

Software Requirement Specification

of

Easy hire

a cs 101 project

by

batch - 6, slot - 12

(wednesday morning) (lecture slot 11)



The project was worked on in two teams, led by Md. Tahir Patel, under the mentorship of Dileep Singh (T.A.) .

TEAM 1 :	Roll Number
• Kshitij Jayakrishnan	111030011
• Mehak Priya	111030029
• Mohak Mehta	111030026

TEAM 2:	
• Akshat Kadam	111030006
• Jaideep Sontake	111030008
• Md. Tahir Patel	111030002
• Mohit Khatri	111030014

Introduction : what inspired the project?

The basis of this project lies in simplifying the almost everyday activity of hiring a cab. After conducting a small survey amongst the team members themselves we landed up short listing the following problems which served as an inspiration to this project -

- The common man faces the very problem of not being able to locate a cab. For example a person A is towards the remoter part of the city in the non busy hours of the day. Neither does the cab driver going by any nearby region has any clue that where a customer would be and vice versa.
 - A person new to the city, say a tourist has no clue about the fare, and in many cases it so happens that the cab driver ends up overcharging.
 - In fact, Cab drivers in the real world exercise discretion in choosing their customers. In the end usually the customer becomes totally dependent on the cab driver rather than both of them depending on each other equally!
 - Often the customer tends to get anxious, that when will next available cab pass by!
-

After analyzing all these problems the following problem statement was designed.

Problem statement

The aim of this project is to design an APPLICATION SOFTWARE (can be extended to a mobile application) in which as per user's input of his position and destination (in a hypothetical city) the program allocates him the nearest cab in his region.

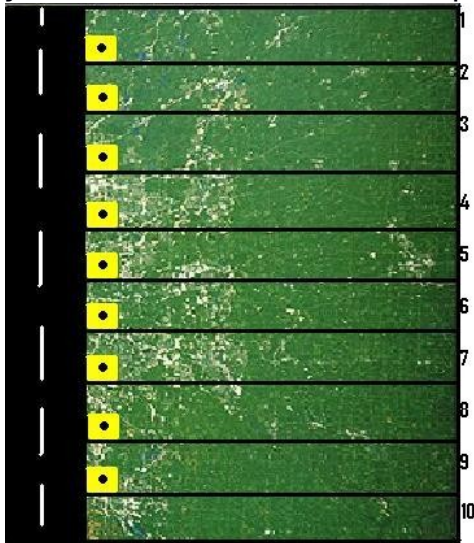
The fare is deducted from his account.

The city's components:

- ▢ Sectors: It has 10 sectors, laid out in a straight line, as shown
- ▢ Check-points: Every sector has a check -point, where a person can come and request for a cab.
- ▢ Cab: Every sector has 5 cabs initially.

The Model Map (for this project)

- yellow boxes indicate checkpoints



- green boxes indicate the sectors
- to the extreme left is the only available road for travel.

Implementation(what is the use of this project!!!)



- ▯ methodology of locating the nearest possible cab, reduces not only the anxiousness of both the customer and the cab driver but shall also reduce the communication gap between them , thus making things easy.

- ▯ It can be very easily implemented after , certain modifications especially in well planned cities like Chandigarh which has a 2 dimensional linear map.

Software requisites:

EzWindows

Linux OS

Gcc or gnu compiler

Various libraries included:

