# Personalized Recipe Discovery: Elevating User Engagement and Retention

## A Smart Solution for Grocery List and Meal Planning Apps

Babak Hosseini

Github: https://github.com/bab-git

# Business Question

How can grocery list and meal planning apps significantly increase user engagement, retention, and monetization potential by leveraging existing user data to provide personalized recipe recommendations?

# Why It's Important to Users?

- **Time-saving**: Less time spent searching for recipes
- **Personalization**: Tailored recommendations based on past behavior
- **Inspiration**: Combat 'meal fatigue' with new ideas
- **Convenience**: Seamless integration with grocery lists
- **Health alignment**: Suggestions tailored to dietary nee

# Proposed Solution: Smart Recipe Recommendation System

**Proposed Solution:**

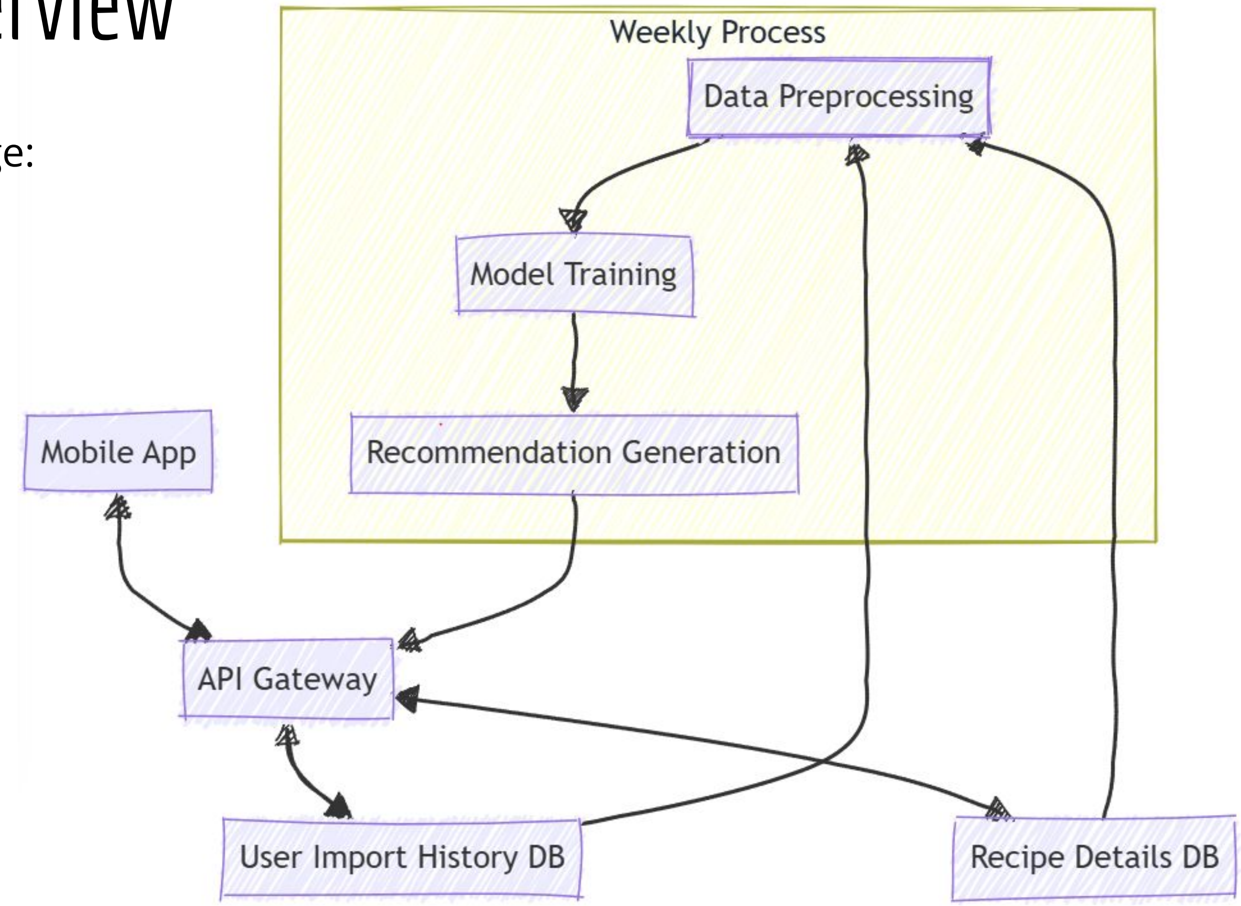
- Seamless recipe import from popular websites (e.g., yummly.com, epicurious.com)
- Advanced recommendation engine for highly relevant suggestions
- Weekly personalized recipe updates based on user selection history
- Powered by big data technologies for scalable, real-time processing
- End-to-end deployment: From data collection to user interface integration

