요구사항 명세서 (SRS)

202212343 강현호 202213523 손천익

지도교수: 유준범 교수님

목차

1. 소개 (Introduction)

1.1 SRS의 목적 (Purpose of SRS)

이 문서의 목적은 유전자 알고리즘을 통해 무작위로 나열되어 있는 여행지와 레스토랑을 효율적으로 배치해 최대한 많은 경험을 할 수 있도록 일정을 짜기 위한 웹사이트를 만들기 위함이다.

1.2 산출물의 범위 (Scope of product)

경로 생성 및 최적화 기능: 사용자가 입력한 출발/도착 노드 및 중간 노드(명소, 식당, 바), 그리고 일일 운영 시간 제약을 기반으로 유효한 여행 경로를 도출합니다. 이 과정에서 유전 알고리즘을 사용하여 노드 간 최단 이동 시간을 고려하고, 경로를 세 가지 유형(활동 우선, 효율 우선, 균형 우선)으로 최적화하여 사용자에게 추천합니다. 또한, 식당 노드는 점심/저녁 시간대와 식사 가능 여부를 자동으로 확인하고 반영하여 경로에 배치합니다.

경로 스케줄링 및 관리 기능: 생성된 최적 경로에 대해 노드별 도착 및 출발 시각이 포함된 시간표 형식의 스케줄을 제공합니다. 추천된 경로는 활동/효율/균형 점수를 함께 제시하여 선택을 돕습니다. 최종 선택된 경로는 사용자 계정에 저장하여 재확인할 수 있습니다.

1.3 정의, 두문자어, 약어 (Definitions, acronyms and Abbreviations)

유전자 알고리즘	생물의 진화 과정을 모방하여 최적의 해답을 찾는 탐색 및 최적화 기법 본 시스템에서는 최적의 여행 경로를 도출하는 데 사용
노드	여행 경로의 구성 요소. 출발지, 도착지, 중간 방문지 등 특정 위치를 의미하며, 호텔, 식당, 명소 등이 이에 해당
피트니스 점수	유전자 알고리즘에서 개별 경로(해답)의 적합성을 평가하는 수치. 활동, 효율, 균형 점수의 가중치 합으로 계산되어 경로의 품질을 나타냄
세션 데이터	사용자가 경로를 생성하기 위해 입력한 임시 정보(노드 리스트, 시간 제약 등)로, 비영구적으로 저장되어 경로 생성에 활용되는 데이터

1.4 참조 문서 (References)

https://developers.google.com/maps/documentation/javascript?hl=ko
https://developers.google.com/maps/documentation/routes?hl=ko
https://developers.google.com/maps/documentation/javascript/places-js?hl=ko

https://developers.amadeus.com/self-service/category/flights/api-doc/airport-and-city-search

https://developers.amadeus.com/self-service/category/flights/api-doc/flight-offers-search/api-reference
https://developers.amadeus.com/self-service/category/hotels/api-doc/hotel-search
https://developers.amadeus.com/self-service/category/hotels/api-doc/hotel-booking

https://docs.tosspayments.com/reference/using-api/api-keys#api-%ED%82%A4-%ED%99%95%EC%9D%B8%ED%95%98%EA%B8%B0
https://docs.tosspayments.com/reference/using-api/api-keys
https://developers.portone.io/api/rest-v2?v=v2

https://www.tta.or.kr/tta/ttaSearchView.do?key=77&rep=1&searchStandardNo=TTAS.KO-11.0022&searchCate=TTAS

https://www.tta.or.kr/tta/ttaSearchView.do?key=77&rep=1&searchStandardNo=TTAS.KO-11.0022&searchCate=TTAS

1.5 SRS 개요 (Overview of rest of SRS)

본 문서는 개발할 앱의 전반적인 특성을 정의하고, 앱이 갖추어야 할 기능적 요구사항을 상세히 명시한다. 또한, 비기능적 요구사항 (성능, 보안 등) 및 외부 인터페이스 관련 요구사항을 포함하여 앱 개발의 기준 문서로 활용된다.

