**Description :**

**U\_BUS**

**IDEA:**

U-bus is a console base application, which is similar to an android application which is used for storing bus’s information; users can register themselves as a driver or a customer. Drivers have buses which have specific bus stops and customers can be assigned to a bus on the basis of nearest bus stops. The main idea is to develop a user friendly registration process and bus management system.

**PROBLEM STATEMENT**:

We intend to handle and organize large sets of data such as the user’s information in a more innovative way.

**MOTIVATION:**

We intend to provide a better solution of intercity transportation at your nearest bus stop. The user interface of the registration process is made easy such that a non-technical person is able to use it as well.

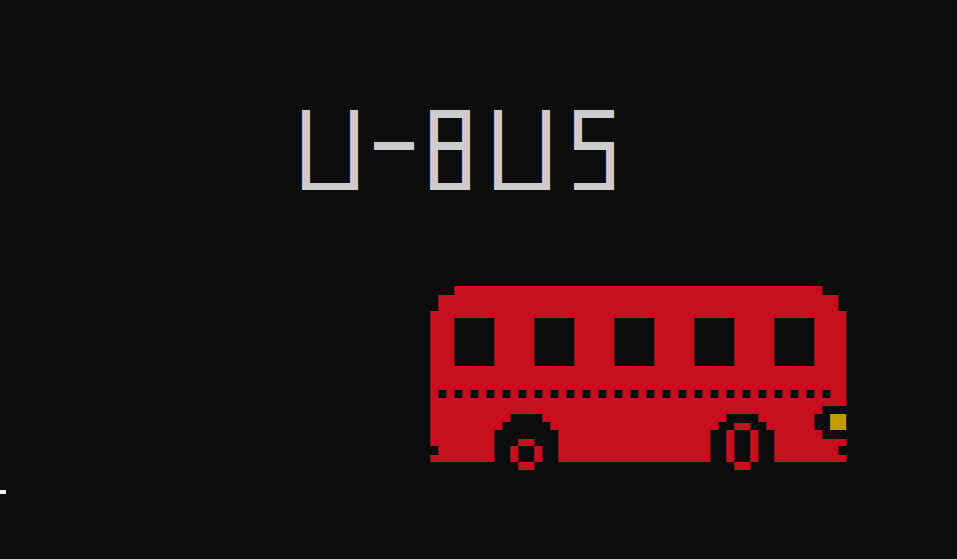
**CONCEPTS:**

Following concepts were used for this project, as data entered is in form of table of linked list and organizing it on basis of string was convenient in Binary Search Tree, as searching of bus which is already installed and account existence during log in is convenient:

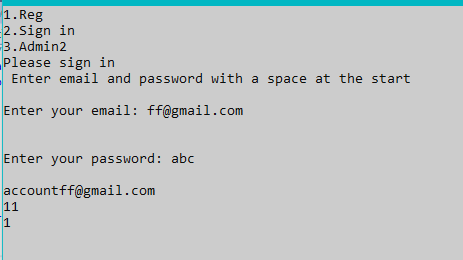
* **Linked Lists:** Linked lists were used for the insertion of the data. Both singly and doubly linked lists were used.
* **Binary Search Tree:** Binary search trees were used to search, retrieve and organize large set of data for efficiency.
* **Tables:** Tables were used to hold the data inserted in the linked list.
* **Minimum Node:** Minimum value can be searched.

