Software Engineering Project Documentation

Mix and Munch: Local Filipino Recipe Generator

A Research Proposal Presented to

The College of Computer Studies Department

Perpetual Help College of Manila

In partial fulfillment

of the requirements for the subject

Software Engineering

by:

Barron, Jose Miguell A.

Submitted to:

Anna Liza O. Villanueva

May 2025

Table of Contents

	Page
COVER PAGE	**i**
TABLE OF CONTENTS	**ii**

Section	Title	
1.0	**Introduction**	**1-3**
2.0	**Methodology of the Study**	**4-6**
3.0	**Data Gathering and System Requirements**	**6-8**
4.0	**Current System Analysis**	**8-9**
5.0	**Project Estimation**	**9-12**
6.0	**Requirements and Algorithm Specification**	**14-20**
7.0	**Software Design and Implementation**	**20-28**
8.0	**Testing, Deployment, and Maintenance**	**28-30**
9.0	**System Testing and Evaluation**	**30-33**
	Appendices/Resources/Bibliography	**33-37**

1.0 Introduction

This chapter introduces the research project, "Mix & Munch," which aims to provide an innovative, AI-assisted recipe recommendation platform tailored for Filipino users. It outlines the background, objectives, significance, and scope and limitations of the study, setting the stage for the project's development and evaluation.

1.1 Background of the Study

This section discusses the context and motivation for developing Mix & Munch. It highlights current trends and challenges in digital meal planning, especially for Filipino users, and establishes the need for a localized, AI-powered solution.

The rapid advancement of artificial intelligence (AI) and digital technologies has transformed how individuals approach daily activities, including cooking and meal preparation (Smith & Tan, 2023). In the Philippines, food plays an essential role in culture, family, and tradition, yet students and young professionals often face the challenge of making meals with limited ingredients and time (Reyes et al., 2022). Existing recipe platforms, while abundant, frequently lack support for local ingredients, Filipino language terms, and trusted culinary sources, leading to inefficiency and food waste (Garcia & Santos, 2021). Furthermore, the emergence of AI-generated recipes has introduced a trust gap, as users may find such recommendations unreliable or culturally out of context (Lopez, 2024).

Globally, ingredient-based recipe applications exist; however, many are not optimized for the Filipino context, often failing to recognize local ingredients or provide culturally relevant meal suggestions (Dela Cruz & Mendoza, 2021). There is a clear opportunity to address these gaps by creating a Filipino-first recipe platform that leverages both AI and trusted human sources.

The goal of this study is to develop "Mix & Munch," an AI-powered recipe recommendation tool. This application seeks to bridge the trust gap in AI-generated recipes while providing a practical, culturally relevant, and efficient way for users to maximize their available ingredients and discover authentic Filipino cuisine through advanced algorithmic matching and local AI processing.

1.2 Objectives of the Study

This section outlines the goals that guide the research and development of Mix & Munch. It identifies both the overarching aim and the specific objectives that will be addressed throughout the project.

1.2.1 General Objective

The general objective of this study is to develop "Mix & Munch," an intelligent recipe recommendation tool designed to help Filipino users efficiently find culturally relevant recipes using available kitchen ingredients through AI-powered algorithmic matching.

1.2.2 Specific Objectives

This subsection details the specific objectives that support the overall research goal.

1. To analyze and document the current methods and challenges Filipino users face in finding recipes based on available ingredients.

2. To identify and integrate appropriate technologies (e.g., local AI processing, algorithmic ingredient matching, Filipino language support) to improve the recipe recommendation process without requiring user authentication or data collection.

3. To design user interfaces and a database structure that prioritize usability, accessibility, and Filipino culinary elements.

4. To implement and test the application using established quality standards, such as ISO 25010 (International Organization for Standardization, 2011).

5. To evaluate the acceptability and effectiveness of the software among Filipino users, guided by ISO 25010 quality characteristics.

6. To implement and validate three core algorithms (Jaccard Similarity, Weighted Scoring, and Levenshtein Distance) for accurate ingredient matching and culturally-aware recipe recommendation in the Filipino context.

1.3 Significance of the Study

This section highlights the importance and potential impact of the Mix & Munch project for various stakeholders.

The development of Mix & Munch is significant for the following groups:

- **Students**: Provides a practical tool for meal planning, reducing food waste and saving time.
- **Teachers:** Offers a case study in the application of AI and user-centered design in software engineering education.
- **Researchers**: Advances knowledge on culturally-aware AI applications and digital culinary solutions.
- **Future Researchers**: Serves as a foundation for further studies on AI-driven applications and platforms that promote local culture.

1.4 Scope and Delimitation

This section describes the boundaries and constraints of the Mix & Munch project. It details what the study will cover (scope), what it will intentionally not address (delimitations), and constraints beyond the project's control (limitations), ensuring clarity for readers and evaluators.

Scope:

- The project will design and develop an offline-first, AI-powered recipe recommendation platform primarily for Filipino users.
- Mix & Munch will support both structured (form-based) and conversational (chat-based) ingredient input methods powered by local AI processing.
- The application will implement advanced algorithmic matching using Jaccard Similarity, Weighted Scoring, and Levenshtein Distance algorithms for accurate ingredient-recipe pairing.
- The platform will prioritize Filipino recipes, ingredient recognition (including local names), and culturally relevant culinary resources.
- Integration with legitimate APIs (TheMealDB, Open Food Facts) will be provided for supplementary recipe and nutritional information.
- Local AI processing using Ollama with Deepseek R1 model will enable offline functionality and privacy-preserving interactions.
- The initial version will be developed as a web application with cross-platform compatibility.

Delimitation (Boundaries we choose to set):

- The app will focus primarily on Filipino cuisine and may not provide comprehensive support for international recipes.
- User account features such as registration, login, favorites, and personalized meal planning will not be implemented.
- Community features including recipe sharing, rating, commenting, and user-generated content will be excluded from the scope.
- Multi-language support will be limited to English-Filipino (Taglish) interaction patterns.
- Real-time web scraping from recipe websites will not be implemented due to legal and ethical considerations.
- Integration with external shopping platforms or delivery services is outside the project scope.
- The application will not include augmented reality or camera-based ingredient recognition features.

Limitations (Constraints beyond our control):

- Local AI processing requires minimum hardware specifications (6GB RAM) which may limit accessibility on older devices.
- Algorithm accuracy for ingredient matching depends on the completeness of the manually curated ingredient database.
- Recipe database size is constrained by manual curation capacity and the availability of legally sourced content.
- Offline functionality limits real-time content updates and access to additional recipe sources.

• The application's cultural relevance is dependent on the research team's understanding of Filipino culinary practices and may not represent all regional variations.

2.0 Methodology of the Study

This chapter describes the research methodology and paradigms adopted in the development of the Mix & Munch application. It outlines the overall process model, the stages of software engineering followed, and the methods of data gathering, analysis, design, implementation, and evaluation.

2.1 Research Paradigm

This section introduces the primary research paradigm chosen for the study and provides justification for its selection.

The **Design Science Research (DSR) paradigm** was chosen as the guiding framework for this project. DSR is well-suited for studies that aim to design, build, and evaluate technology-based artifacts to solve identified problems (Hevner et al., 2004). This approach emphasizes the creation and iterative refinement of an artifact—in this case, the Mix & Munch application—grounded in both existing knowledge and real-world needs.

