Interactive Intelligent Kitchen Helper

CAU-IIKH

20182705 고주형 20185784 김호성
20182610 손희승 20162874 이준협
20142611 이하람

What "IIKH" is

IIKH

Recipe

- Recipe Name
- Description of Recipe
- Needed Ingredients

Management

Plan

- Plan Name
- Meal Plan of
  Breakfast
  Lunch
  Dinner

"Convenience"
of Kitchen User

Steak Recipe
Cream Pasta Recipe
Lamb Curry Recipe
Fried Chicken Recipe
Curry Recipe
Garlic Sou
Onion Soup Recipe

```
**********  Here's Menu  **********

 >> Enter [1] to <SEARCH RECIPE>
 >> Enter [2] to <ADD RECIPE>
 >> Enter [3] to <DELETE RECIPE>
 >> Enter [4] to <VIEW RECIPE>
 >> Enter [5] to <SEARCH PLAN>
 >> Enter [6] to <ADD PLAN>
 >> Enter [7] to <DELETE PLAN>
 >> Enter [8] to <VIEW PLAN>
 >> Enter [9] to <QUIT>
```

What we focused on

"UX"

“UX”

Console…

Hard to type

Hard to edit after Entering

Hard to see what I have entered

Hard to do task at the same time.
- Adding Recipe & Editing Ingredient

Excel…

Easy to type

Easy to edit after Entering

Easy to see what I have entered

Easy to edit every cell at the same time

CSV

Comma Separated Values

```
id,name,released_on,price,created_at,updated_at
24,1000 Piece Jigsaw Puzzle,2012-07-03,14.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
30,360° Protractor,2012-05-03,3.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
17,7 Wonders,2012-04-21,28.75,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
13,Acoustic Guitar,2012-06-06,1025.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
15,Agricola,2012-05-22,45.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
22,Answer to Everything,2012-07-03,42.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
23,Box Kite,2012-05-19,63.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
29,CanCan Music Record,2012-05-09,2.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
12,Chocolate Pie,2012-04-12,3.14,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
9,Dog Toy Bone,2012-06-13,2.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
11,Flux Capacitor,2012-06-01,19.55,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
6,Game Console,2012-06-06,299.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
10,Heated Blanket,2012-07-19,27.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
19,Knights of Catan,2012-06-10,19.95,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
8,Lawn Chair,2012-05-29,34.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
21,Millennium Falcon,2012-04-10,3597200.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
14,Model Enterprise,2012-04-18,27.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
28,Model Train Rails,2012-06-30,45.0,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
3,Oak Coffee Table,2012-07-08,223.99,2012-07-09 16:50:49 UTC,2012-07-09 16:50:49 UTC
```

CSV

Recipe Database

IIKHRecipeDB.csv

C: > Users > duryk > Desktop > IIKH > IIKH > Release > IIKHRecipeDB.csv

```
Actually Delicious Turkey Burgers,1. In a large bowl mix ground turkey seasoned bread crumbs onion egg whites parsley garlic salt and pepper. Form int
Aussie Chicken,Step 1: Rub the chicken breasts with the seasoning salt cover and refrigerate for 30 minutes.    Step 2: Preheat oven to 350 degrees F
Baked Dijon Salmon,1. Preheat oven to 400 degrees F (200 degrees C).    2. In a small bowl stir together butter mustard and honey. Set aside. In anothe
Baked Spaghetti,1. Preheat oven to 350 degrees F (175 degrees C). Lightly grease a 9x13-inch baking dish.    2. Bring a large pot of lightly salted wat
Chicken Enchiladas II,Step 1: Preheat oven to 350 degrees F (175 degrees C). Lightly grease a large baking dish.    Step 2: In a medium saucepan over
Corned Beef and Cabbage I,1. Place corned beef in large pot or Dutch oven and cover with water. Add the spice packet that came with the corned beef. C
Curry Stand Chicken Tikka Masala Sauce,Step 1: Heat ghee in a large skillet over medium heat and cook and stir onion until translucent about 5 minutes
Easy Tuna Patties,Beat eggs and lemon juice in a bowl; stir in Parmesan cheese and bread crumbs to make a paste. 1. Fold in tuna and onion until well-
Grilled Salmon I,1. Season salmon fillets with lemon pepper garlic powder and salt.   2. In a small bowl stir together soy sauce brown sugar water and
Marinated Tuna Steak,1. In a large non-reactive dish mix together the orange juice soy sauce olive oil lemon juice parsley garlic oregano and pepper.
Meatball Sandwich,1. Preheat the oven to 350 degrees F (175 degrees C).    2. In a medium bowl gently mix by hand the ground beef bread crumbs Italian
Philly Cheesesteak Sandwich with Garlic Mayo,1. In a small bowl combine mayonnaise and minced garlic. Cover and refrigerate. Preheat oven to 500 degre
Roasted Rack of Lamb,1. Preheat oven to 450 degrees F (230 degrees C). Move oven rack to the center position.    2. In a large bowl combine bread crum
Salsa Chicken,Step 1: Preheat oven to 375 degrees F (190 degrees C)    Step 2: Place chicken breasts in a lightly greased 9x13 inch baking dish. Sprin
Sausage Pasta,1. Bring a large pot of lightly salted water to a boil. Add pasta and cook for 8 to 10 minutes or until al dente; drain and reserve.
Spicy Garlic Lime Chicken,Step 1: In a small bowl mix together salt black pepper cayenne paprika 1/4 teaspoon garlic powder onion powder thyme and par
Suki's Spinach and Feta Pasta,1. Bring a large pot of lightly salted water to a boil. Cook pasta in boiling water until al dente; drain.    2. Meanwhi
Turkey Pot Pie I,1. Preheat oven to 425 degrees F (220 degrees C). Roll out bottom pie crust press into a 10 inch pie pan and set aside.    2. Melt 2
```

