A Mini Project Report

on

**BORING COMPANY**

**Course:** Design and Analysis of Algorithms Lab

Sem: IV Sec: CSE-B

By

ASHISH PERALA

1602-19-733-065

&

GOPI KRISHNA KULAKARNI

1602-19-733-069



**Department of Computer Science & Engineering**

**Vasavi College of Engineering (Autonomous)**

**Ibrahimbagh, Hyderabad-31**

**2021**

**ACKNOWLEDGEMENT**

We whole heartedly thank **Mr. Elon Musk** for contributing so much to this world and make our life comfortable with his new ideologies.

We would also like to express our special gratitude to our mam **Dr.V.Sireesha** who supported and guided us in making this project successful.

We are grateful to our Head of Department, **Dr. T. Adilakshmi**, for her steady support and the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, **Dr. S.V. Ramana** garu , as well as the management of the institute, for having designed an excellent learning atmosphere.

**ABSTRACT**

Generally, we see a lot of traffic during these days. Traffic congestion has been one of the major issues that most metropolises are facing despite measures being taken to mitigate and reduce it. There have been attempts to develop congestion measurement indices for heavily motorized countries. Even after the Metro Rail System has included in many metropolitan cities, we did not observe much change. In this regard, our project aims to develop a prototype to alleviate traffic congestion and enable rapid transit across densely populated areas.

Concepts used : Arrays, Linked Lists, Queues, Stacks(Using Files), Graphs, Files

Design Strategy: Greedy Algorithm

Language used : C Programming Language.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.No | Contents | Page No. |
| 1 | Abstract | 4 |
| 2 | Introduction | 5 |
| 3 | Description &  Algorithm | 6-7 |
| 4 | Implementation/Code | 8-32 |
| 5 | Results/Screenshots | 33-37 |
| 6 | Conclusion & Future Scope | 38 |
| 7 | References | 38 |

**INTRODUCTION**

What is this BORING COMPANY? The main self-stated goal of this project is to solve the problem of soul-destroying traffic by building a network of transportation corridors underground. With tunnels, traffic is pushed underground, limiting noise, and preventing the division of communities that a large multilane highway exacerbates.

**WORKING**

There will be boring trays which carries them through tunnels. The user who ever feels the road traffic is heavy can enter the tray which carries the user into the tunnels. These tunnels are paths connected from one city to other internally. So, the user chooses his/her source and destination and enter the tray. Within stipulated limit of time the user reaches his destination, and the tray would be empty so that next person who is waiting in the queue for the same tray can enter the tray and so his journey begins. At his destination he/she would be charged an amount for his travel which can be debited through his wallet. The user can also deliver packages. The user can also opt to travel through boring buses.

**DESCRIPTION**

We have considered MAP OF HOGWARTS for prototype and is shown below:![Diagram

Description automatically generated]()

Firstly, we have considered eight regions which enables user to travel through and each of two regions have connected paths with distance (in km) mentioned, respectively.

**PHASE 1:** LOGIN/SIGNUP AND DEALING WITH ACCOUNTS

When the user starts the application, he/she is asked to login/signup.

We have also added options to check his/her travel history, clear it if needed, and check his wallet balance.

**PHASE 2:**

The user is asked to select his/her source and destination and is asked to opt a mode of travel.

Case I: If the user wants a boring tray; he is asked to pick a tray in which he would like to begin his/her journey. Tray suggestions will be displayed to make his/her choice. Every tray has a queue.

* If the queue is empty user will be asked to enter the tray.
* If queue is not empty the user will be asked to wait for some time until the tray becomes work free. When the queue becomes empty, he will be asked to enter the tray. The cost and time of travel will be displayed. He will be travelling at a speed of 180 kmph once he/she enters the tunnel. His current journey details will be pushed into the stack of history file.

Case II : If the user wants to travel in a boring bus or deliver a package, he is asked to select a source and destination and if the wallet balance is sufficient for his journey he can travel/deliver else it gets denied.

After his/her travel, he/she is asked to pay for their travel and here wallet comes in to picture debiting the cost of the journey from their respective account.