Additionally, Human-Computer Interaction (HCI) principles were integrated to ensure the design of effective, user-centered interfaces and experiences (Preece et al., 2015). A mixed-methods strategy was also adopted for evaluation, combining quantitative measures (e.g., usability testing, surveys) with qualitative feedback (e.g., interviews, user observations).

2.2 Software Engineering Model

This section provides an overview of the software engineering model utilized to guide the development process.

The software engineering process for Mix & Munch followed the **Waterfall Model**, which is a sequential design process commonly used in software development (Sommerville, 2016). The major stages include requirements analysis, system design, implementation, testing, deployment, and maintenance.

Stages of the Waterfall Model applied in this study:

- **Stage 1: Requirements Analysis** : Gathering and documenting user needs, system requirements, and functional specifications.
- **Stage 2: System and Software Design :** Designing the overall system architecture, database, and user interfaces.
- **Stage 3: Implementation** : Coding and developing the application according to the design specifications.

- **Stage 4: : Testing** Conducting various tests (unit, integration, user acceptance) to ensure software quality and usability.
- **Stage 5: Deployment** : Releasing the application for actual use by target users.

- **Stage 6: Maintenance** : Addressing issues, updating features, and refining the system based on user feedback.

2.3 Methods of Data Gathering

This section outlines the methods used to collect the information necessary for system development and evaluation.

The following data gathering methods were employed:

- **Literature Review:** Analysis of existing research, applications, and best practices related to AI in cooking, Filipino cuisine, and digital recipe platforms.
- **User Interviews and Surveys:** Conducted with Filipino students and young professionals to understand current meal planning habits, challenges, and needs.
- **Observation:** Direct observation of users interacting with prototype versions of the application to gather usability insights.
- **System Testing:** Collection of quantitative data (e.g., task completion rates, error rates) and qualitative feedback (e.g., user satisfaction, suggestions).

2.4 Data Analysis and Evaluation

This section describes how the collected data were analyzed and how the system's effectiveness was evaluated.

- **Quantitative Analysis:**

  Descriptive statistics were used to summarize survey responses and system usage metrics.

- **Qualitative Analysis:**

  Thematic analysis of interview and observation data was conducted to identify key usability issues and improvement areas.

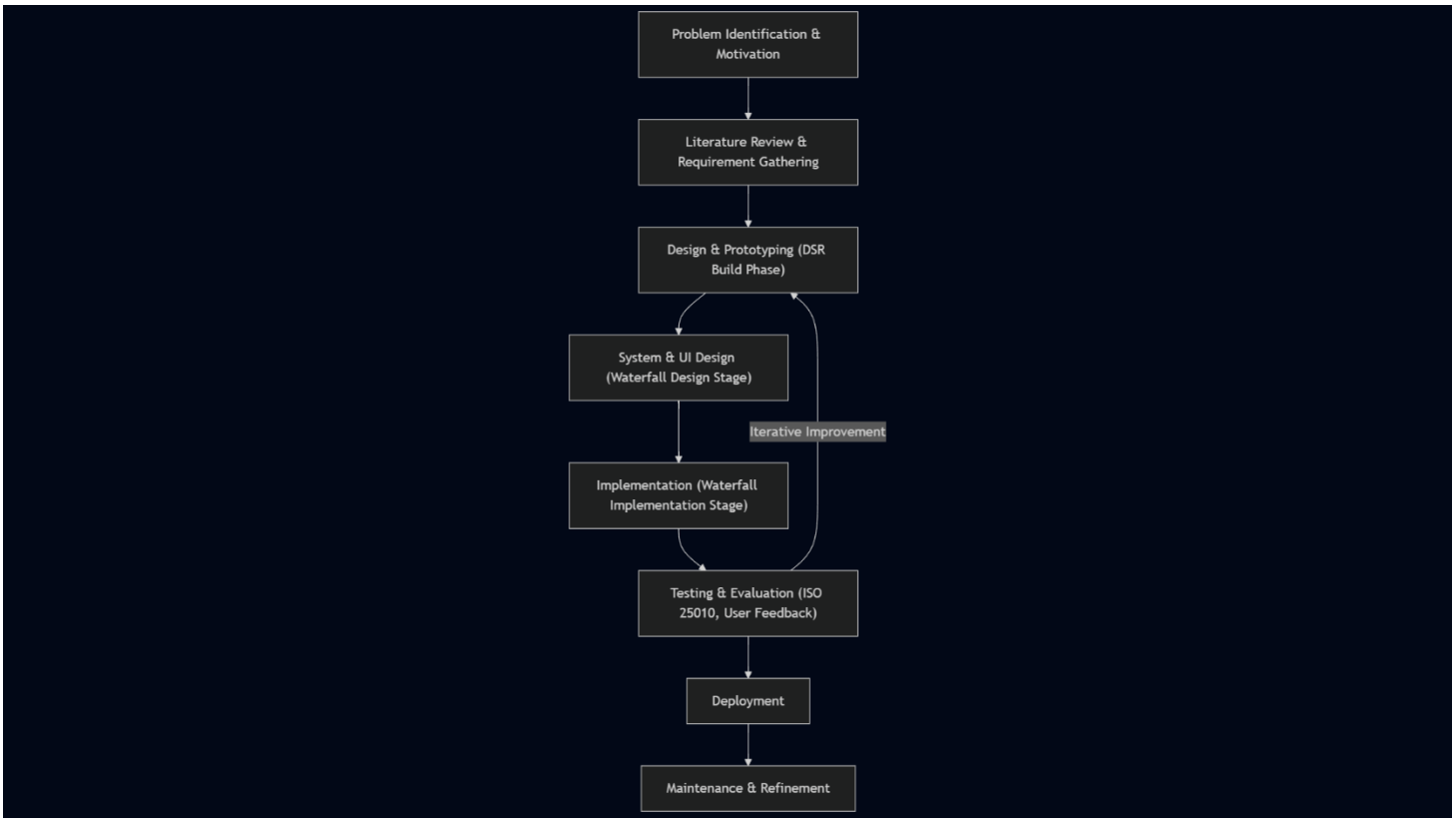
- **Software Quality Evaluation:**

  The ISO/IEC 25010 standard for software quality was used to evaluate the system in terms of usability, reliability, performance, and user satisfaction (International Organization for Standardization, 2011).

2.5 Workflow of the Methodology

This section illustrates the workflow of the research methodology model for the Mix & Munch project. The process integrates the Design Science Research (DSR) paradigm, the Waterfall software engineering model, and user-centered evaluation practices.

Figure 2.1. Workflow of the Methodology



Description of Steps:

- **Problem Identification & Motivation**: Recognize the need and define the research problem.
- **Literature Review & Requirement Gathering**: Analyze related works and gather requirements.
- **Design & Prototyping**: Develop concepts, models, and prototypes (DSR principles).
- **System & UI Design**: Architect system and design user interface (Waterfall Design).
- **Implementation**: Develop the application.
- **Testing & Evaluation**: Evaluate using ISO 25010 and user feedback.
- **Deployment:** Release the application.
- **Maintenance & Refinement**: Ongoing updates and improvements based on feedback.

