KitchenSync

Developer and User Manual

Version: 1.0

Prepared by:

Chris Nederhoed

Tyler Son

David Tran

Institution: Florida Institute of Technology

Advised By: Dr. Fitzroy Nembhard

Submitted To: Dr. Philip Chan

Date: April 12, 2025

Contents

User Manual

Introduction & System Overview

What is KitchenSync?

KitchenSync is a smart pantry and meal planning companion designed to help home chefs, families, and students take control of their kitchen. Whether you're tracking what’s in your fridge, planning meals ahead, or trying to cut down grocery costs, KitchenSync brings all the tools you need into one clean, easy-to-use platform.

It replaces the need for multiple apps and manual tracking by offering a complete solution that syncs your inventory, recipes, and shopping habits—so you always know what’s on hand, what’s expiring, what you can cook, and what you need to buy.

Key features and goals (e.g., managing pantry inventory, planning meals)

Pantry & Inventory Management: Track what you have in your kitchen across spaces like the fridge, freezer, pantry, or cabinet.

Barcode & Receipt Scanning: Add items quickly by scanning barcodes or uploading receipts. No more guessing what you bought!

Smart Notifications: Get alerts when items are running low, close to expiring, or need to be used soon.

Recipe Management: Save, create, and share your favorite recipes with detailed steps, ingredients, equipment, substitutions, and nutrition info.

Meal Planner: Schedule meals day-by-day or week-by-week with automatic deduction from inventory and reminders for prep/cook times.

Shopping Assistant: Build store-specific lists, compare prices at local or online stores, and calculate meal costs per serving.

Ready-to-Cook Suggestions: Instantly see recipes you can make based on what's in your kitchen. Get recommendations for close matches or meals needing minor prep.

Admin Tools: Admins can seed recipes, moderate content, and manage users and feedback for a safe and helpful community.

Target users: Home cooks, students, families, administrators

KitchenSync was built with a wide range of users in mind:

- Home Cooks: Want to streamline cooking, reduce waste, and make meal prep easier.
- Busy Students: Need fast, budget-conscious meal planning based on what's in the dorm fridge.
- Families: Can plan meals for the whole household and share lists or menus.
- Admins and Moderators: Maintain the recipe library and community content.

How the system helps reduce food waste and plan meals efficiently

Many people struggle with forgetting what's in the fridge, throwing out spoiled food, or buying duplicates at the store. Existing solutions offer parts of the fix—but KitchenSync combines all of it into one smooth workflow.

By giving users visibility into their kitchen, personalized recipe suggestions, and automatic grocery list generation, KitchenSync makes cooking more convenient, budget-friendly, and sustainable.

Whether you're batch-prepping for the week or throwing together a meal from scraps, KitchenSync is your reliable companion in the kitchen.

Getting Started

System requirements (Java version, Python dependencies, OS)

To run KitchenSync, your device must meet the following minimum requirements:

Operating System: Windows 10 or higher (64-bit)
Internet Connection:

- Required for API calls (Open Food Facts, USDA, price scraping)
- Required for syncing with AWS DynamoDB and retrieving community content

Installation/launch instructions

1. Download KitchenSync
2. Unzip the file
3. Run the Installer

Creating a user account

1. Launch KitchenSync and select “Sign Up” on the login screen.
2. Enter:
    a. A display name
    b. A valid email address
    c. A secure password
3. Click “Create Account”. Your data will be securely stored using AWS DynamoDB.
4. After account creation, you’ll be redirected to the user dashboard.

Logging in / resetting password

1. On the main login screen, enter your email and password.
2. If you forget your password, click “Forgot Password?” and follow the prompts to reset it.
3. A verification email or temporary code may be used (feature depends on deployment settings).

Role-based access (User vs Admin)

KitchenSync uses role-based access to tailor functionality:

- User:
    - Add/manage inventory,
    - create/share recipes,
    - plan meals,
    - generate shopping lists,
    - scan receipts and barcodes,
    - set preferences
- Admin:
    - Add and moderate community recipes
    - Remove flagged/inappropriate content
      Edit or remove user-generated feedback
    - Manage user accounts and usernames
    - Seed initial recipe database

Using the Inventory System

How to add an ingredient

KitchenSync offers multiple ways to add ingredients to your inventory—whether you prefer a manual approach, scanning barcodes, or uploading receipts with OCR.

