Mastering Embedded System Online Diploma

www.learn-in-depth.com

First term (Final Project 2)

Project : Students Database Using Queue

Eng. Mohamed Adel

My Profile
https://www.learn-in-depth.com/online-diploma/mohamedaadelezz3%40gmail.com

Table OF Contents

1. Chapter One : Introduction

Case Study :

Creating a students database where you can :

1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information

The data structure used is Queue (FIFO) first in first output

The code is divided into a Queue header file where the main definitions for the queue implementation is present

The other header file is a students database header file where all the APIS is present and it’s included in the main function

2. Chapter Two : Code Implementation

2.1Main

```
 *  Created on: May 22, 2022
 *      Author: Mohamed Adel
 */

#include "students_database.h"

int main(){

    int choice;

    DPRINTF("Welcome to the student mangement system\n");
    while(1){

        DPRINTF("Choose the task that you want to perform\n");
        DPRINTF("1. Add the student details manually\n");
        DPRINTF("2. Add the student details from text file\n");
        DPRINTF("3. Find the student details by roll number \n");
        DPRINTF("4. Find the student details by first name\n");
        DPRINTF("5. Find the student details by course ID\n");
        DPRINTF("6. Find the total number of students\n");
        DPRINTF("7. Delete the students details by roll number\n");
        DPRINTF("8. Update the students details by roll number\n");
        DPRINTF("9. Show all information\n");
        DPRINTF("10. To Exit\n");
        DPRINTF("Enter your choice to perform the task: ");
        scanf("%d",&choice);

        switch(choice){

        case 1:
            add_student_manually();
            break;
        case 2:
            add_student_file();
            break;
        case 3:
            find_by_roll_no();
            break;
        case 4:
            find_by_first_name();
            break;
        case 5:
            find_by_course_id();
            break;
        case 6:
            find_total_no_of_students();
            break;
        case 7:
            delete_student();
            break;
        case 8:
            update_student();
            break;
        case 9:
            show_student();
            break;
        case 10:
            return 0;
            break;
        default :
            DPRINTF("This choice is wrong please enter another one\n");
            break;
        }
    }
    return 0;
}
```

