

Team name : 愛情WA

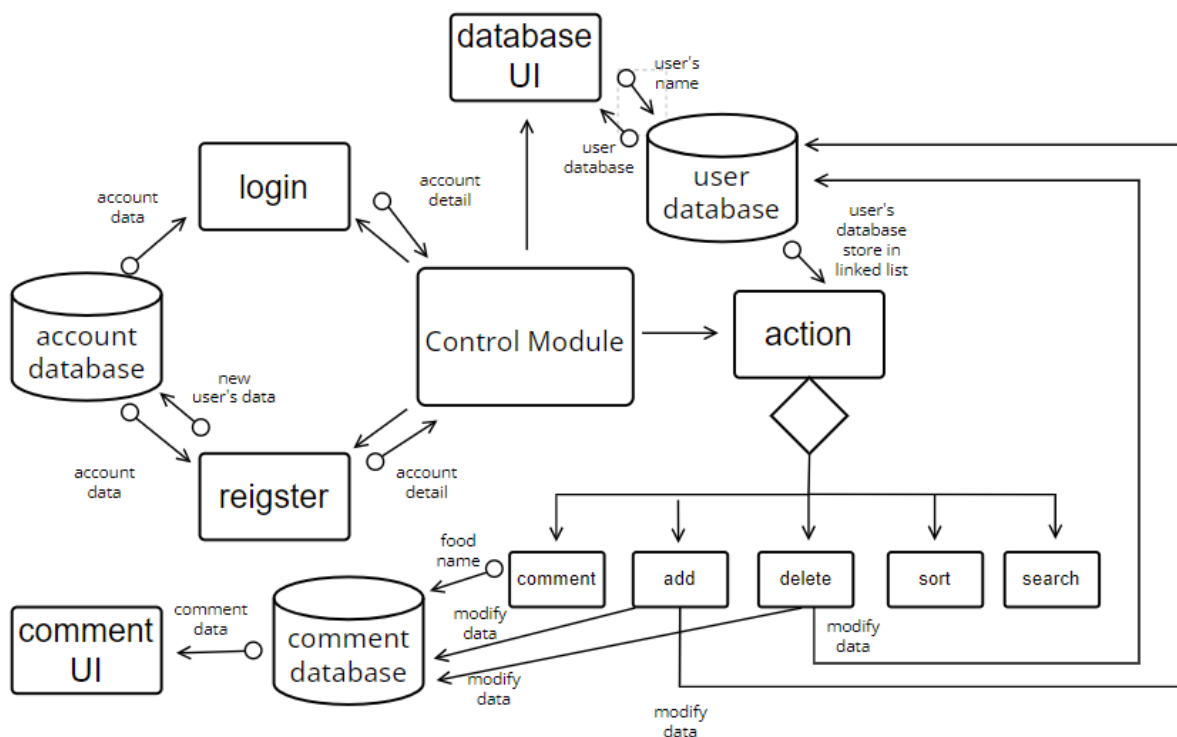
- 周宥閔 411410010 陳胤 411410016
- 張原肇 411410013 鄭俊良 411410065 鄒語蘋 411410064

PAPER REPORT

Introduction:

This project is used to **record food**, including the food, restaurant, the price, and the rating. It serves as a reference for each time you are racking your brain to find out what to order for lunch or dinner. Additionally, it has built up a function where you can write down your own review. You can also see the average rating of the food and various reviews from other users, which can be used as a reference for choosing food!

Program Design:



All the data in the database was stored in a text file, whenever the function called for data, it would read the data in the text file and create a linked list to store it.

Sign in:

- Account : 20 letters is the maximum input, the account username entered should be exactly the same as the username the user registered with.
- Password : 20 letters is the maximum input, the password entered should be exactly the same as the username the user registered with too.

If any mistakes or typos are to be made, the program will prompt the user to retry or register an account.

Register:

- Account : 20 letters is the maximum input, the name should only consist of characters and numbers, and the name shouldn't repeat with the previous account.
- Password : 20 letters is the maximum input, the passwords should consist of at least 2 uppercase characters and 2 lowercase characters, moreover, the passwords also need at least one number.

Except for the requirement above, the password shouldn't be the same as the account or similar to it (5 letters length).

The user can choose whether to type their password or to upload textfile.

Terminal UI:

It has two parts, database UI and comment UI.

Database UI : prints out the food name as char with at most 20 letters, store name as char with at most 20 letters, price as int, score as int, and page as int. Each page contains 10 pieces of data.

