One stop Management System ::::

Code :   
#include<iostream>

#include<ctime>

#include<string>

#include<queue>

#include<stack>

using namespace std;

static int count;

class Ticket\_Management{

public :

int id;

string name;

int priority;

string service\_description;

time\_t Creation\_time;

bool Status;

time\_t close\_time;

Ticket\_Management\* next\_tick = NULL;

Ticket\_Management(string nm, string desc, int p, int i){

id = i;

name = nm;

service\_description = desc;

priority = p;

Creation\_time = time(0);

close\_time = 0;

Status = true;

Ticket\_Management\* next\_tick = NULL;

}

};

class Ticket\_list{

public :

Ticket\_Management\* head;

stack<Ticket\_Management\*> Ticket\_Resolution\_log;

queue<Ticket\_Management\*> Pending\_Ticket;

Ticket\_list(){

head = NULL;

count = 1;

}

Ticket\_Management\* Add\_Ticket(string nm, string des, int p, int i){

Ticket\_Management\* ticket = new Ticket\_Management(nm,des,p,i);

Pending\_Ticket.push(ticket); //Push Pending ticket in Queue

if(head == NULL){

head = ticket;

cout<<"Ticket with id : "<<ticket->id<<", Insert Successfully at Time : "<<ctime(&ticket->Creation\_time)<<endl;

return head;

}else{

Ticket\_Management\* Temp = head;

while(Temp->next\_tick != NULL){

Temp = Temp->next\_tick;

}

Temp->next\_tick = ticket;

ticket->next\_tick = NULL;

cout<<"Ticket with id : "<<ticket->id<<", Insert Successfully at Time : "<<ctime(&ticket->Creation\_time)<<endl;

return ticket;

}

}

void Remove\_Ticket(int search){

if(head == NULL){

cout<<"No Ticket to Remove....\a"<<endl;

return;

}

Ticket\_Management\* Temp = head;

while(Temp != NULL){

Temp = Temp->next\_tick;

if(Temp->id == search){

if(Temp->Status){

Temp->Status = false;

Pending\_Ticket.pop();

Ticket\_Resolution\_log.push(Temp);

Temp->close\_time = time(0);

cout<<"Ticket with id : "<<search<<" is Remove Successfully at Time : "<<ctime(&Temp->close\_time)<<endl;

return;

}

}

}

cout<<"Ticket with id : "<<search<<", not found...\a"<<endl;

}

bool Search(string nm, int i){

if(head == NULL){

cout<<"Ticket List is Empty....\a"<<endl;

return false;

}else{

Ticket\_Management\* Temp = head;

while(Temp != NULL){

if(Temp->id == i){

if(Temp->name.compare(nm) == 0){

cout<<"Ticket with id : "<<i<<", & name : "<<nm<<", is Found in Ticket List, Status : "<<boolalpha<<Temp->Status<<"....\a"<<endl;

return true;

}

}

Temp = Temp->next\_tick;

}

}

cout<<"Ticket with id : "<<i<<", & name : "<<nm<<", not Found in Ticket List....\a"<<endl;

return false;

}

void Sort\_Ticket(){

if(head == NULL){

cout<<"Ticket List is Empty..."<<endl;

return;

}else{

int n = 0;

Ticket\_Management\* Temp = head;

while(Temp != NULL){

n++;

Temp = Temp->next\_tick;

}

cout<<"Enter Option for Sorting :\n"

"1-by creation time\n"

"2-by name \n"

"3-by Priority "<<endl;

int option;

cin >>option;

if(option == 1){

cout<<"-----SORT LIST BY CREATION TIME------"<<endl;

Ticket\_Management\* Temp = head;

Ticket\_Management\* Arr[n];

//Copy Elements to Array

for(int i=0; i<n; i++){

Arr[i] = Temp;

Temp = Temp->next\_tick;

}

//Use Bubble Sort for Sorting

for(int i=0; i<n; i++){

for(int j=0; j<n-1; j++){

if(Arr[j]->Creation\_time > Arr[j+1]->Creation\_time){

Ticket\_Management\* swap = Arr[j];

Arr[j] = Arr[j+1];

Arr[j+1] = swap;

}

}

}

//Array is Sort by Creation time now put it in list again

head = Arr[0];

Ticket\_Management\* tempswap = head;

for(int i=1; i<n; i++){

tempswap->next\_tick = Arr[i];

tempswap = tempswap->next\_tick;

}

}else if(option == 2){

cout<<"-----SORT LIST BY NAME------"<<endl;

Ticket\_Management\* Temp = head;

Ticket\_Management\* Arr[n];

//Copy Elements to Array

for(int i=0; i<n; i++){

Arr[i] = Temp;

Temp = Temp->next\_tick;

}

//Use Bubble Sort for Sorting

for(int i=0; i<n; i++){

for(int j=0; j<n-1; j++){

if(Arr[j]->name > Arr[j+1]->name){

Ticket\_Management\* swap = Arr[j];

Arr[j] = Arr[j+1];

Arr[j+1] = swap;

}

}

}

//Array is Sort by Creation time now put it in list again

head = Arr[0];

