Bon Appetit!

Project by: Scott Ragan, Priyanka Nahar, Yiqi Ou

Problem Statement:

A Personal Chef has many clients who have different dietary preferences.

Each week the chef plans the recipes which he is going to cook for the week.

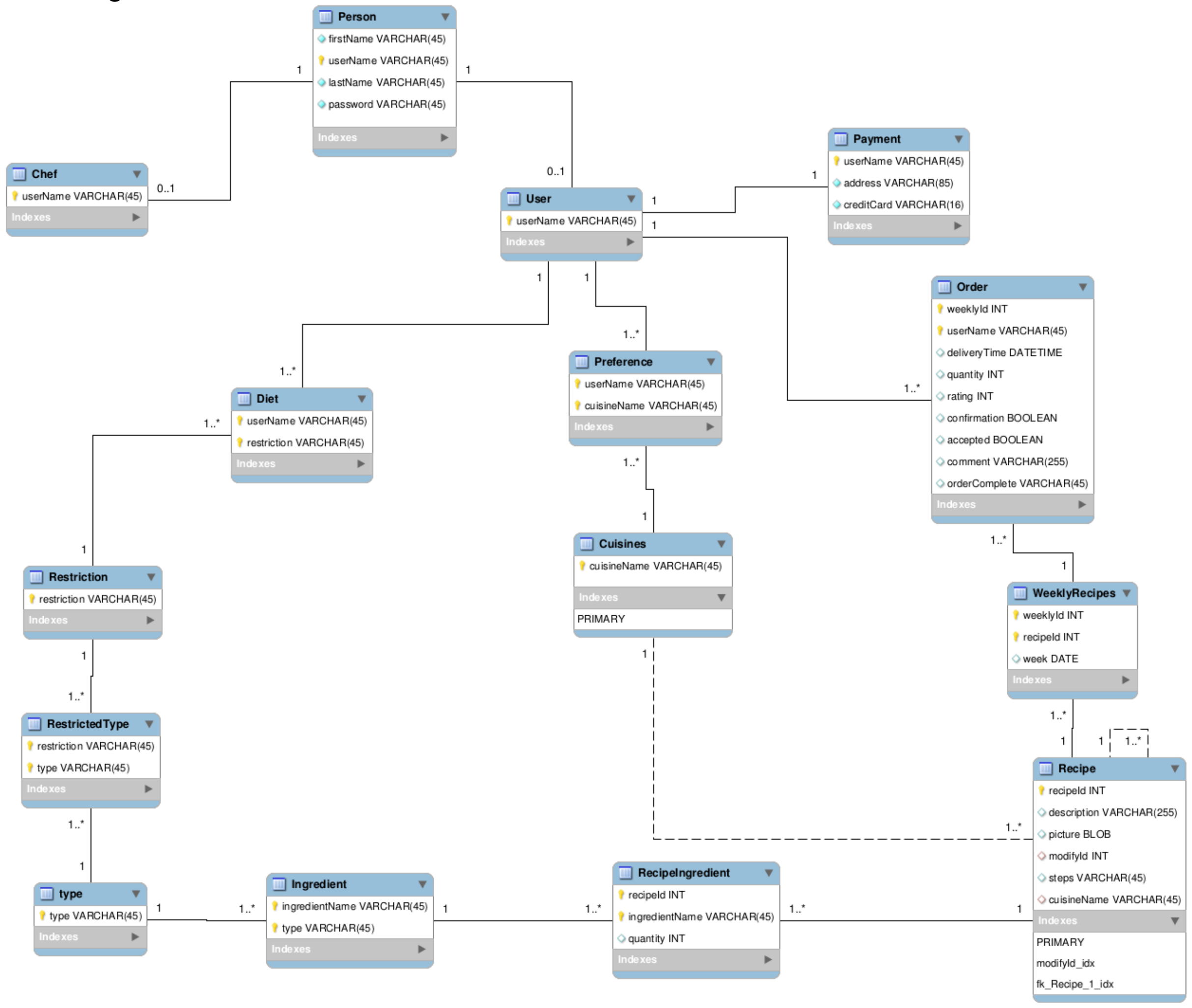
Clients wish to choose the recipes in accordance to their preferences, and the chef wants to make meals that most of his clients can eat.