2.2Queue

```c
/*
 * queue.c
 *
 *  Created on: May 22, 2022
 *      Author: Mohamed Adel
 */

#include "queue.h"
QUEUE_STAT QUEUE_init(QUEUE_DATA_TYPE * container,unsigned int len ,QUEUE_BUFFER * buffer){

    if(container){

        buffer->base = container;
        buffer->head = container;
        buffer->tail = container;
        buffer->length = len;
        buffer->count = 0;

        return QUEUE_NO_ERROR;
    }
    else
        return QUEUE_NULL;
}

QUEUE_STAT QUEUE_add(QUEUE_BUFFER *buffer,QUEUE_DATA_TYPE student){

    if(!buffer->base||!buffer->head||!buffer->tail)
        return QUEUE_NULL;
    else if(QUEUE_isFull(buffer) == QUEUE_FULL)
        return QUEUE_FULL;

    else{
        int i;
        QUEUE_DATA_TYPE *temp = buffer->tail;

        for(i = 0; i < buffer->count; i++){
            if(temp->roll == student.roll){
                return QUEUE_ITEM_ID_NOT_UNIQUE;
            }
            temp++;
        }

        *(buffer->head) = student;
        buffer->count++;

        if(buffer->head == ((buffer->base)+((buffer->length)-1)))
            buffer->head = buffer->base;
        else
            (buffer->head)++;

        return QUEUE_NO_ERROR;
    }
}

QUEUE_STAT QUEUE_findByRL(QUEUE_BUFFER *buffer,int rl,QUEUE_DATA_TYPE * pstudent,int objective){

    if(!buffer->base||!buffer->head||!buffer->tail)
        return QUEUE_NULL;
    else if(QUEUE_isEmpty(buffer) == QUEUE_EMPTY)
        return QUEUE_EMPTY;
    else{
        int i, j;
        QUEUE_DATA_TYPE *temp = buffer->tail;
        for(i = 0; i < buffer->count; i++){
            if(rl == temp->roll){
                if(objective == QUEUE_FIND)
                    *pstudent = *temp;
                else if(objective == QUEUE_UPDATE)
                    *temp = *pstudent;
                else{
                    for(j = i; j < buffer->count; j++){
                        if(j == buffer->count-1){
                            QUEUE_DATA_TYPE null_student = {0};
```

```
QUEUE_STAT QUEUE_findByRL(QUEUE_BUFFER *buffer,int rl,QUEUE_DATA_TYPE * pstudent,int objective){

    if(!buffer->base||!buffer->head||!buffer->tail)
        return QUEUE_NULL;
    else if(QUEUE_isEmpty(buffer) == QUEUE_EMPTY)
        return QUEUE_EMPTY;
    else{
        int i, j;
        QUEUE_DATA_TYPE *temp = buffer->tail;
        for(i = 0; i < buffer->count; i++){
            if(rl == temp->roll){
                if(objective == QUEUE_FIND)
                    *pstudent = *temp;
                else if(objective == QUEUE_UPDATE)
                    *temp = *pstudent;
                else{
                    for(j = i; j < buffer->count; j++){
                        if(j == buffer->count-1){
                            QUEUE_DATA_TYPE null_student = {""};
                            *temp = null_student;
                        }
                        else{
                            *temp = *(temp+1);
                            temp++;
                        }
                    }

                    buffer->count--;
                    if(buffer->head == buffer->base)
                        buffer->head = ((buffer->base)+((buffer->length)-1));
                    else
                        buffer->head--;
                }
                return QUEUE_NO_ERROR;
            }
            temp++;
        }
        return QUEUE_ITEM_NOT_FOUND;
    }

}

QUEUE_STAT QUEUE_findByFName(QUEUE_BUFFER *buffer,char *name,QUEUE_DATA_TYPE * student){

    if(!buffer->base||!buffer->head||!buffer->tail)
        return QUEUE_NULL;
    else if(QUEUE_isEmpty(buffer) == QUEUE_EMPTY)
        return QUEUE_EMPTY;
    else{
        int i, j = 0;
        QUEUE_DATA_TYPE *temp = buffer->tail;
        for(i = 0; i < buffer->count; i++){
            if(!stricmp(name,temp->fname)){
                *(student+j) = *temp;
                j++;
            }
            temp++;
        }
        if(j == 0)
            return QUEUE_ITEM_NOT_FOUND;
        else
            return QUEUE_NO_ERROR;
    }
}
```

```
QUEUE_STAT QUEUE_findByCID(QUEUE_BUFFER *buffer,int cid,QUEUE_DATA_TYPE * student){

    if(!buffer->base||!buffer->head||!buffer->tail)
        return QUEUE_NULL;
    else if(QUEUE_isEmpty(buffer) == QUEUE_EMPTY)
        return QUEUE_EMPTY;
    else{
        int i, j = 0, k;
        QUEUE_DATA_TYPE *temp = buffer->tail;
        for(i = 0; i < buffer->count; i++){
            for(k = 0; k < 5; k++){
                if(cid == temp->cid[k]){
                    *(student+j) = *temp;
                    j++;
                }
            }
            temp++;
        }
        if(j == 0)
            return QUEUE_ITEM_NOT_FOUND;
        else
            return QUEUE_NO_ERROR;
    }
}
void QUEUE_printAll(QUEUE_BUFFER *buffer){

    DPRINTF("--------------------------\n");
    if(!buffer->base||!buffer->head||!buffer->tail){
        DPRINTF("[ERROR] Students database is not yet initialized\n");
    }
    else if(QUEUE_isEmpty(buffer) == QUEUE_EMPTY){
        DPRINTF("[ERROR] Students database is empty \n");
    }
    else{
        int i, j;
        QUEUE_DATA_TYPE * temp = buffer->tail;
        for(i = 0; i < buffer->count; i++){

            DPRINTF("Student first name: %s\n",temp->fname);
            DPRINTF("Student last name: %s\n",temp->lname);
            DPRINTF("Student roll number: %d\n",temp->roll);
            DPRINTF("Student GPA: %.2f\n",temp->gpa);
            for(j = 0; j < 5; j++){
                DPRINTF(" The course ID is: %d\n",temp->cid[j]);
            }
            temp++;
            DPRINTF("--------------------------\n");

        }
    }
}

QUEUE_STAT QUEUE_isFull(QUEUE_BUFFER *buffer){
    if(buffer){
        if(buffer->count == buffer->length)
            return QUEUE_FULL;
        else
            return QUEUE_NO_ERROR;
    }else
        return QUEUE_NULL;
}

QUEUE_STAT QUEUE_isEmpty(QUEUE_BUFFER *buffer){
    if(buffer){
        if(buffer->count == 0)
            return QUEUE_EMPTY;
        else
            return QUEUE_NO_ERROR;
    }else
        return QUEUE_NULL;
```

