2019

Design Specification

ECO-TRAVEL

Student Number	Name
2014314081	김주한
2015310965	류민재
2016314421	나빌
2015314230	선명우
2015310434	권태완

TABLE OF CONTENTS

1. Preface

This chapter defines the expected readership of the document, and briefly introduces the purpose of each document structure. This chapter also describes version history including a rational for creation of a new version and a summary of the changes made in each version.

1.1 Readership

This document is intended to be read by a variety of stakeholders who are included in the process of system develop and maintenance. Examples of stakeholders can be ranged from software engineers, system architectures to anyone who is concerned with the development process.

1.2. Document Structure

A. Introduction

본 문서를 서술하는 데 사용된 다양한 다이어그램과 표현 도구들에 대해 설명하고, 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 설명한다.

B. System Architecture

시스템의 각 서브시스템으로 분류하고, 각각의 구조를 개괄적으로 기술한다. 시스템의 전체 기능이 각 서브시스템으로 어떻게 분산 및 할당되었는지 설명한다.

C. Protocol Design

시스템의 각 컴포넌트, 특히 프론트엔드 시스템과 백엔드 시스템간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

D. Database Design

Requirements 문서에서 규정된 데이터베이스 요구 사항을 기반으로, 각 데이터 엔티티의 속성과 관계를 ER diagram을 통해 표현하고 최종적으로 Relational Schema, SQL DDL를 작성한다.

E. Testing Plan

요구사항 명세서에서 밝혔던 verification 과 validation 검사를 시행하는 것이 목적이다. Test의 단계적 실행계획을 밝히고, 각각의 단계에 맞춘 Test Case를 사전에 작성한다.

F. Development Plan

시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.

G. Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2. Introduction

This chapter introduces the various diagrams and tools used in the design of this system, and describe the scope of development of this system.

2.1. Applied Diagram

A. UML

본 소프트웨어 시스템 설계에서는 UML의 Class Diagram, Sequence Diagram, State Diagram을 사용하여 개발할 시스템의 구조를 시각화 한다. UML(UML, 통합 모델링 언어, 영어: Unified Modeling Language)은 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어이다. 이 표준은 UML을 고안한 객체 관리 그룹에서 관리 하고 있다. UML은 소프트웨어 집약 시스템의 시각적 모델을 만들기 위한 도안 표기법을 포함한다. 통합 모델링 언어는 객체 지향 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화할 때 사용한다.

B. Package Diagram

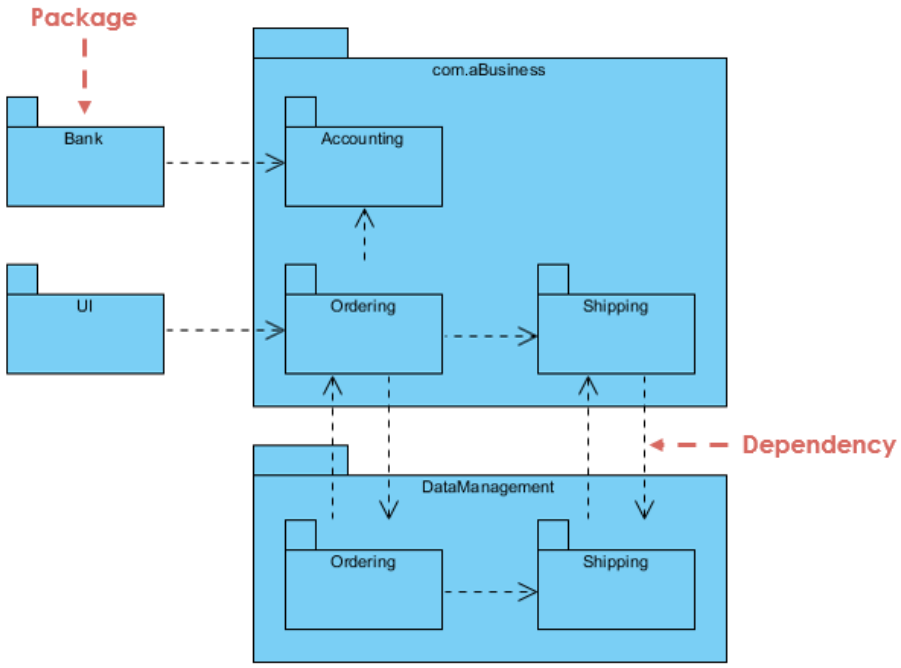

Figure 1 : Package Diagram Example

Package란 해당 프로그램상에서 프로그램 컴포넌트로 나누었을 때, 개발자가 관리하기 쉬운 단위의 작은 집합이라고 볼 수 있다. 패키지는 기능적으로 관련된 여러 하위 class들로 이루어져있으며, 패키지 단위는 독립적인 형태로 다른 시스템에 응용하여 사용할 수 있다. 패키지는 UML 시스템 모델의 기본 구성 요소로, 전체 시스템이라 함은 다른 모든 패키지들, 다이어그램 및 요소를 포함 하는 개념으로 볼 수 있다. 본 문서 상의 패키지 다이어그램은 각 class별 속성(Attribute) 및 해당 클래스의 기능(Methods)을 보여주고 상속 등 클래스와 클래스 간의 관계를 나타낸다.

C. Deployment Diagram

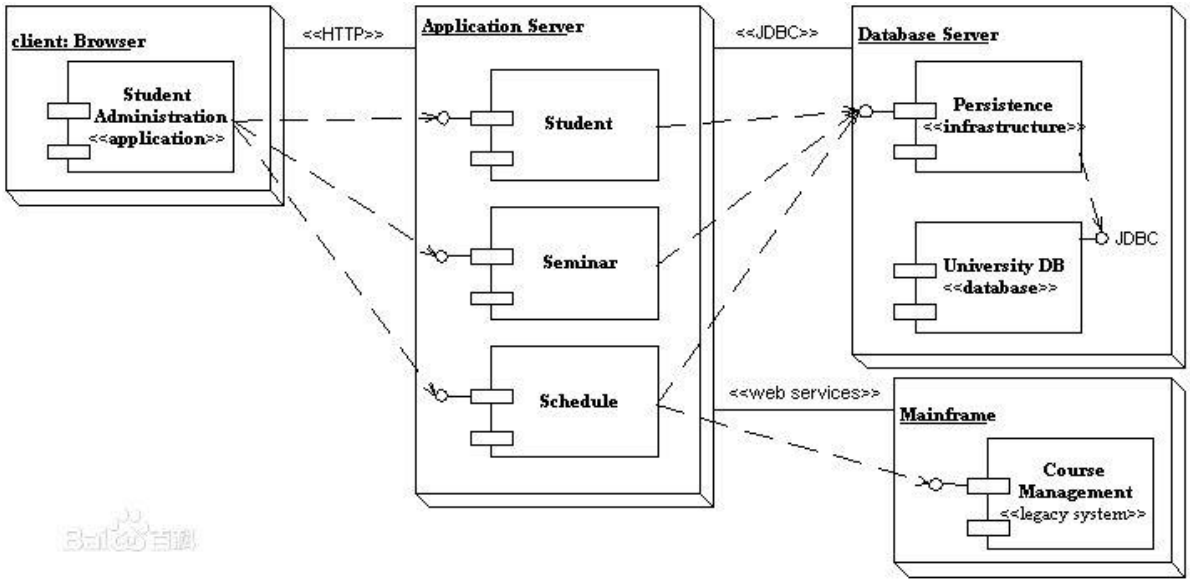

Figure 2 : Deployment Diagram Example

Deployment diagram이란 해당 시스템을 작동하기 위해 사용되는 네트워크, 하드웨어 및 소프트웨어 등이 어떻게 배치되어 있고 이러한 것들 간의 관계가 어떻게 나타나는 지를 보여주는 diagram을 말한다. 이들의 배치 상태를 다이어그램으로 표현함으로써 시스템의 구성요소 및 리소스에 대한 내용을 보다 빠르게 이해할 수 있게 된다. Deployment diagram 상에서 사용되는 도형은 직육면체로 시스템에서 업무를 처리하는 능력을 가진 장치의 단위인 노드를 의미한다. 해당 다이어그램은 노드의 단위가 정해지고, 컴포넌트가 정의된 뒤 하드웨어의 사양이 확정되는 시점에 작성되어야 하므로, 시스템이 전부 완성된 뒤에 작성된다.

D. Class Diagram

클래스 다이어그램이란, 객체지향 프로그램 상에서 가장 기본이 되는 단위인 class들이 주체가 되어 나타나는 다이어그램이다. 해당 클래스에 대한 상세한 정보를 담고 있다. 예컨대, class의 속성값(Attribute), 메서드(Method) 과 해당 클래스에서 상속받고 있는 클래스, 구현한 인터페이스 등의 클래스와 관련된 내용은 모두 보여지게 된다. 클래스 다이어그램을 바탕으로 실제 코드를 생성하게 되므로 구체적으로 나타나게 된다.

E. State Diagram

State Diagram은 event-oriented diagram으로 시스템 내에서 해당 이벤트가 어떻게 작동하는 지를 다이어그램으로 상세히 기술하게 된다. 주로 상태가 변화하여 어떤 결과값이 도출된다거나 상태가 다른 상태로 변화하는 경우, 시스템은 어떻게 돌아가는 지를 상태 중심적으로 표현한다. 따라서 해당 다이어그램을 통해 시스템 내에 어떤 상태가 존재하는지, 상태의 변화가 어떤 형태로 일어나게 되는 지를 알 수 있으며, 상태에 대한 시스템의 반응도 알아볼 수 있다.

F. Sequence Diagram

Sequence Diagram은 시스템 내에서의 각 컴포넌트들이 주고 받는 메시지의 흐름을 시간 순차적으로 표현하는 상호작용 다이어그램이다. 상호작용 다이어그램은 다이어그램의 수직 방향이 시간을 나타내고, 메시지 교류의 주체인 객체와, 객체를 통해 실질적인 데이터를 주고 받는 메시지(오퍼레이터)를 보여주는 역할을 하며, Sequence Diagram이란 시스템과 그의 Actor, 그리고 System Components간의 상호작용을 나타내는 UML Diagram이다. 이는 시스템의 특정 use case에서 일어나는 상호작용들의 sequence로 표현된다.

G. ER Diagram

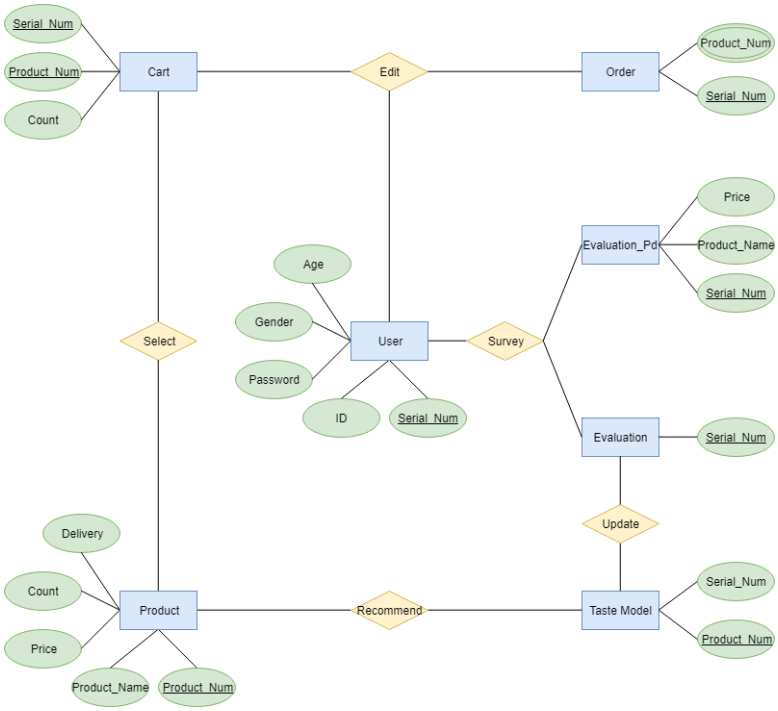

Figure 3 : ER Diagram Example

ER Diagram은 데이터베이스에서 사용되는 다이어그램으로, 각 개체들 간의 관계를 표현하고 있다. 앞선 다이어그램들과 달리 UML에서 지정된 다이어그램은 아니며, 이와는 별개로 시스템의 데이터베이스를 구성하는 과정에서 entity간의 관계를 도형을 통해 알아볼 수 있다. ER은 entity-relationship으로, 하나의 개체(entity)는 분리된 물체 하나를 표현한다. 개체는 사각형으로 표현되며, 관계(relationship)는 다이아몬드로 표현된다. 개체나 관계는 특성을 가질 수 있으며, 이 특성들은 관계 집합에 실선으로 연결된 타원형으로 표현한다

2.2 Applied Tools

A. Draw.io

Draw.io는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 직접 다이어그램에 사용하기 위해 도형을 만들 필요가 없다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있기 때문에 도형을 정렬하기 편리하다. 이 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

2.3 Project Scope

Scope Description	In Scope: - 여행 일정 공유 시스템 - 영수증 자동인식처리 시스템 - 소비지도(PayMap) 가시화 시스템
	Out of Scope: - 여행지 추천 시스템 - 여행 예산 예상 시스템 - 여행물품 연계 서비스(가격비교 등)
Project Deliverables	안드로이드, ios 두 플랫폼 모두에서 동작하는 여행가계부 App
Acceptance Criteria	- Performance of main subsystem - Usability - Customer satisfaction
Constraints	팀원 모두 안드로이드, ios App 개발 경험이 없음 팀원 미팅 시간 조율의 어려움 주어진 개발기간이 길지 않음
Assumptions	혼자 여행가는 여행객보다 친구와 가족과 함께 여러명이서 여행을 가는 경우에, 이 서비스가 완벽히 구현이 된다면, 일정공유와 여행가계부 작성에 있어서 어려움을 덜어줄 것으로 예상됨. 팀원 미팅 시간을 조율하기 어려운 점은 제약조건이자 전제사항이므로, git을 통하여 각각의 sub-system을 한명씩 맡아서 개발하는 식으로 진행.