1. Manual entry
    a. Access the Inventory Tab:
       From your dashboard, click on the “Inventory” tab.
    b. Click “Add Ingredient”:
       A pop-up form will open.
    c. Fill in the Details:
        i. Name: Enter the ingredient name (e.g., “Whole Milk”).
        ii. Quantity: Specify the amount (e.g., “2”).
        iii. Unit: Choose a unit from the dropdown (cups, pints, liters, etc.).

iv. Location: Select where the item is stored (Fridge, Freezer, Pantry, or Custom).
v. Expiration Date: (Optional) Choose the expiration date using the date picker.
vi. Tags: Optionally, add tags such as “Expiring Soon,” “Dairy,” or “Organic” by typing what you want to tag a item with
vii. Save the Ingredient:
Click the “Save” button. The ingredient will now appear in your inventory list.

2. Barcode scanning
    a. Open the Add Ingredient Section:
    b. Select “Scan Barcode”:
    Instead of manual entry, choose the barcode scan option.
    c. Align Barcode:
    Point your camera at the barcode on the product’s packaging.
    d. Automatic Data Retrieval:
    The system uses the barcode data to retrieve product details from a linked database (e.g., Open Food Facts). If the information is found, it will auto-populate fields such as name and ingredients.
        i. If there is no data available then you will need to add the information by hand
    e. Review and Save:
    Verify the displayed details, choose location and tags, then click “Save.”

3. Receipt OCR upload
    a. Navigate to Receipt Upload:
    In the Inventory section, select the “Receipt Upload” option in “Add Ingredients”.

b. Upload Receipt Image:
   Choose an image file (JPEG, PNG) or take a photo of your receipt.
c. OCR Processing:
   KitchenSync uses Tesseract OCR and Easy OCR to extract item details from the receipt. The system then displays a list of detected products.
d. Verify Items:
   Check the list and manually correct any errors or missing data (e.g., product names and quantities).
e. Confirm & Save:
   Once verified, submit the list to add the ingredients to your inventory.

Editing or deleting items

Editing an Ingredient:

1. In your inventory list, right click on the ingredient you want to edit and select edit.
2. The detail pane will open with current information.
3. Make the necessary changes (e.g., update quantity, change expiration date).
4. Click “Update” to save changes.

Deleting an Ingredient:

1. Select the ingredient from your list by right clicking.
2. Click the “Delete” button
3. Confirm the deletion. The item will be removed from your inventory.

Viewing by location (Fridge, Freezer, Pantry)

KitchenSync organizes items based on where they are stored:

- Fridge: Fresh dairy, leftovers, and perishables.
- Freezer: Frozen meals, vegetables, and meats.
- Pantry/Cabinet: Dry goods, canned items, and spices.

How to View:

1. In the Inventory tab, you will see filter buttons
2. Views with “Fridge,” “Freezer,” or “Pantry” to filter the displayed items.
3. Items will automatically rearrange based on your selection for easier browsing.

Understanding tags (Expiring Soon, Dairy, Protein)

Tags help categorize ingredients for efficient tracking and notifications. Examples include:

- Expiring Soon (Automatic based on date set, if it is within 3 days of expiring then it will be marked as expiring soon): Mark ingredients that need immediate use.
- Dairy: For all milk-based and cheese products.
- Protein: Meats, legumes, and other protein sources.

Using Tags:

1. When adding or editing an ingredient, you can then input custom tags.
2. Use tags to filter your inventory—for example, you can display only “Expiring Soon” items or Items you tagged with “meat”
3. Tags help KitchenSync automate notifications and suggestions based on your current inventory for recipes in the Community Recipe Page or in the Meal Planner.

Recipes & Meal Planner

Creating a recipe (steps, images, equipment)

1. Adding a New Recipe:

- Access the Recipe Manager:
  Navigate to the “My Recipes” section from your dashboard.

