CookBook : Your Virtual Kitchen Assistant

Team ID : NM2025TMID36263

Team Size : 5

Team Leader : DEEPA S

Team member : YOGAPRIYA J

Team member : KALIYAPERUMAL V

Team member : KATHER MAIDEEN A

Team member : PRIYANKA V

Introduction

CookBook: Your Virtual Kitchen Assistant is an all-in-one intelligent cooking platform designed to simplify, streamline, and elevate the home cooking experience. From recipe management to meal planning, pantry tracking, and shopping list generation, CookBook acts as a smart, responsive kitchen companion. Whether you're an experienced chef or a home cook looking for inspiration, CookBook helps users make informed, organized, and creative culinary decisions.

As modern cooking becomes increasingly digitized, CookBook bridges the gap between traditional recipe use and smart kitchen automation. It centralizes all kitchen-related activities in a unified interface that is accessible via web, mobile, or voice-enabled devices. By offering tools like voice-guided cooking modes, pantry-based recipe suggestions, and automatic ingredient scaling and substitution, the system turns fragmented cooking tasks into a seamless digital workflow.

This documentation provides a comprehensive overview of CookBook's architecture, features, data models, user flows, and integration points. It serves as a foundational blueprint for developers, designers, and stakeholders interested in building, extending, or adopting this virtual assistant platform in a real-world setting.

Table of Contents

---

1. Overview

“CookBook: Your Virtual Kitchen Assistant” is a system that helps users manage recipes, plan meals, track pantry inventory, generate shopping lists, and assist while cooking (with step-by-step guidance, voice prompts, etc.). It’s akin to having a smart companion in the kitchen that ties together all cooking-related tasks.

The system can work across devices (web, mobile, voice assistants) and should offer intelligent assistance (e.g. ingredient substitution, scaling, suggestions) and seamless workflows.

The documentation here describes the design in full detail, so that developers, product managers, or stakeholders can build or assess such a system.

---

2. Goals & Scope

2.1 Goals

- Centralize all cooking-related tasks (recipes, shopping, pantry, scheduling) into one system
- Reduce friction in cooking workflows (e.g. no manual conversions, no forgetting ingredients)
- Assist the user during cooking (e.g. voice guidance, timers, step-by-step)
- Learn user preferences and suggest recipes, meal plans
- Handle multiple devices, offline usage, sync
- Be extensible (support plugins, integrations with grocery APIs, smart home, etc.)

2.2 Scope: What's in / out

In scope:

- Recipe management (create, import, edit, categorize)
- Ingredient parsing, normalization, unit conversion
- Pantry / inventory tracking
- Shopping list generation & optimization
- Meal planning / scheduling
- Cooking mode / assistant (step-by-step, timers, voice)
- Sync across devices
- Basic suggestion engine / recommendation
- User preferences (dietary restrictions, allergens, cuisine tastes)
- Import recipes (web scraping, image OCR, link imports)
- Sharing / exporting (PDF, share link)

Out of scope (initially, or optional):

- Grocery delivery / placing orders
- Integration with smart kitchen appliances (though planning for this is possible)
- Advanced AI cooking — e.g. full recipe generation (though limited assist is possible)
- Robotic cooking execution
- Very heavy video / multimedia cooking lessons

---

3. Key Features & Use Cases

Here are typical user-facing features / flows:

