2 소프트웨어 인터페이스 요구사항**

* 클라이언트는 비밀번호와 Microsoft 인증이 가능하여야 한다.
* 윈도우와 맥os를 포함한 웹을 실행할 수 있는 모든 PC에서 구동될 수 있도록 한다.

**3.3 통신 인터페이스 요구사항**

* 인터넷 속도는 500MBps 이상을 유지한다..
* 인터넷 속도 500Mbps 이상을 유지한다.

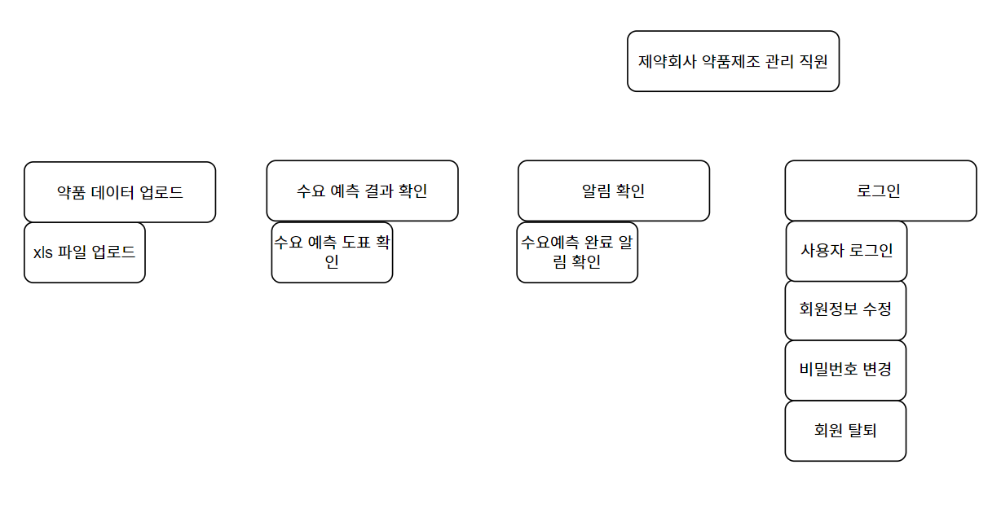
**4. 사용자 인터페이스 요구사항 및 설계**

**4.1 사용자 인터페이스 요구사항**

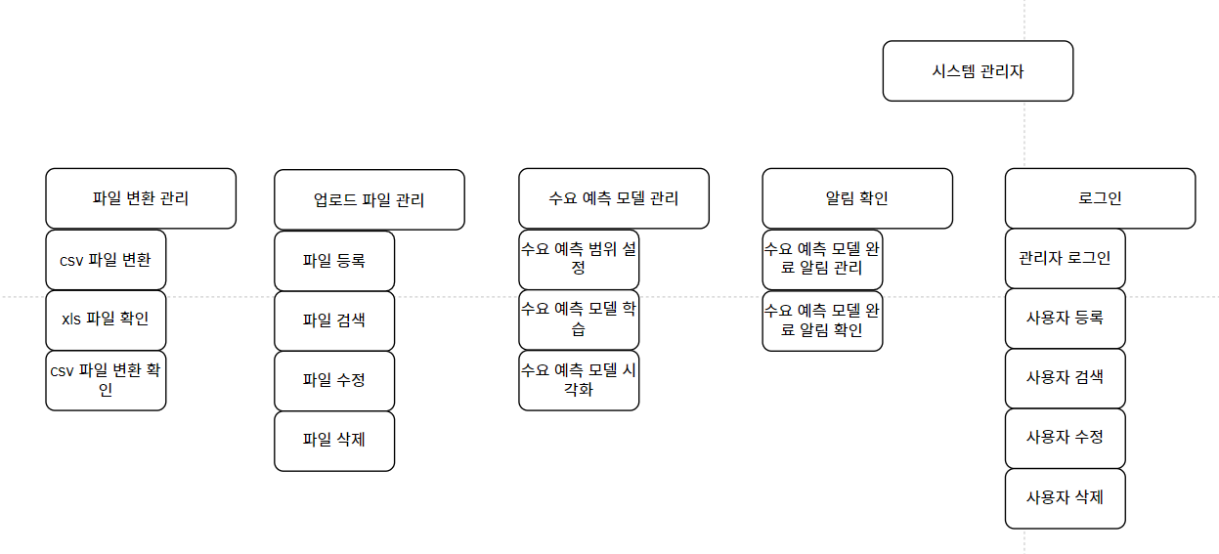
* 사용자는 회원가입을 통해 로그인에 사용 할 ID와 PW를 만들 수 있다.
* 사용자는 회원정보를 수정하거나 탈퇴할 수 있다.
* 사용자는 수요예측하고자 하는 약품데이터를 xls파일로 업로드 할 수 있다.
* 사용자는 알림확인을 통해 수요예측 완료 여부를 알 수 있다.
* 사용자는 수요예측 결과 확인을 통해 수요 예측 도표를 확인 할 수 있다.
* 시스템 관리자는 관리자 인증을 통해 관리자로 로그인 할 수 있다.
* 시스템 관리자는 사용자 관리를 통해 사용자를 등록, 검색, 수정, 삭제 할 수 있다.
* 시스템 관리자는 업로드 파일 관리를 통해 파일을 등록, 검색, 수정, 삭제 할 수 있다.
* 시스템 관리자는 파일 변환 관리를 통해 변환 전 xls 파일과 변환된 csv 파일을 확인 할 수 있다. 그리고 csv 파일로 변환되는 과정을 관리 할 수 있다.
* 시스템 관리자는 수요예측 모델 관리를 통해 수요예측 범위를 설정하고, 학습을 진행하고 , 도표화를 진행할 수 있다.
* 시스템 관리자는 알림관리를 통해 수요예측 모델 학습 완료 여부를 알림으로 관리 할 수 있다.