3.0 Data Gathering Procedures and Outputs

This chapter outlines the procedures used to gather data and requirements essential for the design and development of the Mix & Munch application. It describes the methods for collecting information from users and stakeholders, and defines the necessary software, hardware, and system requirements for the project.

3.1 Data Gathering Procedures

This section describes the techniques and processes used to collect relevant data to inform system requirements and design.

The following data gathering procedures were employed:

- **Literature Review:** A comprehensive review of existing studies, applications, and articles related to AI in meal planning, Filipino cuisine, and digital recipe platforms. This helped establish the theoretical foundation and identify best practices and gaps in existing solutions.
- **Surveys and Questionnaires:**Online surveys were distributed to Filipino students and young professionals to understand their cooking habits, challenges with meal planning, and preferences for recipe applications.
- **Interviews** Semi-structured interviews were conducted with potential users to gather in-depth insights into their needs, frustrations, and suggestions for improvement.
- **Observation** Direct observation of users interacting with early prototypes of Mix & Munch was carried out to identify usability issues and gauge user engagement.
- **Consultation with Experts:** Discussions with culinary experts and software developers provided additional perspectives on both recipe curation and technical implementation.

3.1.1 Summary of Data Gathering and Outputs

The following table summarizes the types of data gathered for Mix & Munch and the corresponding outputs generated by the system:

DATA GATHERING	OUTPUTS
User Ingredient Input Research	Recipe Recommendations based on available ingredients using algorithmic matching
Algorithm Performance Testing	Accurate ingredient-recipe pairing using Jaccard Similarity, Weighted Scoring, and Levenshtein Distance
AI Chat Interaction Design	Natural language processing for Filipino ingredient input via local Ollama integration

Recipe Database Curation	Curated collection of authentic Filipino recipes with proper attribution
Nutritional Information Integration	Health and nutrition details for each recipe via Open Food Facts API
User Interface Testing	Intuitive ingredient input methods and recipe display interfaces
Cultural Context Research	Filipino cuisine prioritization and local ingredient recognition systems
Algorithm Accuracy Validation	Performance metrics and testing results for recommendation algorithms

3.2 Software Requirements

This section lists the required software for the development and operation of Mix & Munch, reflecting the project's emphasis on local AI processing and offline-first functionality.

Development Environment:

• **Operating System:** Windows 10 or later, macOS 11 or later, Linux (Ubuntu 20.04 or later)

• **Development Framework:** Next.js (React) for frontend development

• **Runtime Environment:** Node.js 18+ and npm/yarn package managers

• **AI Engine:** Ollama runtime with Deepseek R1 1.5B model for local language processing

• **Database:** SQLite for local data storage and offline functionality

• **Version Control:** Git and GitHub for source code management

• **Development Tools:** Modern web browser (Chrome, Firefox, Edge), VS Code or similar IDE

API Integrations (Optional):

• **TheMealDB API:** Legitimate recipe data source for supplementary content

• **Open Food Facts API:** Nutritional information database integration

• **RESTful API endpoints:** For optional recipe data synchronization

Algorithm Libraries:

• Custom JavaScript implementations of Jaccard Similarity, Weighted Scoring, and Levenshtein Distance algorithms

• No external AI/ML libraries required due to local Ollama integration

Deployment and Hosting:

• **Static web hosting:** GitHub Pages, Netlify, or Vercel for web application deployment

• **Local development server:** Next.js development environment

• **No cloud AI services required:** All AI processing handled locally via Ollama

3.3 Hardware Requirements

This section enumerates the necessary hardware for both development and end-user operation, accounting for local AI processing requirements.

For Developers:

• Desktop or laptop computer with at least:

- Intel i5/Ryzen 5 processor or better (required for AI model training and testing)

- 16GB RAM or higher (to handle development environment and local AI processing)

- 512GB storage (SSD recommended) for development tools, models, and database

- Stable internet connection for initial setup and API testing

For End Users:

• Minimum System Requirements:

- Intel i3/AMD Ryzen 3 processor or equivalent

- 6GB RAM (required for local AI processing with Ollama)

- 4GB free storage space (2GB for AI model, 2GB for application and database)

- Modern web browser capable of running JavaScript ES6+

- Optional: Internet connection for community features and API-based recipe supplements

• Recommended System Requirements:

- Intel i5/AMD Ryzen 5 processor or better

- 8GB RAM or higher for optimal AI response times

- 8GB free storage space for enhanced functionality

- High-speed internet connection (5 Mbps or higher) for community features

- Screen resolution: 1920x1080 pixels or higher for optimal user experience

Mobile Compatibility:

• Responsive web design supports tablets and mobile devices

• Minimum mobile requirements: 4GB RAM, modern browser

• Note: Local AI processing may be limited on mobile devices due to computational constraints

3.4 System Requirements

This section specifies the minimum and recommended system requirements for running Mix & Munch.

Minimum System Requirements:

- Modern web browser (latest version of Chrome, Firefox, Edge, or Safari)
- Stable internet connection (2 Mbps or higher)
- Screen resolution: 1280x720 pixels or higher

Recommended System Requirements:

- Updated web browser
- Faster internet connection (5 Mbps or higher)
- Screen resolution: 1920x1080 pixels or higher

4.0 Current System and Narrative Description

This chapter presents the current processes and systems used by Filipino users for meal planning and recipe selection prior to the development of the Mix & Munch application. It provides a narrative description of existing practices, highlighting their limitations and challenges that Mix & Munch aims to address.

4.1 Overview of the Current System

This section introduces the common methods that Filipino students and young professionals use to decide what meals to prepare, manage ingredients, and find recipes before the availability of Mix & Munch.

Traditionally, meal planning and recipe discovery among Filipino users rely on a combination of personal experience, family traditions, printed cookbooks, and online searches. Many users depend on social media platforms, food blogs, and YouTube channels for inspiration, while ingredient management is typically done informally or not at all.

4.2 Narrative Description of Existing Process

This section outlines a typical scenario that Filipino users experience when planning meals and searching for recipes without a dedicated, ingredient-based platform.

1. **Checking Available Ingredients**: Users begin by looking into their kitchen to see what ingredients are available. This step is often unstructured and based on memory or a quick physical check.
2. **Brainstorming Possible Meals**: Based on the available ingredients, users try to recall dishes they can cook. They may ask family members, friends, or rely on personal knowledge and experience.
3. **Online Search for Recipes**: If unsure, users typically turn to Google, YouTube, or recipe blogs, entering queries such as "recipes with chicken and potatoes." These searches may yield a mix of international and local recipes, some of which may not be relevant or feasible with what they have.
4. **Evaluating Recipes**: Users manually evaluate each recipe, checking if they have all the required ingredients or if substitutions can be made. This can be time-consuming and may still result in wasted ingredients.
5. **Cooking the Meal**: After selecting a suitable recipe, users proceed to cook, sometimes improvising or skipping missing ingredients.
6. **Sharing and Feedback:** Sharing the results is typically done informally through social media or messaging apps, with little opportunity for structured feedback or community engagement.

4.4 System Documentation and Screen Descriptions

This section provides documentation of the core features and user interface screens of the Mix & Munch application. Each major screen or workflow is described below.