- **Add / Import a recipe**

    - Manually enter recipe (title, description, ingredients, steps, images)
    - Import from website (paste URL, the system scrapes)
    - Import from image / OCR (user takes a photo of recipe, the system parses)
- **Recipe editing & organization**
    - Categorize by cuisine, meal type, tags, favorite level
    - Search / filter recipes (by ingredients, time, difficulty, nutrition, etc.)
- **Scale & Convert**
    - Change number of servings — scale ingredients accordingly
    - Convert units (grams ↔ cups, metric ↔ imperial)
    - Adjust for ingredient availability
- **Pantry / Inventory**
    - Track what the user already has (quantities, expiry dates)
    - Automatically suggest when something is missing when viewing recipe / planning
- **Shopping list generation**
    - Add missing ingredients to shopping list
    - Merge duplicates, group by aisle or category
    - Interactive check-off list
    - Export / share / print
- **Meal planning / scheduling**
    - Plan meals for a week / month
    - Link planned meals to recipes
    - Automatically generate shopping list based on plan
    - Swap / reassign meals
- **Cooking Assistant Mode**
    - Step-by-step mode that highlights current step, allows next / back
    - Voice mode: read steps, respond to "next" / "repeat" voice commands
    - Timer integration per step
    - Auto-skip, reminders, progress tracking
- **Suggestions / Recommendations**
    - Based on pantry, user preferences, seasonal produce
    - "You have these ingredients – here are recipes you can make"
    - Smart substitutions (if an ingredient is missing)
- **Sharing, exporting, backups**
    - Export recipe as PDF, share link, share with friends
    - Backup & restore user data
    - Optionally, publish to community
- **User Preferences & Settings**
    - Dietary restrictions / allergies (gluten-free, vegan, etc.)
    - Preferred measurement system (metric, imperial)
    - Favorites, rating of recipes

---

4. System Architecture

Here is a proposed architecture at a high level.

4.1 Components / Modules

Module	Responsibility
Frontend / UI	Web and mobile interfaces; cooking mode; input forms; voice interface
Backend / API Server	Core business logic; data persistence; synchronization; user management
Parser / Recipe Importer	Web scraping, text parsing, OCR, normalization
Conversion & Scaling Engine	Unit conversion, scaling, substitutions
Pantry / Inventory Engine	Track stock, expiry, auto suggestions
Meal Planner / Recommendation Engine	Planning, suggestions, constraints
Shopping List Generator	Build optimized shopping lists
Voice / Speech Module	Speech-to-text, text-to-speech interface
Sync / Offline Module	Local caching, synchronization, conflict resolution
Data Storage / Database	Persistent store for all entities
Third-party Integrations	Grocery API, grocery delivery, smart appliances, external recipe APIs

4.2 Data Flow (simplified)

1. User adds or imports recipe → Parser module processes → normalized recipe data stored in database
2. User views or edits recipe → via API → UI
3. User plans meals → the planner module selects recipes, cross-checks pantry → generates missing ingredients
4. Shopping list: the shopping list module takes missing items + merges duplicates, sorts, etc.
5. User cooks a recipe → cooking mode triggers timers, voice, step navigation
6. As user marks ingredients used, the pantry module updates inventory
7. Sync: changes from mobile or web sync via backend; offline edits are reconciled

4.3 Technology Stack Suggestions

- **Frontend**
  - Web: React, Vue, Svelte, or Angular
  - Mobile: React Native, Flutter, or native iOS/Android
  - Voice: integrate with Web Speech API or underlying platform TTS / STT
- **Backend / API**
  - Language: Node.js, Python, Ruby, Java, etc.
  - Framework: Express.js, Django, Rails, Spring, etc.
  - REST or GraphQL interface
- **Database**
  - Relational (PostgreSQL, MySQL) or document (MongoDB)
  - Possibly a search engine (Elasticsearch) for advanced recipe search
- **Storage for media**
  - Object storage (S3, Azure Blob, Google Cloud Storage)

- **Parser / OCR / NLP**
  - OCR: Tesseract, Google Vision API, etc.
  - NLP: custom parsing, or use pretrained models for ingredient phrase parsing
- **Sync / Offline**
  - Local database on device (SQLite, Realm)
  - Sync logic using conflict resolution (last-write-wins, or merge)
- **Voice / Speech**
  - Use platform TTS / STT (e.g. Google Speech-to-Text, Apple Speech, Web Speech API)
- **Infrastructure / Deployment**
  - Cloud hosting (AWS, GCP, Azure)
  - Docker / Kubernetes for scalability
  - CDN for static assets

---

5. Data Models

This section describes the key data models, their fields, and relationships.