Comment UI : prints out the account name as char with at most 20 letters, score as int, comment as char with at most 200 letters, and count the average score.

Action:

Login and register will send data to the main function, and create a linked list of user's data from the database, and the action function can make modifications based on the linked list.

1) Create new data :

The function will ask the user to input the data, including the food name, the store name, the price of the food, the rating, and whether to type the comment. As the user inputs those data, the function will store those data into the linked list, besides, the comment part will call another function to store the comment user type, and send the data to where the comment was collected.

2) Delete data :

The function will base on the ID which is shown by Database UI, when the user chooses the data it wants to delete, the function will delete it from the linked list, and also call another function to delete all the related data at the comment database.

3) Sort :

The function will ask the user to choose which column it wants to sort, and it can also choose the order, then the function will sort the data from the linked list and the user's requirement.

4) Search :

The function will ask the user to choose which column it want to search, then ask user to type the word or number it wanted to search, then the data correspond to the search will then create a new linked list to store it, but different from the other actions, it won't send back the processed linked list to the main function, when the user finish searching, it will have to leave the function and destroyed the linked list which create temporary.

5) Comment :

The function will ask the user to choose which food's comment it wants to see, just like the delete function does, then it will show all the comments and average rating of the food from the database.

6) logout:

When a user logs out, it will store all the data in the linked list to the user's data text file, and clear the linked list.

Comment :

In the project, we use text files to store the related comments of the food that different users type to the database. In order to read those text files clearly, we use "opendir" and "fopen" to find the right data. In addition, we have three different type of folder, except the user's database folder, one of the other folder is to store the text file that record all the rating from different user of a specific food and a related text file name which is used for store the comment, the other folder is to stored all the comment that just mentioned.

Same as the food database, we stored the comment data in a linked list with new a structure, but there is a difference between food database and comment database, food database will write the data to the text file when they logout the terminal, while comment data will modify as soon as the user change the data, it help user to review the modification immediately.

Basic Part :

1. Add :

Users can add new data to the database by action function, it can also detect whether the data have already been stored in the database.

1) Input data and check if it already been stored

```

printf("please enter the name of the food:(the limit of the char is 20)\n");
printf("enter: ");
fgets(food_name, sizeof(char)*21, stdin);
len = strlen(food_name);
*(food_name+len-1) = '\0';
food_name = change_space(food_name);
printf("please enter the store of the food:(the limit of the char is 20)\n");
printf("enter: ");
fgets(store_name, sizeof(char)*21, stdin);
len = strlen(store_name);
*(store_name+len-1) = '\0';
store_name = change_space(store_name);
check = 0;
if(check_repeat_food_store(list, food_name, store_name) == -1){
    printf("you already score this food, maybe try a new one\n");
    check = 1;
}

```

```

static int check_repeat_food_store(struct linked_list *list, char *food, char *store){
    while(list!=NULL){
        if(strcmp(food, list->food_name)==0 && strcmp(store, list->store_name)==0){
            return -1;
        }
        list = list->next;
    }
    return 1;
}

```

2) Stored the data into the linked list

```

void insert_front(struct linked_list **list, char food_name[], char store_name[], int price, float score, char comment_txt[]){
    struct linked_list *new_node = malloc(sizeof(struct linked_list));
    new_node->next = NULL;
    strcpy(new_node->food_name, food_name);
    strcpy(new_node->store_name, store_name);
    strcpy(new_node->comment_txt, comment_txt);
    new_node->price = price;
    new_node->score = score;
    if(*list == NULL){
        *list = new_node;
        return;
    }
    else{
        struct linked_list *now = *list;
        while(now->next != NULL){
            now = now->next;
        }
        now->next = new_node;
        return;
    }
}

```

2. Delete:

Users can delete data by action function, since we let users delete databases based on the ID shown on the UI, we only have to check whether the ID exists.

1) Enter data to check whether it can delete

```

struct linked_list *delete_list(struct linked_list *list, char user[], int page){
    int ID;
    struct linked_list *node = list;
    printf("which ID you want to delete?\n");
    printf("enter:");
    scanf("%d", &ID);
    getchar();
    ID += page*10; // 確認UI return的數值
    // printf("%d", ID);
    for(int i=0; i<ID; i++){
        if(node==NULL){
            printf("there is no data to delete!\n");
            return list;
        }
        node = node->next;
    }
    // printf("%s %s\n", node->food_name, node->store_name);
    delete_data(node->food_name, node->store_name, user);
    list = delete_linked_list(list, ID);
    return list;
}

```

2) Delete the data

```

struct linked_list *delete_linked_list(struct linked_list *list, int position){
    struct linked_list *new_node;
    if(position == 0){
        return list->next;
    }
    for(int i = 0; i<position; i++){
        new_node = new_node->next;
    }
    new_node->next = new_node->next->next;
    return list;
}

```

3. Traverse:

The data will be printed out by the database UI function. In order to improve the user's using experience, we only print out 10 data on one page, and the user can change the page to view the data.

