What-Can-We-Cook?

Team: Pseudocode Warriors

Team Members

- Ben Oz Elhadad – UX/UI
- Darell Isaac Sam – Backend Developer
- David Lucero – Docs/QA
- Nicholas Smit – Lead Backend Developer
- Sebastian Lopez – Backend Developer
- Stanley Lu – SQL Database Developer

Project Overview

- Web app to help users find recipes based on ingredients they have.
- Users can:
  - Search recipes by name or ingredients.
  - View recipe details with ingredients, instructions, optional video.
  - Maintain a personal pantry list (ingredient, quantity, unit).
- Tech stack:
  - **Backend:** FastAPI (Python), SQLite, SQLAlchemy
  - **Frontend:** HTML/CSS/JS
  - **Auth:** JWT-based login/register

Core Requirements

Recipe Discovery

- Browse/search recipes by title, description, ingredient, tag
- Recipes include a title, description, servings, total time, ingredients & steps

Instructional Support

- Step-by-step page (ordered steps)
- Optional video embed

Smart Ingredients List

- All ingredients required for a recipe are listed next to the steps
- Missing ingredients are marked as such according to a user's pantry

Pantry Management

- Users maintain a saved pantry in the app
- Simple add/remove with optional quantities & units

Implementation: High-Level Architecture

Frontend:

- Static pages (Browse, Recipes, Recipe Detail, My Pantry, Submit Recipe, Login/Signup).

Backend:

- FastAPI with routers for auth, user, ingredient, pantry, recipe.
- Controllers layer for business logic.

Database:

- Tables for User, Ingredient, Recipe, PantryIngredient, Category.

Key Features:

- **Fuzzy recipe search:**
  - Uses fuzzy matching on recipe title, description, and ingredient names.
- **Recipe detail page:**
  - Fetches recipe via `/recipes/{id}` and renders ingredients from IDs.
- **Pantry management:**
  - Uses authenticated endpoints to create/read/update/delete.
- **Auth and roles:**
  - JWT-based login/register.
  - Admin-only destructive operations (e.g., deleting users/recipes).

Code Inspection Feedback

- Peer reviewers have noted:
  - Need for clearer structure and comments in some backend functions
  - Opportunities to simplify or split larger controller methods
  - Input validation improvements (e.g., units, negative values)
  - Stronger emphasis on separate, well-structured test cases
  - Consistency in error handling and response codes