Plan Database

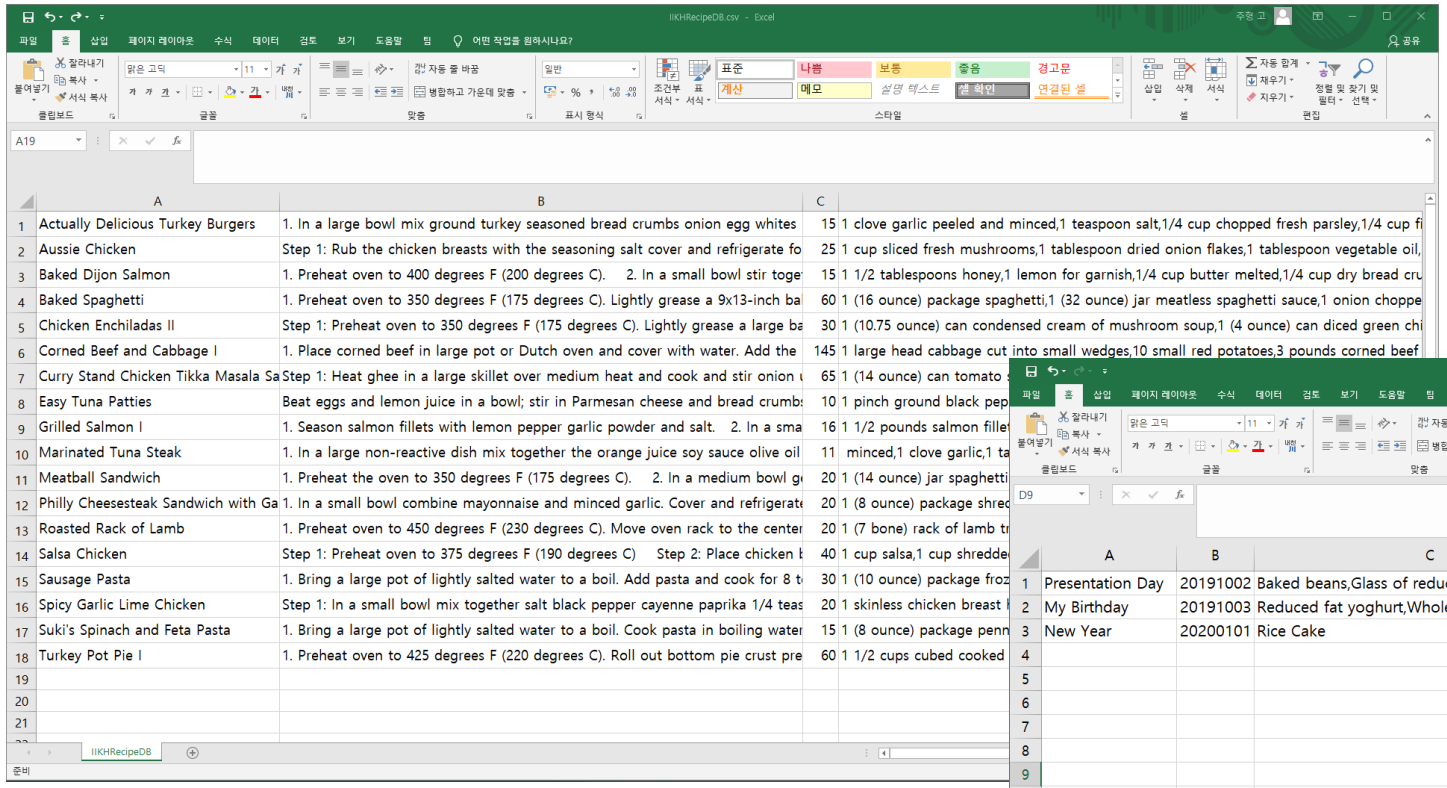
IIKHPlanDB.csv

C: > Users > duryk > Desktop > IIKH > IIKH > Release > IIKHPlanDB.csv

```
Presentation Day,20191002,"Baked beans,Glass of reduced milk,Tomato,Wholegrain toast with polyunsaturated margarine","Coffee with milk,Wholegrain sandw
My Birthday,20191003,"Reduced fat yoghurt,Wholegrain breakfast cereal","Apple,Sandwich with salad and chicken","Cake,Pasta with beef mince and red kidn
New Year,20200101,Rice Cake,Rice Cake,Rice Cake
```

