PROJECT

TOPIC:

Designing a game ("snake game") using different programming languages

1. HTML & java script

2. C language

3. Net beans

K.SAMATHA

ABSTARCT:

Abstract for the Snake game :

Snake game is one of the game that is implemented by using the java and in the game there is a snake that will increase the size by eating the food that is given in the code if the snake touches the end of the dialog box i.e the wall the snake will die and a dialog box is shown in that dialog box score will be shown The score will be shown accoding to the how much food that is eaten by the snake in this way the snake game works

TOOL:

SORTSITE FOR HTML,CSS,JAVASCRIPT:

ABSTRACT FOR SORT SITE:

Used for:

- **Google Webmaster Guidelines** - check for hidden text, single-pixel links, links to bad neighborhoods, sneaky redirects, etc.
- **Bing/Live Webmaster Guidelines** - check for keyword stuffed links and ALT tags, etc.
- **Best-practice optimization guidelines** - from leading industry experts
- Plus link checking, HTML standards compliance, usability and accessibility

TEST COMPLETE FOR NETBEANS

Abstract for the test complete :

TestComplete is used to create and automate many different software test types. Record and playback test creation records a tester performing a manual test and allows it to be played back and maintained over and over again as an automated test. Recorded tests can be modified later by testers to create new tests or enhance existing tests with more use cases TestComplete is an automated testing tool that lets you create, manage and run tests for any Windows, Web or Rich Client software. It makes it easy for anyone to create automated tests. And automated tests run faster, increase test coverage and lower costs. TestComplete's new script-free keyword testing, ease of use enhancements and centralized Start Page make learning automated testing a snap for new users.

TestComplete's flexibility and extensive feature list ensure power users always have a solution to testing challenges. TestComplete is a must-have tool for QA teams that need to do more testing, keep up with rapid development schedules and still deliver software on time.

Advantages of the “test complete tool”:

Simplistic Cross Platform Testing

TestComplete Platform includes Test-complete Web, TestComplete Desktop and TestComplete Mobile thereby allowing automated testing for of the above mentioned platforms.As for Desktop application testing, you can create robust tests at object level so that regression testing doesn’t fail even with the change in GUI. .NET applications can be tested thoroughly through scripts calling various java and .NET classes directly. Windows API can also call DLL and WMI functions. 3rd party control tools like Telerik, Sync fusion, Microsoft etc. can be used and testing apps on virtual machines or on cloud is also feasible.For Web apps, TestComplete Web supports Selenium, in which a single test of code can be utilized for cross browser testing. Selenium Web Driver also provides a huge advantage by allowing TestComplete to execute the web driver’s tests. With the usage of just one test, testing across multiple mobile apps can also be done. It is similar to desktop version but with a surplus of 3rd party controls availability.In the case of Mobile testing, TestComplete Mobile allows users to test varied apps that include native, web iOS, Android and hybrid apps, also with iOS8 and Apple’s new programming language Swift. For mobile apps, gestures can be recorded and executed for testing. Cloud testing is viable leading to reduction in expenses and hardware usage. Swift Execution

TestComplete Platform helps you to create accurate and repetitive automated testing swiftly on different devices, platforms and environments. There is availability of a simple Record and Playback functionality which can be availed by novices as well as experienced testers. A feature ‘Test Visualizer’ identifies the changes done by capturing the screen shots of every single operation performed which in turn, helps in saving time for the developers while debugging. Test Quality Improvement

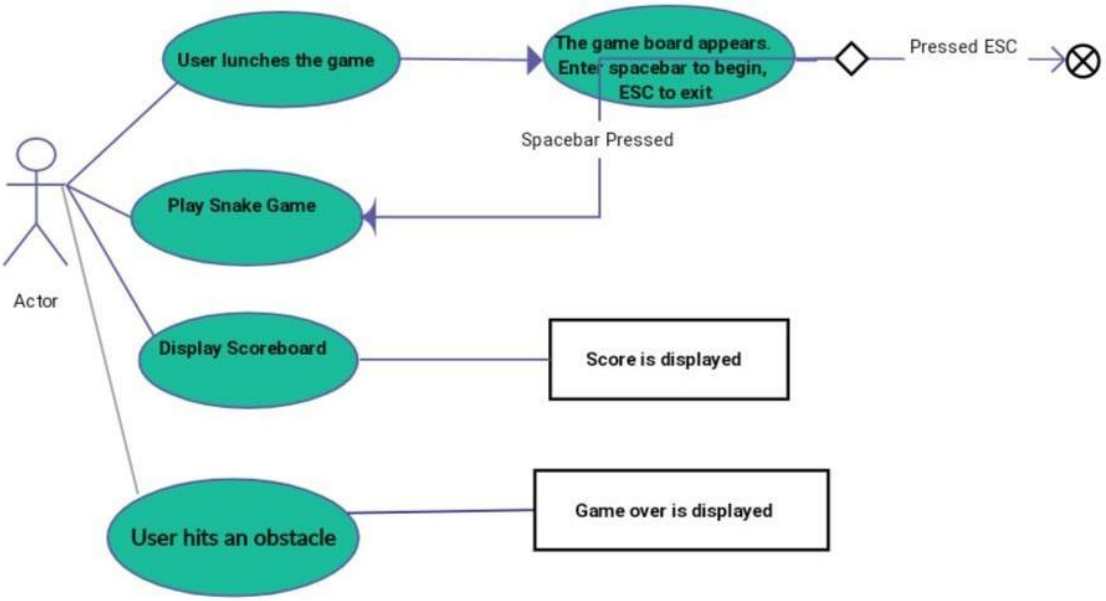
TestComplete's accessibility to internal objects and application properties provides increased test coverage and unique test logic from the base test data. Diverse checkpoints are available for verification of operations. As a part of automated testing, Regression testing is also very much possible, to ensure enhanced quality. Similar to QTP and other tools, TestComplete lets you test at GUI and API level with the help of integration with Soap UI,

Service V to name a few. Trimming the Costs of Testing Because each of them – Desktop, Web and Mobile platform, has individual licensing costs, it is cost effective to purchase the one which is needed. This is sure to save out on initial investments and maintenance costs in future. There are possibilities of executing automated testing on distinct workstations simultaneously, without buying a separate license for each individual machine. Yet another feature which can further reduced costs is Cloud testing. Easy Continuous Integration With its simplistic continuous integration feature, TestComplete turnsout to be easily accessible and usable. Jenkins plug-in can be used to launch and perform the necessary tests. The test logs and bug reports can be automatically attached to popular bug tracking tools like JIRA etc. Soap UI, Service VPro etc. can be used for creating functional API tests to be integrated with TestComplete. Integration with different source control systems like CVS and Microsoft SCCI compatible systems is possible. Also, integration with HP QC, AQ time etc. would help in improving the quality of automated testing applications especially for developers.

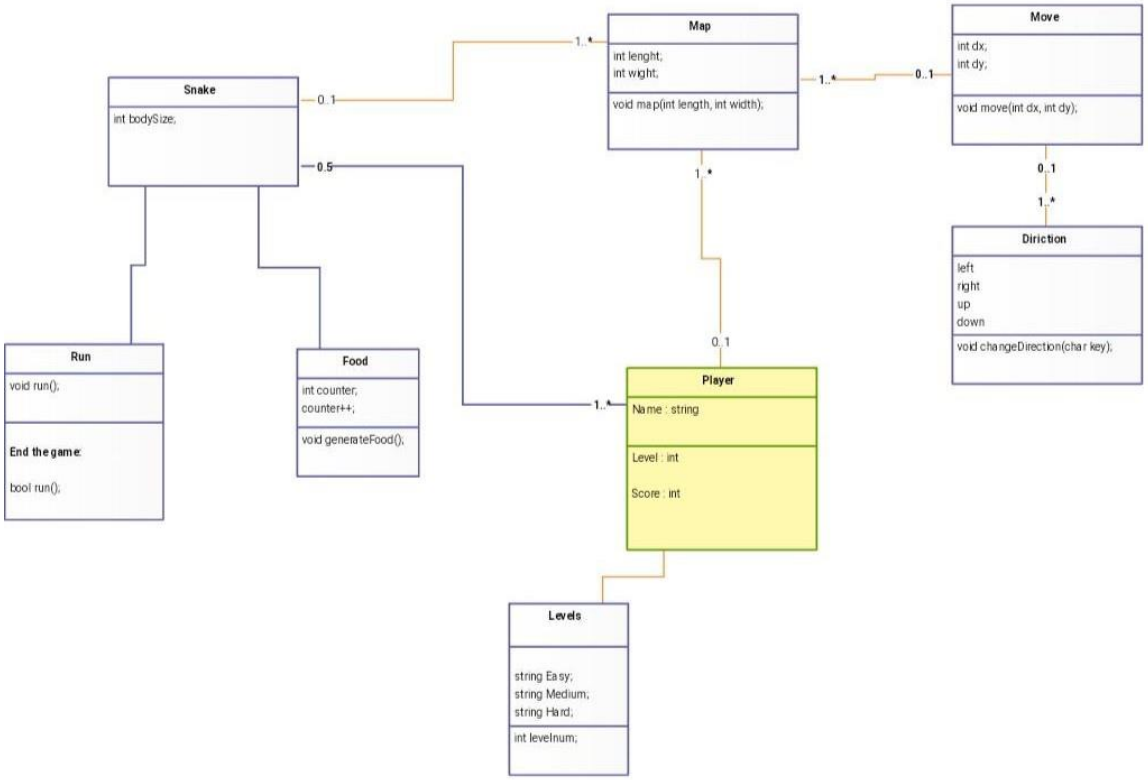
DESIGN:

UML DIAGRAMS:

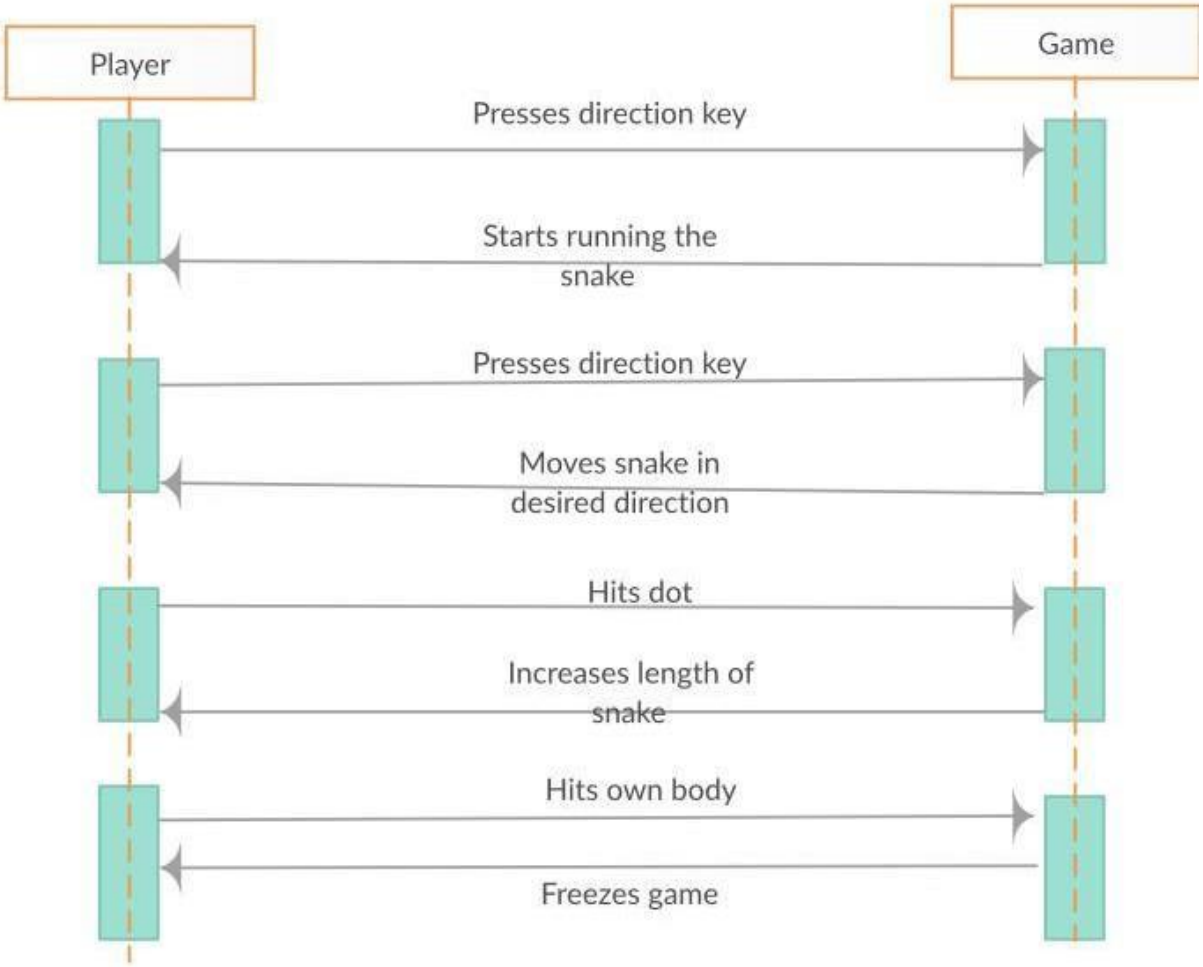
Use case diagram:



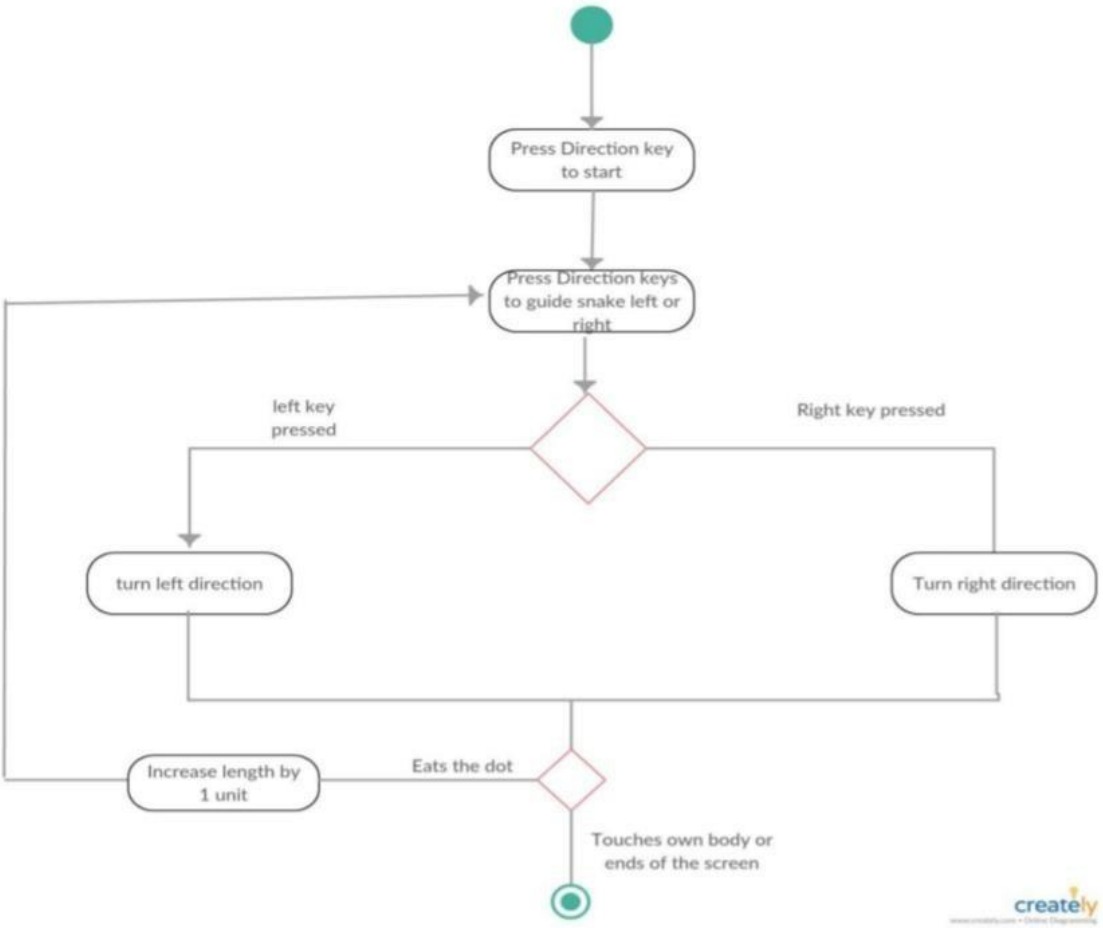
CLASS DIAGRAM:



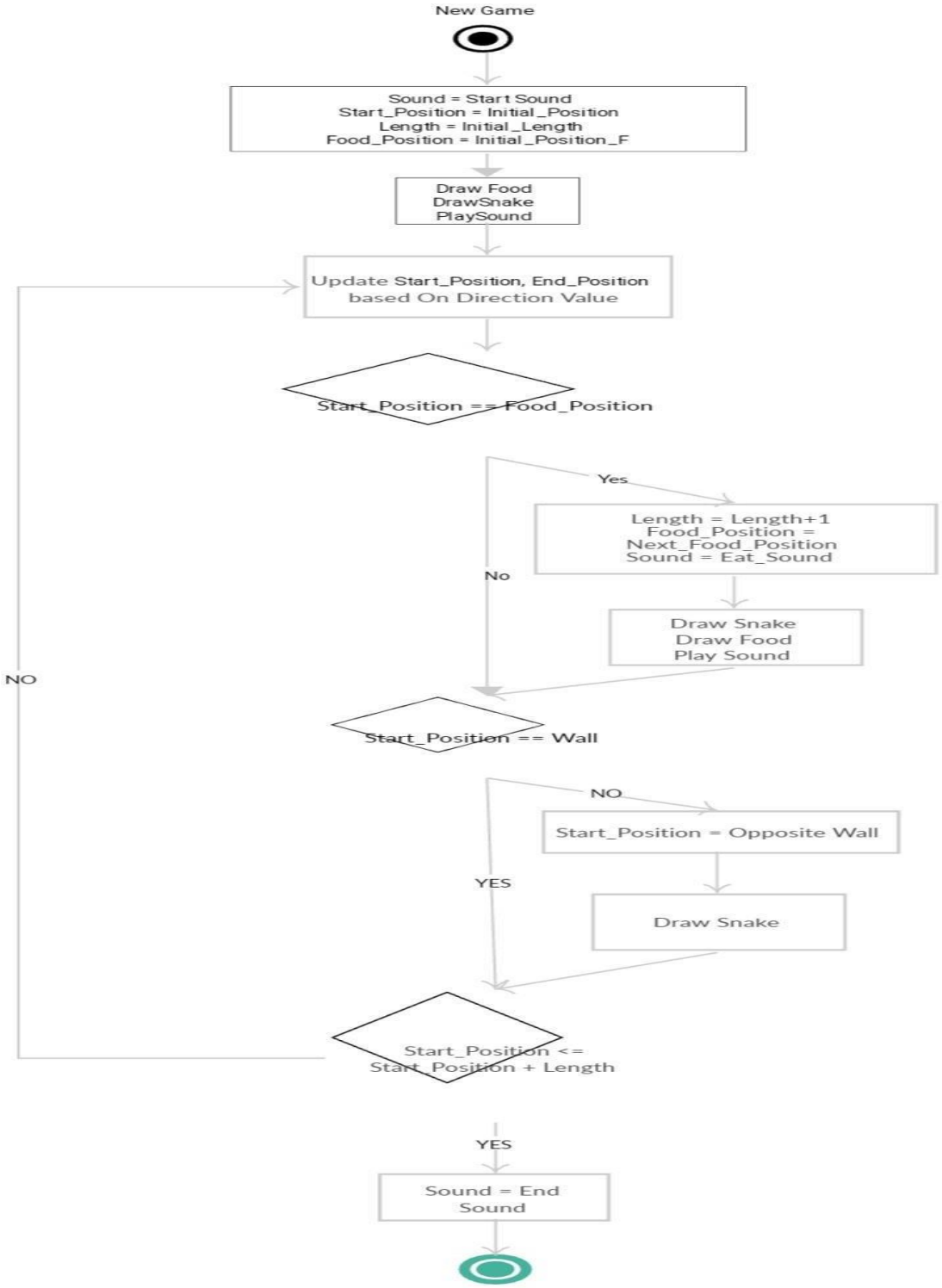
SEQUENCE DIAGRAM:



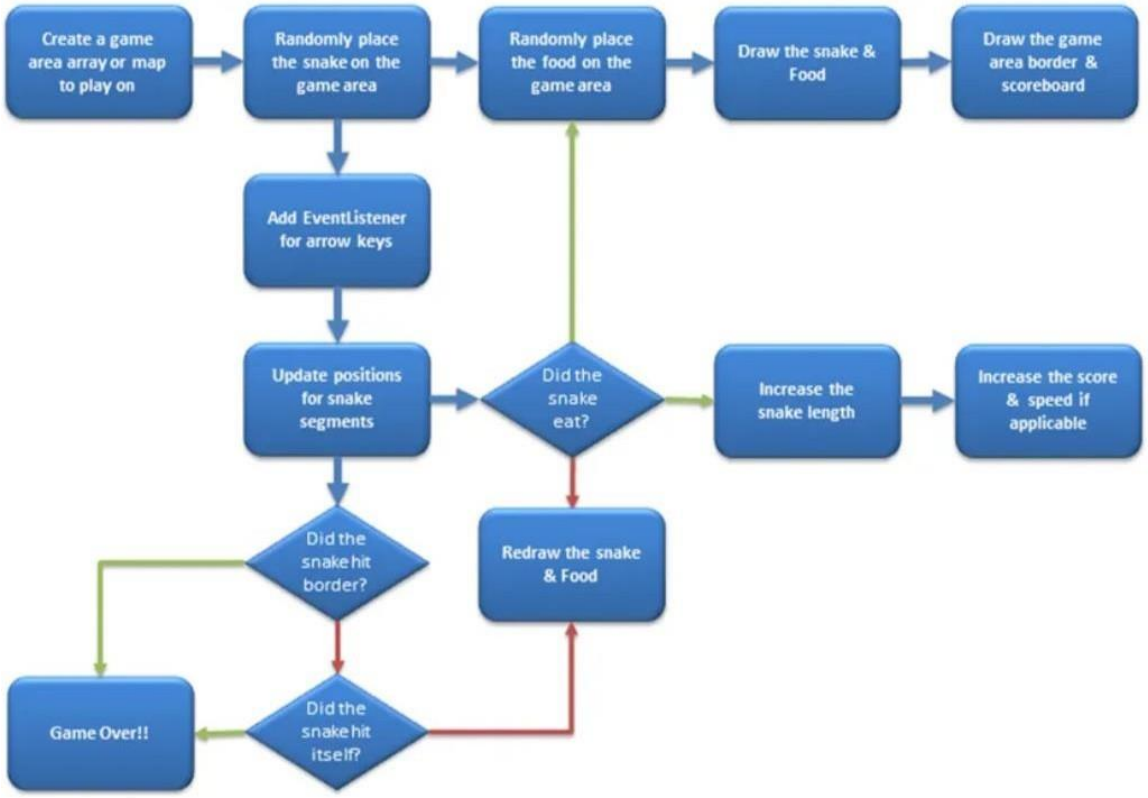
ACTIVITY DIAGRAM:



STATE CHART DIAGRAM:



ARCHITECTURE DIAGRAM:



1. SNAKE GAME USING HTML and JAVASCRIPT:

IMPLEMENTATION:

CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>The Snake Game</title>

</head>

<body style="margin: 0; background-color: #222222">

<div class="game" style="margin: 1rem auto 0; width: 600px">

  <h3 id="score" style="font-family: Arial, Helvetica, sans-serif; color: #ffffff">Press any arrow key to start moving!</h3>

  <h3 id="time" style="font-family: Arial, Helvetica, sans-serif; color: #ffffff">Time: 0.0s</h3>

  <canvas id="snakeGame" width="600" height="600"></canvas>

  <p style="float: right; font-family: Arial, Helvetica, sans-serif; color: #ffffff">Made by Kizuru</p>

</div>


<script>

    const arraysMatch = function (arr1, arr2) {

        if (arr1.length !== arr2.length) return false;
```

```
        for (let i = 0; i < arr1.length; i++) {
            if (arr1[i] !== arr2[i]) return false;
        }

        return true;
    };

    class SnakeGame {
        constructor(x, y, cs, canvas) {
            // Direction (first arg is x movement, second is y movement: [-1, 0]
means 'left', [0, 1] would mean 'down', I hope you get it)
            this.direction = [0, 0];
            // Default states
            this.launched = false;
            this.hasStarted = false;
            // Frames/s
            this.timer = 1e3 / (cs / 2);
            // Running time (for the HTML timer)
            this.runningTime = 0;

            // if there is no context (how would this happen? idk...)
            if (!canvas.getContext('2d')) throw 'nope';

            // Canvas context
            this.ctx = canvas.getContext('2d');
```

```
        // Block Size (px)
        this.cs = cs;
        // Amount of blocks in x coordinates
        this.xC = (x - (x % cs)) / cs;
        // Amount of blocks in y coordinates
        this.yC = (y - (y % cs)) / cs;

        // An apple
        this.apple = [Math.floor(Math.random() * this.xC), Math.floor(Math.random() * this.yC)];

        // The snake (nvm the usage of unknown class lmao)
        this.snake = new (class {
            constructor(x0, y0) {
                this.pos = [[x0, y0], [x0 + 1, y0], [x0 + 2, y0]];
            }
        })((this.xC - (this.xC % 2)) / 2 - 1, (this.yC - (this.yC % 2)) / 2 - 1);
    }

    // The game start method
    async start() {
        // Sey that the game has been started
        this.launched = true;
```

```
    // Initialize keys and background
    this.init()

    // Auto-Restart
    while (1) {
        // While the game still runs
        while (this.launched) {
            // Wait for the next frame
            await this.wait(this.timer);
            // Run the frame handler
            this.run();
        }
    }
}

init() {
    // Background rect
    this.ctx.fillStyle = '#222222';
    this.ctx.fillRect(0, 0, 600, 600);

    // Every time a key is down
    addEventListener('keydown', function (e) {
        // The movements (ordered so I can do some maths c:)
        const moves = ['ArrowUp', 'ArrowLeft', 'ArrowDown', 'ArrowRight'];
```

```
// if the key pressed is a valid move key
if (moves.includes(e.key)) {
    // The move index
    const move = moves.indexOf(e.key);

    if (move % 2 === 0) {
        // Movement: up or down
        if (game.direction[0] !== 0 || !game.hasStarted) {
            // Change the y direction
            game.direction[1] = move === 0 ? -1 : 1;
            game.direction[0] = 0;
            // Set game to started if it is not
            if (!game.hasStarted) game.hasStarted = true;
        }
    } else if (move % 2 === 1) {
        // Movement: left or right
        if (game.direction[1] !== 0 || !game.hasStarted) {
            // Change the x direction
            game.direction[0] = move === 1 ? -1 : 1;
            game.direction[1] = 0;
            // Set game to started if it is not
            if (!game.hasStarted) game.hasStarted = true;
        }
```

```
            }
        }
    });
}

// Method that run on each frame
run() {
    if (this.hasStarted) {
        // Move handler
        const positions = this.snake.pos.splice(0, this.snake.pos.length - 1);
        this.snake.pos = [positions[0].map((p, i) => p += this.direction[i]),
...positions];

        if (arraysMatch(this.snake.pos[0], this.apple)) {
            // Apple Eaten?
            this.snake.pos.push(this.apple);
            this.apple = [Math.floor(Math.random() * this.xC),
Math.floor(Math.random() * this.yC)];
        } else if (Array.from(this.snake.pos).splice(1,
this.snake.pos.length).map(p => arraysMatch(p,
this.snake.pos[0])).includes(true)) {
            // Self hit?
            alert(`You lose! Your total score was ${this.snake.pos.length -
3}!`);
            this.hasStarted = false;
            this.reset();
```

```
            }

            if (this.snake.pos[0].map((p, i) => i === 0 ? p < 0 || p > this.xC - 1 : p < 0 || p > this.yC - 1).includes(true)) {
                // Wall hit?
                alert(`You lose! Your total score was ${this.snake.pos.length - 3}!`);
                this.hasStarted = false;
                this.reset();
            }

            // Update game running time
            this.runningTime += this.timer;

            // Change HTML texts
            document.getElementById('score').innerHTML = `You have ${this.snake.pos.length - 3} points.`;
            document.getElementById('time').innerHTML = `Time: ${(this.runningTime / 1e3).toFixed(1)}s`;
        }

