FlavorFind - A Recipe Finder & Personal Cookbook

Phase 1 Report: Planning & System Design

Prepared For: Dr. Ahmed Antar
Prepared By: Mark William (120210348)
Course: Data Engineering - CSE 428
Date: [Insert Submission Date, e.g., October 26, 2023]
Version: 1.0

1. Introduction / Overview

This document outlines the planning and system design for the "FlavorFind" web application, developed as the final project for the Data Engineering (CSE 428) course. FlavorFind is a dynamic web application designed to allow users to search for recipes using an external online database (TheMealDB API) and also manage their own personal collection of recipes through CRUD (Create, Read, Update, Delete) operations. The core objective is to demonstrate the integration of a web interface, backend logic, and a relational database (MySQL) to create a functional, data-driven application.

2. Project Scope

The scope of FlavorFind Version 1.0 is defined as follows:

In Scope:

- **User Authentication:** Secure registration and login for users.
- **External Recipe Search:** Ability for any user to search recipes via TheMealDB API by name or ingredient.
- **External Recipe Viewing:** Displaying detailed information (ingredients, instructions, image) for recipes found via the external search.
- **Personal Recipe CRUD:** Authenticated users can Create, Read (view list and details), Update, and Delete their own recipes.
- **Database Integration:** All user account information and personal recipes are stored persistently in a MySQL database.

- **Web Interface:** A basic, functional web interface built with HTML, CSS, and Vanilla JavaScript providing access to all in-scope features.
- **API Communication:** Backend API endpoints communicating with the frontend primarily using JSON.

Out of Scope (for Version 1.0):

- Advanced search filtering (e.g., by cuisine, dietary needs, cooking time).
- User ratings or comments on recipes (neither external nor personal).
- Functionality for sharing personal recipes with other users.
- Meal planning features.
- Direct image uploading for personal recipes (users will provide image URLs).
- User roles beyond a standard registered user (e.g., no Admin role implemented).
- Password recovery mechanism ("Forgot Password").
- Mobile-specific UI optimization (focus is on desktop browsers).

The primary target user for the core CRUD functionality is the **Registered User**.

3. Planned Tools / Technology Stack

The following technologies have been selected for the development of FlavorFind V1.0:

- **Backend Framework: Python 3.x** with **Flask**
- **Database: MySQL** (Version 5.7+ / 8.x)
- **Database Interaction (ORM): SQLAlchemy**
- **Frontend: HTML5, CSS3, Vanilla JavaScript (ES6+)**
- **API Integration: Python `requests` library**
- **Password Hashing: Werkzeug security helpers**
- **Version Control: Git** (Used locally initially).

4. System Design Decisions

4.1. Architecture:

FlavorFind will follow a standard three-tier web application architecture:

1. **Presentation Tier (Frontend):** HTML/CSS/Vanilla JS running in the user's browser. Handles UI rendering and user interaction.

2. **Logic Tier (Backend):** Python/Flask application server. Handles incoming requests, business logic, user authentication/authorization, interaction with the database (via SQLAlchemy), and communication with the external TheMealDB API.
3. **Data Tier:** MySQL database storing persistent application data (users, recipes).

4.2. Database Design:

- **Model:** A relational model was chosen, implemented in MySQL, as the data (Users, Recipes, their attributes and relationship) is well-structured.
- **Entities:** The core entities identified are `USER` and `RECIPE`.
- **Relationship:** A one-to-many relationship exists between `USER` and `RECIPE` (one user can create many recipes; one recipe belongs to one user). This is implemented via a foreign key (`user_id`) in the `RECIPE` table referencing the `USER` table. `ON DELETE CASCADE` will be used to maintain referential integrity.
- **ERD/EERD:** The detailed database structure is documented in the submitted ERD and EERD diagrams (using Chen notation as requested). The EERD demonstrates potential future extension via user specialization (RegisteredUser, AdminUser) using the `ISA` concept, even though only the RegisteredUser functionality is implemented in V1.0.
    - *(Reference: `FlavorFind_ERD.PNG`, `FlavorFind_EERD.PNG`)*
- **Key Attributes:** Primary keys (`user_id`, `recipe_id`) are auto-incrementing integers. Unique constraints are applied to `username` and `email`. `password_hash` stores securely hashed passwords. Timestamps (`created_at`, `updated_at`) track record creation and modification.

4.3. API Design:

- The Flask backend will expose RESTful API endpoints for the frontend JavaScript to interact with (e.g., `/api/login`, `/api/recipes`, `/api/recipes/<id>`).
- **JSON** will be the primary data format for requests and responses between the frontend and backend, satisfying the project requirement of using "XML or/and JSON".

4.4. Security Decisions:

- **Authentication/Authorization:** Access to personal recipe CRUD operations is strictly limited to logged-in users who own the specific recipe. Backend authorization checks are critical.
- **Password Storage:** Passwords will be hashed using Werkzeug's secure hashing functions (Ref SEC-03 in SRS).
- **SQL Injection Prevention:** Using SQLAlchemy ORM significantly mitigates SQL injection risks.
- **XSS Prevention:** Basic measures like proper output escaping (often handled by templating engines or careful JS manipulation) will be employed.
- **CSRF Protection:** Basic CSRF protection (e.g., using Flask-WTF or equivalent) should be considered for state-changing form submissions.
- **HTTPS:** Will be essential for deployment.

5. Conclusion

The planning and design phase for FlavorFind has established a clear scope, a suitable technology stack, and a coherent system design. The project aligns well with the Data Engineering course objectives, providing opportunities to implement database design (ERD/EERD), backend development (Flask, SQLAlchemy, MySQL), frontend interaction (HTML/CSS/JS), and external API integration. The design choices prioritize meeting core requirements while leveraging standard, manageable technologies suitable for the developer's experience level.