Project Design

Name: Chia-Yu Wie zID: z5458951

Scope

The recommender operates entirely within the ingredients and recipe domain, designed for international students and home cooks who frequently find themselves with partially used ingredients languishing in the fridge. In the mobile app, a user begins by entering the exact items and quantities they have on hand. The system generates up to 100 candidate recipes in seconds, but it only shows 10-20 cards to users at a time. This is providing enough variety to inspire without overwhelming.

The interface is intentionally designed for mobile: large, tappable buttons and clear dish images make it effortless to browse recipes even while juggling pots and pans. The system starts with the Ingredient Input screen, where users can add and remove items from an interactive list. To ensure data integrity, the input field enforces a controlled vocabulary of ingredient keywords, preventing variant names from conflicting with recipes in the database. When ingredients are entered, the Recommendation List screen displays the best recipe matches.

At first launch—before any personal data exists—the system solves the cold-start problem by displaying a balanced sampler of desserts, mains, soups, and more; the user's initial category selections then seed their preference profile.

As the user base grows, monetization is planned through affiliate grocery partnerships, sponsored recipe placements, and an optional premium tier that includes advanced dietary filters and meal planning. In this way, every saved ingredient becomes not only a delicious meal, but also a long-term business opportunity.

Dataset

The project selects the Food.com Recipes and Reviews dataset from Kaggle (irkaal/foodcom-recipes-and-reviews). This collection comprises over 522,000 unique recipes across 312 categories and 1.4 million user reviews from nearly 272,000 individuals. Each recipe entry includes an identifier, title, comprehensive ingredients list, nutritional information, and step-by-step instructions, while each review record provides a user ID, recipe ID, star rating (1–5), review text, and timestamp.

MovieLens 100K, on the other hand, is much more limited: it contains only 100,000 star ratings from 943 users on 1,682 movies, with each interaction reduced to a single numeric score and basic demographic fields, with no accompanying text or rich item attributes. Collected over only seven months in 1997-1998 and stored as a static

snapshot, it lacks the volume, descriptive metadata, and temporal depth required to support large-scale, content-driven, or continuously retrained recommender systems.

The project's recommendation is based on several core fields from the Food.com dataset. First, each recipe's RecipeIngredientParts and RecipeIngredientQuantities are parsed into a bag-of-ingredients vector and combined with TF-IDF embeddings derived from Description, Keywords, and RecipeCategory. This rich feature representation makes it possible to take a user's fridge inventory—also converted into the same ingredient vector space—and compute cosine similarity to find the closest recipe matches.
At the same time, the app displays the total cooking time—combining CookTime, PrepTime, and TotalTime—so users can immediately see how long each recipe takes and choose one that fits their schedule.

Next, those filtered recipes are ranked based on what the community enjoys—using their average star rating and total number of reviews. Meanwhile, the system remembers which dishes a user has cooked and rated highly—storing RecipeId, ingredient lists, calorie counts, and protein levels—and when new ingredients are entered, it refers to those previous favourites to recommend similar recipes. By combining content matching, rule-based filtering, popularity priors, and personalized history, all sourced from the dataset's rich schema, the recommender delivers relevant, reliable meal ideas.

Even this rich Food.com dataset has its blind spots.
It's been cleaned and flattened for CSV export, so quirky ingredient notes like "a pinch of salt" or unconventional units get dropped—handy for parsing but stripping away the messy details cooks actually write. Because it's hosted on Kaggle, the data comes with fixed train/test splits and star-rating objectives that encourage tuning models for static accuracy, rather than adapting to new recipes or evolving user tastes. Finally, ingredient amounts are stored as abstract proportions (like "1" or "1/4") instead of grams or cups. That makes matching tiny leftovers easier in theory, but can leave users guessing exactly how much flour "1" actually means. Acknowledging these gaps will guide us to collect a bit of extra user input—like actual leftover weights on top of ratings so our recommendations stay practical and precise.

https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews

Methods

This approach uses three simple, rating-based methods that fit the data on hand and can be built quickly.

First, the Popularity Baseline leans on each recipe's average star rating and review count, so brand-new users immediately get well-loved, crowd-approved dishes.

