****

**East West University**

**CSE207 (Data Structure), Section: 2**

**Summer 2017**

**Project Report**

***Project Title:* Train Ticket Reservation System**

***Submitted To:***

**Shamim H Ripon**

Associate Professor,

Department of Computer Science and Engineering.

East West University.

***Submitted By:***

1. **Riffat Sharmin**  
   ID: 2015-2-60-010
2. **Shaykh Siddique**  
   ID: 2016-1-60-053

**Introduction:**

In this emerging world of computers, almost all manual system has switched to computerized system. So, we are developing a software “Train Ticket Reservation System”. We hope that this software will help us to bring more comfort in our life.

This project introduces train ticket reservation system. It explained how reservation has been done in railways. The systematic procedure is explained in this project. This project is developed in c++ language. We have used Stack, Queue, Link list and various functions in this project.

In this project the customers can create a user id to start their reservation for the rail. Then they will be able to have an account. The account contains comprehensive information of the user entered during registration and permits the customer to get access to his or her reservation, enquire about travel fare and the time table of his desired train. And also they can see how many sits are available and also can make fresh reservation. Each passenger will have an unique password.

There is another member involved in this transaction, which is an admin. The admin login using his master password and he have access to all the system of this software. This includes the account information of the customers, description of the trains and their destination.

The main target of our software is bring comfort to the people’s life.

**Key of the project:**

We have used C++ language to build our project.

**Running Platform:**

Windows, Linux.

**Purpose of the project:**

The purpose of this software is to describe the Train Ticket Reservation System which provides the reservation, train timing, enquiry, billing and cancellation on reservation.

Project Functions:

**1. Train Details:**

In this section we can get all the details of the trains. Such as – their names, their destination, avalabilty of sits, arrival and departure time etc.

**2. Reservation:**

In this section users can buy or reserve train ticket. Also they will be able to see the schedule of the trains and train fare.

**3. Cancellation:**

In this section, users can cancel their ticket if they want.

**4. Admin:**

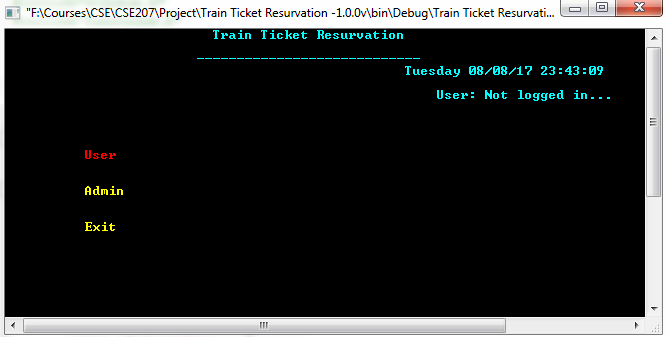
Here, an admin will be able to control all these procedure and also he can change the time table of the trains.And can add or remove train compartments from any train according to their needs.

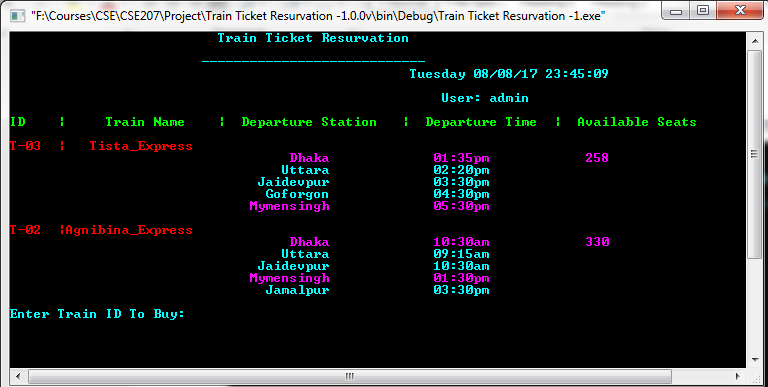
**Risks:** As this is a simulator (virtual) of Train Ticket Reservation based software if it is mix with real stock market based software it will be risky.

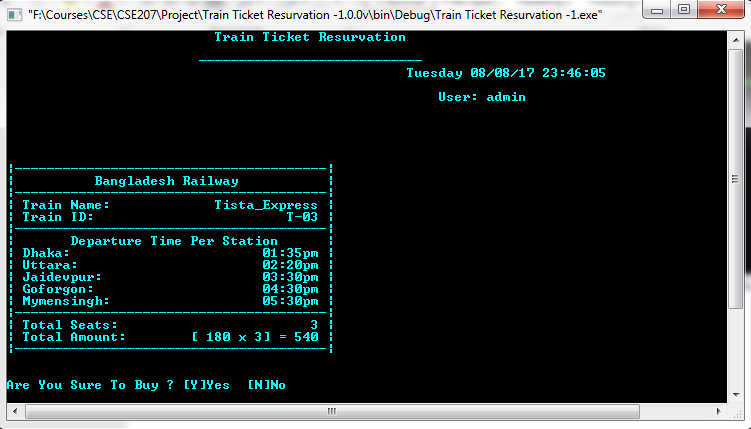
**Goals:** The goal of this software is to introduce general people, how to use stock market based software.

