과제 3) 구인구직 백엔드 서버 만들기 (사람인..?)

이번 과제에서는 사람인 데이터를 활용하여 구인구직 백엔드 서버를 구축합니다. 이를 통해 데이터 크롤링, DB 설계, REST API 개발, 인증 시스템 구축의 전 과정을 경험합니다. 프로그램은 **JCloud를 사용해 배포**해야하며, 이를 통해 클라우드 환경에서의 서버 운영 경험을 쌓을 수 있도록 합니다. **이번 과제에서는 Node.js (Express.js), Flask (Python), Spring Boot (Java) 중 하나를 선택하여 백엔드 서버를 구축하면 됩니다. 이 외에도, 본인이 원하는 언어 및 프레임워크를 선택하여 개발해도 무방합니다.** 또한, SQL (MySQL) 또는 NoSQL (MongoDB) 중 하나를 선택하여 데이터베이스를 설계하면 됩니다. 추가로, Swagger를 이용하여 API 문서화와 JWT 기반 인증 시스템을 구축하면 됩니다.

개발 언어 및 프레임워크: Node.js (Express.js), Flask (Python), Spring Boot (Java) 중 하나 선택 (이 외에 선택 시 면담 요청)

A. 과제 목적

- **웹 크롤링 이해 및 적용**: 사람인에서 채용 공고 데이터를 크롤링하여 수집.
- **데이터베이스 설계**: 크롤링한 데이터를 SQL (MySQL) 또는 NoSQL (MongoDB)로 구조화.
- **REST API 개발**: 크롤링한 데이터를 기반으로 한 다양한 기능의 API 개발 및 회원 인증 기능 구현.
- **문서화 및 인증**: Swagger를 이용한 API 문서화와 JWT 기반 인증 시스템 적용.
- **클라우드 배포**: JCloud를 사용하여 백엔드 서버를 배포.

B. 제출 방법

- 과제 마감일: **2024년 12월 08일 10:00 A.M (KST)**
- 과제 제출 방법: **Google Classroom**을 통해 제출
- **Google Classroom**에 **제출**하기 버튼을 눌러 제출하면 됩니다.
- 아래 파일들을 **하나의 zip파일**로 묶어 다음과 같은 포맷으로 **[WSD-분반-학번-이름-3차과제.zip]** 파일 만들어 제출하면 됩니다.
  - **WSD-1분반-20240000-이경수-3차과제.zip**
    - **WSD-Assignment-03**
      - Node.js/ Flask/ SpringBoot 프로젝트 폴더
      - **라이브러리 및 패키지 파일은 반드시 포함하지 말 것 (용량 문제로 삭제할 것임)**
      - README.md 파일 필수 첨부 (빌드를 위한 명령어 필수)
      - **패키지 파일 필수 첨부 (설치 불가능하면 0점처리)**
        - **npm:** package.json
        - **Python:** requirements.txt
        - **Maven:** pom.xml
        - **Gradle:** build.gradle
        - **Ant:** build.xml
    - **link.pdf**
      - Github repository 주소: https://github.com/~~~
      - 앱이 배포된 주소 (JCloud 포트 번호 포함): https://~~~/~~~:0000
    - **swagger.yaml/ swagger.json/ swagger.pdf 또는 swagger가 배포된 주소**
      - Swagger API 문서 파일 (10% 감점)
      - Swagger API 문서가 배포된 주소 (Swagger를 통해 API가 동작하지 않을 경우 10% 감점)
      - **즉, Swagger를 온라인으로 배포하고 동적으로 테스트 해볼 수 있어야 함.**
    - **crawled-data.sql 또는 crawled-data.json 파일**
      - 크롤링한 데이터가 저장된 DB 파일
      - SQL 덤프 파일 또는 MongoDB 익스포트 파일

C. 유의 사항

- **표절 금지**:
  - 모든 작업은 본인의 노력으로 이루어져야 합니다
  - 타인의 코드를 무단으로 복사하거나 표절할 경우 0점 처리됩니다

- **제출 형식**:
  - 모든 제출물은 지정된 형식을 준수해야 합니다
  - 압축 파일명, 폴더 구조를 반드시 준수해 주세요
  - 미준수 시 채점 상 불이익이 있을 수 있습니다