- (1) iostream
- (2) cstring
- (3) cmath
- (4) cstdlib
- (5) cstdio

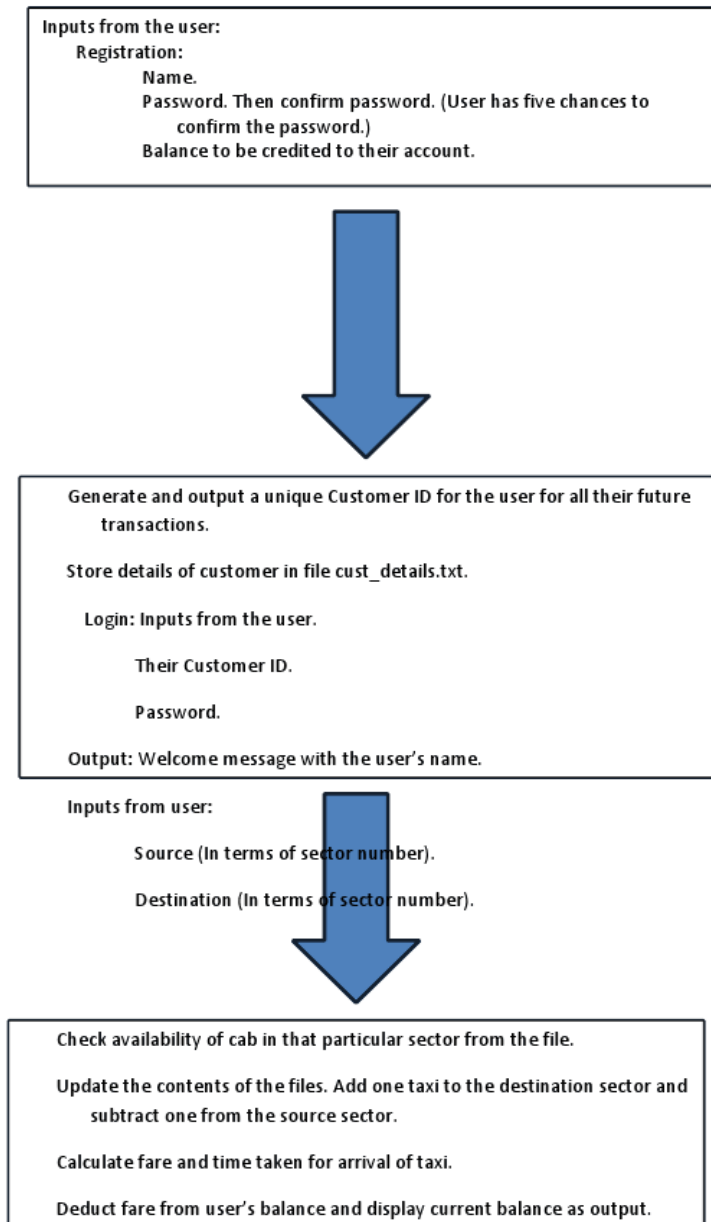
Assumptions

- 1) The above illustrated linear map has been assumed as a model for this project.
- 2) We assume that to travel between any two consecutive sectors in the city, it takes the same time.



- 3) A customer is picked up or dropped at the checkpoint. If idle the cabs will be present at the checkpoints.
- 4) We assume that picking and dropping the customer takes no time.

- 5) Taking a cab from the same check-point takes no time.
- 6) The program is not responsible, about how the user deposits money in his/her account. It is responsible only for the balance in the account.
- 7) Our program assumes a maximum of 1000 customers.



The Basic Flowchart of The Program

Inputs by the user:

1) Login/ Register page –

Inputs for Registration: Name (Username), Password, Contact Number, Balance (for the payment of fares).

Inputs for Login: Username, Password

Inputs for urgent mode: contact number

2)

3) Source, Destination (In terms of sector number)

Outputs:

1) A unique customer ID

2) Fare

3) Bill Statement.

4) Deducted balance card details.

5) Estimated time of travel

6) Time of approach(of cab)

Function prototypes:

Additional functionalities:

1) We can include traffic signals (either red or green) each of which can delay the travel for a certain fixed time.

2) we can include additional functions such booking of cabs and cancellation of cabs

3) We can get login and registration done on a dialog box.

4) Most importantly, for the security and privacy of the user, we are working on a code that shows asterisks instead of the password when the user types it.

5) Time delay can be added so that when a cab is on the , it can be shown that the cab is occupied.

