



CZ3005
ARTIFICIAL INTELLIGENCE

LAB 3 REPORT

Assignment 3: Subway sandwich interactor

LAB GROUP : TS4

Sam Jian Shen (U1821296L)

School of Computer Science and Engineering
Nanyang Technological University

Contents

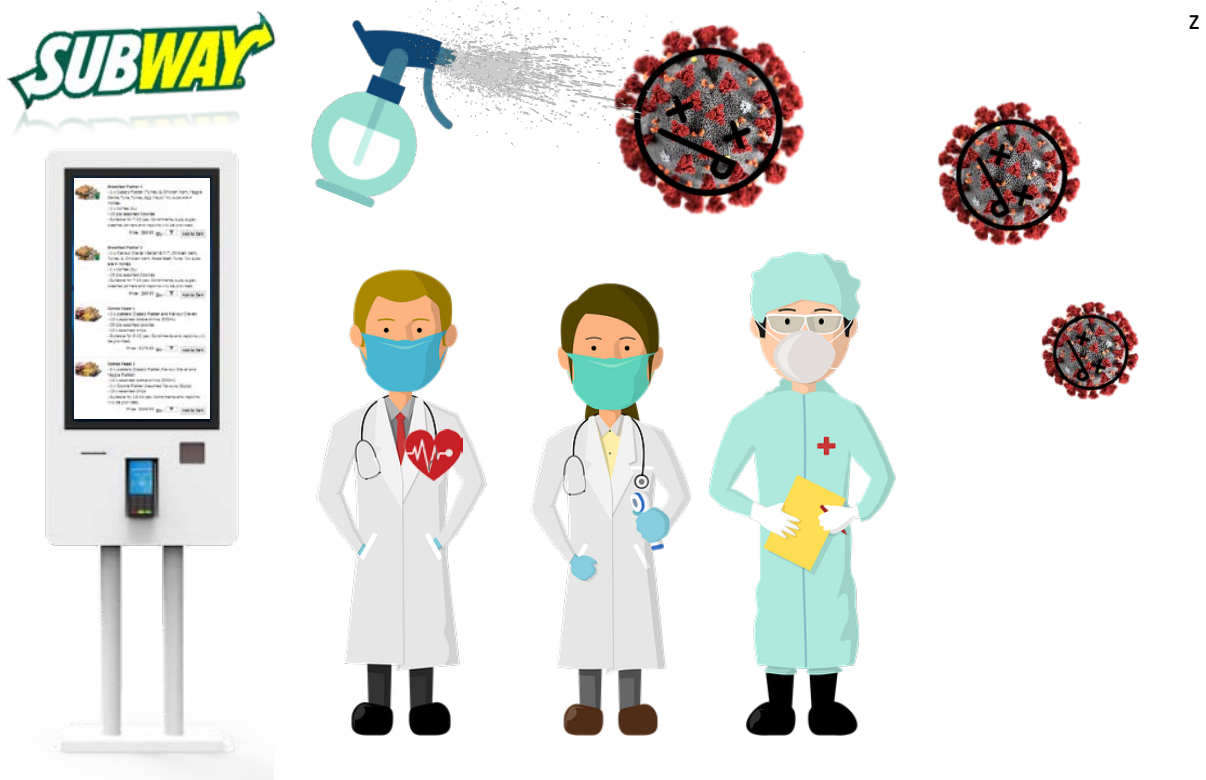
The Requirement	3
The Story	3
The Goal	4
General Software Architecture	4
Prolog + Javascript	4
Prolog Only.....	4
Software Flow	5
Abstract.....	5
Rules.....	5
Assumption	5
Getting Setup and Execute	5
Requirement	5
Prolog + Javascript	5
Prolog Only.....	6
Samples Walkthrough and Highlights	6
Prolog Only.....	6
Prolog + Javascript	7
Appendix	9
References	9
Disclaimer.....	9

The Requirement

The prolog script offers different meal options, sandwich options, meat options, salad options, sauce options, top-up options, side options, etc. to create a customized list of a person's choices. The options should be intelligently selected based on previous choices. For example, if the person chose a veggie meal, meat options should not be offered. If a person chose a healthy meal, fatty sauces should not be offered. If a person chose a vegan meal, cheese top-up should not be offered. If a person chose a value meal, no top-up should be offered.

Since creativity is part of the consideration, I would like to create a meaningful story out of this requirement instead!