1) Print out the data

```

node = list;
printf("| ID | Name | Store | Price | Score |\n");
for(int l = 0; l <= 63; l++){
    printf("-");
}
printf("\n");

for(int i = 0; i < 10; node = node->next){
    if(node != NULL){
        printf("%4d |%*s |%*s |%*d |%.1f |\n", i, 10, node->food_name, 10, node->store_name, 7, node->price, 7, node->score);
    }else{
        break;
    }
    i++;
}
}

```

- 2) Change the page by point to the right position, when it is out of range, it will return and tell the user there is no next page.

```

for(int i = 0; i < page*10; node = node->next){
    if(node == NULL){
        return -1;
    }
    i++;
}
node = list;

```

4. Search:

Users can search data by action function, it can search all the related data, and store it in a linked list, then return it and print by UI function(you can see the UI in the demonstration).

- 1) Search food(string)

```

fgets(txt, sizeof(char)*21, stdin);
len = strlen(txt);
*(txt+len-1) = '\0';
txt = change_space(txt);
// printf("%s\n", txt);
while(node != NULL){
    // printf("%s %s\n", node->food_name, txt);
    if(strcmp(node->food_name, txt) == 0){
        insert_front(&new_list, node->food_name, node->store_name, node->price, node->score, node->comment_txt);
    }
    node = node->next;
}
return new_list;

```

5. Sort:

The user can sort data by action function, and the details can change by user, including the price and the rating. Moreover, users can choose the order from biggest to smallest or smallest to biggest(you can see the UI in the demonstration).

- 1) Sort the data by bubble sort

```

for(int i=0;i<list_count;i++){
    int change = 0;
    prev = list;
    while(node->next!=NULL){
        if(node->score - node->next->score > FLOAT_TOLERANCE){
            #ifdef DEBUG
            printf("xxx%s %s %d %0.1f %s\n",node->food_name,node->score,node->next->score,node->next->food_name);
            printf("xnx%s %s %d %0.1f %s\n\n",node->next->food_name,node->next->score,node->next->next->food_name,node->next->next->score);
            #endif
            if(order == 2){
                if(node == list){
                    list = node->next;
                    node = node_switch(node,node->next);
                }
                else{
                    node = node_switch(node,node->next);
                    prev->next = node;
                }
                change = 1;
            }
        }
        node = node->next;
    }
}

```

6. File I/O:

Open the corresponding txt file, user can put there own text file into the project folder,next enter their text file name to the terminal, then the function can read the word from the text file.

```

upload_password = malloc(sizeof(char) * 100);
printf("please enter your file name(please put your file into the folder)\n");
fgets(upload_password,100,stdin);
len = strlen(upload_password);
*(upload_password+len-1) = '\0';
// printf("%s\n",upload_password);
FILE *file = fopen(upload_password,"r");
fgets(corres_password,100,file);
len = strlen(corres_password);
*(corres_password+len-1) = '\0';
// printf("%s\n",corres_password);
goto upload;

```

Advance Part:

Login: Data type : string(char*), pointer(*ptr)

Data structure : struct

Database system : linked list

```
struct acpd_list *list;
struct linked_list *acpdlist;
//reads txt file
list = acpd_read("acpd.txt");

static char username_input[20], password_input[20];
```

Operations example : String Comparison

```
struct acpd_list *current=list;
while(current!=NULL)
{
    if(strcmp(username_input, current->account)==0 && strcmp(password_input, current->password)==0)
    {
        printf("You have successfully logged in.");
        char *result= current->data_base_txt;
        free_acpd_list(list);
        return result;
    }
    current=current->next;
}
```

Register : **Data type :** int, string(char*), pointer(*ptr)

Data structure : struct

Database system : linked list

```
int access = 0, choice;
char* account_name = (char*)malloc(sizeof(char) * 100);
struct acpd_list *new_node = acpd_read("acpd.txt");
struct acpd_list *ptr;
char *password_total,*register_new = malloc(sizeof(char)*100);
```