**ALGORITHM**

There will be a unique file for each tray, we call it tray queue. Which has the following attributes. At any given time ‘t’, all the users info whoever requested for that tray is displayed on it. And the user’s info who is using that tray at ‘t’ is present on the first line of that file. For instance, let’s consider ‘n’ number of users are requesting for tray numbered as 14 (tray number 1 at location Hogwarts [4]) and users are u1, u2….un. The first user (u1) who requested for that tray gets approved and accessed to the tray. So, there are two options for users (u2, u3 …. un) i.e., either wait for that tray or select another tray. In first case, the waiting time for nth user becomes (t1 + t2 +….t n-1). Where t n is the time required for completion of journey for nth user. In second case, there is no waiting time since the user picks a tray which is currently not used by anyone.

Here, we are greedy with respect to the first requested user and approving his request. Since, the bill is calculated based on the time user spends in tunnels the overall profit remains same. So, if we keep on approving on this basis no tray is kept empty at any given time ‘t’ and overall time is optimized. If we take a look on larger scale like a day’s analysis the profit is also optimized since every second matters.

**IMPLEMENTATION**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

#include<time.h>

#ifdef \_WIN32

#include <Windows.h>

#else

#include <unistd.h>

#endif

#define MAX 256

struct node{

int vertex, weight;

struct node\* next;

};

struct AdjList{

struct node \*head;

};

struct Graph{

int V;

char\* arrayOfCities[8];

struct AdjList\* array;

};

struct MinHeapNode{

int v;

int dist;

};

struct MinHeap{

int size;

int capacity;

int \*pos;

struct MinHeapNode \*\*array;

};

char userName[30];

void reg(){

FILE \*ptr1;

ptr1=fopen("user.txt","a+");

char password[30];

printf("ENTER USERNAME\n");

scanf("%s",userName);

printf("ENTER PASSWORD\n");

scanf("%s",password);

fprintf(ptr1,"%s %s\n",userName,password);

fclose(ptr1);

FILE \*fptr1;

fptr1 = fopen(userName, "a+");

fprintf(fptr1, "W:%d\n",500);

fclose(fptr1);

}

int login(){

char tempUserName[30],tempPass[30],pass[30];

printf("ENTER USERNAME\n");

scanf("%s",userName);

FILE \*ptr1;

ptr1=fopen("user.txt","a+");

while(!feof(ptr1)){

fscanf(ptr1,"%s %s\n",tempUserName,tempPass);

if(strcmp(tempUserName,userName)==0){

pass:

printf("ENTER PASSWORD\n");

scanf("%s",pass);

if(strcmp(pass,tempPass)==0){

printf("\nLOGIN SUCCESSFUL !!!\n");

fclose(ptr1);

return 0;

}

else{

printf("\nWRONG PASSWORD");

goto pass;

}

}

}

printf("\nUSER NOT FOUND PLEASE TRY AGAIN\n");

login();

return 0;

}

int getBalence(char fname[]){

FILE \*ptr1;

ptr1=fopen(fname,"r");

if (!ptr1)

{

printf("ERROR009");

}

fseek(ptr1, -10, SEEK\_END);

int bal;

char ch;

while(1)

{

ch = fgetc(ptr1);

if(ch=='W')

{

fseek(ptr1, 1, SEEK\_CUR);

fscanf(ptr1,"%d\n",&bal);

}

if(ch == EOF)

break;

}

fclose(ptr1);

return bal;

}

void putBalence(char fname[],int bal){

FILE \*ptr1;

ptr1=fopen(fname,"r+");

if (!ptr1)

{

printf("ERROR010");

}

fseek(ptr1, -10, SEEK\_END);

char ch;

while(1)

{

ch = fgetc(ptr1);

if(ch=='W')

{

fseek(ptr1, -1, SEEK\_CUR);

fprintf(ptr1, "W:%d ",bal);

}

if(ch == EOF)

break;

}

fclose(ptr1);

}

void addToHistory(char fname[],char from[],char to[]){

char ch;

FILE \*fptr1,\*fptr2;

char str[MAX], temp[] = "temp.txt";

fptr1 = fopen(fname, "a+");

if (!fptr1)

