RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2023 -2024

S.Y. B. SC.(I.T.) SEM-IV

PROJECT: JOU AND FUN ZONE

Submitted by	Review By
Naivedya Tripathi	Radhika Ma'am
Roll Number: 6217	

Index

Chapter 1
Introduction

About the Project:
Our project is a dynamic web application that showcases the integration of HTML, CSS, JavaScript, and PHP to create interactive and engaging features. It includes classic games like Snake and Tic Tac Toe, as well as a creative tool, Paint, allowing users to create and download artwork. The project aims to demonstrate the versatility of web development technologies and provide a platform for learning and entertainment.

Who Will Use the Project:
The project is designed for a wide audience, including casual users looking for entertainment, aspiring web developers seeking to learn new skills, and educators interested in interactive web development examples. The games and creative tools appeal to users of all ages and backgrounds, making the project accessible and engaging for a diverse audience.

Feasibility of the Project:
The project is highly feasible, as it leverages widely-used technologies like HTML, CSS, JavaScript, and PHP, which have extensive documentation and community support. The development of the project is within reach for most developers, and the features are practical and functional, offering value to users.

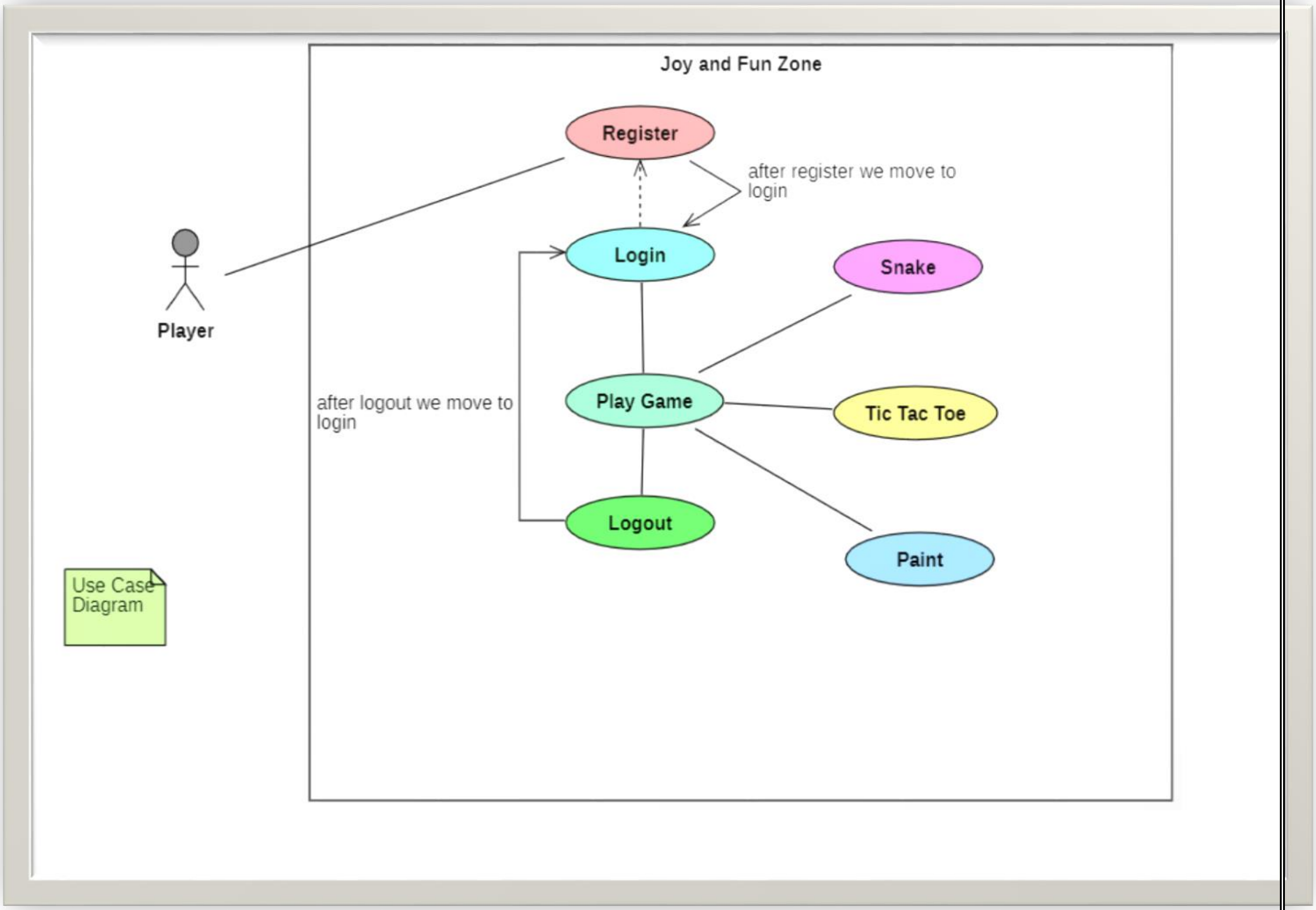
Software and Hardware Requirements:
- Software: A web browser (e.g., Chrome, Firefox, Safari) to access the website.
- Hardware: A computer or mobile device with internet connectivity to run the web application.