**Screenshots of the Output:**

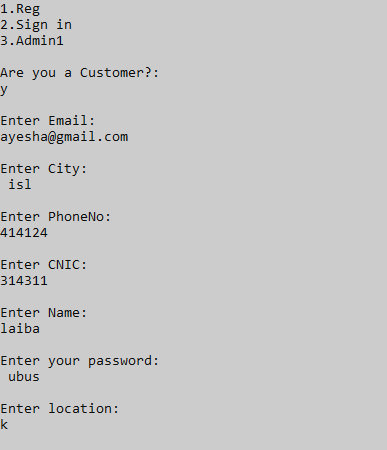
**Starting Screen**

****

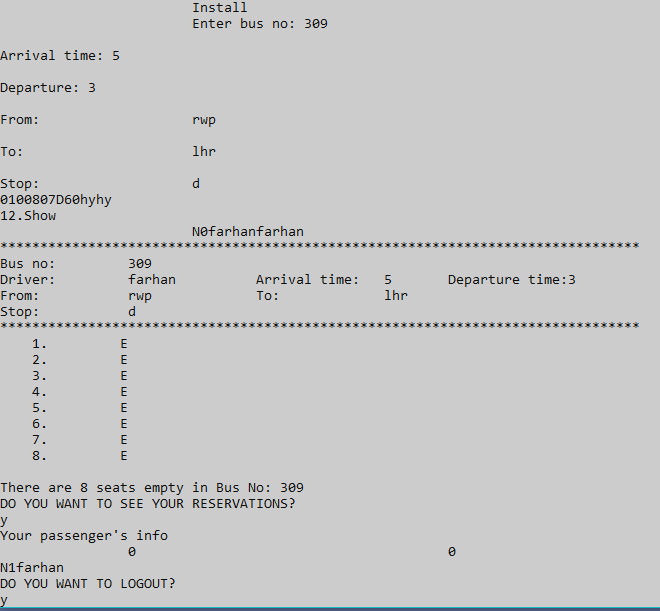
**Login process**

****

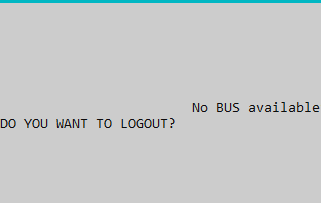
**Registeration Process**

****

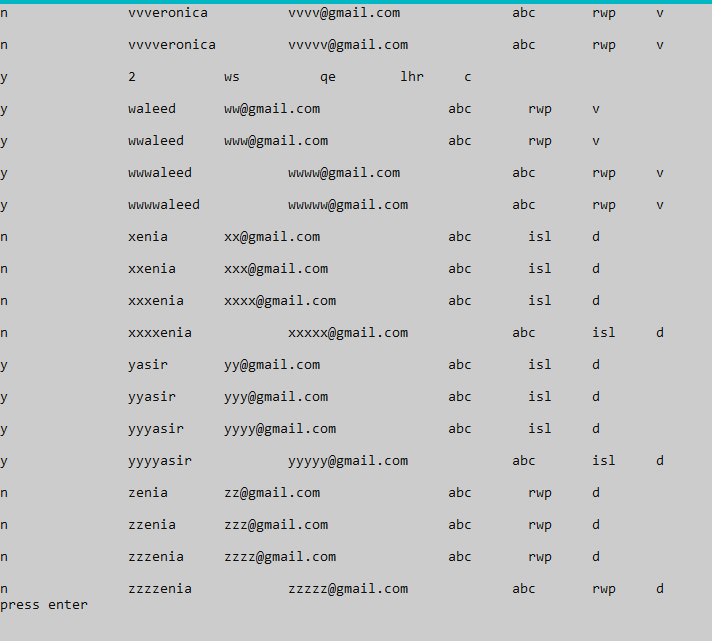
**Installation of bus**

****

**No bus retrived for customer as no bus has matching city and location**

****

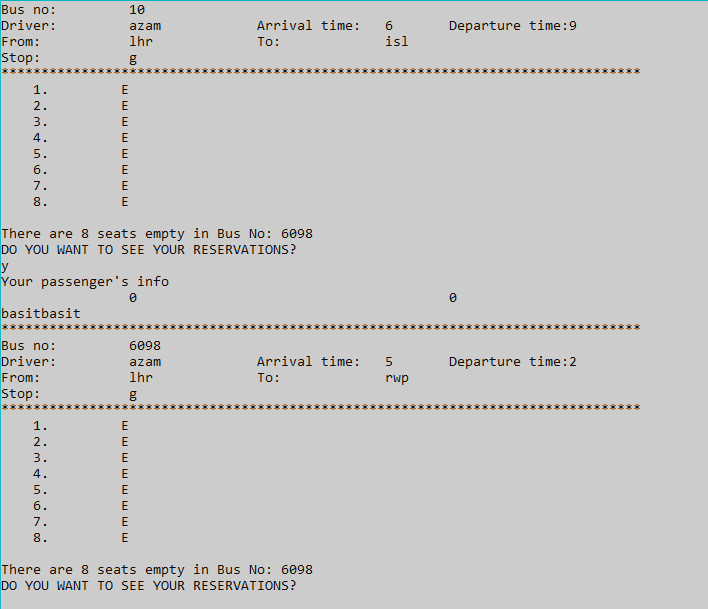
**Retrival of all account information in admin view**

