Part1. 소프트웨어 설계

Chapter1. 요구사항 확인

S1. 현행 시스템 분석

* 구성, 기능, 인터페이스 🡪 아키텍처, 소프트웨어 구성 🡪 하드웨어 및 네트워크
* 현행 시스템 아키텍처 구성도 : 최상위 수준에서 그림으로 표현
* CPU 용량산정 🡪 메모리 용량산정 🡪 디스크 용량산정 🡪 오프소스
* 운영체제 종류 및 특징 : 신뢰도, 성능, 기술 지원, 주변기기, 구축 비용
  + Windows : 중소규모, 개인용
  + UNIX : 대용량 처리, 안정성 요구되는 서버
  + Linux : 중대규모 서버 / 하드웨어 및 소프트웨어 소유 비용이 가장 적다.
  + IOS :
  + Android :
* 네트워크 구성요소
  + 리피터 : 물리층 기능을 수행 / 신호를 증폭기의 일종
  + 허브 : 물리적인 길이를 확장 / LAN에 접속시키는 네트워크 장치
  + 브리지 : 네트워크를 작은 세그먼트로 분할 / 물리계층 🡨🡪데이터링크 연결 / 동종의 LAN을 연결하는 장비 / 물리 계층이 다를 때 사용
  + 스위치 : 링크 계층 네트워크르 결합 / 망을 확장 / 네트워크를 분할
  + 라우터 : 다중 상호 연결망에서 패킷 중계
  + 게이트웨이 : 프로토콜 변환기
* 백본 : 기간망 / 여러 소형 네트워크를 묶어 대규모 파이프라인을 통해 연결
* 미들웨어 : 운영체제와 소프트웨어 애플리케이션 사이의 소프트웨어

S2. 요구사항 확인

* 도출 🡪 분석 🡪 명세 🡪 확인
* UML : 시스템 개발 과정에서 개발자 사이의 의사소통을 위한 모델링 언어
* 애자일 : 경량 개발 방법론 총칭 / 변화에 유연하고 신속, 적응적인 개발론
  + 익스트림 프로그래밍 : TDD(테스트 우선 개발)을 특징 / 고객과 2주 반복개발
  + 스크럼 : 30일마다 동작 가능한 스프린트 / 짧은 시간의 개발을 위해
  + 크리스털 패밀리 : 소규모
  + FDD : 2주정도 반복 개발
  + ASD : 사용자와 고객이 설계에 참가 / 혼란을 대전제로
  + 익스트림 모델링 : UML을 이용한 모델링 중심 방법론
* 기타개발 방법론
  + 폭포수 모델 : 단계적으로 하향식 생명주기 모델 / 가장 고전적
  + 나선형 모델 : 대규모 개발에 적당 / 점증적으로 개발(4단계)

S3. 분석 모델 확인

* 유스케이스 모델 검증 🡪 개념 수준 분석 클래스 검증 🡪 분석 클래스 검증
* 유스케이스 모델 검증 대상
  + 액터 / 유스케이스 / 유스케이스 명세서
* 분석 모델 검토 사항
  + 성능 및 용량 / 시스템 간 상호 운용성 / 시장 성숙도 및 트렌드 / 기술적 위험

Chapter2. 화면설계

S1. UI 요구사항 확인

* UI 설계 원칙
  + 직관성 / 유효성 / 학습성 / 유연성
* 화면구조 : Top, Left, Contents
* 품질 요구사항(ISO/IEC 9126)
  + 기능성 / 신뢰성 / 사용성 / 효율성 / 유지보수성 / 이식성
* 스토리보드 : 디자이너와 개발자가 최종적으로 참고하는 산출문서
* UI 유저인터페이스 : 유저와 컴퓨터 상호간의 소통을 도와주는거
* CRUD : Create / Read / Update / Delete

S2. UI 설계

* WIMP : Window / Icon / Menu / Pointer
* UI/UX 의 구성요소
  + 실용적 / 쾌락적 / 아름다움 / 이로움(목적,의도,타당성)
* 더블 다이아몬드 모델
  + 탐색 🡪 정의 🡪 개발 🡪 산출
* AIO (Activity, Interest, Opinion) : 라이프스타일 분석
* 다이어리 스터디 : 사용자가 비디오 등으로 자신을 촬영하여 기록
* 포커스 그룹 인터뷰 : 소수의 그룹이 개방된 토의 진행 / 사회자가 진행
* 제품분석
  + 3C 분석 : Corporation, Customer, Competitor
  + SWOT 분석 : 강점/약점/기회/위협
  + 4P : Product / Price / Place / Promotion
  + 포지셔닝 : 그래프에 표시하는거
  + 4C : Contents / Communication / Commerce / Community

Chapter3. 애플리케이션 설계

S1. 공통 모듈 설계

* 정확성 / 명확성(한가지로 해석) / 완전성 / 일관성 / 추적성
* 모듈의 재사용
  + 함수 & 객체 / 컴포넌트 / 애플리케이션 재사용
* 응집도 : 기능적 >>>>>>>>>>우연적
* 결합도 : 자료 >>>>>>>>>>내용
* Fan In : 자신을 사용하는 모듈 수
* Fan Out : 자신이 호출하는 모듈 수
* 소프트웨어 아키텍처 : 소프트웨어의 청사진 / 공간을 소모하는 방법에 대한 기술
* 오류 예측 및 대응 방안
  + 액티브-액티브 : 모든 서버가 서비스 제공, 장애 발생 후 다른 서버
  + 액티브-스탠바이 : 평상시에는 하나만 서비스, 장애 발생시 이관

S2. 객체지향 설계

* 스몰토크 : 객체지향언어의 시초
* 에이다 : 국방성에서 개발 / 상속X
* C, C++ : AT&T에서 개발
* Java
* Object-C
* 객체지향의 특징
  + 추상화 / 캡슐화 / 정보은닉 / 자료추상화 / 상속 / 다중상속 / 다형성 / 동적바인딩
* 객체지향 개발 5대 원리
  + SRP : 단일 책임의 원칙
  + OCP : 개방폐쇄의 원칙 / 확장 ㅇ, 변경 X
  + LSP : 리스코브 치환의 원칙 / 서브클래스가 기반타입의 규약을 지켜야함
  + ISP : 인터페이스 분리의 원칙 / 사용하지 않는 인터페이스는 구현하지 않아야
  + DIP : 의존성 역전의 원칙 / 하위 클래스가 상위꺼를 변경하려고 하면 안댐
* 메소드 : 한 객체의 서브루틴 형태로 객체의 속성을 조작
* 메시지 : 클래스로부터 생성된 객체를 사용하는 방법