Table 1 : Project Scope

3. System Architecture – Overall

This chapter describes the overall architecture of the system. In detail, it describes the overall structure of the system, the composition of each subsystem, and the relationship between subsystems, and each structure attaches a diagram to aid understanding.

3.1 System Organization

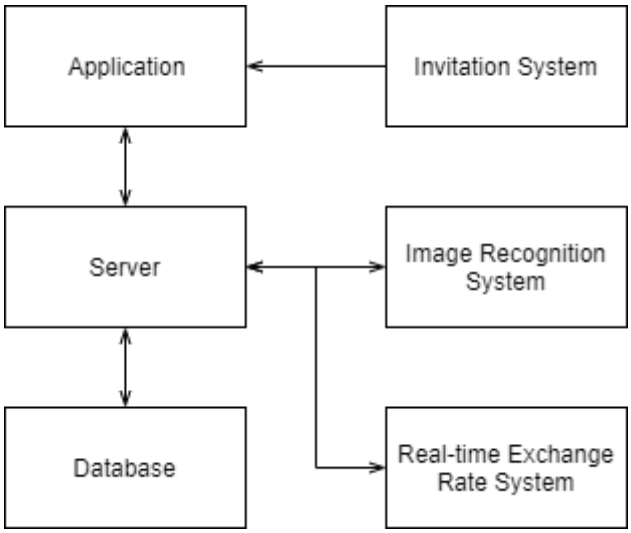

Diagram 1 : System Organization

본 프로젝트는 클라이언트-서버 모델을 적용해 설계했으며, Application이 프론트엔드로서 사용자와의 모든 상호작용을 맡고, 프론트엔드 Application과 백엔드 Server는 JSON 형식을 기반으로 한 HTTP 통신으로 데이터를 송수신한다. 백엔드 Server는 Application으로부터 들어오는 각 요청을 컨트롤러로 분배하고, 필요한 객체 정보를 데이터베이스로부터 읽어와 Application으로 전달하거나 Application에서 넘어온 정보를 데이터베이스에 저장한다.

Application은 여행 동행자를 초대할 때 Invitation System을 실행한다. 이 시스템은 카카오톡 등의 외부 시스템을 연동해 다른 사용자가 함께 여행 정보를 생성하고 수정할 수 있도록 Application으로 정보를 전송한다.

Server는 Application에서 요청이 있을 시 Image Recognition System이나 Real-time Exchange Rate System을 실행한다. Image Recognition System은 소비한 금액을 데이터베이스에 저장할 때, 사용자가 모든 항목을 입력하지 않고 영수증을 인식시켜 입력을 자동화할 수 있게 한다. 자동화된 데이터는 다시 Application으로 전송되어 사용자가 확인한 후 최종적으로 데이터베이스에 저장한다. Real-time Exchange Rate System은 주기적으로 Server에서 미국 1달러 기준 화폐 별 금액을 웹에서 크롤링하는 시스템이다. 크롤링을 통해 각 화폐 별 금액을 Server에 저장해 두고, 사용자가 예산을 책정하거나 소비한 금액을 저장할 때 사용자가 저장하길 원하는 통화와 데이터화된 금액의 통화가 다를 경우 저장해 둔 금액을 기준으로 환산해 결과값을 전송한다.

B. Frontend Application

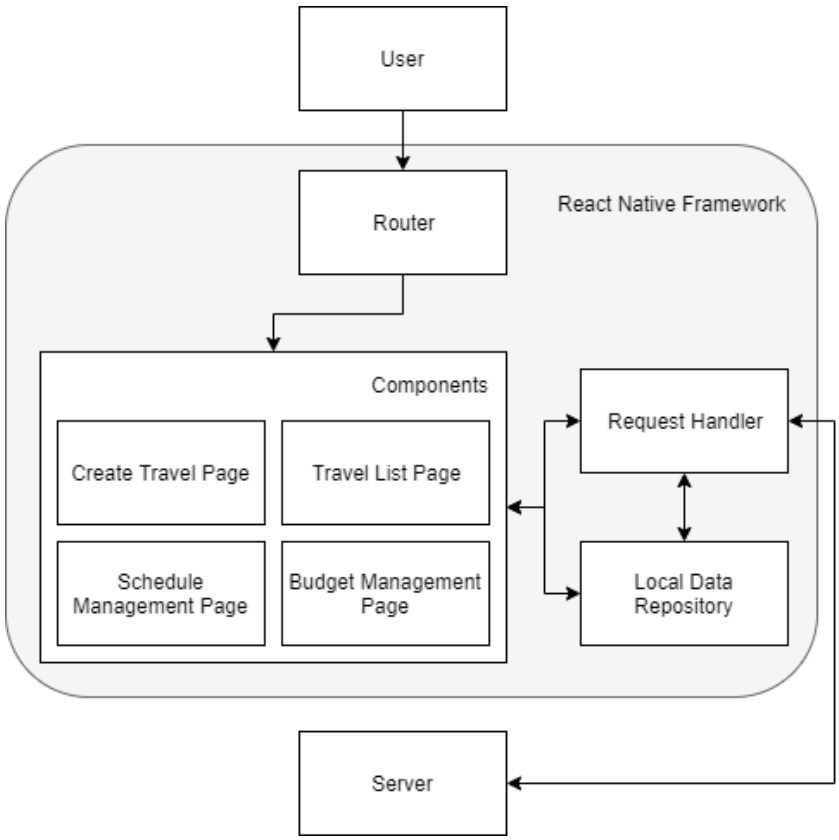

Diagram 2 : Frontend Application

사용자와 상호작용하는 프론트엔드 시스템의 경우, Android와 iOS에서 모두 적용 가능한 Java Script 기반의 React Native 프레임워크를 사용해 각 컴포넌트들을 관리한다. Create Travel, Travel List, Schedule Management, Budget Management 컴포넌트가 시스템의 주요 기능들을 담당하고, 유저 세션 등 프론트엔드에서 자주 쓰이는 Resource들에 대해서는 Local Data Repository에 저장하며 서버 통신에서 발생하는 I/O 지연을 최소화한다. 서버와 통신해야 하는 경우, Request Handler를 통해 데이터를 처리한다.

C. Backend Application

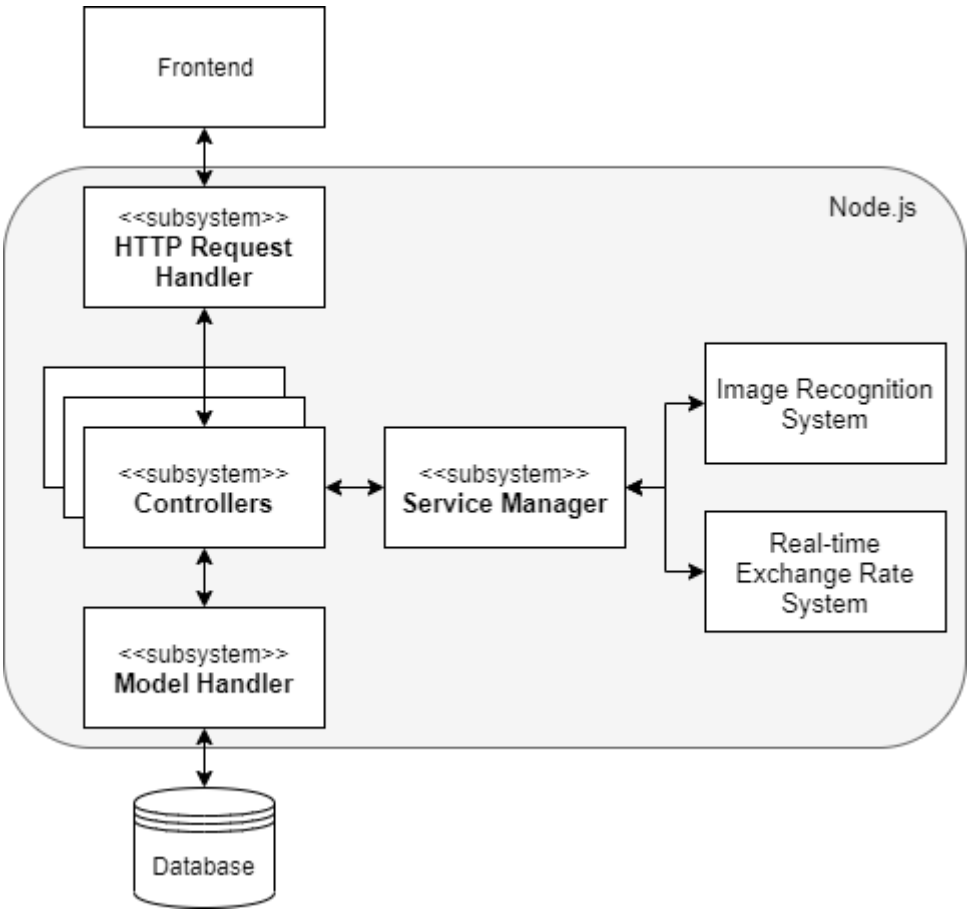

Diagram 3 : Backend Application

프론트엔드의 요청에 따라 데이터를 처리하는 서버의 구조는 위와 같다. 프론트엔드가 Java Script 기반의 React Native 프레임워크를 사용하기 때문에 서버 또한 Java Script 기반의 Node.js 프레임워크를 사용해 개발한다. 서버는 앱에서 받은 HTTP 요청을 해당하는 컨트롤러에 보내 처리한다. 필요한 경우 서비스 핸들러를 통해 이미지 인식이나 환율 계산을 마친 후 모델 핸들러를 거친다. GET/POST 등 요청의 종류에 따라 데이터베이스에 저장되거나 저장된 데이터를 앱으로 보내 사용자가 읽을 수 있도록 한다.

4. System Architecture - Frontend

This chapter describes the structure of the front-end system, which is responsible for interacting with users, and the configuration of each component, and the relationship between components.

4.1 Subcomponents of Frontend

A. Showing Travel List

1. Class Diagram

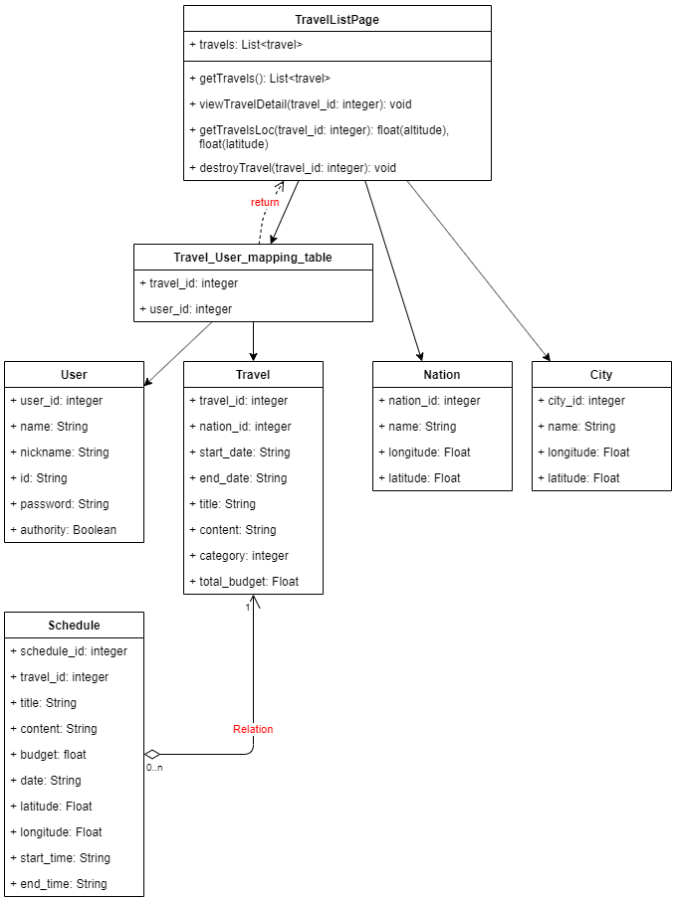

Diagram 4 : Showing Travel List Class Diagram

1.1 TravelListPage – 여행목록 페이지(핸들러)

i. attributes

+travels: 유저가 생성한 여행객체들

ii. methods

+getTravels(): 유저가 생성한 모든 여행객체들을 불러온다.