        // Draw the canvas
        return this.draw();
    }

    draw() {
```

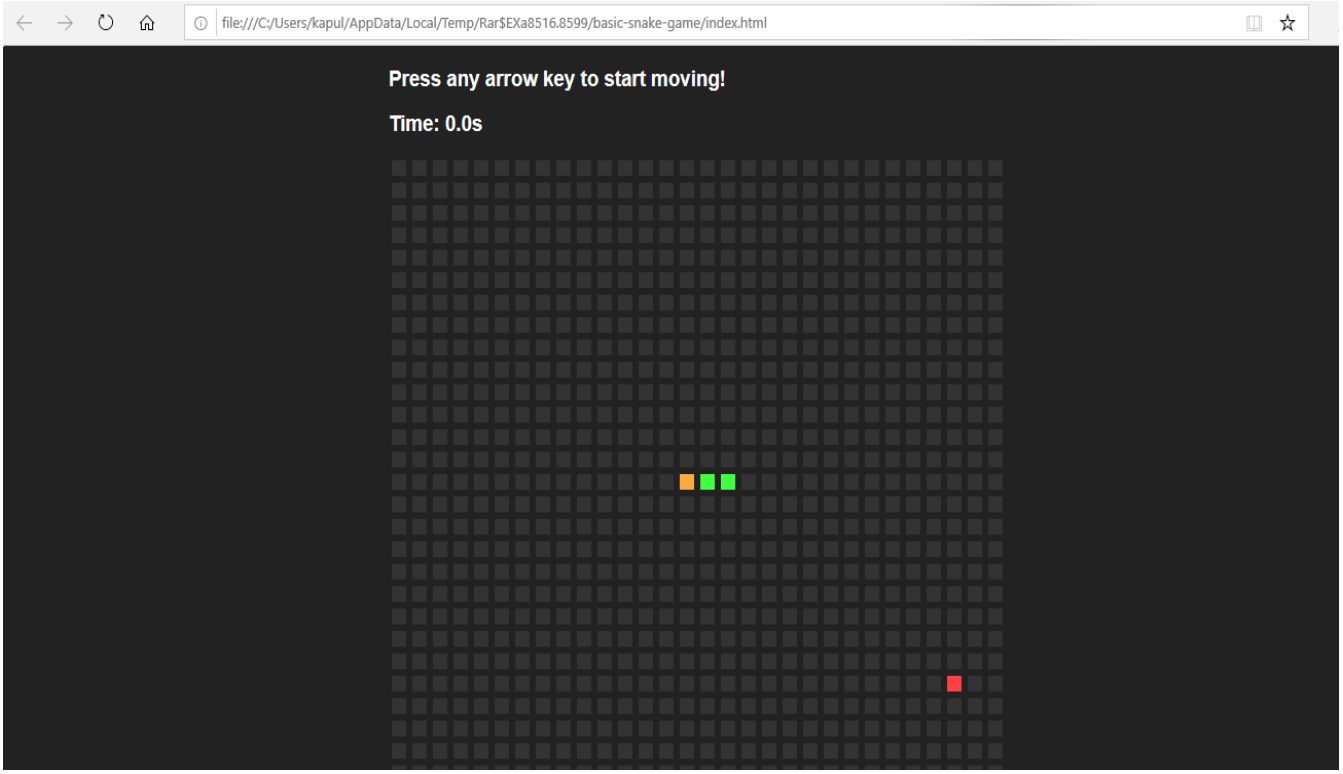
```
        // Some constant info
        const { ctx, cs } = this;

        // For each block in the x line
        for (let xC = 0; xC < this.xC; xC++) {
            // For each block in the y line
            for (let yC = 0; yC < this.yC; yC++) {
                // Location of the block
                const pos = [xC, yC];
                // Block draw offset
                const offset = 3;

                if (this.snake.pos.map(p => arraysMatch(p, pos)).includes(true)) {
                    // If the current block is a part of the snake
                    if (this.snake.pos.map(p => arraysMatch(p, pos)).indexOf(true)
!== 0) ctx.fillStyle = '#41ff41';
                    else ctx.fillStyle = '#ffaa41';
                    ctx.fillRect(xC * cs + offset, yC * cs + offset, cs - 2 * offset, cs -
2 * offset);
                } else if (arraysMatch(this.apple, pos)) {
                    // If the current block is an apple
                    ctx.fillStyle = '#ff4141';
                    ctx.fillRect(xC * cs + offset, yC * cs + offset, cs - 2 * offset, cs -
2 * offset);
                } else {
```

```
                // If the current block is nothing special
                ctx.fillStyle = '#333333';
                ctx.fillRect(xC * cs + offset, yC * cs + offset, cs - 2 * offset, cs - 2 * offset);
            }
        }
    }
}

// Method to use rather than using intervals, very useful btw
async wait(t) {
    return new Promise(resolve => setTimeout(() => resolve(), t));
}

// Method used to reset the whole game
reset() {
    this.apple = [Math.floor(Math.random() * this.xC), Math.floor(Math.random() * this.yC)];

    this.snake = new (class {
        constructor(x0, y0) {
            this.pos = [[x0, y0], [x0 + 1, y0], [x0 + 2, y0]];
        }
    })((this.xC - (this.xC % 2)) / 2 - 1, (this.yC - (this.yC % 2)) / 2 - 1);
```

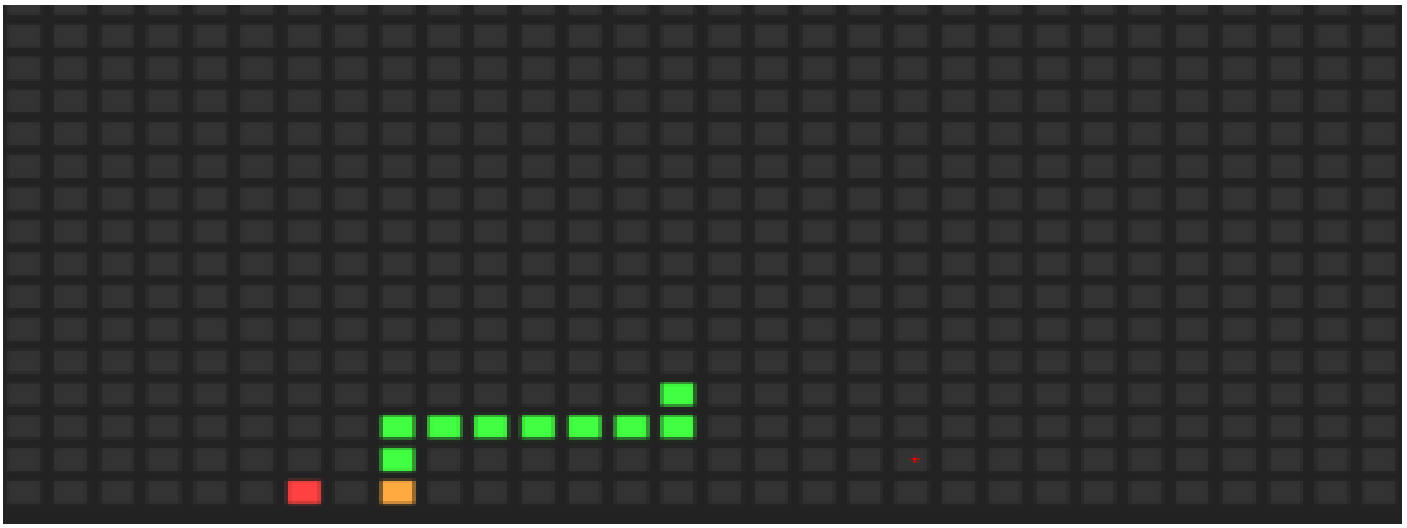
```
        // Reset game time
        this.runningTime = 0;

        // Set score text
        document.getElementById('score').innerHTML = `You have 0 points.`;
        // Set start text
        document.getElementById('time').innerHTML = `Press any arrow key to start moving!`;
      }
    }

    // Create the game instance
    const game = new SnakeGame(600, 600, 20, document.getElementById('snakeGame'));

    // Start the game
    game.start();
</script>
</body>
</html>
```

OUTPUT:

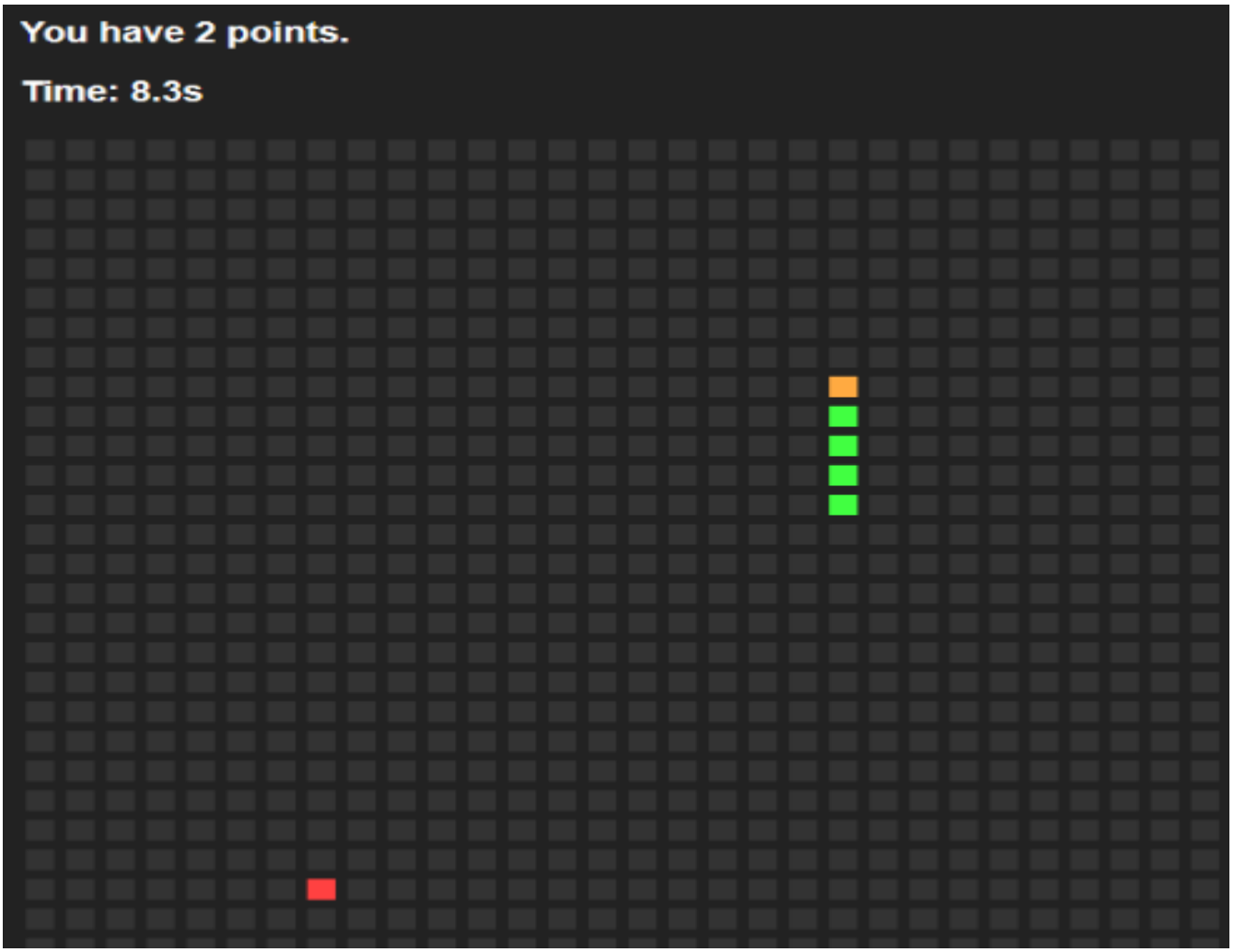
1. press any key to start the snake to move:



2. The snake starts moving and when we press arrow keys the snake eats the food and increases its size.



3.The time and points is shown at the top left corner of the screen.



4. Now when the snake hits the wall it dies and the score is shown.

2. SNAKE GAME USING C LANGUAGE:

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <conio.h>
#include<time.h>
#include<ctype.h>
#include <time.h>
#include <windows.h>
#include <process.h>

#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77

int length;
int bend_no;
int len;
char key;
void record();
void load();
int life;
```

```
void Delay(long double);
void Move();
void Food();
int Score();
void Print();
void gotoxy(int x, int y);
void GotoXY(int x,int y);
void Bend();
void Boarder();
void Down();
void Left();
void  Up();
void Right();
void ExitGame();
int Scoreonly();

struct coordinate
{
    int x;
    int y;
    int direction;
};

typedef struct coordinate coordinate;

coordinate head, bend[500],food,body[30];

int main()
{

    char key;

    Print();

    system("cls");

    load();
```

```
    length=5;

    head.x=25;

    head.y=20;

    head.direction=RIGHT;

    Boarder();

    Food(); //to generate food coordinates initially

    life=3; //number of extra lives

    bend[0]=head;

    Move();   //initialing initial bend coordinate

    return 0;

}

void Move()
{
    int a,i;

    do
    {

        Food();
        fflush(stdin);

        len=0;

        for(i=0; i<30; i++)
```

```
    {
        body[i].x=0;
        body[i].y=0;
        if(i==length)
            break;
    }
    Delay(length);
    Boarder();
    if(head.direction==RIGHT)
        Right();
    else if(head.direction==LEFT)
        Left();
    else if(head.direction==DOWN)
        Down();
    else if(head.direction==UP)
        Up();
    ExitGame();
}
while(!kbhit());
```

```
    a=getch();

    if(a==27)

    {

        system("cls");

        exit(0);

    }
    key=getch();


if((key==RIGHT&&head.direction!=LEFT&&head.direction!=RIGHT)||(key==L
EFT&&head.direction!=RIGHT&&head.direction!=LEFT)||(key==UP&&head.dir
ection!=DOWN&&head.direction!=UP)||(key==DOWN&&head.direction!=UP&
&head.direction!=DOWN))

    {

        bend_no++;

        bend[bend_no]=head;

        head.direction=key;

        if(key==UP)

            head.y--;

        if(key==DOWN)

            head.y++;

        if(key==RIGHT)
```

```
            head.x++;
        if(key==LEFT)
            head.x--;
        Move();
    }
    else if(key==27)
    {
        system("cls");
        exit(0);
    }
    else
    {
        printf("\a");
        Move();
    }
}
void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
```

```
    coord.Y = y;

    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);

}
void GotoXY(int x, int y)
{
    HANDLE a;
    COORD b;
    fflush(stdout);
    b.X = x;
    b.Y = y;
    a = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(a,b);
}
void load()
{
    int row,col,r,c,q;
    gotoxy(36,14);
    printf("loading...");
    gotoxy(30,15);
    for(r=1; r<=20; r++)
    {
        for(q=0; q<=100000000; q++); //to display the character slowly
        printf("%c",177);
    }
    getch();
}
void Down()
{
    int i;
    for(i=0; i<=(head.y-bend[bend_no].y)&&len<length; i++)
    {
        GotoXY(head.x,head.y-i);
        {
            if(len==0)
```

```
            printf("v");
        else
            printf("*");
    }
    body[len].x=head.x;
    body[len].y=head.y-i;
    len++;
  }
  Bend();
  if(!kbhit())
    head.y++;
}
void Delay(long double k)
{
  Score();
  long double i;
  for(i=0; i<=(10000000); i++);
}
void ExitGame()
{
  int i,check=0;
  for(i=4; i<length; i++) //starts with 4 because it needs minimum 4 element to
touch its own body
  {
    if(body[0].x==body[i].x&&body[0].y==body[i].y)
    {
      check++;     //check's value increases as the coordinates of head is equal to
any other body coordinate
    }
    if(i==length||check!=0)
      break;
  }
  if(head.x<=10||head.x>=70||head.y<=10||head.y>=30||check!=0)
  {
    life--;
    if(life>=0)
    {
```

```
            head.x=25;
            head.y=20;
            bend_no=0;
            head.direction=RIGHT;
            Move();
        }
        else
        {
            system("cls");
            printf("All lives completed\nBetter Luck Next Time!!!\nPress any key to
quit the game\n");
            record();
            exit(0);
        }
    }
}
void Food()
{
    if(head.x==food.x&&head.y==food.y)
    {
        length++;
        time_t a;
        a=time(0);
        srand(a);
        food.x=rand()%70;
        if(food.x<=10)
            food.x+=11;
        food.y=rand()%30;
        if(food.y<=10)

            food.y+=11;
    }
    else if(food.x==0)/*to create food for the first time coz global variable are
initialized with 0*/
    {
        food.x=rand()%70;
        if(food.x<=10)
```

```
            food.x+=11;
        food.y=rand()%30;
        if(food.y<=10)
            food.y+=11;
    }
}
void Left()
{
    int i;
    for(i=0; i<=(bend[bend_no].x-head.x)&&len<length; i++)
    {
        GotoXY((head.x+i),head.y);
        {
            if(len==0)
                printf("<");
            else
                printf("*");
        }
        body[len].x=head.x+i;
        body[len].y=head.y;
        len++;
    }
    Bend();
    if(!kbhit())
        head.x--;

}
void Right()
{
    int i;
    for(i=0; i<=(head.x-bend[bend_no].x)&&len<length; i++)
    {
        //GotoXY((head.x-i),head.y);
        body[len].x=head.x-i;
        body[len].y=head.y;
        GotoXY(body[len].x,body[len].y);
        {
```

```
            if(len==0)
                printf(">");
            else
                printf("*");
        }
        /*body[len].x=head.x-i;
        body[len].y=head.y;*/
        len++;
    }
    Bend();
    if(!kbhit())
        head.x++;
}
void Bend()
{
    int i,j,diff;
    for(i=bend_no; i>=0&&len<length; i--)
    {
        if(bend[i].x==bend[i-1].x)
        {
            diff=bend[i].y-bend[i-1].y;
            if(diff<0)
                for(j=1; j<=(-diff); j++)
                {
                    body[len].x=bend[i].x;
                    body[len].y=bend[i].y+j;
                    GotoXY(body[len].x,body[len].y);
                    printf("*");
                    len++;
                    if(len==length)
                        break;
                }
            else if(diff>0)
                for(j=1; j<=diff; j++)
                {
                    /*GotoXY(bend[i].x,(bend[i].y-j));
                    printf("*");*/
```

```
            body[len].x=bend[i].x;
            body[len].y=bend[i].y-j;
            GotoXY(body[len].x,body[len].y);
            printf("*");
            len++;
            if(len==length)
                break;
        }
    }
    else if(bend[i].y==bend[i-1].y)
    {
        diff=bend[i].x-bend[i-1].x;
        if(diff<0)
            for(j=1; j<=(-diff)&&len<length; j++)
            {
                /*GotoXY((bend[i].x+j),bend[i].y);
                printf("*");*/
                body[len].x=bend[i].x+j;
                body[len].y=bend[i].y;
                GotoXY(body[len].x,body[len].y);
                printf("*");
                len++;
                if(len==length)
                    break;
            }
        else if(diff>0)
            for(j=1; j<=diff&&len<length; j++)
            {
                /*GotoXY((bend[i].x-j),bend[i].y);
                printf("*");*/
                body[len].x=bend[i].x-j;
                body[len].y=bend[i].y;
                GotoXY(body[len].x,body[len].y);
                printf("*");
                len++;
                if(len==length)
                    break;
```

```
            }
        }
    }
}
void Boarder()
{
    system("cls");
    int i;
    GotoXY(food.x,food.y);   /*displaying food*/
    printf("F");
    for(i=10; i<71; i++)
    {
        GotoXY(i,10);
        printf("!");
        GotoXY(i,30);
        printf("!");
    }
    for(i=10; i<31; i++)
    {
        GotoXY(10,i);
        printf("!");
        GotoXY(70,i);
        printf("!");
    }
}
void Print()
{
    //GotoXY(10,12);
    printf("\tWelcome to the mini Snake game.(press any key to continue)\n");
    getch();
    system("cls");
    printf("\tGame instructions:\n");
    printf("\n-> Use arrow keys to move the snake.\n\n-> You will be provided foods at the several coordinates of the screen which you have to eat. Everytime you eat a food the length of the snake will be increased by 1 element and thus the score.\n\n-> Here you are provided with three lives. Your life will decrease as you hit the wall or snake's body.\n\n-> YOu can pause the game in its middle by
```

