학생 답안 제출 시스템

코드 읽기 과제를 위한 웹 기반 학생 답안 제출 시스템입니다. 학생들이 주어진 코드를 분석하고 답안을 제출할 수 있는 플랫폼을 제공합니다.

🚀 주요 기능

- **사용자 인증**: JWT 기반 로그인 시스템
- **문제 관리**: 코드 읽기 문제 제시 및 관리
- **답안 제출**: 자유형식 텍스트 답안 작성 및 제출
- **제출 이력**: 개인별 답안 제출 현황 확인
- **반응형 UI**: 데스크톱 및 모바일 환경 지원

🛠 기술 스택

Backend

- **FastAPI**: 고성능 Python 웹 프레임워크
- **MongoDB**: NoSQL 데이터베이스 (MongoDB Atlas 사용)
- **Motor**: MongoDB 비동기 드라이버
- **JWT**: JSON Web Token 기반 인증
- **bcrypt**: 비밀번호 해싱

Frontend

- **HTML5/CSS3**: 현대적이고 반응형인 UI
- **Vanilla JavaScript**: 경량화된 클라이언트 사이드 로직
- **Grid Layout**: 효율적인 레이아웃 구성

📁 프로젝트 구조

```
├── fastapi_main.py      # FastAPI 메인 애플리케이션
├── requirements.txt     # Python 의존성 패키지
├── static/              # 정적 파일 디렉토리
│   ├── login.html       # 로그인 페이지
│   ├── problems.html    # 문제 목록 페이지
│   └── problem.html     # 개별 문제 풀이 페이지
├── user_info.csv        # 사용자 정보 (468명)
└── README.md            # 프로젝트 문서
```

🔧 설치 및 실행

1. 환경 준비

bash

```bash
# Python 3.8+ 필요
python --version

# 가상환경 생성 및 활성화
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
```

2. 의존성 설치

bash

```bash
pip install -r requirements.txt
```

3. MongoDB 설정

MongoDB Atlas를 사용하며, 다음 컬렉션들이 필요합니다:

- **User_info**: 사용자 정보 (ID, PW, Hash_ID)
- **Question_info**: 문제 정보 (Question_id, Question Description, Code)
- **Submission**: 답안 제출 내역

4. 서버 실행

bash

```bash
python fastapi_main.py
```

또는

bash

```bash
uvicorn fastapi_main:app --host 0.0.0.0 --port 8000 --reload
```

서버가 실행되면 `http://localhost:8000`에서 접근 가능합니다.

📋 API 엔드포인트

인증

- `POST /api/login` - 사용자 로그인
- `GET /api/debug/users` - 디버깅용 사용자 목록 조회

문제 관리

- `GET /api/problems` - 전체 문제 목록 조회
- `GET /api/problems/{problem_id}` - 특정 문제 상세 조회

답안 제출

- `POST /api/submit` - 답안 제출
- `GET /api/my-submissions` - 내 제출 내역 조회

페이지 라우팅

- `GET /` - 로그인 페이지
- `GET /problems` - 문제 목록 페이지
- `GET /problem/{problem_id}` - 문제 풀이 페이지

👥 사용자 관리

데이터베이스 구조

User_info 컬렉션:

```json
{
  "ID": 250100,          // 학번 (숫자)
  "PW": 250100,          // 비밀번호 (학번과 동일)
  "Hash_ID": 327         // 해시된 사용자 ID
}
```

Question_info 컬렉션:

```json
{
  "Question_id": 1,
  "Question Description": "다음 코드의 기능을 설명하시오.",
  "Code": "def example_function():\n    return 'Hello World'"
}
```

Submission 컬렉션:

json

```
{
  "user_id": 250100,
  "Hash_ID": 327,
  "problem_id": "q1",
  "question_id": 1,
  "answer": "학생이 작성한 답안",
  "submitted_at": "2025-05-26T10:30:00Z"
}
```

테스트 계정

로그인 시 학번과 동일한 비밀번호를 사용합니다:

- 학번: `250100`, 비밀번호: `250100`
- 학번: `250101`, 비밀번호: `250101`
- 학번: `250102`, 비밀번호: `250102`

🎨 UI/UX 특징

반응형 디자인

- 데스크톱과 모바일 환경 모두 지원
- Grid 레이아웃으로 효율적인 공간 활용
- 직관적인 네비게이션과 사용자 경험

시각적 특징

- 모던한 그라디언트 색상 (`#667eea` → `#764ba2`)
- 카드 기반 레이아웃으로 정보 구조화
- 호버 효과와 부드러운 트랜지션
- 코드 블록 하이라이팅

사용성

- 실시간 글자 수 카운터
- 제출 상태 표시 (제출 완료/미제출)
- 이전 제출 내역 자동 로드
- 답안 수정 및 재제출 지원

🔒 보안 기능

- **JWT 토큰**: 60분 만료 시간 설정

- **비밀번호 해싱**: bcrypt 알고리즘 사용
- **토큰 기반 인증**: 모든 API 엔드포인트 보호
- **세션 관리**: 자동 로그아웃 및 토큰 갱신

🐛 디버깅 기능

개발 및 테스트를 위한 디버깅 엔드포인트:

- `/api/debug/user/{user_id}` - 특정 사용자 조회
- `/api/debug/users` - 모든 사용자 조회 (최대 10명)
- `/api/debug/questions` - 모든 문제 조회

📝 사용 방법

1. **로그인**: 학번과 비밀번호로 로그인
2. **문제 선택**: 문제 목록에서 원하는 문제 선택
3. **답안 작성**: 코드를 분석하고 자유 형식으로 답안 작성
4. **제출**: 답안 제출 버튼 클릭
5. **수정**: 필요시 답안 수정 후 재제출

🚨 주의사항

- MongoDB Atlas 연결 정보는 보안상 환경 변수로 관리하는 것을 권장
- JWT SECRET_KEY는 운영 환경에서 반드시 변경 필요
- 대용량 답안 제출 시 MongoDB 문서 크기 제한(16MB) 고려

📊 시스템 요구사항

- **Python**: 3.8 이상
- **메모리**: 최소 512MB RAM
- **디스크**: 100MB 이상 여유 공간
- **네트워크**: MongoDB Atlas 접속을 위한 인터넷 연결

🤝 기여 방법

1. 이슈 등록 또는 기능 제안
2. Fork 후 브랜치 생성
3. 변경사항 커밋 및 푸시
4. Pull Request 생성

📄 라이선스

이 프로젝트는 교육 목적으로 개발되었습니다.