- **개발 환경**:
  - 꼭! node_modules 폴더는 제외하고 제출합니다
  - **패키지 관리 파일은 반드시 포함되어야 합니다**
  - 꼭! 프로젝트가 정상적으로 실행/빌드되는지 확인 후 제출해 주세요

- **문의 사항**:
  - 과제 수행 중 질문이 있으면 다음 채널을 통해 문의해 주세요:
    - 이메일
    - Discord
    - Google Classroom
  - 문의 시 본인의 학번, 이름, 분반을 함께 기재해 주세요

D. 과제 개요

본 과제는 Node.js를 활용하여 구인구직 정보를 제공하는 백엔드 서버를 개발하는 것을 목표로 합니다. 사람인 웹사이트의 데이터를 크롤링하여 데이터베이스에 저장하고, 이를 기반으로 RESTful API를 개발합니다. 또한 JWT를 활용한 인증 시스템을 구현하고, Swagger를 통해 API 문서화를 진행합니다.

1. 주요 평가 내용

본 과제에서 주요하게 평가할 주요 기능은 아래와 같습니다. 다만, (Optional) **이라고 적혀 있는 것은 평가 대상은 아니지만, 나중을 위해 한 번 공부해보시는 것을 권장드립니다.**

a. 웹 크롤링 구현

- 사람인 웹사이트 크롤링 구현
  - 크롤링을 위한 다양한 라이브러리 활용 (언어/프레임워크 무관)
  - 채용 정보 데이터 수집 및 정제
    - 설계된 DB 구조에 맞는 데이터 정제
    - 데이터 무결성: 이미 있는 데이터는 저장하지 않도록 처리하는 로직
  - 에러 처리 및 재시도 로직 구현
  - 병렬 처리를 통한 크롤링 성능 최적화 (Optional)
- 데이터 수집 및 저장
  - 최소 100개 이상의 채용 공고 데이터 수집
  - 수집된 데이터 구조화 및 정규화
  - 중복 데이터 처리 로직 구현
  - 데이터 업데이트 주기 관리 (Optional)
- 크롤링 자동화 (Optional)

b. 데이터베이스 설계 및 구현

- 데이터베이스 스키마 설계
  - 테이블/컬렉션 구조 설계
    - 자체적인 DB 설계 (Primary Key 필수)
    - 빠른 탐색을 위한 DB 구조 설계
    - 중복이 없는 DB 안정적인 DB 설계
  - 관계 설정 및 제약조건 정의
  - 인덱스 최적화 (Optional)
  - 정규화/비정규화 전략 수립 (Optional)
- 필수 데이터 모델 구현
  - 채용 공고 정보 모델
  - 회사 정보 모델
  - 사용자 정보 모델
  - 지원 내역 모델
  - 북마크/관심공고 모델
  - **그 외 3개 이상 필수**

    - 주기적 크롤링 스케줄링
    - 증분 업데이트 구현
    - 크롤링 로그 관리
    - 실패 복구 메커니즘

- 데이터베이스 최적화 (Optional)
    - 쿼리 성능 최적화
    - 인덱스 전략 수립
    - 캐싱 레이어 구현
    - 데이터 마이그레이션 전략

c. REST API 개발

- 기본 CRUD API 구현
    - 회원 관리 관련 API
        - 회원 가입/로그인 API
            - 회원 가입
            - 로그인
            - 비밀번호 암호화 (Base64 Encoding)
            - 회원 정보 수정
        - 회원 정보 조회 API
        - 회원 탈퇴 API
    - 채용 공고 관련 API
        - 채용 공고 조회 API
        - 채용 공고 검색 API
        - 채용 공고 필터링 API
        - 채용 공고 정렬 API
        - 채용 공고 등록 API
        - 채용 공고 수정 API
        - 채용 공고 삭제 API
    - 지원 관련 API
        - 지원하기 API
        - 관심 등록 API
        - 지원 취소 API
        - 지원 내역 조회 API
    - **그 외 3개 이상 필수**
        - 관심목록 조회API

