RESTAURANT INFORMATION SYSTEM

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 admsys.c File Reference

This file contains all function that are used in the admin mode.

```
#include "smartsys.h"
```

Functions

- int admin_interface (MYSQL *connect)

  This function provides interactivity between admin and the system. This driver fucntion calls other functions according to the user's input.
- int add_dish (MYSQL *connect)

  This function enables admin to add a new dish to the menu, admin can choose name, price, dish type and description(optional)
- int delete_dish_menu (MYSQL *connect)

  This function enables admin to permanently delete all information about a dish from the dish table.
- int un_pause_dish (MYSQL *connect, int mode)

  *This function sets attribute 'pause' in the dish table to **true** when user wants to temporary remove a dish from the menu or to **false** when user wants to resume a previously paused dish.*
- void show_income (MYSQL *connect)

  This function asks user over what period of time they want to display the income and displays it.

2.1.1 Detailed Description

This file contains all function that are used in the admin mode.

Author

Sokolovskii Vladislav

Date

20 Jan 2020

Contained functions enable admin to add a dish to the menu, set its name, price, type and description(optional). Also, admin can delete all information about a particular dish from the dish table, temporary remove a dish from the menu and retrieve it later, display income over a particular period of time (day, month, year).

2.1.2 Function Documentation

2.1.2.1 add_dish()

```
int add_dish (
            MYSQL * connect )
```

This function enables admin to add a new dish to the menu, admin can choose name, price, dish type and description(optional)

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function return SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

Postcondition

A new row to the dish table will be added

2.1.2.2 admin_interface()

```
int admin_interface (
            MYSQL * connect )
```

This function provides interactivity between admin and the system. This driver fucntion calls other functions according to the user's input.

Precondition

The program must be executed with 'admin' parameter

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns **SUCCESS** (1) if no error occured. Otherwise returns **UNSUCCESS** (0)

2.1.2.3 delete_dish_menu()

```
int delete_dish_menu (
            MYSQL * connect )
```

This function enables admin to permanently delete all information about a dish from the dish table.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

Postcondition

A row from the dish table will be deleted

2.1.2.4 show_income()

```
void show_income (
            MYSQL * connect )
```

This function asks user over what period of time they want to display the income and displays it.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database

2.1.2.5 un_pause_dish()

```
int un_pause_dish (
            MYSQL * connect,
            int mode )
```

This function sets attribute 'pause' in the dish table to **true** when user wants to temporary remove a dish from the menu or to **false** when user wants to resume a previously paused dish.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*mode*	- This is the chosen behavior index (pause or resume 1/0)

Returns

Return value is SUCCESS (1) if a dish was successfully un/paused. Otherwise return value is UNSUCCESS (0).

2.2 smartsys.c File Reference

This is the main file of the program containing the main function and some functions that both modes (admin/user mode) share.

```
#include "smartsys.h"
```

Functions

- int main (int argc, char ∗argv[ ])
- int args_check (int argc, char ∗∗argv)

  This function checks arguments from the command line and returns a number corresponding to one of the behaviour indexes.
- void main_menu (char ∗name, int mode)

  This function prints out one type of the main menu (visitor or admin) according to the passed arguments. If the function is called for the first time in visitor mode the personilized welcome message will be printed.
- void return_to_main_menu (int mode)

  This function asks user if they want to return to the main menu.
- void display_dish_menu (MYSQL ∗connect, int type)

  This function prints out all dishes included to the menu in categories according to their type.
- void print_results (MYSQL ∗connect, int cols_num)

  This function prints out context of every column of selected row/rows.
- int unique_random_num (MYSQL ∗connect, int lower, int upper, char ∗table_name, char ∗attribute_name)

  This function generates a number on defined interval and makes sure that this number is unique within a particular table.
- void get_option (int ∗option)

  This function assignes 'option' to the chosen index behavior. The function will ask a user to enter one of the options as long as the entered value will not satisfy any of the given options.
- void yes_no_answer (char ∗answer)

  This function assignes a passed char variable to the one of the options 'Y' or 'N' depending on the user's input.
- int format_check (int format, char ∗string)

  This function checks if given string corresponds to the chosen format.
- bool null_selected (MYSQL ∗connect, unsigned id, char ∗table_name, char ∗attribute_name)

  This function checks if particular attribute value exist in a particular table.
- void remove_new_line (char ∗string)

  This function deletes the newline character from the given string.
- void install_db (MYSQL ∗connect)

  This function creates tables that are necessary for proper work of the program.
- void reg (MYSQL ∗connect)

  This function enables user to create an account in the system. The function calls unique_random_number to generate unique user id.
- void print_usage_msg ()

  This function prints out the usage message.