****

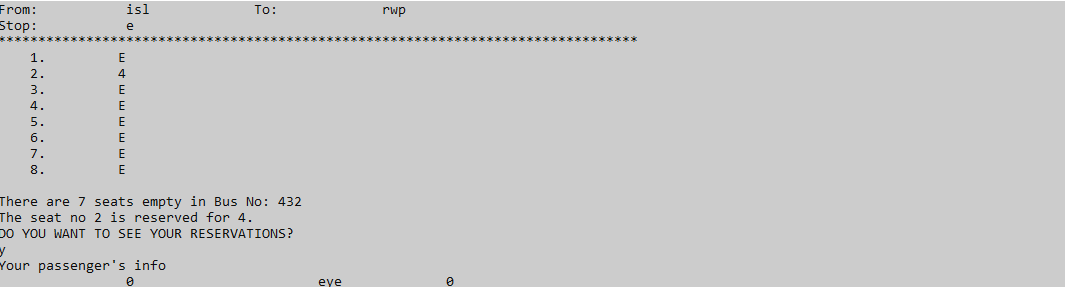
**Seat Reservation**



**Retrieval of both buses, one installed at run time and the one installed beforehand and retrieved from file**



**Information of driver after seat reservation**



**Source Code:**

#include <iostream>

#include <stdio.h>

#include <string>

#include <conio.h>

#include <cstdio>

#include <cstdlib>

#include <sstream>

#include <fstream> // For files

#include <time.h>

#include <windows.h>

#include<stdlib.h>

using namespace std;

HANDLE color;

void delay(unsigned int mseconds) // FUNCTION USED TO DELAY THE OUTPUT BY MILLISECONDS

{

clock\_t goal = mseconds + clock();

while (goal > clock());

}

void gotoxy(int x, int y) // FUNCTION USED TO SET THE COORDINATES

{

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

void graphics(int speed) // jump = position of dinosuar, speed = movement speed of dinosaur

{

int x = 29;

HANDLE colour\_change;

colour\_change = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(colour\_change, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

gotoxy(38, 7);

cout << "Û Û ÛßßßÛ Û Û Ûßßßß " << endl;

gotoxy(38, 8);

cout << "Û Û Û Û Û Û Û " << endl << " ";

gotoxy(38, 9);

cout << "Û Û ßßßßß ÛßßßÛ Û Û ßßßßÛ " << endl << " ";

gotoxy(38, 10);

cout << "Û Û Û Û Û Û Û " << endl;

gotoxy(38, 11);

cout << "ÛÜÜÜÜÜÛ ÛÜÜÜÛ ÛÜÜÜÜÜÛ ÜÜÜÜÛ " << endl << " ";

colour\_change = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(colour\_change, FOREGROUND\_RED); //setting the color of bus as red

static int a = 1; // using static int so it stays till the end of the program

while (1) {

if (x == 113)

break;

// the symbols Ü, Û, ß are used to draw the bus in print statements

gotoxy(x, 18);

cout << " ÜÜÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÜÜ " << endl;

gotoxy(x, 19);

cout << " ÜÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÜ " << endl;

gotoxy(x, 20);

cout << " ÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛ " << endl;

gotoxy(x, 21);

cout << " ÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛ " << endl;

gotoxy(x, 22);

cout << " ÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛÛ ÛÛÛÛ " << endl;

gotoxy(x, 23);

cout << " ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ " << endl;

gotoxy(x, 24);

cout << " ÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛßÛÛ " << endl;

gotoxy(x, 25);

cout << " ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛßßß " << endl;

if (a == 1)

{

gotoxy(x, 26);

cout << " ÛÛÛÛÛÛÛÛÛß ÜÜ ßÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛß ßÛÛÛÛÛÛ ";

color = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(color, FOREGROUND\_GREEN | FOREGROUND\_RED);

cout << "ÛÛ \n";

color = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(color, FOREGROUND\_RED);

gotoxy(x, 27);

cout << " ÛÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ ÜÜ ÛÛÛÛÛÛÜÜÜ " << endl;

gotoxy(x, 28);

cout << " ÜÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÛÛÜ " << endl;

gotoxy(x, 29);

cout << " ßß ßß " << endl;

a = 2;

}

else if (a == 2)

{

gotoxy(x, 26);

cout << " ÛÛÛÛÛÛÛÛÛß ßÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛß ÜÜ ßÛÛÛÛÛÛ ";

color = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(color, FOREGROUND\_GREEN | FOREGROUND\_RED);

cout << "ÛÛ \n";

color = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(color, FOREGROUND\_RED);

gotoxy(x, 27);

cout << " ÛÛÛÛÛÛÛÛ ÜÜ ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÜÜÜ " << endl;

gotoxy(x, 28);

cout << " ÜÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ Û Û ÛÛÛÛÛÛÛÛÜ " << endl;

gotoxy(x, 29);

cout << " ßß ßß " << endl;

a = 1;

