

A close-up photograph of a person's hand holding a silver spoon over a white bowl filled with a steaming, light-colored soup. The steam is rising prominently from the bowl, creating a hazy, warm atmosphere against a dark background.

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

<u>Recipe Discovery</u>	<u>Instructional Support</u>	<u>Smart Ingredients List</u>	<u>Pantry Management</u>
<ul style="list-style-type: none">• Browse/search recipes by title, description, ingredient, tag• Recipes include a title, description, servings, total time, ingredients & steps	<ul style="list-style-type: none">• Step-by-step page (ordered steps)• Optional video embed	<ul style="list-style-type: none">• All ingredients required for a recipe are listed next to the steps• Missing ingredients are marked as such according to a user's pantry	<ul style="list-style-type: none">• 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

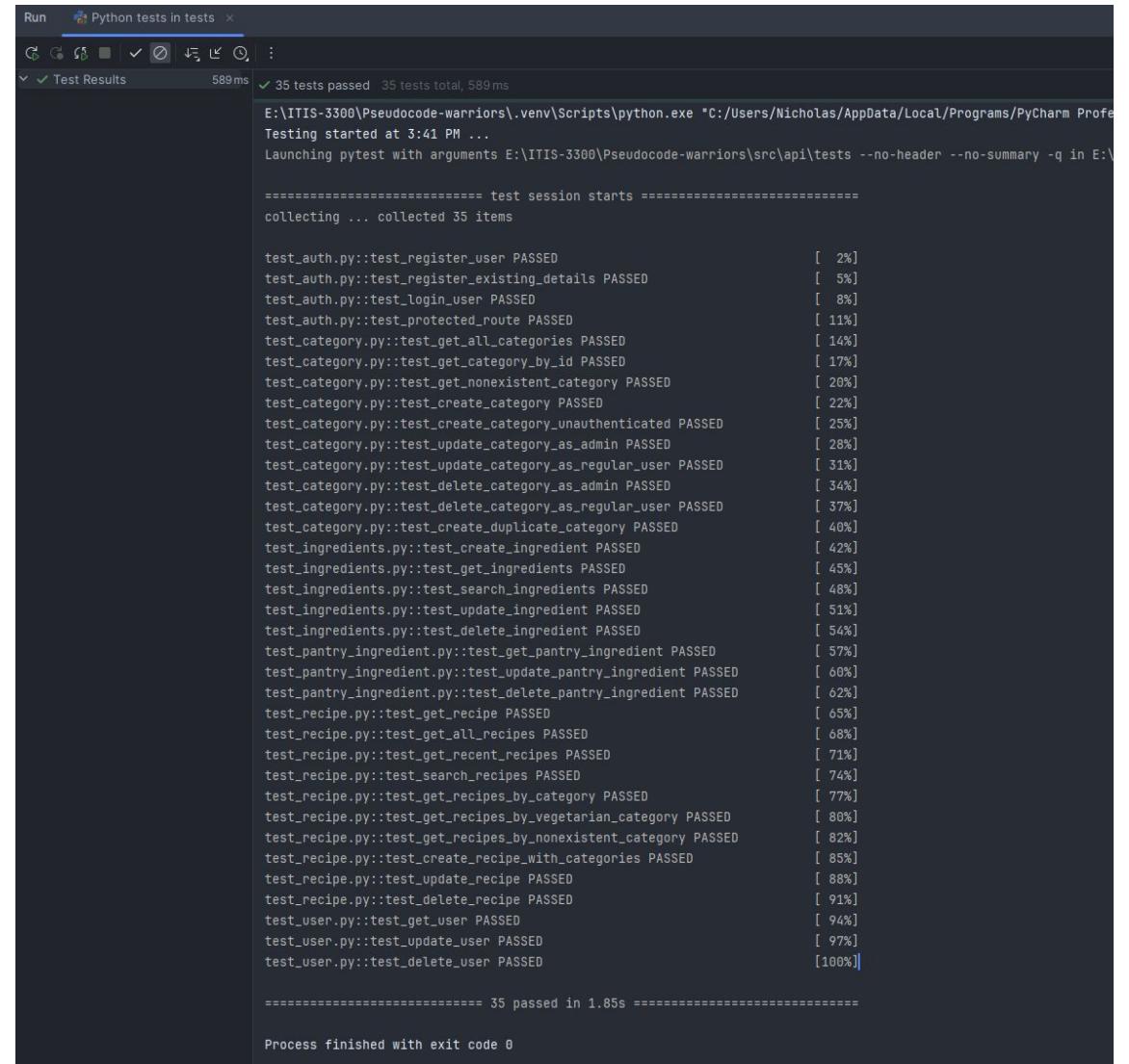
Testing Experiences

- 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.



