Paper Report

Unknown

411410055 蕭子期
411410067 許世昕
411410073 洪子傑
411410091 葉暄鴻

Outline

Introduction

Welcome to our library! In this library, it is divided into two parts, Administrator and Reader parts respectively. When executing, the user will choose which mode to enter , then need to sign up an account, below is a brief introduction for the mode.

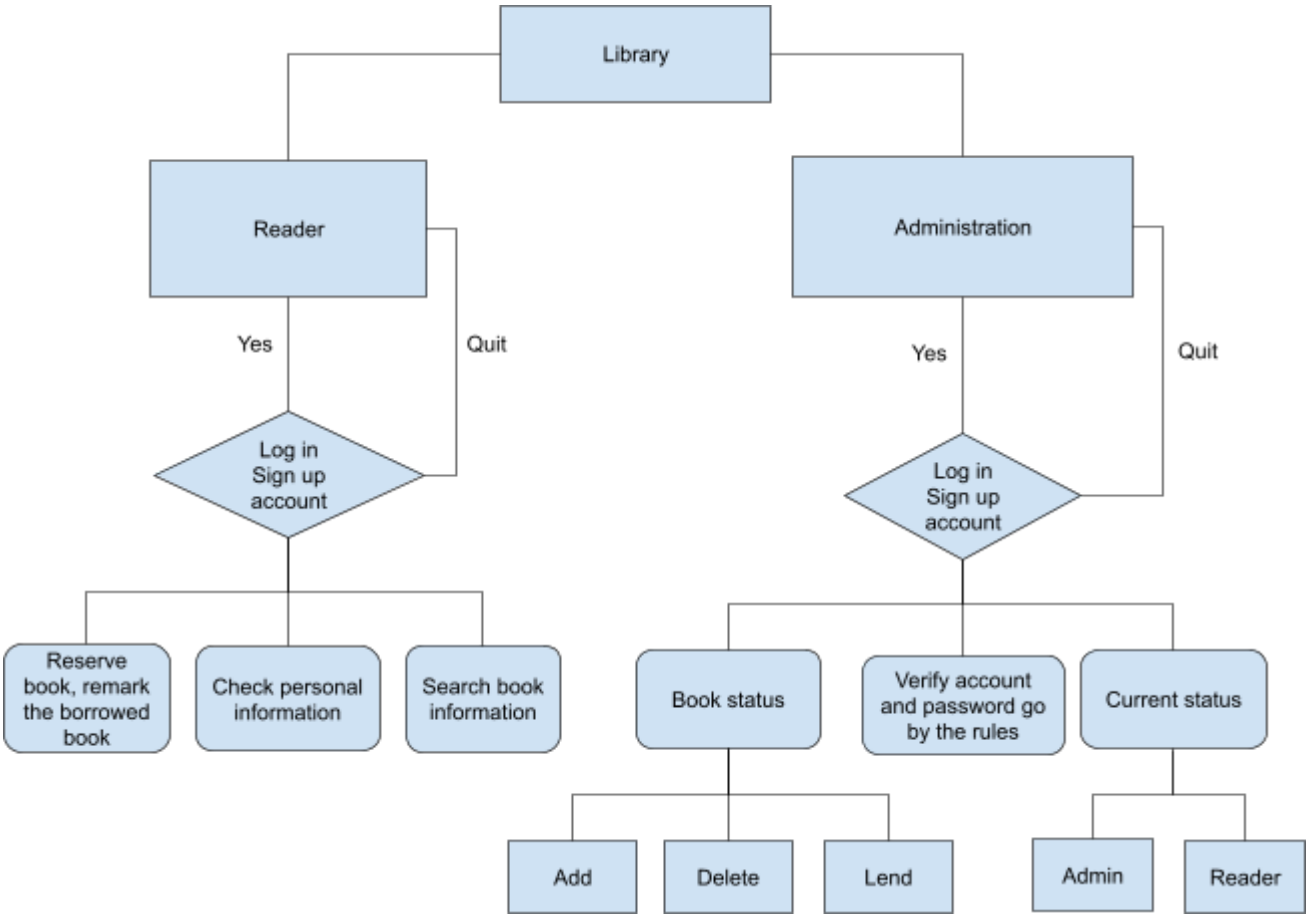
● ADMINISTRATION MODE
- Book status(add, delete, lend).
- Check status(book, reader and admin information)
- Record borrowing history.
- Verify email account and password go by the rules.