- Start New Recipe:
  Click on the “Add Recipe” button to open the recipe creation form.
- Enter Recipe Details:
  - Name & Description: Give your recipe a title and a brief overview.
  - Steps:
    List sequential steps to prepare and cook the dish. You can add multiple steps by clicking “Add Step.”
  - Images:
    Upload images or take photos for each step as well as an overall dish image for the recipe card.
  - Equipment Needed:
    Enter any special kitchen tools or appliances required. You can also specify substitutes if available.
- Additional Information:
  - Prep Time, Passive Time, & Cook Time:
    Indicate how much time is needed for each stage. Prep is time needed to prepare ingredients and other items often referred to as “mise en place”. The Passive time block is where things like dough might need a little extra time to just hang out and proof, and lastly Cook time is the time required to cook the meal.
  - Ingredients:
    Input ingredients with their name, quantity and the unit used, Ideally the unit is in grams as this makes working with the recipe much easier.
  - Nutritional Information:
    Nutritional facts: calories, protein, carbs, fat, will be displayed in a pie chart for each recipe. It should be noted these are just estimates and might not truly reflect the amounts in a recipe..
- Save the Recipe:
  Once completed, click “Save”. Your recipe will be added to your personal collection and can be shared with the community if you choose.

Browsing community recipes

1. **Accessing the Community Recipes Section:**
    - Click on the “Community Recipes” tab from the main menu.
    - The page displays a grid or list of recipe cards, each showing:
        - A thumbnail image
        - Recipe name
    - Hovering over a recipe shows more information about it
    - Clicking on a recipe shows all the information about that recipe
2. **Filtering and Searching:**
    - Tags and Filters:
      Use filters for cuisine type, dietary restrictions (e.g., vegan, gluten-free), and difficulty level.
    - Search Bar:
      Quickly find recipes by entering keywords such as “pasta,” “salad,” or “quick dinner.”
    - Sort Options:
      Sort by cook time, A-Z, or recommended matches based on your current pantry contents, complete and ready are outlined in green, Complete but not ready are in yellow, missing ingredients in orange, and lastly no matches for ingredients are marked in red.
3. **Viewing a Recipe:**
    - Click a recipe card to view detailed instructions, ingredients, equipment, and nutritional info.
    - Option to save the recipe to your personal recipe collection or share feedback in the form of reviews.

Planning a meal (day, week, month views)

Meal Planning Views:

- **Day View:**
  See a detailed schedule for a single day. Each meal (breakfast, lunch, dinner, snack) is shown with its designated time block.
- **Week View:**
  Organize meals for the entire week on a grid that displays each day's planned recipes. Great for meal prepping.
- **Month View:**
  Visualize your meal plan for a longer period and plan for special occasions or events.

Using the Planner Interface:

- **Drag and Drop:**
  Easily assign recipes to specific days and meal slots using a simple drag-and-drop interface.
- **Editable Time Blocks:**
  Click on a meal block to adjust the scheduled time or change the recipe.

Scheduling meals with prep, passive, and cook time

Customizing Meal Phases:

- Prep Time:
  Enter the preparation duration, such as washing, chopping, or initial mix.
- Passive Time:
  Specify any waiting periods needed (e.g., dough rising, marinating).
- Cook Time:
  Set the time for actual cooking or baking.

How to Set Up:

- When scheduling a meal in your planner, enter the date you wish to place the meal

- The system automatically aligns the phases in sequence, ensuring you know when to begin prepping, allow for waiting, and start cooking.
- Visual indicators help distinguish each phase on the planner.
- Life happens and sometimes you may need to move the time blocks around, some of the recipes such as bread have a very specific time limit when it comes to proofing the dough these blocks are marked with a * to denote it is not recommended to move the block

Generating a shopping list

Shopping List Generation:

- After planning meals, navigate to the “MyList” section.
- The system compares your planned recipes with your current inventory.
- **List Creation:**
    - It automatically generates a list of ingredients that are missing or insufficient in quantity.
- **Manual Adjustments:**
    - Users can edit the list to add or remove items, or to adjust quantities.
- **Store Integration:**
    - For a more tailored experience, the system can fetch price estimates for these “needed ingredients”
    - You simply need to press the generate shopping list button for this process to take place after any changes you make.

Smart Features & Recommendations

“What can I cook now?” logic based on inventory

Real-Time Inventory Analysis:
KitchenSync continuously checks your current inventory against its recipe database. Based on the items you have on hand, the system identifies recipes that you can prepare immediately. This feature minimizes waste by suggesting meals that leverage ingredients before they expire.

Dynamic Matching:
Using pattern recognition and filtering algorithms, the system considers both mandatory ingredients and suitable substitutions. For example, if a recipe calls for basil but you only have parsley, KitchenSync may suggest that substitution if it fits the overall flavor profile.

