Product Requirements Document: Recipe Social App

1. Executive Summary

1.1 Product Overview

A mobile application that combines AI-powered recipe generation with social media features, allowing users to create, share, discover, and rate recipes while building a community of home cooks and food enthusiasts.

1.2 Target Audience

Anyone interested in cooking, meal planning, or sharing culinary creations—from beginners to experienced home cooks.

1.3 Key Value Propositions

- AI-powered recipe generation with conversational refinement
- Social discovery and community engagement around recipes
- Personalized recipe recommendations based on user preferences and available ingredients
- Gamification through chef scoring system
- Comprehensive recipe management and organization

---

2. Core Features

2.1 User Authentication & Onboarding

Priority: P0 (Must Have)

Requirements:

- Email/password registration and login
- OAuth integration (Google, Apple Sign-In)
- During onboarding, collect:
  - Dietary restrictions (vegetarian, vegan, gluten-free, dairy-free, nut allergies, etc.)
  - Cooking skill level (beginner, intermediate, advanced)
  - Cuisine preferences (optional)
- Profile setup: username, profile picture, bio
- Account settings management

User Stories:

- As a new user, I want to create an account so I can save and share recipes
- As a user with dietary restrictions, I want to specify them during signup so recipes are tailored to my needs

---

2.2 AI Recipe Generation

Priority: P0 (Must Have)

Requirements:

- Conversational chatbot interface for recipe creation
- User inputs for recipe generation:
  - Available ingredients (optional, from pantry)
  - Dietary restrictions (auto-populated from profile)
  - Cuisine type
  - Cooking time constraints
  - Skill level
  - Number of servings
  - Meal type (breakfast, lunch, dinner, snack, dessert)
- Iterative refinement: users can modify recipes through continued chat
- Chat history saved per recipe creation session (visible only to creator)
- Recipe modifications overwrite the current version
- Final recipe can be saved to user's profile

User Stories:

- As a user, I want to chat with an AI to generate a recipe based on ingredients I have
- As a user, I want to refine a generated recipe by asking for modifications
- As a user, I want to review my chat history for a recipe I'm creating

---

2.3 Recipe Management

Priority: P0 (Must Have)

Requirements:

Recipe Object Structure:

- Title
- Description/Summary
- Ingredients list (with quantities)
- Step-by-step instructions
- Prep time
- Cook time
- Total time
- Servings
- Difficulty level (easy, medium, hard)
- Cuisine type
- Meal type (breakfast, lunch, dinner, snack, dessert)
- Recipe images (multiple photos)
- Nutritional information:
  - Calories per serving
  - Macronutrients (protein, carbs, fats)
  - Micronutrients (vitamins, minerals - key ones)
  - Serving size details
- Creator information
- Rating (average of 1-5 stars)
- Favorite count
- Tags/keywords
- Source (AI-generated, photo upload, link import, manual entry)
- Privacy setting (public/private)

- Creation/modification timestamps

Recipe Input Methods:

1. AI Generation (via chatbot)
2. Photo Upload with OCR/AI extraction
3. URL import with web scraping
4. Manual entry

Recipe Actions:

- Create, edit, delete (own recipes only)
- Save to favorites
- Rate (1-5 stars)
- Comment
- Share (external sharing)
- Remix (create your own version with attribution)
- Add to collections

User Stories:

- As a user, I want to take a photo of a recipe from a cookbook and have it digitized
- As a user, I want to paste a URL and import a recipe from a website
- As a user, I want to organize my recipes into custom collections

---

2.4 Pantry & Ingredients Management

Priority: P1 (Should Have)

Requirements:

- Virtual pantry where users maintain their ingredient inventory
- Add/remove ingredients manually
- Check off ingredients as used
- Ingredient categories (produce, dairy, proteins, grains, spices, etc.)
- Integration with recipe generation (suggest recipes based on pantry items)
- Low stock notifications (optional)

User Stories:

- As a user, I want to maintain a list of ingredients I have so I can find recipes I can make
- As a user, I want to check off ingredients as I use them

2.5 Shopping List

Priority: P1 (Should Have)

Requirements:

- Auto-generate shopping list from recipe ingredients
- Manual addition/removal of items
- Check off items while shopping

- Organize by category/aisle
- Multiple shopping lists support
- Share shopping lists

User Stories:

