Fork & Function

Object-Oriented C++ Application

Group 1: Austin Luong, Damian Perez, John Rilles, & Max Owens – May 2025

Project Overview

A recipe management system for all dietary lifestyles

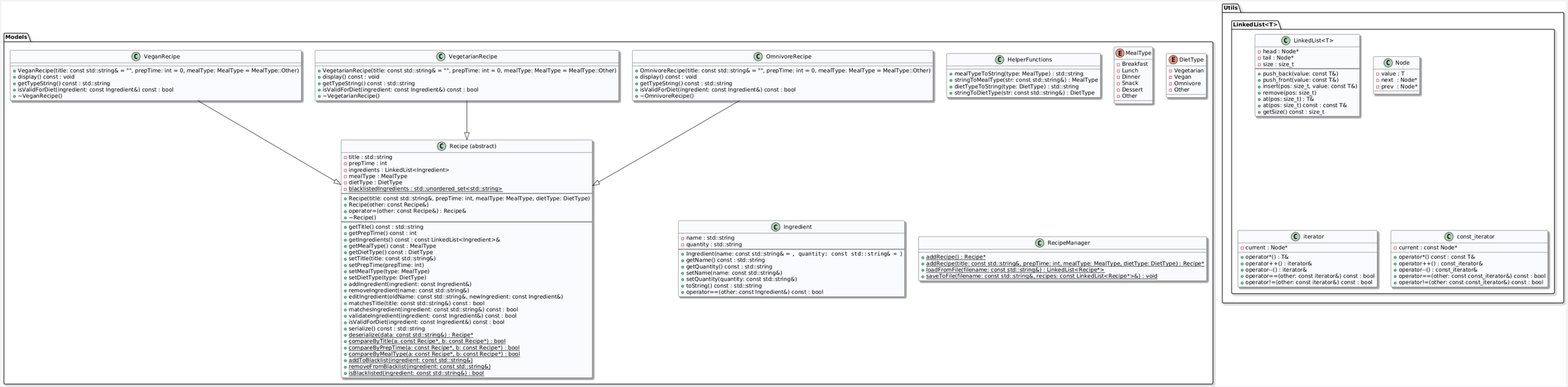
- Supports three recipe types: **Omnivore**, **Vegetarian**, **Vegan**
- Menu-driven CLI (in `main.cpp`)
- Load & save from file (`RecipeManager::loadFromFile, saveToFile`)

Development Stages

How we built it in phases

1. Phase 1: UML & class interfaces
2. Phase 2: Implement `LinkedList<T>` & templates
3. Phase 3: Recipe subclasses & `RecipeManager` logic
4. Phase 4: File I/O integration & persistence
5. Phase 5: Testing, CLI polish, edge-case handling

UML Diagram



Closer look in our Repo!

Key Features

Core OOP and data structure features we implemented

- Template-based `LinkedList<T>` for recipe storage
- Abstract base class `Recipe` → derived `VeganRecipe`, `VegetarianRecipe`, `OmnivoreRecipe`
- File I/O built into `RecipeManager`
- **Sort:** `LinkedList::sort()` alphabetical
- **Search:** `RecipeManager::filterByIngredient()`

Code Highlights

Deep dive into key Recipe class features

```
1 // Fa
2 Recip
  dietT
3     s
4
5
6
7
8
9
10
11
12     }
13 }
```

```
1  template <typename T>
2  class LinkedList {
3  private:
4      struct Node {
5          T value;
6          Node* next;
7          Node* prev;
8          Node(const T& val, Node* n = nullptr, Node* p = nullptr)
9              : value(val), next(n), prev(p) {}
10     };
11     Node* head = nullptr;
12     Node* tail = nullptr;
13
14 public:
15     LinkedList() = default;
16
17     void push_back(const T& val) {
18         Node* node = new Node(val, nullptr, tail);
19         if (!head) head = tail = node;
20         else {
21             tail->next = node;
22             tail = node;
23         }
24     }
25
26     // iterator, clear(), size(), etc. omitted
27 };
```

```
Type
```

Challenges & Lessons

How we built it in phases

- Template Pitfalls: linker errors when separating `.h/.tpp`
- Memory Management: avoiding leaks in `LinkedList` destructor
- Polymorphic I/O: casting back to derived types for file save
- CLI UX: intuitive menu vs. too many options

50+ Recipes

Let’s Demo

What's Next

Feature roadmap and future improvements

- GUI (i.e., web front-end)
- Nutrition & meal-planning module
- User accounts + dietary profiles
- Bulk import/export (JSON/CSV)

Thank You, Questions? Feedback welcome.

Save our Repo!