2.3Students Database

```c
/*
 * students_database.c
 *
 *  Created on: May 22, 2022
 *      Author: Mohamed Adel
 */
#include "students_database.h"

QUEUE_BUFFER buff;

unsigned char init = 0;

void add_student_file(){
    int count = 0;
    struct sinfo student;
    if(init){
        FILE *fp;
        fp = fopen("students.txt","r");

        if(!fp){
            DPRINTF("Couldn't read file\n");
        }
        else{
            while(fscanf(fp,"%d %s %s %f %d %d %d %d %d[^\n]",&(student.roll),student.fname,student.lname,
                    &(student.gpa),&(student.cid[0]),&(student.cid[1]),
                    &(student.cid[2]),&(student.cid[3]),&(student.cid[4]))!=EOF){

                switch(QUEUE_add(&buff,student)){
                case QUEUE_NO_ERROR:
                    DPRINTF("[INFO] Roll number %d saved successfully\n",student.roll);
                    count++;
                    break;
                case QUEUE_ITEM_ID_NOT_UNIQUE:
                    DPRINTF("[ERROR] Roll number %d is already taken\n",student.roll);
                    break;
                default:
                    DPRINTF("[ERROR]  Students database is full\n");
                    break;
                }

            }
            if(count > 0){
                DPRINTF("[INFO] Students details were added successfully\n");
            }
            find_total_no_of_students();
        }
        fclose(fp);
    }else{
        if(QUEUE_init(st,SIZE,&buff) == QUEUE_NO_ERROR){

            init = 1;
            FILE *fp;

            fp=fopen("students.txt","r");
            if(!fp){
                DPRINTF("Couldn't read file\n");
            }
            else{
                while(fscanf(fp,"%d %s %s %f %d %d %d %d %d[^\n]",&(student.roll),student.fname,student.lname,
                        &(student.gpa),&(student.cid[0]),&(student.cid[1]),
                        &(student.cid[2]),&(student.cid[3]),&(student.cid[4]))!=EOF){
                    switch(QUEUE_add(&buff,student)){
                    case QUEUE_NO_ERROR:
                        DPRINTF("[INFO] Roll number %d saved successfully\n",student.roll);
                        count++;
                        break;
                    case QUEUE_ITEM_ID_NOT_UNIQUE:
                        DPRINTF("[ERROR] Roll number %d is already taken\n",student.roll);
                        break;
                    default:
                        DPRINTF("[ERROR]  Students database is full\n");
                        break;
```

```c
void add_student_manually(){

    int i;
    struct sinfo student;

    if(init){
        DPRINTF("-------------------------\n");
        DPRINTF("Add the student details\n");
        DPRINTF("-------------------------\n");
        DPRINTF("Enter the roll number: ");
        scanf("%d",&student.roll);
        DPRINTF("Enter the first name of student: ");
        gets(student.fname);
        DPRINTF("Enter the last name of student: ");
        gets(student.lname);
        DPRINTF("Enter the GPA: ");
        scanf("%f",&student.gpa);
        DPRINTF("Enter the course ID for each course: \n");
        for(i = 0; i < 5; i++){
            DPRINTF("Course %d ID: ",i+1);
            scanf("%d",&student.cid[i]);
        }
        switch(QUEUE_add(&buff,student)){

        case QUEUE_NO_ERROR:
            DPRINTF("[INFO] Student details was added successfully\n");
            break;
        case QUEUE_ITEM_ID_NOT_UNIQUE:
            DPRINTF("[ERROR] Roll number %d is already taken\n",student.roll);
            break;
        default:
            DPRINTF("[ERROR]  Students database is full\n");
            break;
        }
        find_total_no_of_students();
    }
    else{
        if(QUEUE_init(st,SIZE,&buff) == QUEUE_NO_ERROR){
            init = 1;

            DPRINTF("-------------------------\n");
            DPRINTF("Add the student details\n");
            DPRINTF("-------------------------\n");