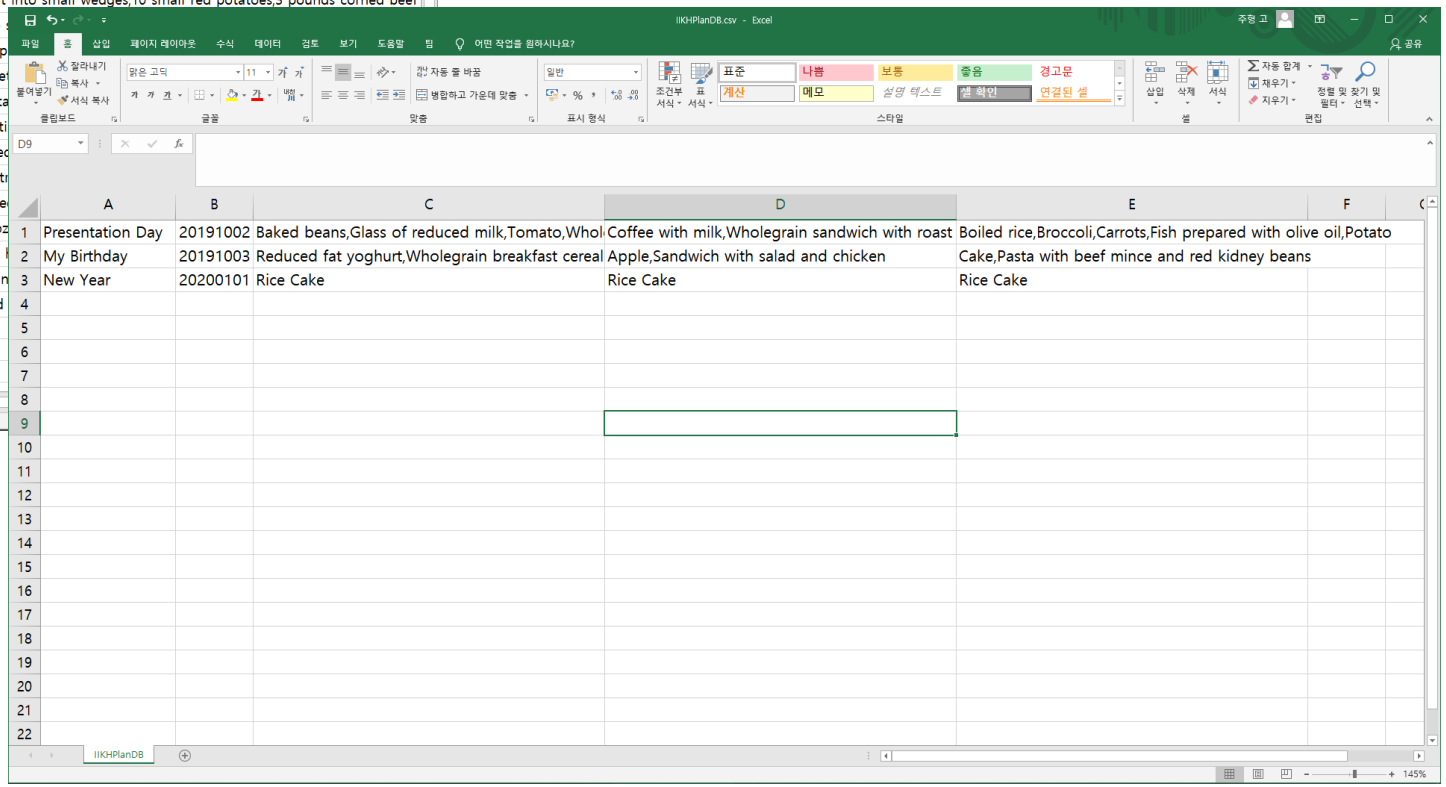
CSV in Excel

Recipe Database



	A	B	C	
1	Actually Delicious Turkey Burgers	1. In a large bowl mix ground turkey seasoned bread crumbs onion egg whites	15	1 clove garlic peeled and minced,1 teaspoon salt,1/4 cup chopped fresh parsley,1/4 cup fi
2	Aussie Chicken	Step 1: Rub the chicken breasts with the seasoning salt cover and refrigerate fo	25	1 cup sliced fresh mushrooms,1 tablespoon dried onion flakes,1 tablespoon vegetable oil,
3	Baked Dijon Salmon	1. Preheat oven to 400 degrees F (200 degrees C). 2. In a small bowl stir toge	15	1 1/2 tablespoons honey,1 lemon for garnish,1/4 cup butter melted,1/4 cup dry bread cru
4	Baked Spaghetti	1. Preheat oven to 350 degrees F (175 degrees C). Lightly grease a 9x13-inch ba	60	1 (16 ounce) package spaghetti,1 (32 ounce) jar meatless spaghetti sauce,1 onion choppe
5	Chicken Enchiladas II	Step 1: Preheat oven to 350 degrees F (175 degrees C). Lightly grease a large ba	30	1 (10.75 ounce) can condensed cream of mushroom soup,1 (4 ounce) can diced green chi
6	Corned Beef and Cabbage I	1. Place corned beef in large pot or Dutch oven and cover with water. Add the	145	1 large head cabbage cut into small wedges,10 small red potatoes,3 pounds corned beef
7	Curry Stand Chicken Tikka Masala Sa	Step 1: Heat ghee in a large skillet over medium heat and cook and stir onion	65	1 (14 ounce) can tomato
8	Easy Tuna Patties	Beat eggs and lemon juice in a bowl; stir in Parmesan cheese and bread crumbs	10	1 pinch ground black pep
9	Grilled Salmon I	1. Season salmon fillets with lemon pepper garlic powder and salt. 2. In a sma	16	1 1/2 pounds salmon fille
10	Marinated Tuna Steak	1. In a large non-reactive dish mix together the orange juice soy sauce olive oil	11	minced,1 clove garlic,1 ta
11	Meatball Sandwich	1. Preheat the oven to 350 degrees F (175 degrees C). 2. In a medium bowl g	20	1 (14 ounce) jar spaghetti
12	Philly Cheesesteak Sandwich with Ga	1. In a small bowl combine mayonnaise and minced garlic. Cover and refrigerate	20	1 (8 ounce) package shred
13	Roasted Rack of Lamb	1. Preheat oven to 450 degrees F (230 degrees C). Move oven rack to the center	20	1 (7 bone) rack of lamb tr
14	Salsa Chicken	Step 1: Preheat oven to 375 degrees F (190 degrees C) Step 2: Place chicken b	40	1 cup salsa,1 cup shredde
15	Sausage Pasta	1. Bring a large pot of lightly salted water to a boil. Add pasta and cook for 8 t	30	1 (10 ounce) package froz
16	Spicy Garlic Lime Chicken	Step 1: In a small bowl mix together salt black pepper cayenne paprika 1/4 teas	20	1 skinless chicken breast
17	Suki's Spinach and Feta Pasta	1. Bring a large pot of lightly salted water to a boil. Cook pasta in boiling water	15	1 (8 ounce) package penn
18	Turkey Pot Pie I	1. Preheat oven to 425 degrees F (220 degrees C). Roll out bottom pie crust pre	60	1 1/2 cups cubed cooked
19				
20				
21				

Plan Database



	A	B	C	D	E	F
1	Presentation Day	20191002	Baked beans,Glass of reduced milk,Tomato,Whol	Coffee with milk,Wholegrain sandwich with roast	Boiled rice,Broccoli,Carrots,Fish prepared with olive oil,Potato	
2	My Birthday	20191003	Reduced fat yoghurt,Wholegrain breakfast cereal	Apple,Sandwich with salad and chicken	Cake,Pasta with beef mince and red kidney beans	
3	New Year	20200101	Rice Cake	Rice Cake	Rice Cake	
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						

Process of IIKH Project

What is our Task?

Specification was ambiguous

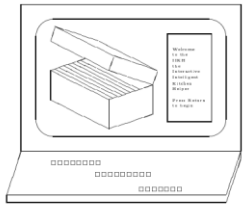


Figure 3.1: – View of the Interactive Intelligent Kitchen Helper.

design techniques. We will illustrate the application of Responsibility-Driven Design with a case study.

3.4 A Case Study in RDD

Imagine you are the chief software architect in a major computer firm. One day your boss walks into your office with an idea that, it is hoped, will be the next major success in your product line. Your assignment is to develop the *Interactive Intelligent Kitchen Helper* (Figure 3.1).

The task given to your software team is stated in very few words (written on what appears to be the back of a slightly-used dinner napkin, in handwriting that appears to be your boss's).

3.4.1 The Interactive Intelligent Kitchen Helper

Briefly, the Interactive Intelligent Kitchen Helper (IIKH) is a PC-based application that will replace the index-card system of recipes found in the average kitchen. But more than simply maintaining a database of recipes, the kitchen helper assists in the planning of meals for an extended period, say a week. The

user of the IIKH can sit down at a terminal, browse the database of recipes, and interactively create a series of menus. The IIKH will automatically scale the recipes to any number of servings and will print out menus for the entire week, for a particular day, or for a particular meal. And it will print an integrated grocery list of all the items needed for the recipes for the entire period.

As is usually true with the initial descriptions of most software systems, the specification for the IIKH is highly ambiguous on a number of important points. It is also true that, in all likelihood, the eventual design and development of the software system to support the IIKH will require the efforts of several programmers working together. Thus, the initial goal of the design team must be to clarify the ambiguities in the description and to outline how the project can be divided into components to be assigned for development to individual team members.

The fundamental cornerstone of object-oriented programming is to characterize software in terms of *behavior*; that is, actions to be performed. We will see this repeated on many levels in the development of the IIKH. Initially, the team will try to characterize, at a very high level of abstraction, the behavior of the entire application. This then leads to a description of the behavior of various software subsystems. Only when all behavior has been identified and described will the software design team proceed to the coding step. In the next several sections we will trace the tasks the software design team will perform in producing this application.

3.4.2 Working through Scenarios

The first task is to refine the specification. As we have already noted, initial specifications are almost always ambiguous and unclear on anything except the most general points. There are several goals for this step. One objective is to get a better handle on the "look and feel" of the eventual product. This information can then be carried back to the client (in this case, your boss) to see if it is in agreement with the original conception. It is likely, perhaps inevitable, that the specifications for the final application will change during the creation of the software system, and it is important that the design be developed to easily accommodate change and that potential changes be noted as early as possible. Equally important, at this point very high level decisions can be made concerning the structure of the eventual software system. In particular, the activities to be performed can be mapped onto components.

In order to uncover the fundamental behavior of the system, the design team first creates a number of *scenarios*. That is, the team acts out the running of the application just as if it already possessed a working system. An example scenario is shown in Figure 3.2.

Simple Browsing

Alice Smith sits down at her computer and starts the IIKH. When the program begins, it displays a graphical image of a recipe box, and identifies itself as the IIKH, product of IIKH incorporated. Alice presses the return button to begin.

In response to the key press, Alice is given a choice of a number of options. She elects to browse the recipe index, looking for a recipe for Salmon that she wishes to prepare for dinner the next day. She enters the keyword Salmon, and is shown in response a list of various recipes. She remembers seeing an interesting recipe that used dill-weed as a flavoring. She refines the search, entering the words Salmon and dill-weed. This narrows the search to two recipes.

She selects the first. This brings up a new window in which an attractive picture of the finished dish is displayed, along with the list of ingredients, preparation steps, and expected preparation time. After examining the recipe, Alice decides it is not the recipe she had in mind. She returns to the search result page, and selects the second alternative.

Examining this dish, Alice decides this is the one she had in mind. She requests a printing of the recipe, and the output is spooled to her printer. Alice selects "quit" from a program menu, and the application quits.

Figure 3.2: An Example Scenario

3.4.3 Identification of Components

The engineering of a complex physical system, such as a building or an automobile engine, is simplified by dividing the design into smaller units. So, too, the engineering of software is simplified by the identification and development of software components. A *component* is simply an abstract entity that can perform tasks–that is, fulfill some responsibilities. At this point, it is not necessary to know exactly the eventual representation for a component or how a component will perform a task. A component may ultimately be turned into a function, a structure or class, or a collection of other components. At this level of development there are just two important characteristics:

- A component must have a small well-defined set of responsibilities.
- A component should interact with other components to the minimal extent possible.