2. 일반적인 기술 사항 (General Description)

2.1 제품의 관점 (Product Perspective)

해당 웹사이트의 주요 목적은 복잡하고 긴 여행 일정을 수립하고, 여러 관광지를 추가할 때 발생하는 경로 및 시간 계획의 어려움을 해소하는 데 있다. 사용자는 이 시스템을 통해 효율적이고 최적화된 여행 일정을 손쉽게 작성하고 관리할 수 있도록 한다

2.2 제품의 기능 (Product Functions)

1. 사용자 관리 및 인증 기능

사용자는 이메일 주소 및 비밀번호를 사용하여 회원가입 및 로그인할 수 있어야 한다.

로그인 후 토큰을 발급하고 그에 따라 인증이 되도록 허용할 수 있어야 한다.

시스템은 사용자의 여행 계획 데이터를 안전하게 보관하고 인증된 사용자에게만 접근을 허용해야 한다.

2. 여행 계획 수립 및 관리 기능
사용자는 여행 기간, 목적지, 출발/도착 지점을 설정하여 새로운 여행 계획을 생성할 수 있어야 한다.
사용자는 계획에 포함될 복수의 방문 지점을 여러 API를 통해 검색하고 노드에 추가할 수 있어야 한다.
사용자는 각 방문 지점마다 예상 체류 시간, 운영 시간, 예상 비용 등의 상세 정보를 변경할 수 있어야 한다.
사용자는 수립한 여행 계획을 일자별/시간대별로 요약된 형태로 확인할 수 있어야 한다.
사용자는 계획을 저장, 편집, 복제 및 삭제할 수 있어야 한다.

3. 최적화 및 추천 기능

시스템은 사용자가 입력한 방문 지점들을 기반으로 최단 이동 거리가 되는 경로를 자동으로 계산하여 제안해야 한다 (경로 최적화).
시스템은 방문 지점 간의 교통 상황을 반영하여 최단 이동 시간을 계산하고 표시해야 한다.
시스템은 사용자의 선호도(예: 역사, 미식)와 현재 계획을 바탕으로 추가 방문 지점을 추천할 수 있어야 한다.

해당 웹사이트는 아래와 같은 기능들을 주요 기능으로 갖는다.

- 노드
  - 항공편 검색
  - 호텔 검색 및 예약
  - 호텔 결제 기능
  - 명소 및 레스토랑 검색 및 시간 추천
- 길찾기
  - 유전자 알고리즘을 통해 경로를 생성
- 지도
  - 노드를 표시
  - 유전자 알고리즘을 통해 얻은 경로 표시
- 저장
  - 저장 후 계정에 저장한 경로를 열람 가능

2.3 사용자 특성 (User Characteristics)

웹 사이트의 사용자는 여행 계획을 효율적이고 최적화된 방식으로 수립하고자 하는 개인 여행자이다. 사용자는 다음과 같은 기능을 수행할 수 있어야 한다.

- 개인 여행자
  - 단독 또는 소규모 비동반 여행
  - 시간 및 비용 관리에 민감한 모든 개인 여행자
- 사용자의 목표
  - 가장 합리적인 이동 동선을 확보
  - 계획 수립에 소요되는 시간을 최소화
  - 여행 중 불필요한 이동 시간 낭비를 방지
  - 개인의 취향에 완벽하게 부합하는 맞춤형 여정 수립
- 검색 버튼을 눌러 필요한 노드를 이벍

2.4 제약사항 (Constraints)

- React를 이용해 프론트엔드를 구현하고 Fast API를 이용해 백엔드를 구현한다.
- 많은 양의 리소스를 잡아먹는 경로 검색이나 유전자 알고리즘을 arq를 통해 부담을 줄인다.
- 계정의 아이디와 비밀번호를 관리할 데이터베이스를 위해 mySQL을 사용한다.
- 경로를 만들 때 사용되는 데이터를 관리할 데이터베이스를 위해 REDIS를 사용한다.
- 공항 데이터나 도시 데이터 등을 검색할 때 csv파일을 이용해서 속도와 api 이용량을 줄인다.
- 사용자들의 니즈를 파악하기 위해 최종 확정된 일정을 데이터베이스에 저장한다.
- 안드로이드 최신 버전을 기준으로 한다.
- 기존 갤러리 어플과 중복되는 데이터로 인한 불필요한 데이터 중복을 줄인다.