The Story



There has been a pandemic disease going throughout the world also known as “COVID-19”. In Singapore, the daily new cases who are tested positive for “COVID-19” are exponentially on the rise. There has been increasing concern where doctors and nurses who are the frontline of the battle against this disease, as they will be the most vulnerable to get an infection as the demand for hospitalization increases. The government has suggested a few ideas to mitigate this concern. One of the project titles called “Subway sandwich interactor” which collaborates with SUBWAY; an American privately held restaurant franchise that primarily sells submarine sandwiches and salads. It is one of the ways to reduce interaction between vendors and customer especially the doctors and nurses. This product helps retain SUBWAY business while upholding the social distancing measures. This product will be implemented throughout Singapore hospitals.

The project team consists of 10 members. I am in-charge of software development. My task was to prototype an interactive interface with **Prolog** and **Web-based** that offered the requirement stated by the project manager aforementioned in “the requirement” section.

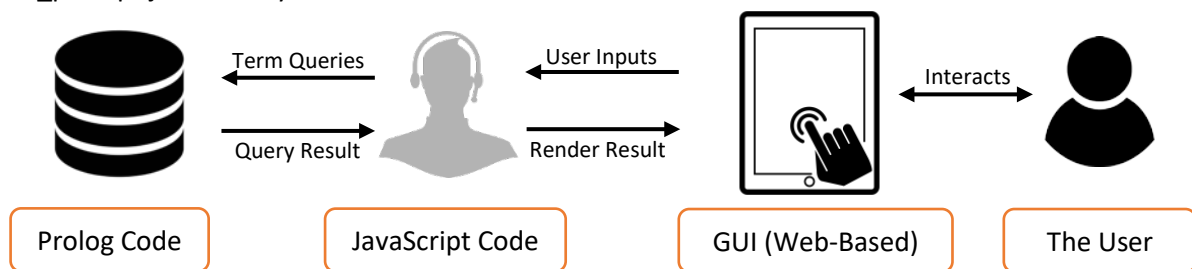
The Goal

- Requirement
 - o A typical subway dialogs with the **minimum of prolog command prompt only** program.
 - o **Each option** must have a certain degree of **constraint base on past selected options**.
 - o **Dynamic Facts** allow the possibility of **manipulation and development of new facts**.
- Ambitious
 - o Integration with **a database file**.
 - o **Web-based** subway dialogs based on **image selection**.
 - o Support **voice assists** that read the queries.

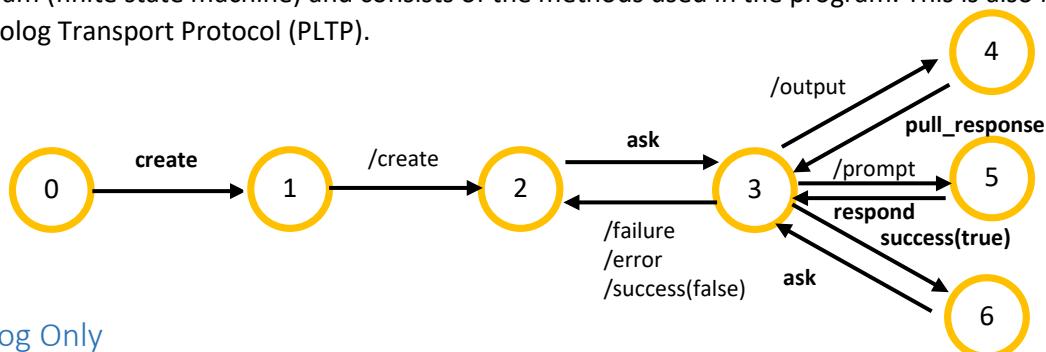
General Software Architecture

Prolog + Javascript

The program is initiated via 'SSI_Run.pl' server, with the front-end run by the browser. The prolog code is run by 'SSI.html'. This is made possible with prolog library 'engine.js'. This package allows JavaScript to run in a web client and control between Prolog and GUI. It is servers as an API to store the options selected by the user and generate new facts/rules based on past choices with 'SSI_prompt.js' as the dynamic visual control of 'SSI.html'.