Variables

- char * server = "localhost"
- char * user = "root"
- char * password = "04072010"
- char * database = "rest_IS"
- unsigned int port = 3306
- unsigned int flag = 0

2.2.1 Detailed Description

This is the main file of the program containing the main function and some functions that both modes (admin/user mode) share.

Author

Sokolovskii Vladislav

Date

20 Jan 2020

The file includes the functions that are used in both modes or function that can work differently depending on the current work mode. Also, there are global variables that are necessary for the connection to the database.

2.2.2 Function Documentation

2.2.2.1 args_check()

```
int args_check (
            int argc,
            char ** argv )
```

This function checks arguments from the command line and returns a number corresponding to one of the behaviour indexes.

Parameters

in	*argc*	- This is the number of arguments passed to the command line
in	*argv*	- This is array of arguments passed to the command line

Returns

This function returns the index of expected behavior according to the argument passed to the command line

Postcondition

One of the possible program flows will be chosen

2.2.2.2 display_dish_menu()

```
void display_dish_menu (
            MYSQL * connect,
            int type )
```

This function prints out all dishes included to the menu in categories according to their type.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*type*	- This is one of the values of ennumeration of types of the dishes

2.2.2.3 format_check()

```
int format_check (
            int format,
            char * string )
```

This function checks if given string corresponds to the chosen format.

Parameters

in	*format*	- This is a value from formats enumeration representing a particular fomat (date, number, dish_type)
in	*string*	- This is a string to be checked

Returns

This function returns SUCCESS (1) in the case when the string corresponds to the format. Otherwise return value is UNSUCCESS (0)

2.2.2.4 get_option()

```
void get_option (
            int * option )
```

This function assignes 'option' to the chosen index behavior. The function will ask a user to enter one of the options as long as the entered value will not satisfy any of the given options.

Parameters

in	*option*	- This is pointer to the option value, which is used to choose the flow of the program

2.2.2.5 install_db()

```
void install_db (
            MYSQL * connect )
```

This function creates tables that are necessary for proper work of the program.

Precondition

The program must be executed with '–install' parametr

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

The program will work correctly

2.2.2.6 main()

```
int main (
            int argc,
            char * argv[] )
```

2.2.2.7 main_menu()

```
void main_menu (
            char * name,
            int mode )
```

This function prints out one type of the main menu (visitor or admin) according to the passed arguments. If the function is called for the first time in visitor mode the personilized welcome message will be printed.

Parameters

in	*name*	- This is the name of the visitor.
in	*mode*	- This is the current mode of the program (visitor or admin)

2.2.2.8 null_selected()

```
bool null_selected (
            MYSQL * connect,
            unsigned id,
            char * table_name,
            char * attribute_name )
```

This function checks if particular attribute value exist in a particular table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*id*	- This is a numerical attribute value to be checked
in	*table_name*	- This is the name of the table in which a number must be checked
in	*attribute_name*	- This is the name of the attribute of a table

Returns

This function returns **false** if id value of attribute_name exists in table_name. Otherwise return value is **true**

2.2.2.9 print_results()

```
void print_results (
            MYSQL * connect,
            int cols )
```

This function prints out context of every column of selected row/rows.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*cols*	- This is the number of colums to print

2.2.2.10 print_usage_msg()

```
void print_usage_msg (
            void  )
```

This function prints out the usage message.

Precondition

The program must be executed with '–help' parameter

2.2.2.11 reg()

```
void reg (
            MYSQL * connect )
```

This function enables user to create an account in the system. The function calls unique_random_number to generate unique user id.

Parameters

in	*correct*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

A new row in visitor table will be created

2.2.2.12 remove_new_line()

```
void remove_new_line (
            char * string )
```

This function deletes the newline character from the given string.

Parameters

in	*string*	- This is the string from where the newline character must be deleted

2.2.2.13 return_to_main_menu()

```
void return_to_main_menu (
            int mode )
```

This function asks user if they want to return to the main menu.