Recommended meals when adding to planner

Smart Suggestions:
When you open the menu to select a recipe to cook the system highlights the recipes based on their level of completeness and ready to cook status: Complete and ready are outlined in green, Complete but not ready are in yellow, missing ingredients in orange, and lastly no matches for ingredients are marked in red. The system tries suggest green recipes first, yellow second, orange third, and if there are not enough of the first three categories then red recipes.

Nutrition estimates and goals

Automatic Nutritional Breakdown:
Every recipe in KitchenSync comes with detailed nutrition estimates, including calories, protein, fats, and carbohydrates. These estimates are computed using nutritional databases (e.g., USDA FoodData Central) linked to your ingredients to create a recipe macro breakdown in the form of pie chart.

Tag filtering (e.g., Vegetarian, Gluten-Free)

Flexible Tagging System:
Recipes and ingredients can be tagged with descriptors such as Vegetarian, Gluten-Free, Vegan, High-Protein, and more. Users can filter both the recipe library and inventory to show only items matching specific dietary tags.
Customized Searches:
When browsing community recipes or planning a meal, applying a tag filter will narrow down the available options to those that fit your dietary lifestyle or restrictions. This saves time and ensures that you only see relevant options, though our system is not perfect and something might slip through the cracks so use your best judgement if something seems off.

Highlighting missing or thaw-required ingredients

Inventory Mismatch Alerts:
When planning a recipe, KitchenSync flags any ingredients that are either missing or available in insufficient quantities. These items are added to your needed ingredient list after adding the meal to your meal planner.
Thaw-Required Indicators:
For ingredients like frozen meat or produce, KitchenSync checks their status. If an essential ingredient is frozen and requires thawing before use, the system will signal this with a visual cue. This helps in planning ahead so that items are ready when you need them.
Automated Adjustments:
The system can automatically subtract used quantities from your inventory once a meal is confirmed, keeping your available ingredients list up to date. If an ingredient is nearing its expiration or isn’t enough for the planned meal, you receive a timely reminder to purchase additional stock.

Support, FAQ, & Tips

Common troubleshooting (e.g., items not showing up, scanning fails)

- **Items Not Showing Up:**
  - **Refresh the View:**
    Click the “Refresh” icon in the Inventory tab.
  - **Check Filters:**
    Ensure you haven't accidentally applied filters that hide some items.
  - **Sync Issues:**
    Verify your internet connection; KitchenSync synchronizes with AWS databases regularly.
  - **Re-login:**
    Log out and log back in to reload your inventory data.
- **Scanning Failures (Barcode/Receipt):**
  - **Barcode Scanning:**
    - Ensure your camera has sufficient focus and lighting.
    - Clean the lens or the barcode if necessary.
    - Re-align the barcode within the scanning frame.
  - **Receipt OCR Upload:**
    - Use a clear, well-lit image.
    - Crop out unnecessary background elements.
    - Confirm that text on the receipt is legible.
  - **Error Messages:**
    If the system displays an error, check the notification area for specific instructions or try restarting the scan.
- **General Technical Issues:**
  - **Update the App:**
    Ensure you are running the latest version of KitchenSync.
  - **Check for System Alerts:**
    Review any on-screen notifications or prompts that suggest steps to resolve issues.

- **Reboot:**
  Try closing and reopening the application.
- **Consult the FAQ:**
  See the frequently asked questions below for any similar reported issues.

Tips for maximizing inventory life

- **Regular Updates:**
  Regularly update your inventory to reflect purchases and consumption. This minimizes waste and ensures recipes are suggested accurately.
- **Tag Expiry Dates:**
  Use the “Expiring Soon” tag to track items close to their use-by date. Plan meals that utilize these ingredients to reduce spoilage.
- **Proper Storage Recommendations:**
  Follow storage tips provided within KitchenSync (e.g., store perishables at the recommended temperature) to extend shelf life.
- **Batch Entry and Receipt Uploads:**
  Use the receipt OCR upload feature to quickly add multiple items at once, ensuring your inventory is always current.
- **Periodic Audits:**
  Once a week, review your inventory to remove expired items and adjust quantities as necessary.

FAQ (e.g., “Can I share recipes?”)

Can I share recipes with other users?

Yes, all users can post their own recipes

Administrators also have tools for moderating shared content.

What happens to my inventory when I plan a meal?

When a meal is marked as made, KitchenSync deducts the used ingredients from your inventory, keeping your stock levels accurate.

How secure is my account information?

KitchenSync uses AWS DynamoDB for secure storage of user data and follows industry-standard encryption practices for sensitive information.