            DPRINTF("Enter the roll number: ");
            scanf("%d",&student.roll);
            DPRINTF("Enter the first name of student: ");
            gets(student.fname);
            DPRINTF("Enter the last name of student: ");
            gets(student.lname);
            DPRINTF("Enter the GPA: ");
            scanf("%f",&student.gpa);
            DPRINTF("Enter the course ID for each course\n");
            for(i = 0; i < 5; i++){
                DPRINTF("Course %d ID: ",i+1);
                scanf("%d",&student.cid[i]);
            }
            switch(QUEUE_add(&buff,student)){

            case QUEUE_NO_ERROR:
                DPRINTF("[INFO] Student details was added successfully\n");
                break;
            case QUEUE_ITEM_ID_NOT_UNIQUE:
                DPRINTF("[ERROR] Roll number %d is already taken\n",student.roll);
                break;
            default:
                DPRINTF("[ERROR] Students database is full\n");
                break;
            }
            find_total_no_of_students();
        }
        else{
            DPRINTF("Error in initialization\n");
```

```c
void find_by_roll_no(){
    int rl, i;
    struct sinfo student;

    DPRINTF("Enter the roll number of the student: ");
    scanf("%d",&rl);

    switch(QUEUE_findByRL(&buff,rl,&student,QUEUE_FIND)){
    case QUEUE_NO_ERROR:
        DPRINTF("The student's details are\n");
        DPRINTF("The first name is %s\n",student.fname);
        DPRINTF("The last name is %s\n",student.lname);
        DPRINTF("The GPA is %.2f\n",student.gpa);
        DPRINTF("The course ID of each course\n");
        for(i = 0; i < 5; i++){
            DPRINTF(" The course ID is: %d\n",student.cid[i]);
        }
        break;
    case QUEUE_ITEM_NOT_FOUND:
        DPRINTF("[ERROR] Roll number %d was not found\n",rl);
        break;
    default:
        DPRINTF("[ERROR] Students database is empty or not yet initialized\n");
        break;

    }


}
void find_by_first_name(){
    char name[50];
    int j, i = 0;
    struct sinfo student[SIZE] = {{""}};

    DPRINTF("Enter the first name of the student: ");
    gets(name);

    switch(QUEUE_findByFName(&buff,name,student)){
    case QUEUE_NO_ERROR:
        while(student[i].roll){
            DPRINTF("The student's details are\n");
            DPRINTF("The first name is %s\n",student[i].fname);
            DPRINTF("The last name is %s\n",student[i].lname);
            DPRINTF("The roll number is %d\n",student[i].roll);
            DPRINTF("The GPA is %.2f\n",student[i].gpa);
            DPRINTF("The course ID of each course\n");
            for(j = 0; j < 5; j++){
                DPRINTF(" The course ID is: %d\n",student[i].cid[j]);
            }
            i++;
        }
        break;
    case QUEUE_ITEM_NOT_FOUND:
        DPRINTF("[ERROR] First name %s was not found\n",name);
        break;
    default:
        DPRINTF("[ERROR] Students database is empty or not yet initialized\n");
        break;
    }
}

void find_by_course_id(){

    int cid, i = 0;
    struct sinfo student[SIZE] = {{""}};

    DPRINTF("Enter the course ID: ");
    scanf("%d",&cid);
```

```c
void find_by_course_id(){

    int cid, i = 0;
    struct sinfo student[SIZE] = {{""}};

    DPRINTF("Enter the course ID: ");
    scanf("%d",&cid);

    switch(QUEUE_findByCID(&buff,cid,student)){
    case QUEUE_NO_ERROR:
        while(student[i].roll){
            DPRINTF("The student's details are\n");
            DPRINTF("The first name is %s\n",student[i].fname);
            DPRINTF("The last name is %s\n",student[i].lname);
            DPRINTF("The roll number is %d\n",student[i].roll);
            DPRINTF("The GPA is %.2f\n",student[i].gpa);
            i++;
        }
        DPRINTF("[INFO] Total numeber of students enrolled: %d\n",i);
        break;
    case QUEUE_ITEM_NOT_FOUND:
        DPRINTF("[ERROR] Course ID %d was not found\n",cid);
        break;