Parameters

in	*mode*	- This is the current mode of the program (visitor or admin)

Postcondition

The main_menu function will be called.

2.2.2.14 unique_random_num()

```
int unique_random_num (
            MYSQL * connect,
            int lower,
            int upper,
            char * table_name,
            char * attribute_name )
```

This function generates a number on defined interval and makes sure that this number is unique within a particular table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*lower*	- This is the lower bound of the interval
in	*upper*	- This is the upper bound of the interval
in	*table_name*	- This is the name of the table in which the generated number must be unique
in	*attribute_name*	- This is the attribute name that will be assigned to the generated number

2.2.2.15 yes_no_answer()

```
void yes_no_answer (
            char * answer )
```

This function assignes a passed char variable to the one of the options 'Y' or 'N' depending on the user's input.

Parameters

in	*answer*	- This is the pointer to the answer variable

2.2.3 Variable Documentation

2.2.3.1 database

```
char* database = "rest_IS"
```

2.2.3.2 flag

```
unsigned int flag = 0
```

2.2.3.3 password

```
char* password = "04072010"
```

2.2.3.4 port

```
unsigned int port = 3306
```

2.2.3.5 server

```
char* server = "localhost"
```

2.2.3.6 user

```
char* user = "root"
```

2.3 smartsys.h File Reference

This header file contains declarations of all functions, enumerations and macroses that program use.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>
#include <mysql.h>
#include <time.h>
```

Macros

- #define smartsys_h
- #define DATELEN 11
- #define MAXSTR 32
- #define MAXLEN 256
- #define MAX_TRIES 5
- #define SUCCESS 1
- #define UNSUCCESS 0
- #define MAXID 999
- #define MENUWIDTH 66
- #define ACTIVE 0
- #define PAUSE 1
- #define PAID 1
- #define BAD_FORMAT -1

Enumerations

- enum work_modes {
  idle, visitor_mode = 11, admin_mode = 22, installation,
  registration, help }
- enum formats { type = 60, number, date_format }
- enum types_of_dishes {
  drink = 12, first, main_dish, dessert,
  snack, all }
- enum admin_main_menu {
  dish_menu = 1, add_dish_menu, del_dish_menu, pause_resume_dish,
  show_income_per, end_admin_session }
- enum user_main_menu { show_menu = 1, create_meal, show_prev_meal, end_usr_session }
- enum sub_menu_options {
  make_order = 1, edit_order, back, add_to_meal = 1,
  del_from_meal }
- enum sub_menu_modes { order_making = 50, editing }

Functions

- int admin_interface (MYSQL *connect)

  This function provides interactivity between admin and the system. This driver fucntion calls other functions according to the user's input.
- int args_check (int argc, char **argv)

  This function checks arguments from the command line and returns a number corresponding to one of the behaviour indexes.
- int log_in (MYSQL *connect, bool payment)

  This function asks for user's id and password in the system, if given data was found in the database the function returns SUCCESS (1). Otherwise if user enter bad data five times the function will return UNSUCCESS (0). Also, the function is user of confirmation of a payment.
- void main_menu (char *name, int mode)

  This function prints out one type of the main menu (visitor or admin) according to the passed arguments. If the function is called for the first time in visitor mode the personilized welcome message will be printed.
- void return_to_main_menu (int mode)

  This function asks user if they want to return to the main menu.
- void yes_no_answer (char *answer)

  This function assignes a passed char variable to the one of the options 'Y' or 'N' depending on the user's input.
- void display_dish_menu (MYSQL *connect, int type)

  This function prints out all dishes included to the menu in categories according to their type.
- int add_dish (MYSQL *connect)

  This function enables admin to add a new dish to the menu, admin can choose name, price, dish type and description(optional)
- int delete_dish_menu (MYSQL *connect)

  This function enables admin to permanently delete all information about a dish from the dish table.
- int delete_dish_meal (MYSQL *connect, int meal_id)

  This function enables a visitor to delete a dish from their order.
- void print_results (MYSQL *connect, int cols)

  This function prints out context of every column of selected row/rows.
- int unique_random_num (MYSQL *connect, int lower, int upper, char *table_name, char *attribute_name)

  This function generates a number on defined interval and makes sure that this number is unique within a particular table.
- int un_pause_dish (MYSQL *connect, int mode)

*This function sets attribute 'pause' in the dish table to **true** when user wants to temporary remove a dish from the menu or to **false** when user wants to resume a previously paused dish.*

- int create_order (MYSQL *connect)

  This function inserts a new row to the meal table.

- int fill_order (MYSQL *connect, int meal_id)

  This function ask user what dishes do they want to include in this particular meal then user is asked if they want to continue choosing, in the case of positive feedback cycle repeats, otherwise the process of making an order will be ended.

- int display_order (MYSQL *connect, int order_id)

  This function prints out name id and price of every dish that was included to the particular meal. On the end total amount is printed.

- int display_prev_orders (MYSQL *connect)

  This function displays id, date and total price of all the previous orders that user has ever had at the restaurant. And asks user for the id of the meal to display.

- int confirm_order (MYSQL *connect, int meal_id, int total)

  This function sets the date and time when the order was made and total order amount.

- int edit_meal (MYSQL *connect, int meal_id)

  This function enables visitor to edit their meal (add or delete a dish)

- void sub_menu (int mode)

  This function prints out a type of sub menu according to the passed argument.

- void get_option (int *option)

  This function assignes 'option' to the chosen index behavior. The function will ask a user to enter one of the options as long as the entered value will not satisfy any of the given options.

- void pay (MYSQL *connect, bool *order_status, int meal_id)

  This function simulates payment process and changes the value order_status in meal table to PAID if log_in function returns the SUCCESS (1) value.

- void show_income (MYSQL *connect)

  This function asks user over what period of time they want to display the income and displays it.

- int format_check (int format, char *string)

  This function checks if given string corresponds to the chosen format.

- void remove_new_line (char *string)

  This function deletes the newline character from the given string.

- bool null_selected (MYSQL *connect, unsigned id, char *table_name, char *attribute_name)

  This function checks if particular attribute value exist in a particular table.

- int user_interface (MYSQL *connect)

  This function provides interactivity between user and the system. This driver fucntion calls other functions according to the user's input.

- void install_db (MYSQL *connect)

  This function creates tables that are necessary for proper work of the program.

- void reg (MYSQL *connect)

  This function enables user to create an account in the system. The function calls unique_random_number to generate unique user id.

- void print_usage_msg (void)

  This function prints out the usage message.

Variables

- MYSQL_RES * result
- MYSQL_ROW row

2.3.1 Detailed Description

This header file contains declarations of all functions, enumerations and macroses that program use.

Author

Sokolovskii Vladislav

Date

20 Jan 2020

2.3.2 Macro Definition Documentation

2.3.2.1 ACTIVE