2.5 가정 및 의존성 (Assumptions and Dependencies)

사용자가 여행을 가는 곳에 가보고 싶은 곳이 어느 정도 존재하는 사람

사용자가 대중교통을 이용하는데 문제가 없는 사람

3. 상세기능 요구사항

3.1.1 기능적 요구사항

< 프로세스 흐름 >

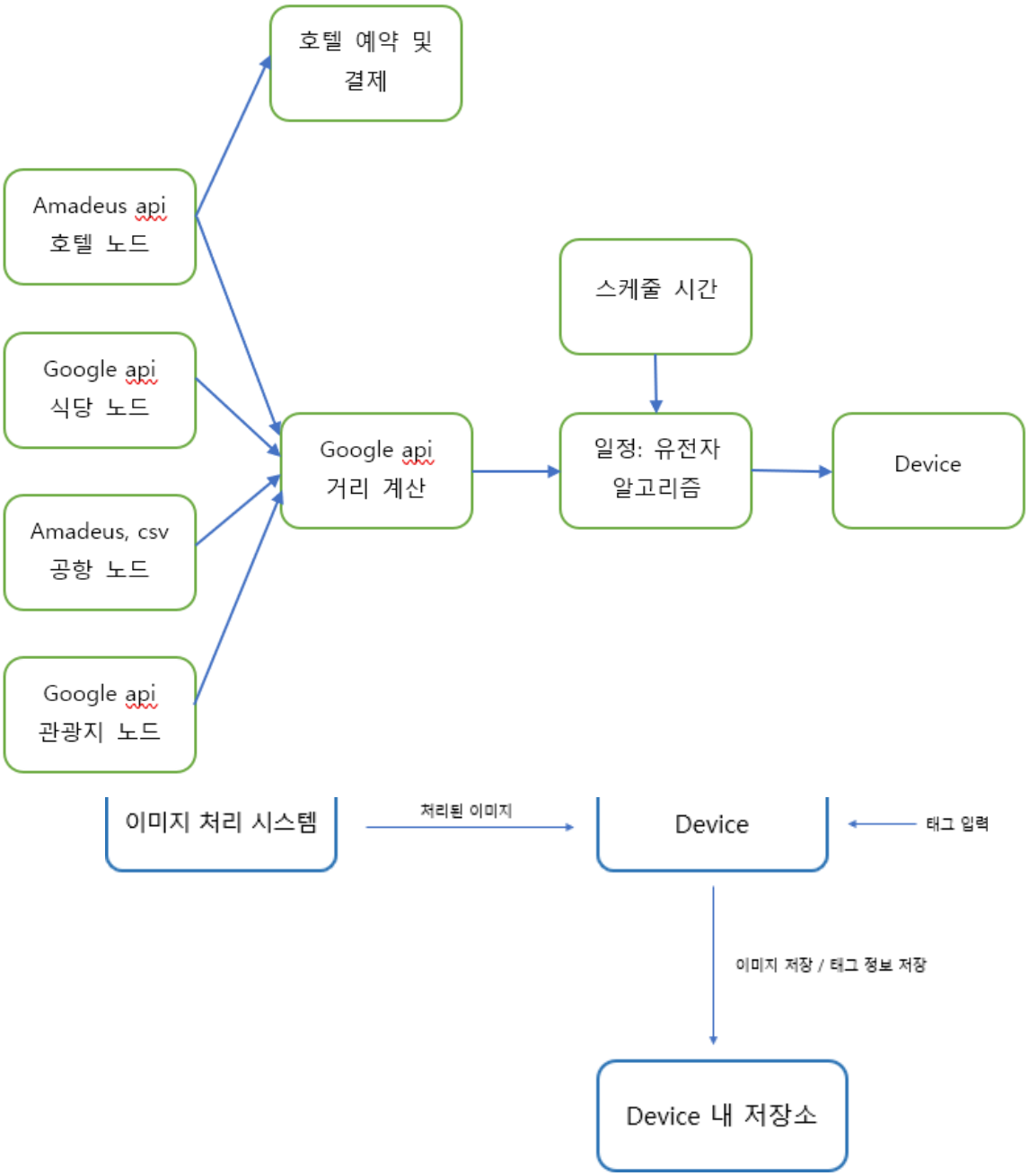

그림 3-2 Device 내 처리된 데이터 저장 (Req-2-1)