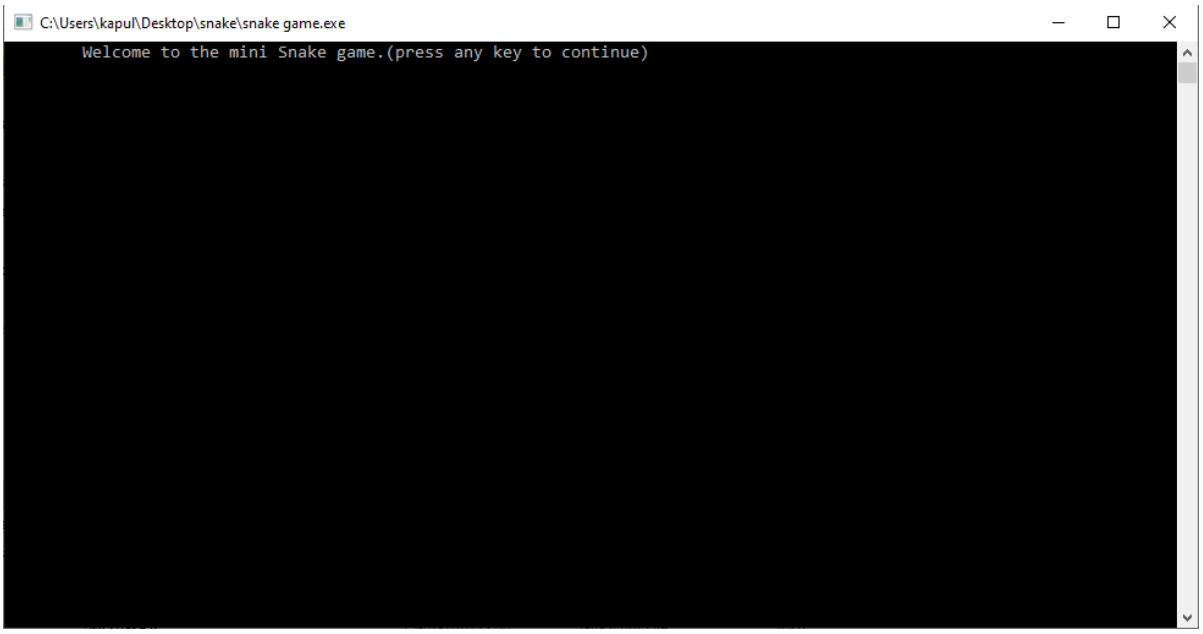
```
pressing any key. To continue the paused game press any other key once again\n\n-
> If you want to exit press esc. \n");
    printf("\n\nPress any key to play game...");
    if(getch()==27)
        exit(0);
}
void record()
{
    char plname[20],nplname[20],cha,c;
    int i,j,px;
    FILE *info;
    info=fopen("record.txt","a+");
    getch();
    system("cls");
    printf("Enter your name\n");
    scanf("%[^\n]",plname);
    //************************
    for(j=0; plname[j]!='\0'; j++) //to convert the first letter after space to capital
    {
        nplname[0]=toupper(plname[0]);
        if(plname[j-1]==' ')
        {
            nplname[j]=toupper(plname[j]);
            nplname[j-1]=plname[j-1];
        }
        else nplname[j]=plname[j];
    }
    nplname[j]='\0';
    //******************************
    //sdfprintf(info,"\t\t\tPlayers List\n");
    fprintf(info,"Player Name :%s\n",nplname);
    //for date and time

    time_t mytime;
    mytime = time(NULL);
    fprintf(info,"Played Date:%s",ctime(&mytime));
    //***************************
```

```
    fprintf(info,"Score:%d\n",px=Scoreonly());//call score to display score
    //fprintf(info,"\nLevel:%d\n",10);//call level to display level
    for(i=0; i<=50; i++)
        fprintf(info,"%c",'_');
    fprintf(info,"\n");
    fclose(info);
    printf("Wanna see past records press 'y'\n");
    cha=getch();
    system("cls");
    if(cha=='y')
    {
        info=fopen("record.txt","r");
        do
        {
            putchar(c=getc(info));
        }
        while(c!=EOF);
    }
    fclose(info);
}
int Score()
{
    int score;
    GotoXY(20,8);
    score=length-5;
    printf("SCORE : %d",(length-5));
    score=length-5;
    GotoXY(50,8);
    printf("Life : %d",life);
    return score;
}
int Scoreonly()
{
    int score=Score();
    system("cls");
    return score;
}
```

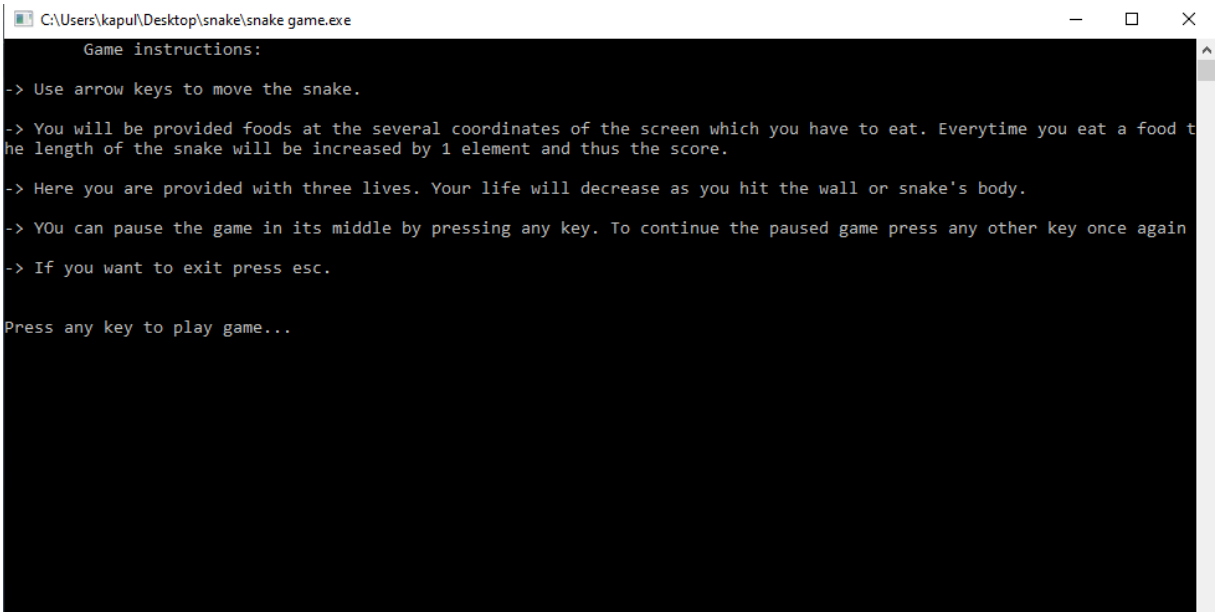
```
void Up()
{
    int i;
    for(i=0; i<=(bend[bend_no].y-head.y)&&len<length; i++)
    {
        GotoXY(head.x,head.y+i);
        {
            if(len==0)
                printf("^");
            else
                printf("*");
        }
        body[len].x=head.x;
        body[len].y=head.y+i;
        len++;
    }
    Bend();
    if(!kbhit())
        head.y--;
}
```

OUTPUT:

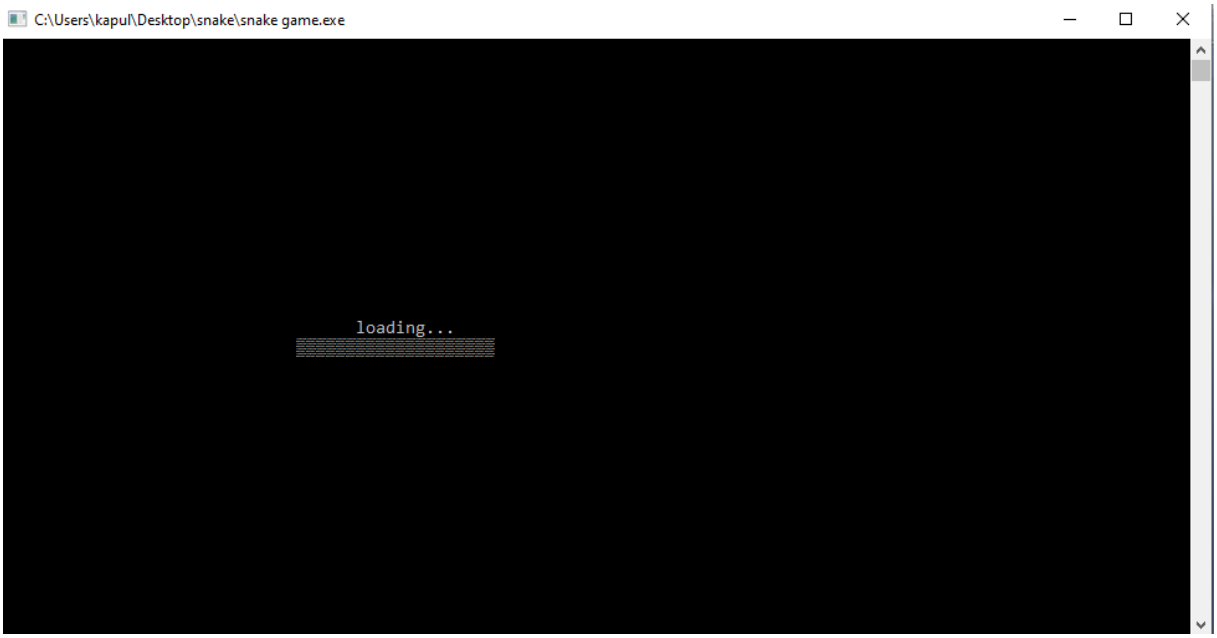
Start screen :