Databases required:

cust_details.txt

stores data of the user viz. customer ID, login id, password, name, contact number

cab_info.txt

stores information of the starting point of the cab and stores the position of the cabs after every travel

cab_num.txt

stores information about how many cabs in each sector

Current Status of the project:

- ▯ implementation of Ez Windows is taking place and the program code has been written for the terminal as of now.
- ▯ We are enclosing the code we have written till now. However, debugging is still in the process and the program is not free of all bugs currently. The three main glitches being faced are -
 1. Cab_avail file prints +2 in the destination instead of +1.

2. There are five attempts for login. However on one incorrect attempt, even if next input is correct, login fails.
3. On the spot registration and login is giving problems.

Preliminary Code written:

```
// Batch 6 slot - 12 ( wednesday morning)
//Files included : Database.txt ; cust_details.txt ; cab_data.txt .

#include<iostream>
#include<cstdio>
#include<fstream>
#include<cmath>
#include<cstdlib>
#include<cstring>
using namespace std;

#define UNITTIME 5
#define UNITFARE 10

class cab
{
    int cab_num,sec_num;
    public:
    char get_cab();
    int get_sector();
    int check_avail(int ,int );
```

```

        int compute(int ,int ,int );
        int t_approach(int );
        float fare(int ,int ,int );
        int time_travel(int ,int ,int );
        int update(int ,int );
        //static int cab_avail[10]={5,5,5,5,5,5,5,5,5,5};
    };

//static int cab_avail[10]={5,5,5,5,5,5,5,5,5,5};

int register_acc(){
    char
    name[100],contact[100],passcode[100],passtemp[100],cust_id[80]={'1','1','5','0','0'
    ,'\0','\0'},linestr[80]={'1','1','5','0','0','0','\0'};

    int exit_count=0,i,j,return_val=0,k,l=0;

    long id_sequence;

    float balance;

    FILE *fp=fopen("cust_details.txt","r+");

    if(fp==NULL){
        cout<<"Could not open file.";
        return 1;
    }

    cout<<"\n-----Register-----"<<endl;

    cout<<"Enter your Name: ";

    cin>>name; // ends on white line
    character (space_bar);

    cout<<"Enter your Contact Number: ";

    cin>>contact;

```

```

cout<<"Enter your desired pass code: ";
cin>>passcode;
cout<<"Confirm password: ";
cin>>passtemp;
while(strcmp(passcode,passtemp)!=0){
    if(exit_count==5){
        cout<<"You have reached your limit for attempts. Program will now
exit."<<endl;
        return -1;
    }
    passtemp[0]='\0';
    cout<<"Password not confirmed. Please confirm password again: ";
    cin>>passtemp;
    if(strcmp(passcode,passtemp)==0)
        break;
    exit_count++;
}
//Reading file.
while(!feof(fp)){
    return_val=fscanf(fp,"%s",linestr);

    //return_val stores the integer that fscanf returns. To prevent reading of the last
value in the file twice we use this method.

    if(return_val==-1){
        break;
    }
}
for(i=0;i<6;i++){
    cust_id[i]=linestr[i];

```

```

    }

    id_sequence=(cust_id[0]-48)*100000+(cust_id[1]-48)*10000+(cust_id[2]-
48)*1000+(cust_id[3]-48)*100+(cust_id[4]-48)*10+(cust_id[5]-48);

    id_sequence++;
    for(j=5;j>=0;j--){
        cust_id[j]=id_sequence%10+48;
        //Adding 48 to get proper ASCII values.
        id_sequence/=10;
    }

    //Adding null character at the end of cust_id.
    cust_id[6]='\0';

    cout<<"Your Customer ID is: "<<cust_id<<". Please note it for future
reference."<<endl;

    cout<<"Enter balance: ";
    cin>>balance;

    //Writing values into file "cust_details.txt".
    fprintf(fp,"%6s,%s,%3.2f,%s,%s\n",cust_id,passcode,balance,name,contact);

    cout<<"You have been successfully registered."<<endl;

    fclose(fp);

return 0;

}

int login(){
    char
linestr[80],cust_id[80],pass_code[100],pass_file[100],custid_file[80],balance[100],na
me[100];;

    int return_val,i=0,k=0,pos,foundflag=0;

    FILE *fp=fopen("cust_details.txt","r");