● DATABASE
- Login account and Sign in account to administration and reader mode.
- Using struct type in Book and Manager, linked list type in Reader and
- Queue structure in storing borrowed history.

● USER-INTERFACE
- Using SDL resources to make buttons more aesthetic and functional.
- Using text boxes to provides a convenient, clear, and interactive way to input or display text.
- Incorporate prompts, error messages, and feedback to provide information to user
- System will clear the screen automatically to ensure a clean and readable interface for a better user experience

● READER MODE
- Check personal information.
- Reverse book from library.
- Search books information (name, author, call number, publisher, ISBN.)

This system is more visible and user-friendly than a normal library system, so we sincerely welcome to experience such an epoch-making system ! Hope you revel in our library !



Program Design

Starting the system, users need to choose two modes, which are reader and administration. Then sign up for the account when first entering in, the system will verify the format's user key in going by the rules. In reader mode, users can search all book’s information, reserve books, or check their personal information. In administration mode,users can check all the status of administrations, books, and readers.

Basic Part

- Three basic data types (int, float, char), and data created using structure

```c
struct books
{
    char book_name[SPACE];
    char author[SPACE];
    char publisher[SPACE];
    int publish_year;
    int amount;        //amount
    char call_number[SPACE];
    char isbn[SPACE];
    float viewer_count;
    int accession_number;
    int status; // to know whether the book is borrowed
}book[MAX_BUF];
```

- String type

```c
printf("| Are you sure you want to delete %-20s |\n", book[i].book_name);
printf("| 1. Yes%46s |\n", " ");
printf("| 2. No%47s |\n", " ");
```

- Linked list

```c
struct readers
{
    char re_name[SPACE];
    int student_id;
    char email[SPACE];
    char re_account[SPACE];
    char re_password[SPACE];
    struct readers *next;
};
struct readers *first = NULL;

extern struct readers *first;
```

- Add

  Administrator can add new book information, and the system will store it into the book database.

```
void add_book()
{
    char book_name[SPACE], author[SPACE], publisher[SPACE], call_number[SPACE], isbn
    if (amount_books == MAX_BUF)
    { // The capacity of the library is full
        printf("Library is full. Cannot add more books.\n");
        return;
    }

    printf("\033[H\033[2J"); // clear screen
    printf("Please enter the book_name: ");
    fgets(book_name, SPACE, stdin);
    if(book_name[strlen(book_name) - 1] == '\n')
        book_name[strlen(book_name) - 1] = '\0';

    printf("Please enter the aurthor: ");
    fgets(author,  SPACE, stdin);
    if(author[strlen(author) - 1] == '\n')
        author[strlen(author) - 1] = '\0';

    printf("Please enter the publisher: ");
    fgets(publisher,  SPACE, stdin);
    if(publisher[strlen(publisher) - 1] == '\n')
        publisher[strlen(publisher) - 1] = '\0';

    printf("Please enter the publish_year: ");
    scanf("%d",&book[amount_books].publish_year);

    fflush(stdin);
```

```
31      printf("Please enter the call_number: ");
32      fgets(call_number,  SPACE, stdin);
33      if(call_number[strlen(call_number) - 1] == '\n')
34          call_number[strlen(call_number) - 1] = '\0';
35
36      printf("Please enter the isbn: ");
37      fgets(isbn,  SPACE, stdin);
38      if(isbn[strlen(isbn) - 1] == '\n')
39          isbn[strlen(isbn) - 1] = '\0';
40
41      strcpy(book[amount_books].book_name, book_name);
42      strcpy(book[amount_books].author, author);
43      strcpy(book[amount_books].publisher, publisher);
44      strcpy(book[amount_books].call_number, call_number);
45      strcpy(book[amount_books].isbn, isbn);
46      book[amount_books].accession_number = accession_numer;
47
48      printf("\nThe book addition is complete.\n");
49
50      accession_numer++;
51      amount_books++;
52  }
```