```
#define ACTIVE 0
```

Zero value indicating that dish is avaliable in the menu

2.3.2.2 BAD_FORMAT

```
#define BAD_FORMAT -1
```

Negative return value indication that given value do not correspond with the pattern

2.3.2.3 DATELEN

```
#define DATELEN 11
```

This is the library which supportes MySQL C API and provides access to the database content The length of date in YYYY-MM-DD format + newline character

2.3.2.4 MAX_TRIES

```
#define MAX_TRIES 5
```

Maximum number of tries to log in

2.3.2.5 MAXID

```
#define MAXID 999
```

Maxim possible id number

2.3.2.6 MAXLEN

```
#define MAXLEN 256
```

Maximum length of a query

2.3.2.7 MAXSTR

```
#define MAXSTR 32
```

Maximum length of a string

2.3.2.8 MENUWIDTH

```
#define MENUWIDTH 66
```

The width of the "window" of app

2.3.2.9 PAID

```
#define PAID 1
```

Non zero value indication that the order was paid

2.3.2.10 PAUSE

```
#define PAUSE 1
```

Non zero value indication that dish is paused

2.3.2.11 smartsys_h

```
#define smartsys_h
```

2.3.2.12 SUCCESS

```
#define SUCCESS 1
```

Non-zero return value indicating successful ending of a function

2.3.2.13 UNSUCCESS

```
#define UNSUCCESS 0
```

Zero return value indicating unsuccessful ending of a function

2.3.3 Enumeration Type Documentation

2.3.3.1 admin_main_menu