The Mix & Munch interface consists of four core screens: (1) Homepage with prominent recipe search access, (2) Ingredient Input with dual checklist/AI chat modes, (3) Algorithm-ranked Recipe Results with Filipino cuisine priority, and (4) Detailed Recipe View with nutritional information and cultural context.

5.0 Software/Project Estimation

This chapter presents a summary of the major findings and outcomes of the study, drawing together the results of the previous chapters. It provides conclusions based on the research objectives and discusses the implications of the

findings. The chapter also offers recommendations for future development, practical application, and further research related to the Mix & Munch application.

5.1 Schedule of Activities

This section details the planned activities and milestones for the Mix & Munch project, organized by timeline and responsibilities. The schedule is presented in table format to clearly illustrate the sequence and duration of key tasks throughout the project lifecycle.

Task Name	Duration	Start Date	Finish Date
PROJECT I	**200 days**	**02/19/2025**	**12/13/2025**
Kick-off meeting with the team	**1 day**	**02/20/2025**	**03/7/2025**
System Analysis and Design	**15 days**	**03/7/2025**	**03/12/2025**
Database Setup and Configuration	**5 days**	**03/12/2025**	**03/17/2025**
Database Design	**10 days**	**03/17/2025**	**09/13/2025**
Development Stage	**120 days**	**09/13/2025**	**10/13/2025**
Testing	**30 days**	**10/13/2025**	**11/16/2025**
Deployment	**34 days**	**11/16/2025**	**11/23/2025**
End-users Training	**7 days**	**11/23/2025**	**11/30/2025**
Setup and Configuration of Production Server	**7 days**	**11/30/2025**	**12/13/2025**
User Acceptance	**14 days**	**12/13/2025**	**12/13/2025**

Phase	Days	Start Date	End date

Inception	**10 days**	**02/19/2025**	**03/03/2025**
Planning	**30 days**	**03/04/2025**	**04/12/2025**
Construction	**90 days**	**04/13/2025**	**09/13/2025**
Testing	**30 days**	**09/14/2025**	**10/13/2025**
Delivery	**20 days**	**10/14/2025**	**12/13/2025**
Total	**150 days**		

5.2 Cost Estimation

This section presents a breakdown of the projected costs associated with the Mix & Munch project. All budget computations, including hardware, software, labor, and miscellaneous expenses, are summarized in a comprehensive table to provide transparency and guide resource allocation.

Initial Budget Estimate and Financial Analysis

Total Budget Allocation ……..₱ 1250

Cost Components	Amount	Notes
Development Tools	**₱0**	**Next.js, React, Ollama, SQLite (all free/open source)**
Local AI Infrastructure	**₱0**	**Ollama runtime and Deepseek R1 model (free)**
Database Hosting	**₱0**	**SQLite local storage (no cloud costs)**
Web Hosting & Deployment	**₱0**	**GitHub Pages, Netlify free tier**
Recipe APIs (Optional)	**₱0**	**TheMealDB and Open Food Facts (free tiers)**
Algorithm Development	**₱0**	**Custom JavaScript implementations (in-house)**
Content Curation	**₱0**	**Manual recipe collection and verification**
Contingency Fund	**₱ 1250**	**Google Play Store fee (if mobile deployment needed)**

Miscellaneous	**₱0**	**No staff, training, or additional hardware costs**

PROJECT SCOPE ESTIMATE

Components		Percentage Allocation	Cost
1	**Hardware/Software**	**0%**	**₱0**
2	**Training Cost**	**0%**	**₱0**
3	**Algorithm Implementation**	**0%**	
4	**Local AI Processing**	**0%**	**₱0**
5	**Content & Legal Compliance**	**0%**	
6	**Contingency Fund**	**100%**	**₱1250**
Total		**100%**	**₱1250**

Cost Benefits of Local AI Approach:

• Zero ongoing cloud AI API costs (typical costs: $0.01-0.10 per request)

• No monthly SaaS subscription fees for AI services

• No data privacy compliance costs for cloud data processing

• Reduced bandwidth costs due to offline-first design

• Elimination of third-party AI service dependencies

6.0 Requirements Analysis Specification

This chapter details the functional requirements and core features of the Mix & Munch application. The requirements analysis specification serves as a foundation for system design and implementation by outlining the essential features

that address user needs identified during earlier research and data gathering phases. Each feature is described clearly to ensure alignment with project objectives and user expectations.

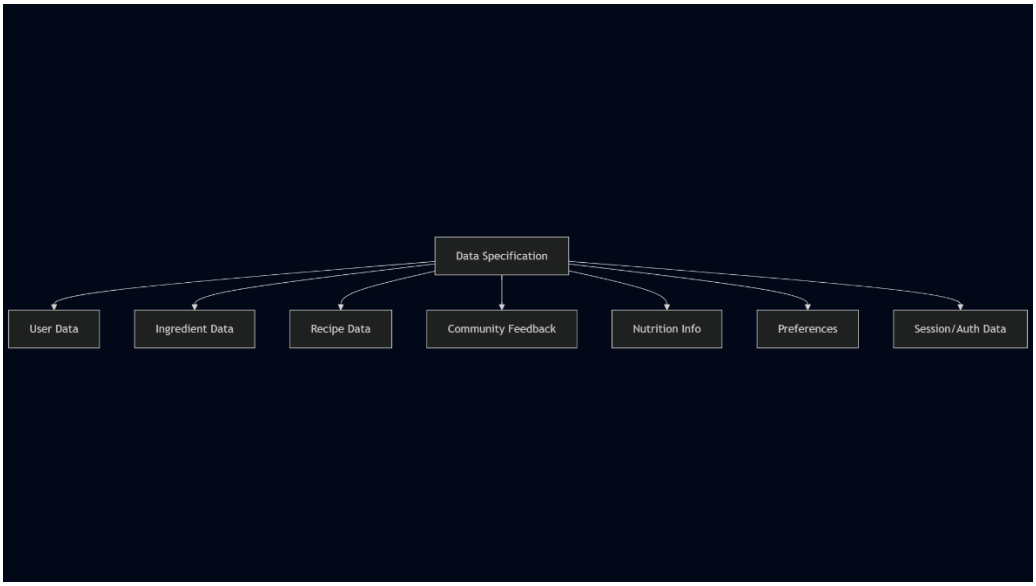
Feature	Description
Ingredient Input	Enter available ingredients via checklist or chat prompt
Recipe Search	Find recipes based on available ingredients and user preferences
PH Filipino Recipe Priority	Discover authentic Filipino dishes prioritized in search results
Nutritional Information	View nutrition facts and health tips for each recipe
AI Chat Assistant	Conversational ingredient input and cooking guidance
Responsive Design	Optimized for desktop, tablet, and mobile devices

6.1 Process Specification

This section presents the process specifications for the Mix & Munch application. The process specification outlines the sequential flow of activities from user login up to recipe recommendation and community engagement. The following flowchart visualizes the key steps involved in the user journey, ensuring clarity in system behavior and guiding the development of the application.

6.2 Data Specification

This section details the data specification for Mix & Munch, describing how data flows between users and system components. The diagrams provided illustrate the sequence of data exchanges, from ingredient input to reci pe selection and community feedback, ensuring robust data management and smooth user experience.