**Project Diagram:**

**Output Screenshot:**

****

****

****

**Source Code:**

main.cpp

1. #include"header.h"
2. #include"stStack.h"
3. #include"stQueue.h"
4. #include"functions.h"
5. **using** **namespace** std;
7. **int** main(){
8. **while**(**true**){
9. userna\_global="Not logged in...";
10. **int** key = main\_Menu();
11. **if**(key==1){
12. //user
13. **bool** choi = user\_menu();
14. }**else** **if**(key==2){
15. //            admin
16. **bool** choi =admin\_login("admin", "1234");
17. **if**(choi)admin\_menu();
18. }**else** **if**(key==99){
19. **return** 0;
20. }**else**{
21. Sleep(500);
22. }
23. }
24. **return** 0;
25. }

header.h

1. #include<bits/stdc++.h>
2. #include<windows.h>
3. #include<conio.h>
4. //    Custom hash define
5. #define pb push\_back
6. #define ull unsigned long long
7. #define debug(x) cerr << #x << " : " << x << endl
8. #define fi first
9. #define se second
10. #define repe(i, n, test) for(int i=n; i<=test; i++)
11. #define rep(i, n, test) for(int i=n; i<test; i++)
13. //        Custom data define
14. #define MOD 1000000007
15. #define EPS 1e-7
16. #pragma comment(linker, "/stack:20000000")
17. #define \_CRT\_SECURE\_NO\_WARNINGS
18. #define MAX 1000005
19. #define default\_value NULL
20. **const** **long** **long** inf = 1000000000LL;
21. **using** **namespace** std;
22. //    global variables
23. string userna\_global="Not logged in...";  /// global variable
24. **class** User{
25. **public**:
26. string user\_name, user\_id, user\_pass;
27. };
28. **struct** Train{
29. string t\_name, t\_id;
30. **int** fares;
31. **int** t\_seats, t\_numOfStaions;
32. map<string, string>t\_time\_mp;
33. vector<string>t\_stations;
34. Train(string t\_n, string t\_i, **int** t\_se, **int** t\_numOfStaions, **int** fares ,vector<string>t\_sta, map<string, string>t\_mp){
35. t\_name = t\_n;
36. t\_id = t\_i;
37. t\_seats = t\_se;
38. t\_stations = t\_sta;
39. t\_time\_mp=t\_mp;
40. }
41. Train(){
42. //empty construction
43. }
44. };

stQueue.h

1. **template**<**class** T>**class** Queue{
2. **public**:
3. Queue();
4. **bool** isEmpty();
5. **void** enqueue(T data);
6. **void** dequeue();
7. T front();
8. **private**:
9. **struct** Node{
10. T date;
11. Node \*next;
12. };
13. Node \*frontPtr;
14. Node \*backPtr;
15. **int** count;
17. };
19. **template**<**class** T>Queue<T>::Queue(): frontPtr(NULL), backPtr(NULL), count(0){
20. //    defult constructor
21. }
22. **template**<**class** T>**bool** Queue<T>::isEmpty(){
23. **return**(count == 0);
24. }
25. **template**<**class** T>**void** Queue<T>::enqueue(T data){
26. Node \*newOne = **new** Node;
27. newOne->date = data;
28. newOne->next = NULL;
29. **if**(isEmpty()){
30. frontPtr = newOne;
31. }
32. **else**{
33. backPtr->next = newOne;
34. }
35. backPtr = newOne;
36. count++;
37. }
38. **template**<**class** T>**void** Queue<T>::dequeue(){
39. **if**(isEmpty()){
40. cout << "Nothing inside" << endl;
41. }
42. **else**{
43. Node \*temp = frontPtr;
44. **if**(frontPtr == backPtr){
45. frontPtr = NULL;
46. backPtr = NULL;
47. }
48. **else**{
49. frontPtr = frontPtr->next;
50. }
51. **delete** temp;
52. count--;
53. }
54. }
55. **template**<**class** T>T Queue<T>::front(){
56. **return** frontPtr->date;
57. }