How do I reset my password?

Use the “Forgot Password?” link on the login screen. Follow the emailed instructions to reset your password securely.

Can I sync multiple kitchens?

Currently, KitchenSync is designed to manage one kitchen inventory per account. Future updates may provide multi-kitchen support.

Glossary of terms (Prep Time, Passive Time, Meal Group ID, etc.)

- **Prep Time:**
  The estimated time needed to prepare ingredients before cooking begins. This includes washing, chopping, and mixing.
- **Passive Time:**
  Periods during cooking where active intervention is not required (e.g., dough rising, marinating).
- **Cook Time:**
  The actual time required for the cooking or baking process to complete.
- **Meal Group ID:**
  A unique identifier assigned to related meal components (e.g., different phases of the same meal) for tracking within a meal plan.
- **Inventory:**
  Your complete list of kitchen items tracked within KitchenSync. Items are organized by location (Fridge, Freezer, Pantry).
- **Tags:**
  Labels assigned to ingredients or recipes (such as “Expiring Soon,” “Dairy,” “Protein,” “Vegetarian,” or “Gluten-Free”) to facilitate filtering and smart

recommendations.

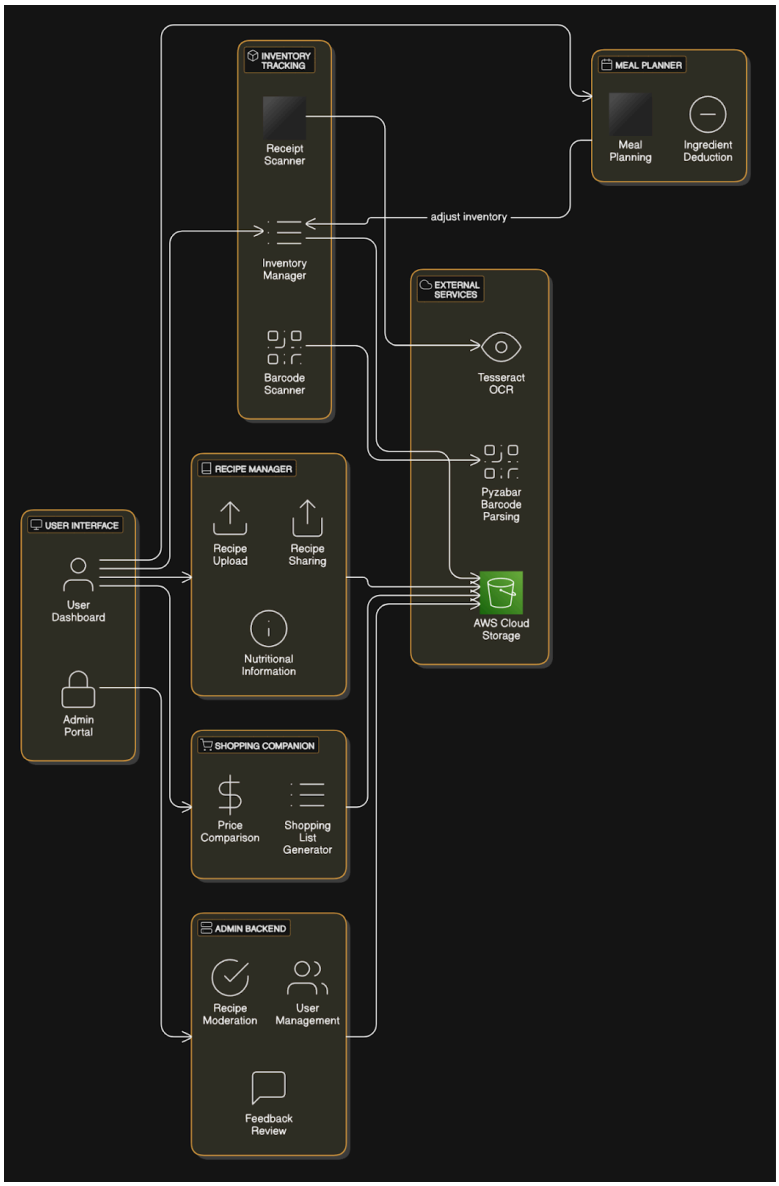
- **OCR (Optical Character Recognition):**
  A technology that converts different types of documents, such as scanned receipts, into editable and searchable data.

- **Barcode Scanning:**
  A quick method to input product information by reading a UPC or EAN barcode on the item’s packaging.