Game instructions:

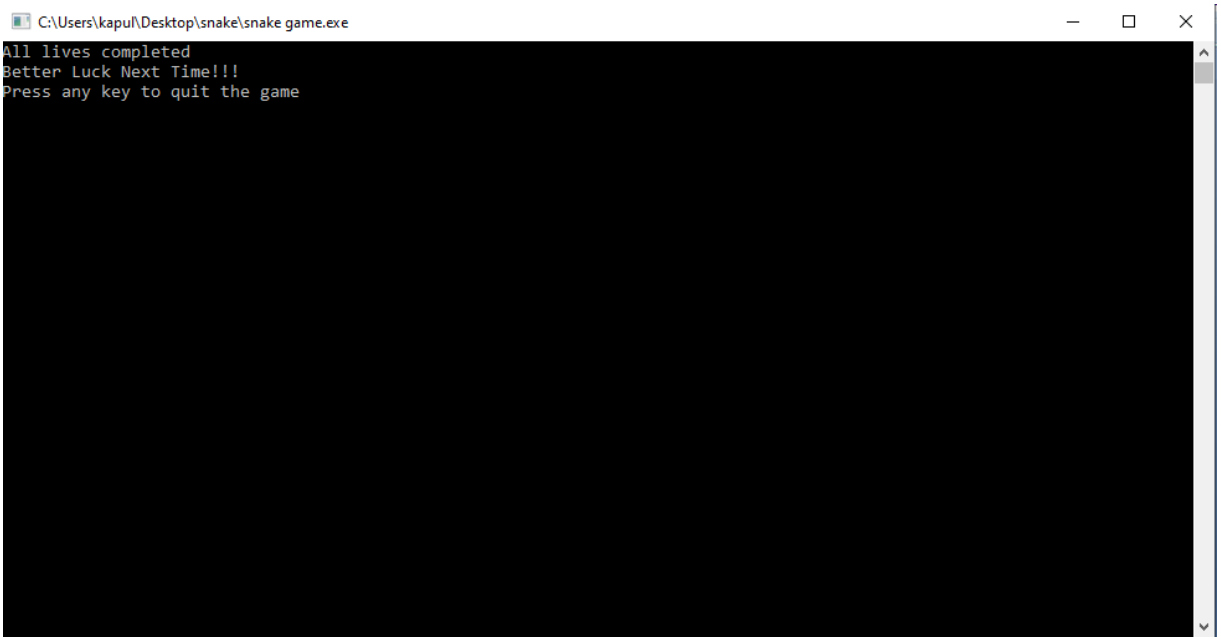


Screen loading:

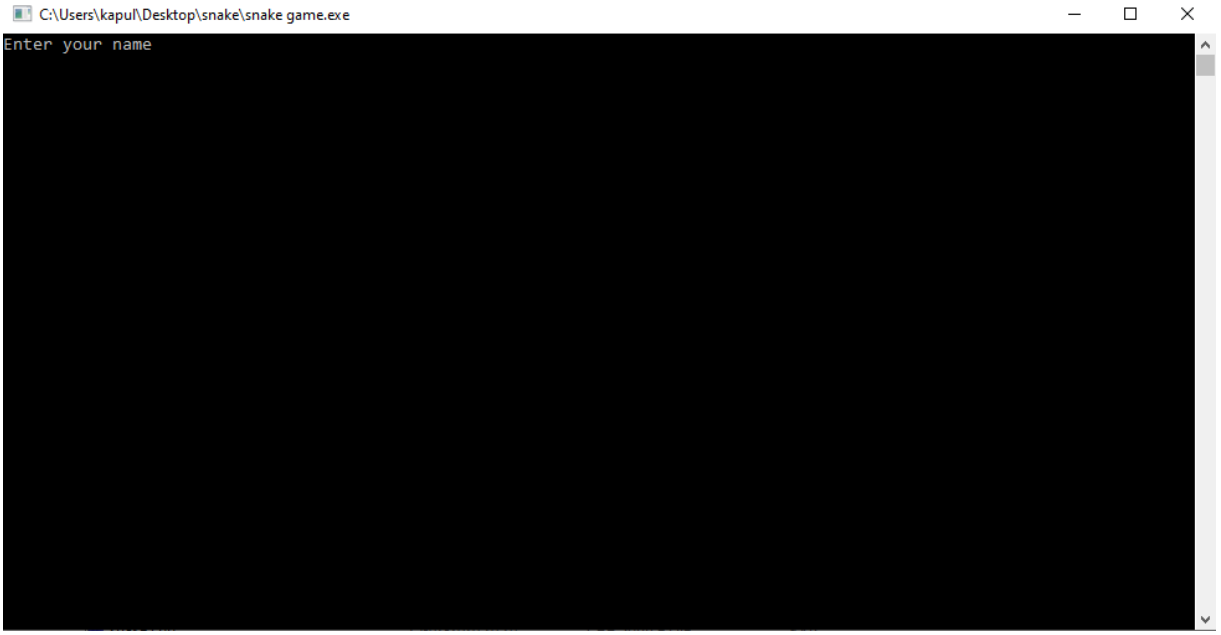


If snake hits the wall it uses its life and saves itself:

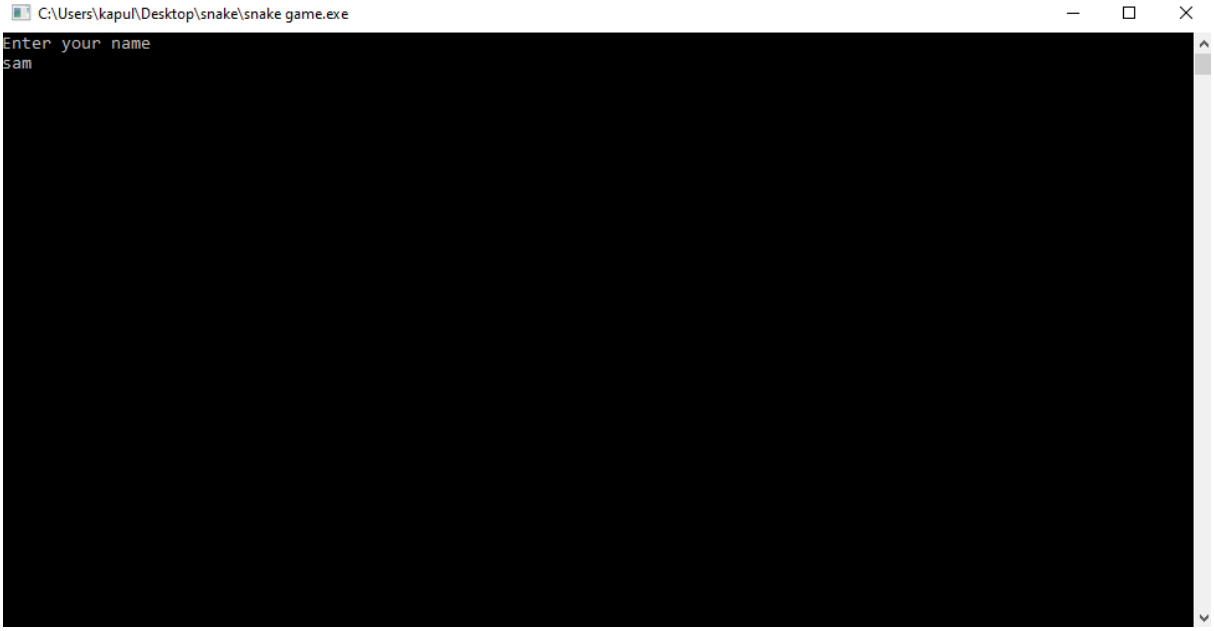
After completion of 3 lives snake dies:



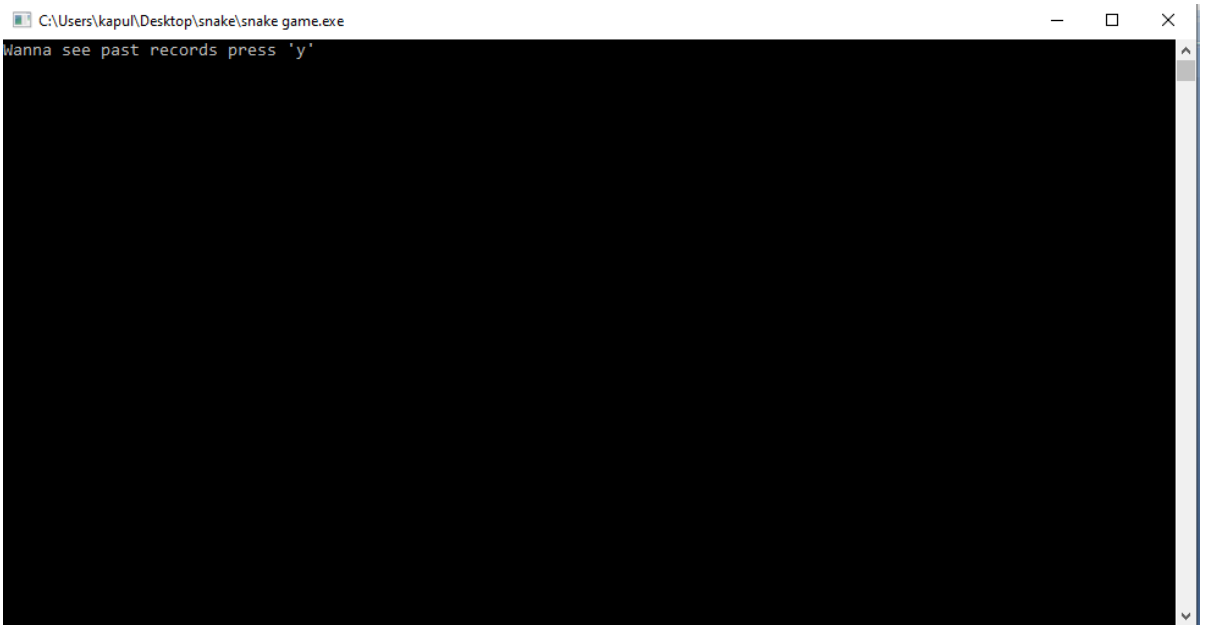
Now it asks for the name:



Asks for entering the name:



After entering name it asks for past records:



3. SNAKE GAME USING NETBEANS:

IMPLEMENTATION
CODE:

DisplayContoller.java

```
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.effect.DropShadow;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.ResourceBundle;

public class DisplayController extends Stage implements Initializable {
    @FXML private BorderPane pane;
    @FXML private Canvas canvas;
    private GraphicsContext gc;
    private Game game;
    private int stepSize;
    private ArrayList<Player> players;
    private DropShadow headGlow;

    public DisplayController() {

    }
```

```
@Override
public void initialize(URL location, ResourceBundle resources) {
        gc = canvas.getGraphicsContext2D();
        headGlow = new DropShadow();
        headGlow.setOffsetX(0f);
        headGlow.setOffsetY(0f);
}

public void setUpDisplay(Game game) {
        setGame(game);
        players = game.players;
        draw();

        for (Player player : players) {
                player.snake.aliveProperty().addListener(v -> {
                        if (!player.snake.getAlive()) {
                                // player DIED MATE
                                playerDied(player);
                        }
                });
        }

        headGlow.setWidth(stepSize * 1.2);
        headGlow.setHeight(stepSize * 1.2);

}

public void displayGameOver() {
        gc.setFill(Color.web("#e74c3c"));
        headGlow.setColor(Color.web("#e74c3c"));

        gc.setGlobalAlpha(0.7);
        gc.setEffect(headGlow);
        gc.fillText("GAME OVER", 20, 150);
        gc.setEffect(null);
        gc.setGlobalAlpha(1);
```

```
}

public void playerDied(Player player) {

        Timeline timer = new Timeline((new KeyFrame(
                        Duration.millis(300),
                        event -> {
                                setDeadColour(player.snake);
                        }
        )));

        timer.setCycleCount(10);
        timer.play();

        timer.setOnFinished(v -> {
                players.remove(player);
                players.trimToSize();
                // FIX!
                //players.add(new AI(stepSize, Game.colours[0], 1));
                //
        });
}

public void setDeadColour(Snake snake) {
        Color deadColourLight = Color.web("#e74c3c");
        Color deadColourDark = Color.web("#c0392b");

        if(snake.getColour().equals(deadColourLight)) {
                snake.setColour(deadColourDark);
        } else {
                snake.setColour(deadColourLight);
        }
}

public void draw() {
        clear();
```

```
		int counter = 0;
		Color colour;
		for (Player player : players) {
			drawSnake(player.snake);

			counter++;
		}

		drawFood();

		if(!game.isRunning()) {
			gc.setFont(new Font("Arial Rounded MT Bold", 58));
			displayGameOver();
		}

	}

	private void drawSnake(Snake snake) {
		drawSnake(snake, snake.getColour());
	}

	public void drawSnake(Snake snake, Color colour) {
		gc.setFill(colour);
		Iterator<SnakePiece> position = snake.descendingIterator();
		SnakePiece current = snake.get(snake.size() - 1);
		position.next(); // move position from tale to piece before tail

		// draw whole body except head and tail
		while (position.hasNext() && (current.getStatus() > 0)) {
			drawBody(current.getPosX(), current.getPosY());
			current = position.next();
		}

		headGlow.setColor(colour);
		gc.setEffect(headGlow);
		drawHead(snake.getFirst().getPosX(),
snake.getFirst().getPosY());
```

```
                gc.setEffect(null);
                drawTail(snake.getLast().getPosX(),
snake.getLast().getPosY());
        }

        public void clear() {
                gc.clearRect(0,0,canvas.getWidth(),canvas.getHeight());
                gc.setFill(Color.web("#2c3e50"));
                //gc.fill();
                gc.fillRect(0,0,canvas.getWidth(),canvas.getHeight());

        }

        public void drawFood() {
                gc.setFill(Color.web("#ecf0f1"));
                gc.fillOval(game.food.getPosX(),        game.food.getPosY(),
stepSize, stepSize);
        }

        public void drawHead(int posX, int posY) {
                gc.fillOval(posX, posY, stepSize, stepSize);

                gc.fillRect(posX,posY,stepSize,stepSize);

        }

        public void drawBody(int posX, int posY) {
                gc.fillRect(posX,posY,stepSize,stepSize);
        }

        public void drawTail(int posX, int posY) {
                gc.fillRect(posX,posY,stepSize,stepSize);
        }


        ////////////////////////////////////
```

```
        public GraphicsContext getGc() {
            return gc;
        }

        public void setGc(GraphicsContext gc) {
            this.gc = gc;
        }

        public Canvas getCanvas() {
            return this.canvas;
        }

        public void setCanvas(Canvas canvas) {
            this.canvas = canvas;
        }

        public BorderPane getPane() {
            return pane;
        }

        public void setPane(BorderPane pane) {
            this.pane = pane;
        }

        public Game getGame() {
            return game;
        }

        public void setGame(Game game) {
            this.game = game;
            stepSize = game.stepSize;
        }
}
```

