Sous Duckling: A Smart Recipe Assistant

Afreen Ahmed
Texas A&M University
afreen04@tamu.edu

Ankitha Prasad
Texas A&M University
ankithap@tamu.edu

Khushi Patel
Texas A&M University
kp1032@tamu.edu

Navya Unnikrishnan
Texas A&M University
navya_unni@tamu.edu

Nikki Rad
Texas A&M University
nikkiraddd@tamu.edu

ABSTRACT

The project, *Sous Duckling: A Smart Recipe Assistant*, presents a novel intelligent user interface designed to act as a dynamic, personalized recipe assistant. Unlike traditional recipe applications, Sous Ducking goes beyond simple search by integrating user-specific constraints and preferences to generate and modify recipes in real-time. It captures user preferences, dietary restrictions, and nutritional goals to suggest recipes, provide ingredient substitutions, generate grocery lists, and deliver step-by-step cooking instructions. Recipes are represented in a structured and adaptable format that enables dynamic personalization during tasks and feedback-driven improvement over time. It is designed to facilitate a continuous, interactive dialogue with the user, allowing for a personalized and adaptive cooking experience. Sous Ducking aims to transform meal planning from a static task into an intuitive and interactive process.

1. INTRODUCTION

Cooking is an integral part of life for most people. The way we approach meal planning and cooking has evolved, yet the tools available to assist us often fail to meet the diverse and dynamic needs of modern users. Recipes are also often rigid, and don't allow customizations, leaving the user confused. This creates a gap for individuals who are short on time, need to accommodate specific dietary restrictions, or are simply looking for creative inspiration using the ingredients they have on hand.

Our project, *Sous Duckling*, is designed to directly address these issues by reimagining the recipe assistant as an intelligent and adaptive kitchen companion. It moves beyond the traditional model of a passive recipe database to create a proactive, context-aware tool. By leveraging user input on available ingredients, allergies, and personal preferences, Sous Duckling dynamically generates and modifies recipes, transforming the user's constraints into creative opportunities. This approach not only streamlines the meal preparation process but also makes cooking more accessible, enjoyable, and less intimidating.

By providing intelligent recipe recommendations and personalized guidance, our software will reduce decision fatigue, support healthy cooking habits, and make cooking a more engaging and stress-free experience. Additionally, its conversational and interactive design helps users feel more connected to the process, simulating a friendly sous chef experience. The adaptability of Sous Duckling also empowers users to experiment with food confidently, fostering creativity in the kitchen. Ultimately, the platform aims to bridge the gap between technology and everyday cooking, making smart meal preparation a seamless part of modern life.

2. DISCUSSION OF ISSUES

2.1. Technical Issues

Data management involves the complex core task of robustly storing and organizing massive amounts of diverse information for recipes, ingredients, and their nutritional values. This foundational data then complicates other features, such as *ingredient substitution*, where the LLM must run to suggest alternatives without compromising the dish's taste or nutritional content. Developing effective *personalization algorithms* is also challenging, as they must accurately analyze detailed data to identify recipes that perfectly suit a user's unique tastes, dietary requirements, and current cooking skill level. All these features must be *integrated* through complex back-end work to seamlessly unify recipe selection, nutrition calculation, step-by-step preparation guides, automated shopping lists, and user feedback into a single system that scales efficiently while maintaining high performance.

2.2. Content Issues

Addressing *recipe quality* is a crucial element, ensuring that all provided recipes are accurate, rigorously tested, and complete. This foundation is further strengthened by the commitment to the *accuracy of nutritional information*, which requires precise macro- and micronutrient data for every recipe. Finally, maintaining the *clarity of instructions* is essential, as the preparation steps must be clearly articulated and designed to be adaptable for users across different cooking skill levels.

2.3. Cognitive Issues

Decision overload can occur when too many recipe options are presented, making selection difficult. Furthermore, instructions must address *skill-level variation*, providing appropriate levels of guidance for both novice and experienced cooks. In the kitchen, timing and multitasking are complex, as users often require support to

effectively coordinate simultaneous steps in more intricate recipes. Finally, *substitution understanding* is a key challenge, requiring shoppers to easily grasp both what an ingredient is being substituted with and the resulting effect on the dish's flavor and nutrition.

2.4. Social Issues

Several social factors can affect user acceptance, starting with *shared cooking environments*, where the system must account for family or friends with differing tastes or complex dietary needs. Another hurdle is building *trust in AI recommendations*, as users may be hesitant to rely on automatically generated nutrition or ingredient substitution advice, especially for health-related matters. Finally, the system must navigate diverse *cultural preferences*, since social norms around certain cuisines, ingredients, or cooking methods will significantly affect recipe acceptance and adoption.

3. APPROACH

3.1. Information Flow

Sous Duckling is a modular pipeline that: (1) captures user context; (2) generates recipes in a structured format; (3) filters and ranks candidates; (4) proposes substitutions under constraints; (5) generates step-by-step guidance; and (6) outputs nutrition and a grocery list. A shared user profile and recipe graph keep selections, swaps, and instructions in sync.