- As a user, I want to generate a shopping list from a recipe so I know what to buy
- As a user, I want to share my shopping list with family members

---

2.6 Recipe Discovery & Feed

Priority: P0 (Must Have)

Requirements:

Personalized Feed Algorithm:

- Recipes from followed users
- Recommended recipes based on:
  - User's pantry ingredients
  - Time of day (auto-suggest breakfast/lunch/dinner)
  - Similar to favorited recipes
  - Dietary restrictions
  - Cuisine preferences
  - Trending/popular recipes
- Filter options:
  - Meal type (breakfast, lunch, dinner, snack, dessert)
  - Cuisine type
  - Cooking time
  - Difficulty level
  - Dietary filters
- Sort options:
  - Most recent
  - Highest rated
  - Most favorited
  - Chef score

Browse Random Recipes:

- "Surprise Me" feature for recipe discovery
- Weighted by relevance to user preferences

User Stories:

- As a user, I want to see recipe suggestions for breakfast when I open the app in the morning
- As a user, I want to discover recipes I can make with ingredients I already have
- As a user, I want to browse random recipes for inspiration

2.7 Search & Discovery

Priority: P0 (Must Have)

Requirements:

Search Capabilities:

- Global search bar
- Search by:
  - Recipe title/keywords
  - Ingredients
  - Username
  - Cuisine type
  - Meal type
  - Tags
  - Dietary restrictions
  - Cooking time range
  - Difficulty level
  - Chef score range
  - Nutritional criteria (e.g., low-carb, high-protein)
- Auto-complete suggestions
- Recent searches
- Advanced filters

User Stories:

- As a user, I want to search for "chicken recipes under 30 minutes"
- As a user, I want to find all vegan pasta recipes
- As a user, I want to search for users with high chef scores

---

2.8 Social Features

Priority: P0 (Must Have)

Requirements:

User Profiles:

- Display:
  - Profile picture
  - Username
  - Bio
  - Chef score
  - Total recipes
  - Follower count
  - Following count
  - Average recipe rating
- Recipe grid (public recipes only)
- Collections (if public)

Following System:

- Follow/unfollow users
- View followers/following lists
- Private accounts (optional future feature)

Interactions:

- Rate recipes (1-5 stars, one rating per user per recipe)
- Favorite recipes (Instagram-style heart)
- Comment on recipes (threaded conversations)
- Remix recipes (create derivative with credit to original)
- Share recipes externally

Chef Score System:

- Calculated based on:
  - Average ratings of user's recipes
  - Total favorites received
- Score displayed prominently on profile
- Leaderboard (optional)

User Stories:

- As a user, I want to follow my favorite recipe creators
- As a user, I want to see my chef score improve as people rate my recipes
- As a user, I want to comment on recipes to share tips or ask questions

---

2.9 Notifications

Priority: P1 (Should Have)

Requirements:

Notification Types (all configurable in settings):

- New follower
- Someone favorited your recipe
- Someone rated your recipe
- Someone commented on your recipe
- Someone remixed your recipe
- New recipe from followed user
- Recipe recommendations
- Pantry low stock alerts (if enabled)

Notification Settings:

- Enable/disable each notification type
- Push notifications
- In-app notifications
- Email notifications (optional)

User Stories:

- As a user, I want to be notified when someone favorites my recipe
- As a user, I want to control which notifications I receive

2.10 Collections & Organization

Priority: P1 (Should Have)

Requirements:

- Create custom recipe collections/folders
- Name and describe collections
- Add/remove recipes from collections
- Public/private collection settings
- Share collections
- Pre-defined smart collections:
  - My Recipes
  - Favorites
  - Recently Viewed
  - Quick & Easy (under 30 min)

User Stories:

- As a user, I want to organize my favorite recipes into themed collections
- As a user, I want to create a "Thanksgiving Dinner" collection to share with family

---

2.11 Offline Access

Priority: P1 (Should Have)

Requirements:

- Download recipes for offline viewing
- Access saved/favorite recipes without internet
- Sync changes when back online
- Indicate offline status clearly
- Limit on number of offline recipes (storage management)

User Stories:

- As a user, I want to access my favorite recipes while cooking, even without internet
- As a user, I want to download recipes before going to the grocery store

---

3. Technical Requirements

3.1 Platform

- **Framework:** Flutter (iOS and Android)
- **Minimum OS Support:**
  - iOS 13.0+
  - Android 8.0+ (API level 26)