Food.java

```
import javafx.beans.property.BooleanProperty;
import javafx.beans.property.SimpleBooleanProperty;

import java.util.Random;

public class Food {
	int posX, posY;
	BooleanProperty eaten = new SimpleBooleanProperty();
	public Food() {
		setEaten(false);
		Random rng = new Random();

		//posX = rng.nextInt(395);
		posX = rng.nextInt(390 - 10 + 1) + 10;
		posX = ((posX + 5) / 10)*10;
		posY = rng.nextInt(390 - 10 + 1) + 10;
		posY = ((posY + 5) / 10)*10;
	}

	public int getPosX() {
		return posX;
	}

	public void setPosX(int posX) {
		this.posX = posX;
	}

	public int getPosY() {
		return posY;
	}

	public void setPosY(int posY) {
		this.posY = posY;
	}
```

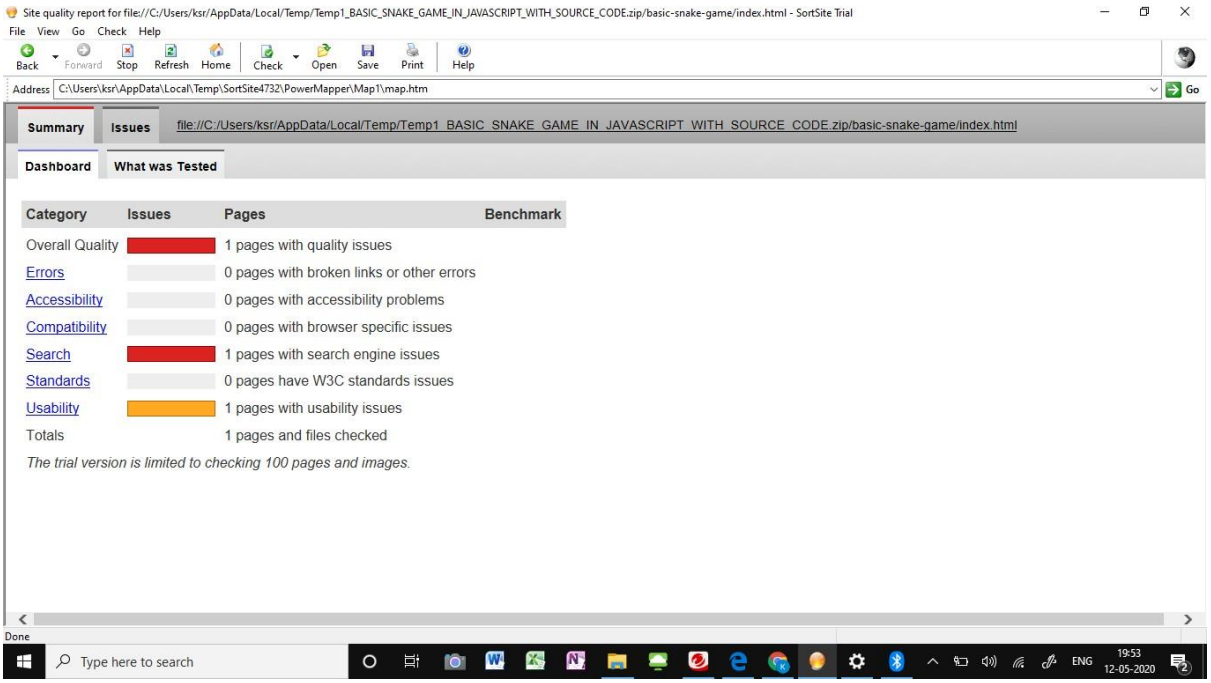
```
        public boolean isEaten() {
            return eaten.get();
        }

        public BooleanProperty eatenProperty() {
            return eaten;
        }

        public void setEaten(boolean eaten) {
            this.eaten.set(eaten);
        }
}
```

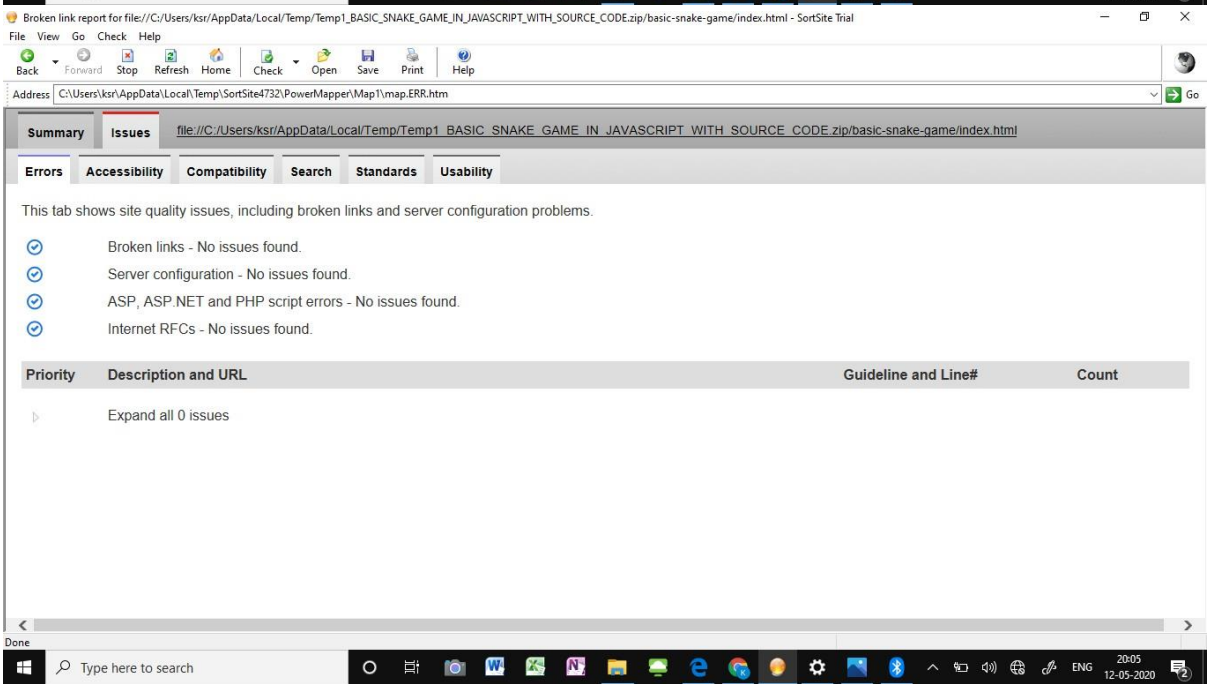
OUTPUT:

TESTING:
TESTCASES:

TEST CASE ID	INPUT	EXOECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	Upwa rd arrow in the key board	Navigate the snake to the upward	Navigated the snake to the upward	pass
2	Downward arrow in the keyboard	Navigate the snake to the downward	Navigated to the downward	pass
3	Leftward key in the keyboard	Navigate the snake to the downward	Navigated to the downward	pass
4	Rightward side key in the key board is pressed	Navigate the snake to the downward	Navigated to the downward of the key	pass
5	Both left and top keys of the keyboard are pressed	Navigate the snake to the north west direction	Navigated to the up word direction	fail
6	Both right bottom keys are pressed	Navigate the south east direction key	Navigated to the downward direction	fail
7	Both left bottom keys are pressed	Navigate to the south west direction of the	Navigated to the downward direction	fail
8	Both right and top keys are pressed	Navigate the snake to the North east direction	Navigated to the Downward direction	Fail

TESTING SNAKE GAME USING HTML ,CSS, JAVASCRIPT :
USING SORTSITE TOOL:





Accessibility validation report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial

File View Go Check Help

Back Forward Stop Refresh Home Check Open Save Print Help

Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.ACC.htm Go

Summary | Issues | file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html

Errors | Accessibility | Compatibility | Search | Standards | Usability

This tab shows accessibility issues, indicating problems for older users, people with disabilities or accessibility needs. Automated testing cannot detect all accessibility issues, so should be used alongside human testing.

Level	WCAG 2.1	Section 508 - 2017
A	✓	✓
AA	✓	✓
AAA	✓	

Key

- Pages with level A issues are unusable for some people
- Pages with level AA issues are very difficult to use
- Pages with level AAA issues can be difficult to use

Priority	Description and URL	Guideline and Line#	Count
	Expand all 0 issues		

Done

Type here to search ENG 20:06 12-05-2020

Browser compatibility report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial

File View Go Check Help

Back Forward Stop Refresh Home Check Open Save Print Help

Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.BUG.htm Go

Summary | Issues | file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html

Errors | Accessibility | Compatibility | Search | Standards | Usability

This tab shows pages that exhibit browser-specific behavior, or trigger browser bugs.

Browser	IE	Edge	Firefox	Safari	Opera	Chrome	iOS			Android	
Version	11	79	72	13	66	79	≤ 11	12	13	≤ 3	4*
Critical Issues	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Major Issues	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Minor Issues	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Key

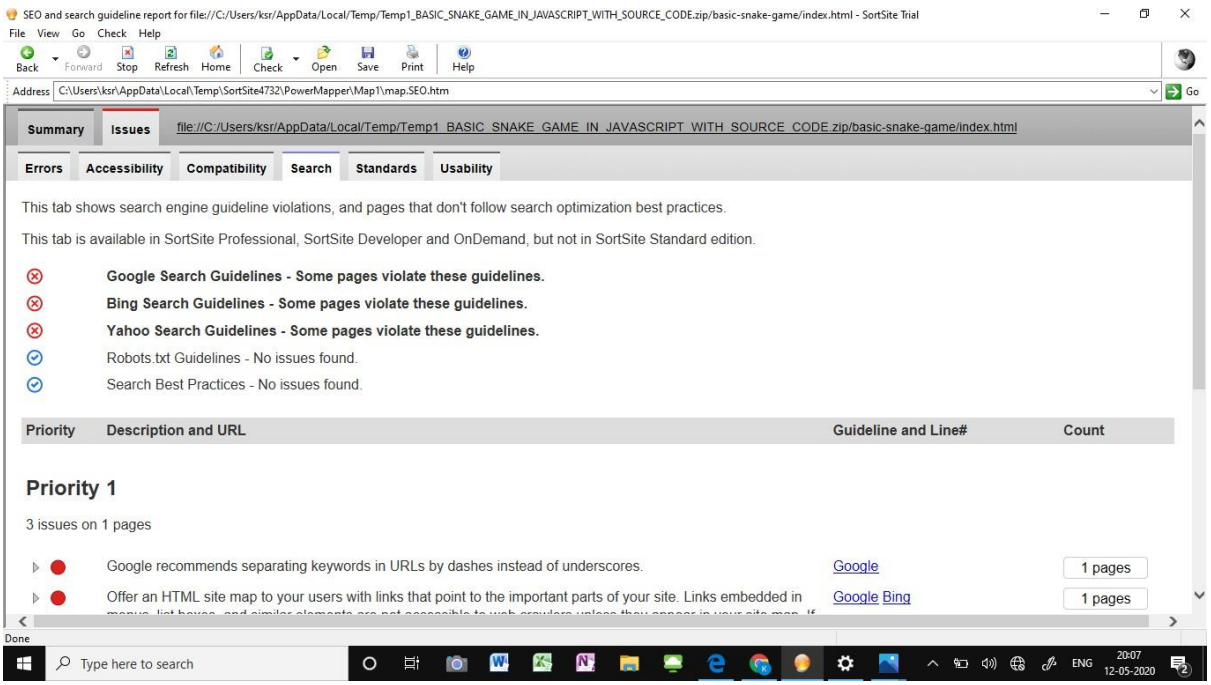
- Missing content or functionality
- Major layout or performance problems
- Minor layout or performance problems

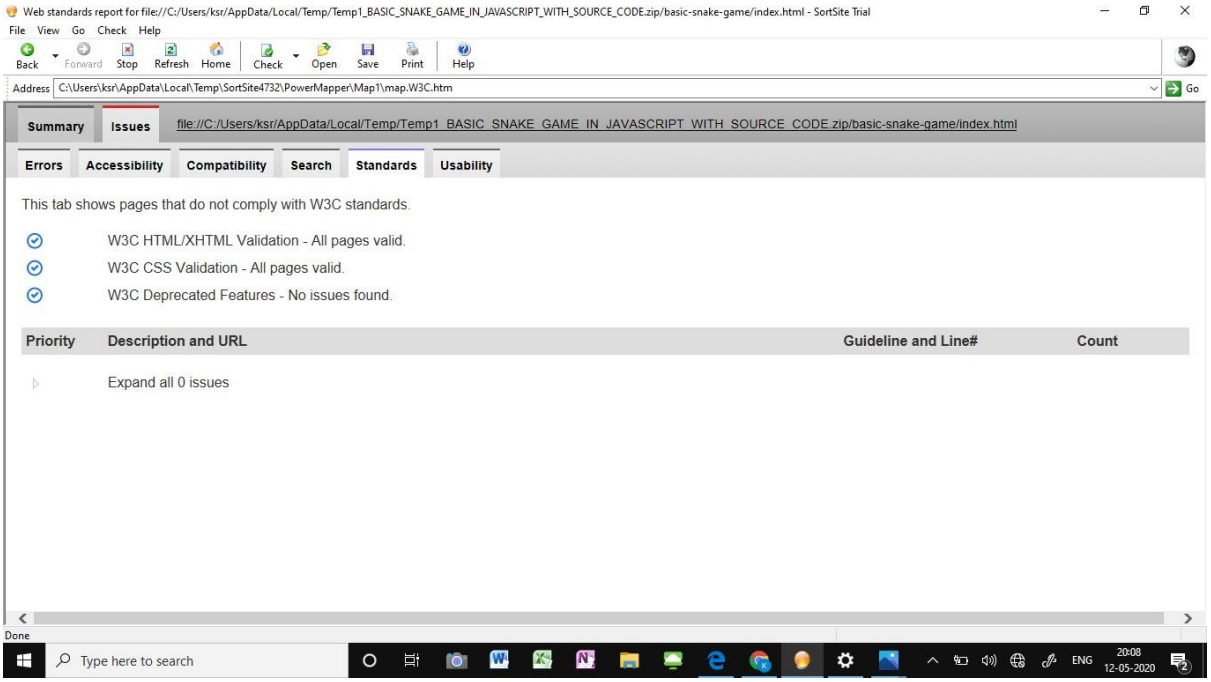
* Most Android devices from 4.4 onwards use Chrome as the default browser, older versions use the original Android stock browser