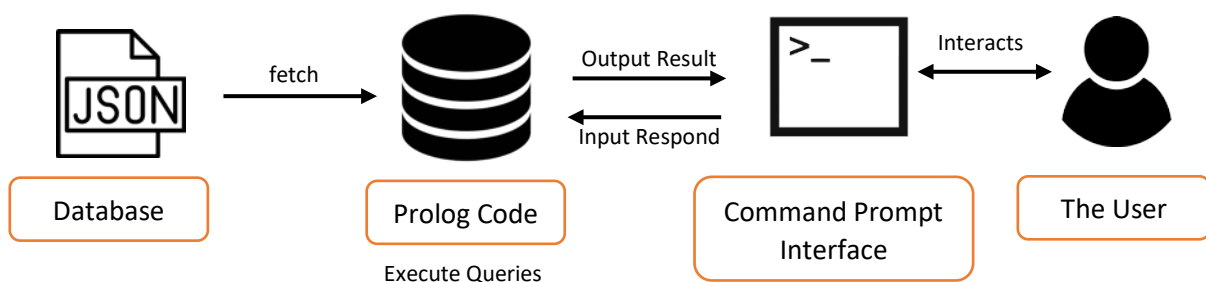


There are many communication methods offer in 'pengines', below depicts the communication diagram (finite state machine) and consists of the methods used in the program. This is also known as Prolog Transport Protocol (PLTP).



Prolog Only

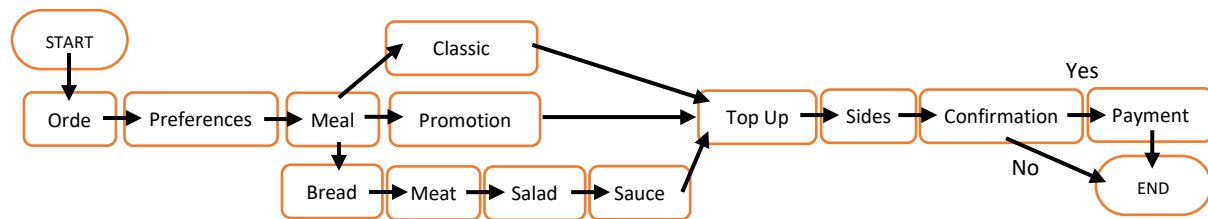
This program is the initial phase of the development phase. It runs with executing from SSI_KBS.pl and using JSON as a database environment to fetch the facts. This is possible from library function from 'http/json'.



Software Flow

Abstract

These are all the main options. The program will restart once it reached the 'end' node.



Rules

1. The User that select '**vegan**' as **preferences** must **not** be able to **choose the meat option**.
2. The User that select '**healthy**' as **preferences** must have **reduced specific options** (like bacon for example).
3. The User that select '**none**' as **preferences** must display **all possible option** given from the database
4. The User that select **not('none')** as preferences must have **reduced specific options or reduce choices of the option** (specific refers to a subset of all options).

Assumption

- All options must be unique (For example, options(food) -> options(food)) is not allowed).
- All food listed must have a unique name. (For example, 'Veggie Delite' and 'veggiedelite' are considered the same name)
- (For Prolog Only) The user must not key in wrong input which results in abort during queries or error.
- (For Prolog + Web) The browser will automatically refresh if idle for a fixed amount of time.
- (For Prolog + Web) The user is using google chrome browser Version 80.0.3987.149 (Official Build) (64-bit) and above.
- (For Prolog + Web) The user has made payment after the selected payment option.
- An option consists of 1 or more choices.

Getting Setup and Execute

Requirement

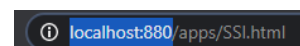
- SWI-Prolog version 8.0.3-1 Installed.
- Google Chrome Browser Version 80.0.3987.149 (Official Build) (64-bit) and above.
- Downloaded and Extracted CZ3005_Subway_Sandwich_Interactor-master.zip.
 - o It consists of a **demo video** for the **prolog web** (javascript)
 - o A **prolog only** application and **README.md** consist of detail setups.
 - o A **prolog + javascript** application.