Operations example : Check whether the account name already exists

```
for(ptr = new_node; ptr != NULL; ptr = ptr->next){
    if(strcmp(ptr->account, account_name) == 0){
        printf("Account already exists\n");
        goto origin;
    }
}
```

Operations example : Similarity


```

char *match;
account_name[strcspn(account_name, "\n")] = '\0';
corres_password[strcspn(corres_password, "\n")] = '\0';
if(strlen(corres_password) <= strlen(account_name)){
    match = strstr(account_name, corres_password);
}
else{
    match = strstr(corres_password, account_name);
}
if (match != NULL && strlen(match) >= 5){
    printf("Your password is too similar to your account\n");
    goto origin;
}

```

Comment :

This design helped control the database of the comments, including create, delete, and search, since we store the data in the text file, it was hard to store the data of the user's name, rating, and the comment all in a text file separated by different food. In order to solve this problem, we separate different type of data text file into different folder, which include the user's food data, the collect of the related food text file which stored the rating from different user, and all the comment of the food. by those folder, we could find and modify the data clearly.

(1) insert the data to comment database

```

DIR *dir = opendir("comment_data_base");
struct dirent *entry;
int found_file = 0;
while((entry = readdir(dir)) != NULL){
    if(strcmp(entry->d_name, comment_txt) == 0){
        found_file = 1;
        char path[100];
        sprintf(path, "comment_data_base/%s", entry->d_name);
        FILE *p = fopen(path, "a");
        if(comment_bool == 2){//要寫comment
            strcat(strcat(strcat(strcat(strcat(strcpy(comment_collect_txt, user), "_"), food_name), "_"), store_name), ".txt");
            if(insert_comment(comment_collect_txt) == -1){
                fprintf(p, "%s %0.1f %c\n", user, score, '0');
            }
            else{
                fprintf(p, "%s %0.1f %s\n", user, score, comment_collect_txt);
            }
        }
        else{
            fprintf(p, "%s %0.1f %c\n", user, score, '0');
        }
        fclose(p);
        break;
    }
}

```

(2) delete data from comment database

```

DIR *dir = opendir("comment_collect");
sprintf(path,"comment_collect/%s",comment_collect_txt);
FILE *new_file = fopen(path,"w");
if(new_file==NULL){
    printf("Error: there is problem creating comment txt file\n");
    printf("Sorry!Please type it later\n");
    return -1;
}
printf("please write down your comment:\n");
fgets(comment_write,200,stdin);
len = strlen(comment_write);
*(comment_write+len-1) = '\0';
// printf("%s\n",comment_write);
printf("wait for a second, data is sending\n");//不知道為甚麼，把這個
fprintf(new_file,comment_write);

```

(3)search for a specific food rating.

```

DIR *dir = opendir("comment_data_base");
struct dirent *entry;
while ((entry = readdir(dir)) != NULL) {
    if (strcmp(entry->d_name, txt) == 0) { // 判斷是否為.txt檔案
        char path[100];
        sprintf(path, "comment_data_base/%s", entry->d_name); // 構造檔案路徑
        // printf("ok?\n");
        FILE *p = fopen(path, "r"); // 打開檔案
        if (p != NULL) {
            // printf("line count = %d\n", line_count);
            char *c[10],tempc[1000];
            int i = 0;
            while(fgets(tempc,100,p)!=NULL){
                // printf("%s\n",tempc);
            }
        }
    }
}

```

```

while(fgets(tempc,100,p)!=NULL){
    // printf("ok?\n");
    int temp = strlen(tempc);
    int count = 0;
    char account[20],comment_txt[20];
    float score;
    // printf("ok22?\n");
    if(tempc[temp-1] != '\n'){
        tempc[temp] = '\0';
    }
    else{
        tempc[temp-1] = '\0';
    }
    c[i] = strdup(tempc);
    sscanf(c[i],"%s %f %s",account,&score,comment_txt);
    insert_front_comment(&new_node,account,score,comment_txt);
    i++;
}

```

conclusion :

food database and comment database make the multiple user become available, this makes different users not only store their own data, it can view others rating and comment on the same food, this advanced part creates a database used for all users, not only one user.