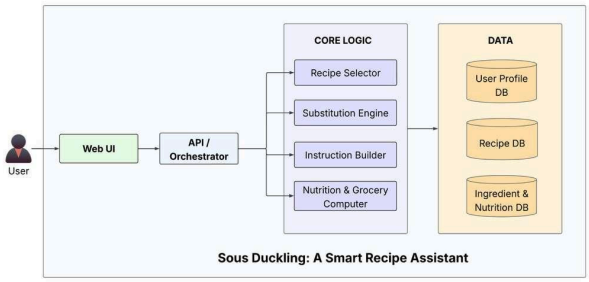

Figure 1. Information Flow

3.2. Data & Profile

Recipes are represented as graphs with nodes for ingredients, tools, and actions; edges encode order, timing, and dependencies. Ingredients carry units, allergens, diet tags, and common swaps; steps track prerequisites, duration, and parallelizability. The user profile stores diets/allergies, dislikes, skill level and updates from feedback.

3.3. Personalized Selection

A hybrid selector applies hard filters (allergens, diet) and soft preferences (cuisine, time) and scores candidates using prior likes, completion signals, and current context (pantry, time).

3.4. Substitutions under Constraints

A substitution engine proposes feasible swaps that preserve method, flavor role, and nutrition where possible, ranked by rule checks plus learned priors. Accepted swaps automatically update quantities, instructions, and nutrient totals.

3.5. Instruction Generation

From the recipe graph, the system produces a single stream that merges/splits steps, safety checks, and schedules parallel tasks. Guidance expands or simplifies based on skill.

3.6. Nutrition & Grocery

After scaling and swaps, nutrition is recomputed and targeted suggestions are offered (e.g., protein-boosting alternatives). The grocery list deduplicates across meals, groups by aisle, and respects pantry and package sizes.

3.7 Dialogue, Evaluation, Risks

We evaluate task success, effort (time-to-choice), adaptation quality, and nutrition fidelity. Key risks include coverage, over-personalization, substitution trust, and parallel-step complexity. These are mitigated with curated seeds, diversity constraints, transparent rationales, and a unified command stream.

4. PRIOR AND RELATED WORK

4.1 Recipe Generation, Retrieval and Recommendation

- Cookpad's AI assistant [1] provides users with tools to create or refine recipes, including ingredient-based suggestions. It also helps users turn photos of dishes into recipes.

- Google FoodMood [2] uses generative AI to create recipes based on cuisines that the user selects. This illustrates how auxiliary user context (beyond ingredients) can be used to steer recipe selection.

- Retrieval Augmented Recipe Generation [3] is a recent and influential work. Rather than generate recipes purely from scratch, it leverages retrieval of semantically similar recipes from a database, then uses a large multimodal model (image + retrieved recipes) to guide generation. To reduce hallucination, they propose Stochastic Diversified Retrieval Augmentation (SDRA) to bring varied contexts from multiple retrieved recipes, and a self-consistency ensemble voting scheme to pick the most coherent output.

- Majumder et al. (2019) [4] introduce the task of personalized recipe generation, where given a partial ingredient list (or dish name) and a user's

past recipe-consumption history, their model fuses technique-level and recipe-level representations via attention to generate full recipe instructions tailored to that user.

- "Flavour Fusion: AI Recipe Generator" [5] proposes a hybrid model that uses a CNN to analyze food images and classify dishes, then passes the classification to an RNN (LSTM) to generate a corresponding recipe (ingredients + steps). Their implementation focuses on a custom dataset of ~100 South Indian dishes, aiming to infer recipes from visual cues

- Lee et al. (2020) present RecipeGPT [6], a system that fine-tunes GPT-2 on a large cooking recipe corpus to support two generation modes: (1) generating cooking instructions from a recipe title plus ingredient list, and (2) generating ingredient lists given a title plus instructions. The system also features an evaluation interface: it highlights overlap between generated and user-given ingredients, retrieves similar human recipes for comparison.

- The "AI-Powered Smart Cooking Assistant" [7] system (Begum & Arif, 2025) integrates a transformer-based language model (GPT-Neo) with modules for recipe generation, dietary filtering, meal planning, and food waste tracking. It dynamically proposes recipes based on available ingredients, user dietary constraints, and preferences, and also supports nutritional assessment and sustainability features.

4.2 Ingredient Substitution and Personalization

- Fatemi et al. (2023) [8] present "Learning to Substitute Ingredients in Recipes," in which they construct a benchmark (Recipe1MSubs) of ingredient substitution pairs and propose a Graph-based Ingredient Substitution Module (GISMo). GISMo uses a graph neural network over generic ingredient relations combined with context from the specific recipe to rank plausible substitutions, and they integrate it into an image-to-recipe pipeline to support user-driven substitution.

- "Ingredient Substitution using LLMs" (2024) [9] investigates leveraging large language models to suggest context-aware replacements for ingredients. Their approach prompts the LLM with both the target recipe context and constraints (e.g. dietary restrictions or missing ingredients) to generate plausible substitutions, and they evaluate the quality and appropriateness of the replacements compared to human judgments.