Chapter4. 인터페이스 설계

S1. 인터페이스 요구사항 확인

* 인터페이스 요구사항 검증 항목
  + 인터페이스 목록 / 과제 범위 / 인터페이스 명세서 / 기술 요건
* 요구사항 관리
  + 요구사항 협상 / 기준선 / 변경관리 / 확인
* 요구사항 개발
  + 요구사항 추출 / 분석 / 명세화 / 확인(검증)
* 비기능적 요구사항 : 성능, 사용의 용이성, 신뢰도, 보안성, 운용상의 제약, 안정성 등
* 요구사항 검토방법
  + 동료 검토 : 2~3명이 리뷰의 형태로 하는거
  + 워크쓰루 : 비정형 검토회의 / 오류를 조기에 검출할고 짧게 하는거
  + 익스펙션 : 다른 전문가 팀이 검사하는거\
  + 프로토 타이핑 : 약식 개발
  + 테스트 설계 : 테스트 케이스 생성 🡪 현실성 있는지
* 요구공학 : 시스템 요구사항 문서를 생성,관리 하기 위해 구조화된 활동의 집합

S2. 인터페이스 대상 식별

* 시스템 아키텍쳐 : 시스템의 구조, 행위, 뷰를 정의하는 개념적 모형
* 인터페이스 시스템
  + 송신시스템 / 수신 시스템 / 중계 서버

S3. 인터페이스 상세 설계

* 데이터베이스 산출물
  + 개체정의서 / 테이블 정의서 / 코드 정의서
* 연계 방식
  + 실시간 동기
  + 실시간 비동기
  + 배치 File/ DB
* 표준 도메인 : 속성값의 범위를 정해 놓고 기본적으로 숫자, 문자열, 일시로 정의

Part2. 소프트웨어 개발

Chapter 데이터 입출력

S1. 논리 데이터 저장소 확인

* 자료구조 : 컴퓨터에서 처리할 자료를 효율적으로 관리하고 구조화시기는 영역
  + 선형구조 : 배열, 큐, 스택, 데크, 연결리스트
  + 비선형구조 : 트리 , 망
  + 정적 자료구조 : 배열, 레코드, 책자
  + 동적 자료구조 : 트리, 인덱스, 스택
* 데이터 모델링 절차
  + 개념데이터 모델링 🡪 논리 데이터 🡪 물리데이터

S2. 물리 데이터 저장소 설계

* 반정규화 : 성능에 중점을 두어 정규화하는거
* 객체 관계 매핑(ORM) : 데이터베이스와 객체지향 프로그래밍 언어간의 변환
  + 독립적으로 작성되어 있음 🡪 개객체 재활용 가능
* 트랜잭션 : 정보를 기록한 파일에 대해 수정할려고 가져오는 행위

S3. 데이터 조작 프로시저 작성

* 프로시저 : SQL을 이용하여 만들어진 데이터를 조작하는 트랜잭션 언어(자동)
* SQL 분류
  + 데이터 정의어(DDL) : Create, drop, rename, alter 등
  + 데이터 조작어(DML) : insert, update, delete
  + 데이터 제어어(DCL) : role, grant, revoke
* 트랜잭션 제어어(TCL) : DML 작업단위 제어 / role, grant, select
* Commit : 물리적인 디스크에 반영 / 이전데이터 복구X / 반영결과 열람가능
* Rollback : 메모리상의 버퍼에만 영향을 미침 🡪 복구가능 / 잠금 풀림

S4. 데이터 조작 프로시저 최적화

* SQL 성능 최적화
  + TKPROF 활용
* 소스코드 인스펙션 : 코드 잘 짯는지 확인하는거 / 수시로 진행

Chapter2. 통합 구현

S1. 모듈 구현

* 단위 모듈 테스트
  + 화이트박스 테스트 / 블랙박스 테스트
* 테스트 목적
  + 회복 / 안전 / 강도 / 성능 / 구조 / 회귀 / 병행
* 스텁 : 사용자화면이 없는 경우 사용 / 하향식

S2. 통합 구현 관리

* IDE(통합 개발 환경) : 개발에 필요한 모든 작업을 하나의 인터페이스로 통합 제공
  + 이클립스 / 라자루스(파스칼) / 비주얼 스튜디오 / 안스 / 엑스코드 / IDEA
* 협업도구 종류
  + 프로젝트 일정 : 트렐로, 레드마인, 지라, 태스크월드, 구글 캘린더
  + 마인트 맵핑 : 마인드마이스터
  + 디자인 공유 : 레드 펜, 스케치, 제플린
  + 아이디어 고융 : 에버노트, 인버전
  + 소스 공유 : 깃헙
  + 문서 공유 : 구글드라이브, 슬라이드

Chapter3. 제품 소프트웨어 패키징

S1. 제품 소프트웨어 패키징

* 제품 소프트웨어 패키징 : 고객에게 전달하기 위한 형태 / 매뉴얼 작성
* 기능식별 🡪 모듈화 🡪 빌드진행 🡪 사용자환경분석 🡪 패키징 적용시험 🡪 개선
* 제품 패키징 릴리즈 노트 : 고객과 잘 정리된 릴리즈 정보를 공유하는 문서
  + 개발팀에서 직접 작성 / 현재시재로 작성
* 애플리케이션 성능 관리(APM)
* 디지털 권리 관리(DRM)
* 패키징 도구 구성 요소
  + 암호화, 키 관리, 암호화 차일 생성, 식별 기술, 저작권 표현, 정책관리, 인증

S2. 제품 소프트웨어 매뉴얼

* 제품 소프트웨어 매뉴얼 개요 : 설명서와 안내서(설치)
  + 제품 SW개요, 설치 파일관리, 설치 아이콘, 삭제방법, 관련 추가 정보
* ISO/IEC 9126 : 소프트웨어 품질 특성과 척도에 관한 지침
* ISO 9001 : 품증 보증 모델

S3. 소프트웨어 버전 관리 도구

* Add 🡪 Check-out 🡪 Commit 🡪 Update 🡪 Diff
* 버전관리 도구 방식
  + 공유 폴더 방식(RCS, SCCS) / 클라이언트서버 방식(CVS,SVN) / 분산 저장소 방식
* 빌드 자동화 도구
  + Makefile(C,C++) / Ant, Maven (Java) / Gradle(안스)