- Delete

  Administrator can delete a book from the database.

```
void delete_book( )//show every book first(use search_book_name()). To make sure whet
{
    int delete_number=0, find=0;
    review_library(); // print every book first
    printf("Please enter the accession number of book you want to delete: ");
    scanf(" %d", &delete_number);

    for (int i = 0; i < amount_books; i++)
    {
        if(book[i].accession_number == delete_number)
        {
            int confirm=0;
            find = 1;
            while(1)
            {
                printf("============================================================\n");
                printf("| Are you sure you want to delete %-20s |\n", book[i].book_na
                printf("| 1. Yes%46s |\n", " ");
                printf("| 2. No%47s |\n", " ");
                printf("============================================================\n");
                printf("Please enter your option: ");
                scanf(" %d", &confirm);
                if(confirm > 2 || confirm < 1)
                {
                    CLEARSCREAN; // clear screen
                    printf("Error Option\n\n");
                    continue;
                }
                else if(confirm == 1)
                {
                    if(i == MAX_BUF - 1)
                    {
                        strcpy(book[i].book_name, "\0");
                        strcpy(book[i].author, "\0");
                        strcpy(book[i].publisher, "\0");
                        book[i].publish_year = 0;
                        strcpy(book[i].call_number, "\0");
                        strcpy(book[i].isbn, "\0");
                        book[i].viewer_count = 0;
                        book[i].accession_number = 0;
                        book[i].status = 0;
                    }
                    else
                    {
                        for(int j = i; j < amount_books; j++)
                        book[j] = book[j + 1];
                    }
```

```
                    printf("\nThe deletion of book is complete.\n");
                    amount_books--;
                    break;
                }
                else // confirm = 2
                    break;
            }
            break;
        }
    }

    if(find == 0)
        printf("\nThere is no book with accession number %d in the library.\n", delet

    return;
}
```

- Traverse
  Print all the books listed in the library.

```c
void review_library() // print every book
{
    int num_books = 0;

    CLEARSCREAN; // clear screen
    // show every books
    printf("================================================================================
    printf("| %20s%-30s| %6s%-14s| %5s%-15s| %-13s| %2s%-13s| %4s%-10s| %-14s| %-17s
    printf("================================================================================
    for (int i = 0; i < amount_books; i++)
    {
        printf("| %-50s| %-20s| %-20s| %-13d| %-15s| %-14s| %-14.0f| %-17d|",
                book[i].book_name, book[i].author, book[i].publisher, book[i].publis
                book[i].isbn, book[i].viewer_count, book[i].accession_number);
        if(book[i].status == 0)
            printf("    %s   |\n", "*");
        else
            printf("        |\n");
        printf("================================================================================
        num_books = 1;
    }

    // If there is no book in library
    if(num_books == 0)
    {
        printf("=======================================\n");
        printf("|     There is no book in library.    |\n");
        printf("=======================================\n");
    }
}
```