st\_Stack.h

1. **template** <**class** Type>**struct** Node{
2. Node(Type data, Node<Type>\* next): data(data), next(next) {}
3. Node\* next;
4. Type data;
5. };
6. **template** <**class** Type>**class** Stack{
7. **public**:
8. Stack() : length(0), topNode(NULL){
9. }
10. ~Stack() {
11. **while** (!isEmpty()) {
12. pop();
13. }
14. }
15. **void** push(Type data){
16. Node<Type>\* newNode = **new** Node<Type>(data, topNode);
17. topNode = newNode;
18. ++length;
19. }
20. Type pop(){
21. **if** (!isEmpty()){
22. Node<Type>\*popped = topNode;
23. Type poppedData = popped->data;
24. topNode = popped->next;
25. --length;
26. **delete** popped;
27. **return** poppedData;
28. }
29. }
30. **bool** isEmpty() {
31. **return** length == 0;
32. }
33. **void** print() **const** {
34. Node<Type>\*tempTop = topNode;
35. **while** (tempTop != NULL) {
36. cout << tempTop->data << endl;
37. tempTop = tempTop->next;
38. }
39. }
40. **int** count() **const** {
41. **return** length;
42. }
43. Type Top(){
44. Node<Type>\*tempTop = topNode;
45. **return** tempTop->data;
46. }
47. **private**:
48. Node<Type>\*topNode;
49. **int** length;
50. };