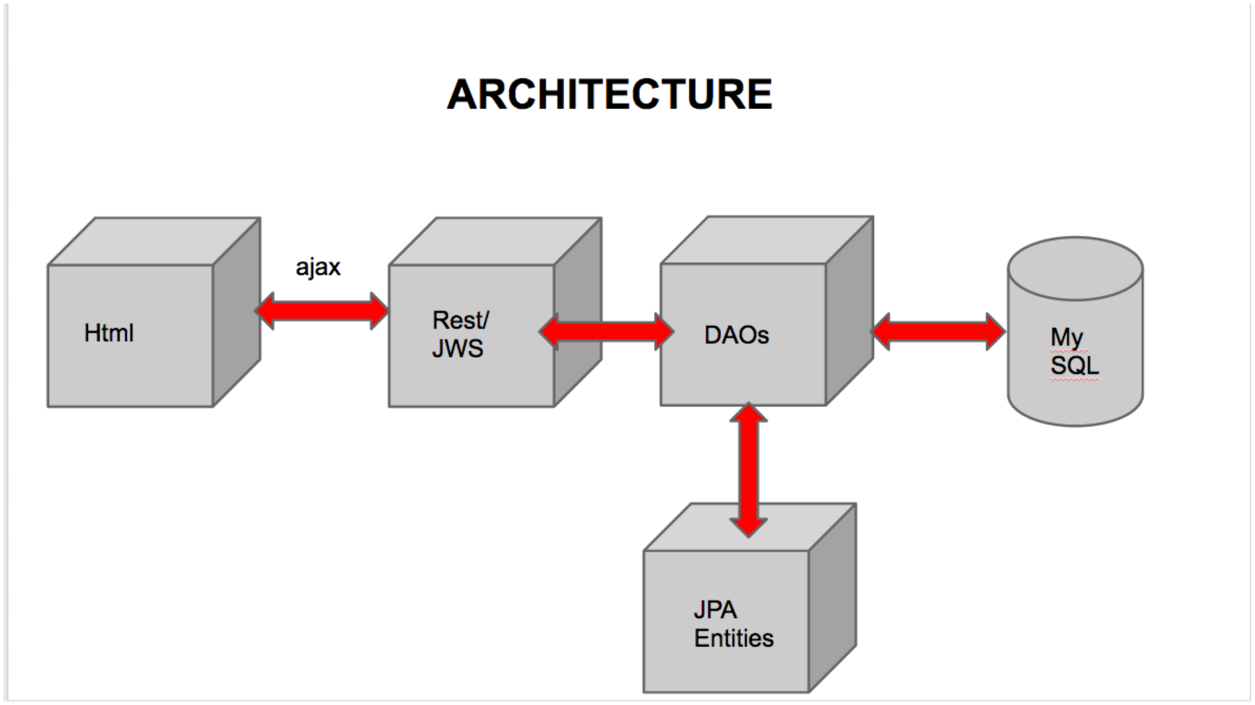
Solution:

Users and chef log into the system to view/modify recipes and weekly meal offerings. The users can select dietary preferences and see which meals contain foods they cannot eat

Depending on the type of ingredients added to the recipe and the individual preferences a customised view of the week meals will be visible by each user/customer.

UML Diagram


Person
firstName VARCHAR(45)
userName VARCHAR(45)
lastName VARCHAR(45)
password VARCHAR(45)
Indexes
Chef
userName VARCHAR(45)
Indexes
User
userName VARCHAR(45)
Indexes
Payment
userName VARCHAR(45)
address VARCHAR(85)
creditCard VARCHAR(16)
Indexes
Order
weeklyId INT
userName VARCHAR(45)
deliveryTime DATETIME
quantity INT
rating INT
confirmation BOOLEAN
accepted BOOLEAN
comment VARCHAR(255)
orderComplete VARCHAR(45)
Indexes
Preference
userName VARCHAR(45)
cuisineName VARCHAR(45)
Indexes
Diet
userName VARCHAR(45)
restriction VARCHAR(45)
Indexes
Cuisines
cuisineName VARCHAR(45)
Indexes
PRIMARY
Restriction
restriction VARCHAR(45)
Indexes
WeeklyRecipes
weeklyId INT
recipeId INT
week DATE
Indexes
RestrictedType
restriction VARCHAR(45)
type VARCHAR(45)
Indexes
Recipe
recipeId INT
description VARCHAR(255)
picture BLOB
modifyId INT
steps VARCHAR(45)
cuisineName VARCHAR(45)
Indexes
PRIMARY
modifyId_idx
fk_Recipe_1_idx
type
type VARCHAR(45)
Indexes
Ingredient
ingredientName VARCHAR(45)
type VARCHAR(45)
Indexes
RecipeIngredient
recipeId INT
ingredientName VARCHAR(45)
quantity INT
Indexes
1
1
0..1
0..1
1
1
1
1
1
1..*
1..*
1..*
1..*
1..*
1
1
1..*
1
1
1..*
1..*
1
1
1..*
1..*
1
1..*
1
1
1..*
1
1..*
1..*
1