3.1.1.1 회원가입과 로그인

구분	내용
요구사항ID	Req-1-1
요구사항명	회원가입
개요	사용자는 아이디와 비밀번호를 사용하여 시스템에 새로운 계정을 생성, 해당 계정 정보를 데이터베이스에 안전하게 저장
입력물	사용할 아이디, 사용할 비밀번호
사전조건	시스템이 MySQL 데이터베이스에 연결
시나리오	(1) 사용자가 유효한 아이디와 비밀번호를 입력하고 회원가입 요청을 전송 (2) 시스템은 중복되는 계정이 없음을 확인, 비밀번호를 Hash 처리 (3) 시스템은 새로운 User 객체를 생성, 데이터베이스에 저장
산출물	회원가입 성공 여부 및 메세지
디자인 제약 사항	비밀번호 입력시에 안보이게끔 처리
예외 시나리오	사용자 이름 중복, 데이터베이스 연결 실패, 비밀번호 최소 길이 제한

3.1.1.1 회원가입과 로그인

구분	내용
요구사항ID	Req-1-2
요구사항명	로그인
개요	사용자가 아이디와 비밀번호를 입력하면, 시스템은 저장된 데이터와 일치하는지 확인, 성공적으로 인증될 경우 유효기간이 설정된 JWT 접근 토큰을 발급
입력물	아이디와 비밀번호
사전조건	사용자가 시스템에 이미 회원가입되어 DB에 계정 정보가 존재 시스템이 MySQL 데이터베이스에 성공적으로 연결
시나리오	(4) 사용자가 작성한 아이디와 비밀번호를 입력하고 로그인 요청을 전송 (5) 시스템은 입력된 username으로 데이터베이스에서 사용자 정보를 조회 (6) 자격 증명이 일치함을 확인, JWT 접근 토큰을 생성
산출물	로그인 성공 여부 및 메세지
디자인 제약 사항	비밀번호 입력시에 안보이게끔 처리
예외 시나리오	틀린 아이디, 틀린 비밀번호, 데이터베이스에 연결이 안됨

3.1.1.2 스케줄 입력

구분	내용
요구사항ID	Req-2
요구사항명	사용자별 일정 목록 임시 저장
개요	인증된 사용자로부터 시작/종료 시간이 포함된 일정 목록을 받아 Redis의 Hash 구조에 임시 저장하고, 요청 시 해당 일정을 순서대로 조회하여 사용자에게 제공
입력물	ScheduleItem:(ScheduleItem은 startDateTime과 endDateTime을 포함)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Redis 클라이언트 연결이 유효 3. 요청 본문이 ScheduleList 모델의 유효성 검사를 통과
시나리오	(1) 인증된 사용자가 일정 목록(ScheduleList)을 POST 요청으로 전송 (2) 시스템은 입력된 일정 목록을 순회하며 순번(0, 1, 2...)을 필드 키로, 일정 객체를 JSON 문자열 값으로 변환하는 맵을 생성 (3) 시스템은 Redis에 맵 데이터를 저장
산출물	입력한 스케줄 저장
디자인 제약 사항	
예외 시나리오	일정 목록이 비어 있는 경우, 일정 데이터가 없는 경우, 인증 실패