● Search

While readers enter the information of the book, the system will make all characters lower then compare by string function. Lastly, all the outcomes will print out on the screen.

```
void search_book_name()
{
    char temporary_book_name[SPACE];
    char book_name[SPACE];
    int found=0;
    CLEARSCREAN; // clear screen
    printf("Please enter the name of the book: ");
    fgets(book_name, sizeof(book_name), stdin);
    if(book_name[strlen(book_name) - 1] == '\n')
        book_name[strlen(book_name) - 1] = '\0';
    lowwer(book_name); // to lower case
    CLEARSCREAN; // clear screen
    for (int i = 0; i < amount_books; i++)
    {
        strcpy(temporary_book_name, book[i].book_name);
        lowwer(temporary_book_name); // to lower case

        if (strstr(temporary_book_name, book_name) != NULL)
        {
            if(found == 0)
            {
                printf("=================================================================
                printf("| %20s%-30s| %6s%-14s| %5s%-15s| %-13s| %2s%-13s| %4s%-10s| %
                printf("=================================================================
            }
            printf("| %-50s| %-20s| %-20s| %-13d| %-15s| %-14s| %-14.0f| %-17d|",
                    book[i].book_name, book[i].author, book[i].publisher, book[i].pub
                    book[i].isbn, book[i].viewer_count, book[i].accession_number);
            if(book[i].status == 0)
            printf("    %s   |\n", "*");
            else
                printf("        |\n");
            printf("=================================================================
            found = 1;
        }
    }

    if (found == 0)
    {
        printf("====================================\n");
        printf("| Sorry, There is no book found. |\n");
        printf("====================================\n");
    }
}

```

- Sort
  - Lines 6 to 17, and lines 25 to 30 implement the bubble sort algorithm.

```
1  void sort_publish_year()
2  {
3      static int order = SMALL;
4
5      /* Large first */
6      for(int i = 0; i < amount_books - 1; i++)
7      {
8          for(int j = 0; j < amount_books - i; j++)
9          {
10             if(book[j].publish_year < book[j + 1].publish_year)
11             {
12                 struct books temp = book[j];
13                 book[j] = book[j + 1];
14                 book[j + 1] = temp;
15             }
16         }
17     }
18
19     if(order == SMALL)
20         order = LARGE;
21     else if(order == LARGE)
22     {
23         order = SMALL;
24         /* Small first, reverse the structure */
25         for(int i = 0; i < (amount_books / 2); i++)
26         {
27             struct books temp = book[i];
28             book[i] = book[amount_books - i - 1];
29             book[amount_books - i - 1] = temp;
30         }
31     }
32 }
```