Next, the system personalizes recommendations by identifying which ingredients appear most often in a user's previously rated 4- or 5-star recipes and boosting new recipes that share those ingredients—each candidate also receiving a small extra weight from its own community rating.

Finally, the Hybrid Content-Rating Fusion method vectorizes ingredient lists via TF-IDF, measures cosine similarity against the user's liked-recipe profile, and blends that similarity score with the recipe's normalized average rating—calibrating the mix on held-out data to balance personal fit and social proof. To handle cold start, the system recommends 1–2 popular recipes from each major category (desserts, mains, soups, etc.) in the first three rounds; from rounds four to six, it pivots to the Ingredient Overlap and Hybrid methods, gradually shifting toward fully personalized suggestions as more feedback becomes available.

Evaluation

While RMSE and MSE on a held-out test set tell the team exactly how far predicted star ratings stray from users' actual scores—lower errors meaning the model is closer to real preferences—the offline evaluation doesn't paint the whole picture. Precision@N, Recall@N, and NDCG@N then measure ranking quality: Precision@N shows what fraction of the top recommendations truly match users' tastes, Recall@N reveals how many of all relevant recipes make it into those top slots, and NDCG@N rewards models that push the very best dishes to the top by weighting hits near the front of the list. Finally, by boiling ratings down into a binary like/dislike label (for example, treating 4- and 5-star as "liked"), an F1-score balances false positives and false negatives—ensuring the model not only ranks well but also classifies user satisfaction reliably.

While offline metrics ensure that the model is technically sound, real-world feedback is needed to confirm its practical impact. To measure genuine user delight, the system tracks the average star rating that users assign to recommended recipes and the share of suggestions that even receive a rating—direct indicators of how well those recommendations resonate, even if they can shift with UI tweaks or test-group makeup. Armed with this knowledge, weaker models are first weeded out using offline metrics, and then the top contenders are pitted against each other in A/B tests comparing their actual rating performance to see which approach truly delights home cooks.

Once candidate models pass both offline and online screening, finer-grained ranking measures come into play. Precision@10 and Recall@10 quantify what fraction of the top ten recommendations each user actually enjoys, but to avoid a situation where a handful of power users dominate the numbers, Precision@10 is first calculated per user and then averaged across the entire audience. When it's time to pick a winner, a minimum bar—say, 25% average Precision@10—is enforced, and among those that

clear this hurdle, the model with the highest NDCG@10 is selected, striking the right balance between relevance and ranking quality.

To support all of the above evaluation in production, the system must also meet strict latency and update demands. Ingredient–recipe similarity vectors are precomputed and indexed with an approximate nearest-neighbor library (e.g. FAISS or Annoy) so that fetching the top matches happens in just a few milliseconds. Immediately after a user submits ratings for the current recommendation round, the system incrementally updates that user's profile vector and reruns the ANN query to produce fresh suggestions in real time. Performance benchmarks show that a single-node FAISS deployment can handle thousands of recommendation requests per second, and that updating a user's profile and fetching their next list of matches takes only a few hundred milliseconds, ensuring both speed and responsiveness.

The user study will enlist 3–5 participants drawn from international students and home cooks who regularly prepare meals. Each person will go through six rounds of interaction: in rounds 1–3, the system will serve recipes based solely on overall popularity, and in rounds 4–6, it will switch to the personalized Hybrid Content-Rating Fusion algorithm. After reviewing each set of ten recipe suggestions, participants will rate each dish on a scale of 1 to 5, providing direct feedback on how much they enjoyed the recommendations. By comparing average ratings from the popularity-only phase to those from the personalised phase, the team can conduct an A/B-style analysis—albeit among the same users—to determine whether the hybrid method actually improves user satisfaction over a simple top-rated feed.

Beyond quantitative feedback in the user study, subjective impressions will round out the evaluation. Following the sessions, each participant completes a short questionnaire. They begin by assigning an overall satisfaction score of 1 to 5, indicating how pleased they were with the recommended recipes as a whole. They then rate each suggestion on a 1-to-5 scale based on how well it matched the ingredients they had on hand. Combining these two scores reveals not only whether users liked the recommendations, but also whether the algorithm truly understood their pantry.