+viewTravelDetail(travel_id: integer) : 생성한 여행객체들 중 특정 여행객체만을 선택하여 불러온다.

+getTravelLoc(travel_id: integer): 생성한 여행객체들 중 특정 여행객체의 여행국가, 도시의 위도, 경도값을 가져온다.

+destroyTravel(travel_id): 특정 여행객체를 삭제한다. 하지만, 한 명의 유저가 여행객체를 삭제하였다고, 곧바로 DB의 여행객체가 삭제되는 것은 아니다. 여러 명이 같이 여행객체를 공유하고 있는 경우, 삭제를 진행한 유저만이 유저-여행 매핑 테이블에서 삭제되어, 나머지 여행일행들은 여행객체에 여전히 접근이 가능하다.

1.2 Travel_User_mapping_table – 여행객체와 유저객체를 연결시키는 테이블객체

i. attributes

+travel_id: 여행객체의 아이디값이다.

+user_id: 유저의 아이디값이다.

User와 Travel은 m:n관계이므로 매핑테이블을 설정함으로써 연결한다.

1.3 User – 유저객체

i. attributes

+user_id: 유저 아이디(primary key)

+name: 유저 이름

+nickname: 유저 닉네임

+id: 유저 아이디(이메일)

+password: 유저 비밀번호

+authority: 유저 권한

1.4 Travel – 여행객체

i. attributes

+travel_id: 여행 아이디(primary key)

+nation_id: 여행지 국가 아이디

+start_date: 여행 출발날짜

+end_date: 여행 마지막날짜

+title: 여행 제목

+content: 여행 컨셉, 이야기 내용 등

+category: 여행 종류(1.혼자 2.가족 3.친구 4.연인)

+total_budget: 여행 총 예산

1.5 Schedule – 여행일정객체

i. attributes

+schedule_id: 여행일정 아이디(primary key)

+travel_id: 여행객체 아이디(Travel:Schedule=1:N)

+title: 여행일정 제목

+content: 여행일정 내용

+budget: 여행일정 예산

+date: 여행일정 날짜

+start_time: 여행일정 시작시간

+end_time: 여행일정 종료시간

+latitude: 여행일정 위도

+longitude: 여행일정 경도

1.5 Nation – 국가객체(약 200여개, 고정값이므로 서버가 아닌 프론트엔드에 저장)

i. attributes

+nation_id: 국가 아이디

+name: 국가 이름

+altitude: 국가 위도

+latitude: 국가 경도

1.6 City – 도시객체(주요여행도시만 고려, 위와 마찬가지로 프론트엔드에 저장)

i. attributes

+city_id: 도시 아이디

+name: 도시 이름

+altitude: 도시 위도

+latitude: 도시 경도

2. Sequence Diagram

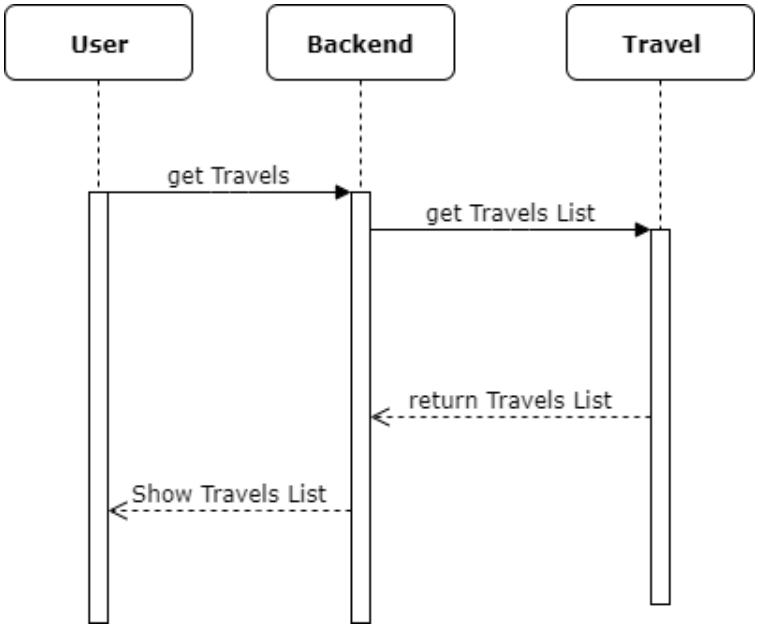

Diagram 5 : Showing Travel List Sequence Diagram

B. Creating New Travel

1. Class Diagram

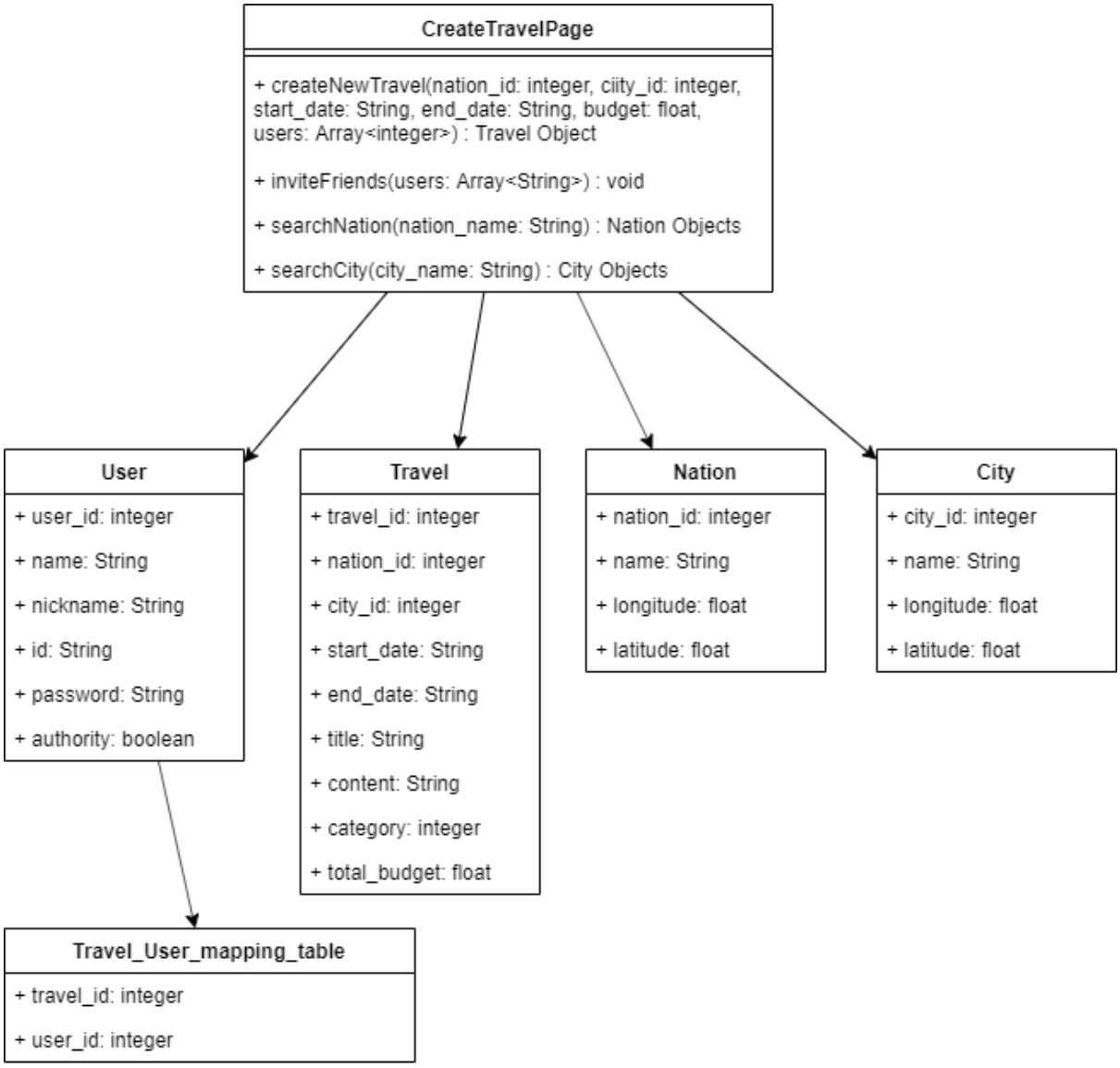

Diagram 6 : Creating New Travel Class Diagram

1.1 CreateTravelPage – 새 여행 만들기 페이지(핸들러)

i. Methods

+createNewTravel(nation_id, city_id, start_date, end_date, budget, users): 조건에 맞게끔 입력값을 받아서 새로운 여행객체를 만들어 DB에 저장한다.

+inviteFriends(users) : 여행을 같이갈 친구에게 초대메세지를 보낸다. 친구가 이를 수락하면, user-travel-mapping-table에 매핑되어 기록된다. 비동기적으로 동작한다.

+searchNation(nation_name): 유저가 검색한 국가를 반환한다. 비동기적으로 동작한다.

+searchCity(city_name): 유저가 검색한 도시를 반환한다. 비동기적으로 동작한다.

ii. 그 이외의 class는 A. Showing Travel List와 같기 때문에 설명을 생략한다.