{

printf("ERROR005\n");

return ;

}

fptr2 = fopen(temp, "a+");

if (!fptr2)

{

printf("ERROR006\n");

fclose(fptr1);

return ;

}

time\_t t;

time(&t);

char info[MAX];strcpy(info,ctime(&t));

fprintf(fptr2, "FROM:%s TO:%s ON %s",from,to,info);

while (!feof(fptr1))

{

strcpy(str, "\0");

fgets(str, MAX, fptr1);

if (!feof(fptr1))

{

fprintf(fptr2, "%s", str);

}

}

fclose(fptr1);

fclose(fptr2);

remove(fname);

rename(temp, fname);

}

void poll(char fname[]){

int ctr = 0,lno=0;

char ch;

FILE \*fptr1,\*fptr2;

char str[MAX], temp[] = "temp.txt";

fptr1 = fopen(fname, "a+");

if (!fptr1)

{

printf("ERROR003\n");

return ;

}

fptr2 = fopen(temp, "a+");

if (!fptr2)

{

printf("ERROR004\n");

fclose(fptr1);

return ;

}

lno++;

while (!feof(fptr1))

{

strcpy(str, "\0");

fgets(str, MAX, fptr1);

if (!feof(fptr1))

{

ctr++;

if (ctr != lno)

{

fprintf(fptr2, "%s", str);

}

}

}

fclose(fptr1);

fclose(fptr2);

remove(fname);

rename(temp, fname);

}

void showHistory(char fname[]){

FILE \*ptr1;

ptr1=fopen(fname,"r");

if (!ptr1)

{

printf("ERROR007");

}

// Read contents from file

char c;

c = fgetc(ptr1);

while (c != EOF)

{

printf ("%c", c);

c = fgetc(ptr1);

}

fclose(ptr1);

}

void clearHistory(char fname[]){

FILE \*ptr1;

ptr1=fopen(fname,"w");

if (!ptr1)

{

printf("ERROR007");

}

fclose(ptr1);

}

//////

struct MinHeapNode\* newMinHeapNode(int v,int dist){

struct MinHeapNode\* minHeapNode =(struct MinHeapNode\*)malloc(sizeof(struct MinHeapNode));

minHeapNode->v = v;

minHeapNode->dist = dist;

return minHeapNode;

}

struct MinHeap\* createMinHeap(int capacity){

struct MinHeap\* minHeap =(struct MinHeap\*)malloc(sizeof(struct MinHeap));

minHeap->pos = (int \*)malloc(capacity \* sizeof(int));

minHeap->size = 0;

minHeap->capacity = capacity;

minHeap->array =(struct MinHeapNode\*)malloc(capacity \*sizeof(struct MinHeapNode));

return minHeap;

}

void swapMinHeapNode(struct MinHeapNode\*\* a,struct MinHeapNode\*\* b){

struct MinHeapNode\* t = \*a;

\*a = \*b;

\*b = t;

}

void minHeapify(struct MinHeap\* minHeap,int idx){

int smallest, left, right;

smallest = idx;

left = 2 \* idx + 1;

right = 2 \* idx + 2;

if (left < minHeap->size &&minHeap->array[left]->dist <minHeap->array[smallest]->dist )

smallest = left;

if (right < minHeap->size &&minHeap->array[right]->dist <minHeap->array[smallest]->dist )

smallest = right;

if (smallest != idx){

struct MinHeapNode\* smallestNode =minHeap->array[smallest];

struct MinHeapNode\* idxNode =minHeap->array[idx];

minHeap->pos[smallestNode->v] = idx;

minHeap->pos[idxNode->v] = smallest;

swapMinHeapNode(&minHeap->array[smallest],&minHeap->array[idx]);

minHeapify(minHeap, smallest);

}

}

int isEmpty(struct MinHeap\* minHeap){

return minHeap->size == 0;

}

struct MinHeapNode\* extractMin(struct MinHeap\*minHeap){

if (isEmpty(minHeap))

return NULL;

struct MinHeapNode\* root =minHeap->array[0];

struct MinHeapNode\* lastNode =

minHeap->array[minHeap->size - 1];

minHeap->array[0] = lastNode;

minHeap->pos[root->v] = minHeap->size-1;

minHeap->pos[lastNode->v] = 0;