```

```

if(fp==NULL){
    cout<<"Could not open file.";
    return 1;
}
cout<<"\n-----Login-----"<<endl;
here:
cout<<"Enter your Customer ID: ";
cin>>cust_id;
cout<<"Enter your password: ";
cin>>pass_code;
while(!feof(fp)){
    return_val=fscanf(fp,"%s",linestr);

    //return_val stores the integer that fscanf returns. To prevent reading of the last
    value in the file twice we use this method.

    if(return_val==-1){
        break;
    }
    for(k=0;linestr[k]!=',';k++){
        custid_file[k]=linestr[k];
    }
    custid_file[k]='\0';

    //To change position of linestr to the character after the first comma.
    k++;

    for(i=0;linestr[k]!=',';i++,k++){
        pass_file[i]=linestr[k];
    }
    pass_file[i]='\0';
    k++;
}

```



```

for(i=0;linestr[k]!='';i++,k++){
    balance[i]=linestr[k];
}
balance[i]='\0';
k++;
for(i=0;linestr[k]!='';i++,k++){
    name[i]=linestr[k];
}
name[i]='\0';
k++;
//Checking the login details entered by the user.
if((strcmp(cust_id,custid_file)==0) && (strcmp(pass_code,pass_file)==0)){
    foundflag=1;
    break;
}
}
if(foundflag==1){
    cout<<"Login Successful.\nWelcome "<<name<<". "<<endl;
}
else{
    system("clear");
    cout<<"Invalid Customer ID/Password"<<endl;
    goto here;
}
return 0;
}

```

```

int cab::check_avail(int S,int D)
{
    int i=0,sign=1,add=0,addf,cabfound=1,sec_num,num;
    FILE *fpoint;
    fpoint=fopen("cab_data.txt","r+");
    fseek(fpoint,D*4,SEEK_SET);
    fscanf(fpoint,"%d %d",&sec_num,&num);
    //num++;
    fseek(fpoint,-3,SEEK_CUR);
    fprintf(fpoint,"%d %d",sec_num,(num+1));           //need to debugg
here
    while(cabfound)
    {
        if(S+i>=0 && S+i<10)
        {
            fseek(fpoint,(S+i)*4,SEEK_SET);
            fscanf(fpoint,"%d %d",&sec_num,&num);
            if(num!=0)
            {
                //cout<<"Cab found in sector : "<<S+i<<endl;
                num--;
                fseek(fpoint,-3,SEEK_CUR);
                fprintf(fpoint,"%d %d",sec_num,num);
                cabfound=0;
                return(i);
            }
        }
        sign=sign*(-1);
    }
}

```

```

        add++;
        addf=sign*add;
        i+=addf;
    }
}

int cab::t_approach(int i)
{
    int t;
    t=fabs(i)*UNITTIME;
    return t;
}

float cab::fare(int S,int D,int flag)
{
    int dist;
    float extra=20.0;
    float fare;
    dist=fabs(D-S);
    if(flag==1)
    {
        fare=extra+(dist*UNITFARE);
        return fare;
    }
    else
    {
        fare=dist*UNITFARE;
        return fare;
    }
}