}

delay(speed);

x++;

}

SetConsoleTextAttribute(colour\_change, BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED); //Changing the colour background and foreground

}

char c;

fstream accf;

fstream uf;

void closeFile()

{

accf.close();

uf.close();

}

char nop[5];

void openFile()// function that opens the file

{

accf.open("account.txt", ios::in | ios::out | ios::app);

uf.open("user.txt", ios::in | ios::out | ios::app);

}

//function that contain account register information

class Name {

public:

char choice, loc;

string name, email, city;

int phone, cnic;

Name\* next;

Name\* prev;

string pas;

//double linked lists

Name(char s, string n, string e, string pa, char loca, int p, int c, string ce, Name\* pre, Name\* nw) {

choice = s;

name = n;

city = ce;

prev = pre;

pas = pa;

loc = loca;

email = e;

phone = p;

cnic = c;

next = nw;

}//single linked lists

Name(char s, string n, string e, string pa, char loca, int p, int c, string ce, Name\* nw) {

choice = s;

name = n;

city = ce;

pas = pa;

loc = loca;

email = e;

phone = p;

cnic = c;

next = nw;

}

};

//to enter data for registration

//data security purposes

class AccessNode {

public:

Name\* head;

Name\* t;

char at;

Name\* heada;

void Enterna();

void signup();

void readFromFileUser();

void admin();

Name\* accountexists(string g, string p);

int count = 0;

AccessNode() {

}

};

int counts;

string names;

//functions for data insertion

char choice() {

char s;

cout << "\nAre you a Customer?:\n";

cin >> s;

return s;

}

AccessNode\* a = new AccessNode();

int pp;

int people = 0;

string nam() {

string s;

cout << "\nEnter Name:\n";

cin.get();

getline(cin, s);

return s;

}

string pass() {

string s;

cout << "\nEnter your password:\n";

cin.get();

getline(cin, s);

return s;

}

char loc() {

char s;

cout << "\nEnter location:\n";

cin >> s;

return s;

}

int allow = 0;

string eil() {

cout << "\nEnter Email:\n";

string e;

cin.get();

getline(cin, e);

return e;

}

string city() {

cout << "\nEnter City:\n";

string e;

cin.get();

getline(cin, e);

return e;

}

int phoe() {

cout << "\nEnter PhoneNo:\n";

int sp;

cin >> sp;

return sp;

}

int cnc() {

cout << "\nEnter CNIC:\n";

int c;

cin >> c;

return c;

}

string l[10];

string drivers;

char lo; int polo = 0;

//function for data insertion of accounts

void AccessNode::Enterna() {

t = head;

at = choice();

string email = eil();

string c = city();

int phone = pp = phoe();

int cnic = cnc();

names = nam();

string p = pass();

lo = loc();

//insertion of data in BST

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

if (head == NULL) {

head = t = new Name(at, names, email, p, lo, phone, cnic, c, NULL, NULL);

}

else {

t->next = new Name(at, names, email, p, lo, phone, cnic, c, head, NULL);

t = t->next;

count++;

break;

}openFile();

//inserting data in file

if (accf.is\_open())

{

accf.seekg(accf.tellp());

accf << "\n" << at << " " << names << " " << email << " " << c << " " << p << " " << lo << " " << phone << " " << cnic;

accf.seekg(accf.tellp());

}

else

cout << "---Sorry, Could not open Accounts File---" << endl << endl;

closeFile();

if (at == 'y')

people++;

}

for (int i = count; i == count; ++i) {

cout << count << at;

l[i] = lo;

}

counts = count;

}

bool t = 1;

class BST // class for Binary Search Tree

{//data searching on basis of string is convenient

public:

bool isEmpty(Name\* root) const // if tree is empty

{

return root == NULL;

}

void inorder(Name\* n);

Name\* searchNode(Name\* root, string a);

Name\* minNode(Name\* n);

Name\* insert(Name\* root, string name, string email, char lo, char op, int phone, int cnic, string p, string c);

};

//bus class

class swvl

{

public:

char busn[5], arrival[5], depart[5], from[10], to[10], stop[10], seat[8][10];

public:

string driver, driv[10];

swvl\* h;

swvl\* next;

swvl\* we;

swvl\* node; swvl\* r; swvl\* heada;

//data insertion of bus

swvl(char busn1[5], char arrival1[5], char depart1[5], char from1[10], char to1[10], char stop1[10], string driver1, swvl\* n) {

node = n;

for (int i = 0; i < 5; i++) {

busn[i] = busn1[i];

arrival[i] = arrival1[i];

depart[i] = depart1[i];