--minHeap->size;

minHeapify(minHeap, 0);

return root;

}

void decreaseKey(struct MinHeap\* minHeap,int v, int dist){

int i = minHeap->pos[v];

minHeap->array[i]->dist = dist;

while (i && minHeap->array[i]->dist <minHeap->array[(i - 1) / 2]->dist)

{

minHeap->pos[minHeap->array[i]->v] =(i-1)/2;

minHeap->pos[minHeap->array[(i-1)/2]->v] = i;

swapMinHeapNode(&minHeap->array[i],&minHeap->array[(i - 1) / 2]);

i = (i - 1) / 2;

}

}

int isInMinHeap(struct MinHeap \*minHeap, int v){

if (minHeap->pos[v] < minHeap->size)

return 1;

return 0;

}

int parent[8];

int getDistance(struct Graph\* graph, int src,int dest){

int V = graph->V;//n.o of vertices

parent[src]=-1;

int dist[V];

struct MinHeap\* minHeap = createMinHeap(V);

for (int v = 0; v < V; ++v)

{

dist[v] = INT\_MAX;

minHeap->array[v] = newMinHeapNode(v,dist[v]);

minHeap->pos[v] = v;

}

minHeap->array[src] =newMinHeapNode(src, dist[src]);

minHeap->pos[src] = src;

dist[src] = 0;

decreaseKey(minHeap, src, dist[src]);

minHeap->size = V;

while (!isEmpty(minHeap))

{

struct MinHeapNode\* minHeapNode =extractMin(minHeap);//minimum in dist array which is unvisited

int u = minHeapNode->v;

struct node\* temp3 =graph->array[u].head;

while (temp3 != NULL)

{

int v = temp3->vertex;

if (isInMinHeap(minHeap, v) &&dist[u] != INT\_MAX &&temp3->weight + dist[u] < dist[v])

{

parent[v]=u;

dist[v] = dist[u] + temp3->weight;

decreaseKey(minHeap, v, dist[v]);

}

temp3 = temp3->next;

}

}

return dist[dest];

}

//////////

float get\_time(struct Graph\* graph,int src, int dest){//returns time of travel

int d;

float timeOfTravel=0.0;

d=getDistance(graph,src,dest);

timeOfTravel=((float)d)/3.0;

return timeOfTravel;

}

void showpath(struct Graph\* graph,int j){

if (parent[j] == - 1)

return;

showpath(graph,parent[j]);

printf("..%s..", graph->arrayOfCities[j]);

}

struct node\* newnode(int vertex, int weight)

{

struct node\* newNode =(struct node\*)malloc(sizeof(struct node));

newNode->vertex = vertex;

newNode->weight = weight;

newNode->next = NULL;

return newNode;

}

struct Graph\* createGraph(int V){

struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph));

graph->V = V;

graph->array = (struct AdjList\*)malloc(V \* sizeof(struct AdjList));

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

graph->array[i].head = NULL;

return graph;

}

void addTunnel(struct Graph\* graph, int src,int vertex, int weight)

{

struct node\* newNode =newnode(vertex, weight);

newNode->next = graph->array[src].head;

graph->array[src].head = newNode;

newNode = newnode(src, weight);

newNode->next = graph->array[vertex].head;

graph->array[vertex].head = newNode;

}

int main() {

int option,trayNum,src,dest,amount,lr,start=0;

char trayq[20];

char passName[25];

int flag=0,travelTime;

float t;

struct Graph\* Hogwarts = createGraph(8);

addTunnel(Hogwarts,0,3,15);

addTunnel(Hogwarts,0,4,13);

addTunnel(Hogwarts,1,4,2);

addTunnel(Hogwarts,1,3,1);

addTunnel(Hogwarts,3,4,2);

addTunnel(Hogwarts,3,5,5);

addTunnel(Hogwarts,5,2,3);

addTunnel(Hogwarts,3,6,3);

addTunnel(Hogwarts,6,7,6);

addTunnel(Hogwarts,6,4,2);

addTunnel(Hogwarts,4,7,20);