1. **Use Case**: User Registers

Description: A user who is not a customer yet and wishes to register in the system.
Actors: Customer
Preconditions: User is not yet a customer
Steps:
Actor Actions
User submits first name, last name, username, password, address and credit card information.
System Responses
INSERTS user profile to User and Payment tables.
Alternate path
User fails to input all required fields. Error Message : User is not registered.
Changes:
Steps:
Actor Actions
User submits first name, last name, username, password.
The User inserts the Credit card information and address via the User Control page.
System Responses
INSERTS user profile to User and Payment tables.

2. **Use Case**: User Logs in

Description: A user who is a customer wishes to access the system.
Actors: Customer

Preconditions: User is a customer.
Steps:
Actor Actions
User submits username, password.
System Responses
SELECT from User ; check Password == Password.
Alternate path
Username does not exist or password does not match; request user login again.
Error Message : Please try again.

3. **Use Case**: User views preferences

Description: A user who has logged in wishes to view which his/her food preferences.
Actors: Customer
Preconditions: User is logged in.
Steps:
Actor Actions
User clicks “view preferences”.
System Responses
SELECT from Preference where Username equals username.
Alternate path
If preferences do not exist.
Message : Please set preferences.
Change:
Alternate path does not exist. The user is allowed to have no preferences.

4. **Use Case**: User edits/adds preferences

Description: A user who has logged in wishes to edit/add his/her food preferences.
Actors: Customer
Preconditions:
1. User is logged in.
2. Views preferences.

Steps:
Actor Actions
User clicks edit/add and selects a cuisine preference from a drop down menu.
System Responses
SELECT from Cuisines table(to populate drop down menu).
INSERT/UPDATE cuisine in preferences.
Alternate path
If user adds preference that exists.
Message : Preference already recorded. No change to preferences.
Change:
Alternate path does not exist.

5. **Use Case**: User views Dietary restrictions

Description: A user who has logged in wishes to view which his/her Dietary restrictions.
.
Actors: Customer
Preconditions: User is logged in.
Steps:
Actor Actions
User clicks “Dietary restrictions”.
System Responses
SELECT from Dietary restrictions where Username equals username.
Alternate path
If Dietary restriction do not exist.
Message : Please set Dietary restrictions.
Change:
Alternate path does not exist. The user is allowed to have no dietary restrictions.

6. **Use Case**: User edits/adds Dietary restrictions

Description: A user who has logged in wishes to edit/add his/her Dietary restrictions.
Actors: Customer
Preconditions:

1. User is logged in.
2. Views Dietary restrictions.

Steps:
Actor Actions
User clicks edit/add and selects a diet from a drop down menu.
System Responses
SELECT from Restriction table(to populate drop down menu).
INSERT/UPDATE Restriction in diet.
Alternate path
If user adds restriction that exists.
Message : Restriction already recorded. No change to dietary restrictions.

Change:
Alternate path does not exist.

7. **Use Case**: Views past week’s recipes

Description: A user who has logged in wishes to view the recipes the chef is preparing for this week.

Actors: Customer
Preconditions:
User is logged in.
Steps:
Actor Actions
1. User clicks on Previous week's recipes and select a particular week.

System Responses
SELECT from Weekly recipes where week = particular week.
Alternate path
If previous week recipes do not exist.
Message : This week's recipes do not exist.

Not done.

8. **Use Case**: Views weeks' recipes.