- 고급 API 기능 구현
    - 필터링 및 검색 기능
    - 페이지네이션 처리
    - 정렬 기능
    - 데이터 집계 API

- API 보안
    - JWT 발급 API
    - JWT 기반 인증 API 및 로직
    - 권한 검사 미들웨어
    - 입력 데이터 검증
    - Rate Limiting (Optional)

- API 최적화 (Optional)
    - 응답 데이터 캐싱
    - N+1 문제 해결
    - 벌크 연산 처리
    - 부분 응답 처리

- API 테스트 (Optional)
    - 단위 테스트 작성
    - 통합 테스트 작성
    - 엣지 케이스 테스트
    - 테스트 커버리지 측정

d. 인증 및 보안 구현

- JWT 기반 인증
    - Access Token 발급 및 검증 (필수)
    - Refresh Token 구현 (가산점)
    - 토큰 갱신 메커니즘 (필수)
    - 토큰 블랙리스트 관리 (Optional)

- 보안 미들웨어 구현
    - 인증 미들웨어
    - 권한 검사 미들웨어
    - 입력 데이터 및 파라미터 검증
    - Rate Limiting (Optional)

- 보안 강화 (Optional)
    - XSS 방지
    - CSRF 보호
    - SQL Injection 방지
    - 암호화 처리

e. API 문서화 (Swagger)

- Swagger 문서 작성
    - API 엔드포인트 설명
    - 요청/응답 스키마 정의
    - 인증 방식 설명
    - 에러 코드 및 처리 방법

- API 테스트 환경 구성
    - Swagger UI 설정
    - 테스트 데이터 제공
    - API 사용 예제 작성
    - 환경별 설정 관리
    - **Swagger를 배포하여 실제 테스트 해볼 수 있어야 함!**

- 문서 자동화 (Optional)
    - 코드 기반 문서 생성
    - 버전 관리
    - 변경 사항 추적
    - 문서 배포 자동화

f. 에러 처리 및 로깅

- 에러 처리 구현
  - 글로벌 에러 핸들러 (미들웨어 등)
  - 커스텀 에러 클래스 (인증 및 데이터 포맷 에러 필수 구현)
  - HTTP 상태 코드 매핑
  - 에러 응답 포맷 통일

- 로깅 시스템 구축 **(가산점)**
  - 요청/응답 로깅
  - 에러 로깅
  - 성능 모니터링
  - 로그 레벨 관리

- 모니터링 시스템 (Optional)
  - 성능 메트릭 수집
  - 알림 시스템 구축
  - 로그 분석 도구 연동
  - 대시보드 구성

g. 코드 최적화 및 주석

- 코드 구조화 및 모듈화
  - MVC 아키텍처 패턴 적용 (필수)
  - 재사용 가능한 유틸리티 함수 구현
  - 프로젝트 폴더 구조 최적화
  - 의존성 주입 패턴 적용

- 코드 문서화 **(가산점)**
  - JSDoc 스타일 주석 작성
    - 함수 설명
    - 파라미터 타입 및 설명
    - 반환값 타입 및 설명
    - 예외 처리 설명
  - 인라인 주석
    - 복잡한 로직 설명
    - 비즈니스 규칙 설명
    - 주요 알고리즘 설명

2. Endpoint 별 주요 구현 내용

a. 회원 관리 API (/auth)

- 회원 가입 (POST /auth/register)
  - 이메일 형식 검증
  - 비밀번호 암호화 (Base64)
  - 중복 회원 검사
  - 사용자 정보 저장

- 로그인 (POST /auth/login)
  - 사용자 인증
  - JWT 토큰 발급
  - 로그인 이력 저장
  - 실패 시 에러 처리

- 토큰 갱신 (POST /auth/refresh)
  - Refresh 토큰 검증
  - 새로운 Access 토큰 발급
  - 토큰 만료 처리

- 회원 정보 수정 (PUT /auth/profile)
  - 인증 미들웨어 적용
  - 비밀번호 변경
  - 프로필 정보 수정

b. 채용 공고 API (/jobs)

- 공고 목록 조회 (GET /jobs)
  - 페이지네이션 처리 (필수)
    - 페이지 크기: 20

c. 지원 관리 API (/applications)

- 지원하기 (POST /applications)
  - 인증 확인
  - 중복 지원 체크
  - 지원 정보 저장
  - 이력서 첨부 (선택)