functions.h

1. **bool** user\_menu();
2. **int** main();
3. **void** admin\_menu();
4. **HANDLE** con=GetStdHandle(STD\_OUTPUT\_HANDLE);  //global variable
5. **void** textattr(**int** color){
6. SetConsoleTextAttribute(con, color);
7. }
8. **void** gettime\_date(){                                    ///return time and date
9. **time\_t** t=time(NULL);
10. **char** tmstr[100];
11. **if**(strftime(tmstr, **sizeof**(tmstr), "%A %c", localtime(&t))){
12. cout<<setw(75)<<tmstr<<endl<<endl;
13. }
14. }
15. **void** displaybar(){                                       ///display name-tag & time-date
17. system("cls");
18. textattr(11);
19. cout<<setw(50)<<"Train Ticket Resurvation"<<endl<<endl;
20. cout<<setw(52)<<"----------------------------"<<endl;
21. gettime\_date();
22. cout<<setw(60)<<"User: "<<userna\_global<<endl<<endl;
23. }
24. **void** gotoxy(**int** x,**int** y){
25. COORD Coord;
26. Coord.X=x;
27. Coord.Y=y;
29. SetConsoleCursorPosition(con,Coord);
30. }
31. **char** menu\_dy(string \*menu\_list, **int** MaxNo\_Menu){
32. **HANDLE** hConsoleOutput;
33. CONSOLE\_CURSOR\_INFO structCursorInfo;
34. hConsoleOutput = GetStdHandle( STD\_OUTPUT\_HANDLE );
35. GetConsoleCursorInfo( hConsoleOutput, &structCursorInfo );
36. structCursorInfo.bVisible = FALSE;
37. SetConsoleCursorInfo( hConsoleOutput, &structCursorInfo );
38. **int** i,
39. xpos = 10,
40. ypos[MaxNo\_Menu];
41. **int** j=10;
42. **for**(**int** i=0; i<MaxNo\_Menu; i++){
43. ypos[i] = j;
44. j+=3;
45. }
46. **for** (i=0; i< MaxNo\_Menu; ++i){
47. gotoxy(xpos, ypos[i] );
48. textattr(14);
49. cout<<menu\_list[i];
50. }
52. i=0;
53. **while**(1)
54. {
55. gotoxy(xpos, ypos[i]);
56. textattr(12 );
57. cout<<menu\_list[i];
59. /\* note : 72 -> UP button
60. 75 -> RIGHT button
61. 77 -> LEFT button
62. 80 -> DOWN button
63. \*/
64. **switch**( \_getch() ){
65. **case** 72: **if**(i>0) {
66. gotoxy(xpos,ypos[i] );
67. textattr(14);
68. cout<<menu\_list[i];
69. --i;
70. }
71. **break**;
72. **case** 80: **if**(i< MaxNo\_Menu-1 )
73. {
74. gotoxy(xpos,ypos[i] );
75. textattr(14);
76. //                printf("%s", menu\_list[i] );
77. cout<<menu\_list[i];
78. ++i;
79. }
80. **break**;
81. **case** 13:
82. **for**(**int** k=0; k<MaxNo\_Menu; k++){
83. **if**(i==k){
84. **int** chr=0;
85. gotoxy (10,1);
86. **return** 'A'+i;
87. }
88. }
89. **break**;
90. }
91. }
92. }
94. //main menu
95. **int** main\_Menu(){
96. displaybar();
97. **int** key;
98. string menu\_1[]={"User", "Admin", "Exit"};
99. **char** ch=menu\_dy(menu\_1, 3);
100. **if**(ch=='A') **return** 1;
101. **else** **if**(ch=='B') **return** 2;
102. **else** **if**(ch=='C') **return** 99;
104. cin>>key;
105. **return** key;
106. }
107. **bool** admin\_login(string admin\_id, string admin\_pass){
108. displaybar();
109. cout<<endl<<setw(20)<<"Login Panel"<<endl<<endl;
110. string usrid;
111. cout<<"Admin ID: ";
112. textattr(15);
113. cin>>usrid;
114. textattr(11);
115. **if**(usrid==admin\_id){
116. string pass ="";
117. **char** ch;
118. cout << "Enter password: ";
119. textattr(15);
120. ch = \_getch();
121. **while**(ch != 13){//character 13 is enter
122. pass.push\_back(ch);
123. cout << '\*';
124. ch = \_getch();
125. }
126. textattr(11);
127. **if**(pass == admin\_pass){
128. userna\_global="admin";
129. **int** o=33, p=33, q=33, r=33, s=33;
130. textattr(10);
131. **for**( ; ; ){
132. displaybar();
133. printf("\nAccess granted %c %c %c %c %c\n", q, o, p, r, s);
134. //Sleep(2);
135. **if**(q<46){
136. q++;
137. }
138. **else** **if**(q==46 && o<46)
139. o++;
140. **else** **if**(q==46 && o==46 && p<46)
141. p++;
142. **else** **if**(q==46 && o==46 && p==46 && r<46)
143. r++;
144. **else** **if**(q==46 && o==46 && p==46 && r==46 && s<46)
145. s++;
146. **else**
147. **break**;
148. }
149. admin\_menu();
150. }**else**{
151. textattr(12);
152. cout <<endl<< "\nAccess aborted...\n";
153. Sleep(800);
154. **return** **false**;
155. }
156. }**else**{
157. textattr(12);
158. cout<<endl<<"Wrong user ID"<<endl;
159. Sleep(800);
160. **return** **false**;
161. }
162. **return** **true**;
163. }
165. **bool** user\_login(){
167. displaybar();
168. cout<<endl<<setw(20)<<"Login Panel"<<endl<<endl;
169. string usrid;
170. cout<<"User ID: ";
171. textattr(15);
172. cin>>usrid;
173. string fn\_no;
174. string id;
175. string pas;
176. string name;
177. ifstream userfile("userfile.txt");
178. **bool** check=**false**;
179. string psrd;
180. **while**(userfile >> id >> pas >>fn\_no){
181. **if**(id==usrid){
182. psrd=pas;
183. check=**true**;
184. **break**;
185. }
186. }
187. **if**(check){
188. string pass ="";
189. **char** ch;
190. textattr(11);
191. cout << "Enter password: ";
192. textattr(15);
193. ch = \_getch();
194. **while**(ch != 13){//character 13 is enter
195. pass.push\_back(ch);
196. cout << '\*';
197. ch = \_getch();
198. }
200. **if**(pass == psrd){
201. userna\_global=usrid;
202. **int** o=33, p=33, q=33, r=33, s=33;
203. textattr(10);
204. **for**( ; ; ){
206. displaybar();
207. printf("\nAccess granted %c %c %c %c %c\n", q, o, p, r, s);
208. //Sleep(2);
209. **if**(q<46){
210. q++;
211. }
212. **else** **if**(q==46 && o<46)
213. o++;
214. **else** **if**(q==46 && o==46 && p<46)
215. p++;
216. **else** **if**(q==46 && o==46 && p==46 && r<46)
217. r++;
218. **else** **if**(q==46 && o==46 && p==46 && r==46 && s<46)
219. s++;
220. **else**
221. **break**;

224. }
225. userfile.close();
226. **return** **true**;
227. textattr(11);
228. }**else**{
229. textattr(12);
230. cout << endl<<"\nAccess aborted...\n";
231. Sleep(800);
232. userfile.close();
233. user\_menu();
234. }
235. }**else**{
236. textattr(12);
237. cout<<"Wrong user ID"<<endl;
238. Sleep(800);
239. userfile.close();
240. user\_menu();
241. }
242. }
244. **void** user\_reg(){
245. displaybar();
246. string usr\_id, usr\_pass, usr\_moNo;
247. cout<<endl<<setw(20)<<"Create New Account"<<endl<<endl;
248. cout<<"Enter User ID: ";
249. textattr(15);
250. cin>>usr\_id;
251. textattr(11);
252. cout<<"Enter Password: ";
253. textattr(15);
254. cin>>usr\_pass;
255. textattr(11);
256. cout<<"Enter Mobile Number: ";
257. textattr(15);
258. cin>>usr\_moNo;
259. textattr(11);
260. ofstream userfile("userfile.txt", ios::app);
261. userfile<<usr\_id<<" "<<usr\_pass<<" "<<usr\_moNo<<endl;
262. userfile.close();
263. system("cls");
264. displaybar();
265. cout<<endl<<endl;
266. textattr(10);
267. cout<<"Registration Completed"<<endl;
268. Sleep(700);
269. }