```
enum admin_main_menu
```

The ennumeraton of admin main menu options

Enumerator

dish_menu	Display the menu option
add_dish_menu	Add a dish to the menu option
del_dish_menu	
pause_resume_dish	Delete a dish from the menu option Temporary remove or resume a dish option
show_income_per	Show income option
end_admin_session	End session option

2.3.3.2 formats

`enum formats`

The ennumeration of possible string formats

Enumerator

type	Type of a dish format
number	Number format
date_format	Date YYYY-MM-DD format

2.3.3.3 sub_menu_modes

`enum sub_menu_modes`

The ennumeration of types of user sub menus

Enumerator

order_making	Make the order sub menu
editing	Edit the order sub menu

2.3.3.4 sub_menu_options

`enum sub_menu_options`

The ennumerations of user sub menu options

Enumerator

make_order	

Enumerator

edit_order	Edit the order information option
back	Back to the main menu option
add_to_meal	Add a dish to the meal option
del_from_meal	Delete a dish from the meal option

2.3.3.5 types_of_dishes

```
enum types_of_dishes
```

The ennumeration of types of the dishes

Enumerator

drink	
first	The first course type
main_dish	The main dish type
dessert	Dessert dish type
snack	Snack dish type
all	Dish type that includes all types of dishes

2.3.3.6 user_main_menu

```
enum user_main_menu
```

The ennumeration of user main menu options

Enumerator

show_menu	Display the menu option
create_meal	Make an order option
show_prev_meal	Show previous meals option
end_usr_session	End session option

2.3.3.7 work_modes

```
enum work_modes
```

The ennumeration of possible program behaviours

Enumerator

idle	
visitor_mode	Visitor behavior index
admin_mode	Admin behavior index
installation	Installation behavior index
registration	Registration of a new user behavoir index
help	Usage message behavior index

2.3.4 Function Documentation

2.3.4.1 add_dish()

```
int add_dish (
            MYSQL * connect )
```

This function enables admin to add a new dish to the menu, admin can choose name, price, dish type and description(optional)

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function return SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

Postcondition

A new row to the dish table will be added

2.3.4.2 admin_interface()

```
int admin_interface (
            MYSQL * connect )
```

This function provides interactivity between admin and the system. This driver fucntion calls other functions according to the user's input.

Precondition

The program must be executed with 'admin' parameter

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns **SUCCESS** (1) if no error occured. Otherwise returns **UNSUCCESS** (0)

2.3.4.3 args_check()

```
int args_check (
            int argc,
            char ** argv )
```

This function checks arguments from the command line and returns a number corresponding to one of the behaviour indexes.

Parameters

in	*argc*	- This is the number of arguments passed to the command line
in	*argv*	- This is array of arguments passed to the command line

Returns

This function returns the index of expected behavior according to the argument passed to the command line

Postcondition

One of the possible program flows will be chosen

2.3.4.4 confirm_order()

```
int confirm_order (
            MYSQL * connect,
            int meal_id,
            int total )
```

This function sets the date and time when the order was made and total order amount.

Precondition

A user chose 'confirm my meal' option in sub menu of making order

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order user wants to confirm
in	*total*	- This is the total price of the particular order

Returns

This function returns the id of the meal user wants to display

Postcondition

After finishing the session a user will be asked to pay

2.3.4.5 create_order()

```
int create_order (
            MYSQL * connect )
```

This function inserts a new row to the meal table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

fill_order funcion will be called

Returns

The function returns the id of newly created order

2.3.4.6 delete_dish_meal()

```
int delete_dish_meal (
            MYSQL * connect,
            int meal_id )
```

This function enables a visitor to delete a dish from their order.

Precondition

edit_meal function must be called before

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order we edit now

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

Postcondition

A dish from the particular order will be deleted

2.3.4.7 delete_dish_menu()

```
int delete_dish_menu (
            MYSQL * connect )
```

This function enables admin to permanently delete all information about a dish from the dish table.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

Postcondition

A row from the dish table will be deleted

2.3.4.8 display_dish_menu()

```
void display_dish_menu (
            MYSQL * connect,
            int type )
```

This function prints out all dishes included to the menu in categories according to their type.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*type*	- This is one of the values of ennumeration of types of the dishes

2.3.4.9 display_order()

```
int display_order (
            MYSQL * connect,
            int order_id )
```

This function prints out name id and price of every dish that was included to the particular meal. On the end total amount is printed.

Parameters

	connect	- This is a MYSQL pointer which provides the connection to the database
in	*meal↩ _id*	- This is the unique identificator of the order we want to display

Returns

This function returns value that is equal to the total amount of the particular order

2.3.4.10 display_prev_orders()