}

for (int i = 0; i < 10; i++) {

from[i] = from1[i];

to[i] = to1[i];

stop[i] = stop1[i];

}

driver = driver1;

}

//single linked list for searching region

swvl(char from1[10], char stop1[10], string d, swvl\* hi) {

for (int i = 0; i < 10; i++) {

from[i] = from1[i];

stop[i] = stop1[i];

}

driver = d;

next = hi;

}

swvl() {}

void install();

void allotment();

void empty();

void show();

void avail();

bool search(string a);

void position(int i);

}

//array obj

bus[10];

string b, aaa, di, fr, too, stopp;

//object for BST searching

//security purpose of data

swvl\* tata = new swvl();

//BST class for bus

class BSTB // class for Binary Search Tree

{

public:

bool isEmpty(Name\* root) const // if tree is empty

{

return root == NULL;

}

void inorder(swvl\* n);

swvl\* searchNodeE(swvl\* root, string a);

swvl\* minNode(swvl\* n);

swvl\* insert(swvl\* root, char busn1[5], char arrival1[5], char depart1[5], char from1[10], char to1[10], char stop1[10], string driver1);

};

//searching BST on basis of driver name

swvl\* BSTB::searchNodeE(swvl\* root, string a) {

if (root == 0) return 0;

else

if (a == (root->driver))

{

cout << root->driver;

b = root->busn;

aaa = root->arrival;

di = root->depart;

cout << b << aaa;

fr = root->from;

too = root->to;

stopp = root->stop;

return root;

}

else if (a < root->driver)

{

return searchNodeE(root->node, a);

}

else if (a > root->driver)

{

return searchNodeE(root->r, a);

}

}bool j;

//searching of driver through the use of name from account class/file

//to check whether a bus is installed or not before run time

bool swvl::search(string a) {

BSTB boj;

swvl\* temm = boj.searchNodeE(tata, a);

if (temm == 0) {

j = false;

cout << j;

return false;

}

else { j = true; return true; }

}

//searching account on basis of email

Name\* BST::searchNode(Name\* root, string a)

{

if (a == (root->email))

{

return root;

}

else if (a < root->email)

{

return searchNode(root->next, a);

}

else if (a > root->email)

{

return searchNode(root->prev, a);

}

else if (root == 0) return 0;

}

//inserting in BST on basis of email

Name\* BST::insert(Name\* root, string name, string email, char lo, char op, int phone, int cnic, string p, string c)

{

if (root == 0)

{

root = new Name(op, name, email, p, lo, phone, cnic, c, NULL);

}

else {

if (email < root->email)

{

root->next = insert(root->next, name, email, lo, op, phone, cnic, p, c);//left node

}

else if (email > root->email) {

root->prev = insert(root->prev, name, email, lo, op, phone, cnic, p, c);//right

}

}

return root;

}

bool truee;

//checking whether account exist

Name\* AccessNode::accountexists(string account, string passcode) // Access data if account already exists

{

BST boj;

Name\* temp = boj.searchNode(heada, account);

cin.get();

if (temp)

{

//checking password and email

if ((temp->pas == passcode) && (temp->email == account)) {

truee = true;

cout << truee;

return temp;

}

else {

truee = false;

return 0;

}

}

}

//min heapify algorithm

Name\* BST::minNode(Name\* n)

{

Name\* current = n;

if (!current->next == NULL)

return minNode(current->next);

else

return current;

}

swvl\* BSTB::insert(swvl\* root, char busn1[5], char arrival1[5], char depart1[5], char from1[10], char to1[10], char stop1[10], string drive)

{

if (root == 00000)

{

root = new swvl(busn1, arrival1, depart1, from1, to1, stop1, drive, NULL);

}

else {

if (drive == root->driver)return root;

if (drive < root->driver)

{

root->node = insert(root->node, busn1, arrival1, depart1, from1, to1, stop1, drive);//left node

}

else if (drive > root->driver) {

root->r = insert(root->r, busn1, arrival1, depart1, from1, to1, stop1, drive);//right

}

}

return root;

}

string meranam;

//inorder traversal for admin

void BST::inorder(Name\* n)

{

if (n == 0)

{

return;

}

inorder(n->next);

cout << "\n" << n->choice << "\t\t" << n->name << "\t " << n->email << "\t\t" << n->pas << "\t " << n->city << "\t " << n->loc << endl;

inorder(n->prev);

}

static int p = 0;

string cii, ss;

swvl\* wish = new swvl();