We will shortly discuss the reasoning behind the second characteristic. For the moment we are simply concerned with the identification of component responsibilities.

3.5 CRC Cards–Recording Responsibility

As the design team walks through the various scenarios they have created, they identify the components that will be performing certain tasks. Every activity that must take place is identified and assigned to some component as a responsibility.

Component Name	Collaborators
Description of the responsibilities assigned to this component	*List of other components*

As part of this process, it is often useful to represent components using small index cards. Written on the face of the card is the name of the software component, the responsibilities of the component, and the names of other components with which the component must interact. Such cards are sometimes known as CRC (Component, Responsibility, Collaborator) cards, and are associated with each software component. As responsibilities for the component are discovered, they are recorded on the face of the CRC card.

3.5.1 Give Components a Physical Representation

While working through scenarios, it is useful to assign CRC cards to different members of the design team. The member holding the card representing a component records the responsibilities of the associated software component, and acts as the "surrogate" for the software during the scenario simulation. He or she describes the activities of the software system, passing "control" to another member when the software system requires the services of another component.

An advantage of CRC cards is that they are widely available, inexpensive, and erasable. This encourages experimentation, since alternative designs can be tried, explored, or abandoned with little investment. The physical separation of the cards encourages an intuitive understanding of the importance of the logical separation of the various components, helping to emphasize the cohesion and coupling (which we will describe shortly). The constraints of an index card are also a good measure of approximate complexity–a component that is expected to perform more tasks than can fit easily in this space is probably too complex, and the team should find a simpler solution, perhaps by moving some responsibilities elsewhere to divide a task between two or more new components.

3.5.2 The What/Who Cycle

As we noted at the beginning of this discussion, the identification of components takes place during the process of imagining the execution of a working system. Often this proceeds as a cycle of what/who questions. First, the design team identifies *what* activity needs to be performed next. This is immediately followed by answering the question of *who* performs the action. In this manner, designing a software system is much like organizing a collection of people, such as a club. Any activity that is to be performed must be assigned as a responsibility to some component.

A popular bumper sticker states that phenomena can and will spontaneously occur. (The bumper sticker uses a slightly shorter phrase.) We know, however, that in real life this is seldom true. If any action is to take place, there must be an agent assigned to perform it. Just as in the running of a club any action to be performed must be assigned to some individual, in organizing an object-oriented program all actions must be the responsibility of some component. The secret to good object-oriented design is to first establish an agent for each action.

3.5.3 Documentation

At this point the development of documentation should begin. Two documents should be essential parts of any software system: the user manual and the system design documentation. Work on both of these can commence even before the first line of code has been written.

The user manual describes the interaction with the system from the user's point of view; it is an excellent means of verifying that the development team's conception of the application matches the client's. Since the decisions made in creating the scenarios will closely match the decisions the user will be required to make in the eventual application, the development of the user manual naturally dovetails with the process of walking through scenarios.

Before any actual code has been written, the mindset of the software team is most similar to that of the eventual users. Thus, it is at this point that the developers can most easily anticipate the sort of questions to which a novice user will need answers. A user manual is also an excellent tool to verify that the programming team is looking at the problem in the same way that the client intended. A client seldom presents the programming team with a complete and formal specification, and thus some reassurance and two-way communication early in the process, before actual programming has begun, can prevent major misunderstandings.

The second essential document is the design documentation. The design documentation records the major decisions made during software design, and should thus be produced when these decisions are fresh in the minds of the creators, and not after the fact when many of the relevant details will have been forgotten. It is often far easier to write a general global description of the software system early in the development. Too soon, the focus will move to the level of individual components or modules. While it is also important to document the module level, too much concern with the details of each module will make it difficult for subsequent software maintainers to form an initial picture of the larger structure.

CRC cards are one aspect of the design documentation, but many other important decisions are not reflected in them. Arguments for and against any major design alternatives should be recorded, as well as factors that influenced the final decisions. A log or diary of the project schedule should be maintained. Both the user manual and the design documents are refined and evolve over time in exactly the same way the software is refined and evolves.

3.6 Components and Behavior

To return to the IIKH, the team decides that when the system begins, the user will be presented with an attractive informative window (see Figure 3.1). The responsibility for displaying this window is assigned to a component called the Greeter. In some as yet unspecified manner (perhaps by pull-down menus, button

Greeter	Collaborators
Display Informative Initial Message	Database Manager
Offer User Choice of Options	Plan Manager
Pass Control to either	
Recipe Database Manager	
Plan Manager for processing	

Figure 3.3: – CRC card for the Greeter.

or key presses, or use of a pressure-sensitive screen), the user can select one of several actions. Initially, the team identifies just five actions:

1. Casually browse the database of existing recipes, but without reference to any particular meal plan.
2. Add a new recipe to the database.
3. Edit or annotate an existing recipe.
4. Review an existing plan for several meals.
5. Create a new plan of meals.

These activities seem to divide themselves naturally into two groups. The first three are associated with the recipe database; the latter two are associated with menu plans. As a result, the team next decides to create components corresponding to these two responsibilities. Continuing with the scenario, the team elects to ignore the meal plan management for the moment and move on to refine the activities of the Recipe Database component. Figure 3.3 shows the initial CRC card representation of the Greeter.

Broadly speaking, the responsibility of the recipe database component is simply to maintain a collection of recipes. We have already identified three elements of this task: The recipe component database must facilitate browsing the library of existing recipes, editing the recipes, and including new recipes in the database.

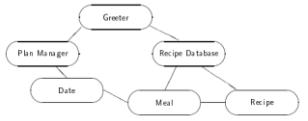


Figure 3.4: – Communication between the six components in the IIKH.

specific meal. In the latter case, control is passed to a Meal component.

The Meal component maintains a collection of augmented recipes, where the augmentation refers to the user's desire to double, triple, or otherwise increase a recipe. The Meal component displays information about the meal. The user can add or remove recipes from the meal, or can instruct that information about the meal be printed. In order to discover new recipes, the user must be permitted at this point to browse the recipe database. Thus, the Meal component must interact with the recipe database component. The design team will continue in this fashion, investigating every possible scenario. The major category of scenarios we have not developed here is exceptional cases. For example, what happens if a user selects a number of keywords for a recipe and no matching recipe is found? How can the user cancel an activity, such as entering a new recipe, if he or she decides not to continue? Each possibility must be explored, and the responsibilities for handling the situation assigned to one or more components.

Having walked through the various scenarios, the software design team eventually decides that all activities can be adequately handled by six components (Figure 3.4). The Greeter needs to communicate only with the Plan Manager and the Recipe Database components. The Plan Manager needs to communicate only with the Date component; and the Date agent, only with the Meal component. The Meal component communicates with the Recipe Manager and, through this agent, with individual recipes.

3.6.4 Interaction Diagrams

While a description such as that shown in Figure 3.4 may describe the static relationships between components, it is not very good for describing their dynamic interactions during the execution of a scenario. A better tool for this purpose is an *interaction diagram*. Figure 3.5 shows the beginning of an interaction diagram for the interactive kitchen helper. In the diagram, time moves forward from the

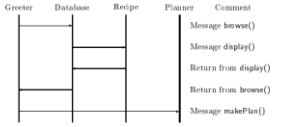


Figure 3.5: – An Example interaction diagram.

top to the bottom. Each component is represented by a labeled vertical line. A component sending a message to another component is represented by a horizontal arrow from one line to another. Similarly, a component returning control and perhaps a result value back to the caller is represented by an arrow. (Some authors use two different arrow forms, such as a solid line to represent message passing and a dashed line to represent returning control.) The commentary on the right side of the figure explains more fully the interaction taking place.