**Key Benefits:**

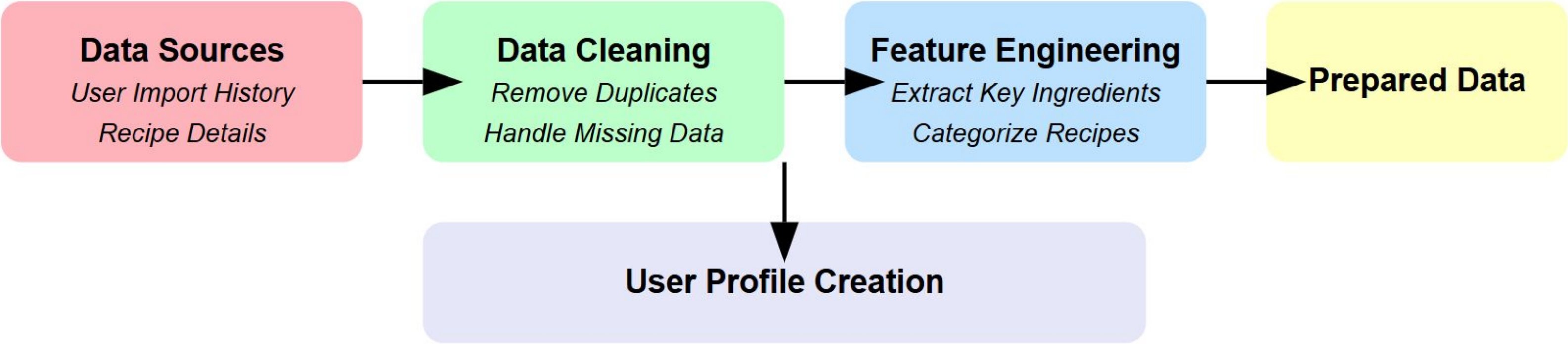
★ Enhanced user engagement
★ Personalized meal planning experience
★ Streamlined grocery shopping process

# System Design Overview

- Data Collection and Storage:
- Data Preprocessing
- Model Architecture
- API Gateway
- Mobile App
- Weekly Recommendation Process



# Data Preparation




**Data Schema**

*User Import History:*

- *user_id: string*
- *recipe_url: string*
- *import_timestamp: datetime*

*Recipe Details:*

- *recipe_id: string*
- *ingredients: list[string]*
- *cuisine_type: string (if available)*
- *meal_type: string (if available)*

*User Profile:*

- *user_id: string*
- *preferred_ingredients: list[string]*
- *preferred_cuisine_types: list[string]*
- *import_frequency: float*

# Feature Extraction:

**Feature vectors:**

- Ingredients, cuisine type, meal type, other attributes

**A - Concatenation of feature vectors**

- R = [w_i * I, w_c * C, w_m * M, w_o * O]
- More interpretable
- Need to infer missing attributes
- Higher dimensionality

**B - Addition of feature vectors**

- R = [w_i * I + w_c * C + w_m * M + w_o * O]  using word embeddings or TF-IDF
- Easier to handle missing attributes
- Less interpretable
- Lower dimensionality

# Model Research and Selection

**Candidate Models:**

1. Collaborative Filtering (CF)
   a. Matrix Factorization techniques (e.g., Alternating Least Squares)
   b. Pros: Captures user behavior patterns and hidden preferences
   c. Cons: Cold start problem for new recipes

