The Lottery System

Sprint Implementation

Project Timeline: 24.8.2022 to 29.8.2022

INDEX

1.Introduction:

ZamoLand Development Authority (ZDA) plans to allot 100 plots to people of the city through a lottery. A token will be available on their website on first-cum-first-serve basis. Only 300 tokens are available. It will have a serial no which is in a pre-decided range. Once the timing is announced they have to grab the tokens by using their unique family id and they get an auto-generated confirmation regarding their participation in the lottery process. On every call a token number is auto-generated out of a list of 300 available tokens. The participant who owns the token will be notified. He/She will have to confirm booking within 5 minutes by paying Rs. 50000 as booking amount. Once done the plot is allotted in his/her name. If the winning participant does not confirm booking then automatically there is a repeating lottery for the same plot. Once a plot is booked its details are updated in the plots database. Another database is maintained which will have the participant details as well as the plot details for the plot he won.

2.Scope:

This project aims to create the development of an automated system of Lottery Process to assist ZDA Authority in storing and retrieving all the information about the registered participants and the winning participants in a way more robust and efficient manner. All the information about a particular participant is stored in a retrievable manner.

3.Purpose:

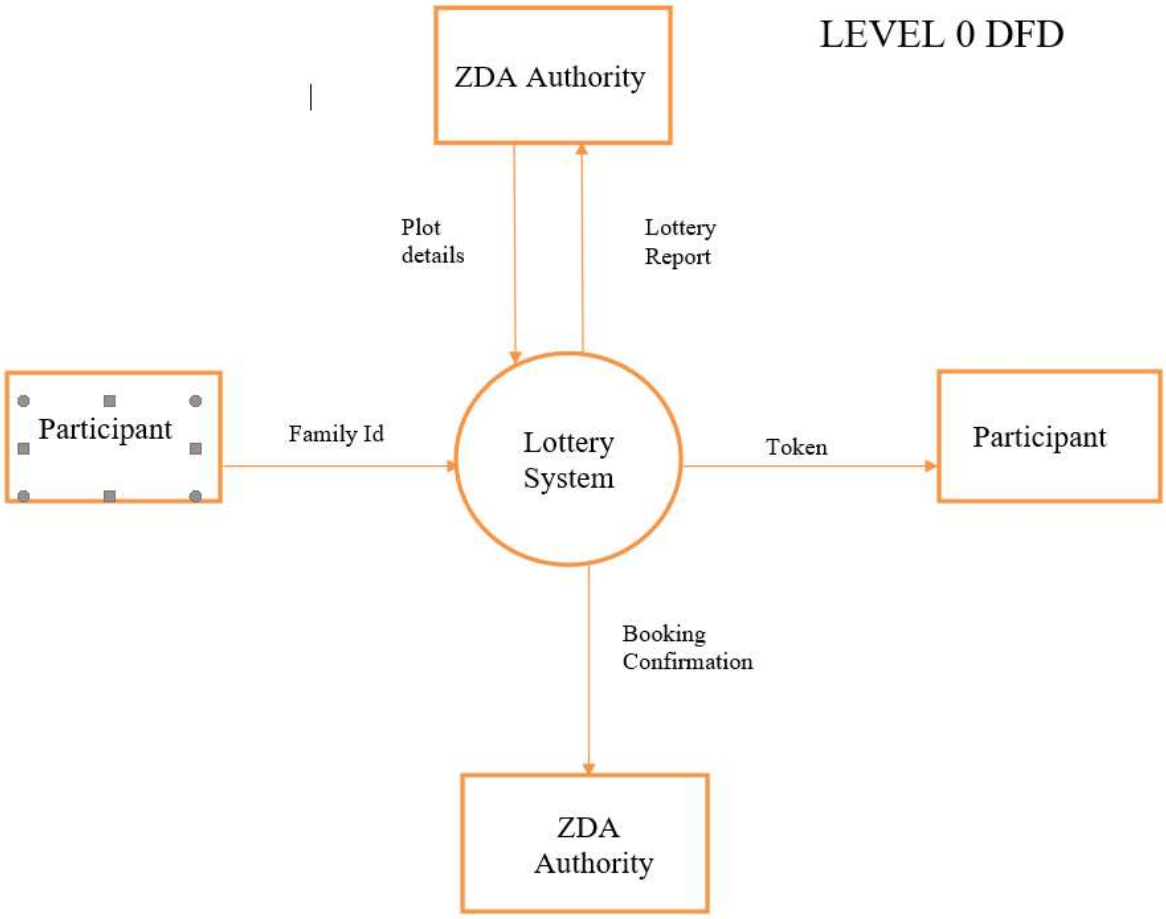
The purpose of this document is to describe the requirements to track all information of a participant allotted by the lottery process system and keep an organized details of all participants. The main purpose of Lottery System is to allot the plot to the people of the city through lottery.