2. Sequence Diagram

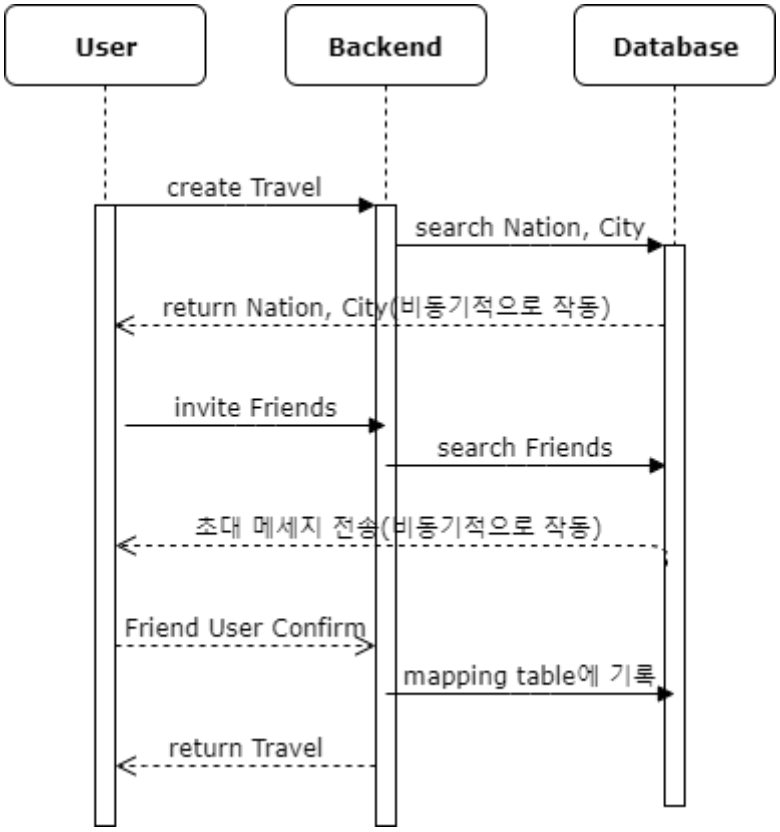

Diagram 7 : Creating New Travel Sequence Diagram

C. Travel Schedule Management

1. Class Diagram

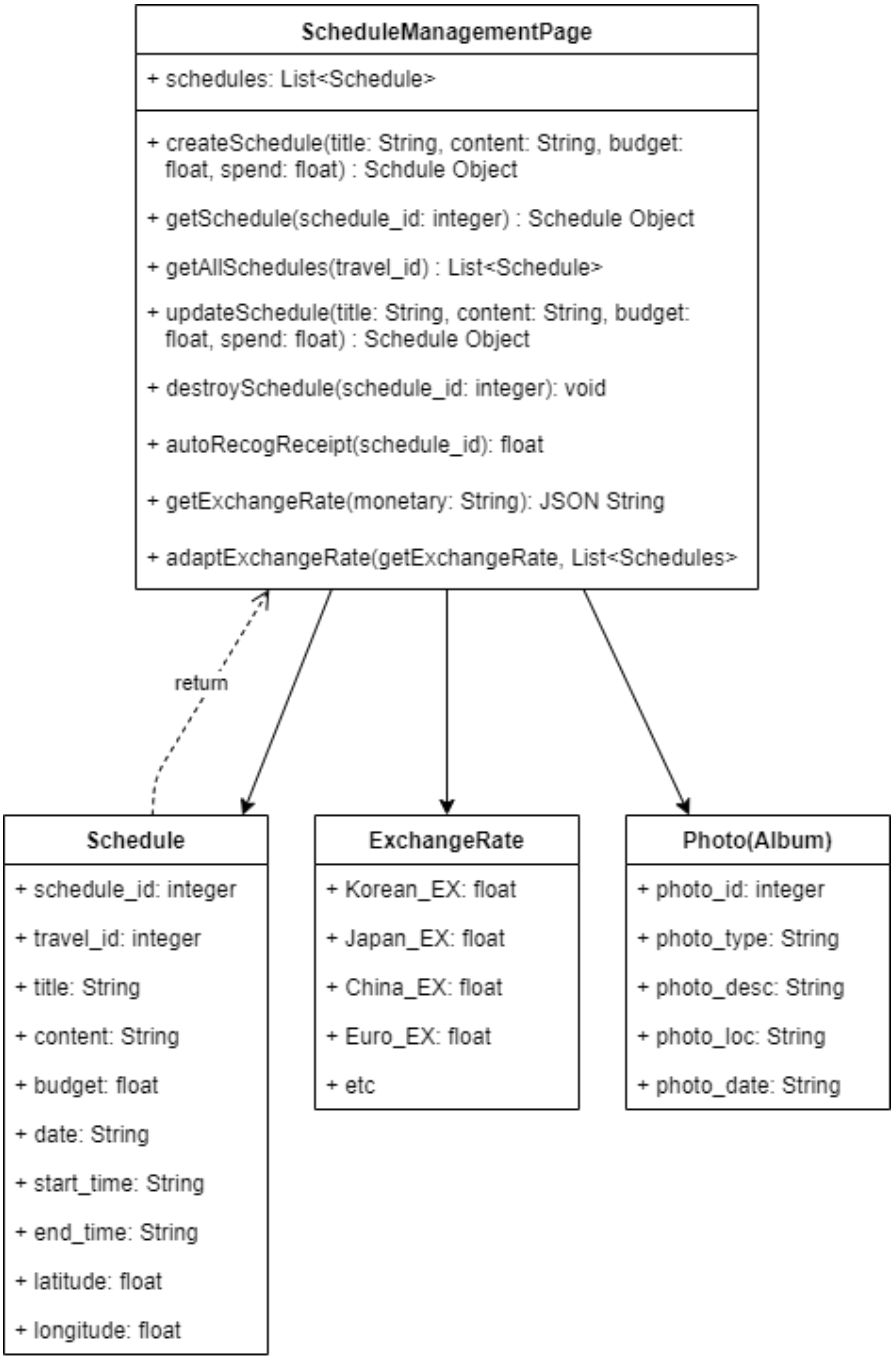

Diagram 8 : Travel Schedule Management

1.1 ScheduleManagementPage – 여행목록 페이지(핸들러)

i. Attributes

+schedules : 특정여행에 대한 모든 schedule

ii. methods

+createSchedule(title, content, budget, float, spend) : 특정 여행객체에 대한 schedule을 생성한다. (Schedule:Travel=N:1)

+getSchedule(schedule_id) : 특정 여행에 대한 schedule객체를 전부 반환한다.

+getAllSchedules(travel_id) : 특정 여행에 대한 모든 schedule객체를 전부 반환한다.

+updateSchedule(title, content, budget, float, spend) : schedule 수정

+destroySchedule(schedule_id) : schedule 삭제

+autoRecogReceipt(schedule_id) : 특정 schedule에 대한 spend를 입력시에, 유저가 영수증 촬영/사진을 제시하면 소계를 자동 입력하도록 한다.

+getExchangeRate(monetary) : 특정 화폐에 대한 환율 반환

+adaptExchangeRate(getExchangeRate, List<Schedule>) : 환율 적용

1.2 ExchangeRate – 환율 정보를 가지고 있는 entity

i. Attributes

* 미 1달러에 대한 환율을 각국 별로 리스트화, 서버에서 주기적으로 갱신

+Korean_Ex : 원달러 환율

+Japan_Ex : 엔달러 환율

+China_Ex : 위안달러 환율

+Euro_Ex : 유로달러 환율

+ etc : 그 외 환율 등을 위와같이 기록한다.

1.3 Photo(Album) – 사진 entity (사진에 대한 정보를 통하여 pay-map 작성)

i. Attributes

+photo_id : 사진 고유 아이디

+photo_type : 사진 타입

+photo_desc : 사진 설명

+photo_loc : 사진 촬영 장소

+photo_date : 사진 촬영 날짜

2. Sequence Diagram

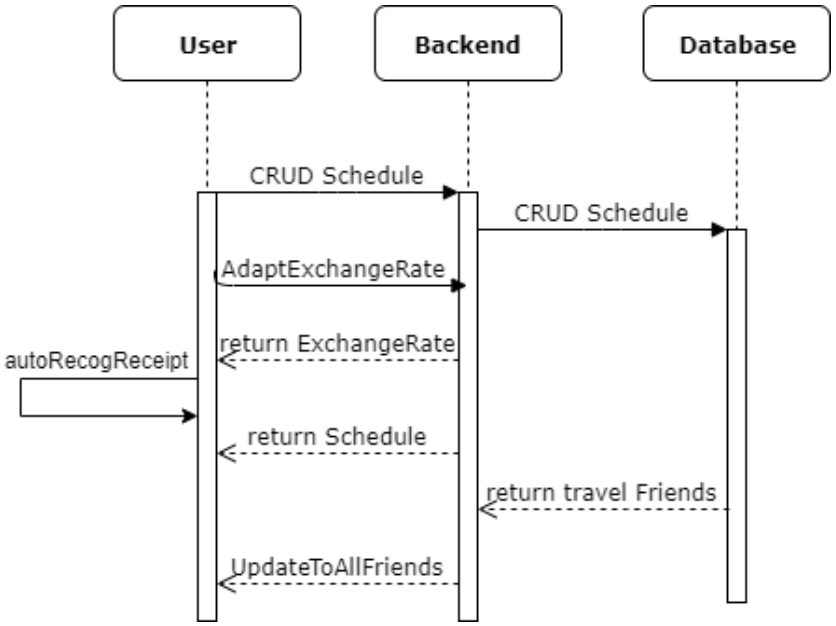

Diagram 9 : Travel Schedule Management Sequence Diagram

D. Budget Management

1. Class Diagram

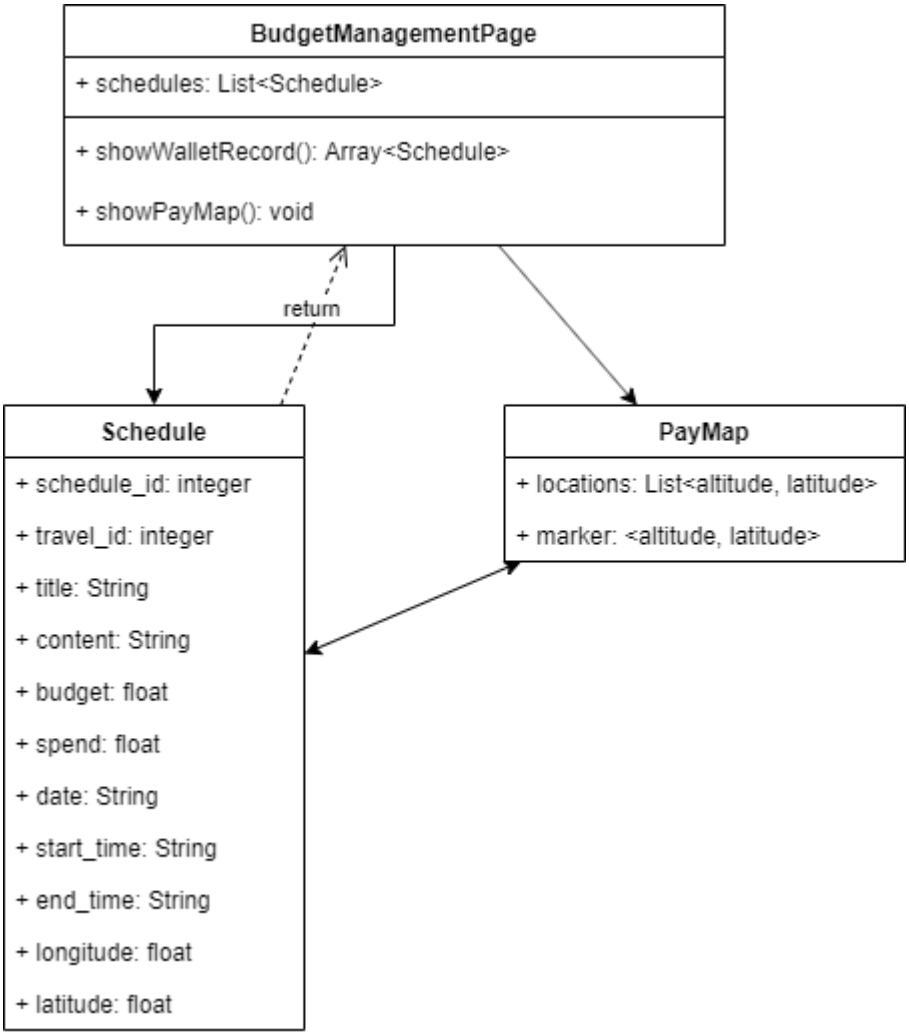

Diagram 10 : Budget Management Class Diagram

1.1 BudgetManagementPage – 예산소비관리 페이지(핸들러)

i. Attributes

+schedules : 특정 여행에 대한 모든 일정들

ii. Methods

+showWalletRecord(): 여행 날짜별로 예산소비 내역을 한눈에 보여준다.

+showPayMap(): 여행지출건들에 대한 위치와 금액을 지도에 표시한다.

1.2 PayMap – 여행지도객체(외부 구글 api에 인풋값을 넘겨주기 위한 객체)

i. Attributes

+locations : 예산 및 소비 지역 위치들 리스트

+marker : 지도에 표시하기 위한 위도, 경도 리스트

2. Sequence Diagram

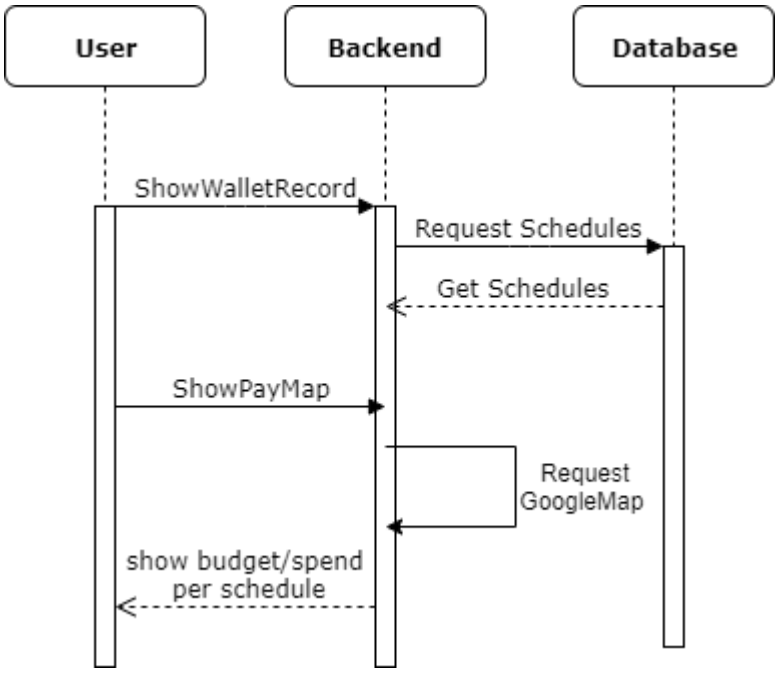

Diagram 11 : Budget Management Sequence Diagram

5. System Architecture - Backend

This chapter describes the back-end systems that handle HTTP requests from the front end of the entire system and the structure of each subsystem.