```
int display_prev_orders (
            MYSQL * connect )
```

This function displays id, date and total price of all the previous orders that user has ever had at the restaurant. And asks user for the id of the meal to display.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns the id of the meal to display

Postcondition

The display_order function will be called

2.3.4.11 edit_meal()

```
int edit_meal (
            MYSQL * connect,
            int meal_id )
```

This function enables visitor to edit their meal (add or delete a dish)

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order user wants to edit

Returns

This function return SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

2.3.4.12 fill_order()

```
int fill_order (
            MYSQL * connect,
            int meal_id )
```

This function ask user what dishes do they want to include in this particular meal then user is asked if they want to continue choosing, in the case of positive feedback cycle repeats, otherwise the process of making an order will be ended.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order we want to fill with dishes

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

2.3.4.13 format_check()

```
int format_check (
            int format,
            char * string )
```

This function checks if given string corresponds to the chosen format.

Parameters

in	*format*	- This is a value from formats enumeration representing a particular fomat (date, number, dish_type)
in	*string*	- This is a string to be checked

Returns

This function returns SUCCESS (1) in the case when the string corresponds to the format. Otherwise return value is UNSUCCESS (0)

2.3.4.14 get_option()

```
void get_option (
            int * option )
```

This function assignes 'option' to the chosen index behavior. The function will ask a user to enter one of the options as long as the entered value will not satisfy any of the given options.

Parameters

in	*option*	- This is pointer to the option value, which is used to choose the flow of the program

2.3.4.15 install_db()

```
void install_db (
            MYSQL * connect )
```

This function creates tables that are necessary for proper work of the program.

Precondition

The program must be executed with '–install' parametr

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

The program will work correctly

2.3.4.16 log_in()

```
int log_in (
            MYSQL * connect,
            bool payment )
```

This function asks for user's id and password in the system, if given data was found in the database the function returns SUCCESS (1). Otherwise if user enter bad data five times the function will return UNSUCCESS (0). Also, the function is user of confirmation of a payment.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*payment*	- This is a boolean value which indicates if the function is used for confirmation of a payment or for log in.

Returns

This function returns SUCCESS (1) if user gave correct id and password. Otherwise return value is UNSU↩ CCESS (0).

Postcondition

Depending on the return value the program will terminate or continue.

2.3.4.17 main_menu()

```
void main_menu (
            char * name,
            int mode )
```

This function prints out one type of the main menu (visitor or admin) according to the passed arguments. If the function is called for the first time in visitor mode the personilized welcome message will be printed.

Parameters

in	*name*	- This is the name of the visitor.
in	*mode*	- This is the current mode of the program (visitor or admin)

2.3.4.18 null_selected()

```
bool null_selected (
            MYSQL * connect,
            unsigned id,
```

```
            char * table_name,
            char * attribute_name )
```

This function checks if particular attribute value exist in a particular table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*id*	- This is a numerical attribute value to be checked
in	*table_name*	- This is the name of the table in which a number must be checked
in	*attribute_name*	- This is the name of the attribute of a table

Returns

This function returns **false** if id value of attribute_name exists in table_name. Otherwise return value is **true**

2.3.4.19 pay()

```
void pay (
            MYSQL * connect,
            bool * order_status,
            int meal_id )
```

This function simulates payment process and changes the value order_status in meal table to PAID if log_in function returns the SUCCESS (1) value.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*order_status*	- This is the value which indicates if order was confirmed or no
in	*meal_id*	- This is the unique identificator of the order user wants to pay for

2.3.4.20 print_results()

```
void print_results (
            MYSQL * connect,
            int cols )
```

This function prints out context of every column of selected row/rows.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*cols*	- This is the number of colums to print

2.3.4.21 print_usage_msg()

```
void print_usage_msg (
            void  )
```

This function prints out the usage message.

Precondition

The program must be executed with '–help' parameter

2.3.4.22 reg()

```
void reg (
            MYSQL * connect )
```

This function enables user to create an account in the system. The function calls unique_random_number to generate unique user id.

Parameters

in	*correct*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

A new row in visitor table will be created

2.3.4.23 remove_new_line()

```
void remove_new_line (
            char * string )
```

This function deletes the newline character from the given string.

Parameters

in	*string*	- This is the string from where the newline character must be deleted

2.3.4.24 return_to_main_menu()