With a time axis, the interaction diagram is able to describe better the sequencing of events during a scenario. For this reason, interaction diagrams can be a useful documentation tool for complex software systems.

3.7 Software Components

In this section we will explore a software component in more detail. As is true of all but the most trivial ideas, there are many aspects to this seemingly simple concept.

3.7.1 Behavior and State

We have already seen how components are characterized by their behavior, that is, by what they can do. But components may also hold certain information. Let us take as our prototypical component a Recipe structure from the IIKH. One way to view such a component is as a pair consisting of *behavior* and *state*.

- The *behavior* of a component is the set of actions it can perform. The complete description of all the behavior for a component is sometimes

Date	Collaborators
Maintain information about specific date	Plan Manager
Date(year, month, day)–create new date	Meal
DisplayAndEdit()–display date information in window allowing user to edit entries	
BuildGroceryList(List &)–add items from all meals to grocery list	

Figure 3.6: – Revised CRC card for the Date component.

major portion of this process is designing the data structures that will be used by each subsystem to maintain the state information required to fulfill the assigned

heuristics used in this process.

If they were not determined earlier (say, as part of the specification of the system), then decisions can now be made on issues that are entirely self-contained within a single component. A decision we saw in our example problem was how best to let the user browse the database of recipes.

As multiperson programming projects become the norm, it becomes increasingly rare that any one programmer will work on all aspects of a system. More often, the skills a programmer will need to master are understanding how one section of code fits into a larger framework and working well with other members of a team.

Often, in the implementation of one component it will become clear that certain information or actions might be assigned to yet another component that will act "behind the scene," with little or no visibility to users of the software abstraction. Such components are sometimes known as *facilitators*. We will see examples of facilitators in some of the later case studies.

An important part of analysis and coding at this point is characterizing and documenting the necessary preconditions a software component requires to complete a task, and verifying that the software component will perform correctly when presented with legal input values.

3.11 Integration of Components

There is also an increasing emphasis on the reuse of general-purpose software components in multiple projects. For this to be successful, there must be minimal and well-understood interconnections between the various portions of the system. As we noted in the previous chapter, these ideas were captured by computer scientist David Parnas in a pair of rules, known as *Parnas's principles*:

- The developer of a software component must provide the intended user with all the information needed to make effective use of the services provided by the component, and should provide no other information.
- The developer of a software component must be provided with all the information necessary to carry out the given responsibilities assigned to the component, and should be provided with *no* other information.

A consequence of the separation of interface from implementation is that a programmer can experiment with several different implementations of the same structure without affecting other software components.

3.8 Formalize the Interface

We continue with the description of the IIKH development. In the next several steps the descriptions of the components will be refined. The first step in this process is to formalize the patterns and channels of communication.

considerable amount of time is spent finding just the right set of terms to describe the tasks performed and the objects manipulated. Far from being a barren and useless exercise, proper naming early in the design process greatly simplifies and facilitates later steps.

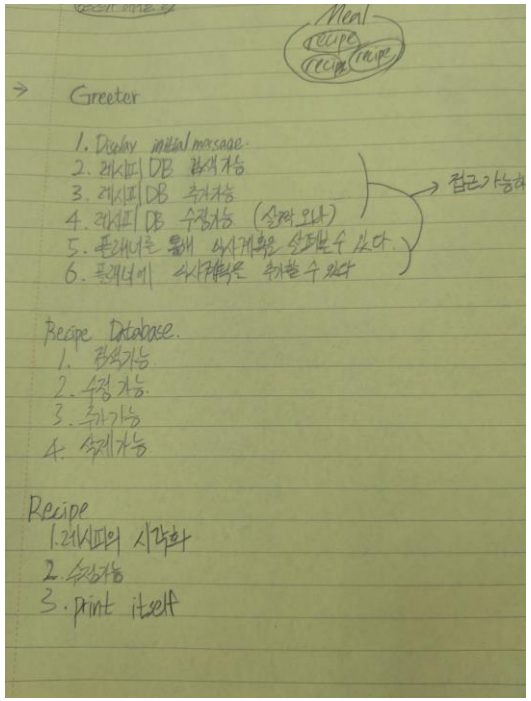
The following general guidelines have been suggested:

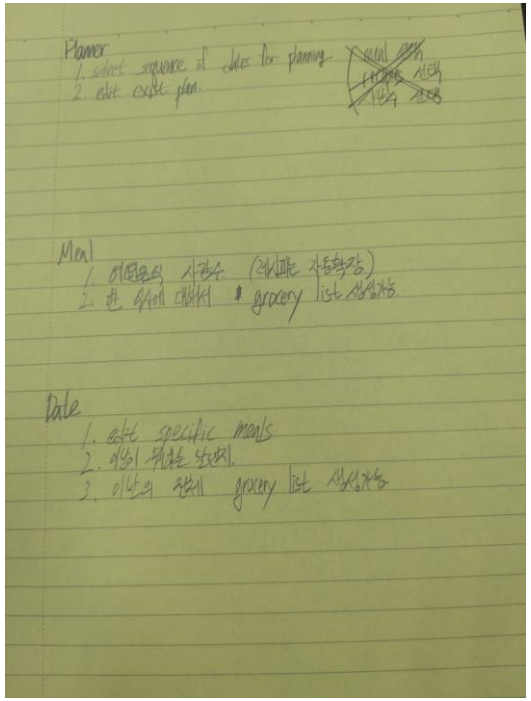
- Use pronounceable names. As a rule of thumb, if you cannot read a name out loud, it is not a good one.
- Use capitalization (or underscores) to mark the beginning of a new word within a name, such as "CardReader" or "Card_reader," rather than the less readable "cardreader."
- Examine abbreviations carefully. An abbreviation that is clear to one person may be confusing to the next. Is a "TermProcess" a terminal process, something that terminates processes, or a process associated with a terminal?
- Avoid names with several interpretations. Does the empty function tell whether something is empty, or empty the values from the object?
- Avoid digits within a name. They are easy to misread as letters (0 as O, 1 as l, 2 as Z, 5 as S).
- Name functions and variables that yield Boolean values so they describe

hence every action must be assigned to some member of the object community. Conversely, the behavior of the members of the community taken together must be sufficient to achieve the desired goal.

The emphasis on *behavior* is a hall-mark of object-oriented programming. Behavior can be identified in even the most rudimentary descriptions of a system, long before any other aspect can be clearly discerned. By constantly being driven by behavior, responsibility-driven design moves smoothly from problem description to software architecture to code development to finished application.

What we did





Refine Specification

IIKH 기획서

목적: 부엌에 쌓인 레시피 카드들을 없애자 + 식단 plan을 만들자??