Chapter4. 애플리케이셔 테스트 관리

S1. 애플리케이션 테스트 케이스 설계

* 상용 소프트웨어
  + 산업 범용 소프트웨어 : 시스템 소프트웨어, 미들웨어, 응용 소프트웨어
  + 서비스 제공 소프트웨어 : 신규개발SW, 프로세스 개선 SW 이런거
* 소프트웨어 테스트
  + 완벽한 테스트X / 개발 초기에 시작 / 결함 집중 / 정황에 의존 / 오류-부재궤변
* 목적에 따른 테스트 : 회복 / 안전 / 강도 / 성능 / 구조 / 회귀 / 병행
* 종류에 따른 테스트 : 명세 시간 / 구조 기반 / 경험 기반
* 테스트 레벨
  + 단위테스트 🡪 통합 테스트 🡪 시스템 테스트 🡪 인수 테스트
  + 모듈,컴포넌트 🡪 인터페이스🡪 전체 테스트 🡪 요구사항을 총족시키는지

S2. 애플리케이션 통합 테스트

* 통합 테스트 방식
  + 점증적인 방식 : 모든 컴포넌트를 사전에 통합하여 한꺼번에 테스트
  + 비점증적인 방식 : 상향식 / 하향식(Stub : 타 모듈의 기능을 단순히 기능) 개발

S3. 알고리즘

* 조건
  + 입력 : 0이상의 제공이 되어야함
  + 출력 : 1개 이상의 결과를 가짐
  + 유한성 : 유한한 횟수를 거쳐야함
  + 효과성 : 사람이 종이와 연필을 이용할 정도로 단순해야 함
* 성능 분석 지표
  + 처리량 : 주어진 시간에 처리할 수 있는 트랜잭션 수
  + 응답 시간 : 입력이 끝난 후 응답 출력이 개시 될 때까지의 시간
  + 경과 시간 : 요구를 입력한 시점부터, 결과가 출력이 완료 될 때 까지
  + 자원 사용률 : 트랜잭선을 처리하는 동안 사용하는 사용량

Chapter5. 인터페이스 구현

S1. 인터페이스 설계 확인

* 인터페이스 기능 : 외/내부 모듈 간에 공통 기능
* 외/내부 모듈 연계 방법
  + EAI : 서로 다른 플랫폼 및 애플리케이션 간의 연결
    - Point-toPoint
    - Hub & Spoke : 모든 데이터 전송 보장
    - Message Bus(ESB 방식)
    - Hybrid
  + ESB : 서비스 중심으로 통합을 지향 / Bus 중심

S2. 인터페이스 기능 구현

* 모듈 간 세부 설계서 확인(컴포넌트/인터페이스 명세서) 🡪 인터페이스 기능 구현 정의
* 인터페이스 구현
  + 데이터 통신을 통한 구현 : JSON(속성-값) AJAX에서 많이 사용 / XML
  + 인터페이스 테이블을 이용 : DB를 이용해
* 시큐어 코딩 : 개발 과정 중에 지켜야 할 보안 활동
* 인터페이스 보안
  + 스니핑 : 중간에 감청하여 기밀성 훼손
  + 스푸핑 : 수동적 해킹 공격 방법

S3. 인터페이스 구현 검증

Part3. 데이터베이스 구축

Chapter1. SQL 응용

S1. 절차형 SQL 작성

* 트리거 : 절차형SQL임 / DML의 이벤트가 발생될 때 작동하는 프로그램 / DB에 저장
  + Begin ~ End 안의 명령문을 자동으로 수행함
  + 실행 시기 : Before(시작전) , After(시작 후)
  + 실행 횟수/방법 : Row(각 줄마다), Statement(SQL문장에 한번만 실행)
  + TCL(DCL) 사용 불가임 🡪 트랜잭션 제어 불가
* 이벤트 : 특정시간에 특정한 쿼리, 프로시저, 함수 등을 실행시키는 기능
  + DML 이벤트 : insert / update, delete
  + DDL 이벤트 : create / alter / drop
  + Database 이벤트 :
* 트랜잭션 : 업무처리를 위한 하나의 논리적인 작업 단위=(SQL의 묶음)
  + 원자성 / 일관성 / 독립성 / 영속성(영구적)
* MtBatis : SQL 맵핑 기반 오픈 소스 프레임워크임
  + SQL 쿼리르 별도의 XML 파일로 분리 후 맵핑을 통해 SQL 실행
  + < 로 시작 하여 </ 로 종료
  + 코드 단순화가능 / SQL그대로 사용 가능 / Spring 프레임워크랑 같이 사용
* 사용자 정의 함수 : 반환되는 값을 조회, 삽입, 수정에 이용함(Void)

S2. 응용SQL 작성

* DML 유형
  + Insert (into)/ select (from)/ update (set)/ delete (from)
* DCL 유형 : 데베에 접근하거나 객체에 권한을 주는 등의 역할을 함
  + 사용자 권한 : DCL 🡪 Grant(권한 부여) +To / Revoke(권한 회수) + To
  + 트랜잭션 : TCL 🡪 Commit(물리적으로 디스크 저장) / Rollback / Checkpoint

Chapter2. SQL 활용

S1. 기본 SQL 작성

* DDL : 데이터 정의어
  + Create / Alter / Drop(데이터 오브젝트 삭제) / Truncate(데이터 오브젝트 내용 삭제)
* 트랜잭션 : 논리적 연산 단위
  + 원자성(Atomicity) : 결과는 모두 반영되거나 모두 취소되어야 함
  + 일관성(Consistency) : 고정요소는 트랜잭션 실행 전후가 같아야 함
  + 격리/독립성(Isolation) : 트랜잭션 실행 중에는 다른 연산 침범 금지
  + 영속성(Durability) : 변화된 상태는 지속적으로 유지될 수 있어야 함
  + 상태 : 활동 / 부분 완료 / 완료 / 장애,실패 / 철회
* 데이터 사전 : 데이터베이스의 데이터를 제외한 모든 정보
  + 메타데이터 : 데이터 사전을 구현하는법, 관리하는 법