- File I/O
  - While the process starts to execute. It will call input_file(). Which will read the information from the text files. Before the process stops. The process will call the output_file(). Which will write all the new data into the text files. To update the information.

```c
FILE *fp_amount, *fp_administrator, *fp_book, *fp_reader, *fp_history;
struct readers *first = NULL;
struct Queue *queue;
void free_reader(struct readers *list)
{
    struct readers *temp;
    while(list != NULL)
    {
        temp = list;
        list = list -> next;
        free(temp);
    }
}
```

```c
void input_file()
{
    /* Open the amount.txt. Which store the amount of administrators, books, accessic
    if((fp_amount = fopen("amount.txt", "r+")) == NULL) // r+ can read and write. The
    {
        printf("Open amount.txt fail\n"); // Error message
        exit(0);
    }

    // Get the information from amount.txt. Store the value into variable
    fscanf(fp_amount ,"%d %d %d %d %d", &number_ad, &amount_books, &accession_numer,

    // Open administrator.txt. Which store the information of every administrator
    if((fp_administrator = fopen("administrator.txt", "r+")) == NULL) // r+ can read
    {
        printf("Open administrator.txt fail\n"); // Error message
        exit(0);
    }

    // Get the information from administrator.txt. Store into administrator
    fread(administrator, sizeof(struct administrators), number_ad, fp_administrator);

    // Open book.txt. Which store the information of every book
    if((fp_book = fopen("book.txt", "r+")) == NULL) // r+ can read and write. The fil
    {
        printf("Open book.txt fail\n"); // Error message
        exit(0);
    }
```

```
44
45      // Get the information from book.txt. Store into book
46      fread(book, sizeof(struct books), amount_books, fp_book);
47
48      // Open reader.txt. Which store the information of every reader
49      if((fp_reader = fopen("reader.txt", "r+")) == NULL) // r+ can read and write. Th
50      {
51          printf("Open reader.txt fail\n"); // Error message
52      }
53
54      // Readers' information is store in linked list
55      struct readers *cur;
56      struct readers *new_node;
57      cur = first;
```

```
58
59      for(int i = 0; i < amount_re; i++)
60      {
61          new_node = malloc(sizeof(struct readers));
62          fread(new_node, sizeof(struct readers) - sizeof(struct readers *), 1, fp_rea
63          new_node -> next = NULL; // Let next point to NULL
64
65          // Put into linked list
66          if(first == NULL)
67          {
68              first = new_node;
69              cur = first;
70          }
71          else
72          {
73              while(cur -> next != NULL)
74                  cur = cur -> next;
75              cur -> next = new_node; // push back
76          }
77
78      }
```

```
80      // Borrowing history
81      queue = createQueue();
82
83      if((fp_history = fopen("history.txt", "r+")) == NULL)
84      {
85          printf("Open history.txt fail\n"); // Error message
86          exit(0);
87      }
88
89      for(int i = 0; i < amount_history; i++)
90      {
91          struct history *newNode = malloc(sizeof(struct history));
92          fread(newNode, sizeof(struct history) - sizeof(struct history *), 1, fp_his
93          newNode -> next = NULL;
94          // If the queue is empty, make the new node both the front and rear
95          if (empty(queue)) {
96              queue->front = newNode;
97              queue->rear = newNode;
98          } else {
99              // Otherwise, add the new node to the rear and update the rear pointer
100             queue->rear->next = newNode;
101             queue->rear = newNode;
102         }
103     }
104
105     /* Back to the beginning of the file */
106     rewind(fp_amount);
107     rewind(fp_administrator);
108     rewind(fp_book);
109     rewind(fp_reader);
110     rewind(fp_history);
111 }
```

```c
void output_file()
{
    /* Use fprintf to write into file */
    fprintf(fp_amount, "%d %d %d %d %d\n", number_ad, amount_books, accession_numer,

    /* Use fwrite to write into file. Which is binary I/O */
    fwrite(administrator, sizeof(struct administrators), number_ad, fp_administrator

    /* Use fwrite to write into file. Which is binary I/O */
    fwrite(book, sizeof(struct books), amount_books, fp_book);

    /* Readers' information is store in linked list */
    struct readers *cur;
    cur = first;

    while(cur != NULL) // Run every node
    {
        /* Don't store the pointer information into file. It might point to weird pl
        fwrite(cur, sizeof(struct readers) - sizeof(struct readers *), 1, fp_reader)
        cur = cur -> next; // To next node
    }

    /* Borrowing history is store in Queue */
    struct history* currentNode = queue->front;
```

```c
    while (currentNode != NULL)
    {
        fwrite(currentNode, sizeof(struct history) - sizeof(struct history *), 1, fp
        currentNode = currentNode->next;
    }

    /* Free readers*/
    free_reader(first);

    /* Free history */
    while (!empty(queue))
        dequeue(queue);
    free(queue);

    /* Back to the beginning of the file */
    fclose(fp_amount);
    fclose(fp_administrator);
    fclose(fp_book);
    fclose(fp_reader);
    fclose(fp_history);
}
```

Advanced Part

- Good output format
  - Output book data in a table, enclose information in boxes.

```
printf("================================\n");
printf("| What do you prefer to do?    |\n");
printf("| 1. Modify reader information |\n");
printf("| 2. Delete reader             |\n");
printf("| 3. Exit                      |\n");
printf("================================\n");
```

- Switch screens (by using clear screen method).
  - Different commands are used to clear the screen based on different operating platforms.

```
#if _WIN32
    #define CLEARSCEARN system("cls")
#elif (__APPLE__)
    #define CLEARSCREAN system("clear")
#endif
```