274. **bool** print\_tickets(string user\_name, Train t\_na, **int** seats){
275. displaybar();
276. cout<<endl<<endl<<endl<<endl;
277. cout<<"|---------------------------------------|"<<endl;
278. cout<<"|          Bangladesh Railway           |"<<endl;
279. cout<<"|---------------------------------------|"<<endl;
280. cout<<"| Train Name: "<<setw(27)<<t\_na.t\_name+" |"<<endl;
281. cout<<"| Train ID: "<<setw(29)<<t\_na.t\_id+" |"<<endl;
282. cout<<"|---------------------------------------|"<<endl;
283. cout<<"|       Departure Time Per Station      |"<<endl;
284. rep(i, 0, t\_na.t\_stations.size()){
285. cout<<"| "<<t\_na.t\_stations[i]+": "<<setw(37-t\_na.t\_stations[i].size())<<t\_na.t\_time\_mp[t\_na.t\_stations[i]]+" |"<<endl;
286. }
287. cout<<"|---------------------------------------|"<<endl;
288. cout<<"| Total Seats: "<<setw(24)<<seats<<" |"<<endl;
289. cout<<"| Total Amount: "<<setw(9)<<"[ "<<t\_na.fares<<" x "<<seats<<"] = "<<seats\*t\_na.fares<<" |"<<endl;
290. cout<<"|---------------------------------------|"<<endl;

293. cout<<endl<<endl;
294. cout<<"Are You Sure To Buy ? [Y]Yes  [N]No";
295. **char** choice = \_getch();
296. **if**(choice=='Y' or choice=='y'){
297. ofstream ticket\_file("User\_tickets/"+user\_name+"\_tickets.txt", ios::app);
298. ticket\_file<<"|---------------------------------------|"<<endl;
299. ticket\_file<<"|          Bangladesh Railway           |"<<endl;
300. ticket\_file<<"|---------------------------------------|"<<endl;
301. ticket\_file<<"| Train Name: "<<setw(27)<<t\_na.t\_name+" |"<<endl;
302. ticket\_file<<"| Train ID: "<<setw(29)<<t\_na.t\_id+" |"<<endl;
303. ticket\_file<<"|---------------------------------------|"<<endl;
304. ticket\_file<<"|       Departure Time Per Station      |"<<endl;
305. rep(i, 0, t\_na.t\_stations.size()){
306. ticket\_file<<"| "<<t\_na.t\_stations[i]+": "<<setw(37-t\_na.t\_stations[i].size())<<t\_na.t\_time\_mp[t\_na.t\_stations[i]]+" |"<<endl;
307. }
308. ticket\_file<<"|---------------------------------------|"<<endl;
309. ticket\_file<<"| Total Seats: "<<setw(24)<<seats<<" |"<<endl;
310. ticket\_file<<"| Total Amount: "<<setw(9)<<"[ "<<t\_na.fares<<" x "<<seats<<"] = "<<seats\*t\_na.fares<<" |"<<endl;
311. ticket\_file<<"|---------------------------------------|"<<endl;
312. ticket\_file.close();
313. displaybar();
314. textattr(10);
315. cout<<endl<<endl<<"\tYou have successfully buy"<<endl;
316. cout<<endl<<"\tHappy Journey"<<endl;
317. Sleep(600);
318. **return** **true**;
319. }
320. **else**
321. **return** **false**;
322. }