4. Intended Audience:

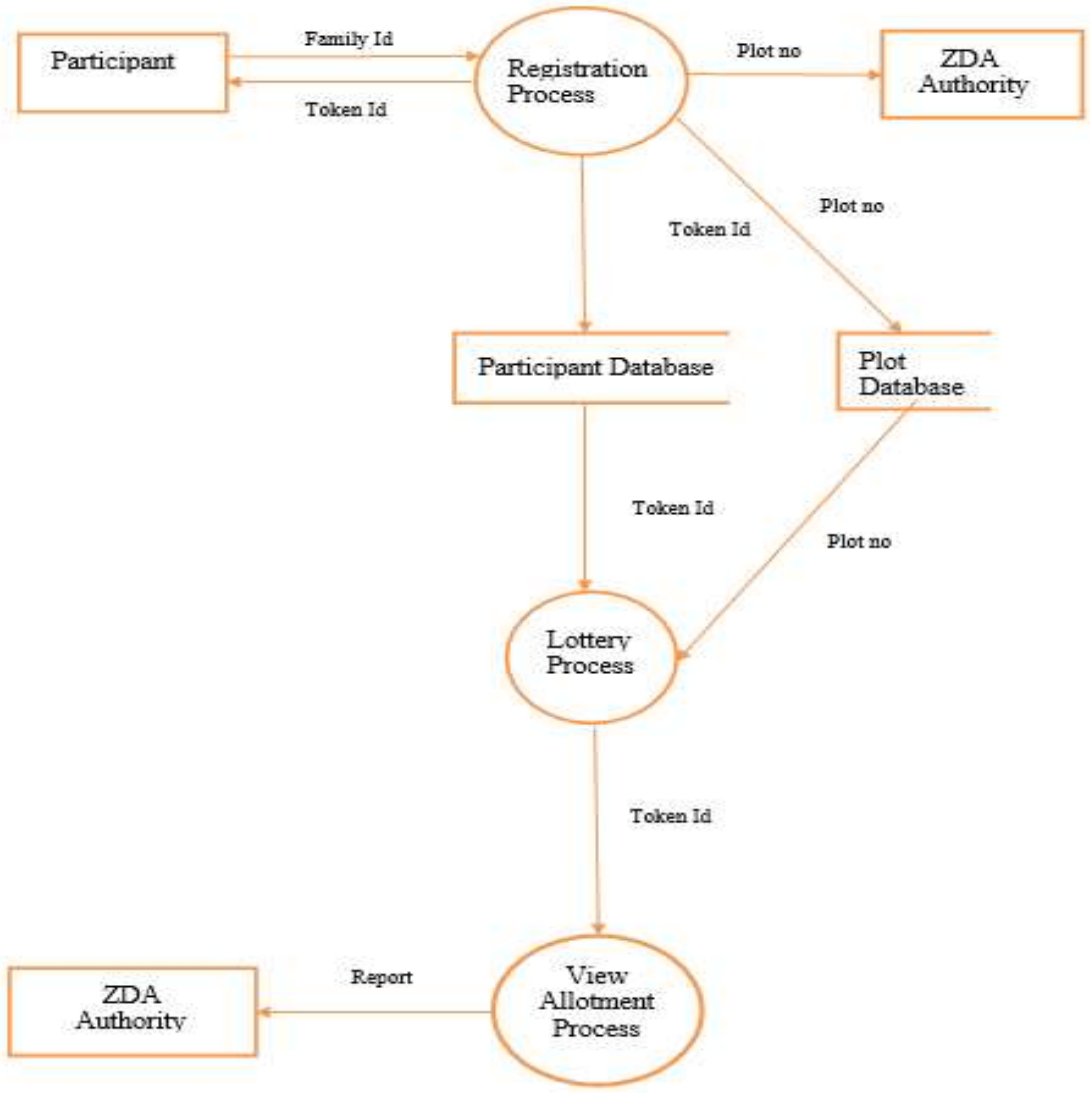
The target audience are the families of the ZamboLand who are participating in the Lottery process. Out of 500 households only 300 families are allowed to grab the token out of which only 100 families will be benifitted.

5.Design Overview:

5.1. Data Flow Diagram:



LEVEL 1 DFD


Participant
Family Id
Token Id
Registration Process
Plot no
ZDA Authority
Token Id
Plot no
Participant Database
Plot Database
Token Id
Plot no
Lottery Process
Token Id
ZDA Authority
Report
View Allotment Process