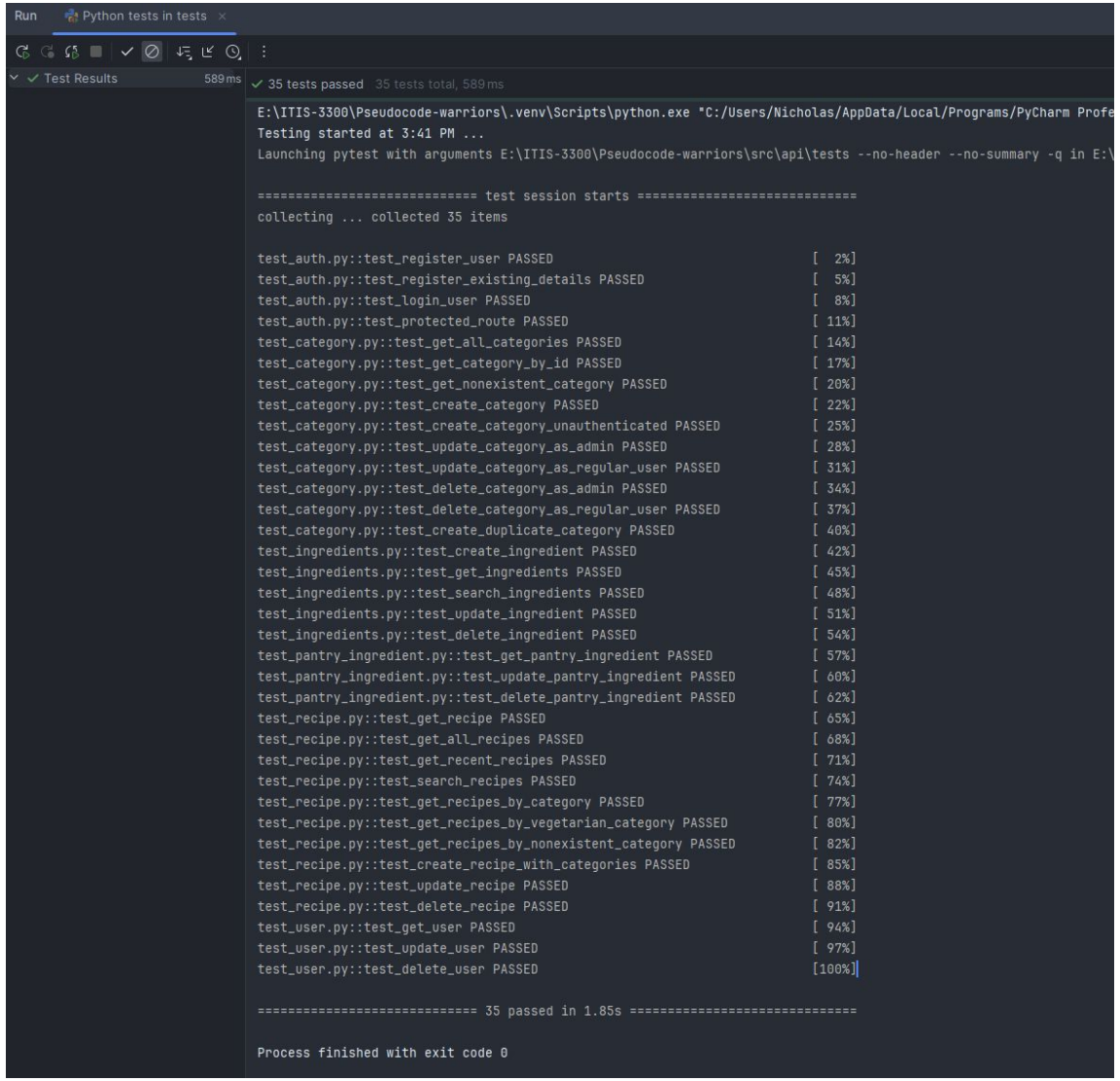
Testing Experiences

- Unit testing provided via `pytest`
- 35 unique tests across the exposed API surface
  - Covers CRUD operations for each route
  - Covers database constraints (i.e. duplicate registrations)
  - Covers JWT authentication verification
  - Covers permission checks for privileged routes (protected operations)
  - Covers any additional features that the front end needs to operate

Test Suite Results

- All tests pass as expected
- Provides extensive coverage of provided API services

•Results are reproducible - a runtime database is created, seeded, and tested with each launch to ensure a clean slate between runs.



Limitations & Constraints

Due to time constraints, some extended goals were dropped, and the overall project scope was adjusted to prioritize the stability of the core system and delivering the most important features.

- Recipe submissions were implemented without a queue/approval system
  - This could be implemented in the future once admin/moderator management pages are in place

