이루다 프로젝트 Flask 코드 주요 함수 분석

📋 전체 아키텍처 구조

- **Flask 웹 프레임워크** 기반
- **SQLite 데이터베이스** 사용
- **Flask-Login**으로 사용자 인증 관리
- **Excel 파일**에서 정부정책 데이터 로드
- **OpenAI API** 연동 준비 (현재 주석 처리)

🔐 사용자 인증 관련 함수

`User` 클래스

```python
class User(UserMixin):
    def __init__(self, id, email, name)
```

- Flask-Login과 연동되는 사용자 모델
- 사용자 ID, 이메일, 이름 정보 관리

`load_user(user_id)`

```python
@login_manager.user_loader
def load_user(user_id)
```

- Flask-Login에서 세션 관리용
- 데이터베이스에서 사용자 정보 조회

`register()` - 회원가입

```python
@app.route('/register', methods=['GET', 'POST'])
def register()
```

기능:

- 사용자 기본정보 입력 (이름, 이메일, 비밀번호)
- **프로필 정보** 수집 (주거상황, 소득수준, 지원필요분야)

- 비밀번호 해시화 후 DB 저장
- 이메일 중복 검증

`login()` - 로그인

```python
@app.route('/login', methods=['GET', 'POST'])
def login()
```

기능:

- 이메일/비밀번호 검증
- 세션 생성 후 대시보드로 리다이렉트

🏠 핵심 페이지 라우트

`home()` - 메인 페이지

```python
@app.route('/')
def home()
```

- 서비스 소개 및 첫 진입점

`dashboard()` - 대시보드

```python
@app.route('/dashboard')
@login_required
def dashboard()
```

핵심 기능:

- 사용자별 **맞춤형 로드맵** 조회
- 최근 생성된 로드맵의 title, description, priority_areas, timeline 표시
- JSON 데이터를 파이썬 객체로 변환하여 템플릿에 전달

`dashboard_stats()` - 대시보드 통계 API

```python
```

```
@app.route('/dashboard-stats')
@login_required
def dashboard_stats()
```

기능:

- 활성 할일 개수 (`pending`, `in_progress` 상태)
- 완료된 할일 개수 (`completed` 상태)
- 추천 정책 개수 (현재 하드코딩: 3개)

🤖 AI 상담 및 로드맵 생성

`chat()` - AI 챗봇 API

```
python

@app.route('/chat', methods=['POST'])
@login_required
def chat()
```

기능:

- 사용자 메시지 수신
- **OpenAI API** 연동 준비 (현재 비활성화)
- `generate_mock_response()` 함수로 임시 응답 생성
- 페이지 이동 제안 기능 포함

`generate_mock_response(message)`

```
python

def generate_mock_response(message)
```

응답 패턴:

- '로드맵', '계획' → 로드맵 생성 안내
- '정책', '지원' → 정책 검색 안내
- 'todo', '할일' → 할일 관리 안내
- '안녕', 'hello' → 개인화된 인사
- '도움', 'help' → 서비스 기능 소개

`check_for_page_suggestion(message)`

```python
def check_for_page_suggestion(message)
```

기능:

- 메시지 내용 분석
- 적절한 페이지로 이동 제안
- `/roadmap`, `/policies` 페이지 안내

`generate_roadmap()` - AI 로드맵 생성

```python
@app.route('/generate-roadmap', methods=['POST'])
@login_required
def generate_roadmap()
```

핵심 프로세스:

1. 사용자 프로필 정보 조회 (주거상황, 소득수준, 지원필요분야)
2. **맞춤형 로드맵 생성** (현재는 샘플 데이터)
3. 로드맵을 JSON 형태로 데이터베이스 저장
4. timeline: 1개월, 3개월, 6개월 단계별 계획

🗺️ 로드맵 관리 기능

`roadmap()` - 로드맵 페이지

```python
@app.route('/roadmap')
@login_required
def roadmap()
```

기능:

- 사용자의 최신 로드맵 조회
- **진행률 계산** (완료된 할일/전체 할일)
- 로드맵과 진행률을 템플릿에 전달

`roadmap_detail_plan()` - 상세 계획 생성 API

```python
@app.route('/roadmap/detail-plan', methods=['POST'])
@login_required
def roadmap_detail_plan()
```

기능:

- 기간(period)과 목표(goals) 입력받아 상세 계획 생성
- `generate_detail_plan()` 함수 호출

`generate_detail_plan(period, goals)`

```python
def generate_detail_plan(period, goals)
```