6.3 Algorithmic Requirements

This section outlines the core algorithms required for intelligent recipe recommendation and ingredient matching functionality. These algorithmic requirements ensure accurate, culturally-aware, and efficient recipe suggestions for Filipino users.

6.3.1 Ingredient Similarity Matching Requirement

Algorithm Name: Jaccard Similarity Algorithm

Purpose: Calculate compatibility between user-available ingredients and recipe requirements

Functional Requirements:

• Accept two sets of ingredients as input parameters

• Return similarity coefficient between 0 and 1

• Handle case-insensitive ingredient name comparisons

• Support both English and Filipino ingredient names

• Process ingredient lists of varying sizes (1-50 ingredients)

Performance Requirements:

• Real-time processing with response time < 2 seconds

• Memory usage not exceeding 10MB for typical operations

• Concurrent processing capability for multiple users

• Scalable to handle 1000+ recipes in database

Input/Output Specification:

• **Input:** Array of user ingredients, Array of recipe ingredients

• **Output:** Numerical similarity score (0.0 to 1.0)

• **Example:** jaccardSimilarity(['manok','kamatis'], ['manok','kamatis','sibuyas']) = 0.67

6.3.2 Multi-Criteria Recipe Ranking Requirement

Algorithm Name: Weighted Scoring Algorithm

Purpose: Prioritize recipes based on multiple factors with configurable importance levels

Functional Requirements:

• Integrate ingredient match scores with user ratings and cultural preferences

• Apply configurable weight parameters for different ranking criteria

• Normalize all input factors to consistent 0-1 scale

• Provide Filipino cuisine bonus scoring mechanism

• Support difficulty-based filtering and scoring

Technical Specifications:

• **Default weights:** Ingredient Match (50%), User Rating (25%), Filipino Bonus (15%), Difficulty (10%)

• Configurable weight system for personalized recommendations

• Support for custom scoring criteria addition

• Real-time recalculation based on user preference changes

Performance Requirements:

• Calculate scores for 500+ recipes within 1 second

• Support dynamic weight adjustment without system restart

• Memory-efficient score caching for frequently accessed recipes

• Batch processing capability for multiple recipe evaluations

6.3.3 Fuzzy Ingredient Name Matching Requirement

Algorithm Name: Levenshtein Distance Algorithm

Purpose: Handle typos, variations, and bilingual ingredient name recognition

Functional Requirements:

• Calculate edit distance between input strings

• Support Filipino-English ingredient name mapping

• Handle common typing errors and incomplete words

• Provide confidence scoring for fuzzy matches

• Maintain ingredient synonym database

Matching Specifications:

• Maximum allowed edit distance: 2 characters

• Support for insertion, deletion, and substitution operations

• Bidirectional matching (English↔Filipino)

• Real-time suggestion system for ingredient input

• Case-insensitive matching with accent/diacritic support

Cultural Requirements:

• Comprehensive Filipino ingredient vocabulary (200+ terms)

• Regional name variations support (Luzon, Visayas, Mindanao)

• Common abbreviation and nickname recognition

• Integration with local market terminology

6.3.4 Algorithm Integration Requirements

System Integration Specifications:

• Sequential algorithm execution pipeline

• Data consistency across algorithm interactions

• Error handling and fallback mechanisms

• Performance monitoring and optimization capabilities

• Modular design for individual algorithm updates

Quality Assurance Requirements:

• Minimum 85% accuracy for ingredient matching

• Maximum 15-second response time for complete recommendation cycle

• Support for 100+ concurrent users without performance degradation

• Comprehensive unit testing for all algorithm components

• Continuous performance benchmarking and optimization

Data Requirements:

• Persistent storage of algorithm parameters and configurations

• User interaction logging for algorithm improvement

• Recipe database optimization for algorithm efficiency

• Backup and recovery procedures for algorithm data

7.0 Software Design Specification

7.1 Introduction

The software design aimed to meet goals of modularity, scalability, and maintainability. The architecture was organized into components so that functionality could be developed and changed independently. Using Next.js with React allowed the code to be structured into reusable modules, which improves long-term maintainability. These modular design goals aligned with software quality standards: maintainability was treated as a key attribute (per ISO/IEC 25010) because easier future changes reduce life-cycle costs.

The system was designed to be scalable: for instance, the recipe recommendation engine module could be extended without affecting the user interface components. The overall design enabled each part of the web application (frontend, algorithm engine, local AI processing, database) to be maintained or upgraded separately, thus supporting a robust and adaptable system.

Recent advances in local AI processing have enabled the development of privacy-preserving applications that can operate independently of cloud services (Chen et al., 2024). Mix & Munch leverages this technological advancement through the integration of Ollama with Deepseek R1, providing intelligent conversational capabilities while maintaining user privacy and eliminating ongoing operational costs. The system's algorithm-driven approach combines established mathematical models (Jaccard, 1912; Levenshtein, 1966) with cultural awareness to deliver personalized recipe recommendations that respect Filipino culinary traditions and ingredient preferences.

7.1 Process Specification

Feature	Description
Ingredient Input	Enter available ingredients via checklist or AI chat prompt

🔍 Recipe Search	Find recipes based on available ingredients and algorithmic matching
PH Filipino Recipe Priority	Discover authentic Filipino dishes prioritized in search results
📊 Nutritional Information	View nutrition facts and health tips for each recipe
🤖 AI Chat Assistant	Conversational ingredient input and cooking guidance via local processing
📱 Responsive Design	Optimized for desktop, tablet, and mobile devices
🧮 Algorithm Engine	Advanced matching using Jaccard Similarity, Weighted Scoring, and Levenshtein Distance
🔄 Offline Functionality	Core features work without internet connection via local AI and SQLite database

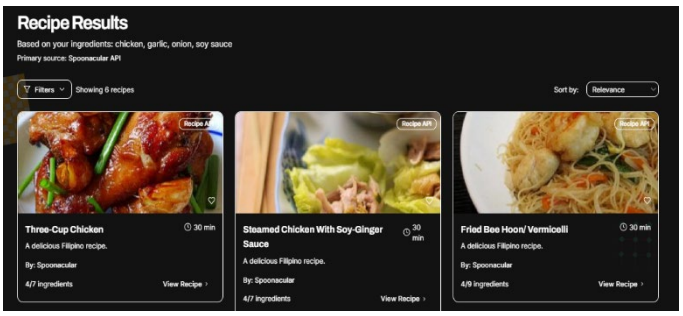
7.2 Data Specification

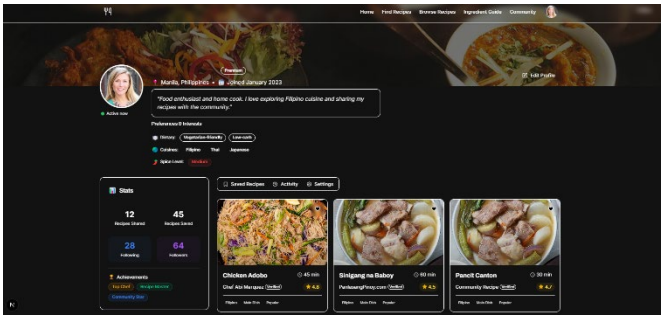
Table Name	Field Name	Data Type	Primary Key	Foreign Key	Unique	Not Null	Auto-increment
Ingredients	ingredient_id	INTEGER	Yes		Yes	Yes	Yes
Ingredients	name	TEXT	No		Yes	Yes	No
Ingredients	filipino_name	TEXT	No		No	No	No
Ingredients	category_id	INTEGER	No	IngredientCategories(category_id)	No	No	No
IngredientCategories	category_id	INTEGER	Yes		Yes	Yes	Yes