```

```

}

int cab::time_travel(int S,int D,int fare)
{
    int dist,t_est;
    dist=fabs(D-S);
    t_est=dist*UNITTIME;
    return t_est;
}

int cab::update(int snum,int D)
{
    FILE *fp;
    int i=0;
    long pos;
    int cab_no,sector;
    fp=fopen("Database.txt","r+");
    if(fp==NULL)
    {
        cout<<"Database Open Failure"<<endl;
        //return 1;
    }
    fscanf(fp,"%d %d",&cab_no,&sector);
    while(!feof(fp))
    {
        if(snum==sector)
        {
            fseek(fp,-4,SEEK_CUR);
            fprintf(fp,"%d %d",cab_no,D);

```

```

        cout<<"Cab available\nCab Number : "<<cab_no<<" In Sector
number : "<<sector<<endl;

        //cab_avail[D-1]--;

        return 1;

    }

    fscanf(fp,"%d %d",&cab_no,&sector);

}

}

```

```

int main(){

    system("clear");

    int ans;char choice;

    do{

        cout<<"1.Login\n2.Register\n3.Exit"<<endl;

        cout<<"Enter choice number: ";

        cin>>ans;

        system("clear");

        switch(ans){

            case 1: login();

                //cout<<"\n Do you want to go to the main menu? Type y/n: ";

                //cin>>choice;

                break;

            case 2: register_acc();

                cout<<"\n Do you want to go to the main menu? Type y/n: ";

                cin>>choice;

                break;

            case 3: return 1;

            default:cout<<"Invalid choice."<<endl;

        }

    }while(choice=='y' || choice=='Y');

}

```

```

    }
} while(choice=='y' || choice=='Y');
    if(choice=='n')
    {
        cout<<"Thank You"<<endl;
        return 1;
    }
    cab c;
    int S,D,flag,z;
    float f;
    cout<<"Enter Source : ";
    cin>>S;
    cout<<endl;
    cout<<"Enter Destination : ";
    cin>>D;
    z=S+c.check_avail(S,D);
    f=c.fare(S,D,0);
    cout<<"Time of Approach for Cab is : "<<c.t_approach(z-S);
    cout<<"\nFare is : "<<f<<endl;
    c.update(z,D);
return 0;
}

```

DIARY

1) Pre-meet

Agenda-everybody directed to think about topics for the project and come up with the layout about its execution

Date-25th September

Time duration- individual time spent varied from 1 hr 30 minutes

2)meet 2(lab)

September 28

time : 9:30am to 10:30am

agenda-

- each team member to speak on his idea for 5 minutes followed by an analysis of his/her idea as well as a viability check on it.
- After all the reviews have been done ,chose two best topics. after preparing their layout , the more feasible and interesting topic of the two to be chosen .
- Chose the leader of the team

outcome-

- topic of the project decided
- time and agenda of the next meet decided.
- Everyone told to write the layout of the project as he/she perceives it and speak on it in the next meet.
- Md. Tahir Patel unanimously chosen the leader.

2) Meet 3

September 29

Time -10:30am to 11:30am

Agenda:

- Each team member to speak on his/her layout perception and execution of the project
- Decide on the most favourable layout

Outcome:

- Various aspects and approaches to the topic discussed
- Each team member directed to prepare a sample algorithm of the program

4)meet 4

October 4

Time: 10:25 am to 11:30am

- Discussion of modules and tentative division of teams.

5)meet 5(lab)

October 5

Time: 8:30 to 10:30 am

Lab meeting

- Sub teams finalized.
- Discussion of abstract and project documentation.
- Sub Team1: Kshitij Jayakrishnan, Mehak Priya, Mohak Mehta. To deal with Registration of user, login of user and getting other inputs.
- Sub Team 2: Tahir Patel, Akshat Kadam, Mohit Khatri, Jaideep Sontakke. To deal with updation of available cabs at various sectors.

6)meet6

October 6

Time 10:20pm to 11:30pm

Meeting with TA.

- Project topic discussed, limitations observed and problem statement modified accordingly.