2. Content-Based Filtering
   a. TF-IDF or Word2Vec for ingredient similarity
   b. Pros: Works well with recipe attributes, no cold start problem
   c. Cons: May miss serendipitous recommendations

3. Hybrid Approach
   a. Combining CF and Content-Based methods
   b. Pros: Leverages strengths of both approaches
   c. Cons: More complex to implement and tune

# Model Research and Selection

**Evaluation Metrics:**

1. **Offline Metrics**
   a. MAP (Mean Average Precision)
   b. Precision@K (e.g., Precision at top 5 recommendations)
   c. [if order is important]: nDCG (Normalized Discounted Cumulative Gain)
   d. Explain: These metrics help assess ranking quality without user interaction

2. **Online Metrics**
   a. Click-Through Rate (CTR)
   b. Save Rate (how often users save recommended recipes)
   c. Recipe View Time
   d. Explain: These metrics measure actual user engagement with recommendations

# Model Research and Selection

**Model Selection:**

1. **Cross validation:**
    a. Parameter selection
    b. Offline without real test
2. **A/B test:**
    a. Model structure selection
    b. Top-performing models from cross validation
    c. Real test on users, splitting them into test groups

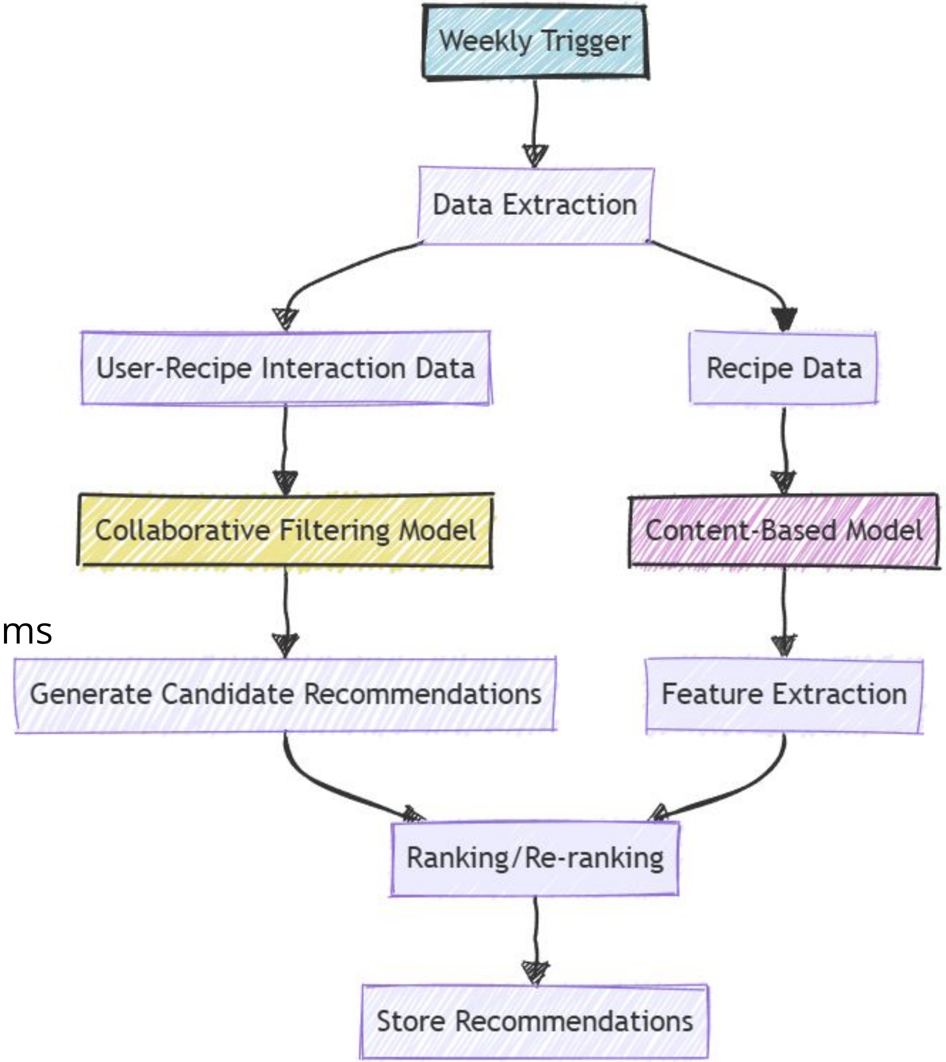
# Hybrid Algorithm

**Steps:**

- CF: Set of 100 candidates for each user
- Content-based: Re-rank them to top 10

**Why?**

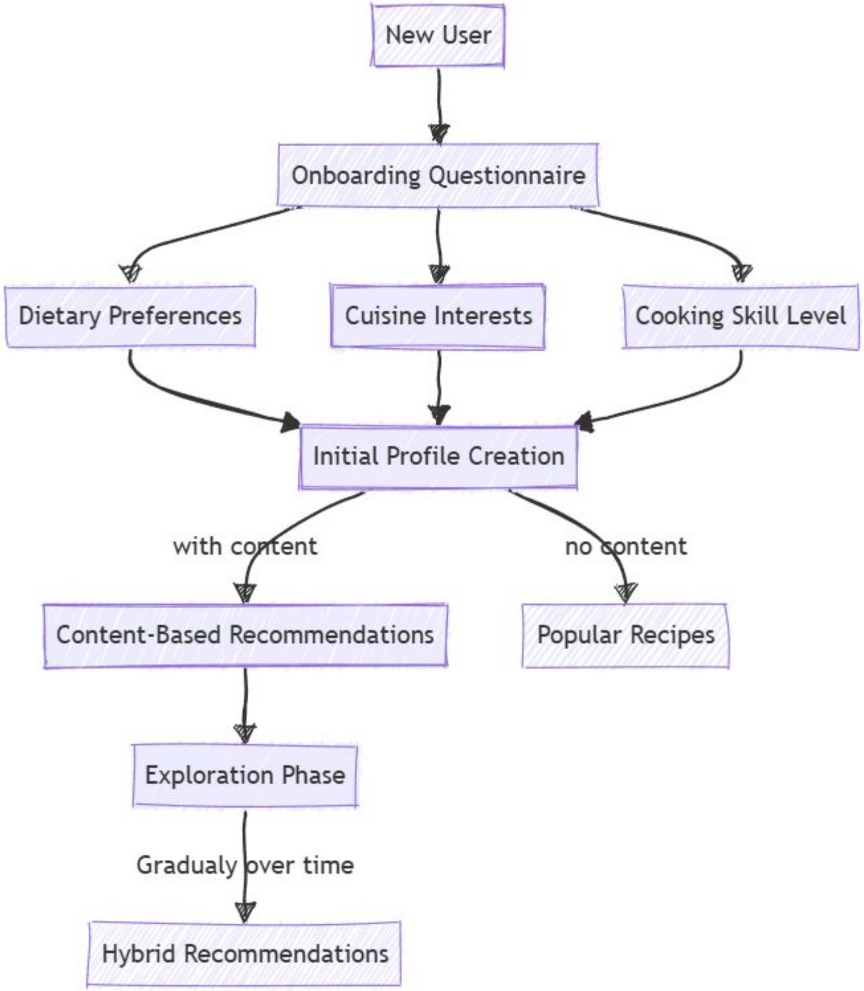
- Complementary Strengths
- Handling Data Sparsity and Cold Start Problems
- Scalability and Computational Efficiency



# New users - cold start

- Initial profile questionnaire
- Content-based relevant suggestions
- Or just popular suggestions

- New users will switch to hybrid over time



# Technical Implementation

**Data Pipeline and Processing:**

- ETL (Extract, Transform, Load) process:
  - Extracts new user-recipe interactions and recipe data
  - Transforms data (cleaning, feature engineering)
  - Loads into a Data Lake (e.g., Amazon S3, Azure Data Lake)
- Implements data versioning for reproducibility
- Runs on a weekly schedule (e.g., using Apache Airflow)

