DIGITAL MEAL PLAN | CORE | Symfony Webservice with Basic VC

Definition of Done

A user story is considered **Done** when all of the following criteria are met:

- All acceptance criteria for the user story are fully implemented and verified
- Symfony code follows framework conventions and best practices
- Controllers are properly structured with appropriate error handling
- Routes are correctly configured and tested
- Templates render without errors and display dynamic content
- Services are properly injected and utilized
- Code follows PSR-12 coding standards and includes meaningful comments
- The application runs without PHP errors or warnings
- All routes are tested with valid and invalid parameters

User Story 1: Symfony Installation

As a WEB DEVELOPER I want Symfony (PHP Framework) installed on a local webserver (on my personal laptop), so that I can develop the meal plan system locally.

Acceptance Criteria

- The most current version (stable) of Symfony is ready to use locally
- Setup has been performed according to https://symfony.com/doc/current/setup.html
- A Demo Website is available, featuring the Catering4Schools GmbH's logo and name

User Story 2: Current Menu Display

As a STUDENT, I want to be able to see the current menu, so that I can decide what meals I want to pre-order.

Acceptance Criteria

- A single page rendering a static overview for the current menu is available
- TWIG is used for the View
- The route name is /menu/current (later to be the variable menu id)
- A menu entry consists of the following: meal name, allergens, nutritional information, date
- Presentation takes the form of a simple table without special CSS
- The detail view serves mock data defined in the controller

User Story 3: Comprehensive Menu Overview

As a STUDENT, I want to be able to see all available menus, so that I can get a comprehensive overview of meal options.

Acceptance Criteria

- A route for showing all available menus exists

- The user can navigate to the detail view for a menu and back to the overview
- The detail view serves mock data defined in the controller
- The mock data includes the menu id which is passed via the route