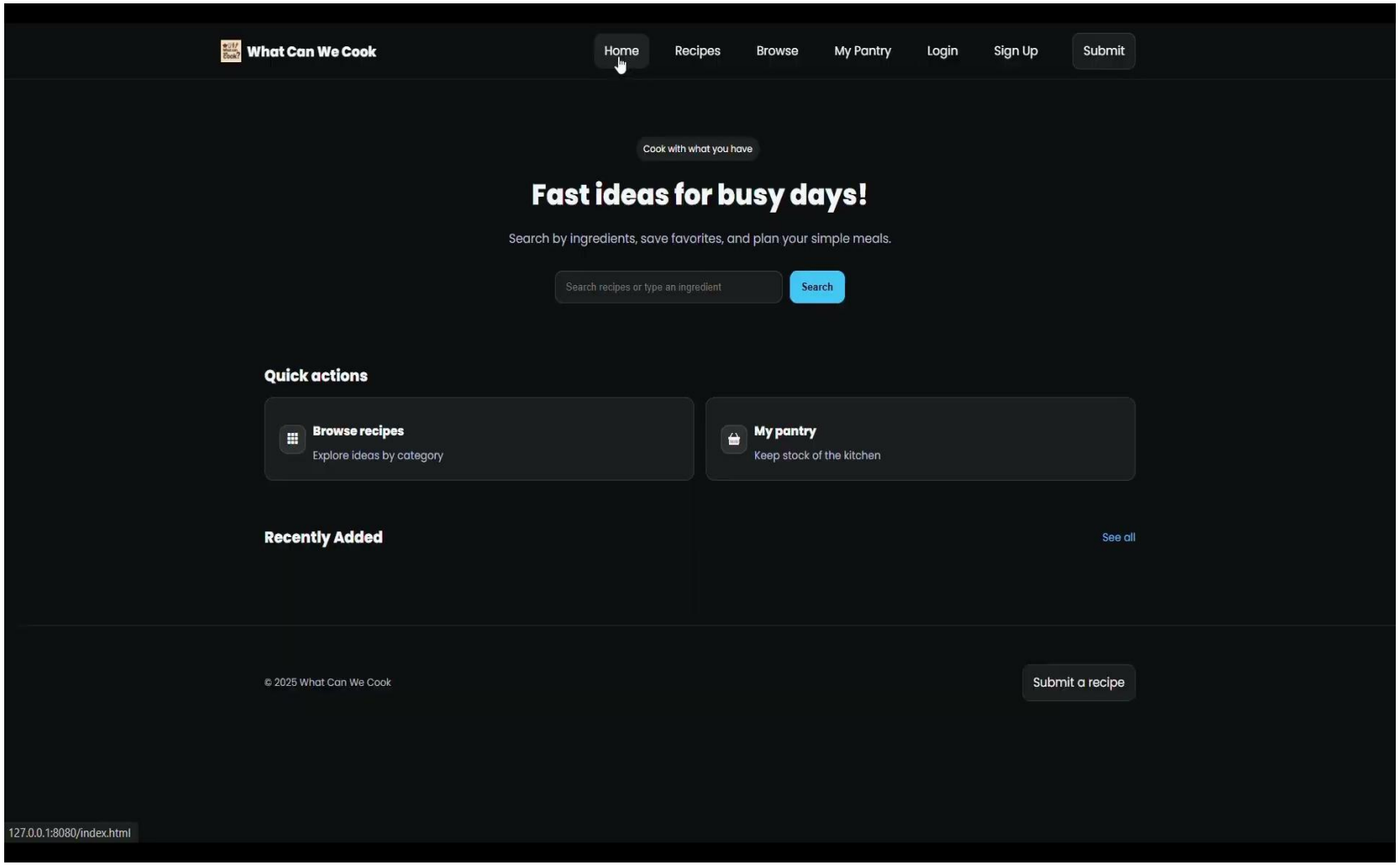
Limitations & Constraints

- Grocery list exports were dropped
  - This is partially rectified by the pantry-aware recipe views, which show what ingredients a user has/lacks