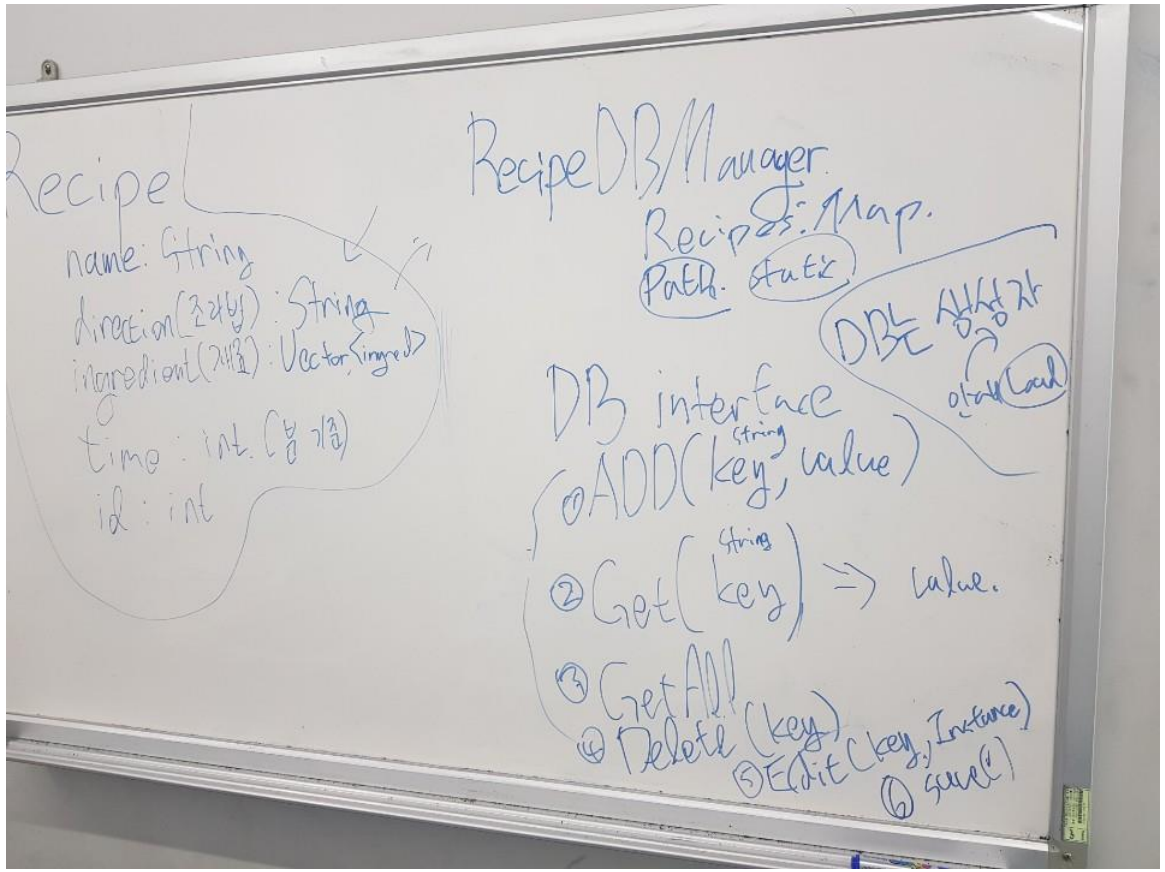
구현해야하는 것:

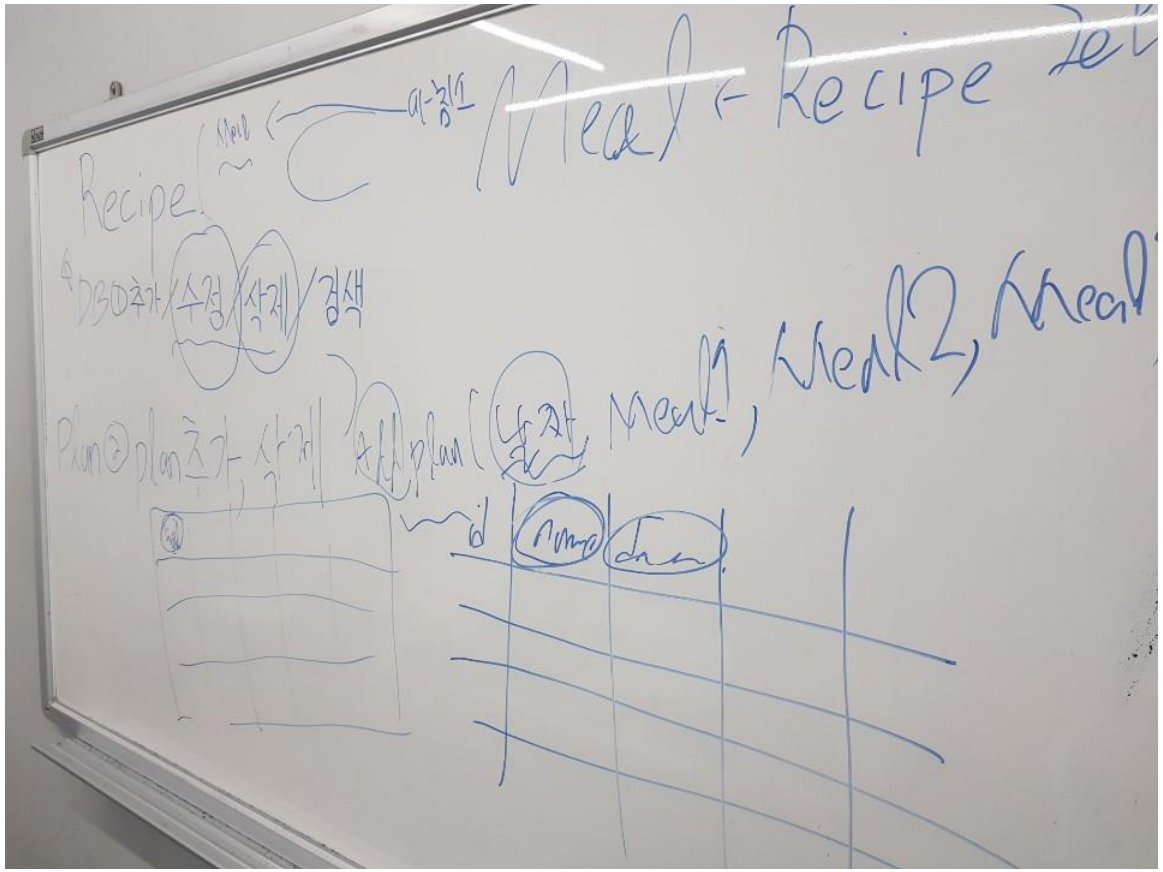
1. 초기화면: 레시피 박스, IIKH처럼 보이게 하기
   - 리턴 버튼 누르면 시작.
2. IIKH 시작화면: 여러가지 메뉴 선택 가능
   A. Recipe 전체 보기
      -> Ex) 이 메뉴를 열어 연어를 찾을 수 있어야 함.
      구체적으로, Salmon을 검색하면 여러가지 레시피들을 찾을 수 있어야함.
      (필터링 가능해야 됨) Ex) Salmon + Ketchup => 2개 다 들어간 레시피 가져와야 됨.
      레시피 선택 -> 새로운 창 with (Picture, 재료, 조리법(순서), 예상 시간).
   B. DB에 새로운 Recipe 추가
   C. DB에 존재하는 Recipe 수정
   D. Meal Or Recipe의 재료 출력
      == 밀(아침 점심 저녁) 선택 -> 포함 모든 식사 레시피에 필요한 재료 출력.
   E. Plan 추가
      == 식사(meal: 아침 점심 저녁) 정보 계획하기(날짜 선택 후, 밀(아침 점심 저녁) 입력).
   F. Plan 출력
      == 식사 계획 살펴보기(PlanManager).
   G. Quit 메뉴 선택
      -> 프로그램 종료.