S2. 고급 SQL 작성

* 뷰의 단점 : 인덱스 가질 수 없음 / 정의 불가(삭제하고 다시) / 내용 수정 불가
* 인덱스 : B-tree 알고리즘 사용 / 인덱스를 사용하지 않으면 Full-Scan함
* 집합 연산자
  + Union 합집합 / Union All / Intersection 교집합 / Except 차집합
* 논리적 조인
  + Inner join(내부조인) / Outer join(외부조인)
* 물리적 조인
  + Nested Loop Join : 랜덤으로 다 조인함
  + Merge Join : 연결고리가 없어도 됨 / 크기가 차이가 많이 나면 비효율적
  + Hash Join : 해시값을 이용한 조인

Chapter3. 논리 데이터 베이스 설계

S1. 관계 데이터베이스 모델

* 스키마 : 릴레이션이 어떻게 구성되어 있는지 정보에 대한 기본적인 구조 정의
* 무결성 제약조건
  + 도메인 무결성 제약조건 : 각 속성의 도메인에 지정된 값만을 가져야함
  + 개체 무결성 제약조건 : =기본키 제약조건 : 기본키는 Null X, 하나의 값만
  + 참조 무결성 제약조건 : 외래키 제약조건
* 시스템 카탈로그 : 데이터베이스에 관한 모든 정보를 메타 데이터 형태로 가지는거

S2. 데이터 모델링 및 설계

* 데이터 모델링 : 개념을 단순화/추상화시켜 DB화 하는 과정
  + 개념적 데이터 모델링 : 요구사항 수집/분석 🡪 추상화, ERD 만드는 과정까지
  + 논리적 데이터 모델링 : 정보의 구조와 규칙을 명확하게 표현하는 기법
    - ERD를 통해 DBMS에 맞게 맵핑하여 속성 추출/정규화/표준화
    - 데이터 모델링이 최종적으로 완료된 상태
  + 물리적 데이터 모델링 : 논리적 모델을 물리적 구조를 정의/구현
* 키의 종류
  + 기본키(Primary Key) : 후보키 중에서 특별히 선정된 키
  + 후보키(Candidate Key) : 유일하게 구분할 수 있는 최소한의 속성들의 집합
  + 대체키(Alternate Key) : 기본키가 아닌 후보키
  + 슈퍼키(Super Key) : 유일성은 만족/ 최소성은 만족X
  + 외래키(Foreign Key)
  + 슈퍼키 > 후보키 > 대체키 > 기본키
* 데이버베이스 정규화
  + 이상현상
    - 갱신 이상 : 갱신 시 데이터의 불일치
    - 삽입 이상 : 불필요한 정보를 함께 저장하지 않고서는 정보 저장 불가
    - 삭제 이상 : 필요한 정보를 삭제하지 않고서는 정보삭제 불가

Chapter4. 데이터베이스 설계

S1. 물리요소 조사 분석

* 스토리지 : 대용량 데이터를 저장하기 위해 구성된 시스템
  + DAS : 서버안에 배치하거나 서버에 직접 연결된 스토리지 하위 시스템
  + NAS : 서버와 저장장치를 네트워크로 연결하는 방식
  + SAN : 위에 두개 합친거
* 분산 데이터베이스
  + 테이블 위치 분산 / 테이블 분할 분산 / 테이블 복제 분산 / 요약 분산
* 데이터 이중화 구성
  + Active-Active : 동시에 서비스 구동
  + Active-Standby : 한쪽만 구동
* 접근 제어
  + 게이트웨이 방식 : DBMS에 접속하는 별도 통로를 설치/접근
  + 스니핑 방식 : 패킷을 복사하여 DB접근 제어 서버에 전달
  + 에이전트 방식 : DB서버에 접근제어를 설치

S2. 데이터베이스 물리속성 설계

* 파티셔닝 : 큰 테이블이나 인덱스를 작은 단위로 물리적으로 분할
  + Range 파티셔닝 / List / Composite / Hash
  + Horizontal 수평 방법 / Vertical 수직 방법
  + Join 할때 비용을 증가시킴 / 데이터 검색 성능 증가
* 클러스터링 : 저장공간을 가지고 있는 하나의 오브젝트=클러스터
  + 범위를 정해서 모아 같은 위치에 저장하는거 🡪 수정 성능 감소시킴
* 백업의 방식
  + 핫 백업 : 온라인 상태를 유지한 채 백업
  + 콜드 백업 : DB서버를 중지한 후 데이터 백업
  + 물리 백업 : 파일 자체를 그대로 백업 🡪 문제 발생 파악/검토 어려움
  + 논리 백업 : 각 SQL문 등으로 저장
* 데이터 지역화 : 데이터를 효율적으로 저장하는 법 (물리 DB)
  + 시간 지역성 : for / while 같은 반복문 - 조건 변수를 다시 사용할 가능성이 높음
  + 공간 지역성 : A[0], A[1]처럼 연속배열 됐을 때 사용할 가능성이 높음

S3. 물리 데이터베이스 모델링

* 데이터베이스 무결성
  + 개체 / 참조 / 속성 / 키 / 사용자 정의 / 도메인 무결성
  + 보장방법 : 응용 프로그램 / 트리거 / 제약조건
* 반정규화
  + 테이블 병합 / 테이블 분할 / 테이블 추가 / 칼럼 추가

S4. 데이터베이스 정규화

* 정규화의 원칙 : 무손실 표현 / 자료의 중복성 감소 / 분리의 원칙
* 1NF : 릴레이션의 모든 속성값이 원자값을 가질 때
* 2NF : 제1정규형이고 기본키가 아닌 속성이 기본키에 완전 함수 종속일 때
* 3NF : 제2정규형이고 기본키가 아닌 속성이 기본키에 비이행적으로 종속할 때
* BCNF : 3NF에서 조금더 강화한거 🡪 모든 결정자가 후보키 집합에 속한 정규형

S5. 물리 데이터 모델 품질 검토

* 물리 데이터 모델 품질 기준
  + 정확성 / 완전성 / 준거성 / 최신성 / 일관성 / 활용성
* CRUD 분석 : Create / Read / Update / Delete : 전체 업무의 인터페이스 파악 가능
* SQL 성능 튜닝
  + Where 안에서 Where의 조건을 가급적 모두 사용한다.
  + Or 보다 And를 사용하라 / 함수 사용 자제
  + Not보다는 범위로, 범위 지정 시 Between 사용 등등…..