5.4. Flowcharts:

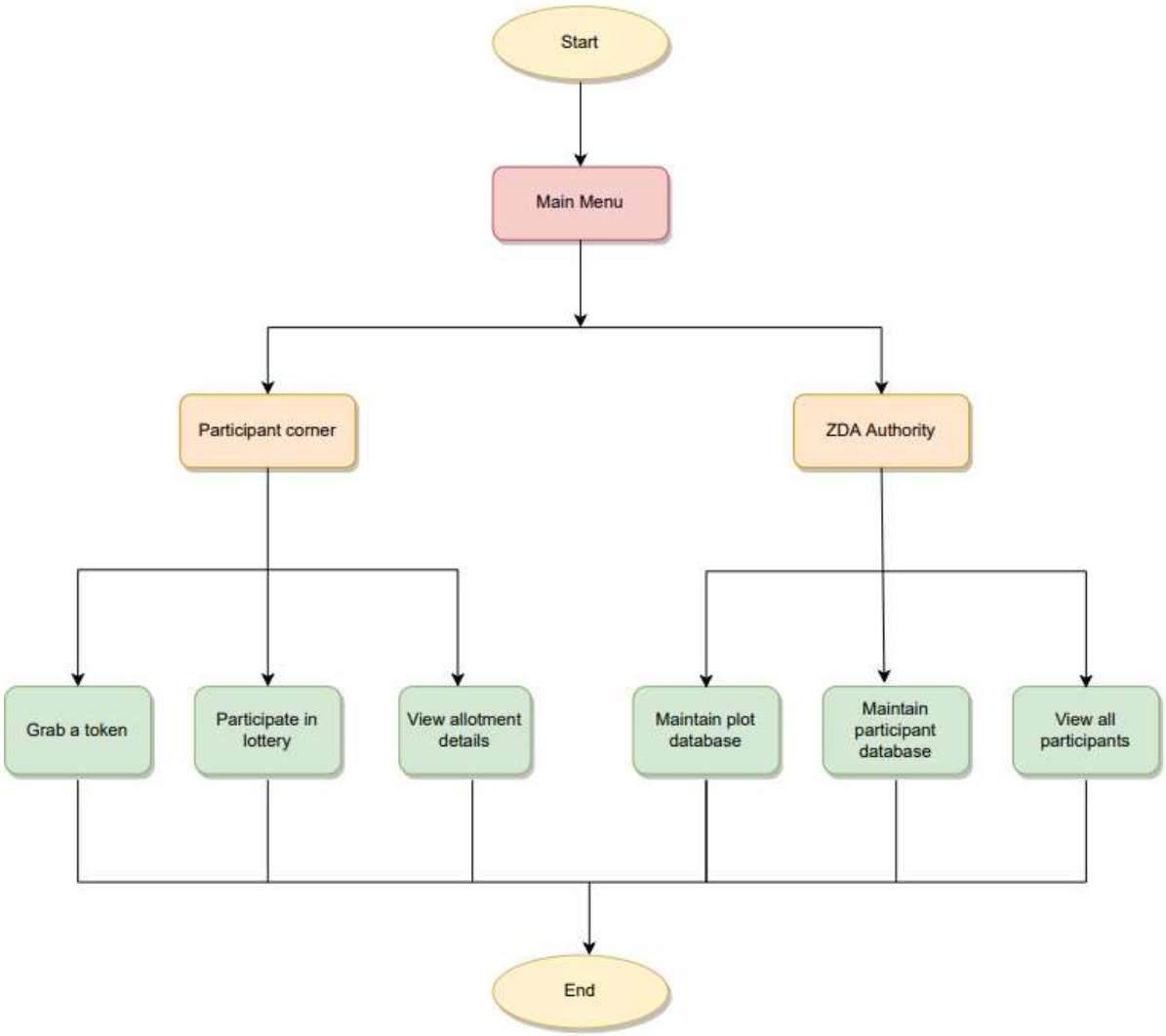


Fig.5.4.1: Flowchart of Main Menu

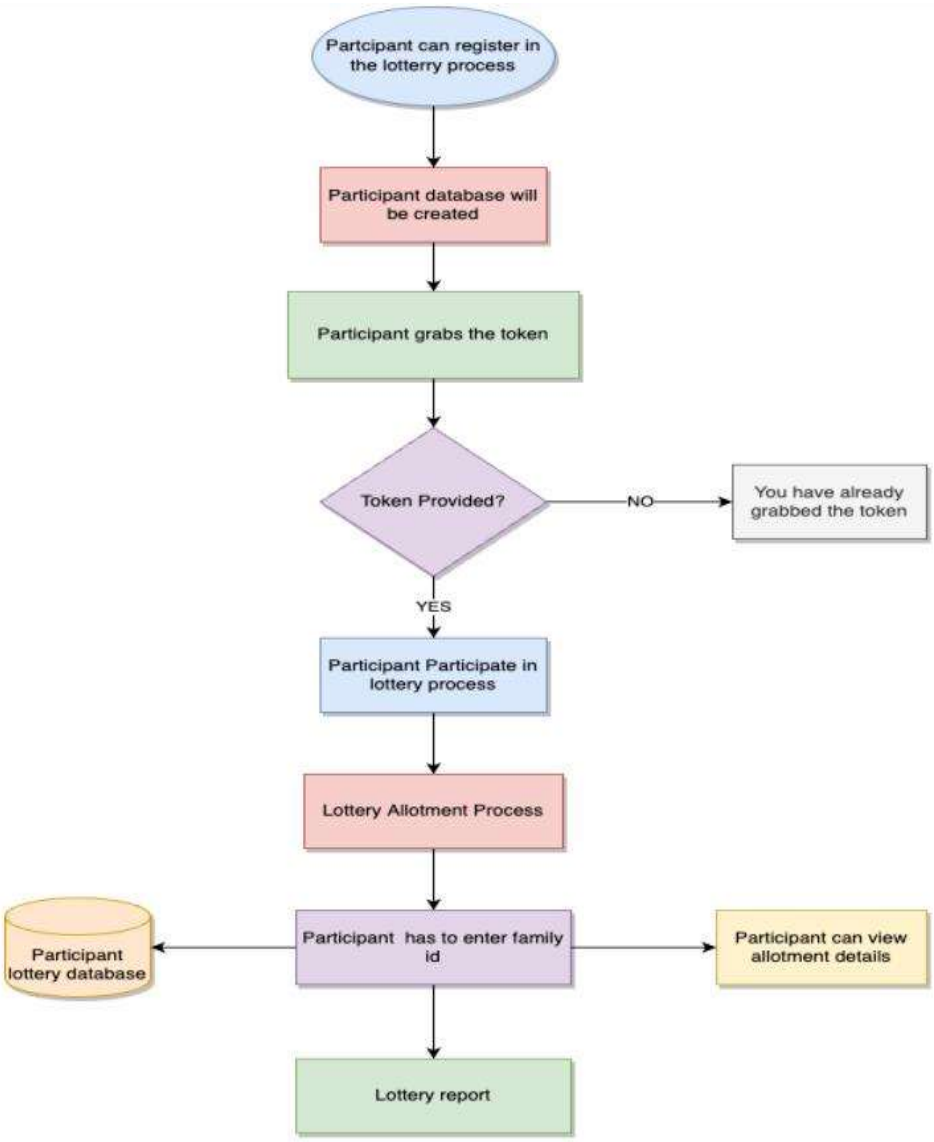


Fig.5.4.2: Flowchart of Participant Corner

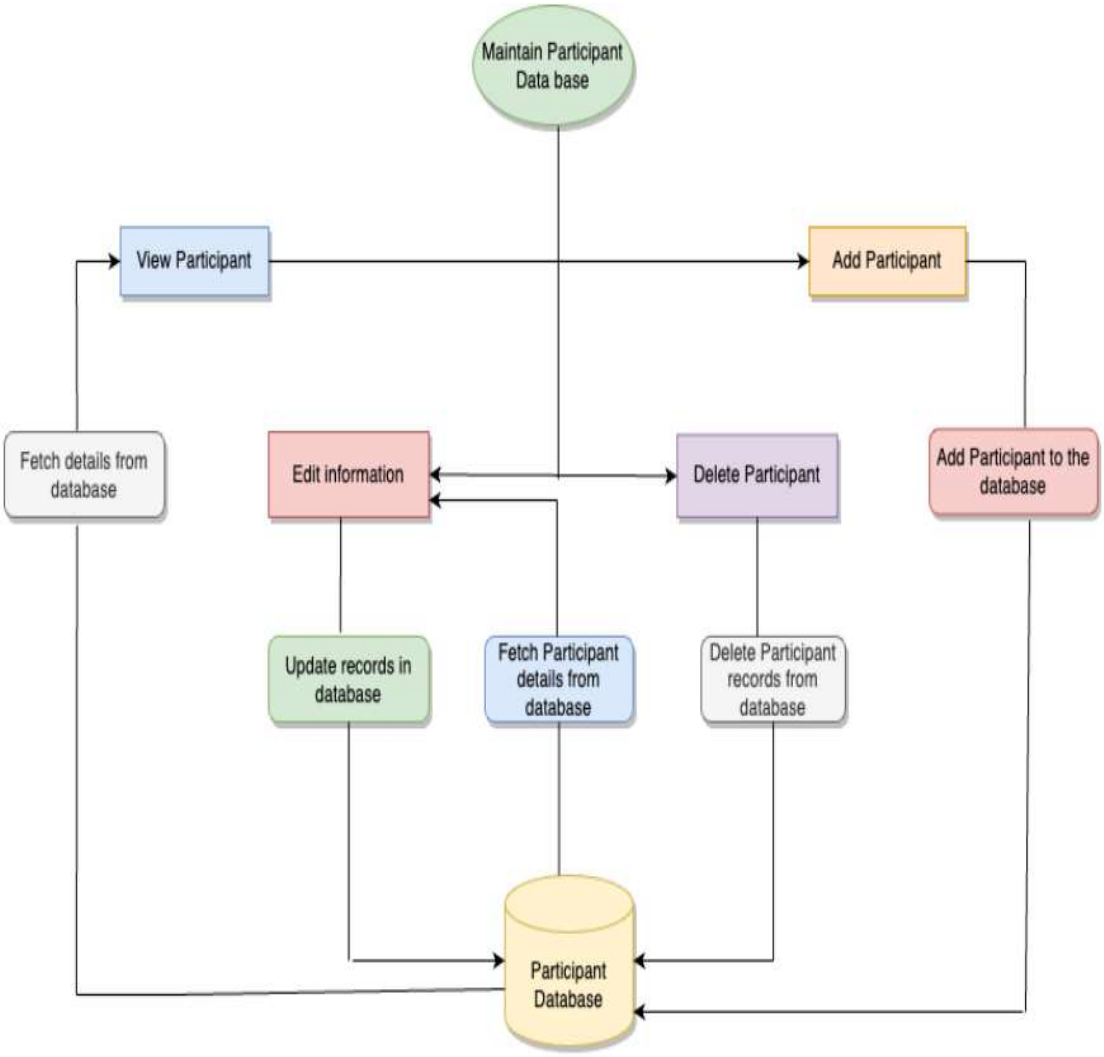


Fig.5.4.3: Flowchart of Maintain Participant Database

6. System Architecture:

6. System Architecture:

6.1.Functions:

6.1.1.Participant Corner: Out of 500 households from the city only 300 participants are allowed to grab a token which is available on the website.2 tokens cannot be issued to a single family. So now the participant can register by entering their unique family id and can view the allotment details through the auto generated confirmation message.

6.1.2.ZDA Authority: ZDA Authority maintains plot database by adding, editing, deleting and viewing plot details. It also maintains participant database by adding, editing, removing and viewing all participants database. They can also view the Lottery report. ZDA authorities have to supply a password to access the admin features online

6.1.2.1.Add Plot Details: ZDA Authority adds plot details into database like plot no, Size (in sq. ft.), price of plot, etc.

6.1.2.2.Edit Plot Details: ZDA Authority can edit the plot details and modify the database as and when required.

6.1.2.3.Delete Plot Details: ZDA Authority can delete the plot details from the existing database.

6.1.2.4.View Plot Details: ZDA Authority can view the plot details.

6.1.2.5.Add Participant Details: ZDA Authority adds participant details into database like participant name, slno. (Auto-generated), family id, token no. Plot no., Amount to be paid. The last two fields are automatically updated after allotment.

6.1.2.6.Edit Participant Details: ZDA Authority can edit the participant details and modify the participant database as and when required.

6.1.2.7.Remove Participant Details: ZDA Authority can remove the participant details from the existing database.

6.1.2.8.View Participant Details: ZDA Authority can view all the participant details.