**4.2 사용자 인터페이스 설계 (관리자용, 고객용, App용 등)**

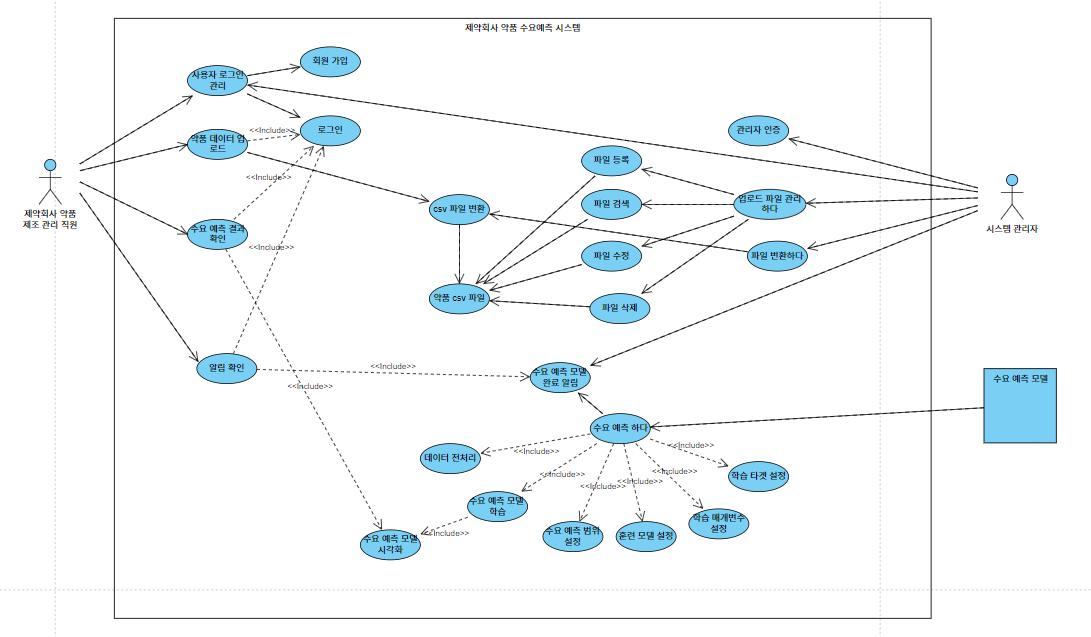
**4.2.1제약회사 약품제조 관리 직원용 UI**

****

**4.2.2 시스템 관리자용 UI**

****

**5. 기능적 모델링**

**5.1 사용사례 다이어그램 **

**5.2 사용사례 명세(사용 시나리오)**

**5.2.1 회원가입**

|  |  |
| --- | --- |
| Use Case Name | 회원가입 |
| Goal | 사용자가 시스템에 ID와 PW를 등록한다. |
| Actor | 제약회사 약품 제조 관리 직원 |
| Trigger | 시스템에 회원가입 버튼 클릭 |
| Relationships | Association : 회원정보 수정 , 비밀번호 변경 , 회원 탈퇴 |
| Normal Flow of Events | 1. 사용자가 회원가입 버튼을 누름 2. 사용자가 로그인에 사용할 ID를 입력 3. 사용자가 로그인에 사용할 PW를 입력 4. ID 와 PW가 중복되지 않는지 확인 5. 비중복이 확인되면 ID , PW 생성 6. 회원가입 성공 |
| Alternate/Exceptional Flows | 4a-1. ID가 이미 존재하면 1번 과정으로 돌아간다.  4a-2. PW가 이미 존재하면 1번 과정으로 돌아간다. |

**5.2.2 로그인**

|  |  |
| --- | --- |
| Use Case Name | 로그인 |
| Goal | 사용자가 시스템에 로그인을 한다. |
| Actor | 제약회사 약품 제조 관리 직원 |
| Trigger | 시스템에 로그인 버튼 클릭 |
| Relationships | Include : 약품 데이터 업로드 , 수요예측 결과 확인 , 알림확인 |
| Normal Flow of Events | 1. 사용자가 로그인 입력 버튼 클릭한다. 2. 사용자가 ID 입력 3. 사용자가 PW 입력 4. 사용자가 로그인 버튼을 클릭한다. 5. 사용자DB에서 ID와 PW가 존재하는지 확인 6. 로그인 성공 7. 로그인된 홈페이지를 반환한다, |
| Alternate/Exceptional Flows | 5a-1. ID가 존재하지 않아 로그인에 실패시 2번 과정으로 돌아간다.  5a-2. ID의 PW가 일치하지 않아 로그인에 실패시 2번 과정으로 돌아간다. |

**5.2.3 약품 데이터 업로드**

|  |  |
| --- | --- |
| Use Case Name | 약품 데이터 업로드 |
| Goal | 사용자가 시스템에 수요예측을 원하는 약품데이터를 업로드한다. |
| Actor | 제약회사 약품 제조 관리 직원 |
| Trigger | 시스템에 xls약품데이터를 업로드 하고 파일 업로드 버튼을 클릭 |
| Relationships | Association : csv 파일 변환  Include : 로그인 |
| Normal Flow of Events | 1. 사용자가 약품 데이터 업로드 항목을 누름 2. 사용자가 수요예측을 원하는 약품 데이터를 xls파일로 업로드 한다. 3. 업로드 된 파일의 확장자가 xls인지 확인한다. 4. 사용자가 파일 업로드 버튼을 클릭한다. 5. 업로드된 파일이 csv파일로 변환된다. 6. 변환된 csv파일이 약품DB에 저장된다. |
| Alternate/Exceptional Flows | 3a. xls 파일이 아닌 다른 파일 확장자에 경우 파일 업로드가 거부된다. |

**5.2.4 알림 확인**

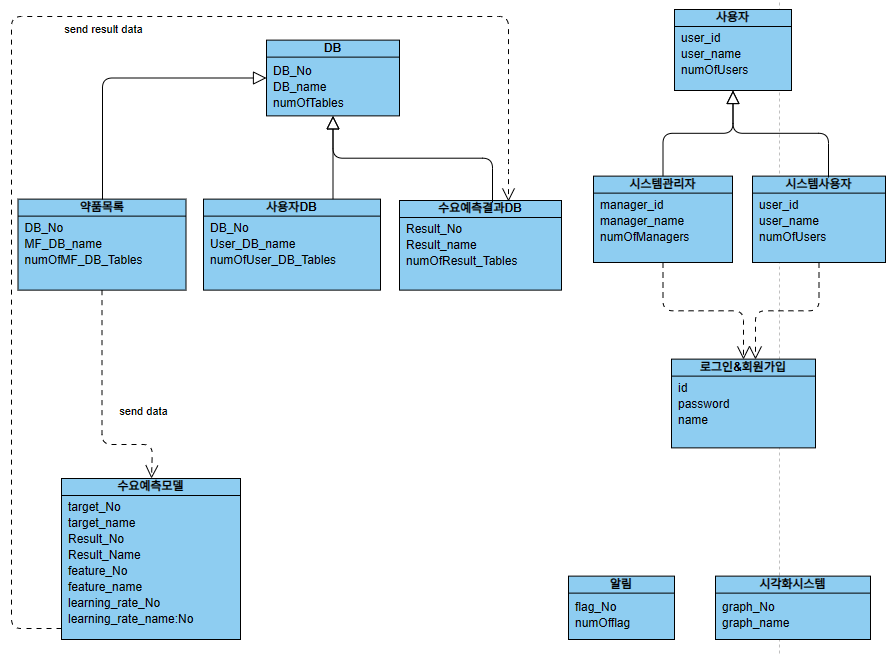
|  |  |
| --- | --- |
| Use Case Name | 알림 확인 |
| Goal | 사용자가 수요예측이 완료되었는지 알림을 확인한다. |
| Actor | 제약회사 약품 제조 관리 직원 |
| Trigger | 시스템에 알림 확인 버튼 클릭 |
| Relationships | Include : 로그인 , 수요 예측 모델 완료 알림 |
| Normal Flow of Events | 1. 사용자가 알림 확인버튼 누름 2. 사용자가 수요예측이 완료되었는지 알림을 확인 3. 수요예측이 완료되었으면 사용자는 알림 확인 가능 |
| Alternate/Exceptional Flows | 2a. 수요예측이 아직 완료되지 않았으면 알림을 보내지 않으므로 알림을 확인할 수 없다. |