Description: A user who has logged in wishes to view the previous weeks recipes.
Actors: Customer
Preconditions:
User is logged in.
Steps:
Actor Actions
User clicks on "This week's recipes."
System Responses
SELECT from Weekly recipes where week= current week.
The system matches the restricted ingredients to the week's recipes and highlights these to the user using the Restriction,type table etc.
Alternate path
If weekly recipes have not been generated.
Message : This week's recipes have not been created yet.

9. **Use Case**: Chooses weekly meal & schedule delivery time (quantity and comments)

Description: A user who has logged in wishes to choose this week's recipes that the chef has prepared for the week.
Actors: Customer
Preconditions:
User is logged in.
The week's recipe has been created.
User is viewing current week recipes.
Steps:
Actor Actions
1. User selects a recipe from "This week's recipes."
2. User enters quantity.
3. User enters delivery time.

4. User enters comments if any.
5. Clicks add order.

System Responses
SELECT from Weekly recipes where week = current week.
INSERT the order selected into order table.
Alternate path
If user clicks on the recipe they already have an order for.
Message: Past order already exists. Are you sure you want to make changes (order is updated).

Changes:
Alternate path does not exist.

10. **Use Case**: Views the order before confirmation

Description: A user who has logged in, selected this weeks recipe and wants to view the order summary before confirming the order.
Actors: Customer
Preconditions:
User is logged in.
User has viewed and selected the week's meal.
User has added the meal/s to his order.

Steps:
Actor Actions
User is viewing/updating the order.
System Responses
System select order from order table and update.

Not done

Use Case: Confirms and pays.
Description: A user who has logged in, selected this weeks recipe and wants to confirm the order.
Actors: Customer
Preconditions:
User is logged in.
User has viewed and selected the week's meal.
User has added the meal/s to his order.
User has viewed his order.
Steps:
Actor Actions

User clicks on “Confirm Order.”
System Responses
Authenticates the user’s payment information.
UPDATE confirmation field in order table is set to true.

Alternate path
If the user’s payment information is not authentic.
Message : Please update payment information
(The confirmation field in order table is set to ‘False’’)

If user clicks on cancel order.
Message: Order cancelled.
(Order is delete from order table.)

Not done.

12. **Use Case**: View order history
Description: A user who has logged in, wishes to view his/her past orders.
Actors: Customer
Preconditions:
User is logged in.
User has past orders
Steps:
Actor Actions
User clicks on “View order history”
System Responses
SELECT from order table, past orders (date < today)
Alternate path
If user has no past history, button is unavailable.

Changes:
Alternate path does not exist.

13. **Use Case**: Rates past orders
Description: A user who has logged in and wishes to provide rating scores on orders that were completed in the past
Actors: Customer
Preconditions:
User is logged in.
User has past orders, which were completed
Steps:
Actor Actions

User clicks on “View order history”
User selects a rating for a past order
System Responses
UPDATE order table, with rating on past order
Alternate path
If user has no past history, rating is unavailable

Not done.

14. **Use Case**: chef login
Description: A user, who is a chef, wishes to authenticate into the system.
Actors: Chef
Preconditions:
User is of type chef
User isn’t logged into system
Steps:
Actor Actions
User submits username, password.
System Responses
SELECT from User ; check Password == Password.
Check to see that UserID is on Chef table
Alternate path
Username does not exist or password does not match; request user login again.
Error Message : Please try again.
Username is not on Chef table
-Log user in as normal customer

15. **Use Case**: chef creates ingredient
Description: A chef is building his/her ingredients list, and wishes to create a new ingredient, which he/she may use in a future recipe.
Actors: Chef
Preconditions:
User is of type chef
User is logged in
Steps:
Actor Actions
User clicks “add ingredient”
User enters ingredient name
User enters ingredient type (if type exists)
System Responses
INSERT ingredient, type to Ingredient table

Alternate path
Ingredient already exists
-Error message: Ingredient already exists
Type doesn't exist
-Error message: Type doesn't exist, add type before adding to ingredient.
Change:
Alternate path:
Will not throw that message but will not add a duplicate to the list.

16. **Use Case**: Chef creates recipe/adds from yummly
Description: A user, who is a chef, wishes to create/add recipe into the system.
Actors: Chef
Preconditions:
User is of type chef
User is logged into system.
Steps:
Actor Actions
User creates recipes or user pulls recipe from Yummly.
System Responses
INSERT recipe in Recipe table or pulls recipes from Yummly and inserts in Recipe table.