5.1 Overall System Architecture

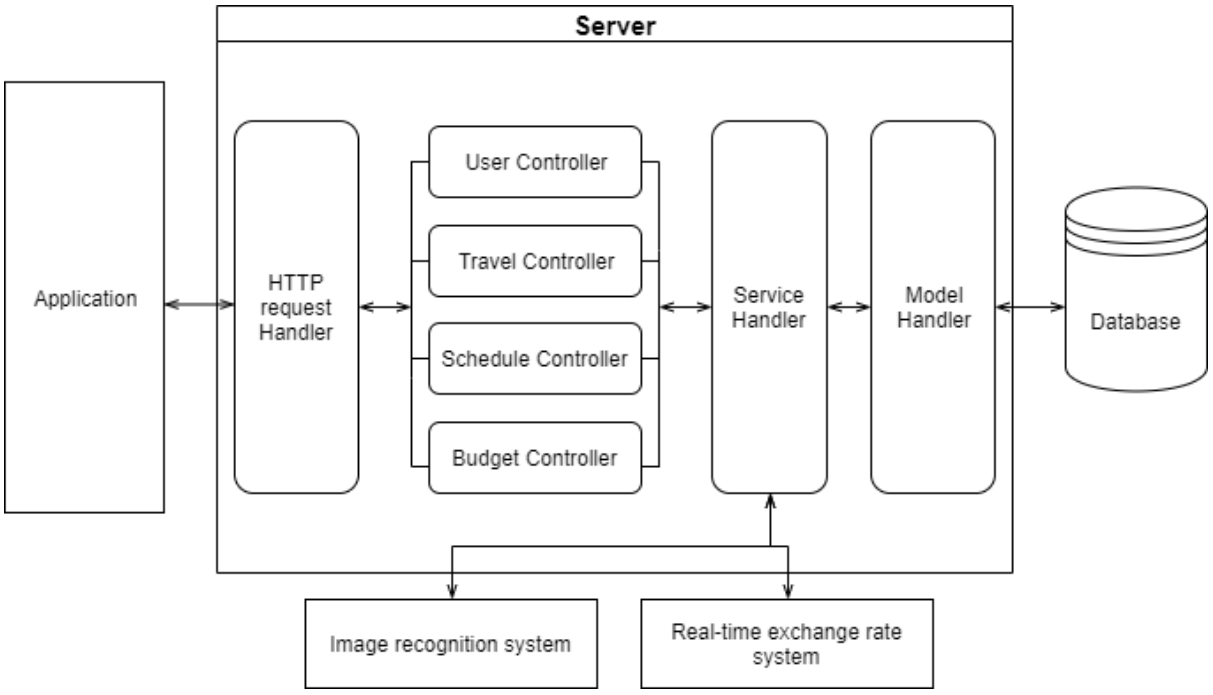

Diagram 12 : Overall System Architecture

전체 프로젝트 구조는 크게 앱, 서버, 데이터베이스로 이루어져 있다. 백엔드 시스템은 Diagram 3의 Server로 나타낸 부분으로, Application과 Database 간 데이터 및 HTTP 요청을 처리한다. 또한 시스템에 필요한 Image recognition system과 Real-time exchange rate system의 처리도 담당한다.

5.2 Subcomponents

A. Backend(Controller, Handler)

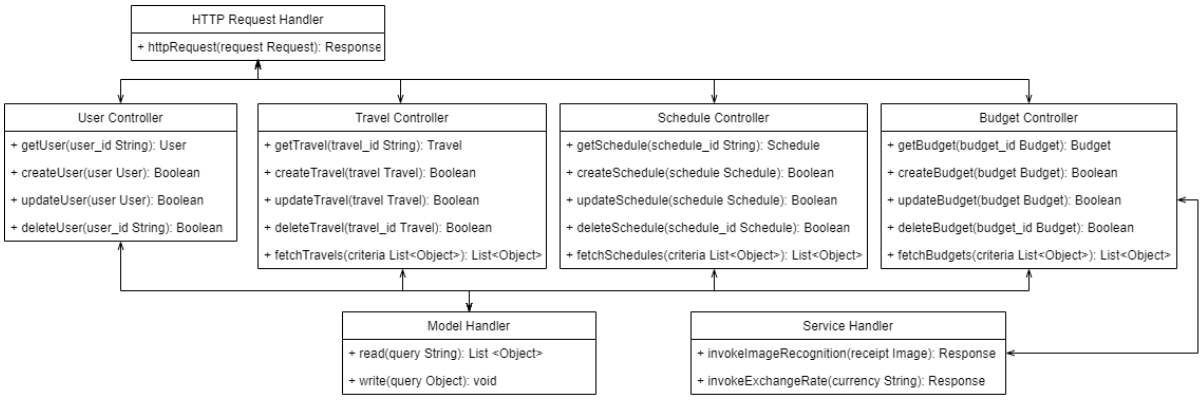

Diagram 13 : Overall Backend Handler System

1. HTTP Request Handler: 프론트엔드 HTTP 요청 처리 객체
   A. Attributes: 없음
   B. httpRequest(request Request): 서버로 들어온 HTTP 요청을 처리해 각 컨트롤러로 분배한 후 컨트롤러의 응답을 반환하는 메서드

2. Controller 공통
   A. attributes: 없음
   B. getEntity(entity_id String): 엔티티를 가져오는 메서드
   C. createEntity(entity Entity): 엔티티를 생성하는 메서드
   D. updateEntity(entity Entity): 엔티티를 수정하는 메서드
   E. deleteEntity(entity_id): 엔티티를 삭제하는 메서드

F. fetchEntities(criteria List<Object>): 검색 조건(criteria)에 맞는 엔티티를 가져오는 메서드

3. Model Handler

A. read(query String): 쿼리에 맞는 데이터를 데이터베이스에서 읽어오는 메서드

B. write(query Object): 쿼리에 맞는 데이터베이스 테이블에 데이터(Object)를 저장하는 메서드

4. Service Handler

A. invokeImageRecognition(receipt Image): 영수증 인식 시스템을 실행하는 메서드

B. invokeExchangeRate(currency String): 인자로 들어온 통화 종류에 따라 실시간 환율을 적용한 값을 반환하는 메서드

B. Image Recognition System

1. Class Diagram

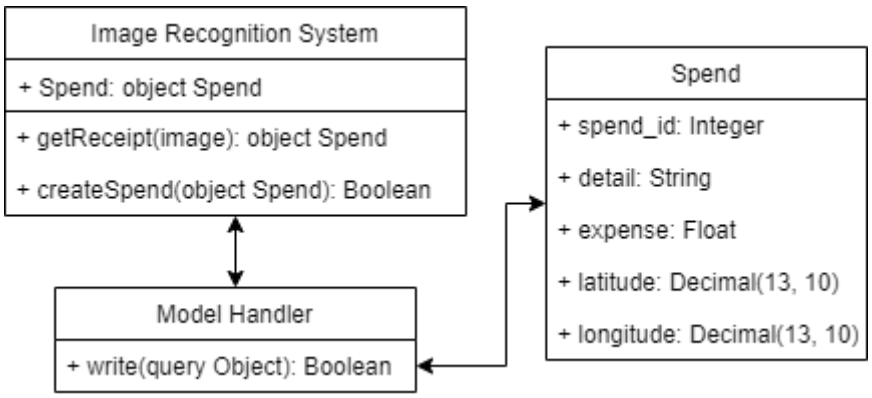

Diagram 14 : Image Recognition System Class Diagram

Image Recognition System은 사용자의 영수증을 인식해 데이터를 자동 입력해주는 시스템이다. 따라서 영수증의 이미지를 인식해 Spend 테이블에 맞춰 데이터 형식을 변환한다. 변환된 데이터는 먼저 프론트엔드로 반환되어 사용자의 확인을 거친 후 Model Handler를 거쳐 데이터베이스에 저장되고 저장 성공/실패에 따라 Boolean 값으로 Response를 반환한다.

2. Sequence Diagram

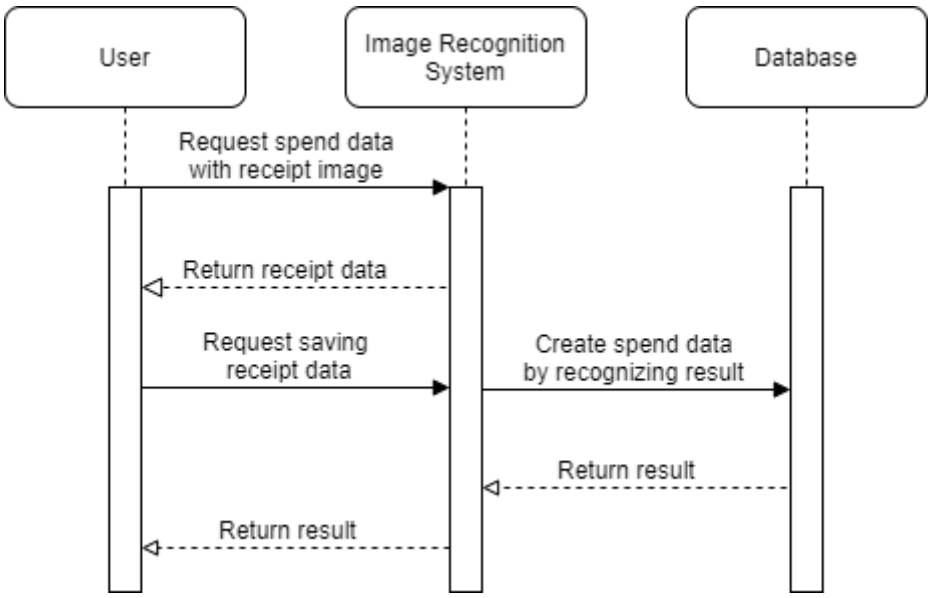

Diagram 15 : Image Recognition System Sequence Diagram

C. Real-time Exchange Rate System

1. Class Diagram

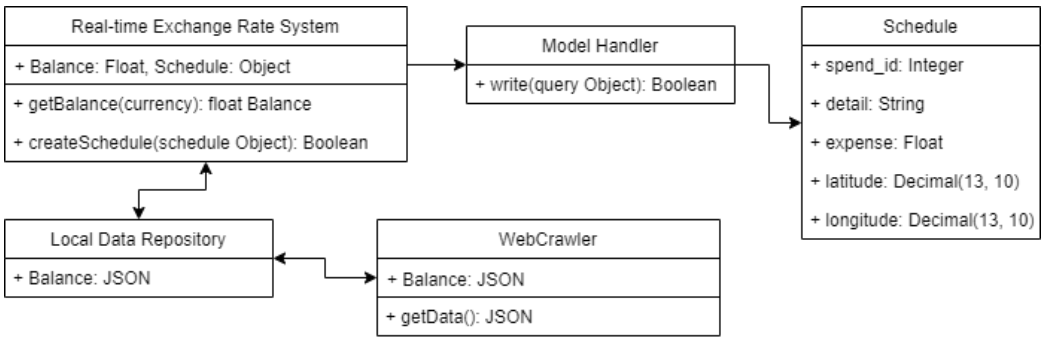

Diagram 16 : Real-time Exchange Rate System Class Diagram

Real-time Exchange Rate System은 웹 크롤러를 통해 주기적으로 1달러를 기준으로 한 각 나라 화폐로의 환산 금액을 로컬 데이터로 저장하며 기존 데이터를 업데이트한다. 사용자가 새 스케줄을 등록하면서 화폐를 바꾸고자 하면 로컬 데이터에 저장된 환전 금액으로 환산한 후 스케줄에 등록할 수 있도록 한다.

2. Sequence Diagram

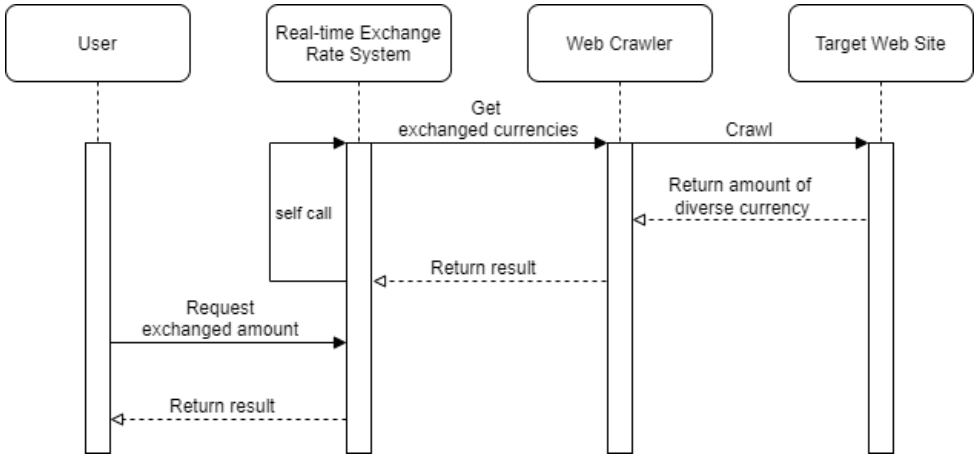

Diagram 17 : Real-time Exchange Rate System Sequence Diagram

6. Protocol Design

This chapter describes the structure used for the interaction between each subsystem, especially between the front-end and back-end application server systems, and describes how each interface is defined.