Technology Used:
- HTML: Defines the structure of web pages.
- CSS: Styles the HTML elements to enhance visual appeal.
- JavaScript: Adds interactivity and dynamic behavior to the web pages.
- PHP: Handles server-side processing and database interactions for dynamic content.
- MySQL (optional): Used for storing and retrieving data, such as high scores in games.

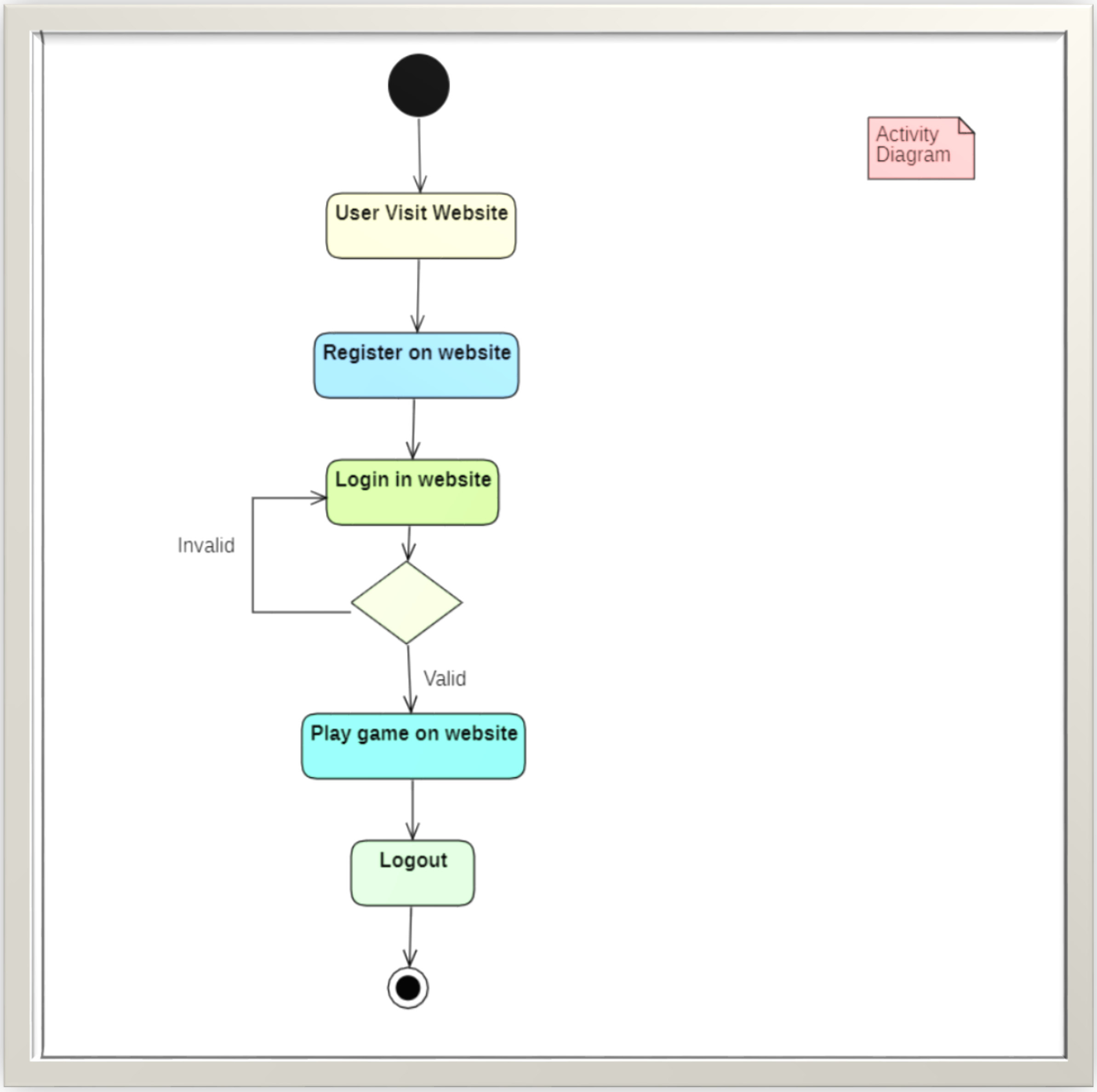
The project's technology stack is chosen for its compatibility, versatility, and ability to create rich, interactive web experiences.