Hogwarts->arrayOfCities[0]="Hogsmeade Station";

Hogwarts->arrayOfCities[1]="Black Lake";

Hogwarts->arrayOfCities[2]="Shrieking Shack";

Hogwarts->arrayOfCities[3]="Quidditch Pitch";

Hogwarts->arrayOfCities[4]="Hogwarts";

Hogwarts->arrayOfCities[5]="Hogsmeade village";

Hogwarts->arrayOfCities[6]="Hagrid's Hut";

Hogwarts->arrayOfCities[7]="Forbidden Forest";

start01 :printf("\nLOGIN OR SIGNUP TO CONTINUE\nFOR LOGIN PRESS 1 FOR REGISTER PRESS 2\n");

scanf("%d",&lr);

switch(lr){

case 1:{

login();

}

break;

case 2:{

reg();

}

break;

default:{printf("\nWRONG KEY");goto start01;}

}

int currentBalence,updatedBalence;

start02 :printf("\nWHAT DO YOU LIKE TO DO\n");

printf("1.TO BOOK A BORING TRAY FOR YOUR CAR\n2.TO BOOK A BORING BUS\n3.TO DELIVER A PACKAGE\n4.CHECK YOUR PREVIOUS HISTORY\n5.CLEAR YOUR PREVIOUS HISTORY\n6.TO CHECK YOUR WALLET BALENCE\n7.LOGOUT\n");

scanf("%d",&option);

switch(option){

case 1:{

printf("\nYou can Enter and Exit Trays from regions listed here\n0.HOGSMEADE STATION\n1.BLACK LAKE\n2.SHRIEKING SHACK\n3.QUIDDITCH PITCH\n4.HOGWARTS\n5.HOGSMEADE VILLAGE\n6.HAGRID'S HUT\n7.FORBIDDEN FOREST");

printf("\nENTER SOURCE AND DESTINATION INDICES\n");

scanf("%d%d",&src,&dest);

printf("THERE ARE 4 TRAYS AVAILABLE AT SELECTED LOCATION\nPICK ANY ONE OF THEM AND ENTER TRAY NUMBER\n");

scanf("%d",&trayNum);

sprintf(trayq, "%d", trayNum);

char a[10];

sprintf(a, "%d", src);

strcat(trayq,a);

FILE \*fptr1;

fptr1 = fopen(trayq, "a+");

if (!fptr1) {

printf("\nERROR 001\n");

return 0;

}

char ch=getc(fptr1);

if(ch==EOF){

t=get\_time(Hogwarts,src,dest);

currentBalence=getBalence(userName);

if(currentBalence<(t\*30)){

printf("YOUR BALENCE IS NOT ENOUGH FOR TRAVELLING");

goto start02;

}

printf("QUEUE IS EMPTY PLEASE PROCEED TO THE TRAY\nPRESS 1 ONCE YOU ENTERED THE TRAY\n");

check01:scanf("%d",&start);

if(start==1){

printf("\nWELCOME TO THE TUNNELS YOU WILL BE TRAVELLING AT A SPEED OF 180 KM/HR");

printf("\nYOU WILL BE TRAVELLING VIA %s..",Hogwarts->arrayOfCities[src]);

showpath(Hogwarts,dest);

printf("\nYOUR TOTAL TIME OF TRAVEL IS %f MINUTES\nAND COST OF YOUR TICKET IS %f",t,(t\*30));

updatedBalence=currentBalence-(int)t\*30;

fprintf(fptr1,"%s FROM:%d TO:%d\n",userName,src,dest);

//sleep(t\*60);

fclose(fptr1);

addToHistory(userName,Hogwarts->arrayOfCities[src],Hogwarts->arrayOfCities[dest]);

printf("\n\nYOUR JOURNEY IS COMPLETED");

poll(trayq);

putBalence(userName,updatedBalence);

goto start02;

}

else{

printf("\nWRONG CHOICE PLEASE ENTER AGAIN\n");

goto check01;