- 지원 내역 조회 (GET /applications)
  - 사용자별 지원 목록
  - 상태별 필터링
  - 날짜별 정렬

- 지원 취소 (DELETE /applications/:id)
  - 인증 확인
  - 취소 가능 여부 확인
  - 상태 업데이트

d. 북마크 API (/bookmarks)

- 북마크 추가/제거 (POST /bookmarks)
  - 인증 확인
  - 북마크 토글 처리
  - 사용자별 저장

- 북마크 목록 조회 (GET /bookmarks)
  - 사용자별 북마크
  - 페이지네이션
  - 최신순 정렬

        - 정렬 기준 제공
    - 필터링 기능 (필수)
        - 지역별
        - 경력별
        - 급여별
        - 기술스택별
    - 검색 기능 (필수)
        - 키워드 검색
        - 회사명 검색
        - 포지션 검색
- 공고 상세 조회 (GET /jobs/:id)
    - 상세 정보 제공
    - 조회수 증가
    - 관련 공고 추천

e. 기타 구현 필수 API

- 이 외에 D-1에서 명시된 기능은 필수적으로 구현되어야 하며, 해당 기능에 대한 API가 존재해야 합니다.
- 구체적인 조건이 명시되지 않은 경우, 해당 기능에 대한 API를 자유롭게 설계하셔도 무방합니다.
- API 응답 형식 통일 (예시)
    - 성공 응답
      ```
      {
          "status": "success",
          "data": {
              // 실제 데이터
          },
          "pagination?: {
              "currentPage": 1,
              "totalPages": 10,
              "totalItems": 100
          }
      }
      ```
    - 실패 응답
      ```
      {
          "status": "error",
          "message": "에러 메시지",
          "code": "ERROR_CODE"
      }
      ```

E. 평가 기준

세부적인 평가 항목들은 위 D-1와 D-2에 적혀있는 내용을 바탕으로 합니다. 다만, 세부 배점 등은 추후 공지 예정이니, 우선 D-1/D-2을 기준으로 개발하시면 될 것 같습니다 🥹

아래 내용은 변경될 수 있으니 개발하시면서 공지 꼭 확인해주세요!!

a. 필수 구현 항목

- **웹 크롤링 구현**
    - 사람인 웹사이트의 채용 정보 크롤링
    - 최소 100개 이상의 데이터 수집
    - 데이터 정제 및 구조화
    - 중복 데이터 처리 로직
    - 에러 처리 및 재시도 메커니즘

- **데이터베이스 설계 및 구현**
    - MySQL 또는 MongoDB 데이터베이스 구현
    - 최소 8개 이상의 테이블/컬렉션 설계
    - Primary Key 및 관계 설정
    - 데이터 정규화
    - 효율적인 쿼리를 위한 인덱스 설정

- **REST API 개발**
    - 회원 관리 API (회원가입, 로그인, 정보수정)
    - 채용 공고 관련 API (CRUD)
    - 지원 관련 API (지원, 취소, 조회)
    - 검색 및 필터링 API
    - 페이지네이션 구현
    - 이 외 D-1/D-2에 정의된 API
    - 추가 3개 이상의 기능성 API

- **인증 및 보안 구현**
    - JWT 기반 인증 시스템
    - 토큰 발급 및 갱신
    - 인증 미들웨어 구현
    - 보안 처리 (Base64 암호화 등)

- **API 문서화 (Swagger)**
    - 모든 API에 대한 Swagger 문서 작성
    - 요청/응답 스키마 정의
    - API 테스트 환경 구성
    - 배포된 Swagger UI 제공

- **에러 처리 및 로깅**
    - 글로벌 에러 핸들러 구현
    - 커스텀 에러 클래스 정의
    - 통일된 에러 응답 형식
    - 에러 로깅 시스템

- **코드 최적화 및 문서화**
    - MVC 아키텍처 패턴 적용
    - 모듈화 및 재사용성 고려
    - README 작성 및 프로젝트 문서화

b. 점수표

- 개발한 API가 아래 기준을 충족할 경우 점수를 획득하며, 점수에 따라 해당 과제 학점이 결정됩니다.
- **점수-학점 기준**
  - 추후 공지

c. 상세 채점 항목

- 추후 공지