```
void return_to_main_menu (
            int mode )
```

This function asks user if they want to return to the main menu.

Parameters

in	*mode*	- This is the current mode of the program (visitor or admin)

Postcondition

The main_menu function will be called.

2.3.4.25 show_income()

```
void show_income (
            MYSQL * connect )
```

This function asks user over what period of time they want to display the income and displays it.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database

2.3.4.26 sub_menu()

```
void sub_menu (
            int mode )
```

This function prints out a type of sub menu according to the passed argument.

Parameters

in	*mode*	- This is the index of particular sub menu

2.3.4.27 un_pause_dish()

```
int un_pause_dish (
            MYSQL * connect,
            int mode )
```

This function sets attribute 'pause' in the dish table to **true** when user wants to temporary remove a dish from the menu or to **false** when user wants to resume a previously paused dish.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*mode*	- This is the chosen behavior index (pause or resume 1/0)

Returns

Return value is SUCCESS (1) if a dish was successfully un/paused. Otherwise return value is UNSUCCESS (0).

2.3.4.28 unique_random_num()

```
int unique_random_num (
            MYSQL * connect,
            int lower,
            int upper,
            char * table_name,
            char * attribute_name )
```

This function generates a number on defined interval and makes sure that this number is unique within a particular table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*lower*	- This is the lower bound of the interval
in	*upper*	- This is the upper bound of the interval
in	*table_name*	- This is the name of the table in which the generated number must be unique
in	*attribute_name*	- This is the attribute name that will be assigned to the generated number

2.3.4.29 user_interface()

```
int user_interface (
            MYSQL * connect )
```

This function provides interactivity between user and the system. This driver fucntion calls other functions according to the user's input.

Precondition

The program must be executed with 'order' parameter

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns **SUCCESS** (1) if no error occured. Otherwise returns **UNSUCCESS** (0)

2.3.4.30 yes_no_answer()

```
void yes_no_answer (
            char * answer )
```

This function assignes a passed char variable to the one of the options 'Y' or 'N' depending on the user's input.

Parameters

in	*answer*	- This is the pointer to the answer variable

2.3.5 Variable Documentation

2.3.5.1 result

```
MYSQL_RES* result
```

2.3.5.2 row

```
MYSQL_ROW row
```

2.4 usersys.c File Reference

This file contains all functions that are used in the user mode.

```
#include "smartsys.h"
```

Functions

- int user_interface (MYSQL *connect)

  This function provides interactivity between user and the system. This driver fucntion calls other functions according to the user's input.
- int log_in (MYSQL *connect, bool payment)

  This function asks for user's id and password in the system, if given data was found in the database the function returns SUCCESS (1). Otherwise if user enter bad data five times the function will return UNSUCCESS (0). Also, the function is user of confirmation of a payment.
- int create_order (MYSQL *connect)

  This function inserts a new row to the meal table.
- int fill_order (MYSQL *connect, int meal_id)

  This function ask user what dishes do they want to include in this particular meal then user is asked if they want to continue choosing, in the case of positive feedback cycle repeats, otherwise the process of making an order will be ended.
- int delete_dish_meal (MYSQL *connect, int meal_id)

  This function enables a visitor to delete a dish from their order.
- int display_order (MYSQL *connect, int order_id)

  This function prints out name id and price of every dish that was included to the particular meal. On the end total amount is printed.
- int display_prev_orders (MYSQL *connect)

  This function displays id, date and total price of all the previous orders that user has ever had at the restaurant. And asks user for the id of the meal to display.
- int confirm_order (MYSQL *connect, int meal_id, int total)

  This function sets the date and time when the order was made and total order amount.
- int edit_meal (MYSQL *connect, int meal_id)

  This function enables visitor to edit their meal (add or delete a dish)
- void sub_menu (int mode)

  This function prints out a type of sub menu according to the passed argument.
- void pay (MYSQL *connect, bool *order_status, int meal_id)

  This function simulates payment process and changes the value order_status in meal table to PAID if log_in function returns the SUCCESS (1) value.

Variables

- bool order_is_confirmed = false
- int current_user_id

2.4.1 Detailed Description

This file contains all functions that are used in the user mode.

Author

Sokolovskii Vladislav

Date

20 Jan 2020

Contained in this file functions enable user to log in make an order, edit it and show previous orders of the user.

2.4.2 Function Documentation

2.4.2.1 confirm_order()