6.1 Axios for HTTP communication with React-Native

Figure 4 : Axios for HTTP communication System

react native에서 웹서버에 데이터를 요청하거나 요청한 데이터를 받아올 때 사용하는 대표적인 API이다. Axios는 여러가지 장점들을 가지고 있습니다. 첫번째로, Axios는 promise call을 통해 비동기 통신을 지원합니다. 또한, 요청 데이터를 자동으로 json데이터로 바꿔줍니다. 게다가 Axios는 호환성이 뛰어나, 다양한 브라우저에서 문제없이 실행이 가능합니다. 마지막으로 Axios는 axios.all() 콜을 통하여 동시 요청 기능을 제공합니다. 결론적으로 Axios는 대부분의 HTTP 통신 요구에 맞는 적절한 패키지로 사용하기 쉬운 API를 제공합니다.

6.2 Real-time Socket Communication

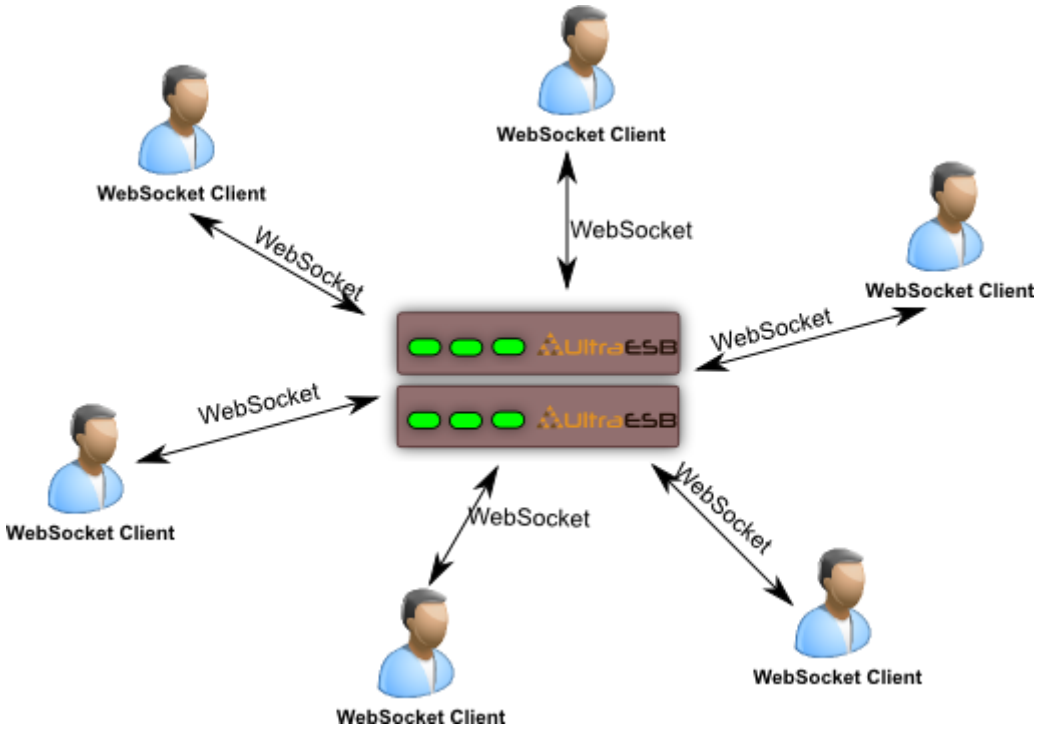

Figure 5 : Real-time Socket Communication Example

본 시스템은 같이 여행을 가는 동행 유저들 간의 실시간 통신을 지원하기 위하여 socket 통신 기술을 사용한다. 소켓은 네트워크에서 실행되는 두 프로그램 사이의 양방향 통신 링크의 한 끝 점이다. 소켓 클래스는 클라이언트 프로그램과 서버 프로그램 사이의 연결을 나타내기 위해 사용된다. 소켓 통신은 같은 네임스페이스로 묶여진 유저들간의 통신을 원활하게 처리할 수 있으며, event-driven modeling에 적합하다. Socket 통신은 계속해서 connection을 맺을 필요 없이, 한 번의 connection으로 데이터 전송 및 처리를 하기 때문에, low-latency communication 환경에서도 잘 대처할 수 있다. 따라서, 해외에서 low-latency 환경에 노출되며, 실시간 통신을 지향하는 우리의 서비스에 필수적인 기술이다.

6.3 JSON

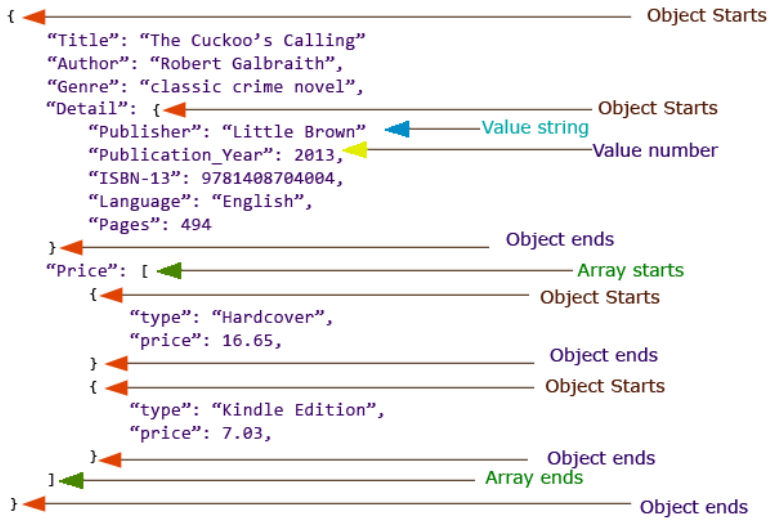

Figure 6 : JSON Example

JSON is a data exchange format that is easy for humans to read and write. JSON is easy to use for both human and machine for parsing and generating. It is a complete language that is independent of other languages in text format, but follows the conventions of the C-family languages such as C, C ++, C #, Java, JavaScript, Perl, Python, etc., which are familiar to programmers.

6.4 Details

A. Login

- Request

Method	POST	
URI	/authentication/login	
Inputs	id, password	
Request Body	id	사용자 ID
	password	사용자 패스워드

- Response

Success Code	/authentication/login	
Failure Code	id, password	
Success Response Body	message	Login Success
Failure Response Body	message	reason of failure

B. Signup

- Request

Method	POST	
URI	/authentication/signup	
Inputs	id, name, nickname, password	
Request Body	id	사용자 ID
	name	사용자 이름
	Nickname	사용자 별명
	password	사용자 패스워드

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID 중복인 경우)	
Success Response Body	message	Signup Success

Failure Response Body	message	reason of failure

C. 새 여행 추가하기

- Request

Method	POST	
URI	/travel/create	
Inputs	Start_date, end_date, nation_id, city_id, title, content, category, budget	
Request Body	Start_date, end_date	여행 시작일, 종료일
	Title	여행 제목
	Content	여행 내용(소개, 이야기)
	Category	여행 종류
	Nation_id	여행 국가
	Array<City_id>	여행 도시
	Array<User_id>	여행 동행 친구
	Total_budget	여행 총예산

- Response

Success Code	200 OK	
Failure Code	400 Not Found	
Success Response Body	Message	Travel successfully create!
Failure Response Body	Message	Reason of failure

D. 여행 목적지(국가, 도시) 검색

- Request

Method	GET	
URI	/travel/search/nation, /travel/search/city	
Inputs	Nation_id, city_id	
Request Body	nation_id city_id	검색할 국가 검색할 도시

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Nations, Cities matched with query	List<Nation> List<City>
Failure Response Body	Message	Reason of failure

E. 여행 같이 갈 친구 검색하기

- Request

Method	GET	
URI	/travel/search/friends	
Inputs	Nickname, string	
Request Body	Nickname String	닉네임 유저명

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	User matched with query	User Object
Failure Response Body	message	reason of failure

F. 여행목록 전체리스트 보여주기

- Request

Method	GET	
URI	/travels	
Inputs	-	
Request Body	User_id	유저 아이디(Primary Key)
	Travel_id	여행 아이디(Primary Key)

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Travels	List<Travel>
Failure Response Body	message	reason of failure

G. 여행 상세보기

- Request

Method	GET
URI	/travel/show
Inputs	-
Request Body	User_id, travel_id

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Travel	Travel object
Failure Response Body	message	reason of failure

H. 여행일정 보여주기

- Request

Method	GET
URI	/travel/schedules
Inputs	-
Request Body	User_id, travel_id

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Schedules	List<Schedule>
Failure Response Body	message	reason of failure

I. 여행일정 추가(수정)하기

- Request

Method	POST
URI	/travel/update
Inputs	Latitude, longitude, title, content, date, start_time, end_time, budget
Request Body	Latitude : 일정 장소(위도)
	Longitude : 일정 장소(경도)
	Title : 일정 제목
	Content: 일정 내용
	Date : 일정 날짜
	Start_time : 일정 시작 시간
	End_time : 일정 종료 시간
	Budget : 일정 예산

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Message	Schedule successfully create!
Failure Response Body	message	reason of failure

J. 예산관리 보여주기

- Request

Method	GET
URI	/travel/schedules/budget
Inputs	-
Request Body	User_id, travel_id

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Schedules budget	List<Schedule>
Failure Response Body	message	reason of failure

K. Show pay-map

- Request

Method	GET
URI	/travel/schedule/google_map
Inputs	-
Request Body	Array<Schedule>

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	Google-map	
Failure Response Body	message	reason of failure

7. Database Design

이 챕터에서는 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 데이터베이스를 구성하는 Entity 간의 관계를 기술하고, 이를 통해 Relational Schema와 SQL DDL을 작성한다.

7.1 ER Diagram

Entity는 총 5개로 User, Group, Travel, Schedule, Budget으로 이루어져 있다. Entity는 직사각형으로 표현하며 Entity 간의 관계는 마름모 모양으로 표현한다. 각 Entity가 가지는 Attribute는 타원형으로 표현되며 Entity마다 데이터 행을 구분하는데 사용할 Primary key는 라벨에 밑줄을 그어 표시한다. 각 Entity의 관계는 1대1, 1대다, 다대다의 관계를 가질 수 있으며 1은 선의 끝에 수직선이 있으며, 다수는 끝이 3개로 갈라지는 선이다. 다수 또는 0일 경우는 선이 갈라지기 이전에 원을 표시한다.

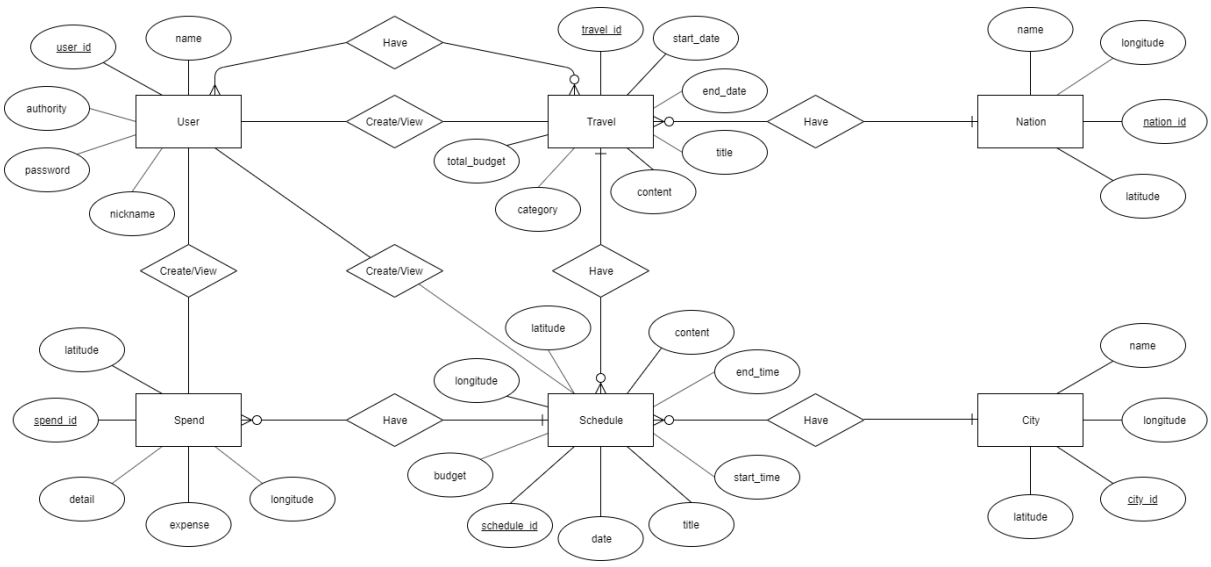

Diagram 18 : ER Diagram

A. Entities

1. User

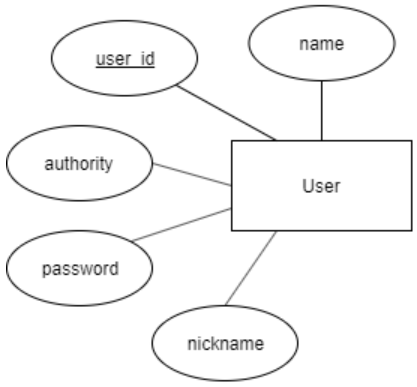

Diagram 19 : User ER Diagram

User Entity는 사용자에 대한 정보를 나타낸다. user_id가 Primary key이며 사용자의 이름, 비밀번호, 권한, 닉네임에 대한 정보를 포함한다.