Prolog + Javascript

1. Double click and navigate into the 'SSI_Prolog_Web' folder.
2. Open and Execute 'SSI_Run.pl' file.
3. Type 'run.' and enter to fire up the server. (reload. or stop. the server is available)
4. Launch 'Google Chrome' browser.
5. At the URL and type 'localhost:880' and enter.
6. Click on the images or buttons to interact with the program.

```
?- run.  
% Started server at http://localhost:880/  
true.
```



Prolog Only

1. Double click and navigate into the 'SSI_PrologOnly' folder.
2. Open and Execute 'SSI_KBS.pl' file.
3. Type 'run.' and enter to run the program. ?- run.
4. Type 'abort.' and enter to abort the program. (reload. also available)

```
Which promotion meal would like to pick?
|: abort
You have chosen abort.
% Execution Aborted
?- 
```

```
?- reload.
"Initialize System Library Usage Successfully"
"Clear Dynamic Memory of Database Facts Successfully"
"Data Retrived <meal> From Database Successfully"
"Data Retrived <custom> From Database Successfully"
"Data Retrived <top_up> From Database Successfully"
```

5. Users must key in the options to continue. Acceptable options for example: 'Veggie Delite', 'veggiedelite'. User input options can be y/n or the options given by the question.

Samples Walkthrough and Highlights

Prolog Only

Below highlights the contrast of 2 samples of dialogue outputs from 2 different preferences.

Sample 1

```

SSSSSSSS. UUUU. UUUU. BBBBBBBBBB BBBBB. UUUUU. UUUU. AAAAA. AAAY. YYYY.
SSSSSSSSSS. UUUU. UUUU. DDDDDDDDDDD. UUUU. UUUU. UUUU. AAAAA. AAAY. YYYY.
SSSSSSSSSS. UUUU. UUUU. BBBBBBBBBB BBBBB. UUUUU. UUUU. AAAAA. AAAY. YYYY.
SSSSS. SSSSS. UUUU. UUUU. DDD. DDD. UUUU. UUUUUUU. UUUU. AAAAAA. ATTY. YYYY.
SSSS. UUUU. UUUU. BBB. BBB. UUU. UUUUUUU. UUUU. AAAAAA. ATTY. YYYY.
SSSSSSS. UUUU. UUUU. DDDDDDDDD. UUUUUUUUUU. UUU. AAAAAA. YYYYYY.
SSSSSSSSSS. UUUU. UUUU. DDDDDDDDD. UUUUUUU. UUUUUUU. AAAA. AAAA. YYYYY.
SSSSSS. UUUU. UUUU. BBBBBBBBBB. UUUUUUU. UUUUUUU. AAAAAA. YYYYY.
SSSS. UUUU. UUUU. BBB. BBB. UUUUUUU. UUUUUUU. AAAAAA. YYYYY.
SSSS. SSSS. UUUU. UUUU. DDD. DDD. UUUUUUU. UUUUUUU. AAAAAA. YYYYY.
SSSSSSSSSSSS. UUUUUUUUUU. BBBBBBBBBB. UUUUU. UUUUU. UAAA. AAAA. YYYYY.
SSSSSSSSSS. UUUUUUUU. DDDDDDDDD. UUUUU. UUUUU. UAAA. AAAA. YYYYY.
SSSSSSSS. UUUUUUU. BBBBBBBBBB. UUUUU. UUUUUUUAAA. AAAA. YYYYY.

Welcome to Subway eatfresh do you want choose our meal or custom sandwich?
|: custom
You have chosen custom.
Do you have any perferences such as vegan or healthy or none?
|: vegan.
You have chosen vegan.
Which sandwich bread options would you like to pick? We have Italian or Wheat.
|: italian.
You have chosen italian.
Which seled options would you like to pick? We have Cucumbers or Tomatoes or Red Onions or none.
|: red onions
You have chosen red onions.
Which type of sauce would you like to have? We have Chipotle Southwest or Honey Mustard or none.
|: chipotle.
You have chosen chipotle.
Sorry 'chipotle' is a invalid input. please re-key the correct option again.
|: chipotle southwest.
You have chosen chipotle southwest.
Do you want to top up to Extreme sandwich size? y/n
|: n.
You have chosen n.
Do you want to add any ala-carte such as Cookies or none?
|: none.
You have chosen none.
Your Order is custom sandwich set, italian sandwich, redonions meat, none salad, chipotlesouthwest sauce. In addition, nonea top up and none side.
Total Price would be $3.9.
Please confirna your order. y/n?
|: y.
You have chosen y.
Would you like to pay in cash or credit_card?
|: cash.
You have chosen cash.
```

Note : there is no meat option since user choose vegan preferences.

The queries execute in sequence order and the list of choice is dependent on user options. Note that if the user key in the wrong input it will re-prompt the user again. This is not available in the web version since it is using define respond value from images as choices.