5.1 Recipe

```
Recipe
- id (UUID)
- title (string)
- description (text)
- servings_default (integer)
- prep_time (duration)
- cook_time (duration)
- total_time (duration)  // optional or computed
- difficulty (enum: easy, medium, hard)
- cuisine (string / enum)
- tags (list of strings)
- nutrition_info (optional: calories, macros, etc.)
- image(s) (list of media references)
- source (URL or "user")
- created_at, updated_at
```

Relationships / child entities:

- **IngredientLine** (belongs to Recipe)

```
IngredientLine
- id
- recipe_id
- name (string)  // e.g. "tomato, diced"
- quantity (float)
- unit (enum / string, e.g. g, ml, cup, tbsp)
- optional (boolean)
- note (string)  // "to taste", "chopped fine"
- linked_ingredient_id (optional id referencing Ingredient master)
```

- **Step / Instruction** (belongs to Recipe)

```
Step
- id
- recipe_id
- sequence_number (int)
- instruction_text (string)
- timer_duration (optional)
- media (optional image/video)
```

- **RecipeMetadata / Extras**
  - rating, user notes, times cooked, favorites, flags (private/public)

5.2 Ingredient (Master / Catalog)

Maintain a master list of known ingredients to normalize names, units, synonyms, substitution groups.

```
Ingredient
- id
- canonical_name (string, e.g. "tomato")
- synonyms (list of strings)
- default_unit (string)
- nutritional_info (per default unit)
- substitution_group (id / key)  // to allow alternate items
- category (e.g. vegetable, dairy, spice)
```

5.3 Pantry / Inventory Item

```
PantryItem
- id
- user_id
- ingredient_id (or free-text fallback)
- quantity (float)
- unit (string)
- opened_date (optional)
- expiration_date (optional)
- location (e.g. "fridge", "pantry shelf")
- notes
```

5.4 Shopping List & List Item

```
ShoppingList
- id
- user_id
- name (string)
- created_at
- updated_at
- completed (boolean)
ShoppingListItem
- id
- shopping_list_id
- ingredient_id (or fallback)
- quantity (float)
- unit (string)
- checked (boolean)
- note (string)
```

5.5 Meal Plan / Scheduled Meal

```
MealPlan
- id
- user_id
- name (e.g. "Weekly Plan 2025-09-24")
- start_date
- end_date
- notes
PlannedMeal
- id
- meal_plan_id
- date (date)
- mealtime (enum: breakfast / lunch / dinner / snack)
- recipe_id
- servings (int)
```

5.6 User / Profile / Preferences

```
User
- id
- username / email
- password hash / auth credential
- preferences:
    - measurement_system (metric / imperial)
    - dietary_restrictions (list: vegan, gluten-free, etc.)
    - allergy list
    - favorite cuisines
    - default number of servings
```

5.7 Media

```
Media
- id
- owner_id (user or recipe)
- media_type (image, video)
- storage_url
- metadata (dimensions, format)
```

---

6. APIs & Interfaces

6.1 Internal / Backend APIs (for UI and other modules)

Here are core REST or GraphQL endpoints (examples). Exact interface / parameters to be refined.

Authentication / User

- `POST /api/signup`
- `POST /api/login`
- `GET /api/user/profile`
- `PUT /api/user/profile`

Recipe APIs

- `GET /api/recipes` — list with filters / search

- `GET /api/recipes/{id}` — get recipe detail
- `POST /api/recipes` — create a recipe
- `PUT /api/recipes/{id}` — update
- `DELETE /api/recipes/{id}`
- `POST /api/recipes/import` — import from URL / image

Ingredient APIs

- `GET /api/ingredients` — search / autocomplete
- `GET /api/ingredients/{id}`

Pantry APIs

- `GET /api/pantry`
- `POST /api/pantry`
- `PUT /api/pantry/{id}`
- `DELETE /api/pantry/{id}`

Shopping List APIs

- `GET /api/shopping-lists`
- `GET /api/shopping-lists/{id}`
- `POST /api/shopping-lists`
- `PUT /api/shopping-lists/{id}`
- `DELETE /api/shopping-lists/{id}`
- `POST /api/shopping-lists/{id}/items`
- `PUT /api/shopping-lists/{id}/items/{itemId}`