}

}

else{

printf("YOU HAVE BEEN ADDED TO QUEUE PLEASE WAIT THE TRAY IS BUSY\n");

sleep(5);

check:fseek(fptr1,0,SEEK\_SET);

char ch=getc(fptr1);

if(ch==EOF){

flag=1;

}

else{

fclose(fptr1);

sleep(5);

fptr1 = fopen(trayq, "a+");

if (!fptr1) {

printf("\nERROR 008\n");

return 0;

}

goto check;

}

if(flag==1){

printf("ITS YOUR TURN NOW PLEASE ENTER THE TRAY\nPRESS 1 ONCE YOU ENTERED THE TRAY\n");

fprintf(fptr1,"%s FROM:%d TO:%d\n",userName,src,dest);

check02:scanf("%d",&start);

if(start==1){

t=get\_time(Hogwarts,src,dest);

sleep(t\*60);

}

else{

printf("\nWRONG CHOICE PLEASE ENTER AGAIN\n");

goto check02;

}

printf("\nWELCOME TO THE TUNNELS YOU WILL BE TRAVELLING AT A SPEED OF 180 KM/HR");

printf("\nYOU WILL BE TRAVELLING VIA %s..",Hogwarts->arrayOfCities[src]);

showpath(Hogwarts,dest);

printf("\nYOUR TOTAL TIME OF TRAVEL IS %f MINUTES\nAND COST OF YOUR TICKET IS %f",t,(t\*30));

updatedBalence=currentBalence-(int)t\*30;

//sleep(t\*60);

addToHistory(userName,Hogwarts->arrayOfCities[src],Hogwarts->arrayOfCities[dest]);

printf("\n\nYOUR JOURNEY IS COMPLETED");

poll(trayq);

putBalence(userName,updatedBalence);

fclose(fptr1);

goto start02;

}

}

goto start02;

}

break;

case 2:

{

printf("\nYou can Enter and Exit buses from regions listed here\n0.HOGSMEADE STATION\n1.BLACK LAKE\n2.SHRIEKING SHACK\n3.QUIDDITCH PITCH\n4.HOGWARTS\n5.HOGSMEADE VILLAGE\n6.HAGRID'S HUT\n7.FORBIDDEN FOREST");

printf("\nENTER SOURCE AND DESTINATION INDICES\n");

scanf("%d%d",&src,&dest);

t=get\_time(Hogwarts,src,dest);

currentBalence=getBalence(userName);

if(currentBalence<(t\*30)){

printf("YOUR BALENCE IS NOT ENOUGH FOR TRAVELLING");

goto start02;

}

printf("\nPRESS 1 ONCE YOU ENTERED THE BUS\n");

check03:scanf("%d",&start);

if(start==1){

printf("\nWELCOME TO THE TUNNELS YOU WILL BE TRAVELLING AT A SPEED OF 180 KM/HR\nYOUR TOTAL TIME OF TRAVEL IS %f MINUTES\nAND COST OF YOUR TICKET IS %f",t,(t\*30));

updatedBalence=currentBalence-(int)t\*30;

sleep(t\*60);

addToHistory(userName,Hogwarts->arrayOfCities[src],Hogwarts->arrayOfCities[dest]);

printf("\n\nYOUR JOURNEY IS COMPLETED");

putBalence(userName,updatedBalence);

goto start02;

}

else{

printf("\nWRONG CHOICE PLEASE ENTER AGAIN\n");

goto check03;

}

}

break;

case 3:

{

printf("\nYou can deliver and recieve pakages from regions listed here\n0.HOGSMEADE STATION\n1.BLACK LAKE\n2.SHRIEKING SHACK\n3.QUIDDITCH PITCH\n4.HOGWARTS\n5.HOGSMEADE VILLAGE\n6.HAGRID'S HUT\n7.FORBIDDEN FOREST");

printf("\nENTER SOURCE AND DESTINATION INDICES\n");

scanf("%d%d",&src,&dest);

t=get\_time(Hogwarts,src,dest);

currentBalence=getBalence(userName);

if(currentBalence<(t\*10)){

printf("YOUR CURRENT BALENCE IS NOT ENOUGH ");

goto start02;

