Lab 2 - TasteBuddies Software Requirements Specification

Benjamin Nissley

Old Dominion University

CS411W

Professor Sarah Hosni

March 28, 2025

Version 2

Table of Contents

List of Figures

1 Introduction

1.1 Purpose

This Software Requirements Specification document provides a comprehensive reference for the developers of the TasteBuddies application. It outlines specific functional and non-functional requirements, system architecture, constraints, and dependencies that developers must understand to successfully build the application.

1.2 Scope

TasteBuddies is a personalized dish-recommendation application designed to address the challenges of restaurant selection by utilizing user taste profiles to generate tailored recommendations. The system will enable users to create personalized taste profiles, match users with compatible taste preferences, provide dish-specific recommendations based on taste profile analysis, facilitate group dining recommendations, and display real-time updates on restaurant conditions and menu changes. The system will not process payments or handle financial transactions, provide delivery services, or generate or modify restaurant menus.

1.3 Definitions, Acronyms, and Abbreviations.

Following is a glossary of terms and definitions used throughout this document in reference to TasteBuddies features:

Crowdsourced Data: User-generated data on restaurant wait times, dish availability, quality, and more.

Curated Reviews: Reviews presented and weighted based on users with similar Taste Profiles.

Daily Dish Report: Provides live updates from TasteBuddies and restaurants, including new reviews, specials, and dishes.

Data Clustering: Grouping diners into clusters with similar preferences to enhance taste profile accuracy and recommendations.

Dining Filters: Options to filter restaurants by location, cuisine, occasion, and current busyness.

Google API: An external tool integrated into the app to provide real-time data on restaurant occupancy and busyness.

Group Dining Algorithm: An algorithm that combines multiple users' profiles to provide recommendations for restaurants and dishes that best match group preferences.

Recommendation Algorithm: An algorithm that provides users with relevant recommendations based on their matched TasteBuddies, taste profile, and previous interactions.

Social Engagement: Encourages users to interact with each other and get involved in the community.

Super TasteBuddies: Taste influencers or food experts with specialized knowledge who can recommend specific cuisines or dishes.

Tailored Recommendations: Personalized recommendations based on a user’s taste profile.

TasteBuddies: Users with highly similar taste profiles, leading to improved recommendations based on aligned tastes.

Taste Matching Algorithm: A key algorithm in the app that pairs users based on similar taste profiles.

Taste Profiles: Personalized profiles created by each user, based on their taste preferences, such as for spicy, sweet, salty, etc.

1.4 References

[1] B. Nissley, "Lab 1 - TasteBuddies Product Description," Old Dominion University, CS411W, Version 4, Feb. 28, 2025.

[2] "Restaurant Impact Report," OpenTable. https://www.opentable.co.uk/c/wp-content/uploads/sites/342/2023/10/opentable_uk-restaurant-impact-report-2023.pdf (accessed Mar. 28, 2025).

[3] T. Perkins, "Your food is more expensive – are US corporate profits to blame?," The Guardian. https://www.theguardian.com/environment/article/2024/jul/26/food-price-inflation-corporate-profit (accessed Mar. 28, 2025).

[4] "Social eating connects communities," University of Oxford. https://www.ox.ac.uk/news/2017-03-16-social-eating-connects-communities (accessed Mar. 28, 2025).

[5] K. Vaghasiya, "15 Fake Review Statistics You Can't Ignore (2024)," WiserNotify. https://wisernotify.com/blog/fake-review-stats/#combating-fake-reviews:-strategies-and-tools (accessed Mar. 28, 2025).

1.5 Overview

The remainder of this document is organized as follows: Section 2 provides the product perspective, functions, user characteristics, constraints, and dependencies. Section 3 details the functional and non-functional requirements that must be implemented. Section 4 includes additional reference materials and appendices.

2 Overall Description

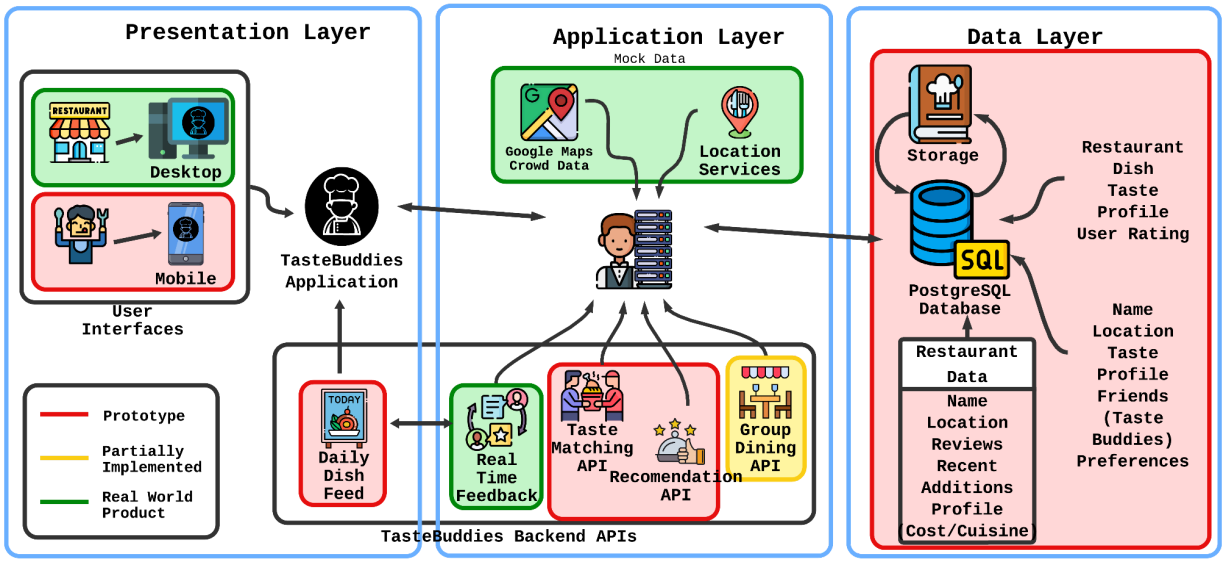
2.1 Product Perspective

TasteBuddies is an application with a three-layer architecture: the presentation layer which includes the frontend interface for users and restaurant partners, the application layer which handles TasteBuddies algorithms like Taste Matching, Dish Recommendation, and Group Dining, and the data layer, containing user data storage, restaurant data, and PostgreSQL database implementation.