설명:

What we did

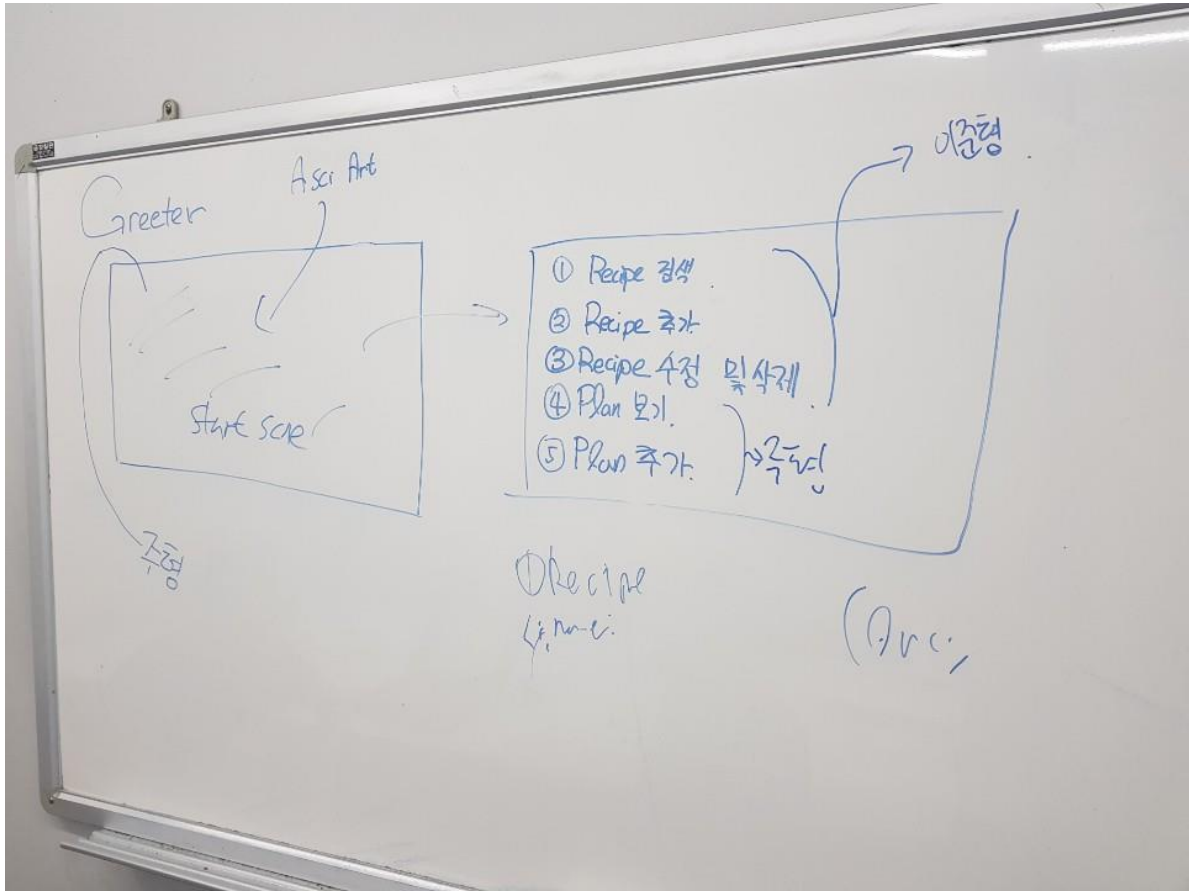
Identifying the Components



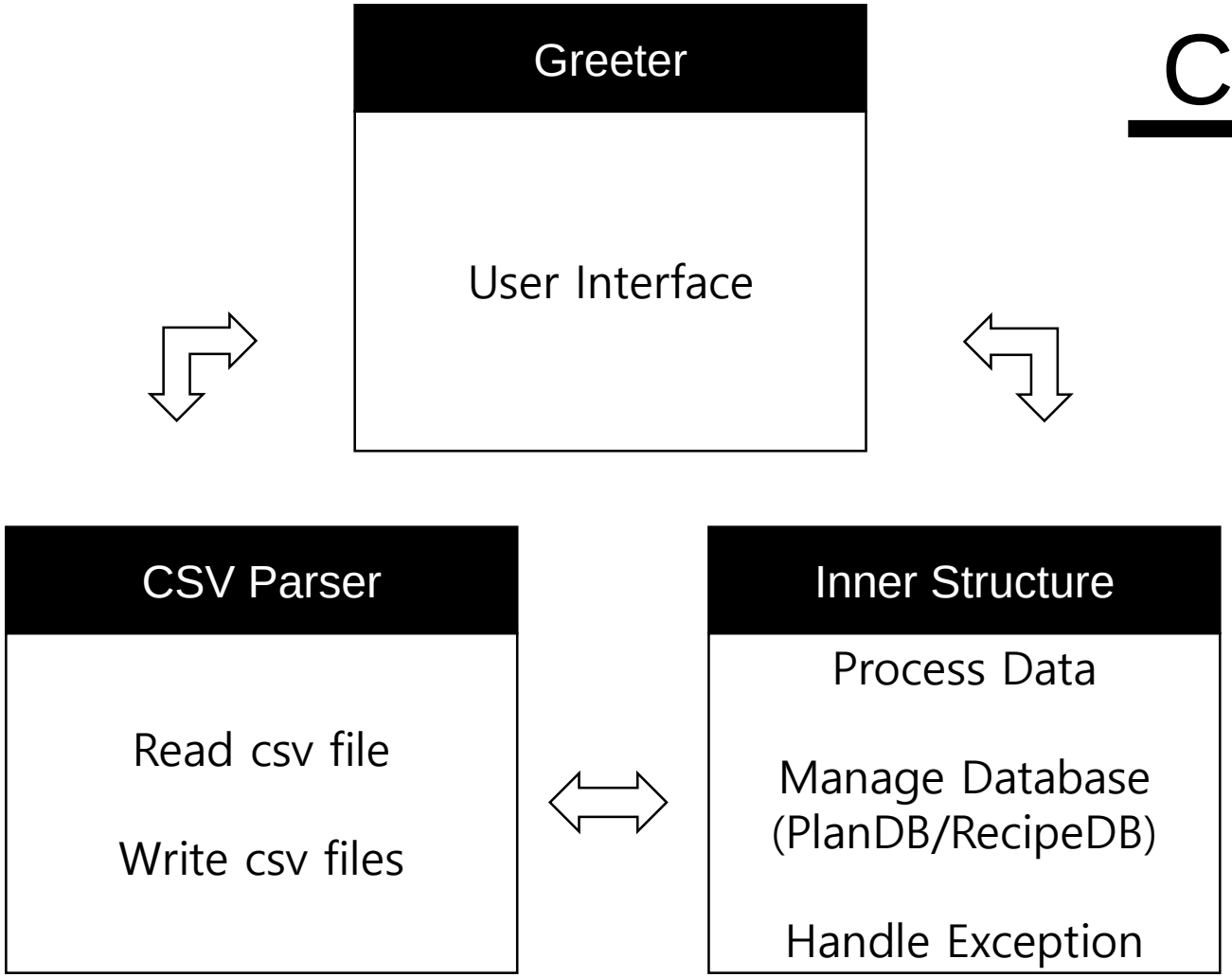


What we did

Identifying the Components



What we did



Components we needed

"Design of Components"

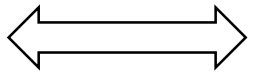
"Whose Responsibility is it?"