Developer Manual

System Architecture & Technologies

System architecture diagram:



Description of how components interact (e.g., Java calls Python)

Technologies used:

Technology	Version / Tool	Purpose & Notes
Java 17 & JavaFX	Java 17, JavaFX 23.0.1	Desktop UI framework. All screens and controls built in JavaFX, layouts designed in FXML.
SceneBuilder	20.0.0	Visually author/manipulate FXML scenes and bind controllers.
Python 3.10	CPython	Backend microservices for OCR, barcode decoding, and price scraping.
Tesseract OCR	v5.x	Parses receipt images into text—integrated via pytesseract.
AWS DynamoDB	Managed NoSQL	Stores Users, Recipes, Inventory, MealPlans.
AWS S3	Managed Object Storage	Hosts recipe images; accessed via pre-signed URLs.
Open Food Facts API	REST v2	Lookup product info (name, ingredients) by UPC for barcodes.
USDA FoodData Central	API v1	Fallback nutrition data when Open Food Facts is missing.
VS Code	1.80+	Primary IDE for both Java and Python development.
GitHub	Git + GitHub Actions	Source control, code reviews, and CI/CD pipelines.

Source Code Structure

- **Project structure overview:**

src/
  main/
    java/
      org/javafx/
          Controllers/ (Controllers for each page + supporting modules)
          Item/ (Item Class)
          Main/ (Main Body Class)
          Recipe/ (Recipe Class)
          AdminPortal.java (Admin portal class)
          Module-info.java (Maven Project connector)
        python/ (All the .py modules)
    resources/org/javafx/Resources/
          css/
          FXML/
          Item Images/
          Recipe Images/
          Test Receipts/
     jsons/
          Collections (Groupings for recipes to be aggregate into)
          flavorMatrix (Matrix of what ingredients pair well together + Flavor combinations)
          IngredientDictionary (Common ingredients with weights in grams)
          itemInventory (User Inventory)
          Lists (Users Lists)
          MealPlans (Meal Plan of current planned meals)
          Recipes
          SpacesAndCategories (User specified locations for storage)
          Substitutions (Common ingredient substitutions)

Key Java Classes & Methods

Overview of key classes:

- Main.java – Entry point
- InventoryController.java
- MealPlannerController.java

Each Controller is responsible for the screen in which it is named after. Functionality of that screen is handled by its respective controller. Some processes such as macros or pricing are handled by a .py script found in the python/ folder.

Sample method descriptions:

Method	Class	Description	Parameters	Returns
addIngredient(...)	InventoryController	Validates input, updates local list and calls DynamoDB API to store a new ingredient	name: String, quantity: double, unit: String, location: String, expiration: LocalDate, tags: List<String>	void
editIngredient(...)	InventoryController	Opens ingredient detail pane, applies changes to both UI and database	ingredientId: String, updatedFields: Map<String, Object>	boolean
deleteIngredient(id)	InventoryController	Removes item from UI list and issues delete to DynamoDB	ingredientId: String	boolean
populateReci	CommunityR	Fetches	*none* (uses	void

peFlowPane( )	ecipesControl ler	filtered recipes from DynamoDB, creates RecipeCard nodes, and adds them to the FlowPane	current filter state within controller)	
addMealToPl an(mealBloc k)	MealPlanner Controller	Inserts a new time block (prep/passive/ cook) into the model and refreshes the calendar grid and DB	mealBlock: Map<String,O bject> (contains recipeID, date, hour, duration, mealGroupId)	void
deleteMealFr omPlan(id)	MealPlanner Controller	Removes all time blocks sharing the same mealGroupId, updates UI and persists deletion in DynamoDB	mealGroupId: double	void
loadDailyNut rition(date)	MealPlanner Controller	Queries planned meals for a given date, aggregates nutrition values, and populates the PieChart widget	date: LocalDate	NutritionData

How to Extend

1. **Adding a New Controller**
    - Create NewFeatureController.java in org.javafx.Controllers.
    - Define corresponding NewFeatureView.fxml under /resources/fxml/.