# Technical Implementation

**Model Training:**

- Uses Apache Spark for distributed processing:
    - Reads data from Data Lake
    - Performs feature engineering at scale
    - Trains recommendation models (CF, Content-Based, Hybrid)
- Hyperparameter tuning using cross-validation
- Logs model performance metrics

**Model Registry and Serving:**

- Stores trained models in a Model Registry (e.g., MLflow)
- Versioning of models for easy rollback
- A/B testing framework for comparing model versions
- Model serving layer (e.g., TensorFlow Serving, MLflow)
    - Loads latest approved model version
    - Generates **batch predictions** for all users

# Technical Implementation

**Deployment and API Integration:**

- RESTful API endpoints:
  - GET /recommendations/{user_id}: Retrieve user's recommendations
  - POST /feedback: Collect user feedback on recommendations
  - FastAPI: performance, python package, auto documentation,
- API Gateway for request routing and throttling
- Caching layer (e.g., Redis) for frequently accessed recommendations
- Use of containerization (Docker) and orchestration (Kubernetes)

# Mobile App Integration

**User interface:**

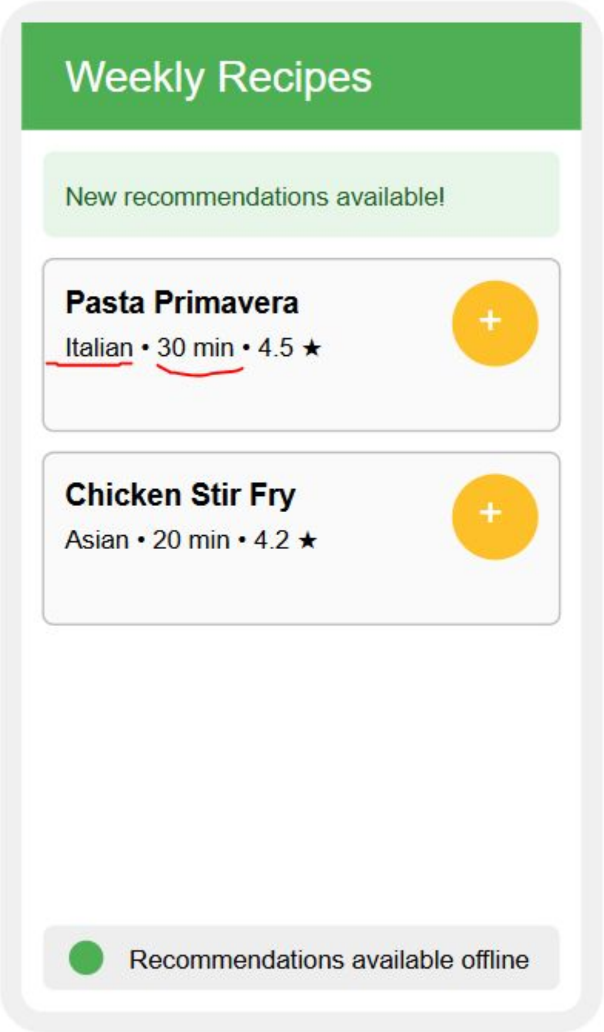
- Weekly notification for new recommendations
- Scrollable list of recommended recipes with key details

**Caching and offline access:**

- Store recommendations locally for offline viewing
- Sync mechanism for updates

**Feedback mechanism:**

- Allow users to rate or save recommended recipes
- Collect implicit feedback (e.g., recipe views, ingredient adds to shopping list)



# Future Improvements and Conclusion

**Potential enhancements:**

- Incorporating seasonality and local food trends
- Personalized recommendation timing
- Integration with shopping list for ingredient-based recommendations
- AI-powered meal planning based on nutritional goals

**Key takeaways:**

- Leverage user import data for highly personalized recommendations
- Scalable, weekly batch process ensures efficient resource use
- Continuous improvement through user feedback and A/B testing
- Increased user engagement and retention through personalized content
- New monetization opportunities through strategic partnerships