    default:
        DPRINTF("[ERROR] Students database is empty or not yet initialized\n");
        break;
    }

}
void find_total_no_of_students(){

    if(init){
        DPRINTF("-------------------------\n");
        DPRINTF("[INFO] Total number of students is %d\n",buff.count);
        DPRINTF("[INFO] You can add up to %d students\n",buff.length);
        DPRINTF("[INFO] You can add %d more students\n",buff.length-buff.count);
    }
    else{
        DPRINTF("[ERROR] Students database is empty or not yet initialized\n");
    }
}
void delete_student(){

    int rl;
    struct sinfo student;

    DPRINTF("Enter the roll number which you want to delete: ");
    scanf("%d",&rl);
    switch(QUEUE_findByRL(&buff,rl,&student,QUEUE_DELETE)){
    case QUEUE_NO_ERROR:
        DPRINTF("[INFO] Roll number %d was deleted successfully\n",rl);
        break;
    case QUEUE_ITEM_NOT_FOUND:
        DPRINTF("[ERROR] Roll number %d was not found\n",rl);
        break;
    default:
        DPRINTF("[ERROR] Students database is empty or not yet initialized\n");
        break;
    }

}
void update_student(){

    int i, rl, choice;
    struct sinfo student;
    DPRINTF("Enter the roll number to update the entry: ");
    scanf("%d",&rl);
    switch(QUEUE_findByRL(&buff,rl,&student,QUEUE_FIND)){
    case QUEUE_NO_ERROR:
        while(choice <= 0 || choice > 5){
```

```
void update_student(){

    int i, rl, choice;
    struct sinfo student;
    DPRINTF("Enter the roll number to update the entry: ");
    scanf("%d",&rl);
    switch(QUEUE_findByRL(&buff,rl,&student,QUEUE_FIND)){
    case QUEUE_NO_ERROR:
        while(choice <= 0 || choice > 5){
            DPRINTF("1.First name\n");
            DPRINTF("2.Last name\n");
            DPRINTF("3.Roll no.\n");
            DPRINTF("4.GPA\n");
            DPRINTF("5.Courses ID\n");
            scanf("%d",&choice);
            switch(choice){
            case 1:
                DPRINTF("Enter new first name: ");
                gets(student.fname);
                QUEUE_findByRL(&buff,rl,&student,QUEUE_UPDATE);
                DPRINTF("[INFO] Updated successfully\n");
                break;
            case 2:
                DPRINTF("Enter new last name: ");
                gets(student.lname);
                QUEUE_findByRL(&buff,rl,&student,QUEUE_UPDATE);
                DPRINTF("[INFO] Updated successfully\n");
                break;
            case 3:
                DPRINTF("Enter new roll no.: ");
                scanf("%d",&student.roll);
                if(QUEUE_findByRL(&buff,student.roll,&student,QUEUE_FIND)==QUEUE_ITEM_NOT_FOUND){
                    QUEUE_findByRL(&buff,rl,&student,QUEUE_UPDATE);
                    DPRINTF("[INFO] Updated successfully\n");
                }else{
                    DPRINTF("[ERROR] Roll number %d is already taken\n",student.roll); }
                break;
            case 4:
                DPRINTF("Enter new GPA: ");
                scanf("%f",&student.gpa);
                QUEUE_findByRL(&buff,rl,&student,QUEUE_UPDATE);
                DPRINTF("[INFO] Updated successfully\n");
                break;
            case 5:
                DPRINTF("Enter new courses ID: \n");
                for(i=0;i<5;i++){
                    DPRINTF("Course %d ID: ",i+1);
                    scanf("%d",&student.cid[i]);
                }
                QUEUE_findByRL(&buff,rl,&student,QUEUE_UPDATE);
                DPRINTF("[INFO] Updated successfully\n");
                break;
            default:
                DPRINTF("[ERROR] Invalid input\n");
                break;
            }
        }
        break;
    case QUEUE_ITEM_NOT_FOUND:
        DPRINTF("[ERROR] Roll number %d was not found\n",rl);
        break;
    default:
        DPRINTF("[ERROR] Students database is not yet initialized\n");
        break;
    }

}
void show_student(){

    if(init){
        QUEUE_printAll(&buff);
    }
```