The screenshot shows the PyCharm Run tool window with the title "Python tests in tests". The status bar indicates "589ms". The "Test Results" section shows 35 tests passed out of 35 total. The log output shows the command run: "E:\ITIS-3300\Pseudocode-warriors\.venv\Scripts\python.exe *C:/Users/Nicholas/AppData/Local/Programs/PyCharm Profe". It also shows the test session starting at 3:41 PM and launching pytest with specific arguments. The log lists 35 individual test cases, each with a progress bar indicating 100% completion. The final message at the bottom states "35 passed in 1.85s" and "Process finished with exit code 0".

```
E:\ITIS-3300\Pseudocode-warriors\.venv\Scripts\python.exe *C:/Users/Nicholas/AppData/Local/Programs/PyCharm Profe
Testing started at 3:41 PM ...
Launching pytest with arguments E:\ITIS-3300\Pseudocode-warriors\src\api\tests --no-header --no-summary -q in E:\ITIS-3300\Pseudocode-warriors\src\api\tests

=====
collecting ... collected 35 items

test_auth.py::test_register_user PASSED [ 2%]
test_auth.py::test_register_existing_details PASSED [ 5%]
test_auth.py::test_login_user PASSED [ 8%]
test_auth.py::test_protected_route PASSED [ 11%]
test_category.py::test_get_all_categories PASSED [ 14%]
test_category.py::test_get_category_by_id PASSED [ 17%]
test_category.py::test_get_nonexistent_category PASSED [ 20%]
test_category.py::test_create_category PASSED [ 22%]
test_category.py::test_create_category_unauthenticated PASSED [ 25%]
test_category.py::test_update_category_as_admin PASSED [ 28%]
test_category.py::test_update_category_as_regular_user PASSED [ 31%]
test_category.py::test_delete_category_as_admin PASSED [ 34%]
test_category.py::test_delete_category_as_regular_user PASSED [ 37%]
test_category.py::test_create_duplicate_category PASSED [ 40%]
test_ingredients.py::test_create_ingredient PASSED [ 42%]
test_ingredients.py::test_get_ingredients PASSED [ 45%]
test_ingredients.py::test_search_ingredients PASSED [ 48%]
test_ingredients.py::test_update_ingredient PASSED [ 51%]
test_ingredients.py::test_delete_ingredient PASSED [ 54%]
test_pantry_ingredient.py::test_get_pantry_ingredient PASSED [ 57%]
test_pantry_ingredient.py::test_update_pantry_ingredient PASSED [ 60%]
test_pantry_ingredient.py::test_delete_pantry_ingredient PASSED [ 62%]
test_recipe.py::test_get_recipe PASSED [ 65%]
test_recipe.py::test_get_all_recipes PASSED [ 68%]
test_recipe.py::test_get_recent_recipes PASSED [ 71%]
test_recipe.py::test_search_recipes PASSED [ 74%]
test_recipe.py::test_get_recipes_by_category PASSED [ 77%]
test_recipe.py::test_get_recipes_by_vegetarian_category PASSED [ 80%]
test_recipe.py::test_get_recipes_by_nonexistent_category PASSED [ 82%]
test_recipe.py::test_create_recipe_with_categories PASSED [ 85%]
test_recipe.py::test_update_recipe PASSED [ 88%]
test_recipe.py::test_delete_recipe PASSED [ 91%]
test_user.py::test_get_user PASSED [ 94%]
test_user.py::test_update_user PASSED [ 97%]
test_user.py::test_delete_user PASSED [100%]

=====
35 passed in 1.85s
Process finished with exit code 0
```