```
int confirm_order (
            MYSQL * connect,
            int meal_id,
            int total )
```

This function sets the date and time when the order was made and total order amount.

Precondition

A user chose 'confirm my meal' option in sub menu of making order

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order user wants to confirm
in	*total*	- This is the total price of the particular order

Returns

This function returns the id of the meal user wants to display

Postcondition

After finishing the session a user will be asked to pay

2.4.2.2 create_order()

```
int create_order (
            MYSQL * connect )
```

This function inserts a new row to the meal table.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Postcondition

fill_order funcion will be called

Returns

The function returns the id of newly created order

2.4.2.3 delete_dish_meal()

```
int delete_dish_meal (
            MYSQL * connect,
            int meal_id )
```

This function enables a visitor to delete a dish from their order.

Precondition

edit_meal function must be called before

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal↩ _id*	- This is the unique identificator of the order we edit now

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUC↩ CESS (0)

Postcondition

A dish from the particular order will be deleted

2.4.2.4 display_order()

```
int display_order (
            MYSQL * connect,
            int order_id )
```

This function prints out name id and price of every dish that was included to the particular meal. On the end total amount is printed.

Parameters

	connect	- This is a MYSQL pointer which provides the connection to the database
in	*meal↩ _id*	- This is the unique identificator of the order we want to display

Returns

This function returns value that is equal to the total amount of the particular order

2.4.2.5 display_prev_orders()

```
int display_prev_orders (
            MYSQL * connect )
```

This function displays id, date and total price of all the previous orders that user has ever had at the restaurant. And asks user for the id of the meal to display.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns the id of the meal to display

Postcondition

The display_order function will be called

2.4.2.6 edit_meal()

```
int edit_meal (
            MYSQL * connect,
            int meal_id )
```

This function enables visitor to edit their meal (add or delete a dish)

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order user wants to edit

Returns

This function return SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

2.4.2.7 fill_order()

```
int fill_order (
            MYSQL * connect,
            int meal_id )
```

This function ask user what dishes do they want to include in this particular meal then user is asked if they want to continue choosing, in the case of positive feedback cycle repeats, otherwise the process of making an order will be ended.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*meal_id*	- This is the unique identificator of the order we want to fill with dishes

Returns

This function returns SUCCESS (1) in the case when no error occured. Otherwise return value is UNSUCCESS (0)

2.4.2.8 log_in()

```
int log_in (
            MYSQL * connect,
            bool payment )
```

This function asks for user's id and password in the system, if given data was found in the database the function returns SUCCESS (1). Otherwise if user enter bad data five times the function will return UNSUCCESS (0). Also, the function is user of confirmation of a payment.

Parameters

in	*connect*	- This is a MYSQL pointer which enables the connection to the database
in	*payment*	- This is a boolean value which indicates if the function is used for confirmation of a payment or for log in.

Returns

This function returns SUCCESS (1) if user gave correct id and password. Otherwise return value is UNSUCCESS (0).

Postcondition

Depending on the return value the program will terminate or continue.

2.4.2.9 pay()

```
void pay (
            MYSQL * connect,
            bool * order_status,
            int meal_id )
```

This function simulates payment process and changes the value order_status in meal table to PAID if log_in function returns the SUCCESS (1) value.

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database
in	*order_status*	- This is the value which indicates if order was confirmed or no
in	*meal_id*	- This is the unique identificator of the order user wants to pay for

2.4.2.10 sub_menu()

```
void sub_menu (
            int mode )
```

This function prints out a type of sub menu according to the passed argument.

Parameters

in	*mode*	- This is the index of particular sub menu

2.4.2.11 user_interface()

```
int user_interface (
            MYSQL * connect )
```

This function provides interactivity between user and the system. This driver fucntion calls other functions according to the user's input.

Precondition

The program must be executed with 'order' parameter

Parameters

in	*connect*	- This is a MYSQL pointer which provides the connection to the database

Returns

This function returns **SUCCESS** (1) if no error occured. Otherwise returns **UNSUCCESS** (0)

2.4.3 Variable Documentation

2.4.3.1 current_user_id

```
int current_user_id
```

2.4.3.2 order_is_confirmed

```
bool order_is_confirmed = false
```

Index