Demonstration:

Register

```

welcome to the database!
log in or register?
    [1]login
    [2]register
    [3]quit
enter: 2
Do you want to create a new account?
[1]Yes [2]No
1
Input your account:(the account name should only use characters and numbers)
pineapple2023
Input your password:(the password need at least two big characters and small
characters, and also at least one number)
PineApple2023

```

Login

```
welcome to the database!
log in or register?
  [1]login
  [2]register
  [3]quit
enter: 1
Username: pineapple2023
Password: PineApple2023
You have successfully logged in.
```

UI

ID	Name	Store	Price	Score
0	Cheese_Burger	McDonald's	90	4.0
1	Beef_Burger	KFC	70	4.5
2	Chicken_Noodle	7-11	65	4.7

Actions :

Create

```

action you can do:
[1]next page
[2]last page
[3]create new data
[4]delete data
[5]sort the data
[6]search the data
[7]see the comment
[8]logout
enter: 3
please enter the name of the food:(the limit of the char is 20)
enter: Cheese Burger
please enter the store of the food:(the limit of the char is 20)
enter: McDonald's
please enter the price of the food:90
please enter the score of the food:4
Do you want to comment this food?
[1]Yes
[2]No
enter:2
| ID | Name | Store | Price | Score |
-----
| 0 | Cheese_Burger | McDonald's | 90 | 4.0 |
-----

```

Delete

ID	Name	Store	Price	Score
0	Cheese_Burger	McDonald's	90	4.0
1	Beef_Burger	KFC	70	4.5
2	Chicken_Noodle	7-11	65	4.7

```

Page 1
action you can do:
[1]next page
[2]last page
[3]create new data
[4]delete data
[5]sort the data
[6]search the data
[7]see the comment
[8]logout
enter: 4
which ID you want to delete?
enter:1
here
| ID | Name | Store | Price | Score |
-----
| 0 | Cheese_Burger | McDonald's | 90 | 4.0 |
| 1 | Beef_Burger | KFC | 70 | 4.5 |
-----

```

Sort

```
action you can do:
[1]next page
[2]last page
[3]create new data
[4]delete data
[5]sort the data
[6]search the data
[7]see the comment
[8]logout
```

enter: 5

which one is your basic type?

```
[1]price
[2]score
```

enter: 1

For what order?

```
[1]great to small
[2]small to great
```

enter: 1

ID	Name	Store	Price	Score
0	Cheese_Burger	McDonald's	90	4.0
1	Beef_Burger	KFC	70	4.5
2	Chicken_Noodle	7-11	65	4.7

Search

```
action you can do:
[1]next page
[2]last page
[3]create new data
[4]delete data
[5]sort the data
[6]search the data
[7]see the comment
[8]logout
```

enter: 6

which one you want to search?

```
[1]food
[2]store
[3]price
[4]score
```

enter: 3

enter you searching: 70

ID	Name	Store	Price	Score
0	Beef_Burger	KFC	70	4.5

Comment

```

Do you want to comment this food?
  [1]Yes
  [2]No
enter:1
please write down your comment:
It's delicioussssss!!!!
wait for a second, data is sending
| ID | Name | Store | Price | Score |
-----
| 0 | Cheese_Burger | McDonald's | 90 | 4.0 |
| 1 | Beef_Burger | KFC | 70 | 4.5 |
-----

```

See the comment

```

| ID | Name | Store | Price | Score |
-----
| 0 | Cheese_Burger | McDonald's | 90 | 4.0 |
| 1 | Beef_Burger | KFC | 70 | 4.5 |
| 2 | Chicken_Noodle | 7-11 | 60 | 5.0 |
-----

Page 1
action you can do:
  [1]next page
  [2]last page
  [3]create new data
  [4]delete data
  [5]sort the data
  [6]search the data
  [7]see the comment
  [8]logout
enter: 7
waht comment you want to see?
enter:(ID) 1
Beef_Burger_KFC.txt
-----
| ID | Account | Score |
-----
| 0 | t | 4.5 |
-----


It's delicioussssss!!!!-----
-----
Average Score: 4.5

```

folders:

comment_collect	2023/6/11 下午 04:27	檔案資料夾
comment_data_base	2023/6/11 下午 04:27	檔案資料夾
data_base	2023/6/11 下午 04:21	檔案資料夾

user data:

 test.txt


×

+

檔案 編輯 檢視

chicken_rice poxi 50 0.1 chicken_rice_poxi.txt
french_frice mcdonald 40 9.6 french_frice_mcdonald.txt
soda mcdonald 38 9.2 soda_mcdonald.txt

specify food data:


 comment_test.txt

●

檔案 編輯 檢視

YuMin 9.5 YuMin_comment_test.txt
In 8.6 In_comment_test.txt

comment:

 comment_lanpa_bbccsb8787.txt

×

檔案 編輯 檢視

good to eat