**CS 571 – Artificial Intelligence**

**Expert System for Recipe Recommendation**

**Project Report**

**Dhruval Shah**

**200361439**

[**shah042@cougars.csusm.edu**](mailto:shah042@cougars.csusm.edu)

**Project Completed**

**Check List**

* ✅ **Yes Did you follow all the requirements in implementing the system?**
* ✅ **Yes Did you create one Word file of your report with the cover page and section headers as specified?**
* ✅ **Yes Did you answer all questions per section?**

**Section 1: The purpose of the Expert System**

1. My project name is Recipe Recommendation System, which recommends various recipes based on Cuisine, Ingredients and taste. So, basically this expert system is for Selection which means Selection of dishes based on available ingredients.

2.The goal of the program is to suggest all the dishes or dessert that user can make based on the availability of ingredients. We start by asking the user if he wants to make a dish or dessert. Based on user’s choice, we ask which cuisine he wants to try. Then we ask for two ingredients and suggest a recipe to user.

As mentioned above, the expert system asks for the available ingredients and suggest a recipe. So, here the user needs to give input that which two ingredients are available to him.

Based on the above inputs, the program provides respective suitable output.

Let us consider an example

Suppose, user wants to make a dish, then user has to enter the following as input.

Do you want to make a dish or sweet? **dish.**

Cuisine? **indian.**

Ingredient1? **rice.**

Ingredient2? **tamarind.**

Spicy level? **70.**

Based on the above input, the expert system will suggest following item.

You can make Puliyoagare

Whole Recipe of Puliyoagare is available on this link: https://www.indianhealthyrecipes.com/puliyogare-recipe/

3. BWC is suitable for this problem domain rather than FWC.

In **Forward Chaining (FWC)**,it starts from the available facts and moves forward by applying inference rules to extract more data. It continues until it finds the goal. It looks for the problems that start from data collection.

In **Backward Chaining (BWC),** we start from the main rule and then based on the RHS of the main rule, we go to the particular sub rule. If no match found, then we backtrack and we do this until we find the solution. Backward changing is more efficient and faster than Forward Chaining as it tests just fewer rules compare to Forward Chaining. Backward Chaining is more suited for our expert system because

we start with a main and then go to sub goals and finally get the result.

BWC, DFS is suitable for this problem domain. In my expert system BWC is suitable because we start from asking user to provide input for what he wants to make. Either dish or dessert. Based on this selection, we ask user to enter available ingredients and spicy level. After collecting input, we check the rules which are true and false. And if the rule is false then we backtrack and check another rule in the DFS tree. And DFS gives a guaranteed solution. Based on user input we try to satisfy one rule, if the rule satisfies then we suggest a dish and if not then we check another rule but the one which has at least one ingredient. If the expert system still can not find the dish, then it will suggest user to make default dish. So, BWC is useful in this domain.

If we try to use FWC in our expert system, then we will need to have large number of facts stored and suppose if the fact is false then we can not get the goal (i.e. to get a dish or dessert name) quickly. But using BWC we can prove our assumption quickly.

**Section 2: Expert System Implementation**

Facts: Facts are some predicate expressions that predicates are followed by some arguments which can be used to make declarative statement about the problem.

1. I have not used any facts in my program. There are few rules and sub-goal rules.
2. In my expert system, there are few types of rules. User input rules, main rules and sub goal rules.