//accessbus class to read data from user/driver/bus file

class accessbus {

public:

swvl\* h; swvl\* we;

swvl\* heada;

accessbus() {}

void read();

};

//admin class

void AccessNode::admin() {

BST\* b = NULL;

b->inorder(heada);

cin.get();

}

//signin

void AccessNode::signup() { // sign in process

string e;

string pass;

t = head;

allow = 0;

c = ' ';

cin.get();

cout << "Please sign in\n Enter email and password with a space at the start\n\n";

cout << "Enter your email:";

cin.get();

getline(cin, e);

cout << "\n\nEnter your password:";

cin.get();

getline(cin, pass);

Name\* n = NULL;

BST boj;

int s = 0;

cin.get();

//checcking whether account exists

t = accountexists(e, pass);

if (truee)

{

drivers = names = t->name;

c = t->choice;

cii = t->city;

ss = t->loc;

//if account is of driver then meranam is set

//mernam is global variable

//using meranam it checked whether a bus is installed or not

if (c == 'n') {

meranam = t->name;

}

cin.get();

allow = 1;

}

}

string name[32];

int phone[32];

//graphical purpose

void vline(char ch)

{

for (int i = 80; i > 0; i--)

cout << ch;

}

int i = 0;

int alina = 0;

char yes = 'n';

//installation o bus

void swvl::install()

{//data enetered

driver = names;

cout << "Enter bus no: ";

cin >> bus[p].busn;

string c = names;

if (yes == 'y')

{

cout << "\nEnter Driver's name: ";

cin.get();

getline(cin, driver);

driv[p] = names;

}

else {

if (yes == ('n'))

{

driver = names;

driv[p] = names;

}

}

cout << "\nArrival time: ";

cin >> bus[p].arrival;

cout << "\nDeparture: ";

cin >> bus[p].depart;

cout << "\nFrom: \t\t\t";

cin >> bus[p].from;

cout << "\nTo: \t\t\t";

cin >> bus[p].to;

cout << "\nStop: \t\t\t";

cin >> bus[p].stop;

cout << p;

BSTB boj;

openFile();

//data of bus written on user file

cout << uf.is\_open();

if (uf.is\_open())

{

uf.seekg(uf.tellp());

uf << "\n" << bus[p].busn << " " << bus[p].arrival << " " << bus[p].depart << " " << bus[p].from << " " << bus[p].to << " " << bus[p].stop << " " << c;

uf.seekg(uf.tellp());

}

else

cout << "---Sorry, Could not open Accounts File---" << endl << endl;

//data inserted in BSTB

if (heada == 00000)

{

heada = boj.insert(0, bus[p].busn, bus[p].arrival, bus[p].depart, bus[p].from, bus[p].to, bus[p].stop, c);

tata = heada;

}

else

{

boj.insert(heada, bus[p].busn, bus[p].arrival, bus[p].depart, bus[p].from, bus[p].to, bus[p].stop, c);

}

//data entered in single linked list

//this list is used for comparing region of bus and customer

we = wish;

cout << wish->next;

while (t == 0) {

if (wish->next == NULL)

{

wish->next = new swvl(bus[p].from, bus[p].stop, c, NULL);

wish = wish->next;

break;

}

else

wish = wish->next;

}

closeFile();

bus[p].empty();

p++;

i = p;

}

//alloting bus

void swvl::allotment()

{

int seat;

top:

int n;

for (n = 0; n <= p; n++)

{

if (strcmp(bus[n].busn, nop) == 0)

break;

}

while (n <= p)

{//reserving seat

if (alina != 1) {

cout << "\nSeat Number: ";

cin >> seat;

if (seat > 8)

{

cout << "\nThere are only 8 seats available in this bus.";

}

else

{

if (strcmp(bus[n].seat[seat / 4], "E") == 0)

{

//reserving the seat

cin >> bus[n].seat[seat / 4];

cout << "your seat is reserved\n";

vline('\*');

//information of bus driver is shown to passenger

cout << "Bus No:" << bus[n].busn << endl;

cout << "Bus Driver:" << bus[n].driver << endl;

cout << "Bus Stop:" << bus[n].stop << endl;

//passenger information is stored for driver access

for (int i = people; i <= people; ++i) {

name[i] = names;

phone[i] = pp;

} vline('\*');

break;

}

else

{//if seat is reserved

cout << "The seat no. is already reserved.\n";

cout << "Do you want to cancel your reservation";

char c;

cin >> c;

if (c == 'y') {

strcpy\_s(bus[n].seat[seat / 4], "E"); name[i] = " ";

phone[i] = 0;