구분	내용
요구사항ID	Req-3-1
요구사항명	공항 노드 생성 및 다중 여정 기반 항공권 검색
개요	사용자가 정의한 출발지/도착지 여정 목록, 여행자 유형, 필터 조건을 바탕으로 Amadeus Flight Offers API를 통해 실시간 항공권 정보를 조회
입력물	공항 목록, 여행자 목록, 검색 조건(최대 연결 횟수, 좌석 등급 등)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Amadeus Access Token이 유효하게 획득 3. 요청 본문이 Amadeus api 모델의 유효성 검사를 통과 4. Redis 클라이언트 연결이 유효
시나리오	(1) 인증된 사용자가 검색 조건을 전송 (2) 시스템에서 유효한 Amadeus 토큰을 획득, 정보를 변환 (3) 시스템은 Amadeus 요청을 전송, 항공권 검색 결과를 수신 및 반환
산출물	Amadeus API에서 반환된 항공권 JSON 데이터
디자인 제약 사항	
예외 시나리오	Amadeus API 응답 오류, 네트워크 타임아웃, 인증 실패

구분	내용
요구사항ID	Req-3-2-1
요구사항명	호텔 노드 생성 및 호텔 검색
개요	사용자의 키워드 입력에 따라 Amadeus API를 통해 여행 도시 정보를 검색, 해당 도시의 호텔 목록을 조회, 체크인/아웃 날짜 및 필터 조건에 맞는 유효한 객실 오퍼를 통합하여 사용자에게 제공
입력물	도시 검색: keyword (문자열) 호텔 검색: 도시코드, 체크인 날짜, 체크 아웃 날짜, 성인 수, 객실 수
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Amadeus Access Token이 유효하게 획득 3. 필수 입력 파라미터가 유효성 검사를 통과 4. Redis 클라이언트 연결이 유효
시나리오	(1) 사용자가 도시명을 입력 (2) 시스템은 Amadeus API에서 도시 목록을 조회, 결과를 필터링하여 반환 (3) 사용자가 도시 코드와 날짜를 포함하여 Amadeus api에 요청 (4) Amadeus에서 도시의 호텔 ID 목록과 데이터를 반환 (5) 시스템은 호텔 ID 목록을 청크(Chunk)로 분할해 전달 (6) 가격, 객실 정보, 위경도가 포함된 최종 오퍼 목록을 사용자에게 반환
산출물	도시 검색 성공: 도시명과 iatacode 형식의 JSON 배열 호텔 검색 성공: 호텔 ID, 이름, 가격, 오퍼 ID 등 포함한 객실 오퍼 목록
디자인 제약 사항	호텔 오퍼 조회 시 성능상의 이유로 호텔 ID는 최대 15개로 분할하여 반복 요청
예외 시나리오	Amadeus API 인증 실패, 호텔 목록 조회 실패, 오퍼 조회 실패, 날짜 유효성 오류

구분	내용
요구사항ID	Req-3-3
요구사항명	Google Places API 기반 명소 검색
개요	도시 이름 기반 좌표 변환(Geocoding)과 키워드/위치 기반의 명소 검색을 Google Places API V2를 통해 수행
입력물	좌표 변환: city_name (문자열) 위치 기반 검색: location (위도, 경도 문자열), type (장소 카테고리) 이름 기반 검색: query (검색어), location (선택적 위치)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. GOOGLE_API_KEY 환경 변수가 유효하게 설정 3. type 파라미터는 장소 카테고리 목록에 포함 4. location 파라미터는 위도, 경도 형식의 유효성을 통과
시나리오	(1) 사용자가 city_name을 입력 (2) 시스템은 GEOCODING_API_URL을 호출하여 해당 도시의 위도, 경도 추출 (3) 사용자가 location 및 type을 포함하여 요청 (4) 시스템은 location을 위경도로 변환하고, 함수를 호출 (5) 함수는 Google Places V2 API에 POST 요청을 전송하고, 결과를 수신
산출물	좌표 변환 성공: {"lat", "lng"} 장소 검색 성공: {"places": [...]} 형식의 장소 목록 (이름, 좌표, 평점, 카테고리 등 포함)
디자인 제약 사항	카테고리를 명시적으로 지정하여 필요한 데이터만 요청
예외 시나리오	API 키 누락, 지오코딩 실패, 네트워크/API 오류, 위치 파라미터 형식 오류