Meal Plan APIs

- `GET /api/meal-plans`
- `GET /api/meal-plans/{id}`
- `POST /api/meal-plans`
- `PUT /api/meal-plans/{id}`
- `DELETE /api/meal-plans/{id}`
- `POST /api/meal-plans/{id}/meals`

Cooking / Assistant APIs

- `GET /api/recipes/{id}/steps`
- `POST /api/recipes/{id}/start-cooking`
- `POST /api/recipes/{id}/next-step`
- `POST /api/recipes/{id}/repeat-step`
- `POST /api/recipes/{id}/timer-expired`

Sync / Offline APIs

- APIs to fetch deltas, push local changes, conflict resolution endpoints

6.2 External APIs / Integrations

Possible integrations:

- Recipe APIs / libraries (e.g. Spoonacular, Edamam) for recipe search / nutrition
- Grocery / retailer APIs (for price, availability, delivery)
- Smart kitchen / IoT (oven, fridge)
- Voice assistants (Google Assistant, Alexa)
- OCR / vision services
- Translation / localization / measurement conversion services

6.3 UI / UX / Voice / Chat Interface

- Traditional UI screens: recipe list, recipe detail, recipe editor, meal planner, pantry, shopping list
- Cooking mode: full-screen progressive mode, “next / back / skip / repeat” buttons
- Voice interface: user can say “next step”, “repeat”, “how much salt?”, etc.
- Chat-like interface: ask “What can I cook with these ingredients?”

---

7. Business Logic & Rules

This section describes core logic that governs behavior.

7.1 Parsing & Normalization

- Ingredient phrase parsing: break “2 (14 oz) cans diced tomatoes” into quantity, unit, name, notes
- Normalize synonyms (“tomato” vs “tomatoes”)
- Map units to canonical units
- Recognize ranges (“1–2 tsp”) and interpret default

7.2 Scaling & Conversion

- Scale ingredient quantities in proportion to new number of servings
- Handle fractional units, mixed units
- Convert between units (e.g. cups → grams) depending on ingredient-specific densities
- Bound checks (don’t produce absurd decimals)

7.3 Substitutions & Alternatives

- Maintain substitution groups (e.g. “buttermilk” → milk + vinegar, or “honey” → maple syrup)
- Allow user overrides
- Suggest substitution when pantry lacks an ingredient

7.4 Pantry Matching & Suggestions

- When viewing recipe, check pantry and highlight missing ingredients
- Use a threshold (e.g. if you have at least x% of needed amount, count as available)

- Suggest recipes you can make given your pantry
- Notify nearing expiration items

7.5 Shopping List Optimization

- Merge duplicate ingredients
- Convert quantities to single unit if possible
- Sort by grocery section / aisle
- Group by store if multi-store lists
- Optional: fetch price estimates or availability

7.6 Meal Planning Constraints & Suggestions

- Respect dietary restrictions, allergies
- Avoid repeating same cuisine too often
- Seasonal / local ingredient suggestions
- Calorie / nutrition target constraints
- Suggest recipes to fill gaps

---

8. Security, Privacy & Permissions

- **Authentication & Authorization**: secure login (OAuth / JWT / session), role checks if multi-user
- **Data Privacy**: recipes and user data should be private by default
- **Encryption**: use HTTPS, encrypt sensitive data at rest if needed
- **Data Access**: only allow users to read their own data
- **Backup & Export**: give users control over exporting or deleting their data
- **Third-party APIs**: manage API keys securely
- **Rate limiting / abuse protection**

---

9. Extensibility & Plugins

Design the system so additional modules or “plugins” can be added:

- Recipe source plugins (e.g. new websites)
- Grocery store plugins (for price/availability)
- Smart appliance connectors
- Custom dietary / nutrition rules
- Community / social sharing module
- AI extension module (e.g. recipe generation, image-based recognition)

Use a plugin architecture or modular design so core remains stable.