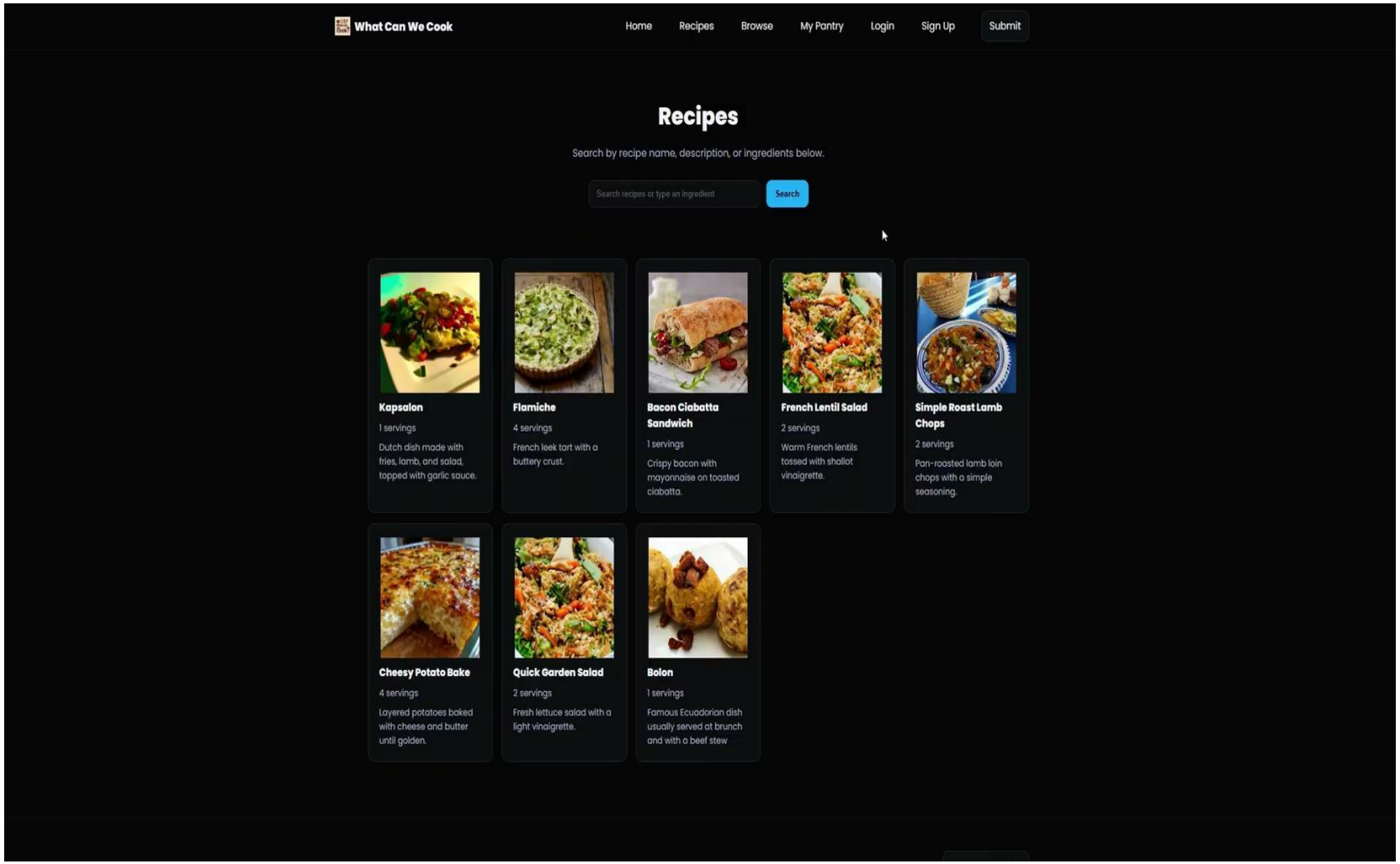
- Favorites and advanced filters (beyond category and text search) were dropped
  - Fuzzy-search across title/description/ingredients still provides an acceptable degree of usability