- Systems like Adaptafood [10] address adaptation of meals or recipes to meet dietary constraints, adjusting portions, ingredients, or methods to respect constraints.

5. DESIGN

Sous Duckling's design centers on approachable intelligence, a conversational assistant that feels intuitive, context-aware, and quietly smart. The interface supports both discovery and execution, helping users choose what to cook, adjust to constraints, and follow adaptive step-by-step guidance.

5.1. User Flow and Screens

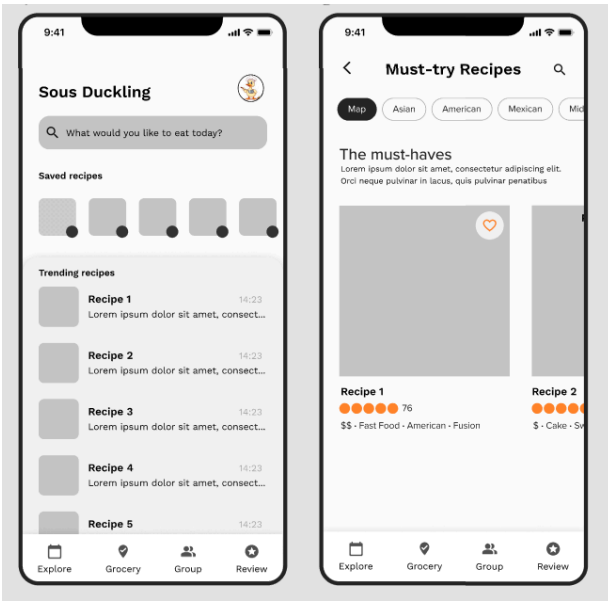

Figure 2. Initial Recipe discovery mockups

Explore & Search: The first screen invites users to express intent conversationally ("What would you like to eat today?"). It surfaces Saved and Trending Recipes personalized to the user's history and profile. The bottom navigation anchors the main modules: Explore, Grocery, Group, and Review. This reflects the modular system pipeline within a single user-friendly loop.

Recipe Discovery: This screen enables browsing by cuisine, filters, and popularity. Each recipe card visualizes key metadata.. The interaction encourages casual exploration while respecting user constraints.

Recipe Details: Once selected, the recipe view shows structured steps with dynamic elements like timers, nutrient summaries, and visual icons for ingredients. These steps are generated from the recipe graph, ensuring that every instruction aligns with the current substitutions, portions, and skill level. *Recipe Details:* Once selected, the recipe view shows structured steps with dynamic elements like timers, nutrient summaries, and visual icons for ingredients.

These steps are generated from the recipe graph, ensuring that every instruction aligns with the current substitutions, portions, and skill level.

Figure 3. Initial Recipe and chatbot mockups

Conversational Adaptation: This screen demonstrates Sous Duckling's dialogue-driven personalization. The assistant actively queries constraints ("Are you allergic to any ingredients here?") and proposes contextually valid swaps ("I am substituting them for tomatoes"). This exemplifies the substitution engine working in natural language. It also highlights transparent reasoning, every adaptation is explained conversationally.

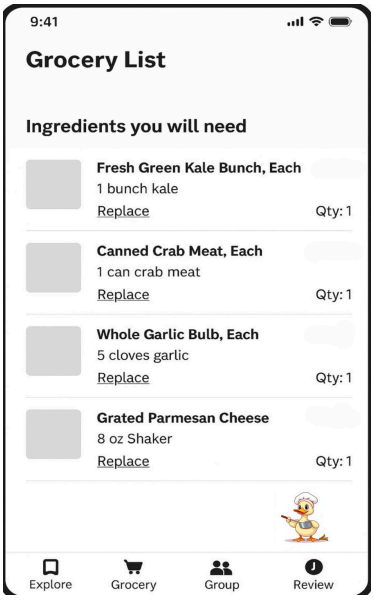

Figure 4. Initial Grocery list suggestion mockup

5.2. Interaction Model

Sous Duckling blends direct manipulation (tapping, filtering, saving) with conversational input. This hybrid model supports both structured exploration and adaptive negotiation: users can switch between browsing and chatting seamlessly. The conversation view doubles as a feedback mechanism, responses update the user profile in real time, refining future recommendations.

5.3. Design Rationale

User Modeling and Personalization: Sous Duckling adopts a hybrid personalization model, combining explicit setup with implicit learning. Users initially provide essential information: dietary restrictions, allergies, dislikes, after which the system refines its model through behavioral cues (recipe choices, substitutions, ratings). The trade-off lies in complexity and data structure design: granular personalization demands a highly structured, graph-based recipe representation capable of handling fine-grained constraints and context-dependent adaptations.

- Advantages: Achieves accurate, dynamic personalization with minimal ongoing effort. Learn progressively without constant user configuration.
- Disadvantages: Complex data modeling and real-time synchronization between user and recipe graphs. Risk of inaccurate inference from limited behavioral data.

Feedback and Rating System: The system incorporates a lightweight feedback mechanism where user interactions continuously refine recommendations. To counter slow early learning, new users can optionally rate a few benchmark dishes during onboarding, giving Sous Duckling an initial taste profile.