}

printf("\nPRESS 1 ONCE YOU PLACED YOUR PACKAGE IN PACKAGE DELIVERY TRAY\n");

check04:scanf("%d",&start);

if(start==1){

printf("\nYOUR PACKAGE WILL REACH %s IN %f MINUTES STARTING FROM NOW...\nAND COST OF YOUR TICKET IS %f.",Hogwarts->arrayOfCities[dest],t,(t\*10));

updatedBalence=currentBalence-(int)t\*10;

putBalence(userName,updatedBalence);

goto start02;

}

else{

printf("\nWRONG CHOICE PLEASE ENTER AGAIN\n");

goto check04;

}

}

break;

case 4:showHistory(userName);goto start02;

break;

case 5:clearHistory(userName);goto start02;

break;

case 6:printf("YOUR CURRENT BALENCE IS %d",getBalence(userName));goto start02;

break;

case 7:return 0;

break;

default:{printf("\nENTER CORRECT CHOICE");goto start02;}

}

return 0;

}

**RESULTS**

**PHASE I**: LOGIN/SIGNUP

Current users’ username and passwords are stored in the file named user.txt. FORMAT:%s(username)[SPACE]%s(password)\n

Text

Description automatically generated

Case I: User who wants to log in (included the case of invalid password also)

Text

Description automatically generated

Case II: User who opts for registration

Text

Description automatically generated

**PHASE II:**

The file named 34 depicts 3 as tray number at source 4(Hogwarts) and the file is queue for the tray.

Graphical user interface, text, application

Description automatically generated

Case I: Since the queue is empty, whoever books this tray will be asked to enter the tray and proceed to his journey. Once he books the tray, he will be added to the file which denotes he is travelling in that particular tray at a particular time ‘t’ and when his journey completes, he will be dequeued from the queue automatically according to time.

Text

Description automatically generated

Case II: When the queue is not empty

Graphical user interface, text

Description automatically generated with medium confidence

Let us assume that gk is travelling from 4(Hogwarts ) to 7(Forbidden Forest). At this time if another user comes he has to pick among two options either he wants to wait for the tray or pick another tray.

Option I: If the user wants to wait

Text

Description automatically generated

When gk completes his journey, the file will become empty and the waiting user will be added to the file, asked to enter the tray and his journey begins as previous case.

Text

Description automatically generated

Option II: If the user wants to select another tray.

Text

Description automatically generated

His history will be saved to a file automatically in the background along with his wallet balance “w”.

A screenshot of a computer

Description automatically generated with medium confidence

If the user opts for viewing his history:

Text

Description automatically generated

Similarly, he can also clear his travel history

Graphical user interface, text, application

Description automatically generated

He can also view his wallet balance

Text

Description automatically generated

**CONCLUSION AND FUTURE SCOPE**

People fore see a great increase in underground construction, numerical estimates are crude at best. Key factors affecting the actual increase are technological improvements reducing costs and an increasing awareness on the part of society and public-works planners of the many potential applications for better use of the underground.

In conclusion, our program is a prototype which provides user in accomplishing a user-friendly journey in reducing time and confusion.

**REFERENCES**

* <https://youtu.be/u5V_VzRrSBI>
* <https://youtu.be/sa4CuC-Zejs>
* A Review of The Literature on The Evaluation and Roles of Tunnels Technology in Managing Environmental Disaster by Nuhu Isah and Maimunah Ali, Department of Technology, Management Faculty of Technology Management and Business , Universiti Tun Hussein Onn, Malaysia.
* Tunnel Systems as a Critical Alternative in Solving Transport Problems by Vladimir V.Makarov, Far Eastern Federal University, Vladivostok 690090, Russia.
* A Survey of Road Traffic Congestion Measures towards a Sustainable and Resilient Transportation System by Tanzina Afrin and Nita Yodo, Department of Industrial and Manufacturing Engineering, North Dakota State University,1410 14th Avenue North, Fargo, ND 58102, USA.
* An Empirical Study of the Effects of Road Tunnel on Driving Performance by Alessandro Calvi, Maria Rosaria De Blasiis, Claudia Guattari, Department of Sciences of Civil Engineering, University Roma Tre, Via Vito Volterra, Rome, Italy.