1. 설계 배경

- 프로젝트 중요성, 필요성: 플랫폼 구축의 필요성 소개 + 이러한 플랫폼을 위해 DB 설계를 함.

- 문제 정의: 플랫폼에서 구현할 기능들 나열 ex. 조회 기능: 기업 목록 조회, 지원한 회원의 제출물 조회

- 기대 효과: 플랫폼의 기대 효과

- 설계 계획서의 내용 요약: RDB 중 하나인 \*\*데이터베이스를 이용, 서버는 MVC 모델로 구현,

2. 설계 목표

- 요구 분석서: 기능 요약

- 설계 과정 중 주요 고려 사항: 엔티티 관계, 무결성

- 시스템 환경 소개: FE, BE, DB, DESIGN PATTERN, OS

- 시스템을 평가할 수 있는 항목: TDD(테스트 주도 개발) 이용.

3. 설계 내용

- 개념적인 DB 설계(ER 다이어그램 작성 그림)

- DB 모습 정리 요약

- 논리적인 DB 설계: 테이블 구조와 테이블 사이의 관련성

- 정규화 과정

- 물리적 설계

- DML, DDL 등

4. 설계 결과 및 분석

- 기능 보완이 필요한 부분 정리 및 개선 방안 제시

- 설계 목표의 달성 정도를 분석해 기술

5. 결론 및 활용 분야

**ERD 설계 요약**

이 설계는 MySQL과 Flask, Flutter를 사용한 당류 계산 웹사이트를 위한 것입니다. 이 웹사이트는 프랜차이즈 카페의 음료 정보를 관리하고, 사용자 정보를 저장하며, 사용자의 음료 소비 기록 및 선호도를 추적합니다.

**테이블 목록 및 필드**

1. Cafes: 프랜차이즈 카페 정보를 저장

cafe\_id: INT, Primary Key, Auto Increment, NOT NULL

cafe\_name: VARCHAR(255), NOT NULL

2. Drinks: 음료 정보를 저장

drink\_id: INT, Primary Key, Auto Increment, NOT NULL

cafe\_id: INT, Foreign Key references Cafes(cafe\_id), NOT NULL

drink\_name: VARCHAR(255), NOT NULL

volume: FLOAT, NOT NULL

sugar\_content: FLOAT, NOT NULL

calories: FLOAT, NOT NULL

3. Users: 사용자 정보를 저장

user\_id: INT, Primary Key, Auto Increment, NOT NULL

username: VARCHAR(255), NOT NULL

email: VARCHAR(255), NOT NULL

password: VARCHAR(255), NOT NULL

4. ConsumptionRecords: 사용자가 소비한 음료 기록을 저장

record\_id: INT, Primary Key, Auto Increment, NOT NULL

user\_id: INT, Foreign Key references Users(user\_id), NULL

drink\_id: INT, Foreign Key references Drinks(drink\_id), NULL

consumption\_date: DATETIME, NOT NULL

5. FavoriteCafes: 사용자가 선호하는 카페를 저장

favorite\_cafe\_id: INT, Primary Key, Auto Increment, NOT NULL

user\_id: INT, Foreign Key references Users(user\_id), NULL

cafe\_id: INT, Foreign Key references Cafes(cafe\_id), NULL

6. FavoriteDrinks: 사용자가 선호하는 음료를 저장

favorite\_drink\_id: INT, Primary Key, Auto Increment, NOT NULL

user\_id: INT, Foreign Key references Users(user\_id), NULL

drink\_id: INT, Foreign Key references Drinks(drink\_id), NULL

**테이블 간 관계**

1. Cafes와 Drinks:

관계: One-to-Many

카디널리티: 1..\*

설명: 한 카페는 여러 음료를 가질 수 있으며, 모든 카페는 여러 음료를 가짐.

2. Users와 ConsumptionRecords:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 사용자는 여러 소비 기록을 가질 수 있으며, 사용자는 소비 기록이 없을 수도 있음.

3. Drinks와 ConsumptionRecords:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 음료는 여러 소비 기록에 나타날 수 있으며, 나타나지 않을 수도 있음.

4. Users와 FavoriteCafes:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 사용자는 여러 선호 카페를 가질 수 있으며, 선호 카페가 없을 수도 있음.

5. Cafes와 FavoriteCafes:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 카페는 여러 사용자의 선호 카페가 될 수 있으며, 아닐 수도 있음.

6. Users와 FavoriteDrinks:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 사용자는 여러 선호 음료를 가질 수 있으며, 선호 음료가 없을 수도 있음.

7. Drinks와 FavoriteDrinks:

관계: One-to-Many

카디널리티: 0..\*

설명: 한 음료는 여러 사용자의 선호 음료가 될 수 있으며, 아닐 수도 있음.

**MySQL 데이터베이스 설정**

**데이터베이스 및 테이블 생성**

CREATE DATABASE sugar;

Use sugar;

CREATE TABLE Cafes (

cafe\_id INT AUTO\_INCREMENT PRIMARY KEY,

cafe\_name VARCHAR(255) NOT NULL

);

CREATE TABLE Drinks (

drink\_id INT AUTO\_INCREMENT PRIMARY KEY,

cafe\_id INT,

drink\_name VARCHAR(255) NOT NULL,

volume FLOAT,

sugar\_content FLOAT,

calories FLOAT,

image\_url VARCHAR(255),

FOREIGN KEY (cafe\_id) REFERENCES Cafes(cafe\_id)

);

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE ConsumptionRecords (

record\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

drink\_id INT,

consumption\_date DATETIME NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (drink\_id) REFERENCES Drinks(drink\_id)

);

CREATE TABLE FavoriteCafes (

favorite\_cafe\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

cafe\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (cafe\_id) REFERENCES Cafes(cafe\_id)

);

CREATE TABLE FavoriteDrinks (

favorite\_drink\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

drink\_id INT,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (drink\_id) REFERENCES Drinks(drink\_id)

);

**이미지 파일 경로를 데이터베이스에 저장:**

이미지를 서버의 파일 시스템에 저장하고, 데이터베이스에는 이미지 파일의 경로(URL)를 저장합니다.

이 방법은 데이터베이스의 크기를 줄이고, 이미지 파일을 독립적으로 관리할 수 있게 해줍니다.

**데이터 삽입**

Json파일에 데이터를 크롤링으로 적재 -> 데이터베이스에 load\_data.py로 삽입 -> Flask 애플리케이션 실행 -> <http://127.0.0.1:5000/drinks>로 접속하면 로드된 데이터를 확인할 수 있음

**기능 구현**

Routes.py

사용자 인증 기능: 사용자 등록, 로그인, 로그아웃

음료 CRUD 기능: 음료 생성, 조회, 수정, 삭제

프론트엔드와 연동하기 위한 API 엔드포인트 제공

**프론트엔드와의 연동**

2.1. API 엔드포인트 설계

백엔드에서 제공할 API 엔드포인트를 설계합니다. 이를 통해 Flutter 프론트엔드와 데이터를 주고받을 수 있습니다.