Sample 2

```

Welcome to Subway eatfresh do you want choose our meal or custom sandwich?
|: custom.
You have chosen custom.
Do you have any perferences such as vegan or healthy or none?
|: none.
You have chosen none.
Which sandwich bread options would you like to pick? We have Italian or Wheat or Honey Oat.
|: wheat.
You have chosen wheat.
Which meat options would you like to pick? We have Bacon or Chicken Breast or Tuna or none.
|: none.
You have chosen none.
Which seled options would you like to pick? We have Cucumbers or Tomatoes or Red Onions or none.
|: none.
You have chosen none.
Which type of sauce would you like to have? We have Chipotle Southwest or Honey Mustard or Savoury Caesar or none.
|: none.
You have chosen none.
Do you want to top up to Extreme sandwich size? y/n
|: extreme.
You have chosen extreme.
Sorry 'extreme' is a invalid input. please re-key the correct option again.
|: y.
You have chosen y.
Do you want to add any ala-certe such as Cookies or Hashbrowns or none?
|: none.
You have chosen none.
Your Order is custom sandwich set, wheat sandwich, none meat, none salad, none sauce. In addition, extreme top up and none side.
Total Price would be $5.5.
Please confirna your order. y/n?
|: y.
You have chosen y.
Would you like to pay in cash or credit_card?
|: credit_card.
You have chosen credit_card.
Your sandwich is preparing. please wait for 3 minutes! Hope you enjoy our service and have a nice day!
```

Note : Sample 2 has shown all possible options as user choose 'none' preferences as compare to Sample 1 where user have chosen 'vegan' as preferences which result in reduce of certain choices

Key features that are exclusive in prolog only version as follows:

- JSON Libraries

- o Getting the keys and values from 'SSI_DB.json'.

```
% Read Data From JSON To Formulate Facts
get_dict_from_json_file(Key,Value) :-
    FPath = 'SSI_DB.json',
    open(FPath, read, Stream),
    json_read_dict(Stream, Data),
    close(Stream),
    get_dict(Key,Data,Value),
    format("\nData Retrieved <-w> From Database Successfully\n", [Key]), nl.
% Syntax : get_dict(Key,JSON Data, Value)

get_dict_from_json_file(side,ValueS),
get_dict_from_json_file(perferences,ValueP),
get_dict(promotion,ValueH,ValuePromotion),
get_dict(classic,ValueH,ValueClassic),
% get side key
% get perferences key
% get promotion key
% get classic key

assertz(topup(TopUp_NList)),
assertz(side(Side_NList)),
assertzPrice(Promotion_NList,Promotion_PList),
assertzPrice(Classic_NList,Classic_PList),
assertzPrice(Sandwich_NList,Sandwich_PList),
% Learn TopUp name Facts
% Learn Side name Facts
% Learn the prices of individual food elements

% Learn the price of each element of food.
assertzPrice([],[]),
assertzPrice([A|_],[_]) :-
    assertzPrice(A,C),
    assertzPrice(B,D).
```

- o Keys and values turn into facts through assertz function.

- List Methods

- o Check if user input exists in the facts using list methods such as member(?ele,+List)

```
% check if user input exist in the fact of list.
checklist(Z,X) :-
    X = 'abort' -> abort;
    X = 'n';
    X = 'y';
    X = 'none';
    factlist(Z,L),
    member(Y,L), % print(X),
    normalizeVar(Y,N,atom),
    (X = N,nl,1);
    sorryMessage(X, false).

% check if order exist of user input.
% allow admin to terminate the system
% if user input no
% if user input yes
% if user choose none as option
% call the list of fact Z and return the proper list
% for every member/element of the list in the order

% check if user option exist in the order, list double quotes are removed to check against user input.
% show invalid message and return false if user input does not exist.
```

- Backtracking

- o If premises below repeat return a false value then backtrack from 'repeat' keyword and continue the next predicate if there is any.

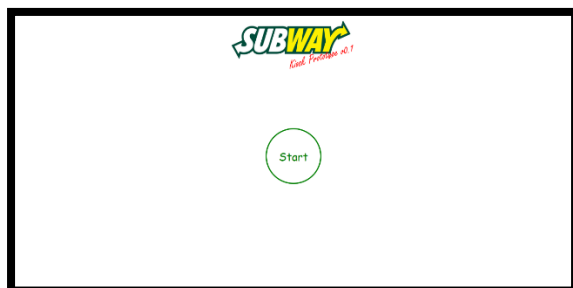
```
write('Welcome to Subway eatfresh do you want choose our'),
order(List),
showList(List),
write('sandwich?'), nl,
repeat,
choose(Option),
checklist(order,Option),
assertz(orderOption(Option)),
ask(1).

% get the options base on the facts
% display the options available
% reprompt user for Option again if invalid input
% call to prompt for user input
% check if user input is valid
% insert the new fact base on user option
```

Prolog + Javascript

Below are all the interfaces of the program and briefly descript how individual interface transit from one screen to another screen.