세부 계획 생성 로직:

- **주거 관련**: 주거급여 신청, 필요서류 준비, 관할 주민센터 접수 등
- **취업 관련**: 이력서 작성, 취업지원 프로그램, 직업훈련, 면접 준비 등
- **경제 관련**: 수입/지출 파악, 생계급여 신청, 가계부 작성, 비상자금 적립 등

`convert_to_todos()` - TODO 변환 API

```python
@app.route('/roadmap/convert-to-todos', methods=['POST'])
@login_required
def convert_to_todos()
```

기능:

- 로드맵의 각 목표를 개별 할일로 변환
- `progress_tracking` 테이블에 저장
- 상태: `pending`, 우선순위: 3으로 설정

🏛️ 정책 검색 및 추천

`load_government_policies()`

```python
```

```
def load_government_policies()
```

기능:

- **Excel 파일**에서 정부정책 데이터 로드
- 중앙부처, 지자체, 민간 데이터 통합
- pandas로 Excel 읽기 후 딕셔너리 리스트로 변환

`policies()` - 정책 검색 페이지

```python
@app.route('/policies')
@login_required
def policies()
```

검색 필터 기능:

- **검색어 필터링**: 서비스명, 기관명, 지원내용, 지원대상에서 검색
- **카테고리 필터링**: 구분 필드 기준
- **추천 정책 필터링**: `get_recommended_policies()` 함수 호출

`get_recommended_policies(policies)`

```python
def get_recommended_policies(policies)
```

추천 알고리즘:

1. 사용자 프로필에서 support_needs 추출
2. 각 정책에 대해 **점수 계산**:
   - 주거지원 매칭: +3점
   - 경제지원 매칭: +3점
   - 취업지원 매칭: +3점
   - 교육지원 매칭: +3점
   - 심리지원 매칭: +2점
   - 자립준비청년 대상: +2점
3. 점수순 정렬하여 상위 20개 반환

👤 사용자 정보 관리

`mypage()` - 마이페이지 조회/수정

```python
@app.route('/mypage', methods=['GET', 'POST'])
@login_required
def mypage()
```

GET 요청: 사용자 정보 및 프로필 조회 **POST 요청**:

- 기본정보 업데이트 (이름, 이메일, 나이)
- 프로필 정보 업데이트 (주거상황, 소득수준, 지원필요분야)
- **비밀번호 변경** (현재 비밀번호 검증 후)

⚠️ 미완성/임시 구현 부분

임시 페이지들

- `application_form_page()`: 신청양식 생성 (임시 HTML 반환)
- `mypage_roadmaps()`: 로드맵 관리 (구현 예정)
- `mypage_todos()`: 할일 관리 (구현 예정)

OpenAI API 연동

```python
# import openai  # 현재 주석 처리
# openai.api_key = os.getenv('OPENAI_API_KEY')
```

- OpenAI API 키 설정 및 import가 주석 처리됨
- 실제 AI 응답 대신 mock response 사용

🔧 기술적 특징

보안

- 비밀번호 해시화 (`generate_password_hash`, `check_password_hash`)
- 로그인 필수 데코레이터 (`@login_required`)
- 이메일 중복 방지

데이터 처리

- JSON 형태로 복잡한 데이터 저장 (로드맵, 지원필요분야)

- pandas를 활용한 Excel 데이터 처리
- SQLite 데이터베이스 연동

확장성

- 환경변수 활용 (`.env` 파일)
- 모듈화된 함수 구조
- OpenAI API 연동 준비 완료

🎯 Q&A 대비 포인트

기술적 질문 예상:

1. **"OpenAI API가 비활성화된 이유는?"** → 개발 단계에서 비용 절약, mock response로 테스트
2. **"정책 추천 알고리즘의 정확도는?"** → 키워드 매칭 기반, 향후 머신러닝 도입 예정
3. **"데이터베이스 확장성은?"** → SQLite에서 PostgreSQL로 전환 계획
4. **"보안 측면에서의 고려사항은?"** → 비밀번호 해시화, 세션 관리, SQL injection 방지

기능적 질문 예상:

1. **"실제 정부 API와 연동 계획은?"** → 복지로 API 연동 검토 중
2. **"사용자 피드백 수집 방법은?"** → 향후 평가 시스템 도입 예정
3. **"로드맵 생성 기준은?"** → 사용자 프로필 기반 규칙 엔진, AI 학습 데이터 추가 예정