---

10. Localization & Internationalization

- Support multiple languages (UI, recipe text)
- Support multiple measurement systems (metric / imperial / local)
- Support local ingredient names / synonyms
- Support regional food categories / cuisines
- Date / time / unit formats by locale

---

11. Offline Support & Syncing

Because users may want to use the app offline (e.g. in kitchen with poor connectivity):

- Maintain a local database on device
- Cache user's frequently used recipes, meal plans
- Changes made offline are queued and later synced
- Conflict resolution (e.g. timestamp-based, or user confirmation)
- Merge non-conflicting changes automatically

---

12. Deployment & Infrastructure

- Use a cloud provider (AWS / GCP / Azure)
- Backend services containerized (Docker)
- Use orchestration (Kubernetes) for scaling
- CDN for static content
- Database with backups & replication
- Object storage for media
- Monitoring, logging, error reporting
- Versioned API support

---

13. Testing Strategy

- **Unit tests** for modules (parser, scaling, substitution, shopping list)
- **Integration tests** for API endpoints
- **End-to-end tests** for UI flows
- **Offline sync tests** (simulate offline / conflict)
- **Performance tests** (scaling recipes, large pantry)
- **Regression tests** for import, parsing correctness
- Use test environments, staging, automated CI/CD

---

14. Monitoring, Metrics & Analytics

- Track usage metrics: recipes added, meals planned, lists generated
- Error / crash reporting
- Performance metrics (API latency, DB queries)
- Analytics on which features are used most (to guide roadmap)
- Logging (with anonymization)

---

15. Example Flows / Use Cases

Use Case 1: Import & Cook a Recipe

1. User pastes a URL → backend calls parser → scrapes HTML, extracts recipe
2. The parsed recipe is normalized and stored
3. User views it, edits small corrections
4. User clicks "Cook" → cooking mode launches
5. Step 1 is shown, "preheat oven to 180 °C"
6. User says "next" → move to step 2
7. Step 2 has a timer → user taps "start timer"
8. Timer finishes → system alerts user to proceed
9. As user marks ingredients used, pantry updates
10. After finishing, user rates recipe, adds notes

Use Case 2: Plan Weekly Meals & Generate Shopping List

1. User opens meal planner
2. User drags or selects recipes for each day's meals
3. System computes required ingredients minus pantry stock
4. Shopping list is generated with missing items
5. User can edit list, mark as "to buy" or check items
6. Export shopping list as PDF or share

Use Case 3: Pantry-Based Suggestion

1. User indicates what they have: e.g. "chicken, onions, tomatoes, curry powder"
2. System searches user's recipe library and external sources
3. Suggest recipes that can be made with those ingredients (or mostly)
4. Offer substitution suggestions if minor missing items

---

16. Next Steps / Roadmap Ideas

- AI-powered recipe generation (from ingredients)
- Integration with grocery delivery / ordering
- Smart appliance / IoT integration (e.g. send recipe to smart oven)

- Multi-user / family shared pantry
- Community / social sharing of recipes
- Macro / nutrition tracking / integration with fitness apps
- Voice assistant (Alexa / Google) skill integration
- Advanced personalization (learning user tastes)

---

17. Glossary

- **Recipe**: A structured set of ingredients and instructions to prepare a dish.
- **IngredientLine**: A component of a recipe, giving name, quantity, unit, and notes.
- **PantryItem**: What the user currently has in stock.
- **Substitution Group**: A grouping of ingredients that can substitute for one another.
- **Cooking Mode**: The interactive mode during cooking (step-by-step, timers)
- **Meal Plan**: A schedule of planned recipes over a time period.

Conclusion

CookBook reimagines the kitchen as a smarter, more intuitive space — where technology not only assists but enhances how we plan, prepare, and enjoy food. By combining structured recipe data, intelligent planning tools, and user-focused design, CookBook becomes more than a digital recipe book — it becomes a trusted kitchen partner.

The modular design and extensibility allow for future innovations, such as AI-powered recipe creation, smart appliance integration, and personalized nutrition tracking. Whether for personal use, family meal planning, or integration into broader food tech ecosystems, CookBook offers a solid foundation for a modern culinary assistant.

With thoughtful features, detailed architecture, and flexibility at its core, CookBook is well-positioned to adapt to evolving user needs and cooking habits, ensuring it remains relevant, reliable, and rewarding to use.

THANK YOU