3.2 Backend Services

- User authentication & management
- Recipe database & storage
- Image storage & CDN
- AI/LLM API integration for recipe generation
- OCR for recipe photo extraction

- Web scraping for URL imports
- Nutrition calculation API/service
- Push notification service
- Search indexing

3.3 Third-Party Integrations

- AI/LLM provider (OpenAI, Anthropic, or similar)
- OCR service (Google Vision, AWS Textract, or similar)
- Nutrition database API (for accurate estimates)
- Analytics (Firebase, Mixpanel)
- Crash reporting (Sentry, Firebase Crashlytics)

3.4 Data & Privacy

- GDPR/CCPA compliance
- User data encryption
- Secure authentication (OAuth 2.0)
- Recipe privacy controls
- User blocking/reporting features
- Content moderation for comments

3.5 Performance Requirements

- App launch time: < 3 seconds
- Recipe feed load time: < 2 seconds
- Image upload time: < 5 seconds (for reasonable sizes)
- AI recipe generation: < 30 seconds
- Offline recipe access: instant
- Search results: < 1 second

---

4. User Flows

4.1 New User Onboarding

1. Download app
2. Create account (email/OAuth)
3. Set dietary restrictions
4. Set skill level and preferences
5. Upload profile picture
6. Follow suggested users (optional)
7. Explore feed

4.2 AI Recipe Generation

1. Tap "Generate Recipe" button
2. Open chat interface
3. Optionally select ingredients from pantry
4. Describe desired recipe via chat
5. AI generates recipe
6. Refine through conversation
7. Review final recipe with nutrition info

8. Save to profile (public/private)

4.3 Recipe Discovery

1. Open app to personalized feed
2. Scroll through recommended recipes
3. Filter by meal type/preferences
4. Tap recipe to view details
5. Rate, favorite, comment, or remix
6. Add to collection or save for later

4.4 Recipe Import (Photo)

1. Tap "Add Recipe"
2. Select "From Photo"
3. Take photo or select from gallery
4. AI extracts recipe details
5. Review and edit extracted information
6. Add images and additional details
7. Save recipe

---

5. UI/UX Considerations

5.1 Design Principles

- Clean, modern, food-focused aesthetic
- Emphasis on high-quality recipe photography
- Intuitive navigation (bottom tab bar)
- Smooth animations and transitions
- Accessibility compliance (WCAG 2.1 AA)

5.2 Key Screens

1. **Home/Feed -** Personalized recipe feed
2. **Search -** Global search and discovery
3. **Generate -** AI chatbot for recipe creation
4. **Profile -** User profile and recipes
5. **Recipe Detail -** Full recipe view with all information
6. **Pantry -** Ingredient inventory management
7. **Collections -** Recipe organization
8. **Notifications -** Activity feed
9. **Settings -** App preferences and account management

5.3 Navigation

- Bottom tab bar: Home, Search, Generate (center/prominent), Pantry, Profile
- Top app bar: context-specific actions
- Floating action button: Quick recipe add

---

6. Success Metrics

6.1 Engagement Metrics

- Daily Active Users (DAU)
- Monthly Active Users (MAU)
- Average session duration
- Recipes generated per user
- Recipes saved per user
- Social interactions (follows, comments, ratings)

6.2 Quality Metrics

- Average recipe rating
- Recipe completion rate (users who cook recipes)
- User retention (D7, D30, D90)
- Chef score distribution

6.3 Technical Metrics

- App crash rate (< 1%)
- API response times
- Image load times
- Search query success rate

---

7. Future Enhancements (Post-MVP)

7.1 Phase 2 Features

- Video recipe tutorials
- Live cooking sessions/streaming
- Meal planning calendar
- Grocery delivery integration
- Recipe scaling calculator
- Cooking timers with notifications
- Voice-guided cooking mode
- Recipe challenges and contests

7.2 Phase 3 Features

- AI meal prep suggestions (weekly planning)
- Nutrition tracking and goals
- Integration with fitness apps
- Restaurant menu recreation
- Wine/beverage pairing suggestions
- Cooking class marketplace
- Premium subscription tier
- Brand partnerships and sponsored content