7)meet 7

October 9

Time 11:30 am to 1 pm

Agenda:

- Team 1's algorithm to be discussed

outcome

- algorithm discussed and tentatively finalised
- team directed to start with sample coding

8)meet 8

October 10

Time -10:15 am to 11:30 am(entire team)

6pm to 8pm(team 1)

Agenda:

- Team 1 to start writing the code on its part of the program
- Team 1 to check the code it has written in the OSL.
- Team 2 to discuss its algorithm

Outcome:

- Team 1 wrote the entire flowchart and layout of the code.
- Team 1 also wrote a sample code to be checked in the OSL.
- Team 1 wrote the function register_acc()
- Team 2 wrote the algorithm partially

9)meet 9

October 11

time: 10pm to 12pm

agenda-

- team 1 to check their code.
- Team 2 had to finish their algorithm

Outcome

- ▯ the primary code of team 1 did not work on the compiler of a personal laptop. Plan to meet in OSL the next day to test the code.

10)meet 10(lab)

October 12

Time: 8:30am to 10:30am.

Venue: OSL

Agenda-

- ▯ Team 1 to test their code.
- ▯ Team 2 to start with their code.

Outcome:

- ▯ Objectives reached.

11)meet11

October 16

Time: 3pm to 5pm(team 1)

Venue: OSL

Agenda:

- ▯ Team 1 to write login() function and main function to link register and login functions.

Outcomes:

- ▯ Code written and tested.

Time 11pm to 1am(team 2)

Venue: Hostel 1 Computer Room.

Agenda:

- ▯ Team 2 to write their code for computation and updation of Database.txt for the availability of cabs.

Outcome:

- ▯ Code written and tested.

October 17

Time:8:30 am to 10 am

Venue: Hostel 1 Computer Room.

Agenda:

- ▯ All team meet for merging of code of both teams.

Outcome:

- ▯ Code merged but some errors found. Errors to be rectified in another meet

INDIVIDUAL TIME SPENT

Mehak priya:

- ▯ read the project documentations of 2010 projects(2hrs)
- ▯ typed the project documentation and diary . (4hrs)
- ▯ read about files(1 hr)
- ▯ discussed abstract with Akshat kadam(1.5 hrs)
- ▯ Evaluating the Code and the SRS document.(5hours 20mins + still in the osl lab!!! ++ still in the h10 comp lab)
- ▯ debugging , conversion of SRS document and uploading! (1 hour)

Kshitij Jayakrishnan

- ▯ read project documentations of 2010 projects (1 hour 15 mins)
- ▯ read about files and programs on files(1hr 30 minutes)
- ▯ Extra time spent on code(3hrs)

- ▯ worked on the flowchart for the SRS documentation. (2hours)

Akshat Kadam

- ▯ read project documentations(1 hr 30 minutes)
- ▯ discussed abstract with Mehak Priya(1 hr 30 minutes)
- ▯ read files and programs on it (2 hr)
- ▯ extra time on code (2 hrs 30 minutes)
- ▯ debugging the code (3hours)

Tahir Patel

- ▯ extra time on codes,logic, algorithms(1hr 30 minutes)
- ▯ read files and programs on it(1hr)
- ▯ worked on the making the algorithm and the module(1hour and 30 mins)
- ▯ made the map for the SRS documentation(1 hour 30 mins)
- ▯ worked on debugging (30mins)

Mohak mehta

- ▯ worked on the idea of the project - easy hire(2 hours 15 mins)
- ▯ designed the basic layout of the project (45 minutes)
- ▯ read on functions and cstring library , files and classes from tata mcgrawhill, c++ reference and cohoon. (3hours)

- ▯ worked on developing the algorithm and the flowchart for the code.(40 mins)
- ▯ assisted kshitij in finding errors.
- ▯ Worked in debugging (40 mins)
- ▯ Worked on the SRS documentation, googled images , worked on the implementation and wrote the SRS along with Mehak and edited it. (6hours 15mins + (still editing!!!))
- ▯ conversion of SRS document and uploading of doc. (30 mins)