- Advantages: Establishes early personalization baseline with minimal input. Enables adaptive refinement through implicit behavioral feedback.
- Disadvantages: Requires significant data volume for high-confidence personalization. Overly detailed surveys may discourage new users or produce unreliable data.

Nutritional Feedback: Nutritional information is presented in a concise, essential format: calories, protein, carbs, fats rather than exhaustive micro-nutrient tables. This reflects the system's role as an advisor, not a diet tracker. Because Sous Duckling supports dynamic substitutions, its nutritional feedback engine recalculates values in real time. For instance, replacing butter with olive oil updates calorie and fat composition immediately.

- Advantages: Quick, actionable insight that maintains focus on cooking. Real-time recalculations improve accuracy and trust.

- Disadvantages: May frustrate advanced users seeking detailed health analytics. Adds computational overhead for live updates.

Autonomy vs. User Control: Sous Duckling employs a balanced autonomy model. It performs low-risk actions automatically (e.g., generating grocery lists) but notifies the user before making critical substitutions or goal-related changes. This balance enhances efficiency without eroding user agency.

- Advantages: Reduces user effort by automating routine steps. Builds transparency through contextual notifications.
- Disadvantages: Designing non-disruptive notifications is challenging. Determining the threshold for "critical" changes is subjective and complex.

Minimal cognitive load: Sous Duckling minimizes cognitive overhead by foregrounding only the most essential user actions: Chat, Saved, Profile, etc. This aligns interaction flow with user intent: ask something, revisit favorites, or adjust preferences without forcing unnecessary navigation or decision-making. The trade-off lies in interface density: reducing visible actions simplifies choices but limits immediate discoverability of secondary features.

- Advantages: Lowers decision fatigue, accelerates task flow, and supports first-time users who rely on clear, stable anchors.
- Disadvantages: Some advanced capabilities may become "hidden in plain sight," requiring additional guidance or onboarding to reveal deeper functionality.

Progressive disclosure: The system adopts progressive disclosure to avoid overwhelming new users. Non-essential features (e.g., pantry syncing, history-based recommendations, preference editing) appear only after authentication, tailoring complexity to user readiness. This design choice ensures that novice users explore core functionality first, while experienced users gain access to richer tools at their own pace.

- Advantages: Reduces UI clutter, prevents early-stage confusion, and enables feature rollout in a controlled, contextualized way.
- Disadvantages: Some users may not realize certain features exist until they create an account, potentially delaying full engagement or lowering perceived capability.

6. IMPLEMENTATION

This describes the system architecture, data flow, component responsibilities for Sous Duckling: A Smart Recipe Assistant. It translates the design and approach into an actionable architecture and implementation. It is built as a modular, service-oriented application that connects multiple backend microservices with an interactive frontend client. The architecture is designed to allow independent development and scaling of individual components such as user profile management, recipe retrieval, substitution, nutrition computation, and conversational assistance.

6.1 Architecture

6.1.1. Frontend (React-based web client built with Vite + MUI)

The frontend delivers a clean, conversational interface where users can explore, adapt, and follow recipes. The frontend supports strong accessibility features (semantic structure, ARIA roles, keyboard navigation), mobile-responsive layouts for seamless use across devices, and theme context switching to toggle effortlessly between light and dark modes.

Core UI modules include the chat interface, onboarding flow, profile settings, saved recipe library, grocery list and global navigation header. All interactions use REST endpoints configured through environment variables for flexible deployment.

6.1.2. API Layer (REST endpoints served via FastAPI + Uvicorn)

This acts as the gateway between the client and backend logic. It handles request validation, routing, and communication for recipe generation, substitutions, and preference updates. This layer ensures consistent data flow and orchestrates backend module behavior.

6.1.3. Backend Modules

The backend logic was split into multiple modules that each encapsulate a domain of application intelligence:

- *LLM Interface:* Coordinates with the language model for chat responses, personalized suggestions, and interactive cooking guidance.
- *Recipe Retrieval Module*: Filters and retrieves recipes, applying preference constraints (diet, allergens, time) and ranking rules.
- *Substitution Engine:* Generates ingredient alternatives that respect dietary constraints, recipe structure, and flavor compatibility.
- *Context Manager*: Tracks user state across interactions like active constraints, ongoing sessions, prior chats, and cooking progress.

6.1.4. Database Layer (PostgreSQL + SQLAlchemy + Alembic)

Persistent data is stored in PostgreSQL for profiles and metadata, while Neo4j or JSONB structures handle

graph-based recipe data. ML models (ranking, substitution) are stored in a model registry for versioned deployment (see Appendix B).

6.2 Data Flow and Interface

The landing page is intentionally minimal and inviting, designed to help users get started without any friction. Instead of overwhelming them with options, the interface presents a clean layout with clear navigation and a single prominent button that encourages users to begin chatting with the cooking assistant. This simplicity keeps the experience approachable, reduces cognitive load, and makes it easy for both new and returning users to jump straight into discovering or generating recipes.