---

8. Decisions Made

1. **AI/LLM Provider:** GPT-4 Turbo (OpenAI API)
2. **Monetization Strategy:** TBD (to be determined in future planning)
3. **Content Moderation:** User banning system for inappropriate images, recipes, or bios. Implement reporting functionality and admin review process
4. **Recipe Copyright/URL Imports:** For problematic links, display error message: "Cannot generate recipe from the given link." Manual review for edge cases
5. **Maximum File Size for Recipe Images:** 10MB per image (allows high-quality photos while being reasonable for mobile)
6. **Localization/Internationalization:** TBD (initially English-only, expand based on user demand)
7. **Social Login Providers:** Google and Apple Sign-In only
8. **Community Guidelines:** Required (draft during development, include in onboarding)
9. **Recipe Versioning/Edit History:** Not visible to users (only current version shown)

---

9. Timeline & Phases

Phase 1: MVP (3-4 months)

- User authentication & profiles
- AI recipe generation
- Recipe CRUD operations
- Basic social features (follow, rate, favorite)
- Recipe feed with basic algorithm
- Search functionality
- Photo upload for recipes

Phase 2: Enhanced Social (2-3 months)

- Comments & discussions
- Recipe remixing
- Notifications
- Collections
- Chef score system
- Pantry management
- Shopping lists

Phase 3: Discovery & Polish (2 months)

- Advanced search and filters
- Improved feed algorithm
- Offline access
- URL import
- Photo OCR for recipe extraction
- Performance optimization
- Bug fixes and refinements

---

10. Appendix

10.1 Competitive Analysis

- **Yummly:** Recipe discovery with smart recommendations
- **Tasty:** Video-focused recipes with social elements
- **Paprika:** Recipe management and organization
- **Cooklist:** Pantry management and meal planning
- **Cookpad:** Community recipe sharing

Differentiation: AI-powered recipe generation combined with robust social features and personalized discovery algorithm based on pantry ingredients and real-time context (time of day, dietary needs).

10.2 Technical Stack

- **Frontend:** Flutter (Dart)
- **Backend:** Supabase
- **Database:** PostgreSQL (via Supabase)
- **Storage:** Supabase Storage
- **AI:** OpenAI GPT-4 Turbo API
- **Search:** Supabase Full-Text Search (PostgreSQL)
- **Authentication:** Supabase Auth (Google, Apple OAuth)
- **Real-time:** Supabase Realtime (for notifications, live updates)
- **Edge Functions:** Supabase Edge Functions (for server-side logic)

10.3 Supabase Architecture

Database Schema (PostgreSQL)

users

- id (uuid, primary key)
- email (text, unique)
- username (text, unique)
- display_name (text)
- bio (text)
- profile_picture_url (text)
- chef_score (numeric, default 0)
- skill_level (text) - beginner/intermediate/advanced
- dietary_restrictions (jsonb) - array of restrictions
- created_at (timestamp)
- updated_at (timestamp)

recipes

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- title (text)
- description (text)
- ingredients (jsonb) - structured ingredient list
- instructions (jsonb) - step-by-step array
- prep_time (integer) - minutes
- cook_time (integer) - minutes
- total_time (integer) - minutes

- servings (integer)
- difficulty_level (text) - easy/medium/hard
- cuisine_type (text)
- meal_type (text) - breakfast/lunch/dinner/snack/dessert
- nutrition (jsonb) - calories, macros, micros
- tags (text[]) - array of tags
- source_type (text) - ai/photo/url/manual
- source_url (text, nullable)
- is_public (boolean, default true)
- average_rating (numeric, default 0)
- rating_count (integer, default 0)
- favorite_count (integer, default 0)
- created_at (timestamp)
- updated_at (timestamp)

recipe_images

- id (uuid, primary key)
- recipe_id (uuid, foreign key → recipes)
- image_url (text)
- is_primary (boolean, default false)
- order (integer)
- created_at (timestamp)

follows

- id (uuid, primary key)
- follower_id (uuid, foreign key → users)
- following_id (uuid, foreign key → users)
- created_at (timestamp)
- UNIQUE(follower_id, following_id)

ratings

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- recipe_id (uuid, foreign key → recipes)
- rating (integer) - 1-5
- created_at (timestamp)
- updated_at (timestamp)
- UNIQUE(user_id, recipe_id)

favorites

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- recipe_id (uuid, foreign key → recipes)
- created_at (timestamp)
- UNIQUE(user_id, recipe_id)

comments

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- recipe_id (uuid, foreign key → recipes)
- parent_comment_id (uuid, nullable, foreign key → comments) - for threading
- content (text)