2. Travel

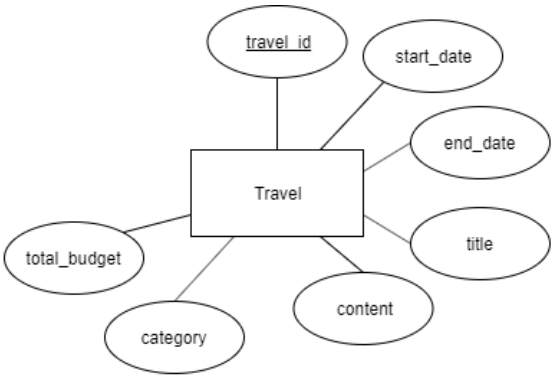

Diagram 20 : Travel ER Diagram

Travel Entity는 사용자의 여행 단위의 정보를 포함한다. travel_id가 Primary key이며 여행 시작일과 종료일, 여행 제목과 내용, 카테고리, 총 예산에 대한 정보를 포함한다.

3. Schedule

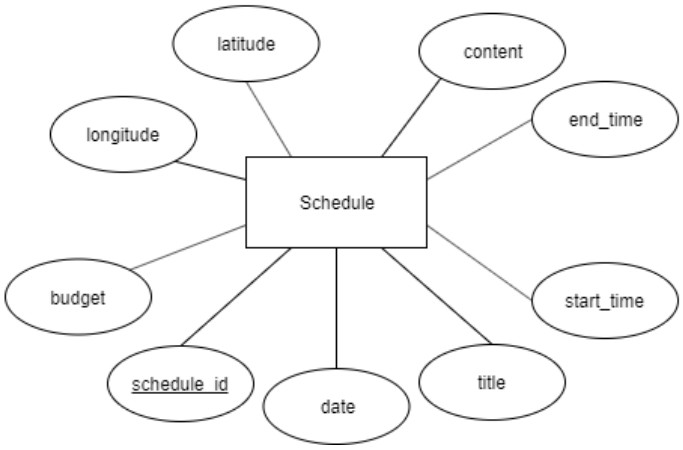

Diagram 21 : Schedule ER Diagram

Schedule Entity는 각 여행에 들어있는 여행 일정에 대한 정보이다. schedule_id가 Primary key이며 날짜와 시작 시간, 종료시간, 일정의 제목과 내용, 예산과 위치에 대한 정보를 포함한다.

4. Spend

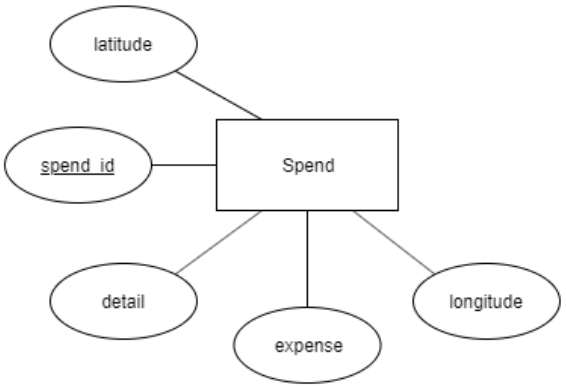

Diagram 22 : Spend ER Diagram

Spend Entity는 각 일정에 사용자가 행하는 지출에 대한 정보이다. spend_id가 Primary key이며 지출액과 지출 내용, 지출을 한 위치에 대한 정보를 포함한다.

5. Nation

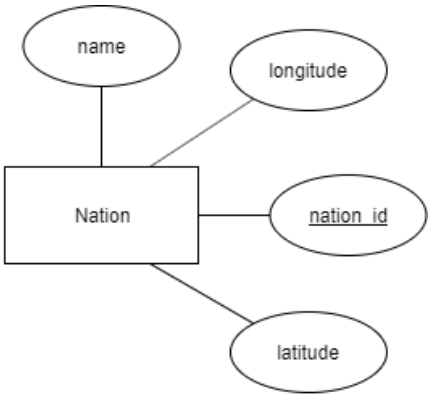

Diagram 23 : Nation ER Diagram

Nation Entity는 여행을 할 국가에 대한 정보이다. nation_id가 Primary key이며 국가

이름과 국가의 위도와 경도 좌표에 대한 정보를 포함한다.

6. City

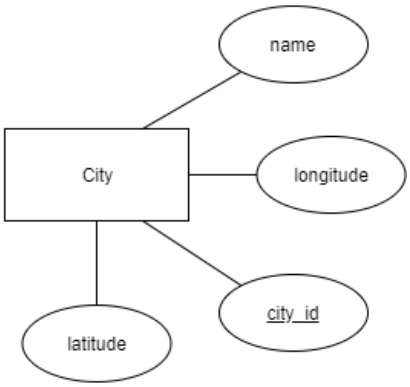

Diagram 24 : City ER Diagram

City Entity는 여행을 할 도시에 대한 정보이다. city_id가 Primary key이며 도시 이름과 도시의 위도와 경도 좌표에 대한 정보를 포함한다.

B. Relations

1. Have

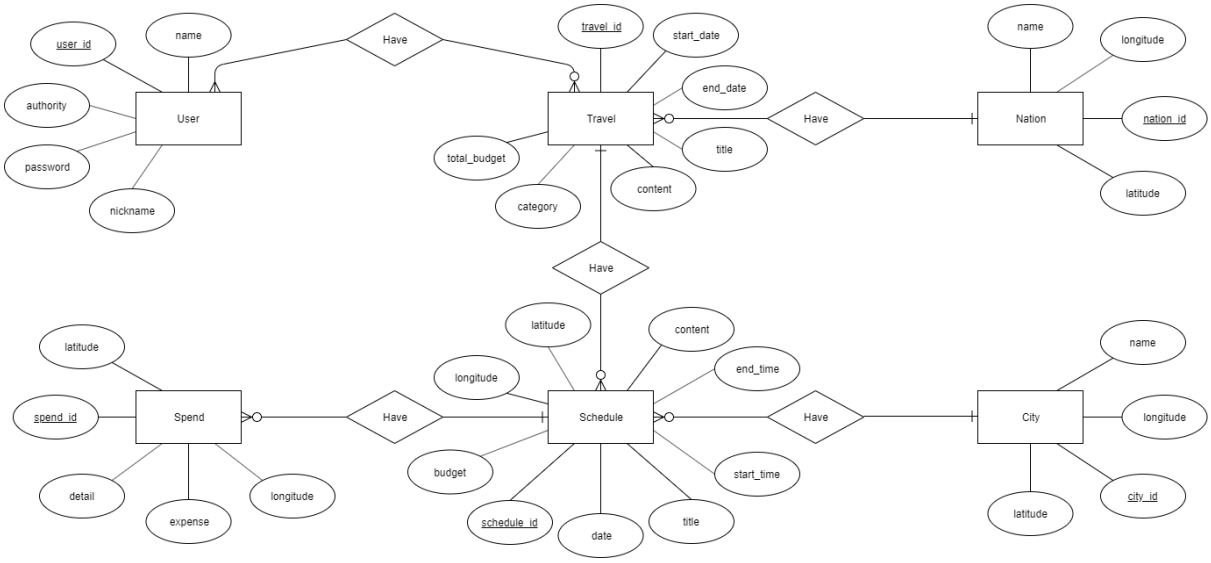

Diagram 25 : Have Relations Diagram

Have 관계는 Entity가 다른 Entity를 여러 개 포함하는 관계이다. 사용자는 여행이 없거나 여러 개의 여행을 가질 수 있으며, 여행은 하나 이상의 사용자를 가진다. 여행에 포함된 사용자들은 여행에 대한 정보를 공유한다. 여행은 여러 개의 일정들과 여행 국가 정보로 구성되어 있으며 각 일정은 여러 개의 지출들과 도시에 대한 정보를 포함한다.

2. Create/View

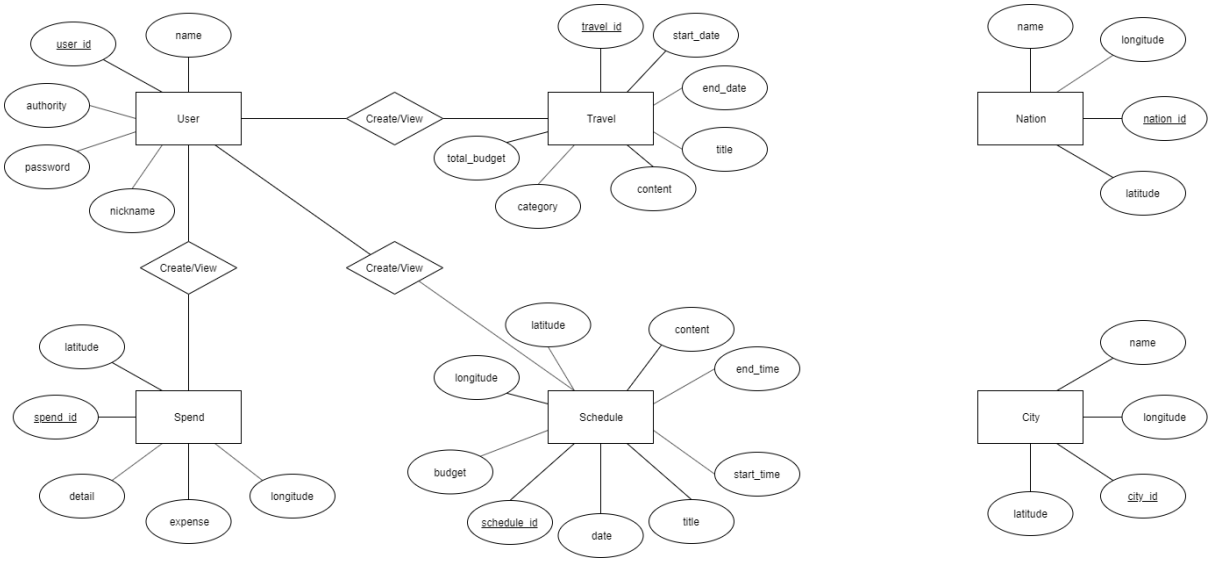

Diagram 26 : Create/View Diagram

Create/View 관계는 사용자가 생성하고 조회하는 Entity와의 관계이다. 사용자는 여행과 그에 해당하는 일정을 만들고, 각 일정마다 소비하는 지출 내용을 생성 및 조회할 수 있다.

7.2 Relational Schema

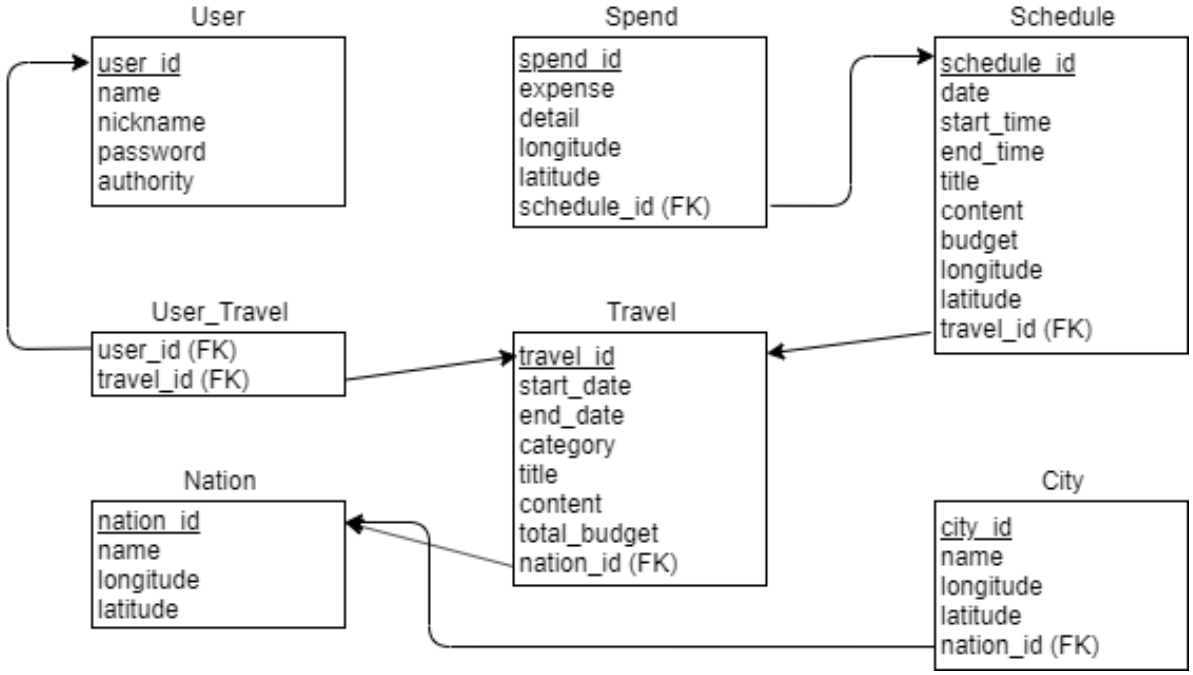

Diagram 27 : Relational Schema Diagram

7.3 SQL DDL

A. User

```
CREATE TABLE User {
        user_id INT NOT NULL,
        name VARCHAR(20) NOT NULL,
        nickname VARCHAR(20) NOT NULL,
        password PASSWORD NOT NULL,
        authority VARCHAR(10) NOT NULL,
        PRIMARY KEY (user_id)
};
```