Chapter5. 데이터 전환

S1. 데이터 전환 기술

* 데이터 전환 : 기존의 시스템의 정보를 새로운 정보 시스템에 운용 가능하도록 전환
  + 정제 🡪 추출 🡪 변환 🡪 적재 및 검증
* ETL(Extractoin, Transformation, Loading : 필요한 데이터를 추출/변환/전송/로딩 과정
  + 동시에 진행될 수도 있음.
* 파일 처리 기술
  + SAM(순차 파일) : 물리적 연속 공간에 순차적으로 기록(자기 테이프)
  + ISAM(색인 순차 파일) : 색인을 이용하여 순차적으로(자기 디스크)
  + DAM(직접 파일) : 해싱 함수를 계산해서 물리적 주소 직접 접근
  + VSAM(가상(동적) 기억 접근 방식) : 동적 인덱스 방법을 이용한 색인 순차 파일
  + 역파일 : 찾아보기 기능 있음.

S2. 데이터 전환 수행

* 전환 계획 및 요건정리 🡪 전환 설계 🡪 전환 개발 🡪 전환 테스트 및 검증 🡪 전환
* 데이터 검증 종류
  + 로그 검증 / 기본 항목 검증 / 응용프로그램 검증 / 응용 데이터 검증
* 데이터 검증 방법
  + 추출 검증 / 전송 검증 / 전환 검증(Tool 이용) / 적재 검증 / 통합 검증

S3. 데이터 정제

* 데이터 정제 방법
  + 결측치 처리 : 무시 / 수작업으로 / 전역상수 / 평균값
  + 잡음 있는 데이터 처리 : 구간화 / 단순,복합 회귀값 / 군집화
  + 데이터 불일치 문제 처리

Part4. 프로그래밍 언어활용

Chapter1. 서버프로그램 구현

S1. 개발환경 구축하기

1. 개발 환경 준비

* 소프트웨어 도구
  + 구현도구 : Eclipse, Visual Studio 등
  + 테스트 도구 : xUnit, Spring Test 등
  + 형상 관리 도구 : CVS, Subversion, Git
  + 빌드 도구 : Maven(dependency), Ant(java기반), Gradle(JVM 기반)

1. 개발 환경 준비 수행 순서

* 시스템 환경 분석🡪구현도구 선정🡪빌드도구 선정🡪형상관리도구 선정🡪테스트도구 선정
* 개발 언어 선정 5가지 : 적정성, 효율성, 이식성, 친밀성, 범용성

1. 개발 환경 구축

* 클라이언트, 서버, 시스템 소프트웨어, 개발 소프트웨어 환경

S2. 공통 모듈 구현하기

1. 공통 모듈에 대한 이해 : 재사용이 가능하게 패키지로 제공하는 독립된 모듈
2. 공통 모듈의 재사용

* 재사용의 범위 : 함수와 객체 / 컴포넌트 / 애플캐이션 재사용
* 재사용의 유형 : 편의적 재사용(내부 재사용, 외부 재사용), 계획적 재사용
* 재사용의 사례 : 소프트웨어 라이브러리, 디자인패턴, 프레임워크

1. 소프트웨어 모듈 응집도

* 기능적 -> 순차 -> 통신 -> 절차 -> 시간 -> 논리 -> 우연적

1. 소프트웨어 모듈 결합도

* 자료 -> 스탬프 -> 제어 -> 외부 -> 공통 -> 내용

1. 소프트웨어의 모듈화

* 모듈화의 원리
  + 정보 은폐, 자료 추상화, 모듈의 독립성, 분할과 지배
* 모듈화의 종류
  + 설계측면 : 모듈, 컴포넌트, 서비스
  + 구현측면 : 매크로, 함수, 인라인

1. 공통 모듈 수행하기

* 공통 모듈의 상세 설계를 기반으로 공통 모듈 확인 🡪 공통 모듈 구현

1. 공통 모듈 테스트

* 테스트 케이스 : 요구사항을 준수하는지 검증하기위한 테스트 조건을 명세
  + 식별자, 테스트 항목, 입력 명세, 출력 명세, 환경 설정, 특수 절차 요구, 의존성 기술
* 테스트 프로세스
  + 계획/제어 🡪 분석/설계 🡪 구현/실행 🡪 평가 🡪 완료
  + 단위 테스트 케이스 작성 🡪 내부 검토 🡪 고객에게 승인 🡪도구 설정🡪명세화

S3. 서버프로그램 구현하기

1. 업무 프로세스 확인

* 프로세스 모델 구성 요소 : 고객, 프로세스, 공급자, 입력, 출력
* 프로세스 구성 요소 : 책임자, 맵, Task 정의서, 성과 지표, 조직, 리더십

1. 서버 프로그램 수현

* 프레임워크 특징 : 모듈화, 재사용성, 확장성, 제어의 역흐름
* 소프트웨어 개발 보안 : 취약성의 원인=보안 약점을 최소화 ( 시큐어 코딩)
* API : 응용 프로그램 인터페이스 : 파일 제어, 창 제어, 문자 제어 등 인터페이스 제공
  + Window API : 직접 운영체제에 상호 작용할 수 있도록(dll 사용)
  + Java API : 필요한 클래스들을 미리 구현하여 클래스 계층 구조로 설계
  + 웹 API : 다른 서비스에 요청을 보내고 응답을 받기위해 정의된 명세

1. 서버 프로그램 테스트

* 자신이 개발한 프로그램을 테스팅X / 낚시의 법칙 / 파레토의 법칙 / 계획단계부터 / 주기적으로 / 오류-부재의 궤변 : 사용자의 요구사항을 만족하지 못한다면 X
* 코드 인젝션 : 유효하지 않은 데이터를 실행으로 인한 버그 사용 오류
* 버퍼 오버플로 : 메모리를 다루는데 오류
* HTTP 헤드 인젝션 : 공격자가 헤드 필드에 개행문자 등을 삽입(수동적)

S4. 배치 프로그램

1. 배치 프로그램 구현(Batch Processing)= 일괄처리 ==== 스프링 프레임워크 기반임

* 컴퓨터 프로그램 흐름에 따라 순차적으로 자료를 처리하는 방식
* 필수요소
  + 대용량 데이터
  + 자동화 – 심각한 오류 없이는 사용자 개입없이 동작
  + 견고함 - 유효하지 않은 데이터도 처리
  + 안정성 – 문제가 언제 발생했는지 추적 가능해야 함
  + 성능 – 주어진 시간내에 처리완료, 동시에 동작하는 다른 것들 방해 금지
  + 자체 제공 컴포넌트

1. 배치 프로그램 테스트

* 디버그 : 논리적인 오류를 찾아내는 과정 🡪 찾으면 실행 중단
* 스프링 배치 : 스프링 프레임워크 기반, 자체 제공 컴포넌트, 견고함&안정성
* Quartz 스케쥴러 : 스프링 프레임워크에 플러그인 해서 유연성 제공
  + Schedule, Job, JobDetail, Trigger