}

}

}

}

}

if (n > p)

{

cout << "Enter correct bus no.\n";

goto top;

}

}

//passenger information

//for driver access

void passenger() {

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

cout << "Your passenger's info" << endl;

cout << "\t\t" << i << "\t\t\t" << name[i] << "\t\t" << phone[i] << endl;

}

}

//if seat is not reserved

//seat is empty/E

void swvl::empty()

{

for (int i = 0; i < 8; i++)

{

strcpy\_s(bus[p].seat[i], "E");

}

}

//showing information of bus for driver

void swvl::show()

{

int t = i;

int n;

char number[5];

for (n = 0; n <= t; n++)

{

for (int i = 0; i < 9; ++i) {

for (int i = 0; i < 10; ++i)

{

if (bus[n].stop != l[i])

break;

}

break;

}

break;

}

while (n <= t)

{

//bus information is shown if bus is installed before run time

if (j)

{

cout << "\n";

vline('\*');

cout << "\nBus no: \t" << b

<< "\nDriver: \t" << bus[n].driver << "\t\tArrival time: \t"

<< aaa << "\tDeparture time:" << di

<< "\nFrom: \t\t" << fr << "\t\tTo: \t\t" <<

too << "\n" << "Stop: \t\t" << stopp << "\n"; int s;

stringstream i(b);

i >> s;

s = n % 10;

t--;

vline('\*');

//checking whether seat is reverved or not

bus[0].position(s); int a = 1;

for (int j = 0; j < 8; j++)

{

a++;

if (strcmp(bus[s].seat[j], "E") != 0)

cout << "\nThe seat no " << (a - 1) << " is reserved for " << bus[s].seat[j] << ".";

}

cout << "\nDO YOU WANT TO SEE YOUR RESERVATIONS?\n";

char laiba;

cin >> laiba;

//passenger info is retrieved

if (laiba == 'y')

{

passenger();

}

}

cout << drivers << driv[n];

if (drivers == driv[n])

{

cout << "\n";

vline('\*');

//information of bus that is installed during run time

cout << "\nBus no: \t" << bus[n].busn

<< "\nDriver: \t" << bus[n].driver << "\t\tArrival time: \t"

<< bus[n].arrival << "\tDeparture time:" << bus[n].depart

<< "\nFrom: \t\t" << bus[n].from << "\t\tTo: \t\t" <<

bus[n].to << "\n" << "Stop: \t\t" << bus[n].stop << "\n";

vline('\*');

//seat reservation

bus[0].position(n);

int a = 1;

for (int j = 0; j < 8; j++)

{

a++;

if (strcmp(bus[n].seat[j], "E") != 0)

cout << "\nThe seat no " << (a - 1) << " is reserved for " << bus[n].seat[j] << ".";

}

cout << "\nDO YOU WANT TO SEE YOUR RESERVATIONS?\n";

char laiba;

cin >> laiba;

if (laiba == 'y')

{//passenger info

passenger();

}

}

p = i;

n++;

}

if (n > t)

{

}

}

//seat reservation is checked

void swvl::position(int l)

{

int s = 0; p = 0;

for (int i = 0; i < 8; i++)

{

cout << "\n";

s++;

//if seat is not reserve

if (strcmp(bus[l].seat[i], "E") == 0)

{

//show reservation in alignment

cout.width(5);

cout.fill(' ');

cout << s << ".";

cout.width(10);

cout.fill(' ');

cout << bus[l].seat[i];

p++;

}

else

{ //if seat is reserve

cout.width(5);

cout.fill(' ');

cout << s << ".";

cout.width(10);

cout.fill(' ');

cout << bus[l].seat[i];

}

}

cout << "\n\nThere are " << p << " seats empty in Bus No: " << bus[l].busn;

}

//available buses

void swvl::avail()

{

bool chale = false;

string namkia;

string region;

we = wish;

swvl\* yay = new swvl();

BSTB boj;

for (int n = 0; n <= p; n++)

{

int c = -1;

for (int j = 0; j <= counts; j++) {

//finding buses according to location/sector and city

while (t == 0) {

region = we->from;

if ((ss == we->stop) && (cii == region)) {

namkia = we->driver;

yay = boj.searchNodeE(tata, namkia);

if(t==0)

{strcpy\_s(nop, yay->busn);

//available buses data installed already

cout << "\nBus no: \t" << b << "\nDriver: \t" << namkia

<< "\t\tArrival time: \t" << aaa << "\tDeparture Time: \t"