    - Register the new scene in Main.java and add navigation via the side menu.
2. **Defining Methods**
    - Follow existing patterns:
        - **UI binding:** annotate with @FXML
        - **Event handlers:** name methods onXxxButtonClick()
        - **Persistence:** use DynamoDB SDK calls in a separate DataService helper class

Python Modules & Integration

BarcodeModule.py – Decodes barcodes, uses Open Food Facts API
ReceiptProcessor.py – Extracts text from receipt image via Tesseract
Integration via ProcessBuilder or Runtime.exec

Invocation

- Java controllers launch Python scripts using either

```
ProcessBuilder pb = new ProcessBuilder("python", "BarcodeModule.py", imagePath);
```

- Or

```
Runtime.getRuntime().exec(new String[] {"python", "ReceiptProcessor.py"});
```

Data Exchange

- **Input** (CLI args or JSON via STDIN):
    - Barcode script gets image file path as argument.
    - Receipt script reads image path or raw bytes from STDIN.
- **Output** (JSON via STDOUT):
    - Each script writes a JSON array or object representing parsed items.

```python
# ReceiptProcessor.py
import sys, json, pytesseract, cv2, re

def process_receipt(image_path):
    img = cv2.imread(image_path)
    text = pytesseract.image_to_string(img)
    # Simple regex to find “Name   Qty”
    lines = text.splitlines()
    items = []
    for line in lines:
        m = re.match(r"(.+?)\s+(\d+)", line)
        if m:
            items.append({
                "name": m.group(1).strip(),
                "quantity": int(m.group(2))
            })
    return items

if __name__ == "__main__":
    path = sys.argv[1]
    try:
        parsed = process_receipt(path)
        print(json.dumps({"items": parsed}))
    except Exception as e:
        print(json.dumps({"error": str(e)}))
        sys.exit(1)
```

Best Practices

- JSON-Only I/O: Keep all inter-process communication in JSON for reliable parsing.
- Version Pinning: Use requirements.txt to lock library versions (pyzbar==0.1.8, pytesseract==0.3.9).
- Timeouts & Retries: Configure Java’s ProcessBuilder with timeouts and retry logic to handle OCR delays or network hiccups.
- Logging: Have Python scripts emit logs to STDERR (not JSON) for debugging without polluting STDOUT.

Database & API Details

DynamoDB schema for:

- Recipes (DBID, category, complexity, cookTime, description, feedback, ingredients, name, passiveTime, prepTime, servings, specialEquipment, Steps, tags, UserId)

```
private void uploadRecipe() {
      if (!validateInputs()) {
         System.out.println("Please fill in all required fields.");
         return;
      }

      //Change to get the UsersID
      String userId = "testUserID123";  // e.g. "alice123"

      String recipeName = UploadRecipeName.getText();
      int prepTime = Integer.parseInt(recipeETAPrep.getText());
      int cookTime = Integer.parseInt(recipeETA.getText());
      int passiveTime = Integer.parseInt(recipeETAPassive.getText());
      int servings = Integer.parseInt(recipeYield.getText());
      String description = recipeDescription.getText();

      Map<String, AttributeValue> recipeItem = new HashMap<>();

      String recipeDBID = UUID.randomUUID().toString();


      recipeItem.put("Recipe",
AttributeValue.builder().s(recipeDBID).build());
      recipeItem.put("UserId",
AttributeValue.builder().s(userId).build());
      recipeItem.put("name",
AttributeValue.builder().s(recipeName).build());
      recipeItem.put("prepTime",
AttributeValue.builder().n(String.valueOf(prepTime)).build());
      recipeItem.put("cookTime",
AttributeValue.builder().n(String.valueOf(cookTime)).build());
```

```
      recipeItem.put("passiveTime",
AttributeValue.builder().n(String.valueOf(passiveTime)).build());
      recipeItem.put("servings",
AttributeValue.builder().n(String.valueOf(servings)).build());
      recipeItem.put("description",
AttributeValue.builder().s(description).build());

      recipeItem.put("ingredients", AttributeValue.builder()
         .l(ingredientEntries.stream()
            .map(val -> AttributeValue.builder().s(val).build())
            .collect(Collectors.toList()))
         .build());

      recipeItem.put("steps", AttributeValue.builder()
         .l(preparationSteps.stream()
            .map(step -> AttributeValue.builder().s(step).build())
            .collect(Collectors.toList()))
         .build());

      recipeItem.put("tags", AttributeValue.builder()
         .l(tags.stream()
            .map(tag -> AttributeValue.builder().s(tag).build())
            .collect(Collectors.toList()))
         .build());

      recipeItem.put("equipment", AttributeValue.builder()
         .l(equipment.stream()
            .map(eq -> AttributeValue.builder().s(eq).build())
            .collect(Collectors.toList()))
         .build());

      try {
         // Upload Recipe to DynamoDB
         database.putItem(PutItemRequest.builder()
               .tableName("Recipes")
               .item(recipeItem)
               .build());

         // Upload Image to S3
         // add multi image functions
```

```
        if (selectedImageFile != null) {
              String s3Key = userId + "-" + recipeDBID + ".jpg";
              s3Client.putObject(PutObjectRequest.builder()
                 .bucket("kitchensyncimages")
                 .key(s3Key)
                 .build(),
                 RequestBody.fromFile(selectedImageFile));

              System.out.println("Image uploaded to S3: " + s3Key);
        }

        System.out.println("Recipe uploaded successfully!");

        loadCommunityRecipes(); // Refresh community recipes

        addRecipePaneP2.setVisible(false);
        myRecipesPane.setVisible(true);

     } catch (Exception e) {
        e.printStackTrace();
        System.out.println("Error uploading recipe.");
     }
  }
```