Chapter2. 프로그래밍 언어 활용

S1. 기본 문법 활용하기

* 컴퓨터 시스템 구조
  + 중앙 처리 장치(CPU) : 산술논리 연산 장치, 제어 장치, 레지스터
  + 기억 장치
  + 입력 장치
  + 출력 장치
* 필드 : 파일 구성의 최소 단위 (=항목(item))
* 프로그램의 구성요소
  + 자료 구조
  + 알고리즘
* 할당 : 변수에 메모리 공간을 바인딩 하는 적업
  + 정적 할당 : 변수에 메모리 공간이 정적으로 할당 ( 주로 전역 변수 )
  + 스택 기반 할당(자동 할당) : 정적으로 할당되지만 메모리 공간은 실행시간 중에
  + 동적 할당 : 포인터나 참조변수로 참조 가능(C언어 -> malloc)
* 주석처리
  + C, C++, Java : /\* \*/
  + HTML : <!-- -->
  + 비주얼 베이직 : ‘
* 연산 우선순위 : 괄호 🡪 함수 🡪 산술 연산자 🡪 관계 연산자 🡪 논리 연산자

S2. 언어 특성 활용하기

* Fortran : 1954년 개발 / 최초의 고급 언어
* Cobol : 1960년 미국 국방성
* Lisp : 인공지능 관련 언어
* Pascal : 1971~ 1980 정보은닉X
* C : 1972년 유닉스 운영체제 / 절차지향적
* C++ : 상속, 객체지향 언어
* Javascript : 스크립트 언어(컴파일 필요없음)
* Perl : 텍스트 처리용 인터프리터 언어
* Python : 이식성높음, 인터프리터/ 객체지향/ 스크립트 언어
* C# : .net환경
* 컴파일러 : 기계어 번역 도구 / Fortran, C 고급언어를 기계어로 번역
* 인터프리터 : 문장을 하나씩 실행
* 실행하는 방식에 따른 분류
  + 명령형 언어 : =절차형 언어 / Fortran, Cobol, Pascal, C
  + 함수형 언어 : 함수들로 프로그램을 구성하여 호출하는 방식 / Lisp
  + 논리형 언어 : 규칙에 대한 활성화 조건이 만족되면 실행됨 / Prolog
  + 객체지향 언어 : 객체 간의 메시지 통신을 이용 / Java, C++

S3. 라이브러리 활용하기

* 라이브러리 구성 : 도움말, 설치 파일, 샘플 코드
* 표준 라이브러리 : 프로그래밍 언어가 기본적으로 가지고 있는 라이브러리
* 외부 라이브러리 : 별도의 파일을 설피해야함
* Import 패키지명.모듈명 / Input, Output 파라미터만 노출 됨

Chapter3. 응용 SW 기초 기술 활용

S1. 운영체제 기초 활용하기

* 운영체제 : 하드웨어를 제어하는 소프트웨어
  + 기능 : 자원관리, 보호, 하드웨어 인터페이스 제공, 사용자 인터페이스 제공
  + 구성 : 제어프로그램(감시, 작업관리, 데이터 관리) / 처리 프로그램(언어번역,서비스)
  + 유형 : 일괄처리/ 다중 프로그래밍 / 실시간 처리 / 분산 처리 / 클라이언트서버 / P2P / 컴퓨터 환경(그리드, 클라우드, 사물 인터넷)
* 커널 : 컴퓨터가 부팅될 때 주기억 장치에 적재된 후 실행
* 운영체제 핵심 기능
  + 메모리 관리 : 고정 분할 방식/ 가변 분할 방식 / 가상 메모리 기법
    - 페이징 기법 : 고정분할 방식을 이용한 가장 메모리 관리 기법
    - 페이지 테이블 매핑 방식 : 페이지 테이블을 이용함
    - 세그먼테이션 기법 : 세그먼트=Limit 과 주소=Address 이용
    - 캐시 직접 매핑 : 메모리 블록이 캐시로 올라 올 때 항상 같은 위치에
    - 요구 페이징 : 사용자가 요구할 때 해당 페이지를 메모리를 가져오는 것
* 입출력 속도 향상
  + 스풀링 : 디스크의 일부를 버퍼처럼 사용
  + 버퍼링 : 주기억 장치의 일부를 버퍼로 사용(CPU와 I/O 장치간의 속도 차이 극복)
  + DMA방식 : 데이터를 기억 장치 버스를 통해 직접 기억 장치로 전송🡪사이클 스틸
* 배시 쉘 기능
  + Alias : 명령어 단축 기능
  + History : 기존의 사용했던 명령을 위,아래로 선택가능
  + 연산기능
  + Job Control : 작업의 입/출력을 관리, 순서 계획
* 인터럽트 종류(현재 수행중인 프로세스를 중단하거나 외부의 입력으로 인해 중단)
  + SVC(Supervisor Call) : 사용자로부터 생기는 인터럽트
  + 입출력 인터럽드 : 하드웨어적 인터럽트, 입출력 확인/준비/할당/완료 시
  + 외부 인터럽트 : 운영체제 소속이 아닌 외적인 요인으로 부터의 인터럽트
  + 재시간 인터럽트 : 사용자에 의해 메모리에 다시 상주 시킬 때 발생(재부팅)
  + 프로그램 검사 인터럽트 : 오버플로우나 분모가 0일 때, 비정상적인 프로그램
  + 기계 검사 인터럽트 : 시스템의 기계 고장으로 인한 인터럽트
* PCB(프로세스 제어 블록) : 프로세스의 존재를 정의
* CPU 스케쥴링
  + 작업 스케줄링 : 어떤 작업이 자원을 차지 할지를 결정
  + CPU 프로세스 : CPU가 다음 프로세스를 받을 때 준비 완료 프로세스에 할당
  + 선점 스케쥴링 : 실행중이지않은 프로세스가 CPU를 차지하는 기법
  + 비선점 스케쥴링 : CPU를 할당 받으면 다른 프로세스가 점유 할 수 없음
  + 우선순위 스케쥴링 : FIFO이고 비선점 알고리즘임
  + FCFS 스케쥴링 : 가장 간단한 비선점 알고리즘(먼저오면 먼저온대로)
  + RR 스케쥴링 : 대화식 사용자를 위한 시분할 / 선점 방식
    - 할당에 시간에 맞춰서 그만큼씩 하고 넘겨줌
  + SJF 스케줄링 : 수행시간이 가장 짧다고 판단되는거부터 비선점으로 함
  + SRT 스케줄링 : SJT를 선점 형태로(도착시간에 수행시간이 가장 짧은거부터)
  + HRN : SJF의 단점 보안 비선점 방식
* 교착상태 : 시스템 내에서 자원을 사용하기 위해 경쟁하거나 블록킹된 상대
  + 상호배제 / 점유와 대기 / 비선점 / 환영대기