[1] Initial Screen



[2] Order Options Screen



[1] When the user (first enter/refresh/selected 'no' in confirmation screen) will display this screen. The user has to press start to execute the 1st query. All prolog initialize code will be loaded such as the dynamic facts and clear any past selection such selected fact, invoice fact, choices and price facts.

For example :

```
% Declare dynamic memory
% Clear previous assertz dynamic facts (if any) :- dynamic selected/2.
:- retractall(selected(_,_)).
:- retractall(invoice(_)).
:- dynamic custom/2.
:- dynamic meal/2.
```

[2] Before rendering the screen, 1st check ensures no selected facts on **order**, get the facts from the **order**. The **order** facts will be used to render the options on the interface such as option text and images. Next, the queries will roll out the question in a sequence manner. Next, all the text rendering on the interface will be filtered and convert to speech. The program will wait for the user to respond by clicking on the images. When images are clicked, it will trigger a response with defined value such that it guarantees matches one of the options in the **order** list. Next, this will generate new selected **order** facts through prolog assertz function. Lastly, this will invoke the next options when all previous premises are true.

(Note: The bolded 'order' can be replaced with most other options as the logic goes similarly. Therefore the description below will highlight additional premises before moving to the next option.)

- The prolog code that describe the above [2] screen.

```
% All various options user can choose base on developing facts and options chosen by user
options(order) :-
    not(selected(_,order)),           % if order selection does not exist
    order(X),                         % get facts from order
    term_to_atom(X, Prompt),          % convert term to atom to prompt user
    penguin_input(Prompt, AnswerO),   % call penguin to SSI_prompt.js for input (Query)
    penguin_output(AnswerO),          % call penguin to SSI_prompt.js for output (Queries' answer)
    assertz(selected(AnswerO,order)), % Add queries answer as selected facts
    options(perferences).              % go to next options to choose
```

- The brief javascript code that describe the above [1] and [2] screen.

- o Declaration of various method of handler

```
var penguin = new Penguin({
  ask: 'run',
  onprompt: handlePrompt,
  onoutput: handleOutput,
  onsuccess: handleSuccess,
  onfailure: handleFailure,
  onerror: handleError,
});
run -> options(order).
```

- o Display questions, images options, and text options.

```
if (step == 0) {
  readText = "Welcome to Subway eatfresh, please pick a classify sandwich!";
}
if (step == 1) {
  readText = "Do you have any perferences?";
}
if (step == 2 && jQuery.inArray( 'meal', outputs) != -1) {
  $("#fp").append('');
  $("#mp").append('<div class = onex' + words[i] + '>' + words[i] + '</div>');
}
```

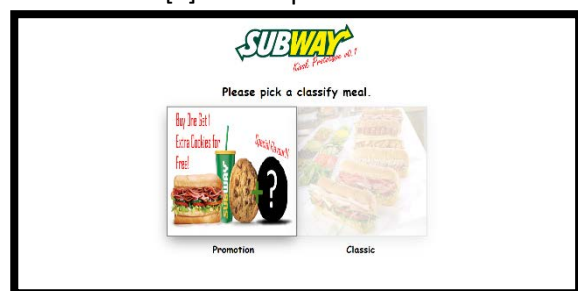
- o Generate Voice

```
function textToSpeech(str) {
  // Invoke speech object
  voice = new SpeechSynthesisUtterance
  voice.rate = 1;
  voice.pitch = 0.5;
  voice.text = str;
  // Speak Operation
  window.speechSynthesis.speak(voice);
}
```

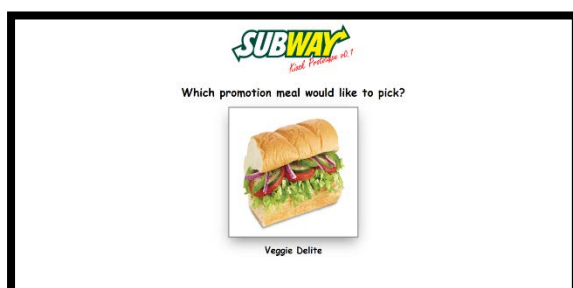
[3] Preferences Option Screen



[4] Meal Option Screen



[5] Promotion Option Screen



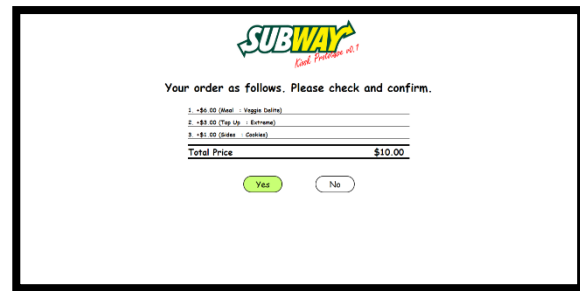
[6] Top Up Option Screen



[7] Sides Option Screen



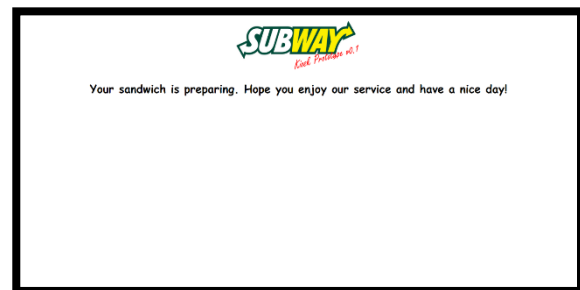
[8] Confirmation Option Screen



[9] Payment Option Screen



[10] End Screen



[3] Before going to the next options, prolog will go through a filter process that updates the new facts base on user preferences. 1st get the preferences list and find a predicate that matches the user preferences. Next, get the list from the preferences and update the list by comparing the existing list. The new list will only consist of both elements that exist in both lists. Before learning the new fact from the list, preferences facts need to be empty first.

[8] To generate the invoice, prolog will get all the past selected options to get the price facts of the individual selected choice. The sum of all individual selected price will be the total price. This will be used as facts and these facts will be rendered accordingly.

```
[8] selected(A,order),
(A = meal -> selected(B,A); true),
((A = custom -> (selected(N1,bread),selected(N2,salad), selected(N3,sauce), selected(N4,meat)));
((B = promotion -> (selected(N1,B), P2 is 0, P3 is 0, P4 is 0));
((B = classic -> (selected(N1,B), P2 is 0, P3 is 0, P4 is 0))),
selected(N5,topup),
selected(N6,side),
price(N1,P1),
(N5 = normal -> (P5 is 0);price(N5,P5)),
(N6 = none -> (P6 is 0);price(N6,P6)),
(A = custom -> (
((N2 = none -> P2 is 0 ); price(N2,P2)),
((N3 = none -> P3 is 0 ); price(N3,P3)),
((N4 = none -> P4 is 0 ); price(N4,P4)));true),
PA is P1+P2,
PB is PA+P3,
PC is PB+P4,
PD is PC+P5,
P is PD+P6,
((A = custom -> (assertz(invoice([N1,P1,N2,P2,N3,P3,N4,P4,N5,P5,N6,P6,total,P])));
(assertz(invoice([N1,P1,N5,P5,N6,P6,total,P])))).
```

```
[3] filter(X):-
perferences(X,L),
custom(bread,L1),
custom(meat,L2),
custom(salad,L3),
custom(sauce,L4),
meal(classic,L5),
meal(promotion,L6),
side(L7),

intersection(L1, L, NL1),
intersection(L2, L, NL2),
intersection(L3, L, NL3),
intersection(L4, L, NL4),
intersection(L5, L, NL5),
intersection(L6, L, NL6),
intersection(L7, L, NL7),

retractall(custom(_,_)),
retractall(meal(_,_)),
retractall(side(_)),

assertz(custom(bread,NL1)),
```

Appendix

References

- <https://pengines.swi-prolog.org/docs/index.html>
- https://en.wikipedia.org/wiki/2019%E2%80%932020_coronavirus_pandemic
- <https://www.subway.com/en-SG>

Disclaimer

The story information provided by me on this document is makeup base on certain facts of the real world, however, I make no representation or warranty of any kind, express or implied, regarding the accuracy, adequacy, validity, reliability, availability or completeness of any information provided in this documentation. This document is meant for assessment and educational purposes only.