APIs used:

- Open Food Facts (nutrition) + USDA Nutritional Information DB

Image storage

S3 Bucket -> For Cloud Storage on the community recipe page

Local Files -> Locally saved Recipes

Image naming convention: UserId + “-” + RecipeId + stepNumber. png

Recipe Id’s are generated using UUID to make sure similarly named recipes are still able to be uploaded and not to have issues with handling images. The stepNumber is only for recipes that have more then one image to be displayed.

Development Setup & Contribution Guide

How to run the project locally (JavaFX setup, Python requirements)

Prerequisites

- Java 17+ with JavaFX SDK 23.0.1
- Python 3.10+ with pyzbar, pillow, pytesseract, requests
- Tesseract OCR installed and on your PATH
- AWS Credentials configured (for DynamoDB/S3 access)
- Git and VS Code (or your IDE of choice)

1. Clone Repository
    a. git clone https://github.com/YourOrg/KitchenSync.git
    b. cd KitchenSync
2. Install Python Dependencies
    a. pip install -r requirements.txt
3. Build & Run JavaFX App
    a. mvn clean package
    b. java --module-path /path/to/javafx-sdk/lib \
    c. --add-modules javafx.controls,javafx.fxml \
    d. -jar target/KitchenSync.jar

How to add a new controller/screen

Design FXML

- Create src/main/resources/fxml/NewFeatureView.fxml in SceneBuilder.

Controller Class

- Add NewFeatureController.java under org.javafx.Controllers.
- Annotate UI elements with @FXML and implement init/event methods.

Wire It Up

In Main.java, add a menu/button action to load your new FXML:

```
FXMLLoader loader = new FXMLLoader(getClass()
    .getResource("/fxml/NewFeatureView.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
```

Test

- Run the app and navigate via your new menu item or button.

Coding guidelines (e.g., method naming, error handling)

Naming

- Classes: PascalCaseController (e.g., InventoryController)
- Methods: camelCaseAction() (e.g., onAddIngredientClick())
  Constants: SCREAMING_SNAKE_CASE

Error Handling

- Catch exceptions at the controller boundary
- Log detailed errors to STDERR or a log file
- Surface user-friendly messages via dialog boxes

Formatting

- 4-space indentation, max line length 100
- Use @FXML only on fields & handler methods

Documentation

- Javadoc for public methods and classes
- Inline comments for non-obvious logic

How to submit features or fix bugs

1. Branching
    a. git checkout -b feature/your-feature-name
    b. # or
    c. git checkout -b bugfix/issue-123-description

2. Implement & Test
    a. Write code, update FXML, add unit/integration tests.
3. Commit
    a. feat: add barcode auto-detect on inventory screen
    b. fix: handle null image paths in ReceiptProcessor
4. Push & PR
    a. git push origin feature/your-feature-name
5. Code Review
    a. At least one approval required.
    b. Address review comments with follow-up commits.
    c. Merge via “Squash and Merge” to keep history clean.

Git best practices:

- **Keep main Clean:** Always rebase or merge main before creating a PR.
- **Small, Focused PRs:** Aim for < 200 lines of change.
- **Descriptive Commits:** Start with feat:, fix:, chore:, or docs:.
- **Issue Tracking:** Link PRs to issue tracker tickets for traceability.
- **CI/CD:** Ensure all checks (unit tests, linting) pass before merging.