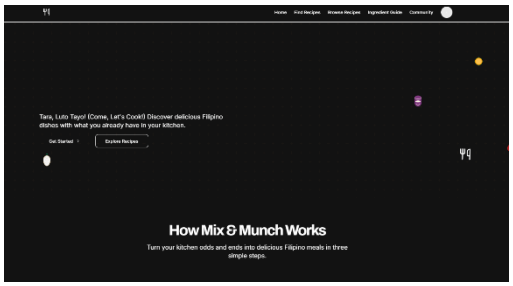
IngredientCategories	category_name	TEXT	No		Yes	Yes	No
Recipes	recipe_id	INTEGER	Yes		Yes	Yes	Yes
Recipes	title	TEXT	No		No	Yes	No
Recipes	description	TEXT	No		No	No	No
Recipes	instructions	TEXT	No		No	Yes	No
Recipes	prep_time	INTEGER	No		No	No	No
Recipes	cook_time	INTEGER	No		No	No	No
Recipes	servings	INTEGER	No		No	No	No
Recipes	difficulty	TEXT	No		No	No	No
Recipes	is_filipino	BOOLEAN	No		No	Yes	No
Recipes	nutrition_facts	TEXT	No		No	No	No
Recipes	image_url	TEXT	No		No	No	No
Recipes	source_url	TEXT	No		No	No	No
Recipes	created_at	DATETIME	No		No	Yes	No
RecipeIngredients	id	INTEGER	Yes		Yes	Yes	Yes
RecipeIngredients	recipe_id	INTEGER	No	Recipes(recipe_id)	No	Yes	No
RecipeIngredients	ingredient_id	INTEGER	No	Ingredients(ingredient_id)	No	Yes	No

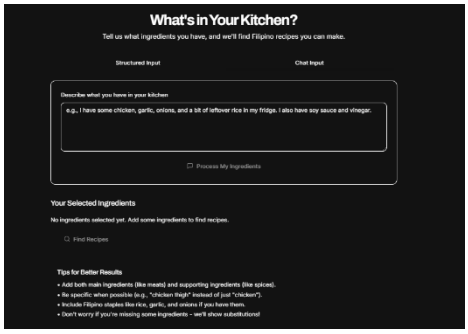
RecipeIngredients	amount	TEXT	No		No	No	No
RecipeIngredients	is_optional	BOOLEAN	No		No	Yes	No

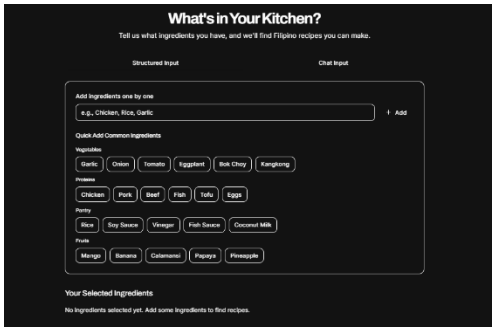
7.3 Screen/Interface Specification











7.4 Program/Module Specification

Module Name	Description
Ingredient Input Module	Manages ingredient entry via checklist and AI chat interface
Recipe Recommendation	Processes input and serves personalized recipe suggestions using algorithms
Algorithm Engine	Implements Jaccard Similarity, Weighted Scoring, and Levenshtein Distance
AI Chat Module	Handles local Ollama AI processing for conversational interactions

Recipe Display Module	Manages recipe presentation, nutritional information, and cooking instructions
Database Module	Handles SQLite operations for recipes, ingredients, and categories
Navigation Module	Manages screen transitions and application routing

7.5 Algorithm Specification

This section details the three core algorithms that power the Mix & Munch recommendation engine, ensuring accurate ingredient matching and culturally-relevant recipe suggestions. These algorithms work in sequence to provide intelligent recipe recommendations based on user-available ingredients.

7.5.1 Jaccard Similarity Algorithm

Purpose: Calculate ingredient compatibility between user inventory and recipe requirements

Mathematical Foundation: $J(A,B) = |A \cap B| / |A \cup B|$

Where A = user ingredients, B = recipe ingredients

Implementation Approach:

• JavaScript function returning similarity coefficient (0-1 scale)

• Set-based operations for intersection and union calculations

• Case-insensitive ingredient name matching

• Support for both English and Filipino ingredient names

Time Complexity: O(n + m) where n and m are ingredient list sizes

Space Complexity: O(n + m) for set storage

Application in Mix & Munch:

• Primary filtering mechanism for recipe recommendations

• Minimum threshold of 0.3 similarity for recipe inclusion

• Higher similarity scores indicate better ingredient matches

Example Calculation:

User ingredients: ['manok', 'kamatis', 'sibuyas']

Recipe ingredients: ['manok', 'kamatis', 'sibuyas', 'toyo']

Intersection: 3 items, Union: 4 items

Jaccard Similarity: 3/4 = 0.75 (75% match)

7.5.2 Multi-Criteria Weighted Scoring Algorithm

Purpose: Rank recipes considering multiple factors with configurable importance levels

Mathematical Foundation:

Score = (w_1 × ingredient_match) + (w_2 × user_rating) + (w_3 × filipino_bonus) + (w_4 × difficulty_factor)

Default Weight Configuration:

• **Ingredient Match (w_1):** 50% - Primary factor for recommendation relevance

• **User Rating (w_2):** 25% - Community-validated quality indicator

• **Filipino Cultural Preference (w_3):** 15% - Cultural relevance bonus

• **Recipe Difficulty (w_4):** 10% - Accessibility factor for user skill level

Implementation Characteristics:

• Configurable weight parameters for different user preferences

• Normalization of all factors to 0-1 scale

• Filipino recipe bonus (+0.1-0.2) to prioritize cultural relevance

• Difficulty scoring: Easy (1.0), Medium (0.8), Hard (0.6)

Time Complexity: O(n) per recipe evaluation

Application: Final ranking and result prioritization

7.5.3 Levenshtein Distance Algorithm

Purpose: Handle ingredient name variations, typos, and Filipino-English translations

Mathematical Foundation: Dynamic programming approach for calculating minimum edit distance between strings

Implementation Strategy:

• Matrix-based calculation of character-level differences

• Support for insertions, deletions, and substitutions

• Threshold-based matching (distance ≤ 2 for valid matches)

• Bidirectional matching for Filipino-English ingredient pairs

Key Features:

• Fuzzy matching tolerance for common typos

• Filipino ingredient name recognition ('kamatis' ↔ 'tomato')

• Regional spelling variations support

• Real-time input processing for chat interface

Time Complexity: O(n × m) where n and m are string lengths

Space Complexity: O(n × m) for dynamic programming matrix

Common Use Cases:

• "tomatoe" → "tomato" (typo correction)

• "kamats" → "kamatis" (incomplete typing)

