

RESTAURANT INFORMATION SYSTEM

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

| | | |
|----------------------------|--|----|
| admsys.c | This file contains all function that are used in the admin mode | ?? |
| smartsys.c | This is the main file of the program containing the main function and some functions that both modes (admin/user mode) share | ?? |
| smartsys.h | This header file contains declarations of all functions, enumerations and macroses that program use | ?? |
| usersys.c | This file contains all functions that are used in the user mode | ?? |

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 function 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 occurred. 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 function 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 occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
|----|----------------|--|

Returns

This function returns SUCCESS (1) in the case when no error occurred. 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 | <i>connect</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>mode</i> | - 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 personalized 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 assigns '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 assigns 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 | <code>argc</code> | - This is the number of arguments passed to the command line |
| in | <code>argv</code> | - 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>type</i> | - This is one of the values of enumeration 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 | <i>format</i> | - This is a value from formats enumeration representing a particular format (date, number, dish_type) |
| in | <i>string</i> | - 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 assigns '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 | <i>option</i> | - 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 | <i>connect</i> | - 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 personalized welcome message will be printed.

Parameters

| | | |
|----|-------------|--|
| in | <i>name</i> | - This is the name of the visitor. |
| in | <i>mode</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>id</i> | - This is a numerical attribute value to be checked |
| in | <i>table_name</i> | - This is the name of the table in which a number must be checked |
| in | <i>attribute_name</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>cols</i> | - This is the number of columns 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 | <i>correct</i> | - 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 | <i>string</i> | - 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 | <i>mode</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>lower</i> | - This is the lower bound of the interval |
| in | <i>upper</i> | - This is the upper bound of the interval |
| in | <i>table_name</i> | - This is the name of the table in which the generated number must be unique |
| in | <i>attribute_name</i> | - 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 assigns a passed char variable to the one of the options 'Y' or 'N' depending on the user's input.

Parameters

| | | |
|----|---------------|--|
| in | <i>answer</i> | - 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 function 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 personalized 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 assigns 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 assigns '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 function 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 available 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 supports 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 enumeration 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 enumeration 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 enumeration 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 enumerations 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 enumeration 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 enumeration 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 enumeration 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 behavior 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 occurred. 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 function calls other functions according to the user's input.

Precondition

The program must be executed with 'admin' parameter

Parameters

| | | |
|----|----------------|---|
| in | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
|----|----------------|---|

Returns

This function returns **SUCCESS** (1) if no error occurred. 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 | <i>argc</i> | - This is the number of arguments passed to the command line |
| in | <i>argv</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identifier of the order user wants to confirm |
| in | <i>total</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
|----|----------------|---|

Postcondition

fill_order function 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identifier of the order we edit now |

Returns

This function returns SUCCESS (1) in the case when no error occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
|----|----------------|--|

Returns

This function returns SUCCESS (1) in the case when no error occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>type</i> | - This is one of the values of enumeration 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

| | | |
|----|----------------|---|
| | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - 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 | <i>connect</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identificator of the order user wants to edit |

Returns

This function return SUCCESS (1) in the case when no error occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - 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 occurred. 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 | <i>format</i> | - This is a value from formats enumeration representing a particular format (date, number, dish_type) |
| in | <i>string</i> | - 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 assigns '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 | <i>option</i> | - 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 | <i>connect</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>payment</i> | - 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.

2.3.4.17 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 personalized welcome message will be printed.

Parameters

| | | |
|----|-------------|--|
| in | <i>name</i> | - This is the name of the visitor. |
| in | <i>mode</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>id</i> | - This is a numerical attribute value to be checked |
| in | <i>table_name</i> | - This is the name of the table in which a number must be checked |
| in | <i>attribute_name</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>order_status</i> | - This is the value which indicates if order was confirmed or no |
| in | <i>meal_id</i> | - This is the unique identifier 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>cols</i> | - This is the number of columns 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 | <i>correct</i> | - 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 | <i>string</i> | - 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 | <i>mode</i> | - 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 | <i>connect</i> | - 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 | <i>mode</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>mode</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>lower</i> | - This is the lower bound of the interval |
| in | <i>upper</i> | - This is the upper bound of the interval |
| in | <i>table_name</i> | - This is the name of the table in which the generated number must be unique |
| in | <i>attribute_name</i> | - 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 function calls other functions according to the user's input.