구분	내용
요구사항ID	Req-3-4
요구사항명	Google Places API 기반 레스토랑 검색
개요	도시 이름 기반 좌표 변환(Geocoding)과 키워드/위치 기반의 레스토랑 검색을 Google Places API V2를 통해 수행
입력물	좌표 변환: city_name (문자열) 위치 기반 검색: location (위도, 경도 문자열), type (음식 카테고리) 이름 기반 검색: query (검색어), location (선택적 위치)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. GOOGLE_API_KEY 환경 변수가 유효하게 설정 3. type 파라미터는 레스토랑 카테고리 목록에 포함 4. location 파라미터는 위도, 경도 형식의 유효성을 통과
시나리오	(1) 사용자가 city_name을 입력 (2) 시스템은 GEOCODING_API_URL을 호출하여 해당 도시의 위도, 경도 추출 (3) 사용자가 location 및 type을 포함하여 요청 (4) 시스템은 location을 위경도로 변환하고, 함수를 호출 (5) 함수는 Google Places V2 API에 POST 요청을 전송하고, 결과를 수신
산출물	좌표 변환 성공: {"lat", "lng"} 장소 검색 성공: {"places": [...]} 형식의 장소 목록 (이름, 좌표, 평점, 카테고리 등 포함)
디자인 제약 사항	카테고리를 명시적으로 지정하여 필요한 데이터만 요청
예외 시나리오	API 키 누락, 지오코딩 실패, 네트워크/API 오류, 위치 파라미터 형식 오류

3.1.1.3거리 계산

구분	내용
요구사항ID	Req-4
요구사항명	경로 이동 거리 계산 작업 비동기 예약
개요	사용자로부터 요청을 받아, 경로 최적화에 필요한 이동 거리 계산과 같은 시간이 걸리는 작업을 비동기 작업 큐(Arq)에 예약
입력물	**없음** (이전에 저장된 것들 사용)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Arq Redis 연결 풀(arq_redis)이 유효하게 설정 3. GOOGLE_API_KEY가 유효하게 설정
시나리오	(1) Arq Worker가 user_id를 받아 calculate_distances_task를 실행 (2) 시스템은 Redis를 스캔하여 사용자 관련 모든 장소의 위도, 경도를 불러옴 (3) 시스템은 이미 처리된 장소 ID 목록과 현재 불러온 모든 장소 ID를 비교하여 새롭게 추가된 장소만 식별 (4) 모든 조합을 origins와 destinations를 최대 10개로 분할하여 비동기 API 작업 리스트를 생성 (5) 생성된 모든 API 작업들을 최대 동시 호출 제한내에서 실행 (6) Google Distance Matrix API 응답을 수신, 이동 시간 및 거리를 추출하여 레코드 목록을 생성 (7) 파이프라인을 사용하여 redis에 저장 작업을 처리, 리스트에 추가
산출물	노드 사이의 대중교통 이용 시 걸리는 시간)
디자인 제약 사항	
예외 시나리오	Redis 장소 조회 실패, 장소 데이터X, Google API 호출 오류, 데이터 추출 오류

구분	내용
요구사항ID	Req-5
요구사항명	최적화 유형 기반 비동기 경로 생성 및 결과 관리
개요	사용자의 Redis 세션 데이터(장소 및 거리 정보)를 기반으로 유전 알고리즘을 실행하여, 지정된 최적화 유형에 맞는 여행 경로를 생성
입력물	없음(지금까지 redis에 저장된 것)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Arq Redis 연결 풀(arq_redis)이 유효하게 설정 3. Redis에 하나 이상의 유효한 호텔 및 공항 노드와 거리 행렬 데이터가 저장
시나리오	(1) 사용자가 최적화 유형을 포함하여 함수에 요청 (2) 시스템은 최적화 타입을 통해 유전자 알고리즘을 Arq 큐에 예약 (3) Arq Worker는 작업을 수신, Redis에서 모든 노드와 거리 데이터를 불러옴 (4) 시스템은 일자별 시간 제약을 고려하여 유전 알고리즘을 반복 실행 (5) GA는 활동, 효율성, 균형 점수를 계산하고, 타입에 따라 가중치를 부여하여 최적 경로를 탐색 (6) 최종적으로 선정된 최적 경로를 시간표 형식으로 변환 및 임시 저장
산출물	정렬된 일자별 경로 목록 (노드 이름, 시간, 점수 포함)
디자인 제약 사항	NumPy 배열 기반으로 처리, 사용자 요청과 분리된 비동기 큐(Arq)를 통해 처리
예외 시나리오	필수 노드 누락, 유효 경로 없음, 작업 중 오류, 작업 ID 오류, 토큰 만료/인증 실패