**5.2.5 수요예측 결과 확인**

|  |  |
| --- | --- |
| Use Case Name | 수요예측 결과 확인 |
| Goal | 사용자가 수요예측 결과를 도표로 확인한다. |
| Actor | 제약회사 약품 제조 관리 직원 |
| Trigger | 시스템에 수요 예측 결과 확인 버튼 클릭 |
| Relationships | Include : 로그인 , 수요 예측 모델 시각화 |
| Normal Flow of Events | 1. 사용자가 수요예측 결과 확인버튼 클릭 2. 수요예측이 완료되었는지 확인 3. 시각화시스템으로 수요예측결과를 넘겨줌 4. 수요예측 결과를 시각화 5. 시각화된 수요예측 결과를 확인 |
| Alternate/Exceptional Flows | 2a-1. 사용자가 업로드한 약품데이터가 없으면 결과를 확인할 수 없음.  2a-2. 수요예측 학습이 진행중이면 결과를 확인할 수 없다. |

**6. 구조적 모델링**

**6.1 클래스 다이어그램**



**6.2 CRC cards과 클래스 명세**

**6.2.1 Class Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| 사용자ID | user\_ID | 20 | String | PK |
| 사용자이름 | user\_name | 10 | String | PK |
| 사용자수 | numOfUsers | 100000 | int | NN |
| public class User {  private List<User> users;  public User() {  this.users = new ArrayList<>();  }  // 사용자 추가  public void addUser(User user) {  this.users.add(user);  }  // 사용자 삭제  public void deleteUser(User user) {  this.users.remove(user);  }  // 사용자 검색  public User searchUser(String userID) {  for (User user : users) {  if (user.getUserID().equals(userID)) {  return user;  }  }  } // 사용자이름, 사용자 수 업데이트  public void updateUser(User user, String newUserName, int new NumOfUsers) {  user.setUserName(newUserName);  user.setNumOfUsers(newNumOfUsers);  } | | | | |

**6.2.2 Class Manger**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| 관리자ID | manager\_ID | 20 | String | PK |
| 관리자이름 | manager\_name | 10 | String | PK |
| 관리자수 | numOfManager | 100000 | int | NN |
| public class Manager {  private List<Manger> manager;  public Manager() {  this.manger = new ArrayList<>();  }  // 관리자 추가  public void addManger(Manger manger) {  this.Manger.add(manger);  }  // 관리자 삭제  public void deleteManger(Manger manger) {  this.manger.remove(manger);  }  // 관리자 검색  public Manger searchmanger(String mangerID) {  for (Manger manger : manger) {  if (manger.getUserID().equals(mangerID)) {  return manger;  }  }  } // 관리자이름, 관리자 수 업데이트  public void updateManger(Manger manger, String newMangerName, int new NumOfManger) {  user.setUserName(newMangerName);  user.setNumOfManger(newNumOfManger);  } | | | | |

**6.2.3 Class DB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| DB번호 | DB\_No | 10000 | int | PK,FK |
| DB이름 | DB\_name | 10 | String | PK |
| 테이블수량 | numOfTables | 100000 | int | NN |
| public class Database {  private int dbNo; //DB번호  private String dbName; //DB이름  private int numOfTables; // 테이블 수량    public Database(int dbNo, String dbName, int numOfTables) {  this.dbNo = dbNo;  this.dbName = dbName;  this.numOfTables = numOfTables;  }  }  public void addDB(Database DB) { //데이터베이스 추가  this.Database.add(DB);  }  public void deleteDB(Database DB) { //데이터베이스 삭제  this.Database.delete(DB);  }  public void searchDatabase(String DB) { //데이터베이스 검색  for (Database DB : DB) {  if (Database().equals(DB)) {  return DB;  }  } | | | | |

**6.2.4 Class Medicine\_List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| DB번호 | DB\_No | 10000 | int | PK,FK |
| 약품파일DB이름 | MF\_DB\_name | 10 | String | PK |
| 약품파일테이블수량 | numOfMF\_DB\_Tables | 100000 | int | NN |
| public class Medicine\_List{  private int DB\_No; //DB 번호  private String MF\_DB\_name; //약품파일DB이름  private int numOfMF\_DB\_Tables; //약품파일 테이블 수량    public Database(int DB\_No, String MF\_DB\_name, int  numOfMF\_DB\_Tables) {  this.dbNo = dbNo;  this.dbName = dbName;  this.numOfTables = numOfTables;  }  }  public void addMF(Medicine\_List MF) { //약품파일 추가  this.Medicine\_List.add(MF);  }  public void deleteMF(Medicine\_List MF) { //약품파일 삭제  this.Medicine\_List.delete(MF);  }  public void searchMF(String MF) { //약품파일 검색  for (Mdicine\_List MF : MF) {  if (Medicine\_List().equals(MF)) {  return MF;  }  } | | | | |

**6.2.5 Class User\_Database**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| DB번호 | DB\_No | 10000 | int | PK,FK |
| 사용자DB이름 | User\_DB\_name | 10 | String | PK |
| 사용자DB테이블수량 | numOf\_User\_DB\_Tables | 100000 | int | NN |
| public class User\_Database{  private int DB\_No;  private String User\_DB\_name;  private int numOf\_User\_DB\_Tables;    public User\_Database(int DB\_No, String User\_DB\_name, int  numOf\_User\_DB\_Tables) {  this. DB\_No = dbNo;  this. User\_DB\_name = dbName;  this. numOf\_User\_DB\_Tables = numOfTables;  }  }  public void addUserDB(User\_Database U\_DB) { //사용자데이터베이스 추가  this.User\_Database.add(U\_DB);  }  public void deleteUserDB(User\_Database U\_DB) { //사용자데이터베이스 삭제  this.User\_Database.delete(U\_DB);  }  public void searchUser\_Database(User\_Database U\_DB) { //사용자데이터베이스 검색  for (User\_Database U\_DB : U\_DB) {  if (User\_Database().equals(U\_DB)) {  return U\_DB;  }  } | | | | |

**6.2.6 Class Result\_Database**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| DB번호 | Result\_No | 10000 | int | PK |
| 결과이름 | Result\_name | 10 | String | PK |
| 결과테이블수량 | numOfR\_Tables | 100000 | int | NN |
| public class Result\_Database{  private int Result\_No;  private String Result\_name;  private int numOfR\_Tables;    public Result\_Database (int Result\_No, String Result\_name, int  numOfR\_Tables) {  this. Result\_No = RNo;  this. Result\_name = dbName;  this. numOfR\_Tables = numOfRTables;  }  }  public void addResultDB(Result\_Database R\_DB) { //수요예측결과DB 추가  this.Result\_Database.add(R\_DB);  }  public void deleteResultDB(Result\_Database R\_DB) { //수요예측결과DB 삭제  this.Result\_Database.delete(R\_DB);  }  public void searchResult\_Database(String R\_DB) { //수요예측결과DB 검색  for (Result\_Database R\_DB : R\_DB) {  if (Result\_Database().equals(R\_DB)) {  return R\_DB;  }  } | | | | |

**6.2.7 Class Demand Forcasting\_Model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼명** | **변수명** | **범위** | **타입** | **KEY** |
| 타겟번호 | Target\_NO | 10000 | int | NN |
| 타겟명 | Target\_name | 100 | String | NN |
| 결과번호 | Result\_NO | 10000 | int | PK,FK |
| 결과명 | Result\_name | 100 | String | PK,FK |
| 변수번호 | feature\_NO | 10000 | int | NN |
| 변수명 | feature\_name | 100 | String | NN |
| 학습률번호 | learning\_rate\_NO | 10000 | int | NN |
| 학습률이름 | learning\_rate\_name | 10000 | String | NN |
| public class DemandForecastModel {  private int targetNo;  private String targetName;  private int resultNo;  private String resultName;  private int featureNo;  private String featureName;  private int learningRateNo;  private String learningRateName;  // 수요예측 모델 학습 매개변수 설정  public DemandForecastModel(int targetNo, String targetName, int resultNo, String resultName, int featureNo, String featureName, int learningRateNo, String learningRateName) {  this.targetNo = targetNo;  this.targetName = targetName;  this.resultNo = resultNo;  this.resultName = resultName;  this.featureNo = featureNo;  this.featureName = featureName;  this.learningRateNo = learningRateNo;  this.learningRateName = learningRateName;  }    public void setRange() {  // 범위 설정 로직  System.out.println("Range is set.");  }    public void runModel() {  // 모델 실행 로직  System.out.println("Model is running.");  }    public void setTarget() {  // 타겟 설정 로직  System.out.println("Target is set.");  }    public void addResult() {  // 결과 추가 로직  System.out.println("Result is added.");  }    public void deleteResult() {  // 결과 삭제 로직  System.out.println("Result is deleted.");  }    public void addFeature() {  // 변수 추가 로직  System.out.println("Feature is added.");  }    public void deleteFeature() {  // 변수 삭제 로직  System.out.println("Feature is deleted.");  }  public void setLearningRate() {  // 학습률 설정 로직  System.out.println("Learning rate is set.");  }  } | | | | |

**6.2.8 Class Visualizasion system**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼** | **변수명** | **범위** | **타입** | **Key** |
| 그래프번호 | graph\_NO | 10000 | int | PK |
| 그래프명 | graph\_Name | 100 | String | PK |
| public class VisualizationSystem {  private int graph\_NO;  private String graph\_Name;  public VisualizationSystem(int graph\_NO, String graph\_Name) {  this.graph\_NO = graph\_NO;  this.graph\_Name = graph\_Name;  }  // 그래프 번호 getter, setter  public int getGraph\_NO() {  return graph\_NO;  }  public void setGraph\_NO(int graph\_NO) {  this.graph\_NO = graph\_NO;  }  // 그래프 이름  public String getGraph\_Name() {  return graph\_Name;  }  public void setGraph\_Name(String graph\_Name) {  this.graph\_Name = graph\_Name;  }  // 그래프 생성 메소드  public void buildGraph() {  }  } | | | | |

**6.2.9 Class Alarm**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **컬럼** | **변수명** | **범위** | **타입** | **Key** |
| 알림번호 | flag\_NO | 10000 | int | PK |
| 알림이름 | flag\_name | 10 | String | PK |
| public class Notification {  private int flag\_NO;  private String flag\_name;  private int alarm;  public Notification(int flag\_NO, String flag\_name) {  this.flag\_NO = flag\_NO;  this.flag\_name = flag\_name;  this.alarm = 0; // 초기 상태는 알람이 꺼진 상태로 설정  }  public int getFlag\_NO() {  return flag\_NO;  }  public void setFlag\_NO(int flag\_NO) {  this.flag\_NO = flag\_NO;  }  public String getFlag\_name() {  return flag\_name;  }  public void setFlag\_name(String flag\_name) {  this.flag\_name = flag\_name;  }  public int getAlarm() {  return alarm;  }  public void setAlarm(int alarm) {  this.alarm = alarm;  }  // 알림 켜기  public void alarm\_ON() {  this.alarm = 1;  }  // 알림 끄기  public void alarm\_OFF() {  this.alarm = 0;  }  // 알림 상태 전송 메소드  public void sendflag() {  }  } | | | | |

**7. 데이터베이스 스키마 명세**

**7.1 약품파일DB 스키마 테이블 명세서**

|  |  |  |
| --- | --- | --- |
| **MF\_DB\_NO** | **MF\_DB\_name** | **numOfMF\_DB\_Tables** |

**7.2 사용자DB 스키마 테이블 명세서**

|  |  |  |
| --- | --- | --- |
| **User\_DB\_NO** | **User\_DB\_name** | **numOfUser\_DB\_Tables** |

**7.3 수요예측결과DB 스키마 테이블 명세서**

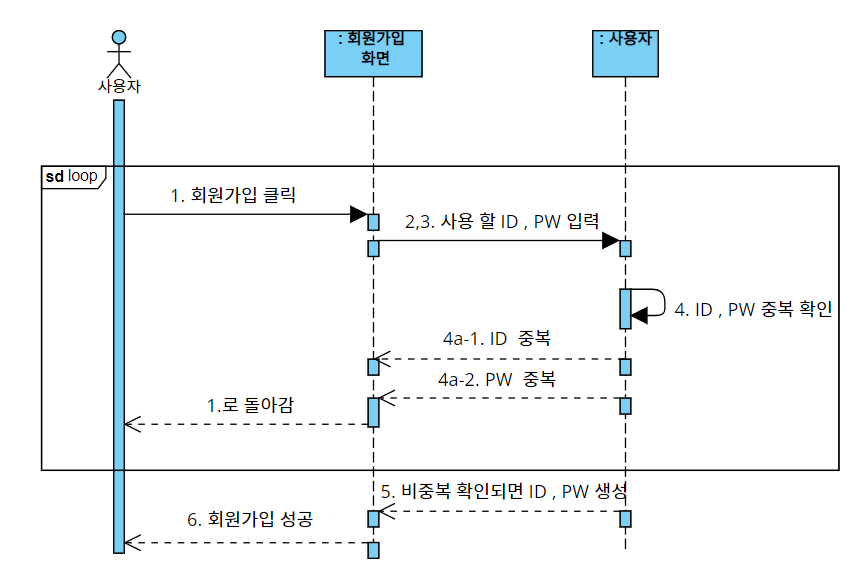
|  |  |  |
| --- | --- | --- |
| **Result\_NO** | **Result\_name** | **numOfResult\_DB\_Tables** |

**7.4 수요예측모델 스키마 테이블 명세서**

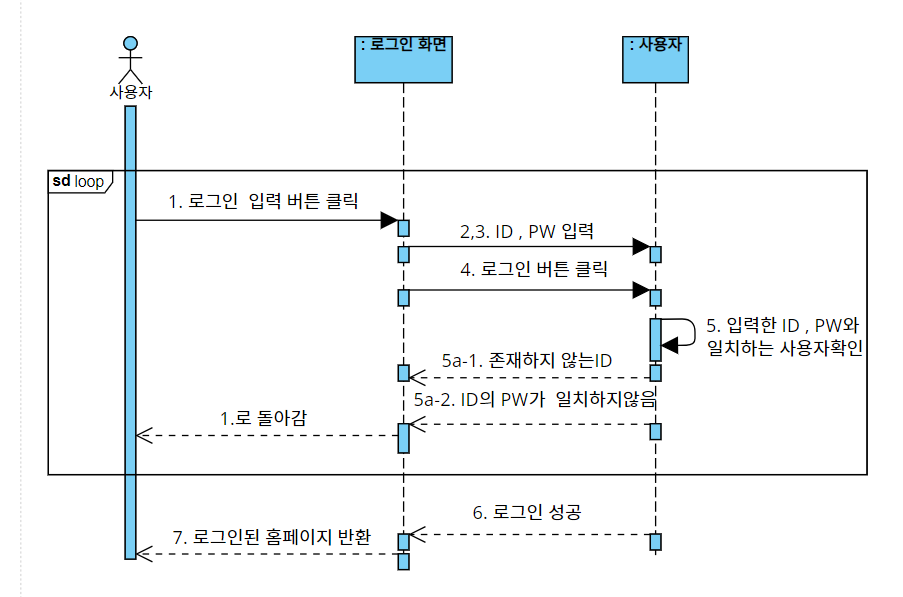
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Target\_NO** | **Target\_name** | **Result\_NO** | **Result\_name** | **feature\_NO** | **feature\_name** | **learning\_rate\_NO** | **learning\_rate\_name** |

**8. 행위 모델링: 사용사례에 대한 시퀀스 다이어그램**

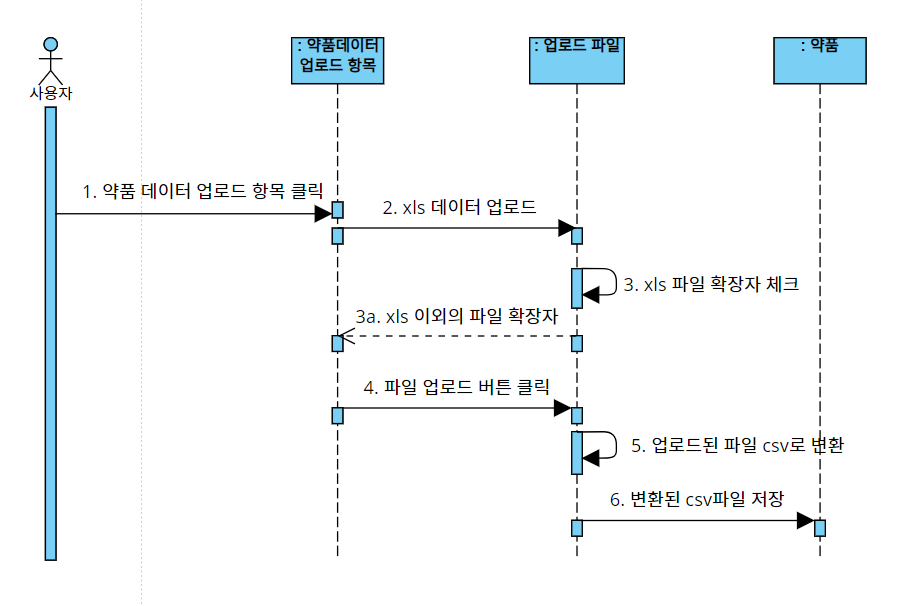
**8.1 회원가입(5.2.1)에 대한 시퀀스 다이어그램**

****

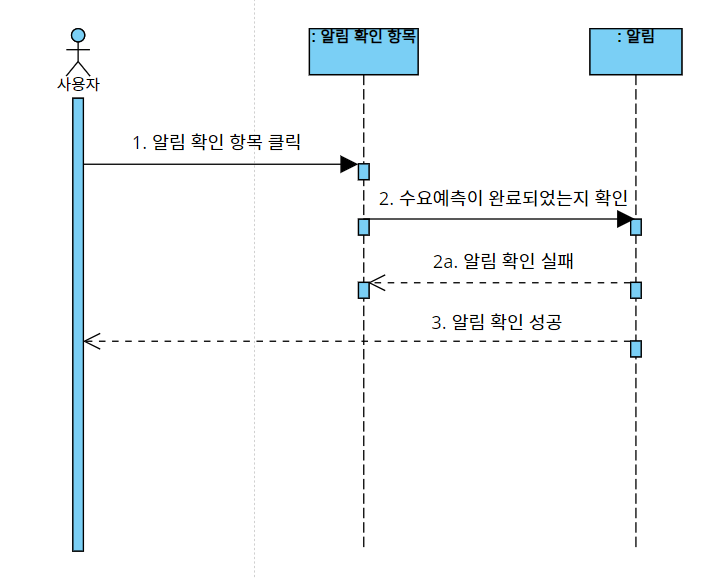
**8.2 로그인 (5.2.2)에 대한 시퀀스 다이어그램**

****

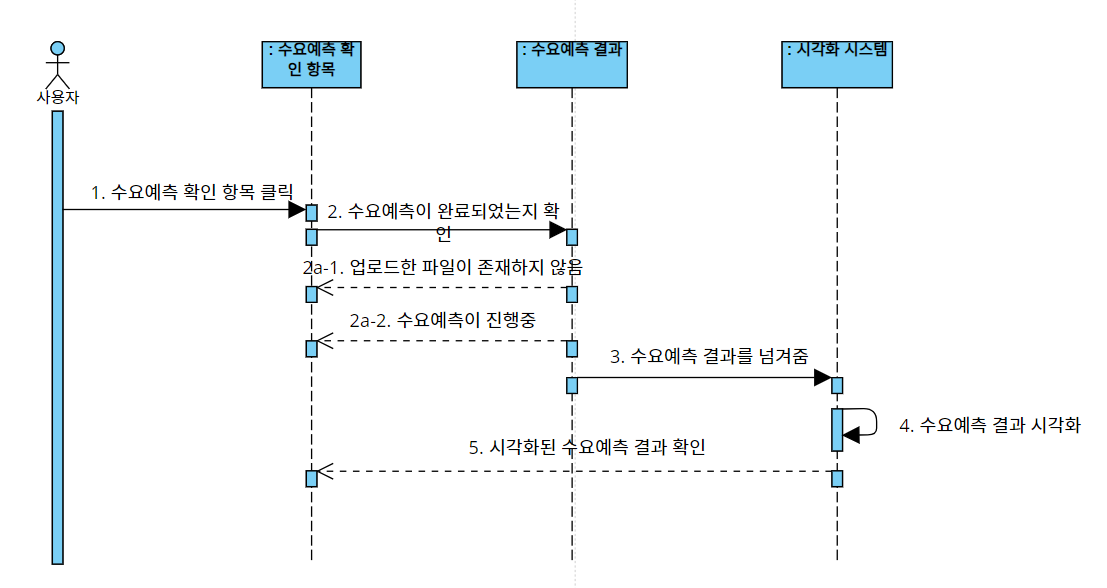
**8.3 약품데이터 업로드(5.2.3)에 대한 시퀀스 다이어그램**

****

**8.4 알림확인(5.2.4)에 대한 시퀀스 다이어그램**



**8.5 수요예측 결과확인(5.2.5)에 대한 시퀀스 다이어그램**

****

**9. 클래스의 메소드 알고리즘 명세**

**9.1 Class 사용자 메소드 알고리즘 명세**

**9.1.1 Class 사용자::addUser() 알고리즘 명세**

/\* ---------------------------------------------------------

인터페이스: addUser(id, pw)

기능: 사용자 클래스에서 새로운 회원을 등록한다.

매개변수: id=회원 id

pw=회원 pw

Return Value:0, else 에러코드 (“이미 존재하는 ID 입니다.”)

-----------------------------------------------------------\*/

def addUser(id,pw ): //사용자 id, pw 입력

{

//사용자 DB 확인

//만약 사용자DB에 id가 없다면 -> 새로운 사용자 추가

userDB[id] = pw

//만약 사용자DB에 id가 있다면

print("이미 존재하는ID입니다.")

}

**9.1.2 Class 사용자::deleteUser() 알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:deleteUser(id)

기능: 사용자 클래스에서 기존 회원을 삭제한다.

매개변수: id= 회원 id

Return Value:0, else 에러코드 (“이미 없는 ID 입니다.”)

-----------------------------------------------------------\*/

def deleteUser(id ): //삭제할 id 입력

{

//만약 사용자DB에 id가 있다면 -> 사용자 삭제

del userDB[id]

//만약 사용자DB에 id가 없다면

print("이미 없는 ID 입니다.")

}

**9.1.3 Class 사용자::updateUser() 알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:updateUser(id,pw)

기능: 사용자 클래스에서 사용자 정보를 수정한다.

매개변수: id= 회원 id

pw= 회원 pw

Return Value:0, else 에러코드 (“존재하지 않는 ID 입니다. 올바른 ID를 입력해주세요.”)

-----------------------------------------------------------\*/

def updateUser(id , pw): //수정할 id를 입력

{

//사용자 DB 확인

//만약 사용자DB에 id가 있다면 -> 어떤 정보를 수정할지 입력 받기

print("1. 비밀번호 변경")

print("2. 사용자 추가")

print("3. 사용자 삭제")

choice = input("선택: ")

If choice == 1 :

{

//1번을 입력받을 시(비밀번호 변경)

//새로운 비밀번호 입력

new\_pw = input("새로운 비밀번호 입력: ")

userDB[id] = new\_pw

print("비밀번호 변경 완료")

}

elIf choice == 2 : //2번을 입력받을 시 (사용자 추가)

{

addUser(id, pw):

}

elIf choice == 3 ://3번을 입력받을 시 (사용자 삭제)

{

deleteUser(id)

}

//만약 사용자 DB에 id가 없다면

print("존재하지 않는 ID입니다. 올바른 ID를 입력해주세요.")

}

**9.1.4 Class 사용자::searchUser() 알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:searchUser(id)

기능: 사용자 클래스에서 기존 회원을 조회한다.

매개변수: id= 회원 id

Return Value:0, else 에러코드 (“존재하지 않는 ID 입니다.”)

-----------------------------------------------------------\*/

def searchUser(id): //조회할 id 입력

{

//사용자 DB 확인

//만약 사용자DB에 id가 있다면 -> 해당 정보를 출력

get (‘id’)== id;

//만약 사용자DB에 id가 없다면

print("존재하지 않는 ID 입니다.")

}

**9.2 Class DB 메소드 알고리즘 명세**

**9.2.1 Class DB::addDatabase()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:addDatabase(data\_id)

기능: 데이터베이스에 데이터 추가

매개변수: data\_id = 약품데이터\_id

-----------------------------------------------------------\*/

def addDatabase(data\_id):

# 데이터베이스에 데이터 추가

database.append(data\_id)

/\* ---------------------------------------------------------

인터페이스: addDatabase(data\_id)

기능: DB 클래스에서 새로운 Database를 등록한다.

매개변수: data\_id= 약품 database id

Return Value:0, else 에러코드 (“이미 존재하는 ID 입니다.”)

-----------------------------------------------------------\*/

def addDatabase(data\_id)://data\_id 입력

{

//DB 확인

//만약 DB에 data\_id가 없다면 -> 새로운 database 추가

create database[data\_id]

//만약 DB에 data\_id가 있다면

print("이미 존재하는ID입니다.")

}

**9.2.2 Class DB::deleteDatabase()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:deleteDatabase(data\_id)

기능: DB 클래스에서 Database를 삭제한다.

매개변수: data\_id = 약품 database\_id

Return value: 0, else 에러코드("삭제할 데이터가 없습니다")

-----------------------------------------------------------\*/

def deleteDatabase(data\_id): //data\_id 입력

{

//DB 확인

//만약 DB에 data\_id가 있다면 -> database 삭제

del database[data\_id]

//만약 DB에 data\_id가 없다면

print("삭제할 데이터가 없습니다.")

}

**9.2.3 Class DB::searchDatabase()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:searchDatabase(data\_id)

기능: DB 클래스에서 기존 database를 조회한다.

매개변수: data\_id= 약품 database id

Return Value:0, else 에러코드 (“존재하지 않는 data\_id 입니다.”)

-----------------------------------------------------------\*/

def :searchDatabase(data\_id): //조회할 data\_id 입력

{

//DB 확인

//만약 DB에 id가 있다면 -> 해당 정보를 출력

get(‘id’)== data\_id;

//만약 DB에 data\_id가 없다면

print("존재하지 않는 data\_id 입니다.")

}

**9.2.4 Class DB::updateDatabase()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:updateDatabase(data\_id)

기능: DB 클래스에서 database 정보를 수정한다.

매개변수: id= data\_id

Return Value:0, else 에러코드 (“존재하지 않는 ID 입니다. 올바른 ID를 입력해주세요.”)

-----------------------------------------------------------\*/

def updateDatabase(id):

{

//수정할 id를 입력

//사용자 DB 확인

//만약 DB에 id가 있다면 -> 어떤 정보를 수정할지 입력 받기

print("1. 데이터 변경")

print("2. 사용자 추가")

print("3. 사용자 삭제")

choice = input("선택: ")

If choice == 1 :

{

//1번을 입력받을 시(데이터 변경)

//새로운 값 입력

INSERT INTO DB[data\_id] ("새로운 데이터 입력: ")

print("데이터 변경 완료")

}

elIf choice == 2 : //2번을 입력받을 시(data 추가)

{

//data\_id 입력

def addDatabase(id):

//DB 확인

//만약 DB에 data\_id가 없다면 -> 새로운 database 추가

create database[data\_id]

//만약 DB에 data\_id가 있다면

print("이미 존재하는ID입니다.")

}

elIf choice == 3 ://3번을 입력받을 시(data 삭제)

{

//data\_id 입력

def deleteDatabase(id):

//DB 확인

//만약 DB에 data\_id가 있다면 -> database 삭제

del database[data\_id]

//만약 DB에 data\_id가 없다면

print("삭제할 데이터가 없습니다.")

}

//만약 사용자 DB에 id가 없다면

print("존재하지 않는 ID입니다. 올바른 ID를 입력해주세요.")

}

**9.3 Class 데이터전처리 메소드 알고리즘 명세**

**9.3.1 Class 데이터전처리::upload()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:upload(xls\_file)

기능: 데이터전처리 클래스에서 xls 파일을 변환 시스템에 업로드 해 csv파일로 변환한다.

매개변수: file\_id = upload 할 파일

Return Value:0, else 에러코드 (“업로드실패!”)

-----------------------------------------------------------\*/

Def upload(file\_id):

{

//xls파일을 csv파일로 변환

output\_file = xls\_file.replace(".xls", ".csv")

convert\_xls\_to\_csv(input\_file, output\_file)

//변환된 csv 파일 DB 업로드

Database.append(file\_id)

//변환된 csv 파일 DB 업로드 성공시

printf(“업로드성공!”)

//변환된 csv 파일 DB 업로드 실패시

printf(“업로드실패!”)

}

**9.4 Class 로그인&회원가입 메소드 알고리즘 명세**

**9.4.1 Class 로그인&회원가입::login()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:login(id, pw)

기능: 로그인&회원가입클래스에서 사용자의 로그인 처리

매개변수: id = 회원 id

pw = 회원 pw

입력: input("아이디를 입력하세요: ") = 사용자 id 입력

input("비밀번호를 입력하세요: ") = 사용자 pw 입력

Return value = print("로그인 성공"), else 에러코드 ("로그인 실패")

-----------------------------------------------------------\*/

def login(id,pw):

{

// 사용자 아이디 입력

user\_id = input("아이디를 입력하세요: ")

// 사용자 비밀번호 입력

user\_pw = input"비밀번호를 입력하세요: ")

// users 딕셔너리에 있는 경우

if user\_id in users and users[user\_id] == user\_pw:

print("로그인 성공")

// users 딕셔너리에 있는 경우

else:

print("로그인 실패")

}

**9.4.2 Class 로그인&회원가입::logout()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:logout(id, pw)

기능: 로그인&회원가입클래스에서 사용자의 로그아웃 처리

매개변수: id = 회원 id

pw = 회원 pw

변수: is\_logged\_in = 로그인 상태 확인

Return value = True print("로그아웃 성공"), else 에러코드 ("로그아웃 실패")

-----------------------------------------------------------\*/

def logout(id,pw):

{

//로그인 상태 확인

is\_logged\_in = False

if is\_logged\_in == True: // 로그인 상태인 경우

print("로그아웃 성공")

is\_logged\_in = False

else: // 로그인 상태가 아닌 경우

print("로그아웃 실패")

}

**9.4.3 Class 로그인&회원가입::makeid()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:makeid(id)

기능: 로그인&회원가입클래스에서 사용자의 아이디 생성 처리

매개변수: id = 회원 id

입력: input("아이디를 입력하세요: ") = 사용자가 생성하려는 아이디 입력

Return value = 0 print("이미 존재하는 아이디입니다."), else ("아이디 입력 완료")

-----------------------------------------------------------\*/

def makeid(id):

{

//사용자가 생성하려는 아이디 입력

user\_id = input("아이디를 입력하세요: ")

if user\_id in users: //users 딕셔너리에 있는 경우

print("이미 존재하는 아이디입니다.")

else: //users 딕셔너리에 없는 경우

users[user\_id] = user\_id

print("아이디 입력 완료")

}

**9.4.4 Class 로그인&회원가입::makePassword()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:makePassword(id, pw)

기능: 로그인&회원가입클래스에서 사용자의 비밀번호 생성 처리

매개변수: id = 회원 id

pw = 회원 pw

입력: input("아이디를 입력하세요: ") = 사용자가 생성한 아이디 입력

input("비밀번호를 입력하세요: ") = 사용자가 생성할 비밀번호 입력

Return value = 0 print("이미 존재하는 아이디입니다."), else ("회원가입 성공")

-----------------------------------------------------------\*/

def makePassword(id,pw):

{

//생성한 사용자의 id 입력

user\_id = input("아이디를 입력하세요: ")

//생성할 사용자의 pw 입력

user\_pw = input("비밀번호를 입력하세요: ")

if user\_id in users: //users 딕셔너리에 있는 경우

print("이미 존재하는 아이디입니다.")

else: //users 딕셔너리에 없는 경우

users[user\_id] = user\_pw

print("회원가입 성공")

}

**9.5 Class 수요예측모델 메소드 알고리즘 명세**

**9.5.1 Class 수요예측모델::set\_range()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:set\_range()

기능: 수요예측모델 클래스에서 학습범위를 설정한다.

매개변수: 없음

Return Value:0, else 에러코드 (“running failed!”)

-----------------------------------------------------------\*/

set\_range():

{

//날짜를 불러오기

import datetime

//시각화 라이브러리 불러오기

import matplotlib.pyplot as plt

//파이썬 pandas 라이브러리 불러오기

import pandas as pd

//특성(features)의 평균과 표준편차 계산

mean\_features = np.mean(features, axis=0)

    std\_features = np.std(features, axis=0)

range\_multiplier = 3

//학습 범위 설정

//수요예측 시작시점

start = datetime.datetime(2021, 3, 1)

//수요예측 끝나는 시점

end = datetime.datetime(2023, 3, 1)

//학습률 설정

lower\_bound = mean\_features - range\_multiplier \* std\_features

    upper\_bound = mean\_features + range\_multiplier \* std\_features

learning\_rate\_setting = f"Learning Rate: {learning\_rate}"

//목표 설정

target\_setting = f"Target: {target}"

//수요예측 수행

run\_model()

}

**9.5.2 Class 수요예측모델::run\_model()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:run\_model()

기능: 수요예측모델에서 학습범위를 바탕으로 수요예측을 진행한다.

매개변수: 없음

Return Value:수요예측 완료 데이터, else 에러코드 (“예측실패!”)

-----------------------------------------------------------\*/

def run\_model(): //시계열 예측

{

    //set\_range에서 정한 학습범위를 바탕으로 수요예측을 진행한다.

target = learning\_rate \* features[0] + learning\_rate \* features[1] + learning\_rate \* features[2] + bias //수요예측 할 목표를 설정한다.

    pd.read\_csv("data.csv") // csv파일을 읽어온다

pd.to\_datetime(database['Date'], format='%Y-%m-%d') // date 포맷 설정

x\_train = forecast\_data[forecast\_data['date']] < start //시작 일자 설정

x\_valid = forecast\_data[forecast\_data['date']] >= end // 종료 일자 설정

//예측할 열(총 생산량) 필터링

mp = x\_train[['date','totalproduce']]

mp\_v = x\_valid[['date','totalproduce']]

//date를 인덱스로 생성

mp.set\_index('date',inplcae=True)

mp\_v.set\_index('date',inplcae=True)

//dateframe 생성

index\_1\_year = pd.date\_range(nyc.index[-1],freq='W',periods = 24, tz= None )

model\_arima = ARIMA(mp , order=(1,0,2))

model\_arima\_fit = model\_arima.fit(disp=-1)

//예측한 값들을 수요예측 결과DB에 저장

addDatabase(model\_arima\_fit)

}

**9.5.3 Class 수요예측모델::settarget()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:settarget()

기능: 수요예측모델 클래스에서 목표를 설정한다.

매개변수: target= 설정목표

Return Value:0

-----------------------------------------------------------\*/

수요예측모델의 목표를 설정한다

def settarget(target):

{

pre\_target = target

}

**9.5.4 Class 수요예측모델::addResult()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:addResult(id)

기능: 수요예측모델 클래스에서 새로운 결과를 추가한다.

매개변수: id= Result\_id

Return Value:0, else 에러코드 (“이미 존재하는 ID 입니다.”)

-----------------------------------------------------------\*/

def deleteResult(id): //삭제할 Result\_id 입력

{

//수요예측결과 DB 확인

//만약 수요예측결과DB에 Result\_id가 없다면 -> Result 추가

add Result\_DB[Result\_id]

//만약 수요예측결과DB에 Result\_id가 있다면

print("이미 존재하는 ID 입니다.")

}

**9.5.5 Class 수요예측모델::deleteResult()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:deleteResult(id)

기능: 수요예측모델 클래스에서 기존 결과를 삭제한다.

매개변수: id= Result\_id

Return Value = 0, else 에러코드 (“존재하지 않는 ID 입니다.”)

-----------------------------------------------------------\*/

def deleteResult(id): //삭제할 Result\_id 입력

{

//수요예측결과 DB 확인

//만약 수요예측결과DB에 Result\_id가 있다면 -> Result 삭제

del Result\_DB[Result\_id]

//만약 수요예측결과DB에 Result\_id가 없다면

print("존재하지 않는 ID 입니다.")

}

**9.5.6 Class 수요예측모델:: addfeature()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:addfeature(feature)

기능: 수요예측모델에 사용할 feature를 추가

매개변수: feature = 수요예측모델에 사용

Return Value:0, else 에러코드 (“존재하지 않는 feature 입니다.”)

-----------------------------------------------------------\*/

def addfeature(feature): //추가할 feature 입력

{

//feature가 없을 경우

if feature not in features:

return "존재하지 않는 feature 입니다."

else:

model.addfeature(feature)

return 0

}

**9.5.7 Class 수요예측모델::deletefeature()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:deletefeature()

기능: 수요예측모델에 사용하지 않을 feature를 삭제

매개변수: feature = 삭제할 매개변수

Return Value = print("feature 삭제 완료") else 에러코드 (“존재하지 않는 feature 입니다.”)

-----------------------------------------------------------\*/

def deletefeature(feature):// 삭제할 매개변수 입력

{

//feature가 없을 경우

if feature not in model.features:

print ("존재하지 않는 feature 입니다.")

else:

model.features.remove(feature)

print( "feature 삭제 완료")

}

**9.5.8 Class 수요예측모델::setLearningrate()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:setLearningrate(learningrate)

기능: 이전에 세팅되었던 학습률 변경

매개변수: learningrate = 학습률

Return Value = print("학습률 변경 완료") else 에러코드 (“학습률 설정에 실패했습니다”)

-----------------------------------------------------------\*/

def setLearningrate(learningrate):

{

if learningrate <=0:

print("학습률 설정에 실패했습니다")

else:

model.learningrate=learningrate

print("학습률 변경 완료)

}

**9.6 Class 알림 메소드 알고리즘 명세**

**9.6.1 Class 알림::alarm\_ON()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:alarm\_ON()

기능: 알림클래스에서 알람 on으로 설정

매개변수: 없음

Return Value : alram\_flag = 1 == 알람 on

-----------------------------------------------------------\*/

def alarm\_ON():

{

alram\_flag = 1 // alarm\_flag 1로 세팅

}

**9.6.2 Class 알림::alarm\_OFF()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:alarm\_OFF()

기능: 알림클래스에서 알람 off으로 설정

매개변수: 없음

Return Value : alram\_flag = 0 == 알람 off

-----------------------------------------------------------\*/

def alarm\_OFF():

{

alram\_flag = 0 #alarm\_flag가 0으로 세팅

}

**9.6.3 Class 알림::sendflag()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:sendflag(alram\_flag)

기능: 알림클래스에서 사용자에게 flag 보냄

매개 변수: alram\_flag = 알람확인 flag

Return Value :

alram flag == 1, message("수요 예측 완료") 보냄

alram flag == 0, message("수요 예측 진행 중") 보냄

-----------------------------------------------------------\*/

def sendflag(alram\_flag):

{

if alram\_flag == 1:

message = "수요 예측 완료"

updateUser(data\_id) //alarm\_flag가 1이면 사용자 db를 업데이트하여 수요예측이 완료되었는지 확인

elif alram\_flag == 0:

message = "수요 예측 진행 중"

}

**9.7 Class 시각화시스템 메소드 알고리즘 명세**

**9.7.1 Class 시각화시스템::buildgraph()알고리즘 명세**

/\*-----------------------------------------------------------

인터페이스:buildgraph(data)

기능: 시각화시스템의 클래스에서 수요예측된 결과를 시각화해서 보여줌

매개변수: data = 수요예측 결과

Return value = 0 print("데이터가 존재하지않음")

x = x축(시간)

y = y축(수요예측결과)

title = 그래프의 제목

xlabel = x축라벨

ylabel = y축라벨

-----------------------------------------------------------\*/

def buildgraph(data) :

{

if not data:

return "데이터가 존재하지않음."

// x축에는 시간을, y축에는 수요예측결과를 표시

x = data['time']

y = data['demand\_prediction']

// 그래프의 제목, x축, y축의 라벨을 설정

plt.title('Demand Prediction Graph')

plt.xlabel('Time')

plt.ylabel('Demand Prediction')

.

plt.plot(x, y)

// 그래프를 보여준다.

plt.show()

}

**10. 요구사항 검토 결과**

프로토타입을 기반으로 요구사항 결과를 검토해야 함.

**11. 결론**

코로나와 같은 여러 감염질환이 발병하는 오늘날의 사회는 약을 중요하게 여겨야한다. 제약회사의 약품제조 수요를 예측하여, 분기별 다음 약품 생산량을 미리 생산할 수 있으면, 약품의 재고를 부족하지 않게 관리를 해줄 수 있다. 이는 사람들에게 안정적으로 약품을 공급할 수 있게 해 줄 것이다.

**부록**

**참고문헌**

**<논문>**

윤동민. "편의점 발주 업무의 자동화를 위한 머신러닝 기반의 적정 발주량 예측 시스템." 국내석사학위논문 동의대학교 대학원, 2018. 부산

 김정아. "수요예측을 활용한 실시간 재고관리시스템 설계 및 구현에 관한 연구." 국내석사학위논문 성균관대학교 일반대학원, 2019. 서울

**<상용 제품>**

(주)이카운트 : 쇼핑몰 관리 , 온라인 발주 , 급여조회, 카드내역 조회 등 웹 기반 기업 재고관리 프로그램

 재고관리 엑셀자동화 프로그램 V46 : 제품의 재고수량 및 금액 등의 현황을 파악하고 부족수량을 확인하여 발주 관리를 할 수 있는 자동화프로그램 엑셀 파일