* **User input rules:** In this expert system, there are two rules which check that whether user has input dish or dessert. Based on this selection they call the main choose rules. For example, the rule called **dishes** which is triggered when user entered dish. This rule calls main choose rule i.e. **choose\_dish().** Suppose, if the user inputs dessert, then the rule called **desserts** is called. And it further calls the rule **choose\_dessert()** main choose rule.
* **Main Chose Rule:** Main choose rules are choose\_dish() and choose\_dessert(). Choose\_dish() is called for getting suggestions for dishes and choose\_dessert() is called for getting suggestions for desserts. There are many ingredients and we have rules for every ingredient. Below are the examples of choose\_dish() and choose\_dessert().
  + choose\_dish(Cuisine, Ingredient1, Ingredient2, Taste, 'Khichu') :- indian(Cuisine), rice\_flour(Ingredient1), ginger(Ingredient2), spicy(Taste).
  + choose\_dessert(Cuisine, Ingredient1, Ingredient2, 'Sweet Rice Recipe') :- mexican(Cuisine), rice(Ingredient1), milk(Ingredient2)
* **Sub Goals Rule:** When the choose\_dish() and choose\_dessert() rules are called, spicy level check. It checks that spicy level is >50 or less. Based on this, it divides a dish into a spicy dish or mild dish.

**Example:**

spicy(Taste) :- Taste > 50.

mild(Taste) :- Taste =< 50.

There are two special sub goals rules which are special\_ingredient() and rice\_indian().

**Example:**

special\_ingredient(Ingredient1, Ingredient2) :- besan\_flour(Ingredient1), ginger(Ingredient2).

rice\_indian(Ingredient1,Cuisine) :- Ingredient1 = rice, Cuisine = indian.

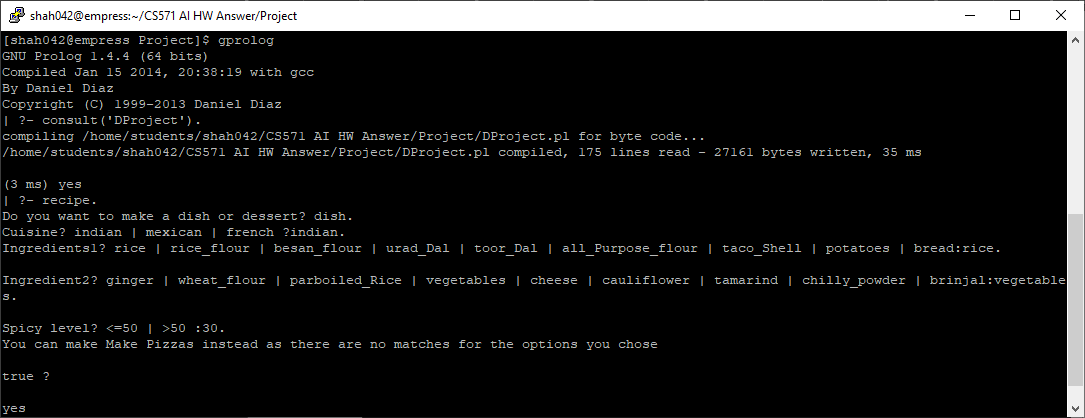
**Section 3: Testing and Results**

The expert system runs successfully and provides desired dishes as per the rules.

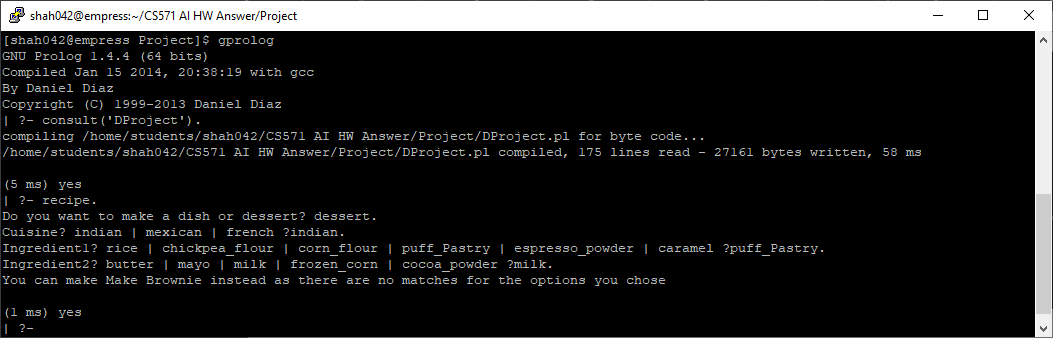
Following are the screenshots of output that we obtain.