Feature Demos

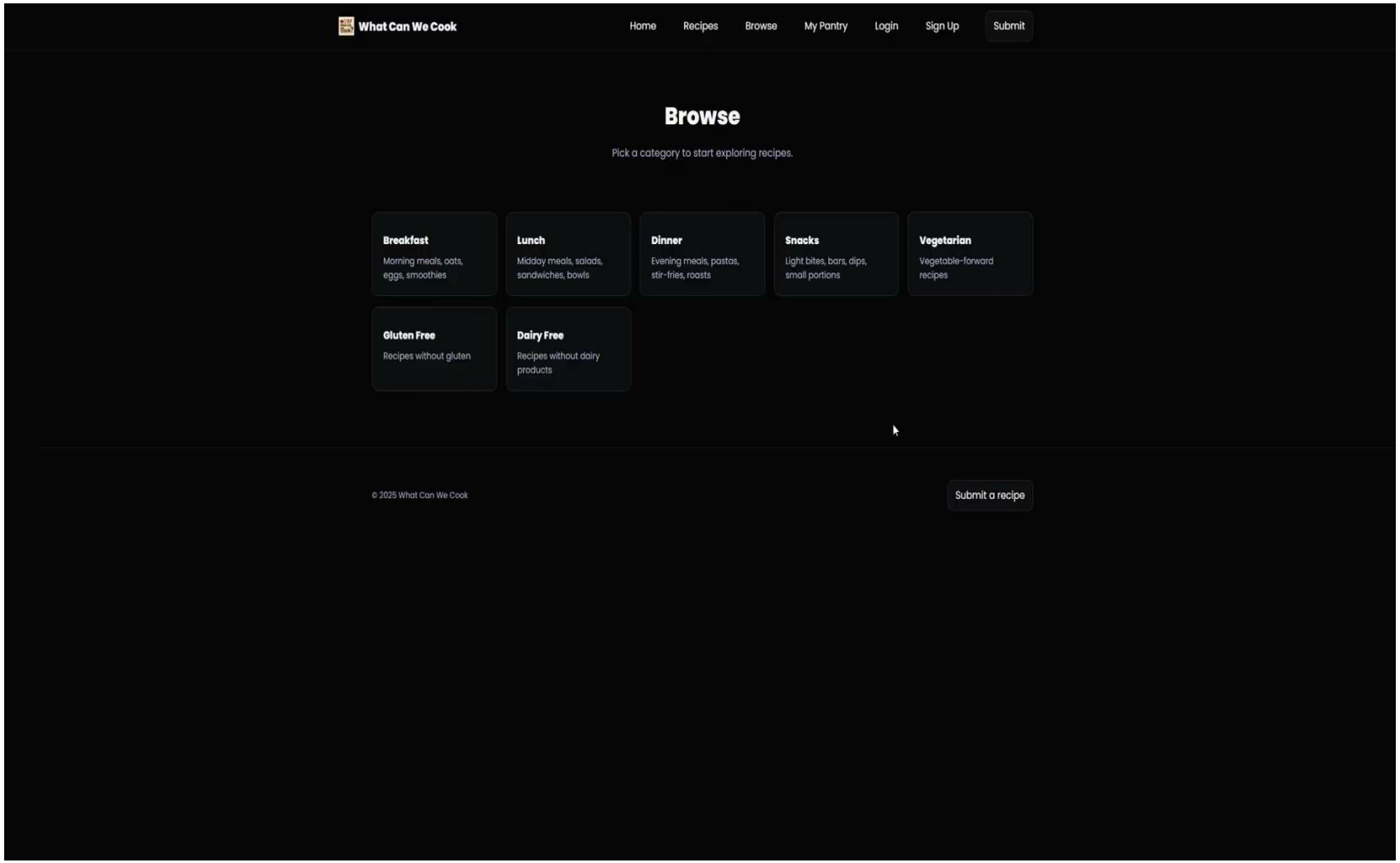
User Interface / Experience



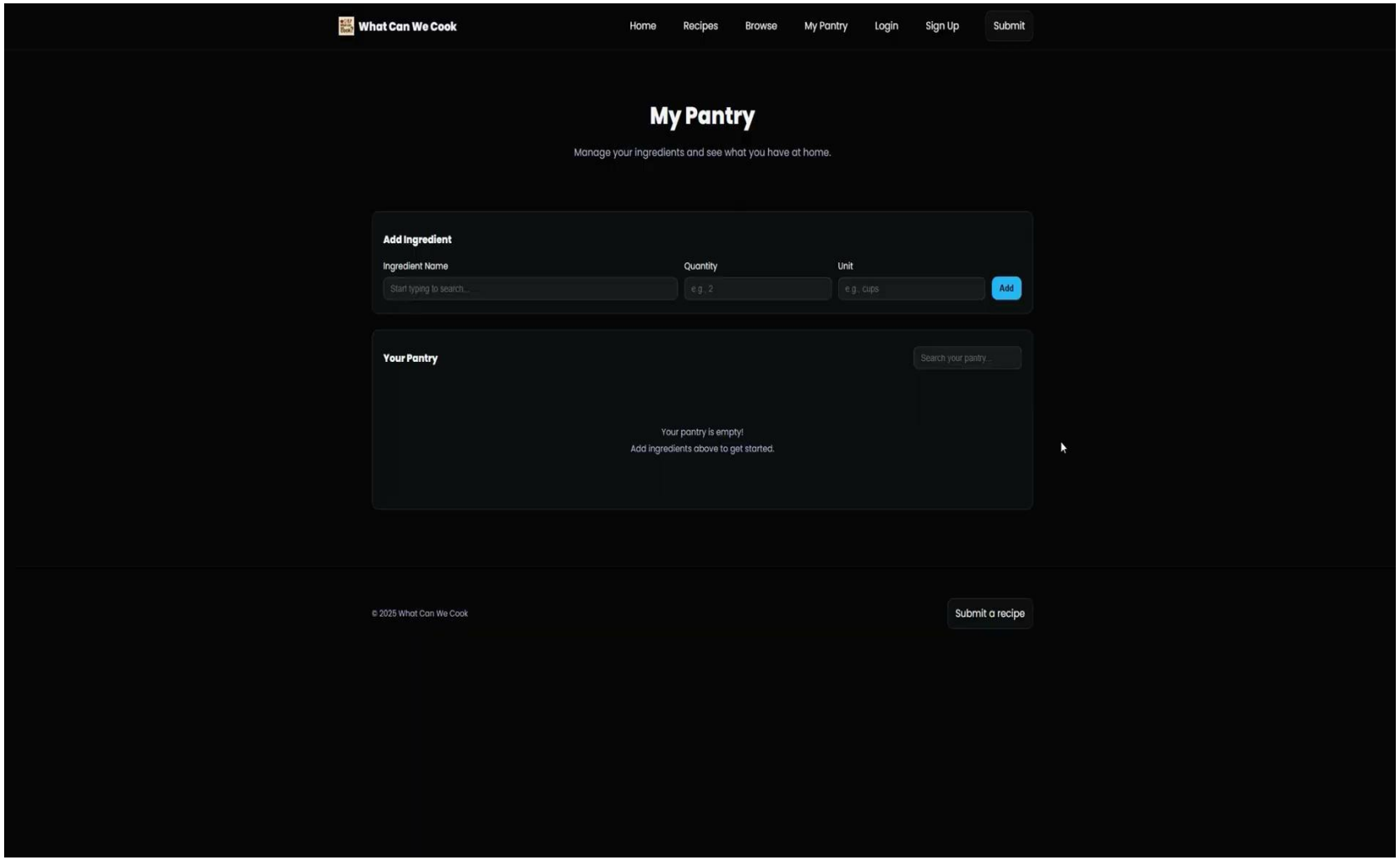
Recipe View / Search



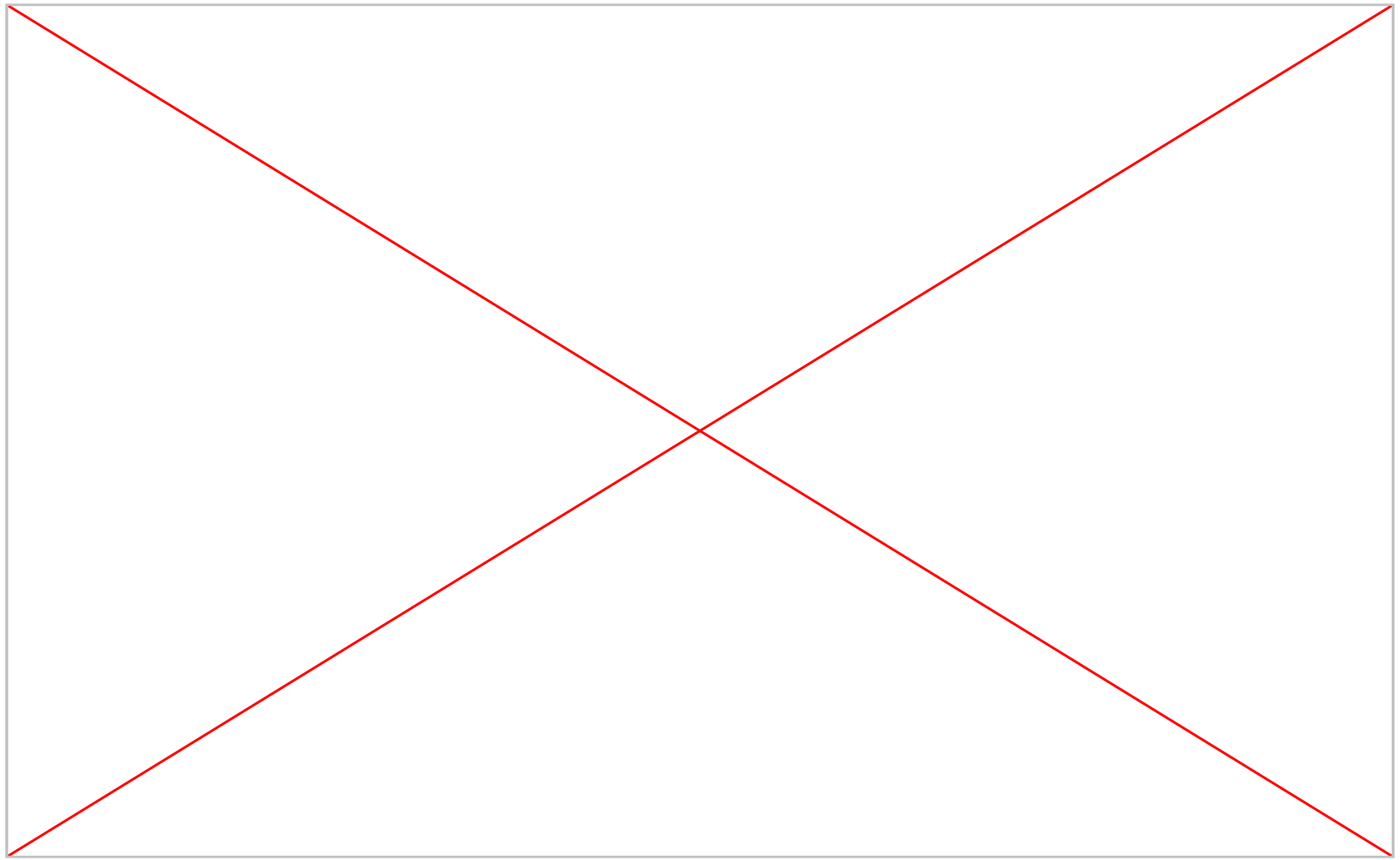
Category Browsing



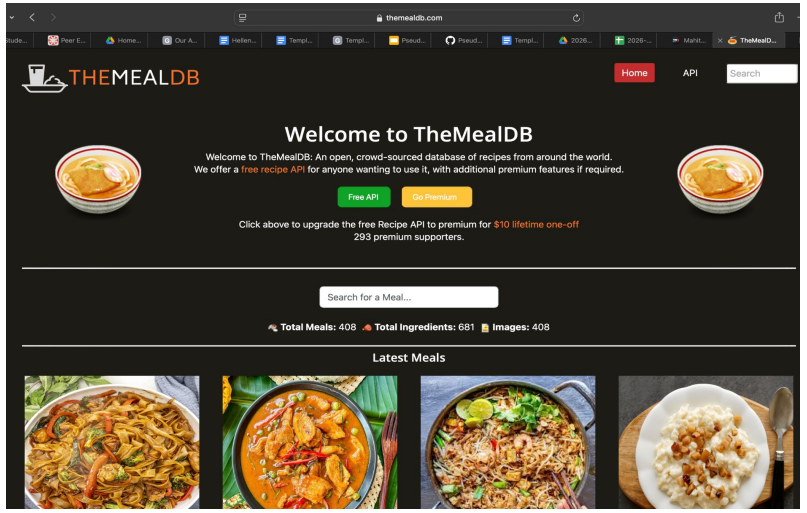
Pantry Management

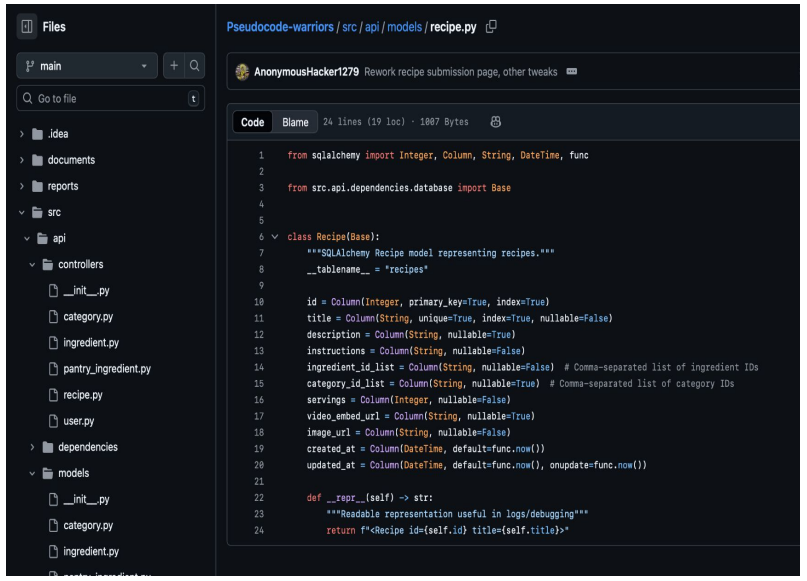


Recipe Submission



```python
def seed_if_needed():
    db = SessionLocal()

    if db.query(Ingredient).count() == 0:
        ingredients = [
            Ingredient(id=1, name="Bacon"),
            Ingredient(id=2, name="Lamb"),
            Ingredient(id=3, name="Mayonnaise"),
            Ingredient(id=4, name="Ciabatta"),