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



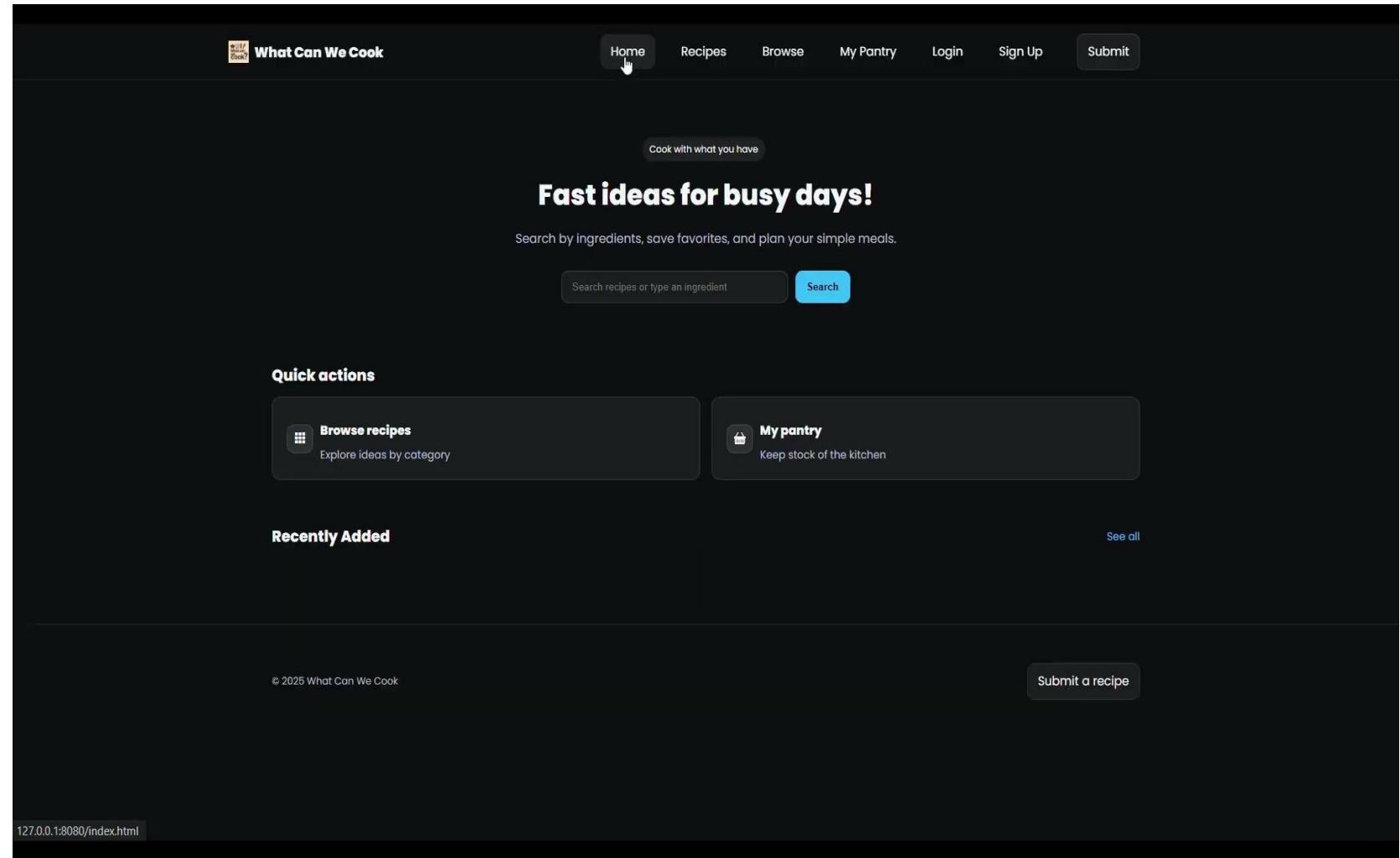
Limitations & Constraints

- 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



The screenshot shows the homepage of the "What Can We Cook" website. The header features a logo with a small icon and the text "What Can We Cook". A navigation bar includes links for "Home", "Recipes", "Browse", "My Pantry", "Login", "Sign Up", and "Submit". Below the header is a button labeled "Cook with what you have". A prominent banner reads "Fast ideas for busy days!". Subtext below the banner says "Search by ingredients, save favorites, and plan your simple meals." A search bar contains the placeholder "Search recipes or type an ingredient" with a blue "Search" button next to it. A "Quick actions" section contains two cards: "Browse recipes" (Explore ideas by category) and "My pantry" (Keep stock of the kitchen). A "Recently Added" section is partially visible, with a "See all" link. At the bottom, there's a copyright notice "© 2025 What Can We Cook" and a "Submit a recipe" button. A footer note at the bottom left indicates the page is "127.0.0.1:8080/index.html".

Recipe View / Search

What Can We Cook

Home Recipes Browse My Pantry Login Sign Up Submit

Recipes

Search by recipe name, description, or ingredients below.



