Smart Meal Planner

Members

Thinh Nguyen – U38848480

Kantemir Stalbekov – U36984825

Mirzo Ulugbek Rustamov – U96686084

Abstract

The Meals Manager app innovatively addresses the challenges of meal planning and food waste by leveraging user-input ingredients to suggest recipes. This software utilizes advanced data structures and algorithms to optimize meal suggestions based on ingredient availability and to minimize leftover ingredients, thereby reducing waste.

At the core of the application are several key data management components: a Meals Data system that catalogues recipes and their critical ingredients marked by an essentiality index and a Common Ingredient Graph that strategically suggests subsequent meals to maximize the use of remaining items.

Through this application, users receive intelligent meal recommendations that make efficient use of all kitchen resources while also offering easy and efficient meal preparation solutions. This project not only streamlines the cooking process but also promotes a sustainable lifestyle by minimizing food waste, a pivotal step towards eco-friendly living.

Implementation Details

Data Structures:

1. Meals Data: This structure is crucial for storing comprehensive meal-related information. Each entry includes the meals name and a list of ingredients, each marked with an "importance index." The index signifies the necessity of the ingredient for the meal; an index of 1 indicates that the meal cannot be prepared without it.

2. Common Ingredient Graph: A graph where each node represents a meal. Edges between nodes reflect the number of ingredients common to both meals connected by the edge. This is used to suggest subsequent meals by maximizing ingredient reuse, thereby minimizing waste.
3. Other STL data structures:
    a. List (Doubly Linked List): Storing each Meals data in a RecipeBook
    b. Priority Queue with min heaps: Storing meal suggestions based on user-input ingredients. Lowest priority index means that the meal has most ingredients appeared on the user-input ingredients list.
    c. Priority Queue with max heaps: Storing next meal suggestion based on number of overlapping ingredients to current meal. Highest priority index means the meal has the greatest number of overlapping ingredients.

Classes and Functions

1. Meal class:

   Purpose: Manages data for individual meals.

   Key Methods:

   updatePriority(string ingredientName, int priority): Decrements the priority of an ingredient based on its availability and importance.

   getTotalPriority(): Aggregates priority scores to determine the likelihood of suggesting the meal.

2. RecipeBook class:

   Purpose: Acts as a database for all meals and ingredients.

   Key Methods:

   prioritizeMeals(vector<string> ingredientList): update the ingredients priority of each meal according to the user-input list of available ingredients generateSuggestionQueue(): creates a priority queue of meals based on their

priority levels. This queue serves as a suggestion list for todays meal based on the availability of ingredients.

3. CommonIngredientMap class:

   Purpose: Uses graph theory to find and suggest meals with overlapping ingredients.

   Key Methods:

   initializeEdges(): draw edges between Meal nodes, with the weight is the number of overlapping ingredients between them.

   nextMeals(string currentMeal): Suggests the next meal to cook based on the highest number of common ingredients with the current meal

Main Algorithms

1. Ingredient-Based Meal Suggestion:

   Upon receiving user input for available ingredients, the application update the priority of each meals. Each potential meal is then processed through the Meal class to update its priority score based on the current availability of ingredients.

2. Priority Queue Management for Meal Suggestions:

   Meals are added to a priority queue where the meal with the highest priority (least ingredient deficits and highest necessity) is suggested first. If the user declines a meal, the next highest priority meal is suggested.

3. Graph-Based Next Meal Suggestion:

   Using the CommonIngredientMap, once a meal is selected, the next suggestions are generated based on a graph search for meals with the most overlapping ingredients. This utilizes a second priority queue to manage these suggestions effectively.

Challenges and Learnings

Challenges

The development of the Meals Manager app presented multiple technical and conceptual challenges:

- Data Management Complexity: One of the initial challenges was efficiently managing and querying large datasets represented by the Meals Data and the Ingredient Lookup Table. Achieving swift and accurate search results required optimizing the data structure choices and query algorithms.
- Algorithm Optimization: The Common Ingredient Graph needed to efficiently suggest next meals based on overlapping ingredients. Crafting an algorithm that could handle dynamic updates and provide quick suggestions under varying conditions was particularly challenging.
- User Interface Design: Ensuring that the user interface was intuitive and user-friendly while providing all necessary functionalities presented a steep learning curve. This was crucial as the ease of entering ingredients and receiving suggestions impacts user satisfaction directly.

Learnings

Despite these challenges, the project was a valuable learning opportunity:

- Advanced Data Structures: The project allowed the team to deepen their understanding of complex data structures such as graphs and hash tables. This included learning how to implement these structures in a way that maximizes efficiency and scalability.
- Algorithmic Efficiency: Developing the ingredient overlap algorithm for meal suggestions enhanced our skills in algorithm design, especially in terms of optimizing existing algorithms for better performance.
- Software Design Principles: We gained insights into the principles of software engineering practices such as modular design and abstraction. This was particularly evident in how different classes and functions were organized to maintain code readability and reusability.
- Team Collaboration and Management: Working on this project improved our teamwork and project management skills. Effective communication and division of tasks were key to overcoming the challenges faced during development.

Contributions

The Meals Manager project was a collaborative effort with clearly defined roles that focused on the backend development and algorithmic logic of the application. Below are the detailed contributions of each team member:

Mirzo Ulugbek Rustamov:

- Primary Roles: Developer responsible for critical data structures and backend logic.
- Specific Contributions:

  Developed and managed the Meal class, implementing essential methods such as updatePriority() which adjusts the priority based on ingredient availability, and getTotalPriority() which aggregates these priorities to guide meal suggestions.

  Integrated the Meal class functionalities with the main application logic to ensure smooth data flow and operation within the system.

Thinh Nguyen:

- Primary Roles: Lead on algorithm development and data integration.
- Specific Contributions:

  Created and optimized the RecipeBook class, which involved writing key functions like addMeal() for incorporating new recipes into the system and findMealsWithIngredient() for efficiently searching through the database based on user inputs.

  Engineered the CommonIngredientMap class and associated algorithms. This included constructing the initializeEdges() function to map out the relationships between different meals based on shared ingredients and the nextMeals() function to recommend future meals based on current selections.

Kantemir Stalbekov:

- Primary Roles: System integration and testing.
- Specific Contributions:

Ensured all individual class components were cohesively integrated, allowing the backend logic to function as a complete unit. This included setting up the main operational flow in main.cpp which ties all separate components like Meal and RecipeBook together.

Led the debugging and testing phases to guarantee that all backend functionalities were reliable and effective under various scenarios. This involved meticulous testing of all backend functions to ensure they met operational standards.