- Use SDL to create buttons for added convenience.

```
void create_menu_window(const char* title) {
    SDL_Renderer *renderer = NULL;
    TTF_Font *font = NULL;
    SDL_Window* window =  SDL_CreateWindow(title, SDL_WINDOWPOS_UNDEFINED, SDL_WINDO
    if (window == NULL) {
        printf("無法創建視窗: %s\n", SDL_GetError());
        return ;
    }
    if (SDL_Init(SDL_INIT_VIDEO) < 0)
    {
        printf("SDL could not initialize! SDL_Error: %s\n", SDL_GetError());
        return ;
    }
    if (TTF_Init() < 0)
    {
        printf("SDL_ttf could not initialize! SDL_ttf Error: %s\n", TTF_GetError());
        return ;
    }
```

```
    renderer = SDL_CreateRenderer(window, -1, 0);
    SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
    if (renderer == NULL)
    {
        printf("Renderer could not be created! SDL_Error: %s\n", SDL_GetError());
        return ;
    }
    font = TTF_OpenFont("C:/Windows/WinSxS/amd64_microsoft-windows-font-truetype-cam
    if (font == NULL)
    {
        printf("Failed to load font! SDL_ttf Error: %s\n", TTF_GetError());
        return ;
    }
    SDL_Texture *texture_reader = NULL;
    SDL_Texture *texture_admin = NULL;
    SDL_Texture *texture_exit = NULL;
    SDL_Color textColor = {255, 255, 255, 255};
    SDL_Surface *surface_reader = TTF_RenderText_Solid(font, "Reader", textColor);
    SDL_Surface *surface_admin = TTF_RenderText_Solid(font, "Admin", textColor);
    SDL_Surface *surface_exit = TTF_RenderText_Solid(font, "Exit", textColor);
    texture_reader = SDL_CreateTextureFromSurface(renderer, surface_reader);
    texture_admin = SDL_CreateTextureFromSurface(renderer, surface_admin);
    texture_exit = SDL_CreateTextureFromSurface(renderer, surface_exit);
    SDL_FreeSurface(surface_reader);
    SDL_FreeSurface(surface_admin);
    SDL_FreeSurface(surface_exit);
    SDL_Surface* imageSurface = IMG_Load("E:/C/.vscode/finalproject2/1.png");
    SDL_Texture* backgroundTexture = SDL_CreateTextureFromSurface(renderer, imageSur
    SDL_FreeSurface(imageSurface);
```

```
    SDL_Rect rect_reader; //create rect
    SDL_Rect rect_admin;
    SDL_Rect button_exit;
    rect_reader.x = 30;
    rect_reader.y = 50;
    rect_reader.w = 150;
    rect_reader.h = 70;

    rect_admin.x = 200;
    rect_admin.y = 50;
    rect_admin.w = 130;
    rect_admin.h = 70;

    button_exit.x = 380;
    button_exit.y = 50;
    button_exit.w = 120;
    button_exit.h = 70;

    SDL_Event choose;
    bool quit = false;
    while(!quit){
        while(SDL_PollEvent(&choose)){
            if (choose.type == SDL_QUIT) {
                choose_menu = -1;
                quit = true;
                break;
            }
```

```
            if(choose.type == SDL_MOUSEBUTTONDOWN){
                int x = choose.button.x;
                int y = choose.button.y;
                if (x > rect_reader.x && x < rect_reader.x + rect_reader.w &&
                    y > rect_reader.y && y < rect_reader.y + rect_reader.h){
                        choose_menu=1;
                        quit = true;
                        break;
                    }
                if (x > rect_admin.x && x < rect_admin.x + rect_admin.w &&
                    y > rect_admin.y && y < rect_admin.y + rect_admin.h){
                        choose_menu=2;
                        quit = true;
                        break;
                    }
                if (x > button_exit.x && x < button_exit.x + button_exit.w &&
                    y > button_exit.y && y < button_exit.y + button_exit.h){
                        choose_menu=3;
                        quit = true;
                        break;
                    }
            }
            SDL_RenderClear(renderer);
            // 渲染背景纹理到整个窗口
            SDL_RenderCopy(renderer, backgroundTexture, NULL, NULL);
            SDL_RenderCopy(renderer, texture_reader, NULL, &rect_reader);
            SDL_RenderCopy(renderer, texture_admin, NULL, &rect_admin);
            SDL_RenderCopy(renderer, texture_exit, NULL, &button_exit);
            SDL_RenderPresent(renderer);
        }
    }
    TTF_CloseFont(font);
    SDL_DestroyTexture(texture_reader);
    SDL_DestroyTexture(texture_admin);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyTexture(backgroundTexture);

    SDL_DestroyWindow(window);
    TTF_Quit();
    SDL_Quit();
}
```