Figure 5: Sous Duckling Landing Page

6.2.1. User Account Creation & Initial Preference Capture

When a new user creates an account in Sous Duckling, the system immediately begins building a personalized profile that guides all future interactions. During the onboarding flow, the user provides essential information such as allergies, dietary preferences, and their cooking skill level, which the frontend collects and sends to the backend through a REST API request. The backend then validates and normalizes these inputs, mapping allergens and dietary tags to canonical categories and converting the cooking skill selection into an internal scale to ensure consistency in downstream processing. Once validated, this information is stored in PostgreSQL across the user and preference tables, forming the foundation of the user's personalized state. At the same time, the system initializes a context object that captures the user's constraints and initial interaction state, enabling tailored filtering, recipe retrieval, substitution handling, and tone-adjusted step-by-step guidance from the LLM. From this moment on, every recommendation, substitution, and instruction generated by Sous Duckling is shaped by the preferences captured during account creation.

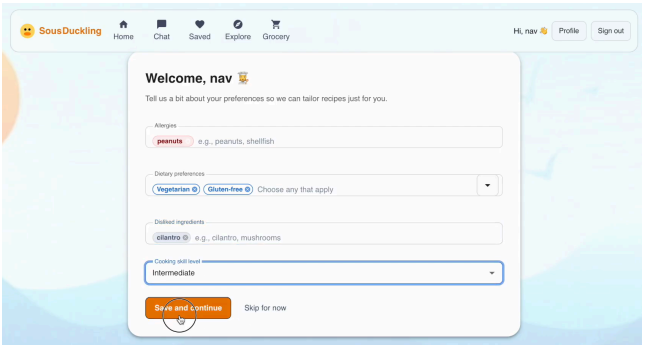

Figure 6: User Preference Capture

6.2.2. LLM-Driven Recipe Creation and Substitution

After the user completes onboarding, they enter the chat interface and request a recipe. This triggers the backend to communicate with the GPT-4o LLM through a structured recipe-generation pipeline. The system constructs a prompt that embeds the user's stored preferences and sends it to the LLM using a minimal OpenAI client wrapper. This wrapper handles authentication, model selection, fallbacks, and response formatting.

The backend then receives a JSON response containing the recipe name, a list of ingredients with quantities, step-by-step instructions, and per-serving nutritional information. The JSON is parsed, sanitized, and normalized using helper functions. Once validated, this generated recipe card becomes the basis for what the user sees in the chat interface (see Appendix C).

Importantly, the same chat window also supports interactive ingredient substitution: users can ask for alternatives, replacements, or adjustments directly in the conversation, and the backend uses a similar prompting mechanism to update the original recipe JSON while preserving schema integrity. The LLM returns a revised recipe that respects constraints and incorporates substitutions seamlessly, creating a continuous conversational loop where users can generate, refine, and personalize recipes in real time.

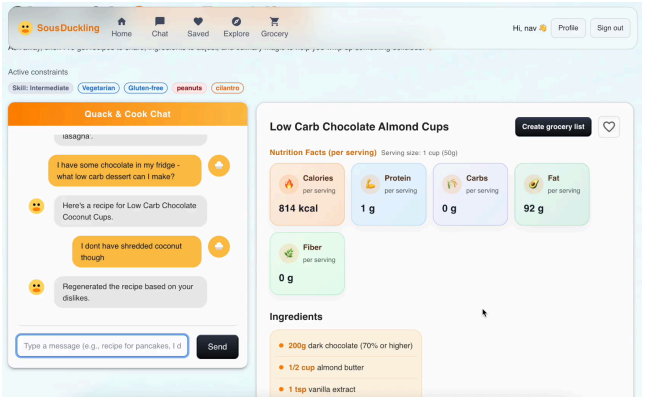

Figure 7: Chat Window

6.2.3. Creating a Personalized Recipe Library

After generating or modifying a recipe in the chat interface, the user can save it to their personal cookbook. When the user hits the like button, the structured recipe JSON containing the title, ingredients, steps, and nutritional information, along with any in-chat substitutions or adjustments is sent to the backend and stored in the recipes table under the user's ID..

In the frontend, the Saved Recipes page displays these entries as card-style previews that show essential details such as ingredient count, step count, and a short description. Users can browse, open, or delete recipes from this library at any time, giving them an evolving personal cookbook shaped by their preferences.

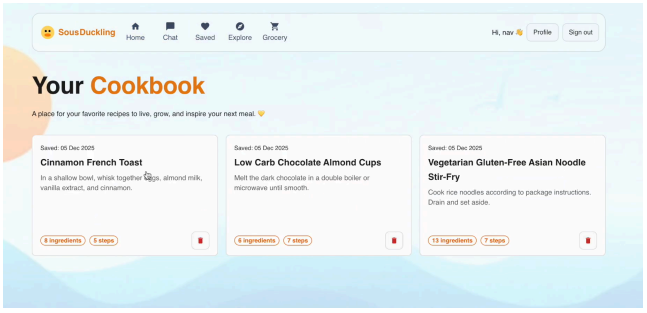

Figure 8: Saved Recipes Page

6.2.4.Grocery List Creation and Management

When viewing a generated or saved recipe, users can click the "Create Grocery List" button to automatically populate a shopping list with all the ingredients they need. The system extracts the structured ingredient data from the recipe JSON and sends it to the backend, where each item is stored in the grocery table under the user's ID.