Precondition

The program must be executed with 'order' parameter

Parameters

| | | |
|-----------|----------------|---|
| <i>in</i> | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
|-----------|----------------|---|

Returns

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

2.3.4.30 yes_no_answer()

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

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

Parameters

| | | |
|-----------|---------------|--|
| <i>in</i> | <i>answer</i> | - 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 function 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identifier of the order user wants to confirm |
| in | <i>total</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
|----|----------------|---|

Postcondition

fill_order function 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identificator of the order we edit now |

Returns

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

| | | |
|----|----------------|---|
| | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - 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 | <i>connect</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - This is the unique identifier of the order user wants to edit |

Returns

This function return SUCCESS (1) in the case when no error occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>meal_id</i> | - 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 occurred. 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 | <i>connect</i> | - This is a MYSQL pointer which enables the connection to the database |
| in | <i>payment</i> | - 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 | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
| in | <i>order_status</i> | - This is the value which indicates if order was confirmed or no |
| in | <i>meal_id</i> | - This is the unique identifier 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 | <i>mode</i> | - 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 function calls other functions according to the user's input.

Precondition

The program must be executed with 'order' parameter

Parameters

| | | |
|----|----------------|---|
| in | <i>connect</i> | - This is a MYSQL pointer which provides the connection to the database |
|----|----------------|---|

Returns

This function returns **SUCCESS** (1) if no error occurred. 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

ACTIVE
 smartsys.h, 16
add_dish
 admsys.c, 4
 smartsys.h, 20
add_dish_menu
 smartsys.h, 18
add_to_meal
 smartsys.h, 19
admin_interface
 admsys.c, 4
 smartsys.h, 20
admin_main_menu
 smartsys.h, 17
admin_mode
 smartsys.h, 20
admsys.c, 3
 add_dish, 4
 admin_interface, 4
 delete_dish_menu, 4
 show_income, 5
 un_pause_dish, 5
all
 smartsys.h, 19
args_check
 smartsys.c, 7
 smartsys.h, 21

back
 smartsys.h, 19
BAD_FORMAT
 smartsys.h, 16

confirm_order
 smartsys.h, 21
 usersys.c, 34
create_meal
 smartsys.h, 19
create_order
 smartsys.h, 22
 usersys.c, 34
current_user_id
 usersys.c, 39

database
 smartsys.c, 12
date_format
 smartsys.h, 18
DATELEN
 smartsys.h, 16

del_dish_menu
 smartsys.h, 18
del_from_meal
 smartsys.h, 19
delete_dish_meal
 smartsys.h, 22
 usersys.c, 35
delete_dish_menu
 admsys.c, 4
 smartsys.h, 23
dessert
 smartsys.h, 19
dish_menu
 smartsys.h, 18
display_dish_menu
 smartsys.c, 8
 smartsys.h, 23
display_order
 smartsys.h, 24
 usersys.c, 35
display_prev_orders
 smartsys.h, 24
 usersys.c, 36
drink
 smartsys.h, 19

edit_meal
 smartsys.h, 24
 usersys.c, 36
edit_order
 smartsys.h, 19
editing
 smartsys.h, 18
end_admin_session
 smartsys.h, 18
end_usr_session
 smartsys.h, 19

fill_order
 smartsys.h, 25
 usersys.c, 36
first
 smartsys.h, 19
flag
 smartsys.c, 12
format_check
 smartsys.c, 8
 smartsys.h, 25
formats
 smartsys.h, 18

- get_option
 - smartsys.c, 8
 - smartsys.h, 26
- help
 - smartsys.h, 20
- idle
 - smartsys.h, 20
- install_db
 - smartsys.c, 9
 - smartsys.h, 26
- installation
 - smartsys.h, 20
- log_in
 - smartsys.h, 26
 - usersys.c, 37
- main
 - smartsys.c, 9
- main_dish
 - smartsys.h, 19
- main_menu
 - smartsys.c, 9
 - smartsys.h, 27
- make_order
 - smartsys.h, 18
- MAX_TRIES
 - smartsys.h, 16
- MAXID
 - smartsys.h, 16
- MAXLEN
 - smartsys.h, 16
- MAXSTR
 - smartsys.h, 17
- MENUWIDTH
 - smartsys.h, 17
- null_selected
 - smartsys.c, 10
 - smartsys.h, 27
- number
 - smartsys.h, 18
- order_is_confirmed
 - usersys.c, 39
- order_making
 - smartsys.h, 18
- PAID
 - smartsys.h, 17
- password
 - smartsys.c, 12
- PAUSE
 - smartsys.h, 17
- pause_resume_dish
 - smartsys.h, 18
- pay
 - smartsys.h, 28
- usersys.c, 37
- port
 - smartsys.c, 13
- print_results
 - smartsys.c, 10
 - smartsys.h, 28
- print_usage_msg
 - smartsys.c, 10
 - smartsys.h, 29
- reg
 - smartsys.c, 10
 - smartsys.h, 29
- registration
 - smartsys.h, 20
- remove_new_line
 - smartsys.c, 11
 - smartsys.h, 29
- result
 - smartsys.h, 32
- return_to_main_menu
 - smartsys.c, 11
 - smartsys.h, 29
- row
 - smartsys.h, 32
- server
 - smartsys.c, 13
- show_income
 - admsys.c, 5
 - smartsys.h, 30
- show_income_per
 - smartsys.h, 18
- show_menu
 - smartsys.h, 19
- show_prev_meal
 - smartsys.h, 19
- smartsys.c, 6
 - args_check, 7
 - database, 12
 - display_dish_menu, 8
 - flag, 12
 - format_check, 8
 - get_option, 8
 - install_db, 9
 - main, 9
 - main_menu, 9
 - null_selected, 10
 - password, 12
 - port, 13
 - print_results, 10
 - print_usage_msg, 10
 - reg, 10
 - remove_new_line, 11
 - return_to_main_menu, 11
 - server, 13
 - unique_random_num, 11
 - user, 13
 - yes_no_answer, 12