6.1.3.View Lottery Report: ZDA Authority can view the lottery report.

6.2. Structures :

6.2.1.Participant:

This structure is used to group participant type structure members like

-Family_id

-name

-participated in lottery

-plot_no

-token_no

-size

-remaining_amount

6.2.2.Plot

This structure is used to group plot type structure members like

-plot_no

-size of plot

-allot

-price of plot

6.2.3.Queue

This structure is used to group queue type structure members like

-token

-self referencing pointer next

7. Design Review Checklist:

7. Design Review Checklist:

Group name	Group 04
Project Name	The Lottery System
Project Manager	
Document Name	Design review checklist
Inspected on	30.08.2022
Inspected by	

8. Code Inspection Log:

8. Code Inspection Log:

8.1. Code Review Checklist:

Name of Originator	
Name of Inspector	
Name of moderator	
Path to design document	
Path to source code files	
Name of the review module/ object (if applicable)	

8.2. Code Review Log:

Project	
Artifact Name	
Artifact Type	
Author	
Reviewer	
Prepration Time (mins)	
No.of Pages Reviewed	

Filled in by Reviewer
Rating Codes *(Filled by inspector)*

Filled in by Reviewer	After meeting Dispostion
Phase-Injected Codes *(Filled in by inspector)*	**Disposition Codes** *(Filled in by inspector)*

#	Location	Rating	Phase-injected	After-meeting defect disposition	Defect description	Disposition Comments	Defect type/Root cause	Phase-detected

9. Tools Report:

9. Tools Report:

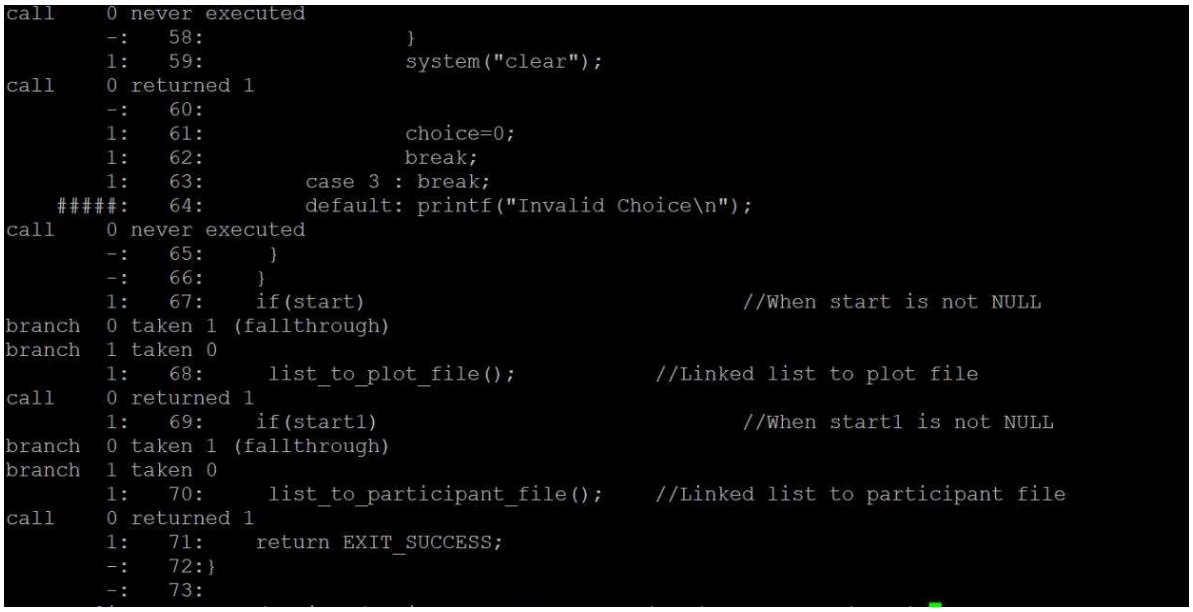
9.1. gcov report:

```
cguser6@instance-1:~/project/Project_Lottery_System/CUT/ToolsReport/gcov$ cat gcov_report
        -:    0:Source:main.c
        -:    0:Graph:main.gcno
        -:    0:Data:main.gcda
        -:    0:Runs:1
        -:    1:/*****************************************************************************
***
        -:    2: *
        -:    3:
        -:    4:    FILENAME: main.c
        -:    5:    DESCRIPTION:This file is used to display the main menu to the user.
        -:    6:    REVISION HISTORY
        -:    7:    DATE            NAME              REASON
        -:    8:    ------------------------------------------------------------------
        -:    9:    25/8/22        USername          This is done for creationof menu
        -:   10:
        -:   11: * ****************************************************************************
***/
        -:   12:
        -:   13:#include <stdio.h>       //Including required Header files
        -:   14:#include <stdlib.h>      //Includes standard libraries
        -:   15:#include <string.h>      //Includes string functions
        -:   16:#include <ctype.h>       //Includes functions of ctype
        -:   17:#include "functions.h"   //Header file
        -:   18:#include "func1.c"       //Including func1.c file
        -:   19:#include "func2.c"       //Including func2.c file
```

```
        -:   19:#include "func2.c"       //Including func2.c file
        -:   20:#include "func3.c"       //Including func3.c file
        -:   21:#define MAXPW 32     //Define Macro
function main called 1 returned 100% blocks executed 93%
        1:   22:int main()
        -:   23:{
        1:   24:    char pw[MAXPW] = {0};
        1:   25:    char *pass = pw;
        1:   26:    FILE *fp = stdin;
        -:   27:    ssize_t nchr;
        1:   28:    int choice=0;
        1:   29:    start=new=ptr=prev=NULL;              //initializing pointer for plot struc
ture.
        1:   30:    start1=new1=ptr1=prev1=NULL;          //initializing pointer for participan
t structure
        1:   31:    plot_file_to_list();                  //brings data from plot file to linke
d list
call    0 returned 1
        1:   32:    participant_file_to_list();           //brings data from participant file t
o linked list
call    0 returned 1
        4:   33:    while(choice!=3)
branch  0 taken 3
branch  1 taken 1 (fallthrough)
        -:   34:    {
        -:   35:     //Displaying Main Menu
```

```
        -:   35:     //Displaying Main Menu
        3:   36:     printf("\n         Main Menu");
call    0 returned 3
        3:   37:     printf("\n-----------------------------\n");
call    0 returned 3
        3:   38:     printf("\n1.Participant Corner\n2.ZDA Authority\n3.Exit");
call    0 returned 3
        -:   39:
        3:   40:     printf("\nEnter choice: "); //Taking users choice
call    0 returned 3
        3:   41:     scanf("%d",&choice);
call    0 returned 3
        3:   42:     switch(choice)
branch  0 taken 1
branch  1 taken 1
branch  2 taken 1
branch  3 taken 0
        -:   43:     {
        1:   44:        case 1: Participant_Corner();           //Function call for Participant_
Corner
call    0 returned 1
        1:   45:                choice=0;
        1:   46:                break;
        1:   47:        case 2: nchr = getpasswd (&pass, MAXPW, 0, fp);
call    0 returned 1
        1:   48:                    printf ("Enter password: ");
```

```
call    0 returned 1
        1:   48:                    printf ("Enter password: ");
call    0 returned 1
        1:   49:                nchr = getpasswd (&pass, MAXPW,0 , fp);
call    0 returned 1
        1:   50:                if((strcmp(pass,"ABC@123"))==0)
branch  0 taken 1 (fallthrough)
branch  1 taken 0
        -:   51:                {
        1:   52:                        system("clear");          //clears the screen
call    0 returned 1
        1:   53:                        ZDA_Authority();         //Function call for ZDA_Authori
ty
call    0 returned 1
        -:   54:                }
        -:   55:                else
        -:   56:                {
    #####:   57:                        printf("You have entered incorrect password");
call    0 never executed
        -:   58:                }
        1:   59:                system("clear");
call    0 returned 1
        -:   60:
        1:   61:                choice=0;
        1:   62:                break;
        1:   63:        case 3 : break;
```