<< di << "\nFrom: \t\t" << fr << "\t\tTo: \t\t\t"

<< too << "\n" << "\nStop: \t\t" << stopp << endl;

vline('\*');

}

chale = true;

break;

}

we = we->next;

}

if (chale)

{//busno string is coverted in int for reservation of seat on basis of bus

stringstream i(b);

i >> n;

n = n % 10;

alina = n;

strcpy\_s(nop, yay->busn);

//available buses data

cout << "\nBus no: \t" << b << "\nDriver: \t" << namkia

<< "\t\tArrival time: \t" << aaa << "\tDeparture Time: \t"

<< di << "\nFrom: \t\t" << fr << "\t\tTo: \t\t\t"

<< too << "\n" << "\nStop: \t\t" << stopp << endl;

vline('\*');

cout << endl;

vline('\_');

break;

}

else {//if no bus available

cout << "\n\t\t\t";

cout << "No BUS available";

}break;

}break;

}

}

void AccessNode::readFromFileUser()

{

BST boj;

int a = 0;

char choice, loc;

string name, email, city, pass;

int phone, cnic;

if (accf.is\_open())

{

while (1)// reaches the end of the file

{

if (accf.eof())

{

break;

}

//reading data from file

accf >> choice >> name >> email >> city >> pass >> loc >> phone >> cnic;

//insertion in BST

if (heada == 0)

{

heada = boj.insert(0, name, email, loc, choice, phone, cnic, pass, city);

}

else

{

boj.insert(heada, name, email, loc, choice, phone, cnic, pass, city);

}

}

}

else

cout << "---Sorry, Could not open File---" << endl << endl;

}

void accessbus::read()

{

BSTB boj;

char busn[5], arrival[5], depart[5], from[10], to[10], stop[10];

string d;

if (uf.is\_open())

{

while (1)// reaches the end of the file

{

if (uf.eof())

{

break;

}

//reading data from user

uf >> busn >> arrival >> depart >> from >> to >> stop >> d;

//inserting data in BSTB

if (heada == 00000)

{

heada = boj.insert(0, busn, arrival, depart, from, to, stop, d);

tata = heada;

}

else

{

boj.insert(heada, busn, arrival, depart, from, to, stop, d);

}

//single linked list for customer and driver's region comparison

if (h == NULL) {

h = we = new swvl(from, stop, d, NULL);

wish = h;

}

else {

we->next = new swvl(from, stop, d, NULL);

we = we->next;

}

//cin.ignore();

}

}

else

cout << "---Sorry, Could not open File---" << endl << endl;

}

//MAIN

int main() {

//graphics

system("mode con: lines=38 cols=120");

gotoxy(34, 20);

graphics(100);

system("CLS");

accessbus\* gb = new accessbus();

openFile();

swvl\* b = new swvl();

int w = 0;

//user file

gb->read();

char n = 'y';

while (n == 'y') {

n = 'n';

if (t == 1) {

//account file

a->readFromFileUser();

closeFile();

cout << "1.Reg\n2.Sign in\n3.Admin";

int tre;

cin >> tre;

if (tre == 1)

{//registring an account

a->Enterna(); system("cls"); a->signup();

}if (tre == 2)

{//sign in

a->signup();

if (b->search(meranam))t = 0;//value of t determine if bus is installed or not

else t == 1;

}

if (tre == 3) {

//admin

a->admin();

cout << "press enter";

cin.get();

}

}

system("cls");

if (allow == 1)

{

cout << "\n\n\n\n\n";

if (c == 'n')

{

w = 0;

if (t == 1)

{

cout << "\t\t\tInstall\n\t\t\t";

}

else {

cout << "do you want to install a bus?";

cin >> yes;

}

}

else

{

if (c == 'y') {

w = 1;

}

}

switch (w)

{

if (a->at == 'n')

{

case 0://case 0 for bus

{if (t == 1 || yes == 'y')

{

bus[0].install();//bus installation

}

cout << "2.Show\n\t\t\t";

bus[0].show();//bus info

break; }

}

else {

case 1: {//case 1 for customer

bus[0].avail();

if (alina != 1)

{

bus[p].allotment();//reservation

}

else

break;

}

}

}cout << "\nDO YOU WANT TO LOGOUT?\n";

cin >> n;

system("cls");

cout << "Do you want to log in?(0 for yes and 1 for no)\n";

cin >> t;

if (t == 0) {

a->signup();

n = 'y';

}

else {

n = 'y';

}

system("cls");

}

else { cout << "Invalid log in\n"; }

}

cout << "GOOD BYE";

}