- Store and access data by queue.
  - We use the characteristic of a queue to store the history of borrowing. We limit the data number. If the store of data reaches the limit, it will pop the oldest data. And add the newest data. The first function is to initialize the queue. Second one is checking whether the queue is empty. Then add a node of queue to store data. The last one is pop the oldest node of the queue.

```c
struct Queue *createQueue()
{
    struct Queue *queue = malloc(sizeof(struct Queue));
    queue->front = NULL;
    queue->rear = NULL;
    return queue;
}

// Function to check if the queue is empty
int empty(struct Queue *queue)
{
    return (queue->front == NULL);
}
```

```c
// Function to add an element to the rear of the queue
void enqueue(struct Queue *queue, char re_name[], char book_name[])
{
    struct history *newNode = malloc(sizeof(struct history));
    strcpy(newNode->re_name, re_name);
    strcpy(newNode->book_name, book_name);
    newNode -> next = NULL;

    // If the queue is empty, make the new node both the front and rear
    if (empty(queue))
    {
        queue->front = newNode;
        queue->rear = newNode;
    } else
    {
        // Otherwise, add the new node to the rear and update the rear pointer
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}
```

```c
void dequeue(struct Queue* queue)
{
    // If the queue is empty
    if (empty(queue))
        printf("Error: Queue is empty.\n");

    // Store the front node
    struct history *frontNode = queue->front;

    // Move the front pointer to the next node
    queue->front = queue->front->next;

    // If the front becomes NULL, update the rear pointer as well
    if (queue->front == NULL)
        queue->rear = NULL;

    // Free the memory occupied by the dequeued node
    free(frontNode);
}
```

- Design an account and password login system.
  - First we ask the user to input their account, and then use a function to check if the registered account is valid (function at line 12). After validating the account, we ask the user to input their password and use a function to check if the password is valid (function at line 26). This completes the registration process.

```
1  void add_reader()
2  {
3      int success=0;
4      char name[SPACE], email[SPACE], account[SPACE], password[SPACE];
5      struct readers *new_reader;
6
7      new_reader = malloc(sizeof(struct readers));
8
9      CLEARSCREAN; // clear the screen
10     printf("Please enter your name: ");
11     fgets(name, SPACE, stdin);
12     if(name[strlen(name) - 1] == '\n')
13         name[strlen(name) - 1] = '\0';
14
15     printf("Please enter your student id: ");
16     scanf("%d", &new_reader -> student_id);
17
18     fflush(stdin);
19
20     int detect=0;
21
22     detect = setjmp(emailbuffer);
23     if(detect == 1)
24         printf("\n");
25
26     printf("Please enter your email: ");
27     fgets(email, SPACE, stdin);
28     if(email[strlen(email) - 1] == '\n')
29         email[strlen(email) - 1] = '\0';
30
31     check_email(email);
```

```
33     while (1)
34     {
35         printf("Please enter your account: ");
36         fgets(account, SPACE, stdin);
37         if(account[strlen(account) - 1] == '\n')
38             account[strlen(account) - 1] = '\0';
39
40         success = set_check_account_re(name, new_reader -> student_id, email, accoun
41         if(success == YES)
42             break;
43     }
44     strcpy(new_reader -> re_account, account);
45
46     printf("Please enter your password: ");
47     fgets(password, SPACE, stdin);
48     if(password[strlen(password) - 1] == '\n')
49         password[strlen(password) - 1] = '\0';
```

```
51      /* Store it */
52      strcpy(new_reader -> re_name, name);
53      strcpy(new_reader -> email, email);
54      strcpy(new_reader -> re_account, account);
55      strcpy(new_reader -> re_password, password);
56
57      new_reader -> next = NULL;
58      amount_re++;
59
60      if(first == NULL)
61          first = new_reader;
62      else /* Push back */
63      {
64          struct readers *cur;
65          cur = first;
66          while(cur -> next != NULL)
67              cur = cur -> next;
68          cur -> next = new_reader;
69      }
70  }
```