325. **void** buy\_tickets(){
326. displaybar();
327. string \_from, \_to;
328. set<string>all\_sta;
329. string tr\_na, tr\_id, tr\_st, tr\_ti;
330. **int** tr\_se, n, tak;
331. vector<string>tr\_sta;
332. ifstream Trainfile ("trainFile.txt");
333. Queue<Train>tr\_que;
334. **int** ind\_t=0;
335. **while**(Trainfile>>tr\_na>>tr\_id>>tr\_se>>n>>tak){
336. tr\_sta.clear();
337. **int** no=n;
338. map<string, string>time\_mp;
339. **while**(n--){
340. Trainfile>>tr\_st>>tr\_ti;
341. all\_sta.insert(tr\_st);
342. tr\_sta.pb(tr\_st);
343. time\_mp[tr\_st]=tr\_ti;
344. }
345. Train T1(tr\_na, tr\_id, tr\_se, no, tak, tr\_sta, time\_mp);
346. T1.t\_numOfStaions=no;
347. T1.fares=tak;
348. ind\_t++;
349. tr\_que.enqueue(T1);
350. tr\_sta.clear();
351. }
352. Trainfile.close();
353. string from\_menu[100];
354. **int** i=0;
355. **for** (set<string>::iterator it=all\_sta.begin(); it!=all\_sta.end(); ++it){
356. from\_menu[i]=\*it;
357. i++;
358. }
359. cout<<"Select The Station From: ";
360. **char** ch = menu\_dy(from\_menu, i-1);
362. displaybar();
363. cout<<"Select Destination: ";
364. cout<<endl<<endl<<endl<<endl<<setw(60)<<"From: "<<from\_menu[ch-'A'];
365. **char** ch2 = menu\_dy(from\_menu, i-1);
366. cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<setw(60)<<"To: "<<from\_menu[ch2-'A'];
367. //check fare
368. string src\_t = from\_menu[ch-'A'];
369. string des\_t = from\_menu[ch2-'A'];
370. Sleep(500);
372. queue<Train>T\_find;
373. queue<Train>tra\_q;
374. **while**(!tr\_que.isEmpty()){
375. Train T3 = tr\_que.front();
376. tr\_que.dequeue();
377. tra\_q.push(T3);
378. **bool** check\_src=**false**, check\_des=**false**;
379. rep(i, 0, T3.t\_stations.size()){
380. **if**(src\_t==T3.t\_stations[i])
381. check\_src=**true**;
382. **if**(T3.t\_stations[i]==des\_t)
383. check\_des=**true**;
384. }
385. **if**(check\_src and check\_des)
386. T\_find.push(T3);
387. }
389. displaybar();
390. textattr(10);
391. cout<<"ID"<<setw(5)<<"|"<<setw(15)<<"Train Name"<<setw(24)<<"|  Departure Station"<<setw(20)<<"|  Departure Time"<<setw(20)<<"|  Available Seats"<<endl<<endl;
392. **while**(!T\_find.empty()){
393. Train T2 = T\_find.front();
394. T\_find.pop();
395. textattr(12);
396. cout<<T2.t\_id<<setw(3)<<"|"<<setw(16)<<T2.t\_name<<endl;;
397. rep(i, 0, T2.t\_stations.size()){
398. textattr(11);
399. **if**(src\_t==T2.t\_stations[i])
400. textattr(13);
401. **if**(des\_t==T2.t\_stations[i])
402. textattr(13);
403. **if**(T2.t\_stations[i].size()>7){
404. cout<<setw(40)<<T2.t\_stations[i]<<setw(20)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
405. }**else** **if**(T2.t\_stations[i].size()<5){
406. cout<<setw(40)<<T2.t\_stations[i]<<setw(21-T2.t\_stations[i].size()+i)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
407. }**else**
408. cout<<setw(40)<<T2.t\_stations[i]<<setw(25-T2.t\_stations[i].size()+i)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
409. **if**(!i)
410. cout<<T2.t\_seats<<endl;
411. **else**{
412. cout<<setw(3)<<endl;
413. }
414. textattr(11);
415. }
416. cout<<endl;
417. }
418. string buy\_id;
419. **int** numOfseats;
420. cout<<"Enter Train ID To Buy: ";
421. cin>>buy\_id;
422. cout<<"Enter Number OF Seats: ";
423. cin>>numOfseats;
425. ofstream TrainFile ("trainFile.txt");
426. **while**(!tra\_q.empty()){
427. Train T2=tra\_q.front();
428. tra\_q.pop();
429. **if**(T2.t\_id==buy\_id){
430. **bool** chkkk=print\_tickets(userna\_global, T2, numOfseats);
431. **if**(chkkk)
432. T2.t\_seats-=numOfseats;
433. }
434. TrainFile<<T2.t\_name<<" "<<T2.t\_id<<" "<<T2.t\_seats<<" "<<T2.t\_numOfStaions<<" "<<T2.fares<<endl;