Test 1: When no conditions matches and there is no recipe available for input ingredients, at that time the default is chosen. That means expert system suggests user to make Pizzas.

**Dishes:**



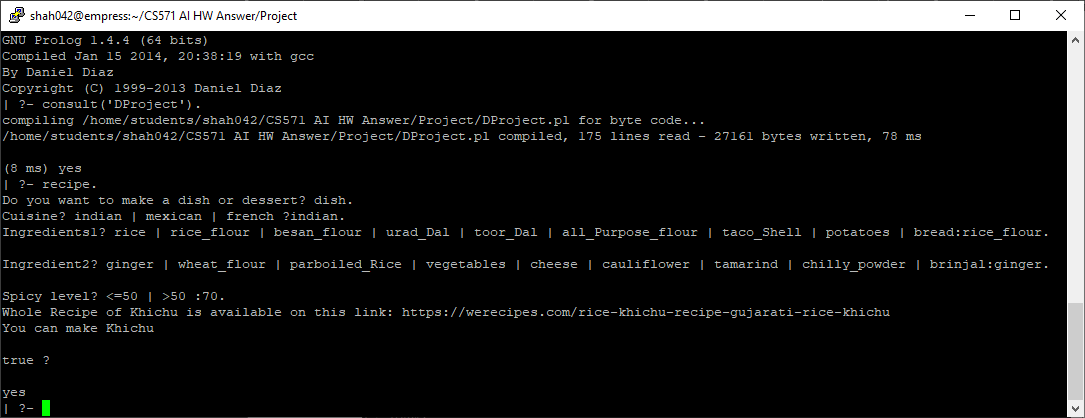
**Brownie:**



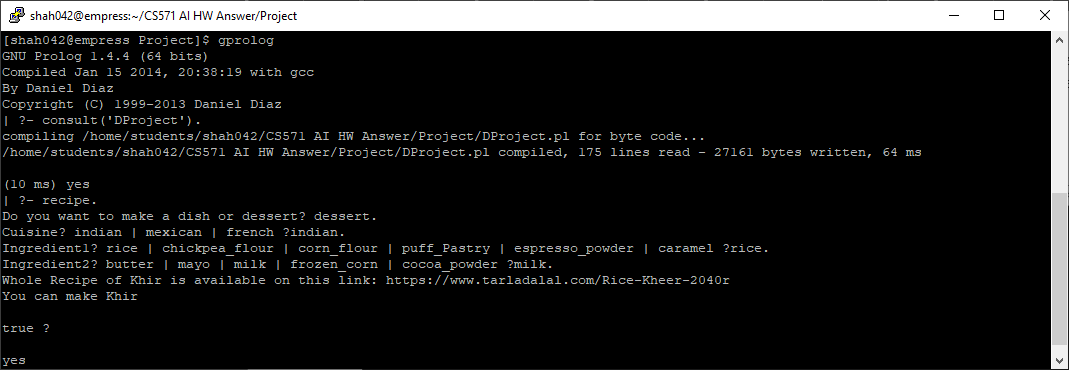
**Test 2:**

In Recipe recommmendation, if user enters the ingridents which make a taste dish, then the expert will suggest user to make that dish and it’ll also provide a web link which user can follow it for detailed recipe.

For Dish:



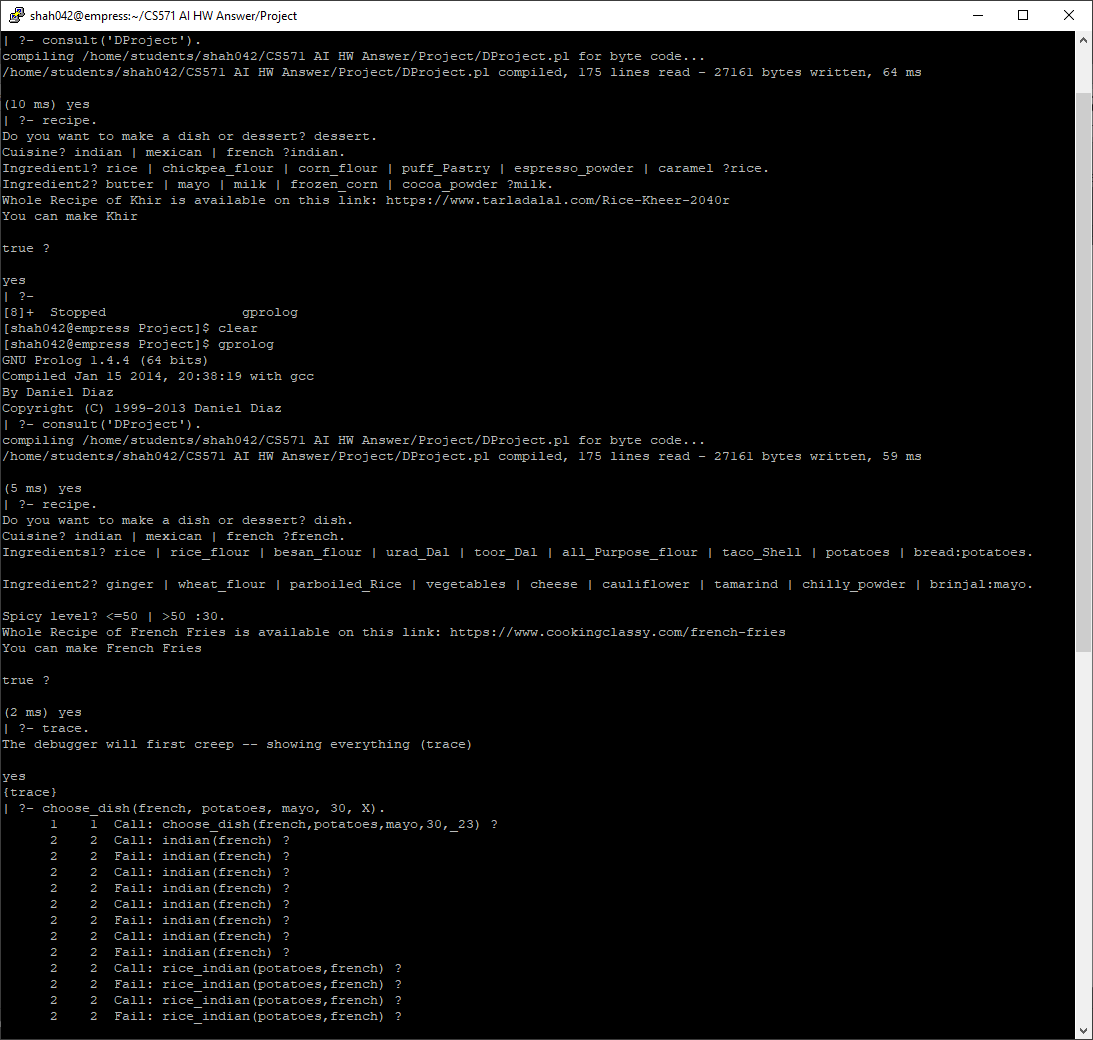
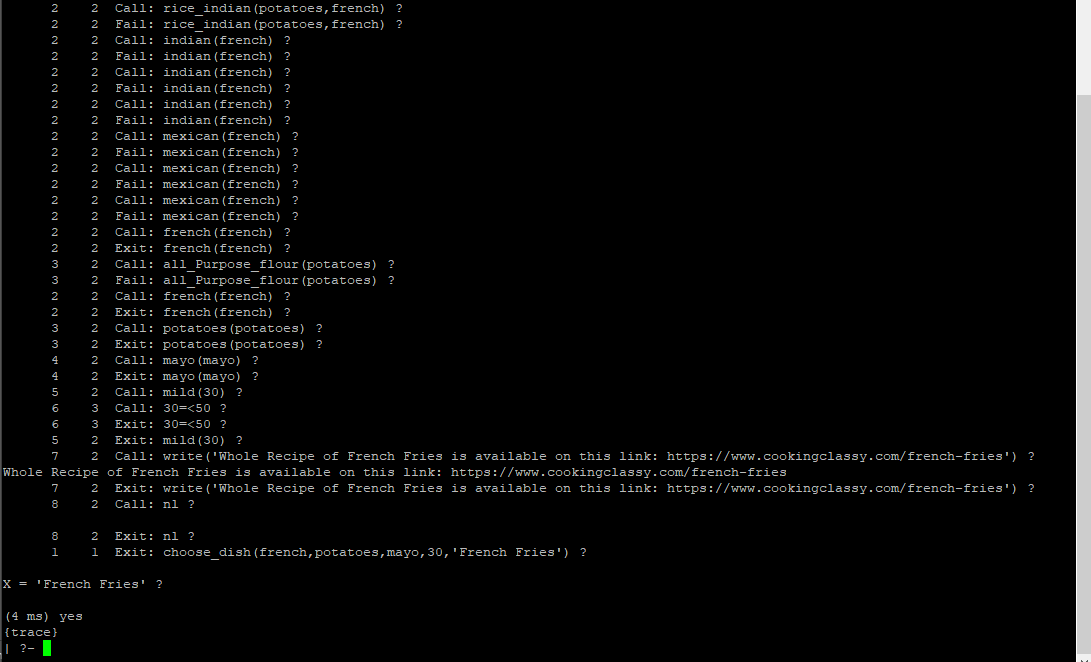
For Brownie:



For Test 3 and 4, we will do backtracking and check whether we are getting correct recipe or not.

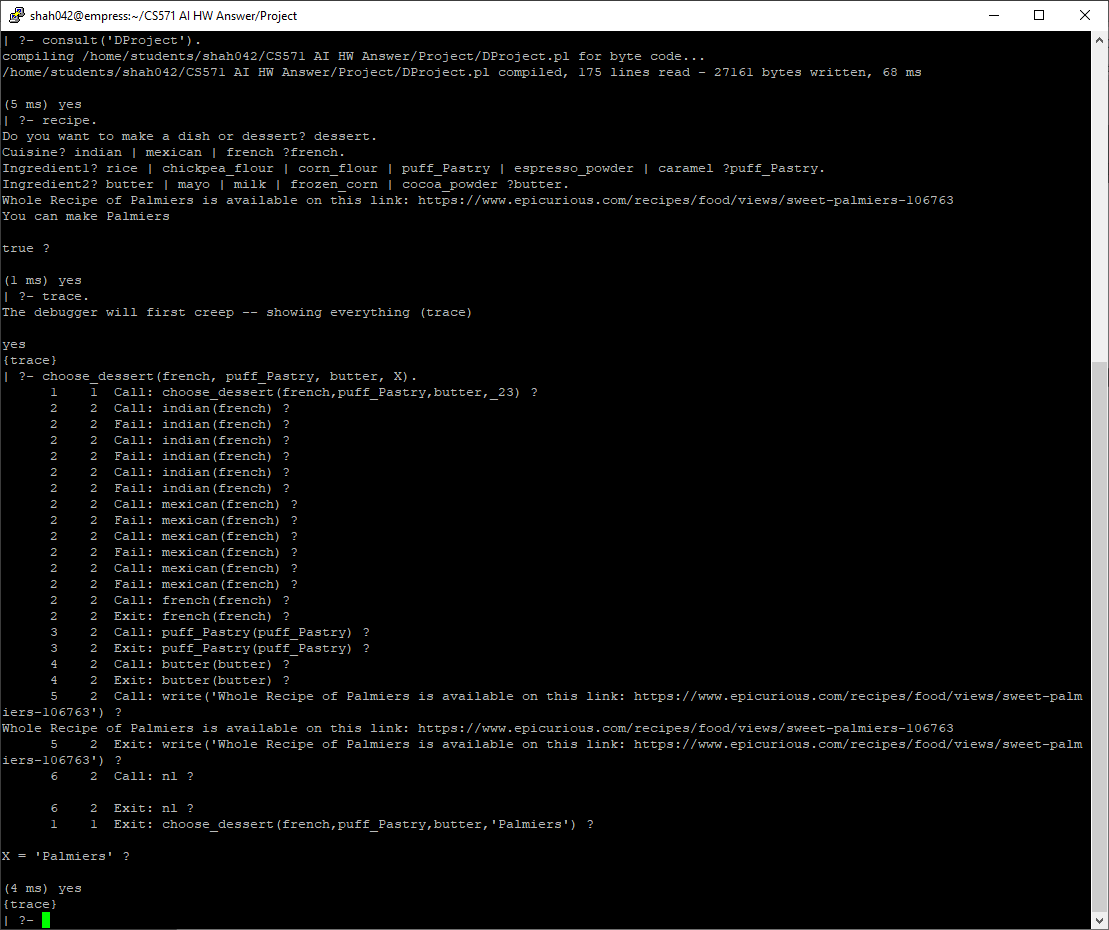
**Test 3:**

choose\_dish(Cuisine, Ingredient1, Ingredient2, Taste, 'French Fries') :- french(Cuisine), potatoes(Ingredient1), mayo(Ingredient2), mild(Taste), write('Whole Recipe of French Fries is available on this link: https://www.cookingclassy.com/french-fries').

**Test 4:**

choose\_dessert(Cuisine, Ingredient1, Ingredient2, 'Palmiers') :- french(Cuisine), puff\_Pastry(Ingredient1), butter(Ingredient2), write('Whole Recipe of Palmiers is available on this link: https://www.epicurious.com/recipes/food/views/sweet-palmiers-106763'),nl.



As we are getting same value of X, one we got from FWC and one we got from this BWC, both are same so our expert is working perfectly fine.

**Section 4: Confidence Analysis**

As per MYCIN method, we know that we can have either two or more rules to conclude a parameter which has different weights of evidence.

We will use MYCIN method to calculate a confidence of the test results of test 3 and test 4.

**Test 3:**

choose\_dish(Cuisine, Ingredient1, Ingredient2, Taste, 'French Fries') :- french(Cuisine), potatoes(Ingredient1), mayo(Ingredient2), mild(Taste).

Calculation:

= Min (french(Cuisine), potatoes(Ingredient1), mayo(Ingredient2), mild(Taste)) \* CF (0.6)

= Min (0.8, 1, 1, 0.9) \* 0.6

= 0.8 \* 0.6

= 0.48

So, **Confidence = 0.48**