- Verify if the email format is correct.
  - We write a simple function to check the format of email. First, there should be an ‘@’ in email. Second, there shouldn’t be more than one ‘@’. Next, there couldn’t exist ‘#’, ‘*’, ‘..’ in email. Last, there should be at least one ‘.’ behind ‘@’ .

```c
void check_email(char email[])
{
    int error = 0;
    int counter = 0;
    char* cptr;
    char* temp;

    printf("\n=====================================================\n");

    if ((cptr = strchr(email, '@')) == NULL) {
        printf("| Email format error: at least one @              |\n");
        printf("=====================================================\n");
        error++;
    }
    else
    {
        temp = cptr;
        while ((cptr = strchr(temp, '@')) != NULL) {
            counter++;
            temp = cptr + 1;
        }

        if (counter > 1) {
            printf("Email format error: more than one @               |\n");
            printf("=====================================================\n");
            error++;
        }
    }
```

```
30      char* sharp;
31      sharp = strchr(email, '#');
32      while (sharp != NULL) {
33          printf("| Email format error: can't exist #               |\n");
34          printf("=================================================\n");
35          error++;
36          sharp = strchr(sharp + 1, '#');
37      }
38
39      char* asterisk;
40      asterisk = strchr(email, '*');
41      while (asterisk != NULL) {
42          printf("| Email format error: can't exist *               |\n");
43          printf("=================================================\n");
44          error++;
45          asterisk = strchr(asterisk + 1, '*');
46      }
47
48      char* doubledot;
49      doubledot = strstr(email, "..");
50      while (doubledot != NULL) {
51          printf("| Email format error: can't exist ..              |\n");
52          printf("=================================================\n");
53          error++;
54          doubledot = strstr(doubledot + 1, "..");
55      }
```

```
57      char* at = NULL;
58      char* dot = NULL;
59      if ((at = strchr(email, '@')) != NULL) {
60          dot = strchr(at + 1, '.');
61          if (dot == NULL) {
62              printf("| Email format error: at least one . behind the @ |\n");
63              printf("=================================================\n");
64              error++;
65          }
66      }
67
68      if (error != 0)
69          longjmp(emailbuffer, 1);
70      printf("| Email format correct!                           |\n");
71      printf("=================================================\n\n");
72  }
```

● When selecting to modify or delete data, all data will be displayed first for the user's convenience in confirmation.

- In the function "modify_ad_information", we call the function "check_ad_information" (line 16), to print all data before making any changes

```
1  void check_ad_information()
2  {
3      CLEARSCREAN; // clear screen
4      printf("=========================================================================\n
5      printf("| %1s%-19s | %5s%-15s | %5s%-15s |\n", " ", "Administrator Name", " ", "
6      printf("=========================================================================\n
7      for (int i = 0; i < number_ad; i++)
8      {
9          printf("| %-20s | %-20s | %-20s |\n", administrator[i].ad_name, administrato
10         printf("=========================================================================
11     }
12 }
13
14 void modify_ad_information()
15 {
16     check_ad_information();
17     char temp_ad_name[30];
18     printf("Please enter the administrator name which you want to modify: ");
19     .
20     .
21     .
22     ...omitted.
```

Demostration

Github repository link:
https://github.com/1508leo/PD_Final_Project