smartsys.h, 13
 ACTIVE, 16
 add_dish, 20
 add_dish_menu, 18
 add_to_meal, 19
 admin_interface, 20
 admin_main_menu, 17
 admin_mode, 20
 all, 19
 args_check, 21
 back, 19
 BAD_FORMAT, 16
 confirm_order, 21
 create_meal, 19
 create_order, 22
 date_format, 18
 DATELEN, 16
 del_dish_menu, 18
 del_from_meal, 19
 delete_dish_meal, 22
 delete_dish_menu, 23
 dessert, 19
 dish_menu, 18
 display_dish_menu, 23
 display_order, 24
 display_prev_orders, 24
 drink, 19
 edit_meal, 24
 edit_order, 19
 editing, 18
 end_admin_session, 18
 end_usr_session, 19
 fill_order, 25
 first, 19
 format_check, 25
 formats, 18
 get_option, 26
 help, 20
 idle, 20
 install_db, 26
 installation, 20
 log_in, 26
 main_dish, 19
 main_menu, 27
 make_order, 18
 MAX_TRIES, 16
 MAXID, 16
 MAXLEN, 16
 MAXSTR, 17
 MENUWIDTH, 17
 null_selected, 27
 number, 18
 order_making, 18
 PAID, 17
 PAUSE, 17
 pause_resume_dish, 18
 pay, 28
 print_results, 28
 print_usage_msg, 29
 reg, 29
 registration, 20
 remove_new_line, 29
 result, 32
 return_to_main_menu, 29
 row, 32
 show_income, 30
 show_income_per, 18
 show_menu, 19
 show_prev_meal, 19
 smartsys_h, 17
 snack, 19
 sub_menu, 30
 sub_menu_modes, 18
 sub_menu_options, 18
 SUCCESS, 17
 type, 18
 types_of_dishes, 19
 un_pause_dish, 30
 unique_random_num, 31
 UNSUCCESS, 17
 user_interface, 31
 user_main_menu, 19
 visitor_mode, 20
 work_modes, 19
 yes_no_answer, 32
smartsys_h
 smartsys.h, 17
snack
 smartsys.h, 19
sub_menu
 smartsys.h, 30
 usersys.c, 38
sub_menu_modes
 smartsys.h, 18
sub_menu_options
 smartsys.h, 18
SUCCESS
 smartsys.h, 17
type
 smartsys.h, 18
types_of_dishes
 smartsys.h, 19
un_pause_dish
 admsys.c, 5
 smartsys.h, 30
unique_random_num
 smartsys.c, 11
 smartsys.h, 31
UNSUCCESS
 smartsys.h, 17
user
 smartsys.c, 13
user_interface
 smartsys.h, 31
 usersys.c, 38

- user_main_menu
 - smartsys.h, [19](#)
- usersys.c, [32](#)
 - confirm_order, [34](#)
 - create_order, [34](#)
 - current_user_id, [39](#)
 - delete_dish_meal, [35](#)
 - display_order, [35](#)
 - display_prev_orders, [36](#)
 - edit_meal, [36](#)
 - fill_order, [36](#)
 - log_in, [37](#)
 - order_is_confirmed, [39](#)
 - pay, [37](#)
 - sub_menu, [38](#)
 - user_interface, [38](#)
- visitor_mode
 - smartsys.h, [20](#)
- work_modes
 - smartsys.h, [19](#)
- yes_no_answer
 - smartsys.c, [12](#)
 - smartsys.h, [32](#)