Users can fine-tune the list by adjusting quantities, removing items they already have, or organizing ingredients by category. These changes allow the dedicated Grocery page to display a clean, grouped overview of everything the user needs for that recipe or multiple recipes combined. This interface acts as a practical, action-oriented extension of the recipe system.

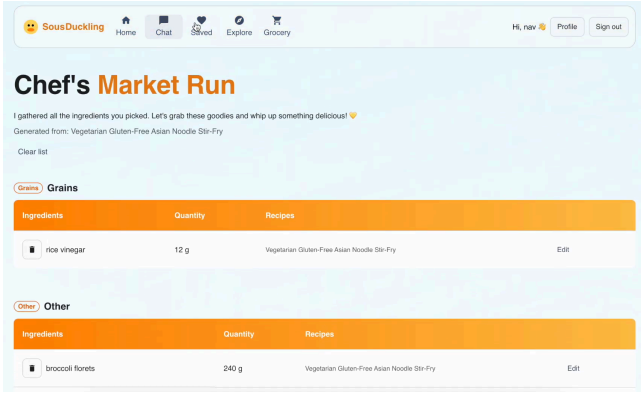

Figure 9: Grocery List Page

6.2.5. Blog-Style Exploration and Interactive Rediscovery

The Explore page functions as Sous Duckling's discovery hub, showcasing trending, highly rated, and frequently saved recipes in a blog-style feed. Users can browse through curated recipe cards, read summaries, and rate the recipes they try, allowing the system to surface popular or seasonally relevant dishes.

When a user clicks on any recipe from the Explore page, they are taken back into the chat interface where the full interactive experience resumes complete with ingredient substitutions, preference-aware modifications, and adaptive step-by-step guidance. This creates a seamless loop between browsing, selecting, and personalizing recipes, ensuring that even casually discovered dishes can be tailored to the user's tastes and dietary needs.

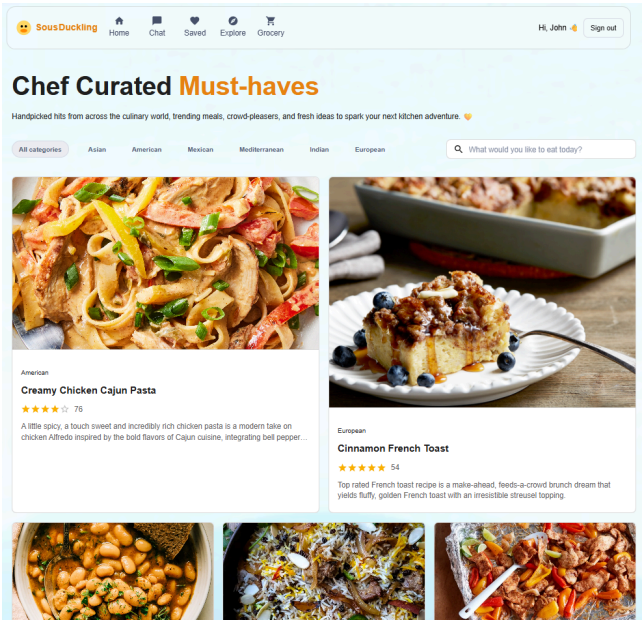

Figure 10: Explore Page

7. EVALUATION

To evaluate the Sous Duckling prototype, we will conduct a mixed-method usability study combining a controlled task-based experiment with lightweight real-world usage. The goal is to assess how effectively the system personalizes recipes, supports interactive modifications, and assists with meal planning.

7.1. Participants

We will recruit 10–12 participants who cook at least occasionally and represent a mix of cooking skill levels (beginner to advanced). Participants will be selected from the student population and should be comfortable using web applications. This range ensures varied recipe needs, dietary restrictions, and interaction styles.

7.2. Procedure

Participants will complete a structured series of tasks using the live prototype:

1. Create an account and enter onboarding preferences (diet, allergies, skill).
2. Request a recipe through the chat interface.
3. Apply modifications using the substitution and preference prompts.
4. Save the final recipe to their cookbook.
5. Generate a grocery list from the recipe and adjust quantities.
6. Explore the Trending/Explore page and open a recipe from the feed.

After the task-based session, participants may use the system freely for a short exploratory period to provide additional impressions.

7.3. Data Collected

We will collect both quantitative and qualitative data:

- Interaction logs: number of prompts, modifications, errors, and time spent on each task.
- Preference adherence metrics: whether the generated recipe respected allergies/dietary restrictions.
- Task completion success and time.
- Usability surveys
- Post-task interviews: perceptions of clarity, trust, usefulness, and satisfaction.