구분	내용
요구사항ID	Req-5
요구사항명	최신 경로 데이터 조회 및 다일정 지도 렌더링
개요	인증된 사용자의 최신 저장 경로 데이터를 백엔드 API를 통해 조회, Google Maps API를 사용하여 일자별 경로선과 마커를 시각화
입력물	없음(이미 저장되어 있음)
사전조건	1. 사용자가 유효한 JWT 토큰을 통해 인증 2. Google Maps API 스크립트 로드 성공 및 window.google.maps 객체 사용 가능 3. 유효한 경로 데이터가 백엔드에 저장
시나리오	(1) 마지막 경로를 조회하고, 해당 ID의 일자별 경로 데이터를 Redis에서 가져와 JSON으로 반환 (2) 클라이언트API 응답 데이터를 수신 (3) Directions Service를 사용하여 일자별 경로 요청 및 DirectionsRenderer로 지도에 경로선 렌더링 (4) 지도 뷰포트를 전체 경로가 보이도록 조정(fitBounds)
산출물	화면에 렌더링된 지도, 일자별 색상 구분된 경로선
디자인 제약 사항	모든 일자별 경로는 DAY_COLORS를 순환하며 고유한 색상으로 구분
예외 시나리오	인증 오류/토큰 만료, 데이터 없음, Directions API 실패, API 키 무효

3.1.1 비기능적 요구사항

- 핵심기능 관련 비기능 요구사항 개요

해당 애플리케이션은 일정을 생성해주는 기능이 핵심인 만큼, 정확도와 속도가 중요하다. 10초 이내의 속도가 이상적이나 사용 가능한 하드웨어를 고려하여 20초 이내의 일정 생성을 목적으로 한다. 또, 노드(관광지, 레스토랑)을 검색하는 기능에 있어서 사용자의 성향을 위해 최대한 여러 카테고리 가 가능하도록 한다.

- 품질 속성에 따른 비기능 요구사항

품질 속성	비기능 요구사항	우선 순위
사용성	이해하기 쉬운 UI를 통해 사용자가 처음으로 웹 사이트를 이용해도 바로 조작 가능하도록 해야함	상
성능	경로 시간 계산 10초 이내 일정 결과 표시 20초 이내	상
변경 가능성	해당 웹은 유전자 알고리즘을 기반으로 하고 있지만 향후 이용자들의 선호하는 데이터가 쌓이면 구글 api가 아닌 실제 데이터를 통해 머신러닝을 통한 새로운 시스템의 추가가 가능해야 함	중
이식성	해당 애플리케이션은 웹 기반으로 제작하며, 모바일OS에서의 구동은 현 재 고려하고 있지 않음	하