            Ingredient(id=5, name="French Lentils"),
            Ingredient(id=6, name="Leek"),
            Ingredient(id=7, name="Butter"),
            Ingredient(id=8, name="Cheese"),
            Ingredient(id=9, name="Potatoes"),
            Ingredient(id=10, name="Lettuce"),
            Ingredient(id=11, name="Green Plantain"),
            Ingredient(id=12, name="Yellow Plantain"),
            Ingredient(id=13, name="Chicharron"),
        ]

        db.add_all(ingredients)
        db.commit()

    if db.query(Category).count() == 0:
        categories = [
            Category(id=1, name="Breakfast", description="Morning meals, oats,
            Category(id=2, name="Lunch", description="Midday meals, salads, sa
            Category(id=3, name="Dinner", description="Evening meals, pastas,
            Category(id=4, name="Snacks", description="Light bites, bars, dips
            Category(id=5, name="Vegetarian", description="Vegetable-forward r
            Category(id=6, name="Gluten Free", description="Recipes without gl
            Category(id=7, name="Dairy Free", description="Recipes without dai
        ]

        db.add_all(categories)
        db.commit()
```





Database Overview

Conclusion

- Delivered a stable core app: recipe search, category browsing, detailed recipe views, pantry management, and recipe submission.
- Implemented pantry-aware recipe views so users can immediately see which ingredients they have and which they're missing.
- Built a structured pytest suite covering authentication, recipes, categories, ingredients, and pantry features to keep changes safer.
- Deferred stretch goals such as grocery list export, favorites, advanced filters, and a submission approval workflow to a future phase due to time and complexity limits.

References

- FastAPI – official documentation
- SQLAlchemy – official documentation
- SQLite – official documentation
- pytest – official documentation