call 0 never executed
-: 58: }
1: 59: system("clear");
call 0 returned 1
-: 60:
1: 61: choice=0;
1: 62: break;
1: 63: case 3 : break;
#####: 64: default: printf("Invalid Choice\n");
call 0 never executed
-: 65: }
-: 66: }
1: 67: if(start) //When start is not NULL
branch 0 taken 1 (fallthrough)
branch 1 taken 0
1: 68: list_to_plot_file(); //Linked list to plot file
call 0 returned 1
1: 69: if(start1) //When start1 is not NULL
branch 0 taken 1 (fallthrough)
branch 1 taken 0
1: 70: list_to_participant_file(); //Linked list to participant file
call 0 returned 1
1: 71: return EXIT_SUCCESS;
-: 72:}
-: 73:

9.2. Splint report:

```
func2.c:383:22: Function returns with global ptr referencing released storage
   func2.c:381:8: Storage ptr released
func2.c:383:22: Function returns with global prev referencing released storage
   func2.c:381:8: Storage prev released
func2.c: (in function view_plot_details)
func2.c:411:22: Only storage ptr->next assigned to unqualified (in post loop
                  increment): ptr = ptr->next
func2.c: (in function add_participant)
func2.c:435:23: Function returns with non-null global new1 referencing null
                  storage
   func2.c:431:7: Storage new1 may become null
func2.c:437:8: Test expression for while not boolean, type int: 1
func2.c:440:3: Return value (type int) ignored: scanf("%d", &new...
func2.c:448:21: Left operand of && is non-boolean (p *):
                  (ptr1) && ptr1->family_id != new1->family_id
func2.c:448:65: Only storage ptr1->next assigned to unqualified (in post loop
                  increment): ptr1 = ptr1->next
func2.c:449:8: Left operand of && is non-boolean (p *):
                 (ptr1) && ptr1->family_id == new1->family_id
func2.c:464:8: Test expression for while not boolean, type int: 1
func2.c:467:3: Return value (type int) ignored: scanf(" %[^\n]s"...
func2.c:468:11: Variable len initialized to type size_t, expects int:
                  strlen(new1->name)
  To allow arbitrary integral types to match any integral type, use
  +matchanyintegral.
```

```
                increment): ptr1 = ptr1->next
nc1.c:196:16: Left operand of && is non-boolean (pt *):
                (ptr) && ptr->plot_no != lottery_no
nc1.c:196:59: Only storage ptr->next assigned to unqualified (in post loop
                increment): ptr = ptr->next
nc1.c:197:5: Left operand of && is non-boolean (pt *):
               (ptr) && ptr->plot_no == lottery_no
nc1.c:210:23: Function returns with non-null global ptr referencing null
                storage
A global variable does not satisfy its annotations when control is
transferred. (Use -globstate to inhibit warning)
 func1.c:196:63: Storage ptr becomes null
nc1.c: (in function view_allotment_details)
nc1.c:233:2: Return value (type int) ignored: scanf("%d", &code)
nc1.c:234:18: Left operand of && is non-boolean (p *):
                (ptr1) && ptr1->family_id != code
nc1.c:234:51: Only storage ptr1->next assigned to unqualified (in post loop
                increment): ptr1 = ptr1->next
nc2.c: (in function ZDA_Authority)
nc2.c:44:3: Return value (type int) ignored: scanf("%d", &cho...
nc2.c:48:12: Return value (type int) ignored: maintain_plot_db()
nc2.c:51:12: Return value (type int) ignored: maintain_partici...
nc2.c:54:12: Return value (type int) ignored: initialize_queue()
nc2.c:57:12: Return value (type int) ignored: view_lottery_rep...
nc2.c: (in function maintain_participant_db)
nc2.c:88:3: Return value (type int) ignored: scanf("%d", &ch)
```