**Test 4:**

choose\_dessert(Cuisine, Ingredient1, Ingredient2, 'Palmiers') :- french(Cuisine), puff\_Pastry(Ingredient1), butter(Ingredient2).

Calculation:

= Min (french(Cuisine), puff\_Pastry(Ingredient1), butter(Ingredient2)) \* CF(0.6)

= Min (0.8, 1, 1) \* 0.6

= 0.8 \* 0.6

= 0.48

So, **Confidence = 0.48**

**Section 5: Adding Machine Learning**

In this Recipe Recommendation System, we have rules for ingredients, cuisine and taste. Suppose, if it does not find ingredient match with the rule, it will suggest to make a default dish. In this case, we could use machine learning such that the system learns from the already existing rules and further learn to give output when a new set of conditions occur. One way to incorporate machine learning into rule-based expert system is **Inductive Learning**. In this case, system is able to generate its knowledge, represents as rule.

This expert system gives a default output when the conditions do not match. Instead using Machine Learning, the system can learn from the existing data and can give the exact output. To learn the expert itself, we need to have large amount of data available, a recipe of many dishes and desserts indexed by the ingredients and taste. Having this large amount of data, actually it is very important because now we do not have to write rules or condition, if we have few rules then also it is sufficient. The system can learn from existing rules and recommend dishes and desserts. We just need to provide more data and using machine learning our expert will learn and make rules.