3.2 외부적인 인터페이스 요구사항

이 항목에서는 애플리케이션의 외부 및 내부 요소 간의 인터페이스에 대한 요구사항을 정의

인터페이스 요구사항 ID	인터페이스 요구사항명	인터페이스 요구사항 설명	우선순위		
			중요도	난이도	우선순위
IR-001	Amadeus 인증	Amadeus API 호출을 위한 토큰을 획득하고 캐싱하여 관리	상	하	3
IR-002	Redis 연결	세션, 캐싱, ARG 위해 Redis 연결을 유지	상	하	2
IR-003	사용자 계정 관리	DB를 통해 회원가입/로그인을 처리하고 사용자 인증 상태를 관리	상	하	1
IR-004	스케줄 생성	여행 시작/종료 날짜와 일자별 시간 제약을 설정하여 Redis에 임시 저장	상	중하	6
IR-005	지도 생성	Google Maps API를 사용하여 생성된 경로를 일자별 색상으로 구분하여 지도에 시각화	중	상	21
IR-006	관광지 검색	Google Places API를 통해 위치/이름 기반의 명소를 검색	중상	중	9
IR-007	레스토랑 검색	Google Places API를 통해 위치/이름 기반의 레스토랑를 검색	중상	중	10
IR-007	거리 계산	Redis에 저장된 모든 장소 쌍 간의 Google Distance Matrix 기반 이동 시간 및 거리를 계산	중상	중	11
IR-008	공항 검색	Amadeus API를 통해 출발/도착 공항 정보 검색	상	중	7
IR-009	공항 CSV 파일	공항 검색 시 내부 CSV 파일을 우선적으로 참조하도록 수정	중	중하	20
IR-010	항공편 검색	Amadeus API를 통해 항공편을 검색	중	중상	15
IR-011	호텔 검색	Amadeus API를 통해 도시별 호텔 오퍼를 검색	중상	중상	8
IR-012	호텔 예약	Redis에서 오퍼 정보를 불러와 최종적으로 Amadeus API를 통해 호텔 예약을 실행	중	상	16
IR-013	유전자 알고리즘	여행 노드 및 거리 행렬을 사용하여 유전 알고리즘 기반의 최적 경로를 비동기로 생성	상	상	12
IR-013	호텔 결제 요청	아임포트 SDK를 사용하여 프런트엔드에서 결제 모듈을 호출하고 서버에 결제 준비를 요청	중	중	17

IR-014	호텔 결제 검증	서버에서 아임포트 API를 통해 결제 금액 위변조 여부를 검증하고, 최종 예약을 실행	중하	중상	18
IR-015	호텔 예약 및 주문 내역 저장	결제 및 Amadeus 예약 성공 시, DB에 예약 내역을 확정 상태로 저장	중하	하	19
IR-017	컴포넌트 간 내비게이션	React Router를 사용하여 여행 계획 단계별로 페이지 이동을 관리	상	하	5
IR-018	세션 기반 상태 표시	사용자 로그인 여부에 따라 UI 버튼 및 접근 권한을 동적으로 표시	상	하	4
IR-019	출력 포맷 통일	모든 검색 결과(항공편, 호텔, 장소)의 데이터 구조 및 UI 표시 형식을 일관되게 유지	상	중	13
IR-020	데이터 정리	세션 종료 또는 사용자 요청 시 영구 경로 외의 모든 임시 세션 데이터를 Redis에서 삭제	하	하	14

4. 성능처리 요구사항

이 항목에서는 하위 5가지 항목으로 나누어서 웹사이트가 달성해야 할 성능 요구 사항 및 성능 목표를 기술한다.

4.1 처리량(Throughput)

거리 계산에서 하나의 google distance matrix api가 동시에 작동할 수 있는 수는 30개로 고정되어 있으므로 처리량은 30명으로 고정되어 있다.

4.2 동시 접속 수(Concurrent Session)

거리 계산에서 하나의 google distance matrix api가 동시에 작동할 수 있는 수는 30개로 고정되어 있으므로 모두가 한번에 거리를 계산한다는 경우하에 동시 접속 수는 30명으로 고정되어 있다. (만약에 사이트가 커지면 데이터를 축적해 한계를 극복할 수 있을 것이다.

4.3 응답 시간(Response Time)

Google place api평균 300ms 이내, Amadeus 평균 800ms 이내, Google Distance Matrix 1,000ms
유전자 알고리즘 최대 3000ms이내

4.4 리소스 사용량(Memory/CPU/Disk/Network Usage)

CPU은 지속적 평균 70% 미만으로 설정, Numba 기반의 유전 알고리즘은 CPU를 집중적으로 사용하지만, 시스템 안정성을 위해 워커 프로세스의 CPU 사용률을 제한한다
메모리 사용량 (Redis) 최대 80% 미만으로 설정한다, Redis가 인메모리 데이터베이스임을 고려하여, 메모리 용량의 80%를 넘지 않도록 관리합니다.

4.5 실패율(Fail Rate)

유전자 알고리즘의 경로 생성 확률이 최소 98% 이상이어야 한다.