3. Chapter Three : Testing Program

3.1Adding student from text file

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 2
[ERROR] Roll number 1 is already taken
[INFO] Roll number 2 saved successfully
[INFO] Roll number 3 saved successfully
[INFO] Roll number 5 saved successfully
[INFO] Students details were added successfully
---------------------------
[INFO] Total number of students is 4
[INFO] You can add up to 50 students
[INFO] You can add 46 more students
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.2Adding Student Manually

```
Welcome to the student mangement system
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 1
-------------------------
Add the student details
-------------------------
Enter the roll number: 1
Enter the first name of student: Mohamed
Enter the last name of student: Adel
Enter the GPA: 3.6
Enter the course ID for each course
Course 1 ID: 1
Course 2 ID: 2
Course 3 ID: 3
Course 4 ID: 4
Course 5 ID: 5
[INFO] Student details was added successfully
-------------------------
[INFO] Total number of students is 1
[INFO] You can add up to 50 students
[INFO] You can add 49 more students
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.3Find the student details by roll number

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 3
Enter the roll number of the student: 1
The student's details are
The first name is Mohamed
The last name is Adel
The GPA is 3.60
The course ID of each course
 The course ID is: 1
 The course ID is: 2
 The course ID is: 3
 The course ID is: 4
 The course ID is: 5
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.4Find the student by first name

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 4
Enter the first name of the student: Mohamed
The student's details are
The first name is Mohamed
The last name is Adel
The roll number is 1
The GPA is 3.60
The course ID of each course
 The course ID is: 1
 The course ID is: 2
 The course ID is: 3
 The course ID is: 4
 The course ID is: 5
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.5Find the student by course ID

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 5
Enter the course ID: 1
The student's details are
The first name is Mohamed
The last name is Adel
The roll number is 1
The GPA is 3.60
[INFO] Total numeber of students enrolled: 1
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.6Find the total number of students

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 6
-------------------------
[INFO] Total number of students is 4
[INFO] You can add up to 50 students
[INFO] You can add 46 more students
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.7Delete a student

```
Enter your choice to perform the task: 6
-------------------------
[INFO] Total number of students is 4
[INFO] You can add up to 50 students
[INFO] You can add 46 more students
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 7
Enter the roll number which you want to delete: 1
[INFO] Roll number 1 was deleted successfully
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task:
```

3.8Update student details

```
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 8
Enter the roll number to update the entry: 9
[ERROR] Roll number 9 was not found
Choose the task that you want to perform
1. Add the student details manually
2. Add the student details from text file
3. Find the student details by roll number
4. Find the student details by first name
5. Find the student details by course ID
6. Find the total number of students
7. Delete the students details by roll number
8. Update the students details by roll number
9. Show all information
10. To Exit
Enter your choice to perform the task: 8
Enter the roll number to update the entry: 5
1.First name
2.Last name
3.Roll no.
4.GPA
5.Courses ID
4
Enter new GPA: 3.1
[INFO] Updated successfully
Choose the task that you want to perform
1. Add the student details manually
```