Ticket\_Management\* tempswap = head;

for(int i=1; i<n; i++){

tempswap->next\_tick = Arr[i];

tempswap = tempswap->next\_tick;

}

tempswap->next\_tick = NULL;

}else if(option == 3){

cout<<"-----SORT LIST BY PRIORITY------"<<endl;

Ticket\_Management\* Temp = head;

Ticket\_Management\* Arr[n];

//Copy Elements to Array

for(int i=0; i<n; i++){

Arr[i] = Temp;

Temp = Temp->next\_tick;

}

//Use Bubble Sort for Sorting

for(int i=0; i<n; i++){

for(int j=0; j<n-1; j++){

if(Arr[j]->priority > Arr[j+1]->priority){

Ticket\_Management\* swap = Arr[j];

Arr[j] = Arr[j+1];

Arr[j+1] = swap;

}

}

}

//Array is Sort by Creation time now put it in list again

head = Arr[0];

Ticket\_Management\* tempswap = head;

for(int i=1; i<n; i++){

tempswap->next\_tick = Arr[i];

tempswap = tempswap->next\_tick;

}

tempswap->next\_tick = NULL;

}else{

cout<<"Invalid Option!!!"<<endl;

Sort\_Ticket();

}

}

}

void Display\_List(){

if(head == NULL){

cout<<"Ticket List is Empty"<<endl;

return;

}else{

cout<<"\n------DISPLAY TICKET LIST----------\a"<<endl;

Ticket\_Management\* Temp = head;

while(Temp != NULL){

cout<<"ID : "<<Temp->id

<<", Name : "<<Temp->name

<<", Status : "<<boolalpha<<Temp->Status

<<", Priority : "<<Temp->priority

<<", Description : "<<Temp->service\_description

<<", Creation Time : "<<ctime(&Temp->Creation\_time)<<endl;

Temp = Temp->next\_tick;

}

cout << "--------------------------------" << endl;

}

}

void Display\_Open\_Tickets(){

if(head != NULL){

cout<<"Display Open Status Ticket---"<<endl;

Ticket\_Management\* Temp = head;

while(Temp != NULL){

if(Temp->Status){

cout<<"ID : "<<Temp->id

<<", Name : "<<Temp->name

<<", Status : "<<boolalpha<<Temp->Status

<<", Priority : "<<Temp->priority

<<", Description : "<<Temp->service\_description

<<", Creation Time : "<<ctime(&Temp->Creation\_time)<<endl;

Temp = Temp->next\_tick;

}

}

}else{

cout<<"List is NULL"<<endl;

}

}

void Display\_Resolution\_Tickets(){

if(head != NULL){

cout<<"Display Resolve Ticket---"<<endl;

Ticket\_Management\* Temp = head;

while(Temp != NULL){

if(!(Temp->Status)){

cout<<"ID : "<<Temp->id

<<", Name : "<<Temp->name

<<", Status : "<<boolalpha<<Temp->Status

<<", Priority : "<<Temp->priority

<<", Description : "<<Temp->service\_description

<<", Closed Time : "<<ctime(&Temp->close\_time)<<endl;

Temp = Temp->next\_tick;

}

}

}else{

cout<<"List is NULL"<<endl;

}

}

};

class Agent\_Management{

public :

int Agent\_id;

string Agent\_name;

Ticket\_Management\* Assigned\_ticket[5];

int counter;

bool Availability;

Agent\_Management\* next\_Agent;

Agent\_Management(int i, string n){

Agent\_id = i;

Agent\_name = n;

Availability = true;

counter = 0;

}

void AssignTicket(Ticket\_Management\* ticket){

if(counter < 5){

Assigned\_ticket[counter] = ticket;

counter++;

}

Availability = false;

}

int Resolve() {

// Check if any tickets are assigned

if (counter == 0 || Assigned\_ticket == nullptr) {

cout << "No tickets are assigned to this agent. Please assign tickets first.\n";

return -1; // Indicating no tickets to resolve

}

// Sort tickets by priority in descending order

for (int i = 0; i < counter - 1; i++) {

for (int j = 0; j < counter - i - 1; j++) {

if (Assigned\_ticket[j]->priority < Assigned\_ticket[j + 1]->priority) {

// Swap if priority of current is less than next

Ticket\_Management\* temp = Assigned\_ticket[j];

Assigned\_ticket[j] = Assigned\_ticket[j + 1];

Assigned\_ticket[j + 1] = temp;

}

}

}

// Resolve the highest-priority ticket

int resolvedTicketId = Assigned\_ticket[0]->id;

cout << "\nResolution of Ticket ID: " << resolvedTicketId

<< " is done successfully by Agent: " << Agent\_name << endl;

// Shift tickets in the array to remove the resolved ticket

for (int i = 1; i < counter; i++) {

Assigned\_ticket[i - 1] = Assigned\_ticket[i];

}

counter--; // Decrease the counter after removing the resolved ticket

return resolvedTicketId;

}

void Display\_Assigned\_Tickets(){

if(counter == 0){

cout<<"No list Assigned yet";

return;