- created_at (timestamp)
- updated_at (timestamp)

collections

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- name (text)
- description (text)
- is_public (boolean, default false)
- created_at (timestamp)
- updated_at (timestamp)

collection_recipes

- id (uuid, primary key)
- collection_id (uuid, foreign key → collections)
- recipe_id (uuid, foreign key → recipes)
- added_at (timestamp)
- UNIQUE(collection_id, recipe_id)

pantry_items

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- ingredient_name (text)
- category (text)
- quantity (text, nullable)
- is_low_stock (boolean, default false)
- added_at (timestamp)
- updated_at (timestamp)

shopping_lists

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- name (text)
- created_at (timestamp)
- updated_at (timestamp)

shopping_list_items

- id (uuid, primary key)
- shopping_list_id (uuid, foreign key → shopping_lists)
- item_name (text)
- quantity (text)
- category (text, nullable)
- is_checked (boolean, default false)
- created_at (timestamp)

notifications

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- type (text) - new_follower/recipe_favorited/recipe_rated/comment/remix
- actor_id (uuid, foreign key → users) - who triggered the notification
- recipe_id (uuid, nullable, foreign key → recipes)

- comment_id (uuid, nullable, foreign key → comments)
- is_read (boolean, default false)
- created_at (timestamp)

ai_chat_sessions

- id (uuid, primary key)
- user_id (uuid, foreign key → users)
- recipe_id (uuid, nullable, foreign key → recipes) - null until saved
- messages (jsonb) - array of chat messages
- created_at (timestamp)
- updated_at (timestamp)

user_reports

- id (uuid, primary key)
- reporter_id (uuid, foreign key → users)
- reported_user_id (uuid, nullable, foreign key → users)
- reported_recipe_id (uuid, nullable, foreign key → recipes)
- reported_comment_id (uuid, nullable, foreign key → comments)
- reason (text)
- status (text) - pending/reviewed/resolved
- created_at (timestamp)

Row Level Security (RLS) Policies

Supabase RLS will enforce:

- Users can only edit their own profile
- Users can only see public recipes or their own private recipes
- Users can only delete their own recipes/comments
- Follows are publicly readable but only modifiable by the follower
- Pantry items are private to each user
- Notifications are private to each user

Storage Buckets

profile-pictures

- Public bucket
- Max file size: 5MB
- Allowed types: image/jpeg, image/png, image/webp

recipe-images

- Public bucket
- Max file size: 10MB
- Allowed types: image/jpeg, image/png, image/webp

recipe-photos-ocr (temporary storage for OCR processing)

- Private bucket
- Auto-delete after 24 hours
- Max file size: 10MB

Supabase Edge Functions

generate-recipe (POST)

- Accepts user input and pantry items
- Calls OpenAI GPT-4 Turbo API
- Returns generated recipe
- Stores chat session

extract-recipe-from-photo (POST)

- Accepts image upload
- Uses OCR service (Google Vision API or similar)
- Extracts recipe details
- Returns structured recipe object

import-recipe-from-url (POST)

- Accepts URL
- Scrapes webpage (with error handling)
- Extracts recipe details
- Returns structured recipe object or error message

calculate-chef-score (Database Function/Trigger)

- Triggered on recipe rating or favorite
- Recalculates user's chef score
- Updates users table

send-notification (Database Function/Trigger)

- Triggered on follows, ratings, favorites, comments
- Creates notification record
- Sends push notification via FCM

update-recipe-stats (Database Function/Trigger)

- Triggered on ratings/favorites
- Updates recipe average_rating, rating_count, favorite_count

Real-time Subscriptions

- **notifications**: Live updates for new notifications
- **comments**: Live comments on recipe pages
- **follows**: Live follower count updates

Full-Text Search

PostgreSQL full-text search on:

- recipe title, description, ingredients, tags
- user username, display_name
- Indexed for performance

Flutter Integration

Use `supabase_flutter` package:

yaml

```yaml
dependencies:
  supabase_flutter: ^latest_version
```

Key features:

- Built-in auth state management
- Real-time listeners
- Storage upload/download
- Database queries with type safety
- Offline support (with local caching)