7.4. Data Analysis

Quantitative data (task times, success rates) will be analyzed using descriptive statistics to identify usability strengths and bottlenecks. Logs will help quantify how often substitutions are used and whether the LLM maintains user constraints. Qualitative interview data will be coded thematically to uncover common patterns in user expectations, confusion points, trust in LLM outputs, and perceived helpfulness of features like saved recipes and grocery lists. Together, these analyses will determine whether the prototype supports intuitive interaction, reduces user effort, and provides reliable personalized recipe guidance.

8. CONCLUSIONS AND FUTURE WORK

We set out to design an intelligent, personalized cooking assistant that reduces the cognitive load of finding and adapting recipes. Through Sous Duckling, we successfully built a working prototype that integrates user preference capture, LLM-driven recipe generation, structured recipe modification, saved recipe management, and a functional grocery list workflow. The system provides an end-to-end experience, from account creation to meal planning, while ensuring recipes respect dietary needs, allergies, and cooking skill levels. In doing so, we demonstrated how conversational interfaces and structured LLM prompting can meaningfully support everyday cooking tasks.

Throughout the project, we learned how essential clear prompt design and structured JSON outputs are for creating reliable LLM-driven interactions. We also discovered the importance of a clean, uncluttered UI in guiding users through a multi-step cooking workflow without overwhelming them. The process revealed how user preferences must be normalized and embedded deeply in the system for consistent personalization. Additionally, building and testing the prototype taught us how intertwined backend logic, frontend usability, and model reasoning are: each must reinforce the others for the experience to feel smooth and trustworthy.

With more time, we would deepen personalization by building taste profiles, tracking pantry items, and generating frequency-based recipe recommendations. We would expand the substitution engine using ingredient embeddings or knowledge graphs to enable richer, context-aware swaps. Nutrition features could be extended to include detailed macro/micronutrient analysis and goal-oriented recipe variants. On the interface side, we would polish mobile accessibility, introduce voice-based interaction, and add analytics dashboards to monitor user satisfaction and common constraints. These enhancements would transform Sous Duckling into a more intelligent, adaptive, and holistic cooking companion.

9. WORK PLAN

Research and Requirement Gathering: Collecting information on existing apps, datasets etc. Finalizing core features, success criteria and the tech stack.
Responsibility: Team

Model Design & Training: Building and training models for Recipe personalization and Ingredient substitution. Evaluating with accuracy, coverage, and user feedback simulations.
Responsible: Navya

User Input & Profile Design: Designing how we capture preferences: diets, allergies, dislikes, cooking skill, time, budget, pantry items, goals. Storing a simple user profile that updates with feedback.
Responsibility: Ankitha

Final Recipe Presentation & Nutrition Analysis: Generating final recipe steps, simplified for the user's skill level. Adding alternative instructions when needed. Computing and presenting nutrition values for the final recipe.
Responsibility: Navya

Grocery List Creation: Generating a clean, categorized shopping list from saved recipes.
Responsibility: Nikki

User Interface Design: Designing an interactive, intuitive UI.
Responsibility: Afreen

System Integration & Testing: Combining models and modules into one smooth workflow. Conducting user testing to refine usability and build trust.
Responsibility: Khushi

Final Documentation & Report: Preparing project documentation, final report and presentation.
Responsibility: Team

10. ACKNOWLEDGEMENTS

We would like to sincerely thank Dr. Frank Shipman for his guidance and for teaching the principles of Intelligent User Interfaces that directly shaped our design decisions throughout this project. His insights into user interaction, user-centered design, and interface evaluation greatly influenced how we approached the architecture and refinement of Sous Duckling.

11. REFERENCES

[1] https://blog.cookpad.com/in/cookpad-ai-assistant/

[2] https://artsandculture.google.com/experiment/food-mood/HwHnGalZ3up0EA?hl=en

[3] Liu, G., Yin, H., Zhu, B., Chen, J., Ngo, C. W., & Jiang, Y. G. (2025, February). Retrieval augmented recipe generation. In 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (pp. 2453-2463). IEEE.

[4] Majumder, B. P., Li, S., Ni, J., & McAuley, J. (2019). Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105*.

[5] R, Dharani & P, Harishini & T, Banupriya & A, Aseera & M, Kalaiselvi. (2025). Flavour Fusion: AI Recipe Generator. International Research Journal on Advanced Engineering Hub (IRJAEH). 3. 1841-1845. 10.47392/IRJAEH.2025.0266.

[6] H. Lee, H., Shu, K., Achananuparp, P., Prasetyo, P. K., Liu, Y., Lim, E. P., & Varshney, L. R. (2020, April). RecipeGPT: Generative pre-training based cooking recipe generation and evaluation system. In Companion Proceedings of the Web Conference 2020 (pp. 181-184).

[7] Fatemi, B., Duval, Q., Girdhar, R., Drozdzal, M., & Romero-Soriano, A. (2023). Learning to substitute ingredients in recipes. arXiv preprint arXiv:2302.07960.

[8] Senath, T., Athukorala, K., Costa, R., Ranathunga, S., & Kaur, R. (2025). Large language models for ingredient substitution in food recipes using supervised fine-tuning and direct preference optimization. Natural Language Processing Journal, 100177.