B. Nation

```
CREATE TABLE Nation {
        nation_id INT NOT NULL,
        name VARCHAR(20) NOT NULL,
        longitude INT NOT NULL,
        latitude INT NOT NULL,
        PRIMARY KEY (nation_id)
};
```

C. City

```
CREATE TABLE City {
        city_id INT NOT NULL,
        name VARCHAR(20) NOT NULL,
        longitude INT NOT NULL,
        latitude INT NOT NULL,
        PRIMARY KEY (city_id)
};
```

D. Travel

```
CREATE TABLE Travel {
        travel_id INT NOT NULL,
        start_date DATE NOT NULL,
        end_date DATE NOT NULL,
        category INT NOT NULL
        title VARCHAR(50),
        content VARCHAR(1000),
        total_budget INT,
        nation_id INT NOT NULL,
        PRIMARY KEY (travel_id),
        FOREIGN KEY (nation_id) REFERENCES Nation(nation_id)
};
```

E. User_Travel

```
CREATE TABLE User_Travel {
        user_id INT NOT NULL,
        travel_id INT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES User(user_id),
        FOREIGN KEY (travel_id) REFERENCES Travel(travel_id)
};
```

F. Schedule

```
CREATE TABLE Schedule {
        schedule_id INT NOT NULL,
        date DATE NOT NULL,
        start_time TIME NOT NULL,
        end_time TIME NOT NULL,
        title VARCHAR(50) NOT NULL,
        content VARCHAR(1000) NOT NULL,
        budget INT,
        longitude INT,
        latitude INT,
        travel_id INT NOT NULL
        PRIMARY KEY (schedule_id),
        FOREIGN KEY (travel_id) REFERENCES Travel(travel_id)
};
```

G. Spend

```
CREATE TABLE Spend {
        spend_id INT NOT NULL,
        expense INT NOT NULL,
        detail VARCHAR(100) NOT NULL,
        longitude INT,
        latitude INT,
        schedule_id INT NOT NULL,
        PRIMARY KEY (spend_id),
        FOREIGN KEY (schedule_id) REFERENCES Schedule(schedule_id)
};
```

8. Testing Plan

Test policies and test cases are described in the Testing Plan. The purpose of this chapter is to plan a process to verify that the entire system is implemented as intended. The Testing Policy describes a step-by-step approach to testing. The Test Case identifies the user based on the example case in each Sub-System and describes the input and output behavior expected from the system accordingly.

8.1 Testing Policy

시스템 개발은 V-model을 적용하여 크게 다섯 단계로 나누어 Testing을 진행한다.

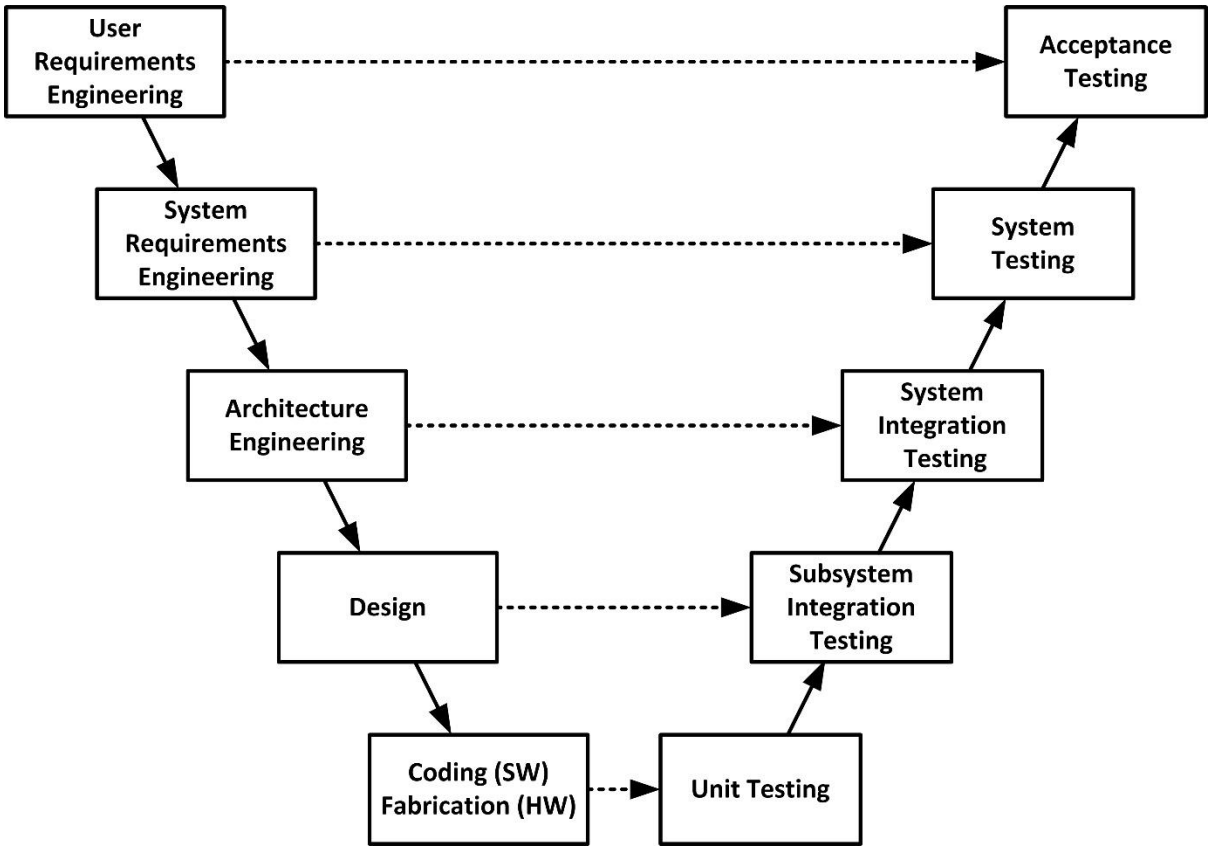

Diagram 28 : V-model Testing Process

A. Unit Testing

Unit Testing은 각 프로그램 단위나 모듈, 클래스들을 Testing하는 것이다. 이는 각 Unit을 개발한 개발자들이 책임지고 Testing한다. 원래의 기능에 적합하게 동작하는지를 확인하여, 검증된 것만 Commit 하도록 한다.

B. Subsystem Integration Testing

Subsystem Integration Testing 또한 각 하위 시스템을 개발한 개발자들이 책임지고 Testing한다. 이 때 주어진 protocol 혹은 Interface 등에 어긋나지 않는지를 확인한다.

C. System Integration Testing

통합된 하위 시스템들을 모아서 단일 시스템으로 합치는 Testing이다. 각 Sub-system 요소들을 Incrementally Integrating 하여 Testing을 진행한다. 이러한 각 요소들이 합쳐진 이후에 전체 시스템에 대하여 최종 Testing을 진행한다.

D. System Testing

명세화된 요구사항들에서 정의한 Functional, Non-functional requirements들을 평가하고, 전체 시스템적인 측면에서 에러가 발생하는 부분을 점검한다.

E. Acceptance Testing

사용자들이 직접 참여하여 사용자 환경에서 시스템을 Testing하는 것이다. 실제 여행객들을 대상으로 테스트를 진행하여 performance와 usability를 체크하며, 피드백을 받는 것으로 전체 테스트 과정을 마무리한다.

8.2 Testing Case

A. Request Handler System

1) 모든 CRUD request들에 대하여 그에 맞는 response activity로 이동을 하는지 확인

2) 각각의 activity들의 동적 이벤트에 따른 transformation도 잘 작동하는지 확인

3) CRUD request들이 DB에 정상적으로 저장되는지 확인

B. User management System

1) 아이디, 닉네임 중복되는 경우 :

중복된 아이디, 닉네임 경고창 띄운다.

2) 비밀번호 불일치 경우 :

아이디, 비밀번호를 다시 확인해주세요라는 에러 메시지를 띄운다.

C. Image Recognition System

1) 사진 자동 인식이 안되었을 경우 :

인식불가 경고메세지와 함께 “다시찍기 or 수기로 입력하기” 선택 경고창을 띄운다.

2) 최초 카메라/앨범 권한을 요청 한 경우 :

만약 유저가 권한 요청을 거부하면, 수기로 입력하는 인풋 폼으로 커서 위치시킨다.

유저가 권한 요청을 승인하면, 더 이상 권한 요청을 물어보지 않는다.

D. Real-time Socket Communication System

1) 네트워크가 불안정하여, 소켓 연결이 끊어진 경우 :

유저의 앱서비스가 강제 종료되지 않도록 하며, 네트워크 연결을 다시 시도하라는 메시지를 띄운다.

2) multi-casting 데이터를 보내는 상황에서 수신하는 쪽의 소켓 연결이 끊어진 경우 :

수신 쪽이 다시 소켓 연결이 되면, notification 알림을 수신 기기로 전송한다.

E. Real-time Exchange Rate System

1) 서버에서 실시간 환율을 크롤링하는데 오류가 발생하는 경우 :

서버가 다운되지 않으며, 서버에 기존에 저장되어 있는 환율을 사용, 적용한다.

9. Development Plan

This chapter describes the technology and development environment to be used in the actual development phase, and describes the development schedule and progress.

9.1 Development Environment

A. Development Framework

Figure 7 : Development Framework - React-Native, Nodejs

프론트엔드 프레임워크로는 React Native를 사용하여 안드로이드, ios를 동시에 개발한다. React-Native (RN)는 Facebook이 2015년 4월에 오픈한 플랫폼 모바일 애플리케이션 개발 프레임워크이고 Facebook이 개발한 JS 프레임워크 React 모바일 애플리케이션 플랫폼에서 파생된 것으로, 현재 iOS와 안드로이드의 양대 플랫폼을 지원하고 있다. RN은 Java-script 언어를 사용하고 HTML의 JSX과 비슷하고 또한 CSS를 통한 모바일 애플리케이션 개발하는 기술이다. 안드로이드 개발 경험이 있는 팀원이 없지만, 자바스크립트 기반의 웹 개발을 해본 팀원들이 있기 때문에, RN이 적은 학습만으로도 어플 개발을 가능하게 할 수 있어 개발언어로 선정되었다

백엔드 프레임워크로는 Nodejs를 이용하여 프론트엔드와 백엔드에서 사용하는 언어를 자바스크립트로 통일시켜서 개발에 있어서 효율성을 극대화시키고자 한다.

B. IDE

Figure 8 : IDE - Visual Studio Code, Expo

MS의 개발 툴 중 최초로 크로스 플랫폼을 지원하는 에디터이며 윈도우, macOS, 리눅스를 모두 지원한다. 이처럼, 여러 운영체제를 지원하기 때문에 팀원간에 개발환경의 일관성을 유지할 수 있어 IDE를 visual studio code로 통일한다. Expo는 카메라, 위치, 알림, 센서 등 다양한 기능을 지원하는 통합 플랫폼 Api이다. 또한, Expo는 리액트 네이티브 앱 개발에 최적화되어 있으며, 배포 업데이트를 자동화해주기 때문에 편리하다. 단점으로 Expo에서만 제공되는 API만 사용이 가능하며, native파일들을 크게 제어할 수 없지만, 우리의 서비스는 native파일을 크게 제어하는 수준이 아니므로 개발, 배포의 편의성에 초점을 두어 Expo를 사용하기로 결정하였다.

C. Version Management Tool

Figure 9 : Version Management Tool - Github

효율적 코드 관리 및 공유를 위해 Github를 사용한다. GitHub는 분산 버전 관리 툴인 Git을 사용하는 프로젝트를 지원하는 웹호스팅 서비스로, 세계적으로 사용되는 Git 호스팅 사이트이다.

10. Index

10.1 Figures

10.2 Diagrams