435. **int** i= 0;
436. **while**(i<T2.t\_stations.size()){
437. TrainFile<<T2.t\_stations[i]<<" "<<T2.t\_time\_mp[T2.t\_stations[i]]<<endl;
438. i++;
439. }
440. }
441. }
443. **void** display\_all\_tickets(){
444. ifstream Ticket\_file("User\_tickets/"+userna\_global+"\_tickets.txt");
445. string line;
446. cout<<endl<<endl<<endl;
447. cout<<endl<<endl<<endl;
448. **int** k=0;
449. **while**(getline(Ticket\_file, line)){
450. cout<<"\t\t"<<line<<endl;
451. k++;
452. }
453. cout<<endl;
454. Ticket\_file.close();
455. **if**(k<3)
456. cout<<"No Tickets Available"<<endl<<endl;
457. cout<<"Press Any To Return ";
458. \_getch();
459. }
461. **void** account\_info(){
462. displaybar();
463. textattr(10);
464. string usr\_na, mobl, pass;
465. ifstream userfile("userfile.txt");
466. **while**(userfile>>usr\_na>>pass>>mobl){
467. **if**(usr\_na==userna\_global) **break**;
468. }
469. cout<<"Username: "<<usr\_na<<endl;
470. cout<<"Mobile Number: "<<mobl<<endl;
471. cout<<endl<<"Press Any To Return ";
472. \_getch();
473. }
475. **void** user\_nav\_menu(){
476. string usr\_nav[]={"Buy Tickets", "Check Own Tickets", "Account Info", "Logout"};
477. displaybar();
478. **char** ch = menu\_dy(usr\_nav, 4);
479. **if**(ch=='A'){
480. //Buy Tickes
481. buy\_tickets();
482. user\_nav\_menu();
483. }**else** **if**(ch=='B'){
484. //Check own
485. display\_all\_tickets();
486. user\_nav\_menu();
487. }**else** **if**(ch=='C'){
488. //Cancel Tickets
489. account\_info();
490. user\_nav\_menu();
491. }**else** **if**(ch=='C'){
492. main();
493. }
494. }
496. **bool** user\_menu(){
497. string usr\_menu[] = {"Login", "Registration New Account", "Back To Main Menu"};
499. displaybar();
500. **char** ch = menu\_dy(usr\_menu, 3);
501. **if**(ch=='A'){
502. //login panel
503. **bool** checker=user\_login();
504. user\_nav\_menu();
505. **if**(checker){
506. //login success
507. }
508. }**else** **if**(ch=='B'){
509. //Registration new account
510. user\_reg();
512. }**else** **if**(ch=='C') **return** **true**;
514. }
516. **void** add\_train(){
517. displaybar();
518. string t\_n, t\_id, t\_st, t\_ti;
519. **int** t\_se, t\_sop;
520. textattr(11);
521. cout<<"Train Name: ";
522. textattr(15);
523. cin>>t\_n;
524. textattr(11);
525. cout<<"Train ID: ";
526. textattr(15);
527. cin>>t\_id;
528. textattr(11);
529. cout<<"Available Seats: ";
530. textattr(15);
531. cin>>t\_se;
532. textattr(11);
533. cout<<"Number of Stoppage: ";
534. textattr(15);
535. cin>>t\_sop;
536. textattr(11);
537. cout<<"Enter Station Root and time: [-1 to end]"<<endl;
539. ofstream Trainfile("trainFile.txt", ios::app);
541. Trainfile<<endl<<t\_n<<" "<<t\_id<<" "<<t\_se<<" "<<t\_sop<<endl;
543. **while**(t\_sop--){
544. textattr(15);
545. cin>>t\_st>>t\_ti;
546. debug(t\_sop);
547. Trainfile<<t\_st<<" "<<t\_ti<<endl;
549. }
550. Trainfile.close();
551. displaybar();
552. cout<<"Successfully Added Train"<<endl;
553. Sleep(700);
554. }
556. **void** edit\_time(){
557. displaybar();
558. string edi\_tr\_id, tr\_na, tr\_id, tr\_st, tr\_ti;
559. **int** tr\_se, n, tak;
560. vector<string>tr\_sta;
561. cout<<"Enter Train ID: ";
562. cin>>edi\_tr\_id;
563. ifstream Trainfile ("trainFile.txt");
564. Stack<Train>tr\_stk;
566. **while**(Trainfile>>tr\_na>>tr\_id>>tr\_se>>n>>tak){
567. tr\_sta.clear();
568. **int** no=n;
569. map<string, string>mp;
570. **while**(n--){
571. Trainfile>>tr\_st>>tr\_ti;
572. tr\_sta.pb(tr\_st);
573. mp[tr\_st]=tr\_ti;
574. }
575. Train T1(tr\_na, tr\_id, tr\_se, no, tak, tr\_sta, mp);
576. T1.t\_numOfStaions=no;
577. T1.fares=tak;