[9] Morales-Garzón, A., Gutiérrez-Batista, K., & Martin-Bautista, M. J. (2025). Adaptafood: an intelligent system to adapt recipes to specialised diets and healthy lifestyles. Multimedia Systems, 31(2), 87.

[10] Begum, G. F., & Arif, M. U. (2025). AI-Powered Smart Cooking Assistant

APPENDIX A - Supplemental Materials

- Demo Video: https://youtu.be/rwydslJS898
- GitHub Repo: https://github.com/Naviunni/sous-duckling-recipe-assistant/tree/main

APPENDIX B - Database Schema

This appendix summarizes the core database tables used in Sous Duckling. The schema supports user personalization, recipe storage, and grocery list management.

B1. Users Table Schema

Column Name	Data Type	Notes
user_id	UUID or SERIAL	Primary Key
email	VARCHAR(255)	Unique, for login
password_hash	VARCHAR(255)	For storing the hashed password
created_at	TIMESTAMP	Default NOW()

Figure B1. *Schema for the Users table, which stores authentication details.*

B2. User Profiles Table Schema

Column Name	Data Type	Notes
profile_id	UUID or SERIAL	Primary Key
user_id	UUID or INTEGER	Foreign Key, links to users.user_id
allergies	TEXT[]	e.g., {'peanuts', 'diary'}
dietary_restrictions	TEXT[]	e.g., {'vegan', 'gluten-free'}
disliked_ingredients	TEXT[]	e.g., {'cilantro'}
skill_level	VARCHAR(50)	beginner/intermediate/advanced

Figure B2. *Schema for the Users Profiles table, which stores user-specific preferences such as allergies, dietary constraints, and cooking skill level.*

B3. Saved Recipes Table Schema

Column Name	Data Type	Notes
recipe_id	SERIAL	Primary Key
user_id	UUID or INTEGER	Foreign Key, links to users.user_id
recipe_title	VARCHAR(255)	Recipe title
recipe_data	JSONB	Stores the entire GPT recipe object.
saved_at	TIMESTAMP	Default NOW()

Figure B3. *Schema for the Saved Recipes table, which stores recipe JSON objects generated or modified by the LLM, along with metadata*

B4. Grocery List Table Schema

Column Name	Data Type	Notes
grocery_id	SERIAL	Primary Key
user_id	UUID or INTEGER	Foreign Key, links to users.user_id
recipe_id	INTEGER	Foreign Key, links to recipes.recipe_id
ingredient_name	VARCHAR(255)	Amount needed (e.g., “2 cups”)
quantity	VARCHAR(100)	Default NOW()

Figure B4. *Schema for the Grocery List table, which stores ingredient items selected or adjusted by the user.*

APPENDIX C - LLM & Prompt Templates

This appendix summarizes how Sous Duckling communicates with the GPT-4o model for recipe generation, recipe modification, and interactive substitution. Only the essential mechanisms and prompt structures are included.

A1. LLM Interface Overview

Sous Duckling uses a minimal OpenAI client wrapper that:

- Loads API credentials from environment variables
- Initializes an OpenAI client only when needed
- Sends structured system + user messages
- Requests strict JSON output using response_format={"type": "json_object"}
- Performs safe JSON parsing with fallbacks

This ensures predictable, schema-consistent interactions and robust behavior.

A2. Recipe Generation Prompt Template

Used when the user requests a new recipe through chat. The system enforces a strict output schema.

```
System:
"You are a helpful cooking assistant. Generate concise, home-cook friendly recipes."

User:
"Generate a complete recipe as compact JSON only. Schema:
{
  'name': string,
  'ingredients': [ { 'name': string, 'quantity': string } ],
  'steps': [ string ],
  'nutrition': { 'calories': string, 'protein': string, 'carbs': string, 'fat': strin
  'serving_size': string
}
Avoid markdown and extra commentary.

Recipe: <RECIPE_NAME>.
Exclude or replace these if possible: <DISLIKES>.
Dietary restrictions to respect: <DIETARY>.
Adjust complexity for a <SKILL_LEVEL> home cook."
```

The backend includes the last few turns of conversation when relevant, maintaining continuity.

A3. Recipe Modification & Substitution Prompt

Used when the user asks for changes such as substituting ingredients, adapting instructions, or adjusting difficulty.

```
System:
"You are a helpful cooking assistant. Modify the given recipe JSON to satisfy user const

User:
"Modify the following recipe JSON to avoid dislikes and apply substitutions,
keeping the same JSON schema including nutrition and serving_size.

Dislikes: <DISLIKES>
Dietary restrictions: <DIETARY>
Adjust complexity for a <SKILL_LEVEL> home cook.
Substitutions: <INGREDIENT_A -> INGREDIENT_B>

Recipe JSON:
<BASE_RECIPE_JSON>"
```

The model returns a full updated recipe, preserving the schema.

A4. Chat-Based JSON Responses

For additional tasks (e.g., classification, ingredient checks), the backend can send arbitrary message lists. This ensures the LLM always returns machine-readable JSON, enabling downstream processing.