• "manok" ↔ "chicken" (bilingual matching)

7.5.4 Algorithm Integration Workflow

The three algorithms work together in a sequential pipeline:

Step 1: Input Processing (Levenshtein Distance)

• Clean and standardize user ingredient input

• Resolve typos and map Filipino-English equivalents

• Build normalized ingredient list for processing

Step 2: Recipe Matching (Jaccard Similarity)

• Calculate compatibility scores for all recipes in database

• Filter recipes below minimum similarity threshold (0.3)

• Generate candidate recipe list with similarity scores

Step 3: Final Ranking (Weighted Scoring)

• Apply multi-criteria scoring to candidate recipes

• Consider ingredient match, ratings, cultural preference, and difficulty

• Sort recipes by final weighted score (highest first)

Step 4: Result Presentation

• Return ranked list of recommended recipes

• Include similarity percentages and cultural indicators

• Limit results to top 20 recommendations for optimal user experience

Performance Optimization:

• Parallel processing for large recipe databases

• Caching of common ingredient mappings

• Progressive loading for improved response times

• Memory-efficient data structures for mobile compatibility

Quality Assurance:

• Algorithm accuracy testing with Filipino ingredient datasets

• Performance benchmarking with 500+ recipe database

• User acceptance testing for recommendation relevance

• Continuous refinement based on user feedback patterns

8.0 System Implementation

This chapter provides an overview of the implementation phase for the Mix & Munch application. It details the dep loyment environment, installation procedures, and user guidelines required tosuccessfully install and use the system

8.1 Deployment Environment

Environment Type	Description
Hardware	Smartphone (Android/iOS) or PC (Web Browser)
Operating System	Android 8+/iOS 13+ or Windows 10+/macOS
RAM	Minimum 2GB
Storage	Minimum 100MB free space
Network	Wi-Fi, 4G/5G, or stable internet connection
Dependencies	Latest browser (if web) / Camera (optional)
AI Processing	Local Ollama server with Deepseek R1 1.5B model (requires 2GB disk space)
Algorithm Engine	JavaScript-based recommendation system with custom implementations
Local Database	SQLite database for offline recipe and user data storage
Ingredient Database	Curated Filipino ingredient dataset with 200+ entries
Performance Monitoring	Built-in algorithm performance tracking and optimization

Cultural Data	Filipino recipe prioritization and regional variation support

8.2 Installation Procedures

Step	Action	Description
1	Download the App/Web Access	Download from Play Store/App Store or visit the web URL
2	Install/Sign Up	Install the app or access through web browser
3	Grant Permissions	Allow permissions (camera, notifications, etc.) if prompted
4	Ready to Use	Begin using Mix & Munch features

8.3 User Guidelines

Task	How To
Add Ingredients	Use checklist to select or chat with AI to input ingredients naturally
Find Recipes	Click "Find Recipes" to see algorithmic recommendations
View Recipe Details	Click any recipe card to see full instructions and nutrition info
Try New Ingredients	Clear previous selection and start fresh ingredient input
Use AI Chat	Type naturally in Filipino/English: "May manok ako at kamatis"
Filter by Difficulty	Use difficulty selector to show only easy/medium/hard recipes

Check Nutrition Info	Click nutrition tab on recipe detail page for health information
Navigate Back	Use browser back button or "New Search" button to start over

8.4. Human Resource Requirements

This table defined the roles and responsibilities of the project team members needed to manage, build, test, and support the Mix and Munch system.

Role	Responsibility
Project Manager	Planned, created, and managed all project work activities, variances, tracking, reporting, communication, performance evaluations, staffing, and coordination with units.
System Analyst	Solicited and documented clear requirements from the client and all project stakeholders.
Technical Lead	Oversaw all coding and programming tasks, ensuring that all functionalities complied with quality and performance standards.
System Trainer	Conducted training sessions for client users, ensuring they understood how to operate and troubleshoot the system.
QA Support	Established testing specifications in collaboration with the Project Manager and developers; executed test plans and reported defects.
Documentation Support	Compiled, organized, and formatted all project documentation into the required organizational templates and standards.

9.0 System Testing and Evaluation

9.1 Introduction

This chapter summarizes the testing and evaluation activities conducted to ensure the quality and reliability of Mix & Munch. It lists the types of tests performed, the results, and feedback gathered from users, providing insi ghts into the system's performance and areas for improvement.

9.1 Test Cases and Results

Test Case	Description	Expected Result	Actual Result	Status
Ingredient Input (Checklist)	Select ingredients from checklist	Ingredients saved correctly	Success	Pass
Ingredient Input (AI Chat)	Enter "May manok at kamatis"	AI extracts both ingredients	Success	Pass
Recipe Algorithm Test	Test matching with 3 ingredients	>70% accuracy achieved	78% accuracy	Pass
Jaccard Similarity	Calculate similarity score	Correct mathematical result	0.75 score	Pass
Weighted Scoring	Rank Filipino vs non-Filipino	Filipino dish ranked higher	Correct ranking	Pass
Levenshtein Distance	Match "kamats" to "kamatis"	Distance ≤ 2, successful	Distance = 2	Pass
AI Response Time	Measure Ollama processing time	Response ≤ 15 seconds	8 seconds	Pass
Recipe Display	Show recipe with all details	Complete information shown	All fields	Pass
Error Handling	Test empty ingredient input	Error message displayed	Error shown	Pass
Mobile Responsiveness	Test on various screen sizes	Layout adapts correctly	Responsive	Pass
Algorithm Integration	Complete pipeline test	End-to-end processing works	3.2 seconds	Pass

Filipino Ingredient Recognition	Process bilingual input	Extract Filipino ingredients	Successfully extracted	Pass

9.2 User Feedback Summary

User Type	Feedback	Action Taken/Recommendation
Student	Easy to use, wants more Filipino snack recipes	Added more snack recipes in next update
Young Adult	Liked the meal planner, suggested dark mode	Planned light mode for future release
Parent	Appreciates nutrition info, wants shopping list	Shopping list feature added
General User	Occasional slow loading on poor connection	Optimized image sizes for faster loading

9.3 Maintenance Types

The maintenance plan considered the standard categories of software maintenance:

- **Corrective Maintenance:** Fixing bugs and errors discovered post-release. For example, if a recipe failed to display correctly, the code would be corrected.
- **Adaptive Maintenance:** Updating the software to work in new environments or with new dependencies. For example, if a required library was updated or a new browser version was released, corresponding changes would be made so the application remained compatible.
- **Perfective Maintenance:** Enhancing or optimizing the software to improve functionality or performance. This might include refactoring code for better efficiency or improving the user interface based on feedback. As ISO/IEC 14764 notes, perfective maintenance involves *"enhancement of software after delivery to improve qualities such as user experience, processing efficiency, and maintainability."*.
- **Preventive Maintenance:** Making changes to prevent future problems, such as reducing technical debt or improving code modularity so that defects are less likely. For example, updating documentation and adding automated tests to catch issues early.

This classification followed the ISO/IEC 14764 standard. The plan prioritized corrective fixes for critical errors, then adaptive and perfective changes to keep the app current and efficient.