9.3. Valgrind report:

```
algrind_report
==240913== Memcheck, a memory error detector
==240913== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==240913== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==240913== Command: ./a.out
==240913==
==240913==
==240913== HEAP SUMMARY:
==240913==     in use at exit: 1,616 bytes in 14 blocks
==240913==   total heap usage: 24 allocs, 10 frees, 21,104 bytes allocated
==240913==
==240913== LEAK SUMMARY:
==240913==    definitely lost: 0 bytes in 0 blocks
==240913==    indirectly lost: 0 bytes in 0 blocks
==240913==      possibly lost: 0 bytes in 0 blocks
==240913==    still reachable: 1,616 bytes in 14 blocks
==240913==         suppressed: 0 bytes in 0 blocks
==240913== Reachable blocks (those to which a pointer was found) are not shown.
==240913== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==240913==
==240913== For lists of detected and suppressed errors, rerun with: -s
==240913== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cguser14@instance-1:/home/cguser6/project/Project_Lottery_System/CUT/Code/SRC$
```

```
cguser14@instance-1:/home/cguser6/project/Project_Lottery_System/CUT/Code/SRC$ cat
algrind_report_with_error
==242407== Memcheck, a memory error detector
==242407== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==242407== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==242407== Command: a.out
==242407==
==242407==
==242407== HEAP SUMMARY:
==242407==     in use at exit: 1,456 bytes in 12 blocks
==242407==   total heap usage: 22 allocs, 10 frees, 20,944 bytes allocated
==242407==
==242407== LEAK SUMMARY:
==242407==    definitely lost: 80 bytes in 1 blocks
==242407==    indirectly lost: 0 bytes in 0 blocks
==242407==      possibly lost: 0 bytes in 0 blocks
==242407==    still reachable: 1,376 bytes in 11 blocks
==242407==         suppressed: 0 bytes in 0 blocks
==242407== Rerun with --leak-check=full to see details of leaked memory
==242407==
==242407== For lists of detected and suppressed errors, rerun with: -s
==242407== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cguser14@instance-1:/home/cguser6/project/Project_Lottery_System/CUT/Code/SRC$
```

