# 1. Introduction and Summary of project

The Smart Meal Planner is a. NET application designed under Windows Forms with C# for pantry management and meal planning. The app is intended to assist users in organizing the contents of their pantry, and creating personalized as well as organized meal plans. This app is a young, fresh take on managing your kitchen; a practical lifestyle application with solid software technology behind it, complemented by comprehensive features to take care of the meal planning and ingredient tracking challenges that one faces at home.

At its core, the Smart Meal Planner is about bridging the gap between what's in your home and getting you to what you can cook within a few days. The app also seeks to solve the problems related to: less wastage of food by using what is available already, finding new recipes that match what is available resource-wise, managing stock in the pantry and planning a balanced meal over a few days. The platform offers its users one convenient location to search for recipes, manage their shopping list and generate automatic meal plans based on their preferred meal types with available ingredients.

The program provides you with five main windows, each with a specific purpose. The Recipe Detail page provides a detailed description of each recipe including the ingredients used, how to cook and serve it and images. The Recipe Research page gives users the ability to search recipes by keyword in titles, ingredients and preparation time. The Ingredient Management Page allows users to see what's already in their pantry, add new ingredients, update name and quantity, and have a grasp on available resources. The Meal Planner page is its marquee feature, creating smart meal plans based on what you have in the pantry and takes into consideration user dietary preferences (for example, vegan), cuisine type and threshold for missing ingredients. Those are links to the content pages, so what you've got there is normal behavior; that's how those Welcome pages are intended to work.

# 2. Development approach

### 2.1 Backend

The back-end of the smart meal planning system is developed by SmartMealPlanner.Core Project The file also makes reference to vEDM module as the Core project has a modular and layered architecture. It is the part of the system that defines the data model, specification and interaction between storage and services. It's the core of the whole project.

The model layer is composed of the primary entities that you will work with in your system: Recipe, Ingredient, Pantry and UserPreference. These classes are the basic data structure used in the system, which store recipe ingredients as well as ingredient amount and user preferences.

The interface layer lets you define abstractions (IRecipeRepository, IPantryRepository, IRecipeService etc) to normalize the behavior of repository and service layers. Such an abstract design ensures the flexibility and testability of the system. In the future, modifying the source of data or extending it will not impact our upper layer logic.

The repository layer contains implementation centric classes like JsonRecipeRepository and the JsonPantryRepository, that are responsible for reading and writing JSON files (e.g. recipes. json and pantry. json). This layer uses a System. Text. Json to serialize and deserialize objects for a very lightweight and readable storage.

The layer we deal with in business is the service one. RecipeService - searching for and filtering recipes, as well as recommending according to users' taste preferences PantryService - stores all the workflow data on ingredients PlannerService – creates daily meal plans

All services are developed with async programming (async await) for not blocking the system when files read/write. The advantage of the modular approach is that the backend can be easily extended to other sources of data (ex.: databases or external APIs) in the near future.

### 2.2 Frontend

The Smart Meal Planner application front-end is designed in Windows Forms (.. Net8. 0). The project utilizes the UseWindowsForm property of the project configuration, and there you have full access toWindows Forms control library.

Both forms are similar in structure -UI creation (BuildUI(), InitializeComponent()), loading of some data (LoadRecipesAsync(), LoadPantryAsync()) and event handling. The forms are separate units that can manage their own init, service creation and data binding.

On the frontend is used quite a lot of event handling for user interaction. Event handlers are used to handle button clicks, form events and when a data grid is manipulated.

For data binding the application uses BindList collections, and the items are also bound for real-time editing. The method allows for UI controls and data models to be bound without having to manually synchronize the binding.

The user interface is designed by using a rich set of Windows Forms controls: DataGridView for data display, ComboBox for selection and NumericUpDown for numeric input; in addition are TextBox components (text input) and PictureBox objects used to display images. ReadOnly, DropDownStyle and SelectionMode properties are also set for each control.

The app controls the forms by passing WelcomeForm in as owner windows and other forms are displayed conservatively using Show(). The navigation system makes the other windows will always be closed that welcome page, as it is.

There are MessageBox dialogues for important messages, errors and not enough ingredient warning that give you instant visual feedback about what the user just did.

## 3. Flowchart

The below is the files flow chart and how they all are integrated:

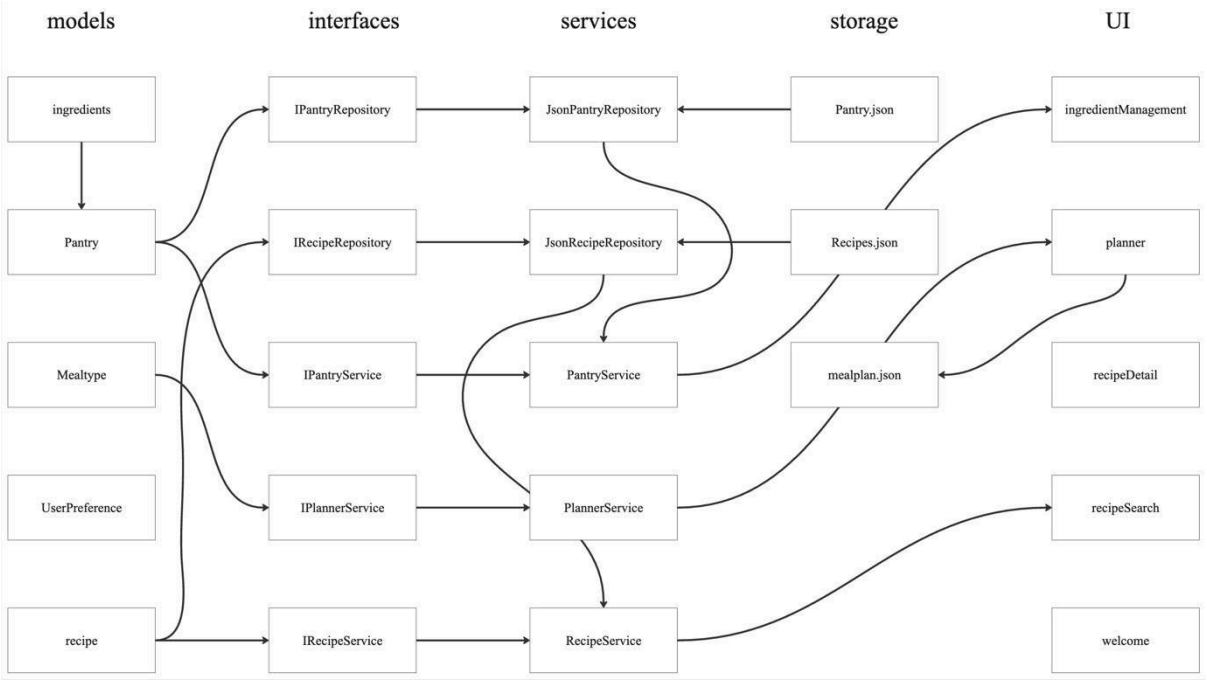


Figure 1. Flow chart of files for the application

The models ingredients, pantry, recipe are located in src/SmartMealPlanner. core/models/ for defining base definitions. The interfaces you can find in src/SmartMealPlanner. core/interfaces/, and service implementation structure. The services do important work for the application, and both JsonPantryRepository and JsonRecipeRepository (in src/SmartMealPlanner. core/repositories/) contains logic for writing/reading storage files Pantry. json and Recipe. json, and PantryService, PlannerService and RecipeService (all in src/SmartMealPlanner. core/services/) -including functions like ApplyCookingAsync(), AssignAsync() or GerRecommendationsAsync()- include essential functionalities. All the UI stuff is in ui/, with individual files and planner will dump the meal plan that it generates to mealplan. json.

## 4. Role of team members

| Name | Email | Student ID | Role |
|---|---|---|---|
| Zhijian Deng | Zhijian.deng@student.uts.edu.au | 24683334 | Frontend designer |
| Shuoyan Xia | Shuoyan.Xia-1@student.uts.edu.au | 24914119 | Backend designer |

Table 1. Table of information and roles of team members

### 4.1 Zhijian Deng

As a frontend programmer Smart Meal Planner, I am responsible for creating and implementing the ui interface of this project, in particular such ui/ components. My role is creating a desktop app that is turning my product's core business logic into a user friendly meal plan.

I created five different classes as Windows Forms, all of them combined providing full front-end architecture. The WelcomeForm would act as a central navigation gateway that is responsible to handle forms and windows relationship management. RecipeSearchForm features advanced search with dynamic filter, async data loading and an embedded DataGridView for the recipe details. IngredientManagementForm offers feature rich management of your pantry via the magic of real time data binding. RecipeDetailForm displays all the information of a recipe, including an image and cooking instructions. GR_AF This PlannerForm combines different groups of user preference to the meal planning algorithms which are intelligent by itself including a data load and asynchronous event handling.

I also design the architecture of the entire project. To develop the frontend, I've also had regular talks with the backend-designer to grasp all the main functions. I wrote section 1, 2.2, 3 and 4.1.

**4.2 Shuoyan Xia**

I was the backend architect of the project and most of our system core logic is developed within SmartMealPlanner. Primary contribution, main project framework, such as the backend service modules and communicating between frontend and backend.

I created some of the fundamental service classes like RecipeService, PantryService and PlannerService. RecipeServiceIt is the heart of the component which has ingredient based recipe matching algorithm, cuisine weight calculation and dietary tag scoring that allows the system to recommend intelligent recipes based on user preferences and availability of ingredients.

I also created JSON repositories (JsonRecipeRepository and JsonPantryRepository) as a means to persist data, so the system could manage nonvolatile data which is passed between destination view controllers. This model is light and easy to clean and move around.

I collaborated with front-end designers to define the interface between the controls used in the user-interface and services provided by the back-end, so that data flow and logic are smooth. I rigorously tested features such as filtering ingredients that I disliked, identifying dietary preferences and accounting for ingredients not in their recipes through countless iterations.