Kapsalon
1 servings
Dutch dish made with fries, lamb, and salad, topped with garlic sauce.



Flamiche
4 servings
French leek tart with a buttery crust.



Bacon Ciabatta Sandwich
1 servings
Crispy bacon with mayonnaise on toasted ciabatta.



French Lentil Salad
2 servings
Warm French lentils tossed with shallot vinaigrette.



Simple Roast Lamb Chops
2 servings
Pan-roasted lamb loin chops with a simple seasoning.



Cheesy Potato Bake
4 servings
Layered potatoes baked with cheese and butter until golden.



Quick Garden Salad
2 servings
Fresh lettuce salad with a light vinaigrette.



Bolon
1 servings
Famous Ecuadorian dish usually served at brunch and with a beef stew

Category Browsing

The screenshot shows the homepage of the "What Can We Cook" website. At the top, there is a navigation bar with links for Home, Recipes, Browse, My Pantry, Login, Sign Up, and Submit. Below the navigation, the word "Browse" is centered in a large, bold font. A sub-instruction "Pick a category to start exploring recipes." is displayed below the title. There are seven categories arranged in two rows: Breakfast (Morning meals, oats, eggs, smoothies), Lunch (Midday meals, salads, sandwiches, bowls), Dinner (Evening meals, pastas, stir-fries, roasts), Snacks (Light bites, bars, dips, small portions), Vegetarian (Vegetable-forward recipes), Gluten Free (Recipes without gluten), and Dairy Free (Recipes without dairy products). At the bottom left, there is a copyright notice: "© 2025 What Can We Cook". On the right side, there is a "Submit a recipe" button.

What Can We Cook

Home Recipes Browse My Pantry Login Sign Up Submit

Browse

Pick a category to start exploring recipes.