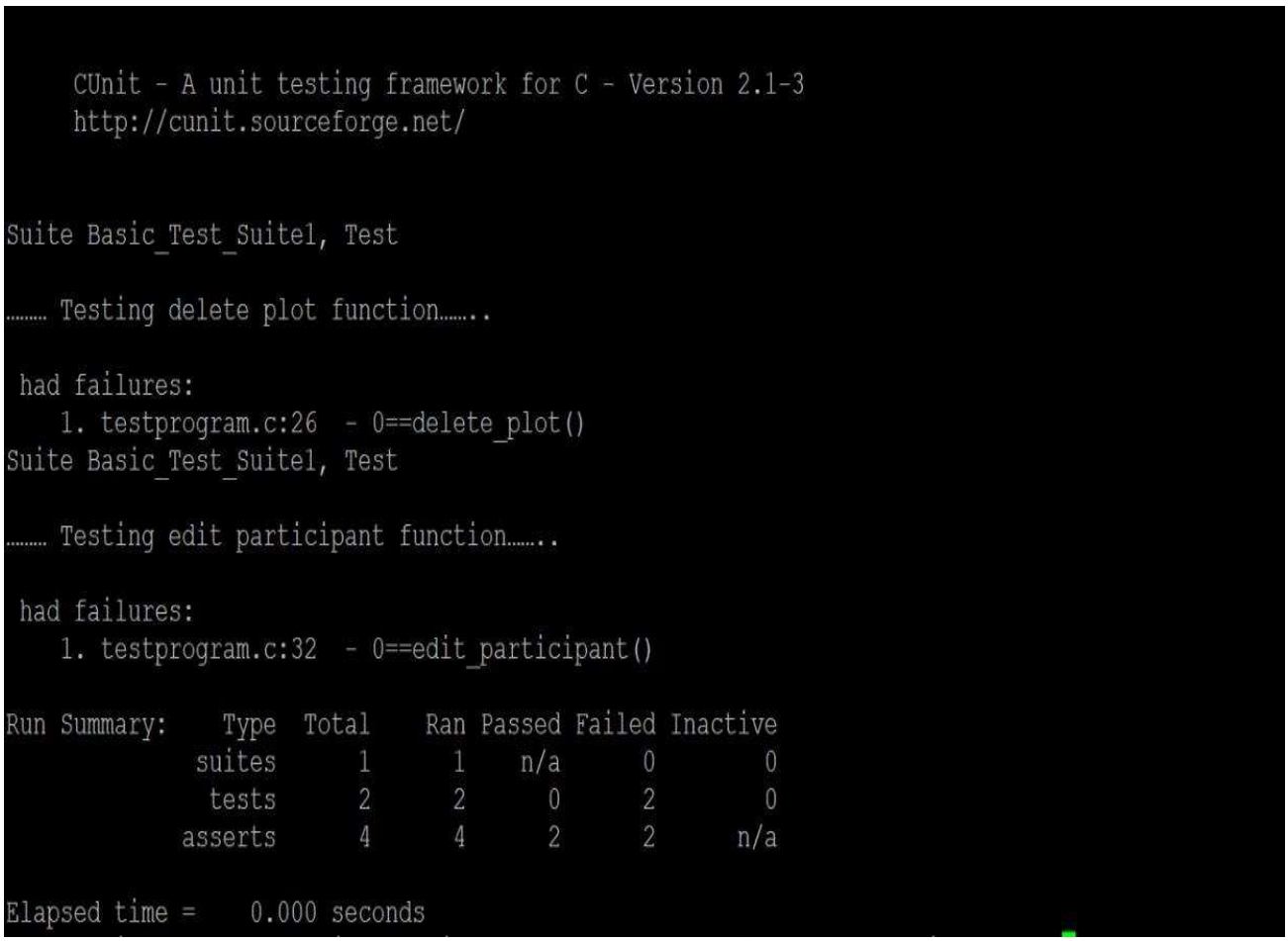
9.4. gprof report:

```
cguser6@instance-1:~/project/Project_Lottery_System/CUT/Code/SRC$ cat analysis.out
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00        1     0.00     0.00  list_to_participant_file
  0.00      0.00     0.00        1     0.00     0.00  list_to_plot_file
  0.00      0.00     0.00        1     0.00     0.00  participant_file_to_list
  0.00      0.00     0.00        1     0.00     0.00  plot_file_to_list

                     Call graph


granularity: each sample hit covers 2 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00       1/1           main [23]
[1]      0.0    0.00    0.00       1         list_to_participant_file [1]
-----------------------------------------------
                0.00    0.00       1/1           main [23]

index % time    self  children    called     name
                0.00    0.00       1/1           main [23]
[1]      0.0    0.00    0.00       1         list_to_participant_file [1]
-----------------------------------------------
                0.00    0.00       1/1           main [23]
[2]      0.0    0.00    0.00       1         list_to_plot_file [2]
-----------------------------------------------
                0.00    0.00       1/1           main [23]
[3]      0.0    0.00    0.00       1         participant_file_to_list [3]
-----------------------------------------------
                0.00    0.00       1/1           main [23]
[4]      0.0    0.00    0.00       1         plot_file_to_list [4]
-----------------------------------------------

Index by function name

   [1] list_to_participant_file [3] participant_file_to_list
   [2] list_to_plot_file       [4] plot_file_to_list
cguser6@instance-1:~/project/Project_Lottery_System/CUT/Code/SRC$
```

10. Testing Report:

10. Testing Report:

10.1. Unit testing Report:



10.2. Integration Testing Report:

Participant corner module

Case 1: Grab a token

```
Your Token number is 1


              Participant Corner
-------------------------------------------------
1.Grab a Token
2.Cancel Token
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice: 1
Enter Family Id: 1002

Your Token number is 2
```

Case 2: Participate in lottery

```
Your Token number is 5


              Participant Corner
-------------------------------------------------
1.Grab a Token
2.Cancel Token
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice: 3
Enter Family Id: 1001

Unfortunately you have not won better luck next time
Winning Lottery number is 110
```

Case 3: Participated in lottery and won

```
Unfortunately you have not won better luck next time
Winning Lottery number is 110


              Participant Corner
-----------------------------------------------
1.Grab a Token
2.Cancel Token
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice: 3
Enter Family Id: 1002

Congratulations You have won a lottery


              Participant Corner
-----------------------------------------------
1.Grab a Token
2.Cancel Token
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice:
```

Case 4: Viewing allotment details

```
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice: 4
Enter Family Id: 1001
                                  ALLOTMENT DETAILS
-------------------------------------------------------------------------------------------
---------------------------------

Serial_No     Name    Family_id   Token    Plot_no    Plot_size     Remaining_Amount
 Participated_in_Lottery
-------------------------------------------------------------------------------------------
---------------------------------

    1     Pradnya           1001         1        0        0          0.00    YES


              Participant Corner
-----------------------------------------------
1.Grab a Token
2.Cancel Token
3.Participate in Lottery
4.View Allotment Details
5.Back to Main Menu
Enter your choice:
```

11. Requirement Traceability Matrix (RTM):

11. Requirement Traceability Matrix (RTM):

Requirement	Design Mapping	Code Mapping	UT Mapping	IT Mapping
LS_01	a	participant_ corner		IT_1
LS_02	b	ZDA_ Authority		
LS_03	c	grab_token		IT_2
LS_04	d	participate_in_lottery		IT_3
LS_05	e	View_allotment_details		IT_4
LS_06	f	add_plot		
LS_07	g	edit_plot	Test_case_1	
LS_08	h	delete_plot	Test_case_2	
LS_09	i	view_plot_details		
LS_10	j	add_participant		
LS_11	k	edit_participant	Test_case_3	
LS_12	l	remove_participant	Test_case_4	
LS_13	m	view_participant_details		
LS_14	n	view_lottery_report		
LS_15	o	maintain_participant_db		
LS_16	p	maintain_plot_db		
LS_17	q	plot_file_to_list		
LS_18	r	list_to_plot_file		

12. Minutes of the meeting: