BCRITW

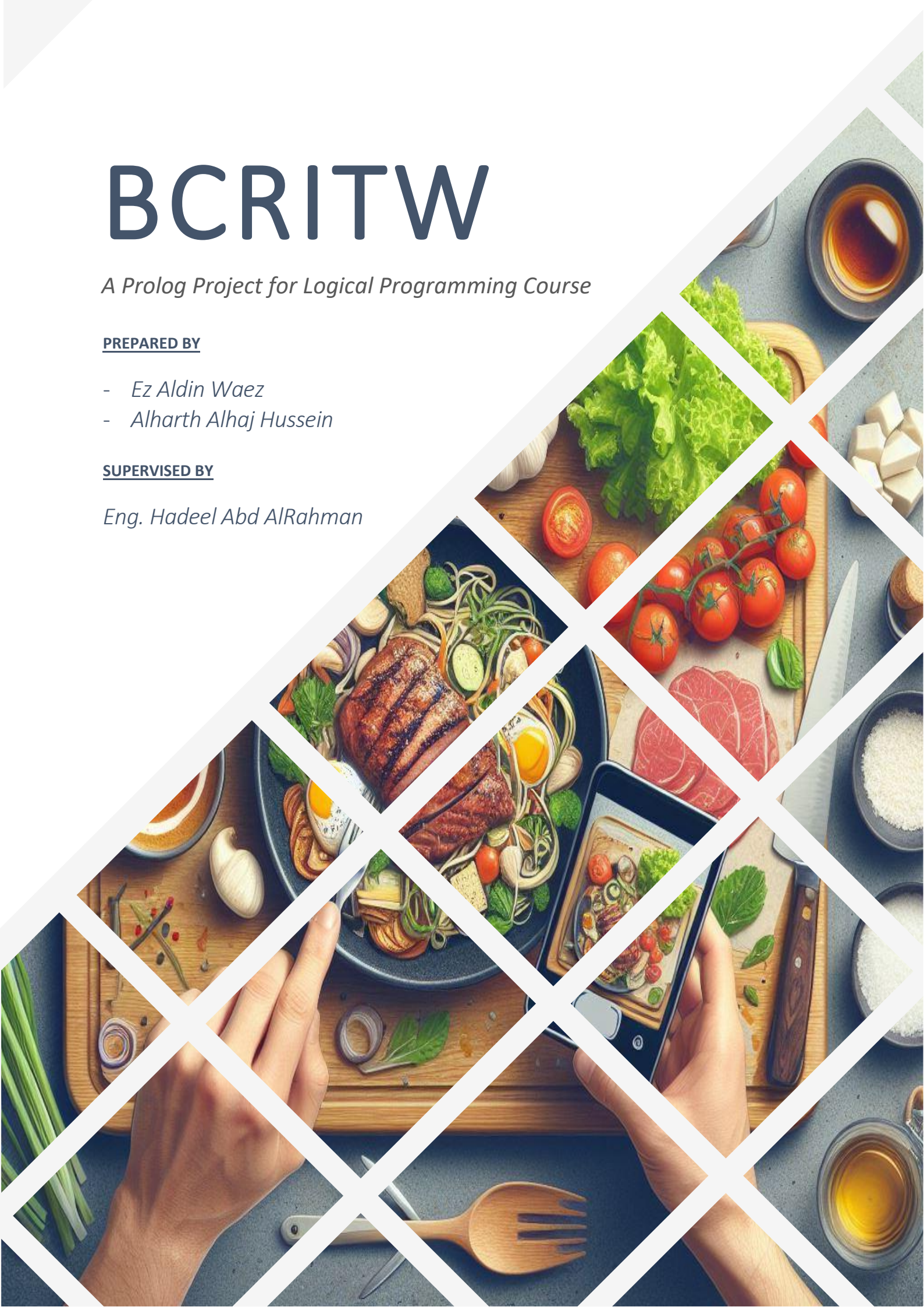
A Prolog Project for Logical Programming Course

PREPARED BY

- *Ez Aldin Waez*
- *Alharth Alhaj Hussein*

SUPERVISED BY

Eng. Hadeel Abd AlRahman



BCRITW

README.md

BCRITW

Best Cooking Recommender In The World.