S2. 네트워크 기초 활용하기

* 회선 교환 방식
  + 동일 경로로만 전달 🡪 대역폭이 고정 / 안정적인 전송률
  + 통신 회선 전달이 선행
  + 기억장치 사용X
  + 다른 단말기와 통신불가
  + 실시간 통신 가능
* 패킷 교환 방식
  + 패킷이란 정보를 분할하여 정보 입력 🡪 현재 네트워크에서 주로 사용
  + 축적 후 전달 방식
  + 빠른 응답시간
  + 실시간 전송 불가
* 이더넷 : Lan의 가장 대표적인 버스 구조 방식
* OSI 7계층
  + 응용계층 : 사용자 친화 환경 제공
  + 표현계층 : 코드 압축, 해제, 보안 기능
  + 세션계층 : 송수신간 논리적 연결, 전송 방향 결정, 종단 시스템 응용
  + 전송(트랜스포트)계층 : 송수신 프로세스 연결, 종단 간 전송 제공, 주소 ( TCP/IP, UDP)
  + 네트워크계층 : 경로 선택 ( 라우터 : 망 연동 장비 )
  + 데이터링크계층 : 오류의 흐름 제어, 데이터 전송 ( 브리지, 스위치)
  + 물리계층 : 싱제 장비 연결 ( 허브 , 리피터 )

S3. 기본 개발 환경 구축하기

* 운영체제 및 운용
  + 윈도우계열 : Window Home / Pro / Pro for Workstation
  + 리눅스/유닉스(오픈소스 기반) : 레드헷 계열
    - Debian / Ubuntu(가장 광범위하게 쓰이는 Linux 운영체제) / Fedora
* 웹 서버 : 웹페이지를 클라이언트로 전달하는 것

5장 정보 시스템 구축 관리

Chapter1. 소프트웨어 개발 방법론 활용

S1. 소프트웨어 개발 방법론 선정

* 응용 소프트웨어 : 산업 범용 소프트웨어 중 하나
* 계획 🡪 요구분석 🡪 설계 -🡪 구현 🡪 검사 🡪 유지보수
* 소프트웨어 개발 모델
  + 폭포수 모델(단계적 생명주기) : 전형적 계발 모델
  + 프로토 타입 모델
  + 나선형 모델
  + 반복적 개발 모델
  + 진화적 개발 모델
  + RAD(Rapid Application Development)
* 정보공학 방법론 : 정보 시스템의 개발을 위해 계획 분석 설계의 정형화된 기법
* 애자일 방법론 : 바로바로 상황에 맞게 / 유연성
  + 익스트림 프로그래밍
  + 스크럴
  + 크리스털
* 관리 3P : People / Problem / Process
* 델파이 기법 : 전무가 감정 기법

S2. 소프트웨어 개발 방법론 테일러링

* 테일러링 : 프로젝트의 상황의 특성에 맞게 방법론 등의 수정 및 보완 작업
  + 규모와 복잡성에 따라 / 프로젝트 구성원에 따라 / 자동화에 따라
* 테일러링 품질 관리 / 소프트웨어 개발 표준
  + ISO 12207
  + CMMI 모델
  + SPICE 모델

Chapter2. IT 프로젝트 정보 시스템 구축 관리

S1. 네트워크 구축 관리

* 자율 사물 : AI를 활용하는 실제 디바이스
* 분산형 클라우드 : 퍼블릭 클라우드 서비스
* 메시 네트워크 : 모든 구간을 무선으로 연결
* SDN(Sotfware Defined Network) : 사용자가 소프트웨어로 네트워크를 제어하는 기술
* RFID : 주파수로 ID 식별
* NFC : 가까운 거리의 무선 통신
* USN : 유비쿼터스 센서 네트워크 : 필요한곳에 RFID 부착해서 네트워크 연결
* SON : 주변상황에 자동으로 적응하여 스스로 망을 구성하는 네트워크(자동구성네트워크)
* BLE : 저전력 블루투스 기술
* 에지 컴퓨팅 : 근거리에서 실시간 처리 방식 🡪 대역폭 사용량 감소
* 백본망(통신국사와 센터를 연결하는 망 : MSPP, DWDM, Metro Ethernet
* 클라우드 서비스 모델 : SaaS, PaaS, IaaS

S2. 소프트웨어 구축 관리하기

* 인간증강 : 인간이 시스템에
* 분산원장기술(DLT) : 중앙관리자, 데이터 저장소가 없음, P2P 망 내 모든 참여자가 공유
  + 블록체인기술 : 온라인 금융 거래 정보를 디지털 장비에 분산 저장
* 디지털 트윈 : 현실 세계에 존재하는 것의 디지털 버전
* 그레이웨어 : 악성 소프트웨어로 사용자의 동의 후 설치, 직접적인 위협X
* 시맨틱 웹 : 컴퓨터가 이해할 수 있는 웹, 기계들 끼리 의사소통할 수 있음
* 정보보안의 3요소
  + 기밀성 / 가용성 / 무결성

S3. 하드웨어 구축 관리하기

* 양자 컴퓨팅 : 정보를 양자비트로
* 고가용성(HA) : 긴시간 동안 지속적으로 운영이 가능한 시스템, 컴포넌트
* RAID : 컴퓨터 저장 장치 🡪 데이터를 분산저장 / 다중화(디스크를 묶어서 사용)
* 패블릿 : 폰+태블릿 5인치 이상의 스마트폰
* 멤리스터 : 메모리 + 레지스터 / 전류의 방향, 크기 등 모든 것을 기억
* Secure OS : 운영체제의 커널에 보안기능을 추가한 것
* 신 클라이언트 PC : 주변장치 없이 기본적인 메모리만 갖춘 PC
* 컨버지드 인프라 : 데이터센터의 3대요소(서버,스토리지,네트워크 인프라)를 조합
* SDI : 소프트웨어 정의를 인프라에 접목한 기술(소프트웨어에 의해 네트워크를 제어(SDN))
* 서버용 운영체제
  + 유닉스 계열 : 솔라리스
  + 리눅스 계열 : 레드 헷, 페도라
* DHCP 서버 : IP주소, 서브넷마스크, 데이트웨이, DNS 자동 할당