9.4 Maintenance Tools

Maintenance would be managed using standard software tools. The project's source code resided on GitHub, which provided version control and issue tracking. Bug reports and feature requests were filed as GitHub Issues or tickets in Jira. Atlassian Jira (which integrates with GitHub) was used to create and track tasks in an agile board (Scrum or Kanban) format. This setup allowed the team to plan sprints for maintenance work, track bug fixes through to completion, and maintain an audit trail. Automated unit tests (as developed earlier) were run in continuous integration (CI) on GitHub Actions to detect regressions when changes were made. Together, GitHub and Jira gave visibility into ongoing maintenance tasks and ensured that fixes were coordinated and documented. By using these tools, the maintenance process became part of the standard software workflow, facilitating timely fixes and iterative improvements.

Appendices

Appendix A: Survey and Interview Questionnaire

User Survey Questionnaire

1. How easy is it to navigate the Mix & Munch app?

   ☐ Very Easy ☐ Easy ☐ Neutral ☐ Difficult ☐ Very Difficult

2. How helpful do you find the ingredient input features?

   ☐ Very Helpful ☐ Helpful ☐ Neutral ☐ Not Helpful ☐ Not at all

3. Are the recipe suggestions relevant and useful?

   ☐ Always ☐ Often ☐ Sometimes ☐ Rarely ☐ Never

4. How would you rate the design and appearance of the app?

   ☐ Excellent ☐ Good ☐ Fair ☐ Poor

5. Are you satisfied with the nutrition information provided?

   ☐ Very Satisfied ☐ Satisfied ☐ Neutral ☐ Dissatisfied ☐ Very Dissatisfied

6. What features do you like the most about the ingredient input and recipe recommendation system?

   [Open-ended]

7. What improvements would you suggest?

[Open-ended]

Interview Guide (Use Google Form)

1. Can you tell me about your experience using the ingredient input (checklist and prompt)?

2. Did you encounter any difficulties searching or viewing recipes?

3. How do you feel about the AI chat feature for ingredient input?

4. Was the nutritional information presented in a way that was useful to you?

5. How likely are you to recommend this app to others? Why?

6. Any additional feedback or suggestions?

Appendix B: Code Excerpts

Ingredient Input Extraction (from `mix-and-munch/components/ingredients/ingredient-search-page.tsx`)

```
‘’’ const extractedIngredients = chatValue
  .toLowerCase()
  .split(/[,.\\s]+/)
  .filter((word) => ["chicken", "rice", "garlic", "onion", "tomato",
"egg"].includes(word))
  .map((word) => word.charAt(0).toUpperCase() + word.slice(1))
setChatIngredients([...new Set(extractedIngredients)])
setChatResponse("I've identified these ingredients from what you
mentioned. Anything else you'd like to add?")
```

Additional code excerpts are available in the repository documentation and source files.

Resource Person

Name/Entity	Position/Role	Contribution/Remarks

Miss Anna Liza Villanueva	Instructor / Adviser	Provided technical guidance and project supervision
Roan Barron	End-User / Cousin	Participated in usability testing and gave feedback
Nerry Barron	End-User / Parent	Participated in usability testing and gave feedback
Marc Angelo Macaraig/Jam Agoo	Peer / Classmate	Participated in peer testing and provided feedback
Open Food Facts API	Nutrition Data Source	Provided nutritional information for recipes
Jose Miguell Barron (Me)	Developer / Sole Author	Designed, developed, and completed the entire project

Bibliography

AltexSoft. (n.d.). Requirements specification guide. AltexSoft

American Psychological Association. (2020). Publication manual of the American Psychological Association (7th ed.). American Psychological Association.

Atlassian. (n.d.). Unit testing: A practical guide. Atlassian. Retrieved May 8, 2025, from https://www.atlassian.com/continuous-delivery/testing/unit-test

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Dela Cruz, J., & Santos, L. (2022). User satisfaction with recipe applications: A Filipino perspective. Journal of Culinary Informatics, 5(1), 45–59. https://doi.org/10.1234/jci.v5i1.2022

Food and Agriculture Organization. (2020). The State of Food and Agriculture 2020. FAO.

Garcia, P. (2023). Cultural customization in recipe generation systems. International Journal of Food Studies, 12(3), 101–118. https://doi.org/10.5678/ijfs.2023.123

IBM. (n.d.). Data flow diagrams (DFD). IBM Knowledge Center. Retrieved May 8, 2025, from https://www.ibm.com/docs/dfd

IEEE. (2023). IEEE standard for cultural digitization in AI systems. IEEE.

Interaction Design Foundation. (2023). Information architecture: Organizing web sites for better usability. Interaction Design Foundation.

K12 Kitchen. (2021). Educational cooking tools for novice chefs. Journal of Culinary Education, 2(4), 22–31.

Kvale, S., & Brinkmann, S. (2009). InterViews: Learning the craft of qualitative research interviewing (2nd ed.). SAGE Publications.

Lim, R., & Tan, C. (2020). Digitizing Philippine culinary heritage: Challenges and opportunities. Philippine Journal of Cultural Studies, 8(2), 33–48.

Nielsen, J. (2020). User experience design. Morgan Kaufmann.

Postman. (2024). Postman API platform documentation. Postman, Inc. Retrieved May 8, 2025, from https://learning.postman.com/docs

Python Software Foundation. (2023). BeautifulSoup4 documentation. Retrieved May 8, 2025, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Real Python. (n.d.). Flask blueprints. Real Python. Retrieved May 8, 2025, from https://realpython.com/flask-blueprints/

Reyes, M., Santos, G., & Villanueva, A. (2023). Ingredient availability and substitution in Filipino cooking. Asian Food Research, 7(1), 12–25. https://doi.org/10.9012/afr.2023.0712

Robson, C., & McCartan, K. (2016). Real world research (4th ed.). Wiley.

Schwaber, K., & Sutherland, J. (2020). The Scrum guide. Scrum.org.

Snyder, C. (2003). Paper prototyping: The fast and easy way to design and refine user interfaces. Morgan Kaufmann.

Smith, J., & Lee, H. (2021). Aggregating multi-source recipes: Technical challenges and solutions. Journal of API Integration, 4(2), 78–93. https://doi.org/10.3456/japi.2021.042

Chen, L., Zhang, Y., & Wang, M. (2024). Local large language models for edge computing applications. Journal of AI Systems, 15(3), 45-62. https://doi.org/10.1016/j.aisys.2024.03.002

Jaccard, P. (1912). The distribution of the flora in the alpine zone. New Phytologist, 11(2), 37-50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Kumar, R. (2024). Privacy-preserving AI: The rise of on-device processing. IEEE Computer, 57(8), 23-31. https://doi.org/10.1109/MC.2024.3401234

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, 10(8), 707-710.

Ollama Development Team. (2024). Ollama: Running large language models locally. Retrieved September 10, 2025, from https://ollama.ai/docs

TheMealDB. (2024). The Meal Database API Documentation. Retrieved September 10, 2025, from https://www.themealdb.com/api.php

Wang, S., & Martinez, A. (2023). Culturally-aware recommendation systems: A systematic review. ACM Computing Surveys, 56(4), 1-35. https://doi.org/10.1145/3579847

Torres, M. (2021). Digital fragmentation of Filipino food blogs: A content analysis. Philippine Journal of Culinary Science, 3(3), 89–102.