For the prototype, the system follows a monolithic architecture hosted on a single server using Flask. The frontend implements server-side rendering via Flask templates with Jinja2. The backend utilizes Python with a RESTful architecture through Flask routes, and the data layer employs SQLAlchemy ORM for database interactions.

Figure 1

TasteBuddies Major Functional Components Diagram



2.2 Product Functions

The core functions of TasteBuddies which are to be fully implemented in the prototype are user account creation and authentication, personalized taste profile creation and modification, user-to-user connections based on taste similarity, finding and connecting with known contacts, dish recommendations based on taste profiles with compatibility scores, group restaurant recommendations, Daily Dish feed displaying personalized recommendations and updates, analytics dashboard for customer preferences and dining patterns, data privacy implementation and management, system security management, and data backup procedures.

The functions to be partially implemented in the prototype are restaurant discovery based on taste-similar users' preferences, ratings and reviews filtered by taste similarity, notification system for restaurant specials matching taste profiles, restaurant rating and review submission.

The functions which are not to be implemented in the prototype are automatic taste profile adjustments based on user reviews, specific dish search functionality by location, dining experience tracking and history, reward system for restaurant discounts, advanced filtering (cuisine, location, price), crowdsourcing system for restaurant information, restaurant profile and menu management, feedback collection from taste-matched customers, restaurant reward system management, customer preference analytics, review response system, restaurant verification system, reward system management, user support system, API integration management, and performance monitoring and optimization.

Figure 2.1

RWP vs Prototype table (a)

Category	Features	RWP	Prototype	Additional Notes
Account Management	Account Creation	✓	✓	
	Login / Authentication	✓	Eliminated	
	Access Permissions and Preferences	✓	Partially Implemented	Access Permissions required for database
	Taste Profile	✓	✓	
Mobile App Features	Social Engagement	✓	Partially Implemented	Find friends only for group matching
	Daily Dish Feed	✓	✓	
	Group Restaurant Matching	✓	Partially Implemented	Implementation is time dependent
	Dish Recommendations	✓	✓	
	Taste Profile Builder	✓	✓	
	Reviews	✓	Partially implemented	Mock data for compatibility matching
	Community Updates	✓	Eliminated	
	Dish Validation	✓	Eliminated	
	Taste Matching	✓	✓	
	Notification Features	✓	Eliminated	
	Engagement Features	✓	Eliminated	
DataBase Management	Data Analytics	✓	Eliminated	
	Data Privacy and Security	✓	✓	
	Trend Reports	✓	Eliminated	
	Data Backups	✓	✓	

Figure 2.2

RWP vs Prototype table (b)

Category		Features	RWP	Prototype	Additional Notes
Expanded User Mobile App Features	Social Engagement	TasteBuddies	✓	✓	
		Super TasteBuddies	✓	Partially implemented	Hard coded
		Add/Find Buddies	✓	✓	
		Follow TasteBuddy	✓	Eliminated	
		Follow Restaurant	✓	Eliminated	
		Add Kudos	✓	Eliminated	
	Live Interactive Updates	Daily Dish feed	✓	✓	
		Add reviews	✓	Partially implemented	Mock data provided
		Post restaurant update	✓	Eliminated	
		Post dish update	✓	Eliminated	
		Notifications	✓	Eliminated	
	Recommend-ation Engine	Taste Profile	✓	✓	
		Read Reviews	✓	Partially Implemented	
		Taste Matching	✓	✓	
		Dish Recommendation	✓	✓	
		Group Restaurant Matching	✓	Partially Implemented	Implementation time dependent
		Rewards	✓	Eliminated	
		Adaptive Taste Profile personalization	✓	Eliminated	Need active data over time
	Search	Restaurant filtering	✓	Eliminated	
		Dish filtering	✓	Eliminated	
	Engagement features	Rewards	✓	Eliminated	
		Badges	✓	Eliminated	
		Challenges	✓	Eliminated	

2.3 User Characteristics

Standard users possess basic smartphone proficiency and use the application to find personalized restaurant and dish recommendations. These users interact with the application interface and tend to engage with the platform regularly, especially when planning to dine out.

Groups consist of users with basic smartphone proficiency who utilize the application to plan group dining experiences. Their primary interaction involves the group creation and management features within the application. These users typically access these features occasionally when organizing meals with multiple participants.

Restaurant Partners, while not implemented in the current prototype, are expected to have basic to intermediate web interface experience. They would use the platform to update menu items and provide special promotions through a dedicated web interface for restaurant management. These partners would likely perform regular updates as their menus and business conditions change.

2.4 Constraints

The prototype must function on standard web browsers, and the frontend interface must be optimized for mobile devices.

2.5 Assumptions and Dependencies

The system relies on several key dependencies for its operation. The SQLAlchemy ORM provides the database abstraction layer necessary for data management and persistence. The Flask framework serves as the foundation for web application development, enabling the creation of the system's core functionality. During the prototype phase, the system will utilize mock data for restaurants and dishes to demonstrate and test its features. Jinja2 is used for rendering templates and facilitating the dynamic generation of web pages and user interface elements.