}else{

for(int i=0; i<counter; i++){

cout<<"ID : "<<Assigned\_ticket[i]->id

<<", Name : "<<Assigned\_ticket[i]->name

<<", Status : "<<boolalpha<<Assigned\_ticket[i]->Status

<<", Priority : "<<Assigned\_ticket[i]->priority

<<", Description : "<<Assigned\_ticket[i]->service\_description

<<", Closed Time : "<<ctime(&Assigned\_ticket[i]->close\_time)<<endl;

}

}

}

};

class Agents{

public :

Agent\_Management\* head;

Agents(){

head = NULL;

}

public :

void add\_Agent(int i, string n){

Agent\_Management\* Agent = new Agent\_Management(i,n);

if(head == NULL){

head = Agent;

}else{

Agent\_Management\* Temp = head;

while(Temp->next\_Agent != NULL){

Temp = Temp->next\_Agent;

}

Temp->next\_Agent = Agent;

Agent->next\_Agent = NULL;

cout<<"Agent Inserted"<<endl;

}

}

void Display\_Agents(){

if(head == NULL){

cout<<"No agent"<<endl;

return;

}else{

cout<<"DISPLAY ALL Agents ...."<<endl;

Agent\_Management\* temp = head;

while(temp != NULL){

cout<<"Id : "<<temp->Agent\_id<<", Name : "<<temp->Agent\_name<<endl;

temp = temp->next\_Agent;

}

}

}

void Display\_Assign\_Tickets(){

if(head == NULL){

cout<<"No Agent"<<endl;

return;

}else{

Agent\_Management\* temp = head;

while(temp != NULL){

temp->Display\_Assigned\_Tickets();

temp = temp->next\_Agent;

}

}

}

};

int main(){

cout<<"-----23k-3001 | ABDULLAH------"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"--------------ONE STOP MANAGEMENT SYSTEM--------------"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

Ticket\_list\* list = new Ticket\_list();

Agents\* agent = new Agents();

int flag1 = 1;

int flag2 = 1;

int opt1;

int opt2;

while(flag1){

cout<<"------MENU-------"<<endl;

cout<<"Choose Option : "<<endl;

cout<<"1.Admin"<<endl;

cout<<"2.Student"<<endl;

cout<<"3.exit"<<endl;

cin>>opt1;

flag2 = 1;

if(opt1 == 1){

while(flag2){

cout<<"Choose Option FOR ADMIN : "<<endl;

cout<<"1.Add Agent"<<endl;

cout<<"2.Display List of Agents"<<endl;

cout<<"3.Display Assigned Ticket"<<endl;

cout<<"4.Sort"<<endl;

cout<<"5.Display all Tickets"<<endl;

cout<<"6.Resolve a Ticket Problem"<<endl;

cout<<"7.Search"<<endl;

cout<<"8.Display List of all Open Tickets"<<endl;

cout<<"9.Display Resolution Tickets"<<endl;

cout<<"10.Menu"<<endl;

cin>>opt2;

if(opt2 == 1){

cout<<"Enter Name : ";

string name;

cin >>name;

cout<<"Enter Id : ";

int id;

cin >> id;

agent->add\_Agent(id,name);

}else if(opt2 == 2){

agent->Display\_Agents();

}else if(opt2 == 3){

agent->Display\_Assign\_Tickets();

}else if(opt2 == 4){

list->Sort\_Ticket();

}else if(opt2 == 5){

list->Display\_List();

}else if(opt2 == 6){

int id = agent->head->Resolve();

list->Remove\_Ticket(id); //change this ticket status

}else if(opt2 == 7){

cout<<"Enter Name & Id to Search : "<<endl;

string nm;

int id;

list->Search(nm, id);

}else if(opt2 == 8){

list->Display\_Open\_Tickets();

}else if(opt2 == 9){

list->Display\_Resolution\_Tickets();

}else if(opt2 == 10){

flag2 = 0;

}else{

cout<<"! Invalid Input....Choose option Correctly.."<<endl;

}

}

}else if(opt1 == 2){

while(flag2){

cout<<"Choose Option for STUDENT : "<<endl;

cout<<"1.Add Ticket"<<endl;

cout<<"2.Sort"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Menu"<<endl;

cin>>opt2;

if(opt2 == 1){

cout<<"Enter Name : ";

string name;

cin >>name;

cout<<"Enter Id : ";

int id;

cin >> id;

cout<<"Enter Description : ";

string des;

cin >>des;

cout<<"Enter Priority : ";

int priority;

cin >> priority;

Ticket\_Management\* tick = list->Add\_Ticket(name,des,priority,id);

agent->head->AssignTicket(tick);

}else if(opt2 == 2){

list->Sort\_Ticket();

}else if(opt2 == 3){

list->Display\_List();

}else if(opt2 == 4){

flag2 = 0;

}else{

cout<<"! Invalid Input....Choose option Correctly.."<<endl;

}

}

}else if(opt1 == 3){

flag1 = 0;

}else{

cout<<"Invalid Option !!"<<endl;

}

}

return 0;

}

OUTPUT :

  
  