578. tr\_stk.push(T1);
579. tr\_sta.clear();
580. }
581. Trainfile.close();
583. ofstream Trainfile2("trainFile.txt");
584. **while**(!tr\_stk.isEmpty()){
585. Train T2 = tr\_stk.Top();
586. tr\_stk.pop();
587. **if**(T2.t\_id==edi\_tr\_id){
588. textattr(12);
589. cout<<"Train Name: "<<T2.t\_name<<endl;
590. cout<<"Train Seats: "<<T2.t\_seats<<endl;
591. cout<<"Edit Seats: ";
592. textattr(15);
593. cin>>T2.t\_seats;
594. textattr(10);
595. cout<<"Stations"<<setw(15)<<"| "<<"Departure Time"<<setw(15)<<"| "<<"New Departure Time"<<endl<<endl;
596. **int** ind=0;
597. **while**(ind<T2.t\_stations.size()){
598. cout<<T2.t\_stations[ind]<<setw(23-T2.t\_stations[ind].size())<<"| "<<T2.t\_time\_mp[T2.t\_stations[ind]]<<setw(22)<<"| ";
599. textattr(11);
600. cin>>T2.t\_time\_mp[T2.t\_stations[ind]];
601. textattr(10);
602. ind++;
603. }
604. }
605. Trainfile2<<T2.t\_name<<" "<<T2.t\_id<<" "<<T2.t\_seats<<" "<<T2.t\_numOfStaions<<" "<<T2.fares<<endl;
606. **int** i= 0;
607. **while**(i<T2.t\_stations.size()){
608. Trainfile2<<T2.t\_stations[i]<<" "<<T2.t\_time\_mp[T2.t\_stations[i]]<<endl;
609. i++;
610. }
611. }
612. **int** kkk;
613. cin>>kkk;
614. }
616. **void** print\_ava\_seats(){
617. displaybar();
618. string tr\_na, tr\_id, tr\_st, tr\_ti;
619. **int** tr\_se, n, tak;
620. vector<string>tr\_sta;
621. ifstream Trainfile ("trainFile.txt");
622. Queue<Train>tr\_que;
624. **while**(Trainfile>>tr\_na>>tr\_id>>tr\_se>>n>>tak){
625. tr\_sta.clear();
626. **int** no=n;
627. map<string, string>mp;
628. **while**(n--){
629. Trainfile>>tr\_st>>tr\_ti;
630. tr\_sta.pb(tr\_st);
631. mp[tr\_st]=tr\_ti;
632. }
633. Train T1(tr\_na, tr\_id, tr\_se, no, tak, tr\_sta, mp);
634. T1.t\_numOfStaions=no;
635. T1.fares=tak;
636. tr\_que.enqueue(T1);
637. tr\_sta.clear();
638. }
639. Trainfile.close();
640. textattr(10);
641. cout<<"ID"<<setw(5)<<"|"<<setw(15)<<"Train Name"<<setw(24)<<"|  Departure Station"<<setw(20)<<"|  Departure Time"<<setw(20)<<"|  Available Seats"<<endl<<endl;
642. **while**(!tr\_que.isEmpty()){
643. Train T2 = tr\_que.front();
644. tr\_que.dequeue();
645. textattr(12);
646. cout<<T2.t\_id<<setw(3)<<"|"<<setw(16)<<T2.t\_name<<endl;;
647. rep(i, 0, T2.t\_stations.size()){
648. textattr(11);
649. **if**(T2.t\_stations[i].size()>7){
650. cout<<setw(40)<<T2.t\_stations[i]<<setw(20)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
651. }**else** **if**(T2.t\_stations[i].size()<5){
652. cout<<setw(40)<<T2.t\_stations[i]<<setw(21-T2.t\_stations[i].size()+i)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
653. }**else**
654. cout<<setw(40)<<T2.t\_stations[i]<<setw(25-T2.t\_stations[i].size()+i)<<T2.t\_time\_mp[T2.t\_stations[i]]<<setw(15);
655. **if**(!i)
656. cout<<T2.t\_seats<<endl;
657. **else**{
658. cout<<setw(3)<<endl;
659. }
660. }
661. cout<<endl;
662. }
663. cout<<endl<<endl<<"Press Enter To Return"<<endl;
664. getch();
665. }

668. **void** admin\_menu(){
669. **if**(userna\_global=="Not logged in...") **return**;
670. string adm\_menu[]={"Add A New Train", "Edit Time Table & Seats", "Check Schedule", "Add New User", "Logout"};
671. displaybar();
672. **char** ch = menu\_dy(adm\_menu, 5);
673. **if**(ch=='A'){
674. //add new train
675. add\_train();
676. admin\_menu();
677. }**else** **if**(ch=='B'){
678. //edit time table
679. edit\_time();
680. admin\_menu();
681. }**else** **if**(ch=='C'){
682. //available seats
683. print\_ava\_seats();
684. admin\_menu();
685. }**else** **if**(ch=='D'){
686. //add new user
687. user\_reg();
688. }**else** **if**(ch=='E'){
689. //        logout
690. main();
691. }
692. }   //thank you