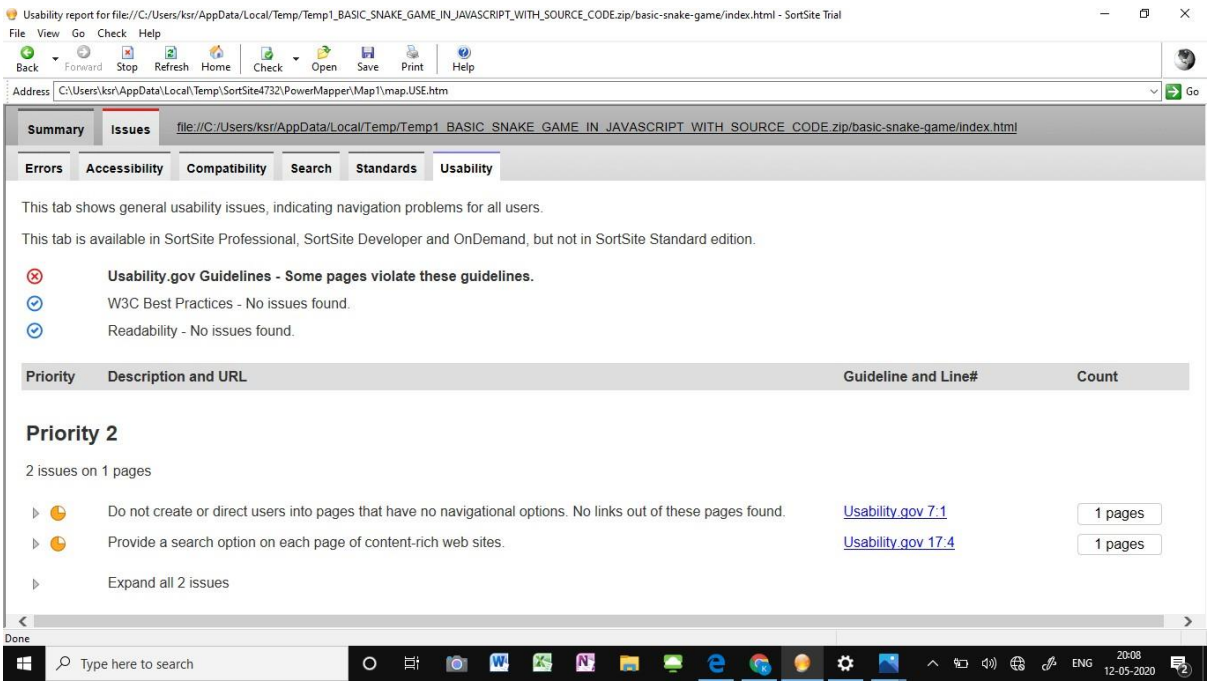
Priority	Description and URL	Guideline and Line#	Count
	Expand all 0 issues		

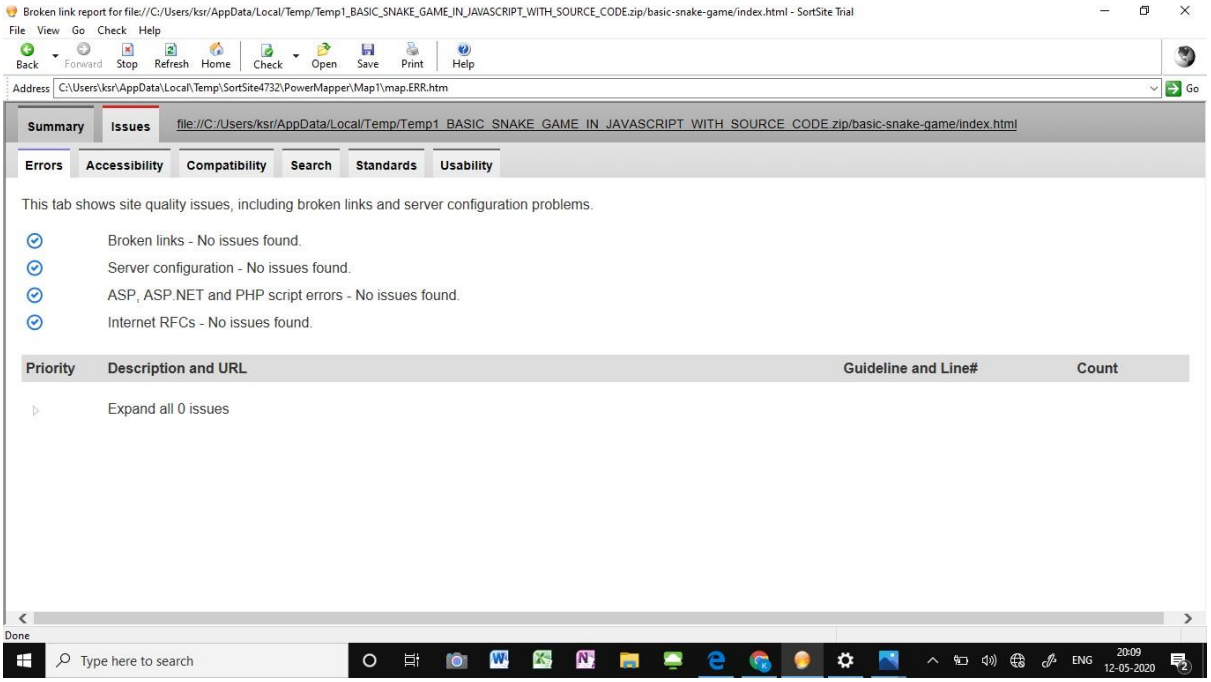
Done

Type here to search ENG 20:07 12-05-2020


SEO and search guideline report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial
File View Go Check Help
Back Forward Stop Refresh Home Check Open Save Print Help
Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.SEO.htm
Go
Summary
Issues
file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html
Errors
Accessibility
Compatibility
Search
Standards
Usability
This tab shows search engine guideline violations, and pages that don't follow search optimization best practices.
This tab is available in SortSite Professional, SortSite Developer and OnDemand, but not in SortSite Standard edition.
Google Search Guidelines - Some pages violate these guidelines.
Bing Search Guidelines - Some pages violate these guidelines.
Yahoo Search Guidelines - Some pages violate these guidelines.
Robots.txt Guidelines - No issues found.
Search Best Practices - No issues found.
Priority
Description and URL
Guideline and Line#
Count
Priority 1
3 issues on 1 pages
Google recommends separating keywords in URLs by dashes instead of underscores.
Google
1 pages
Offer an HTML site map to your users with links that point to the important parts of your site. Links embedded in
Google Bing
1 pages
Done
Type here to search
ENG
20:07
12-05-2020


Web standards report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial
File View Go Check Help
Back Forward Stop Refresh Home Check Open Save Print Help
Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.W3C.htm
Go
Summary
Issues
file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html
Errors
Accessibility
Compatibility
Search
Standards
Usability
This tab shows pages that do not comply with W3C standards.
W3C HTML/XHTML Validation - All pages valid.
W3C CSS Validation - All pages valid.
W3C Deprecated Features - No issues found.
Priority
Description and URL
Guideline and Line#
Count
Expand all 0 issues
Done
Type here to search
ENG
20:08
12-05-2020

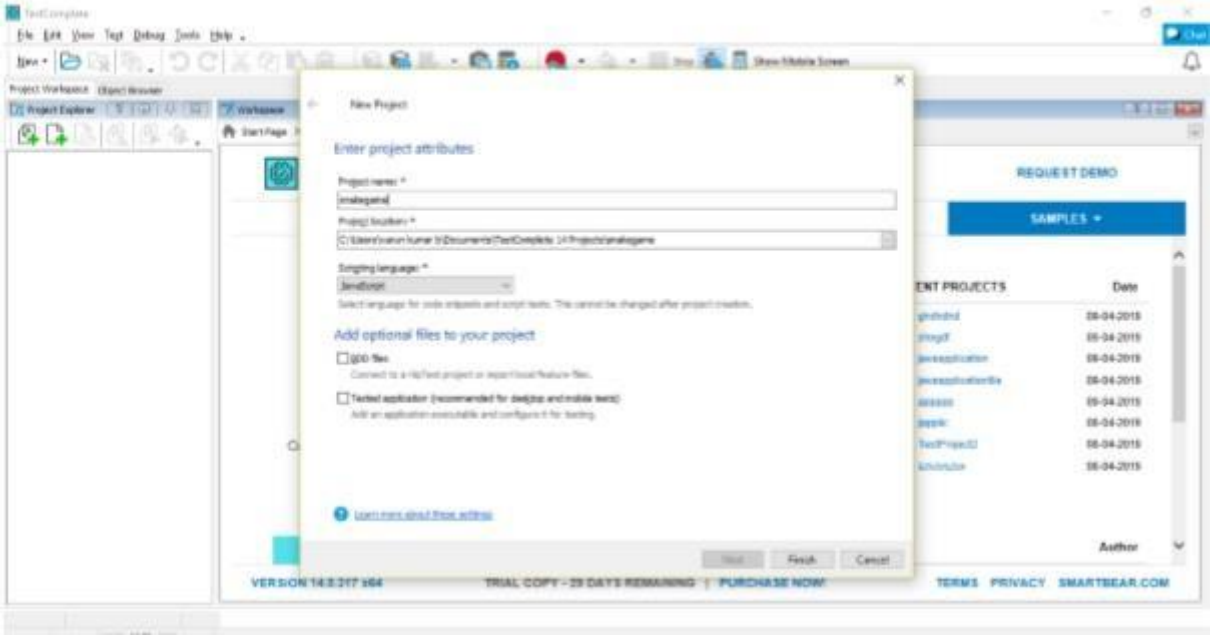

Usability report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial
File View Go Check Help
Back Forward Stop Refresh Home Check Open Save Print Help
Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.USE.htm
Go
Summary
Issues
file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html
Errors
Accessibility
Compatibility
Search
Standards
Usability
This tab shows general usability issues, indicating navigation problems for all users.
This tab is available in SortSite Professional, SortSite Developer and OnDemand, but not in SortSite Standard edition.
Usability.gov Guidelines - Some pages violate these guidelines.
W3C Best Practices - No issues found.
Readability - No issues found.
Priority
Description and URL
Guideline and Line#
Count
Priority 2
2 issues on 1 pages
Do not create or direct users into pages that have no navigational options. No links out of these pages found.
Usability.gov 7:1
1 pages
Provide a search option on each page of content-rich web sites.
Usability.gov 17:4
1 pages
Expand all 2 issues
Done
Type here to search
ENG
20:08
12-05-2020


Broken link report for file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html - SortSite Trial
File View Go Check Help
Back Forward Stop Refresh Home Check Open Save Print Help
Address C:\Users\ksr\AppData\Local\Temp\SortSite4732\PowerMapper\Map1\map.ERR.htm
Go
Summary
Issues
file://C:/Users/ksr/AppData/Local/Temp/Temp1_BASIC_SNAKE_GAME_IN_JAVASCRIPT_WITH_SOURCE_CODE.zip/basic-snake-game/index.html
Errors
Accessibility
Compatibility
Search
Standards
Usability
This tab shows site quality issues, including broken links and server configuration problems.
Broken links - No issues found.
Server configuration - No issues found.
ASP, ASP.NET and PHP script errors - No issues found.
Internet RFCs - No issues found.
Priority
Description and URL
Guideline and Line#
Count
Expand all 0 issues
Done
Type here to search
ENG
20:09
12-05-2020

TESTING SNAKE GAME USING TEST COMPLETE:

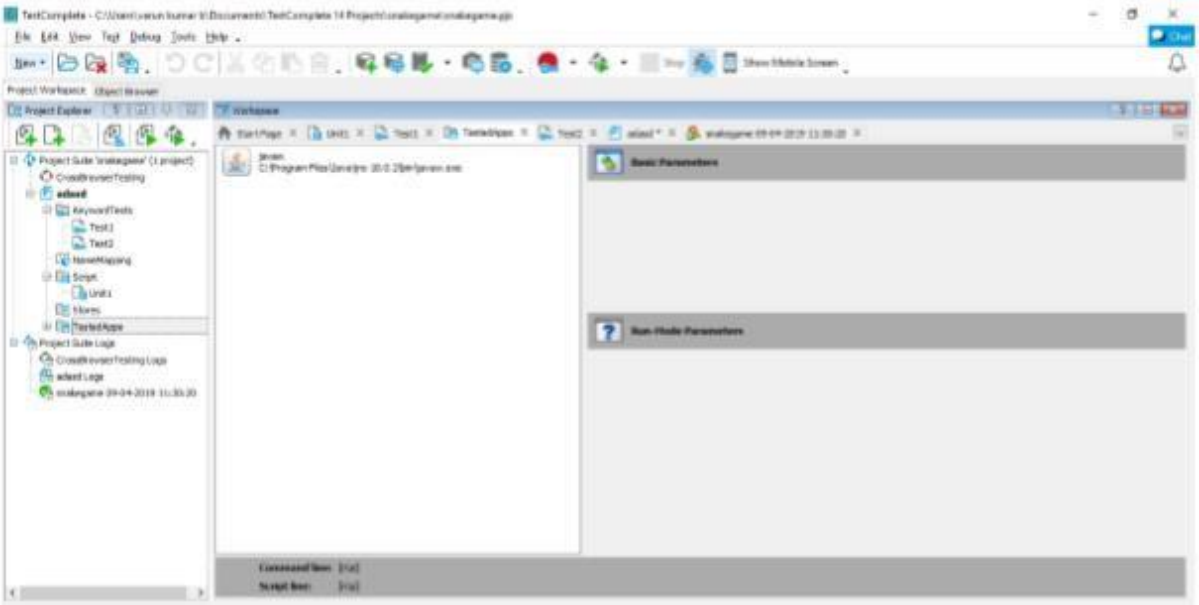
Step:1

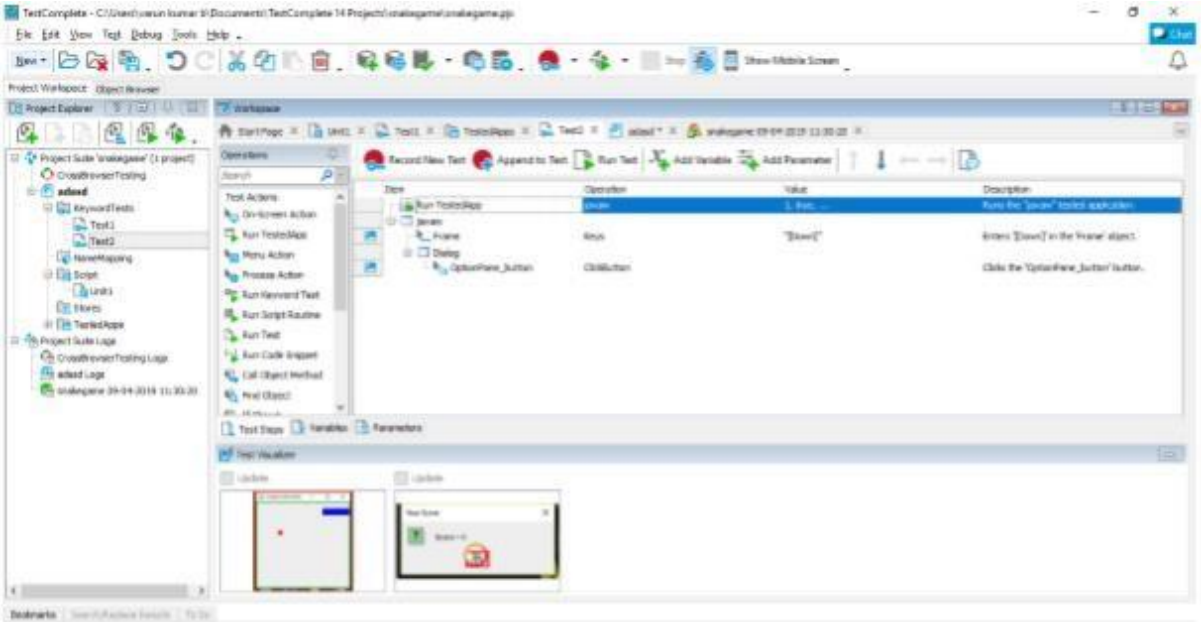
Ctreating of the project



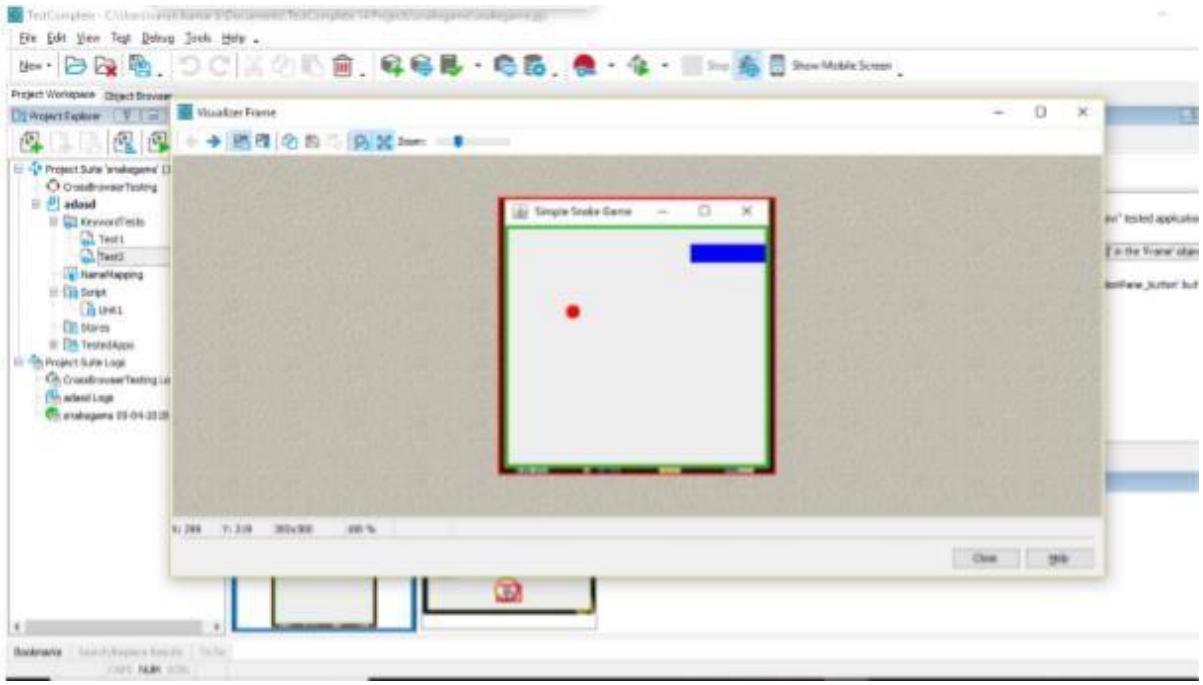
Step:2

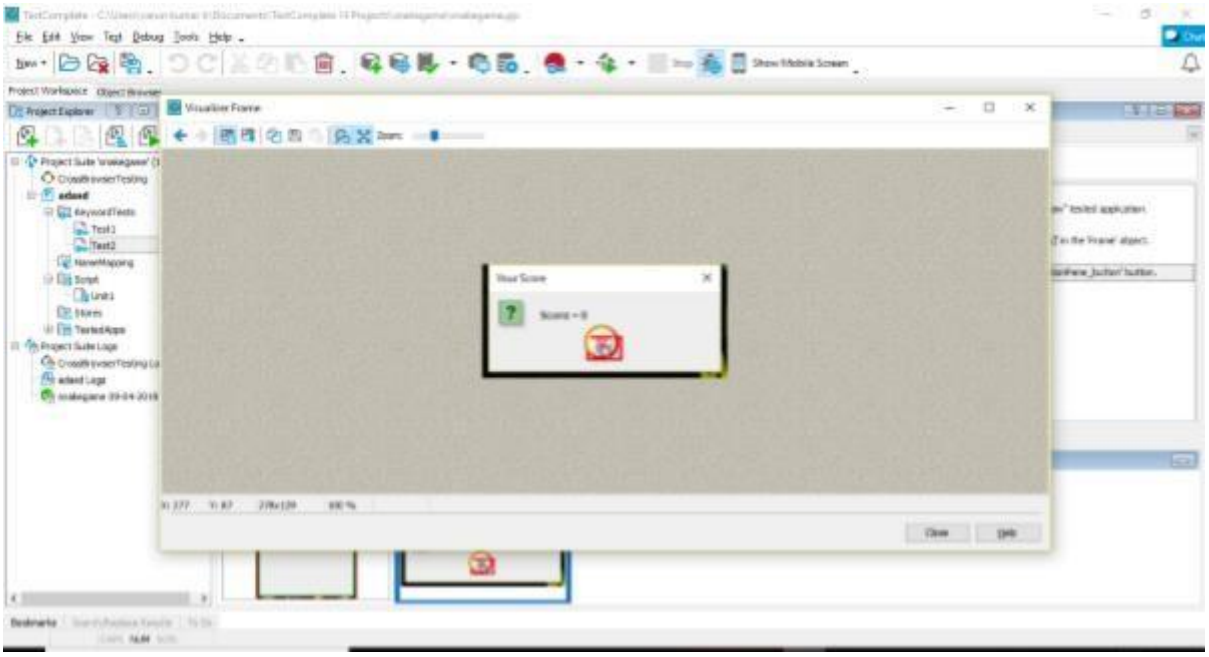
We saved the project and saved in the form of jar app

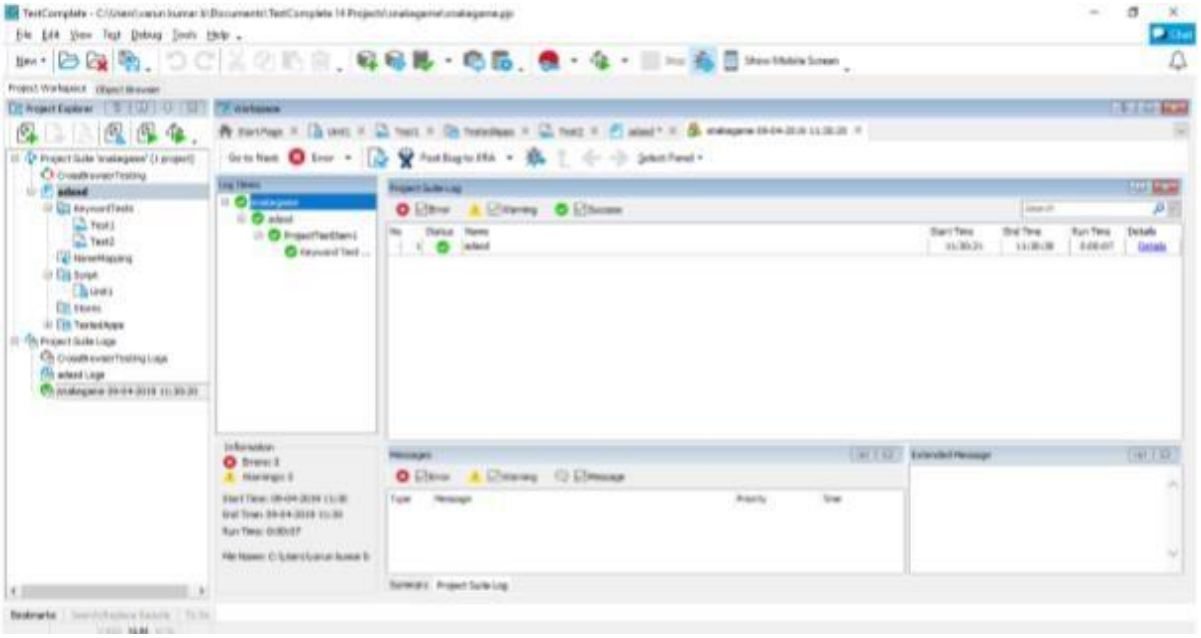




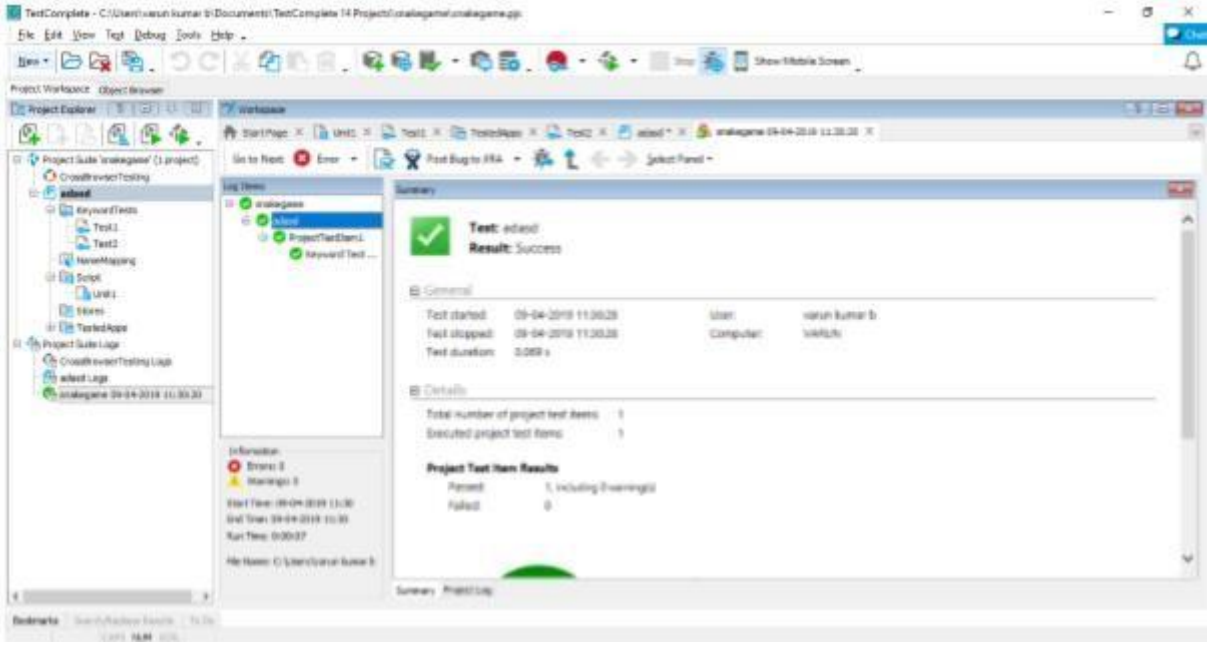
Test cases zoomed view

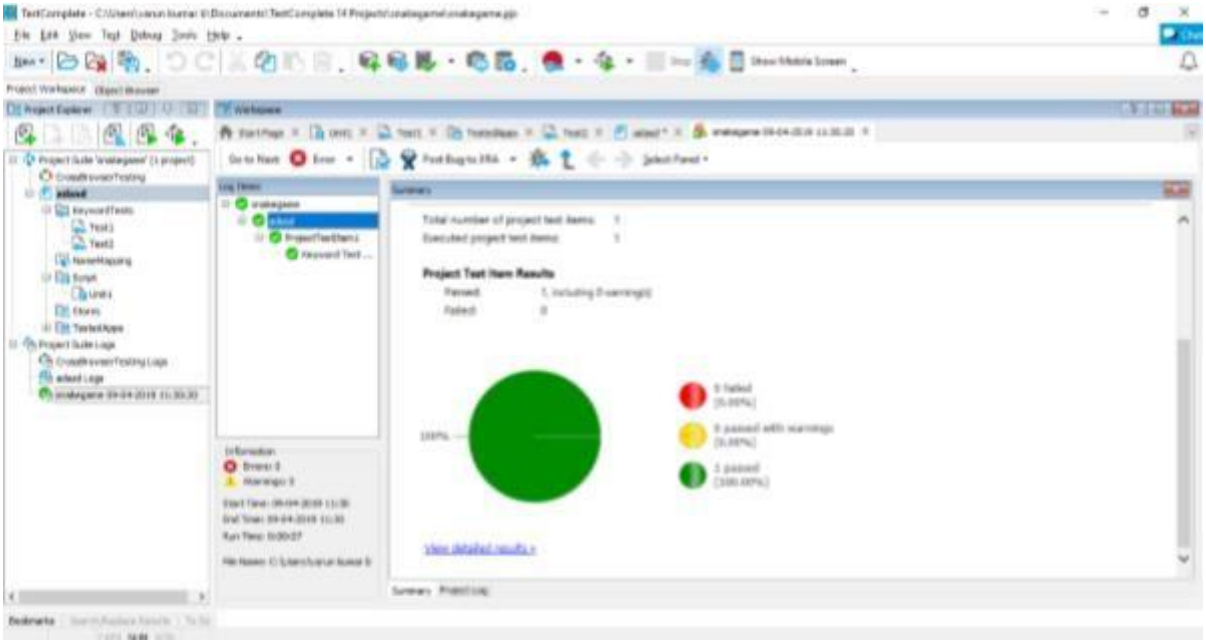

Visualizer Frame
Simple Snake Game


Visualizer Frame
Your Score



Test cases runned successfully




Project Test Item Results