S4. DB 구축 관리하기

* IT기술 트렌드
  + 단순화 / 지능화 / 융합화
* 디지털 아카이빙 : 거대한 문서 저장고 / 콘텐츠 아카이브 / 정보에 대한 접근성 향상
* 브로드 데이터 : 빅데이터와 비슷, 다양한 정보를 일컫는 말
* DB 기술 트렌드
  + GPU 컴퓨팅 : 비디오 카드가 GPU라는 것
  + VRAM : 비휘발성 메모리
  + 스케일 아웃 SQL : 여러대의 노드에 데이터 스토리지 분산 능력
  + 공간 정보 데베 : DB를 2차원 공간으로
  + 텍스트 마이닝 : 자연어 처리 기술(비정형화된) 🡪 정보 추출
  + 블록체인 : 매우 안정적이고 실용적인 분산 데이터 저장소
  + 빅데이터 : 3V로 표현됨 ( 데이터의 양(volume), 데이터 생성 속도(Velocity), 다양성)
  + 디지털 아카이빙 : 정보의 생산/전달/공유 속도를 극대화 & 접근성 향상
  + 클라우드 데이터베이스 : 어디서나 데이터를 활용 가능
* DB 구축 일정 통제 방안
  + 크레싱 기법 : 프로젝트에 인적/물적 자원을 더 투입하여 일정을 단축 시키는 것
  + 중첩 실행 기법(Fast Tracking) : 순차적으로 진행되는 작업을 동시에 병렬로 작업을 진행함
* 로그 파일을 이용한 회복
  + UNDO : 장애가 발생한 후 로그를 보면서 트랜잭션이 변경한 내용을 원상복구

Chapter3. 소프트웨어 개발 보안 구축

S1. 소프트웨어 개발 보안 설계

* SDLC (소프트웨어 개발 생명주기) : 소프트웨어 생성에서 소멸까지의 과정을 단계별
  + 정의 단계(What) 🡪 개발 단계(How) 🡪 유지 보수
* 소프트웨어 개발 보안 적용 사례
  + MS-SDL(마이크로 소프트웨어 Secure Develop Lifecycle) : 이전보다 50% 취약점감소
* 세션 통제 : 세션 = 클라이언트와 서버의 논리적인 연결

S2. 소프트웨어 개발 보안 구현

* 소프트웨어 개발 보안 구현
  + 시큐어 코딩을 이용
  + 소프트웨어 개발 보안 등급
    - 보안 요소 : 조직의 보안기준에 따른 등급 구분
    - 기밀성 : 고객비밀 / 매우극비 / 중간적인 비밀 / 대외비 / 일반정보
    - 무결성 : high / Middle / Low
    - 가용성 : 1분, 시간 ,1주 등 시간
* 암호 알고리즘 : 비번, 주민번호, 계좌 등 중요정보를 암호화된 문장으로 만드는 절차
  + 대칭키 암호화 : 암호화키와 복호화키가 동일함(양방향 암호화 기법)
    - 스트림 암호 / 블록 암호
  + 공개키 암호화 : 공개키와 개인키로 암호화/복호화(암호화키 =/ 복호화키)
    - 공개키 : 지정된 인증기관에 의해 제공되는 키값
  + 해시 : 하나의 문자열을 더 짧은 값으로 변화하는 방식(단방향 암호화 방식)

Chapter4. 시스템 보안 구축

S1. 시스템 보안 설계

* 서비스 공격 유형
  + 위치에 따른 해킹 분류 : 지역적인 공격(Root 권한 획득) / 원격 공격(버퍼 오버플로우🡺이메일 관련/스팸같은거)
* 해킹 수법
  + 사회 공학적 침입 수법 : 시스템 관리자를 속이는거
  + 사용자 도용 : 스니퍼 해킹 프로그램(ID와 패스워드 훔치는거)
  + 시스템 취약점 공격 : 버그 이용하여
  + 호스트 위장 : 호스트가 신뢰하는 시스템이나 네트워크로 위장하는거
  + Land Attack : 출발지와 목적지의 IP 주소를 동일하게 성정하여 공격
  + 죽음의 핑 : 라우팅 과정에서 패킷을 아주 작게 나눔 🡪 시스템 부하
  + Smurf : 다이렉트 브로드캐스트를 악용한 DoS 공격
  + DoS : 관리자 획득, 데이터 파괴 X 🡪 그냥 서비스 이용 못하게 하는거
  + DDoS : echo / discard / daytime / chargen 가 취약함
  + 핑거 : 사용자의 정보를 볼 수 있는 유닉스 기능
  + 소스 라우팅 : 경로에 모두 리스트로 만들어 패킷의 헤더에 넣어 보냄
  + FTP 바운스 공격 : FTP 프로토콜의 구조의 허점을 이용함(20/21번 포트 이용)
  + 메일 폭탄 : 걍 많이 보내는거
  + 스팸메일 : 무단광고
  + 메크로 바이러스 : 매크로 기능을 이용하여 하는거
  + DNS 스푸핑 : DNS서버를 해킹 / 위조 DNS 서버 설치🡪개인정보 탈취
* 지식기반 인증
  + 패스워드
  + 패스프레이즈 : 패스워드보다 긴 문자열을 사용하는거(아이핀)
  + 소유기반 인증
    - 메모리카드 / 스마트 카드 / OTP
  + 생체기반인증
  + 위치기반 인증
  + 서버 인증 : TLS / SSL 프로토콜 연결해서 인증
  + AAA / 권한 부여 / 계정 관리 : 시스템 접근을 허용하기 전에 사용자의 신원을 검증
* 보안 아키텍처 및 프레임워크
  + 시스템 보안 설계 원칙
    - 원칙 1 : 보안수준 == 조직에 주는 가치
    - 원칙 2 : 변화하는 보안의 필요와 요구사항을 수용할 수 있어야 함
    - 원칙 3 : 통합된 보안 서비스를 제공해야 함
    - 원칙 4 : 모든 플랫폼에 걸쳐 일관성 있는 프레임워크 제공해야함

S2. 시스템 보안 구현

* 로그 분석 : 로그 = 시스템이 운영되는 모든 기록을 담고 있는 데이터