Alternate path
If ingredient does not exist.
System adds the ingredient.

Change:
Food2fork API was used. The chef can view these recipes but cannot add them to the database. He has to create his own recipes in order to add them.

17. **Use Case**: Chef accept or deny the order
Description: A user, who is a chef, wants to accept or deny the order the customer has confirmed.
Actors: Chef
Preconditions:
User is of type chef
User is logged into system.
The order has been confirmed by customer.
Steps:
Actor Actions
User click the order and select the options of 'accept' or 'deny'
System Responses
UPDATE the order 'accepted' field in the order table.

Alternate path

Changes:
He has to create his own recipes in order to add them.

18. **Use Case**: chef modifies recipe update/insert
Description: when chef find restriction of customer in the recipe, he/she can modify the recipe. Doing this will create a new modified version of recipe in addition to the original.
Actors: Chef
Preconditions:
User is of type chef
User is logged into system.
Recipe exists
Steps:
Actor Actions
Chef selects an existing recipe and modifies it.
System Responses
The system inserts a new record in the recipe table linked by the original one.
Alternate path
The ingredients do not exist. The system notifies the chef the ingredients which do not exist and prompts the chef to add ingredients.

Not done

19. **Use Case**: chef select recipe for this week
Description: Chef chooses weekly recipes, he is aided by a helpful word cloud of his customers' preferences.
Actors: Chef
Preconditions:
User is logged in.
User is type chef.
User has not yet set weekly recipes.
Steps:
Actor Actions:
-Adds recipes to weekly recipes from list of all recipes.
System Responses
-Adds selected recipes to Weekly Recipe table
-Word Cloud of current customer preferences is displayed to help chef choose this weeks recipes. (Word Cloud is generated from API call using all current client preferenes)

Not done

20. **Use Case**: chef views a recipe (show number of users who have dietary restrictions)
Description: At any time while using the system, the chef may want to view a particular recipe. When he/she sees the recipe, data about which users have dietary restrictions against the recipe will be displayed, along with regular recipe data.
Actors: Chef
Preconditions:
User is logged in.
User is type chef.
Steps:
Actor Actions:
User clicks on a recipe
System Responses
SELECT recipe from recipe
SELECT dietary restrictions from restrictions
SELECT preferences from preferences
SELECT ratings from orders
Display recipe, with counts of users who have restrictions by:
- ingredients
- overall recipe

Display ratings (if recipe was used in past)
Display how well recipe meets current user preferences
Display number of previous orders of recipe (if any)
Display all connected recipes (variations of current recipe)

Not done

Count :
Given Use Cases : 20
Use Case Implemented: 12

Category	**Weight (%)**	
Relational Model or MongoDB	**25**	

Tables (count)	15	
Fields (count)	42	
Foreign Keys (count)	17	
Mapping Tables (count)	4	
Association Tables (count)	1	
Enumeration Tables (count)	2	
Object Model	**20**	
Entity Classes (count)	12	
Entity Fields (count)	53	
DAOs (count)	11	
Create Methods (count)	12	
Find One Methods (count)	14	
Find All Methods (count)	6	
Find Other Methods (count)	2	
Delete Methods (count)	10	
Update Methods (count)	16	
JWS or Express Web Services	**20**	
End Points (Mapped Methods) (count)	26	
JSON Producers (count)	26	
JSON Consumers (count)	7	
GET Methods (count)	16	
POST Methods (count)	8	
PUT Methods (count)	1	
DELETE Methods (count)	0	
Server Web Service Client		

Browser Web Service Client	**15**	
Online Web Services (list names)	food2fork	
Online Web Services (count)	1	
AJAX GET (count)	18	
AJAX POST JSON (count)	5	
AJAX PUT JSON (count)	1	
AJAX DELETE (count)	0	
User Interface	**20**	
Pages (count)	11	
Input Fields (count)	21	
Links (count)	0	
Buttons (count)	16	
Data Tables (count)	1	
Data Lists (count)	9	
JavaScript Libraries (count)	1	
CSS Libraries (count)	1	
Look and Feel (count)	?	