What we did

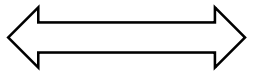
UML

Encapsulation

CSV Parser	
Private	Public
Hidden	read() write()



Inner Structure		
Public	Private	Public
_save() recipesDB plansDB	Hidden	_search() _delete() showAll()



Greeter	
Public	Private
<Menu>	Hidden

Functionality

1. Search Recipe
2. Add Recipe
3. Delete Recipe
4. View Recipe
5. Search Plan
6. Add Plan
7. Delete Plan
8. View Plan
9. Quit

DEMO Time

1. Search Recipe

Search Keyword
"Salmon"

```
You've selected [1] to search recipe
Search : Salmon
There is 2 matching result for "Salmon"
==================================================
==================================================

 >> Recipe Name : Baked Dijon Salmon
 >> Ingredients :
  - 1 1/2 tablespoons honey
  - 1 lemon for garnish
  - 1/4 cup butter melted
  - 1/4 cup dry bread crumbs
  - 1/4 cup finely chopped pecans
  - 3 tablespoons Dijon mustard
  - 4 (4 ounce) fillets salmon
  - 4 teaspoons chopped fresh parsley
  - salt and pepper to taste
 >> Cooking Time(min) : 15
 >> Direction(How To Cook)
  - 1. Preheat oven to 400 degrees F (200 degrees C).    2. In a small bowl stir together butter mustard and honey. S
et aside. In another bowl mix together bread crumbs pecans and parsley.    3. Brush each salmon fillet lightly with h
oney mustard mixture and sprinkle the tops of the fillets with the bread crumb mixture.    4. Bake salmon 12 to 15 mi
nutes in the preheated oven or until it flakes easily with a fork. Season with salt and pepper and garnish with a wed
ge of lemon.
--------------------------------------------------
 >> Recipe Name : Grilled Salmon I
 >> Ingredients :
  - 1 1/2 pounds salmon fillets
  - 1/3 cup brown sugar
  - 1/3 cup soy sauce
  - 1/3 cup water
  - 1/4 cup vegetable oil
  - garlic powder to taste
  - lemon pepper to taste
  - salt to taste
 >> Cooking Time(min) : 16
 >> Direction(How To Cook)
  - 1. Season salmon fillets with lemon pepper garlic powder and salt.   2. In a small bowl stir together soy sauce b
rown sugar water and vegetable oil until sugar is dissolved. Place fish in a large resealable plastic bag with the so
y sauce mixture seal and turn to coat. Refrigerate for at least 2 hours.   3. Preheat grill for medium heat.   4. Lig
htly oil grill grate. Place salmon on the preheated grill and discard marinade. Cook salmon for 6 to 8 minutes per si
de or until the fish flakes easily with a fork.
--------------------------------------------------
계속하려면 아무 키나 누르십시오 . . .
```

2. Add Recipe

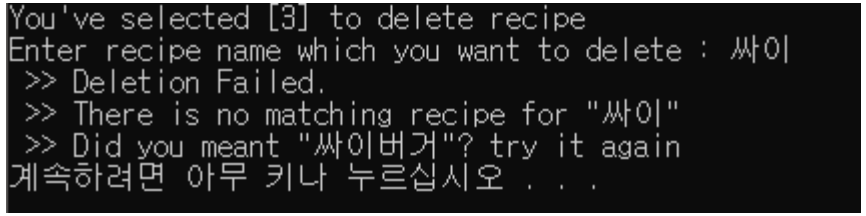
Let's add "Thigh Burger Recipe"

```
You've selected [2] to add recipe
Enter recipe information in [[ NAME -> DESCRIPTION -> TIME -> INGREDIENTS ]] order
Name                    : 싸이버거
Direction       : 1. 계란 2개를 푼 후 허벅지 살을 담군다. 2. 허벅지 살에 튀김을 입힌다. 3. 튀긴다. 4. 빵 사이에 마요네즈와 함께 넣는다.
Cooking Time(min)       : 20
Ingredients (If you want to stop enter "stop" ) : 뼈 없는 닭 허벅지 살
양상추
소금
밀가루
햄버거 빵
식용유
마늘 가루
양파
피클
후추분말
달걀
마요네즈
양파가루
stop
```

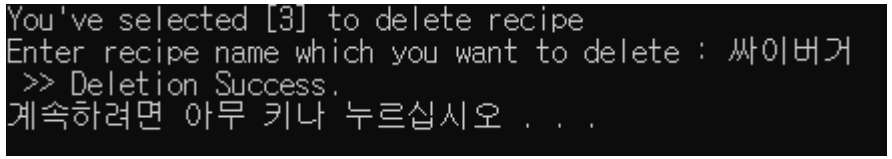
3. Delete Recipe

Let's Delete "Thigh Burger"

Invalid Deletion



Valid Deletion



4. View Recipe

Prints Recipes in alphabetical order

5. Search Plan

Search Plan by date

```
You've selected [5] to search plan
Enter date [[ YEAR -> MONTH -> DAY]] order
Year         : 2019
Month        : 10
Day          : 3
 Plan Name       : My Birthday
 >> Breakfast
 - Reduced fat yoghurt
 - Wholegrain breakfast cereal

 >> Lunch
 - Apple
 - Sandwich with salad and chicken

 >> Dinner
 - Cake
 - Pasta with beef mince and red kidney beans

--------------------------------------------------
계속하려면 아무 키나 누르십시오 . . .
```

6. Add Plan

Let's add "CAU Anniversary Meal Plan"

```
You've selected [6] to add plan
Enter plan information in [[ YEAR -> MONTH -> DAY -> DAYNAME -> BREAKFAST -> LUNCH -> DINNER ]] order
Year         : 2019
Month        : 10
Day          : 11
Plan Name       : CAU 101 Anniversary
Breakfast (If you want to stop enter "stop" ) : CAU Burger
Cola
stop
Lunch     (If you want to stop enter "stop" ) : CAU Tous les Jours
Milk
stop
Dinner    (If you want to stop enter "stop" ) : CAU Burger
stop
```

7. Delete Plan

Let's Delete "CAU Anniversary Meal Plan"

```
You've selected [7] to delete plan
Enter date [[ YEAR -> MONTH -> DAY]] order to delete plan
Year          : 2019
Month         : 10
Day           : 11
 >> Deletion Success.
계속하려면 아무 키나 누르십시오 . . .
```

8. View Plan

Prints Plans in upcoming order

```
You've selected [8] to view plan
 Plan Name      : Presentation Day
 >> Breakfast
 - Baked beans
 - Glass of reduced milk
 - Tomato
 - Wholegrain toast with polyunsaturated margarine

 >> Lunch
 - Coffee with milk
 - Wholegrain sandwich with roast beef

 >> Dinner
 - Boiled rice
 - Broccoli
 - Carrots
 - Fish prepared with olive oil
 - Potato

-------------------------------------------------------
 Plan Name      : My Birthday
 >> Breakfast
 - Reduced fat yoghurt
 - Wholegrain breakfast cereal

 >> Lunch
 - Apple
 - Sandwich with salad and chicken

 >> Dinner
 - Cake
 - Pasta with beef mince and red kidney beans

-------------------------------------------------------
 Plan Name      : CAU 101 Anniversary
 >> Breakfast
 - CAU Burger
 - Cola

 >> Lunch
 - CAU Tous les Jours
 - Milk

 >> Dinner
```

9. Quit

Exit Program

Thank You