Just a `Prolog` project for the college.

Introduction

In this CLI-Based program, you are supposed to tell it what are the ingredients you have (by answering yes/no questions), and it'll recommend you some meals to cook with these ingredients.

Good Look!

Requirements

- `SWI-Prolog`

How To Run

Type the following on your terminal:

- `cd <project-folder-path>`
- `swipl main.pl`

(do not forget to replace `<project-folder-path>` with the actual path)

Then type `run.` to run the program.

Copyrights

Made by *Ez Aldin Waez* & *Alharth Alhaj Hussein* ...

meals.pl

```
meals([
    ['Labania', ['Yogurt','Rice','Meat','Spices','Garlic', 'Egges']],
    ['Kebbah', ['Spices', 'Meat', 'Onion', 'Bulgur']],
    ['Bamia', ['Spices', 'Meat', 'Tomato Souce', 'Okra', 'Bread']],
    ['Safargeliah', ['Spices', 'Meat', 'Tomato Souce', 'Quince']],
    ['Mehshi', ['Spices', 'Meat', 'Vegetables', 'Rice']],
    ['Mulukhiyah', ['Spices', 'Meat', 'Garlic', 'Mulukhiyah Leaves']],
```

```
    ['Fasolia', ['Spices', 'Meat', 'Tomato Souce', 'Garlic', 'Beans']],
    ['Mjadarah', ['Onion', 'Rice', 'Lentil']],
    ['Spaghetti', ['Spices', 'Tomato Souce', 'Vegetables',
'Macaroni']],
    ['Yabraq', ['Spices', 'Meat', 'Garlic', 'Rice', 'Grape Leaves']],
    ['Orman-Blaban', ['Spices', 'Meat', 'Butter', 'Yogurt', 'Corn
Starch', 'Egges']]
]).
```

main.pl

```
:- include('meals.pl').

run :-
    print_welcome,
    meals(Meals),
    check_meals(Meals),
    exit.

print_welcome :-
    write('*****************************'), nl,
    write('*        B-C-R-I-T-W        *'), nl,
    write('*****************************'), nl,
    nl.

check_meals([]) :-
    write('You cannot cook anything else!'), nl,
    nl.
check_meals([H|T]) :-
    check_meal(H),
    check_meals(T).

check_meal([MealName, Ingredients]) :-
    check_ingredients(Ingredients),
    nl,
    write('* You can cook '), write(MealName), write(' *'), nl,
    nl,
    ask_to_complete.

check_meal(_). % it will always return `true`, even if
`check_ingredients` returns `false`.

check_ingredients([]).
check_ingredients([H|T]) :-
    check_ingredient(H)
    check_ingredients(T).
```

```
35
36  check_ingredient(Ingredient) :-
37      yes(Ingredient) -> true ;
38      no(Ingredient) -> fail ;
39      ask_about(Ingredient).
40
41  ask_about(Ingredient) :-
42      write('Do you have '), write(Ingredient), write('? [y/n]: '),
43      read(Reply),
44      (
45          (Reply == y; Reply == yes) -> assert(yes(Ingredient)), true ;
46          (Reply == n; Reply == no) -> assert(no(Ingredient)), fail ;
47          write('Invalid answer! (write \'yes.\' or \'no.\')'), nl,
48          ask_about(Ingredient)
49      ).
50
51  ask_to_complete :-
52      write('* Do you want to complete? [y/n]: '),
53      read(Reply),
54
55          (Reply == y; Reply == yes) -> true ;
56          (Reply == n; Reply == no) -> exit ;
57          write('Invalid answer! (write \'yes.\' or \'no.\')'), nl,
58          ask_to_complete
59      ).
60
61  :- dynamic yes/1, no/1.
62
63  undo :-
64      retract(yes(_)),
65      fail.
66  undo :-
67      retract(no(_)),
68      fail.
69  undo.
70
71  exit :-
72      undo,
73      nl,
74      write('*****************************'), nl,
75      write('* Thanks For Using This App *'), nl,
76      write('*****************************'), nl,
77      nl,
78      halt.
```