Breakfast
Morning meals, oats, eggs, smoothies

Lunch
Midday meals, salads, sandwiches, bowls

Dinner
Evening meals, pastas, stir-fries, roasts

Snacks
Light bites, bars, dips, small portions

Vegetarian
Vegetable-forward recipes

Gluten Free
Recipes without gluten

Dairy Free
Recipes without dairy products

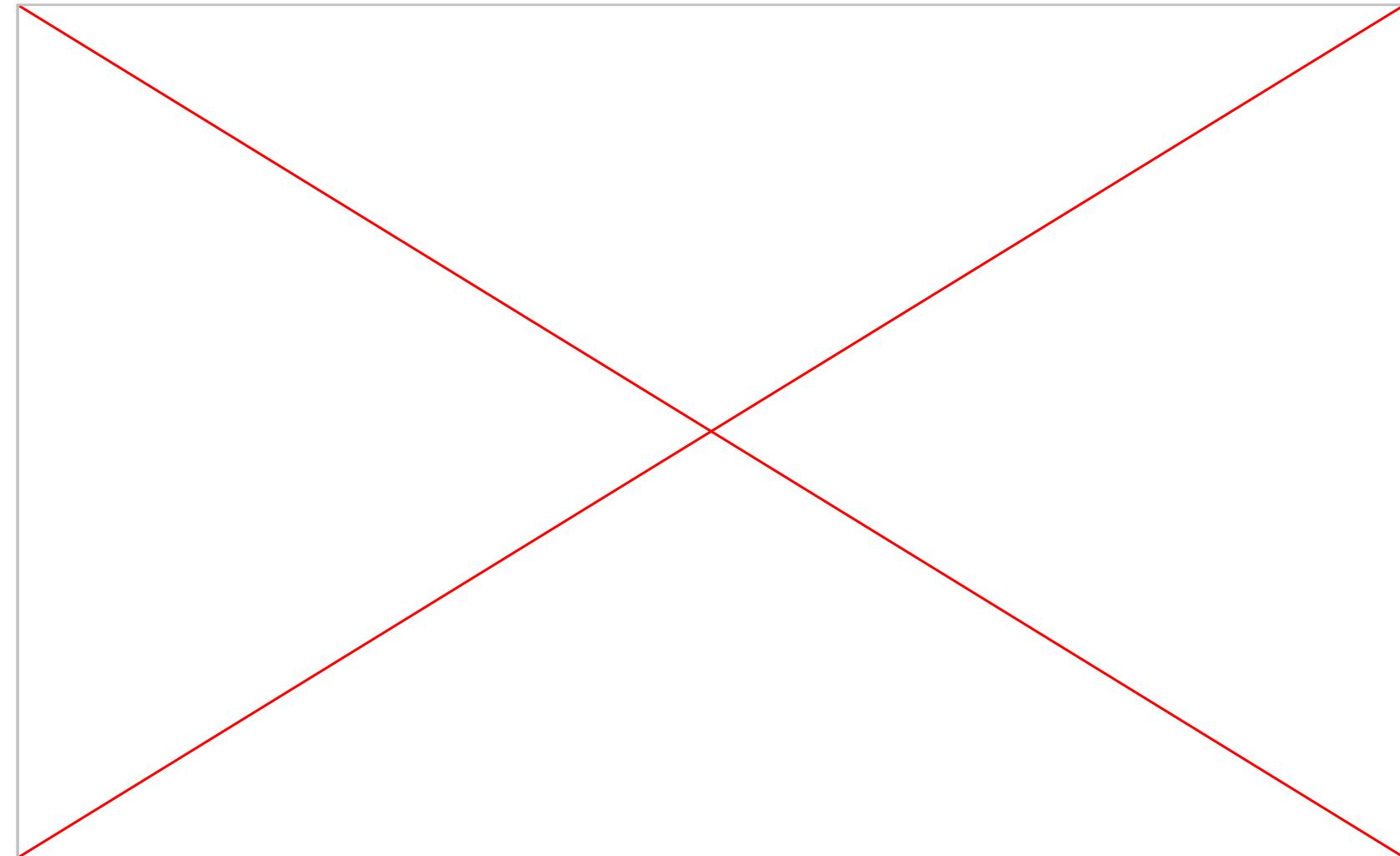
© 2025 What Can We Cook

Submit a recipe

Pantry Management

The screenshot shows the 'My Pantry' section of the 'What Can We Cook' website. At the top, there is a navigation bar with links for Home, Recipes, Browse, My Pantry, Login, Sign Up, and Submit. The main title 'My Pantry' is centered above a subtitle 'Manage your ingredients and see what you have at home.' Below this, there is a form titled 'Add Ingredient' with fields for 'Ingredient Name' (with placeholder 'Start typing to search...'), 'Quantity' (with placeholder 'e.g., 2'), and 'Unit' (with placeholder 'e.g., cups'). A blue 'Add' button is located to the right of these fields. Below this form is a dark panel titled 'Your Pantry' with a search bar labeled 'Search your pantry...'. A message inside the panel states 'Your pantry is empty! Add ingredients above to get started.' At the bottom of the page, there is a copyright notice '© 2025 What Can We Cook' and a 'Submit a recipe' button.

Recipe Submission



```

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()

```

The screenshot shows a dual-pane interface. On the left is a code editor with a dark theme, displaying Python code for seeding a database with ingredients and categories. On the right is a web browser window showing the homepage of TheMealDB, an open-source database of recipes. The browser's address bar shows 'themaldb.com'. The website has a dark header with the logo 'THEMEALDB' and navigation links for 'Home', 'API', and 'Search'. Below the header, there are two images of bowls of food. The main content area features a search bar with 'Search for a Meal...', a summary of data ('Total Meals: 408', 'Total Ingredients: 681', 'Images: 408'), and a section titled 'Latest Meals' with four thumbnail images of different dishes. At the bottom of the browser window, there is a code editor showing the file 'Pseudocode-warriors / src / api / models / recipe.py'. The code defines a SQLAlchemy model for a 'Recipe' class, which includes attributes for id, title, description, instructions, ingredient_id_list, category_id_list, servings, video_embed_url, image_url, created_at, and updated_at.

```

from sqlalchemy import Integer, Column, String, DateTime, func
from src.api.dependencies.database import Base

class Recipe(Base):
    """SQLAlchemy Recipe model representing recipes."""
    __tablename__ = "recipes"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, unique=True, index=True, nullable=False)
    description = Column(String, nullable=True)
    instructions = Column(String, nullable=False)
    ingredient_id_list = Column(String, nullable=True) # Comma-separated list of ingredient IDs
    category_id_list = Column(String, nullable=True) # Comma-separated list of category IDs
    servings = Column(Integer, nullable=False)
    video_embed_url = Column(String, nullable=True)
    image_url = Column(String, nullable=False)
    created_at = Column(DateTime, default=func.now())
    updated_at = Column(DateTime, default=func.now(), onupdate=func.now())

    def __repr__(self) -> str:
        """Readable representation useful in logs/debugging"""
        return f"<Recipe id={self.id} title={self.title}>"

```

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