Chapter 2
Design(Using Star UML)

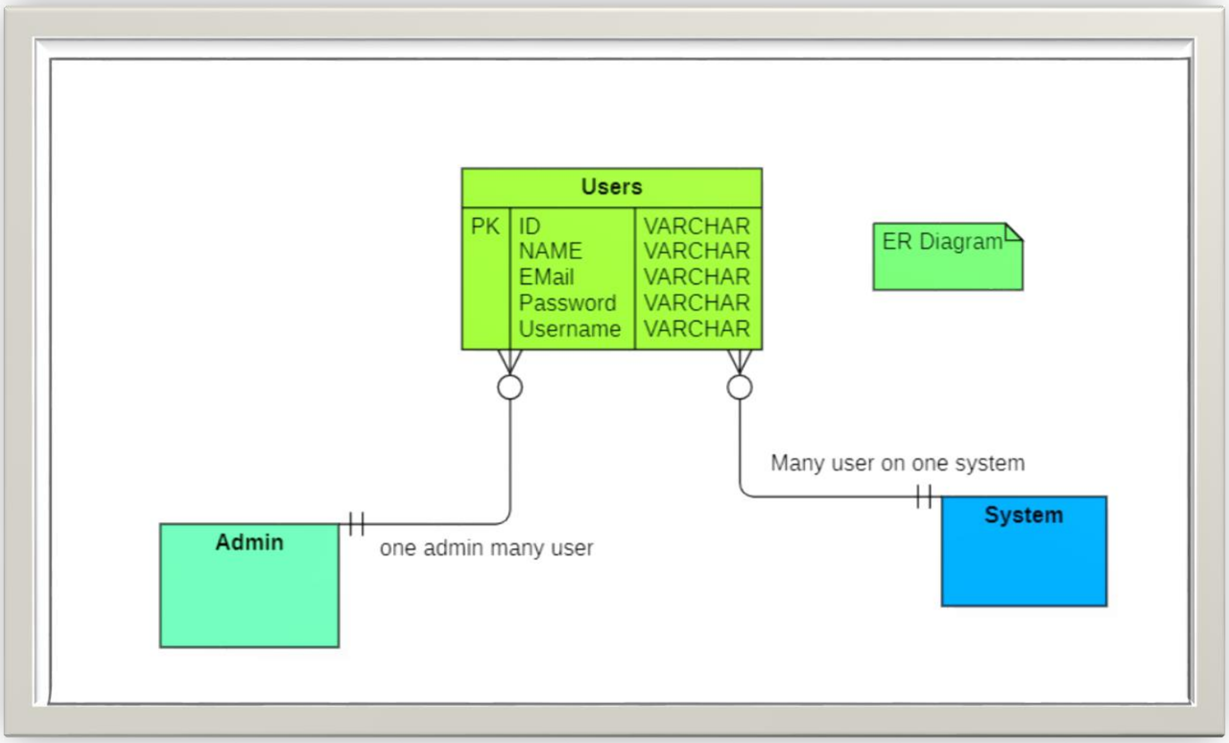
Use Case Diagram



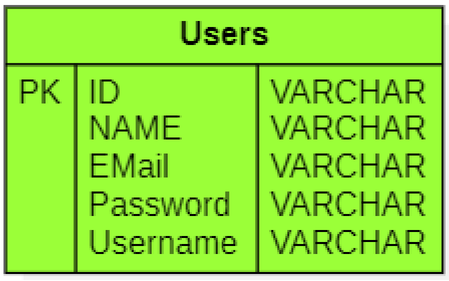
Activity Diagram:

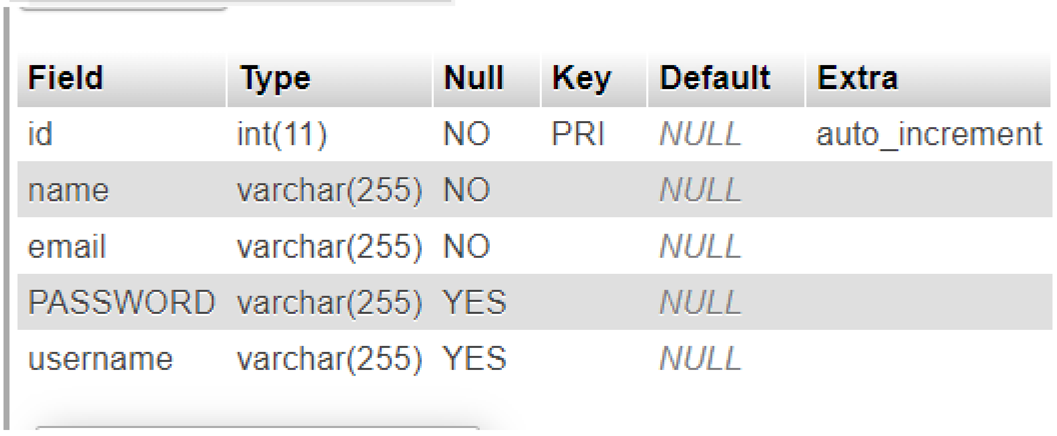


Entity Relationship Diagram:



Database Schema:



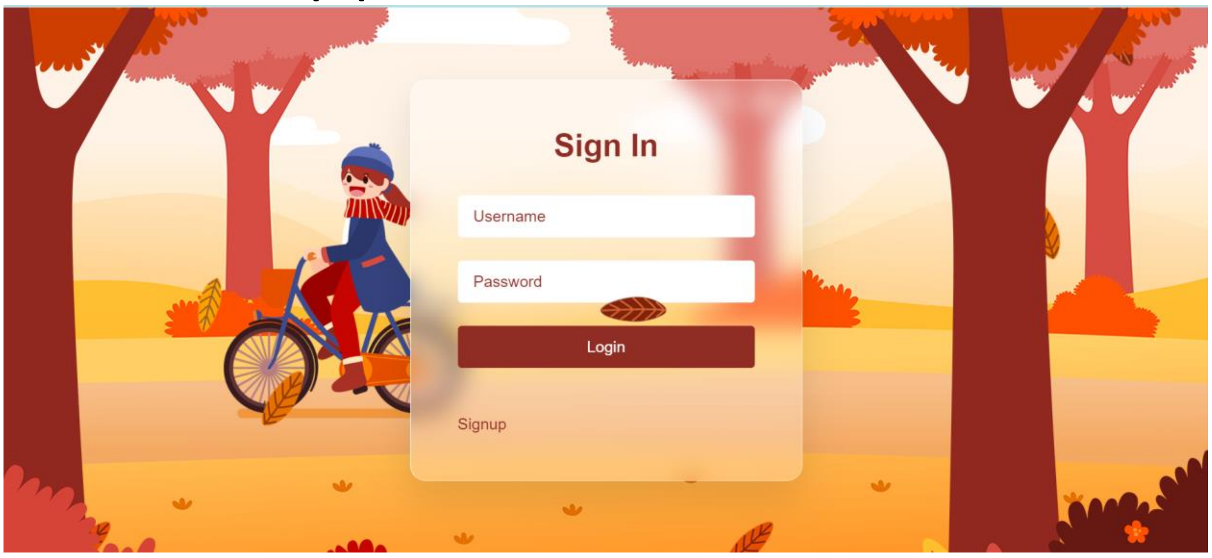


Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
name	varchar(255)	NO		*NULL*	
email	varchar(255)	NO		*NULL*	
PASSWORD	varchar(255)	YES		*NULL*	
username	varchar(255)	YES		*NULL*	

Chapter 3
Implementation

Login Page

File Name: index.php



Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title> Login Page </title>
   <link rel="stylesheet" href="style1.css">
</head>
<body>
   <section>
      <div class="leaves">
         <div class="set">
            <div><img src="leaf_01.png"></div>
            <div><img src="leaf_02.png"></div>
            <div><img src="leaf_03.png"></div>
            <div><img src="leaf_04.png"></div>
            <div><img src="leaf_01.png"></div>
            <div><img src="leaf_02.png"></div>
            <div><img src="leaf_03.png"></div>
```

```
                <div><img src="leaf_04.png"></div>
            </div>
        </div>
        <img src="bg.jpg" class="bg">
        <img src="girl.png" class="girl">
        <img src="trees.png" class="trees">
        <div class="login">
            <h2>Sign In</h2>
            <form action="loginn.php" method="POST">
                <div class="inputBox">
                    <input type="text" name="username" placeholder="Username">
                </div>
                <div class="inputBox">
                    <input type="password" name="password"
placeholder="Password">
                </div>
                <div class="inputBox">
                    <input type="submit" value="Login" id="btn">
                </div>
            </form>
            <div class="group">
                <a href="indexx.php">Signup</a>
            </div>
        </div>

    </section>
</body>
</html>
```

File Name: Loginn.php

```
<?php
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "contact_form_db";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get username and password from the form
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Query to check if the username and password match
    $sql = "SELECT * FROM contact_messages WHERE username='$username'
AND PASSWORD='$password'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        // Redirect to success page if login is successful
        session_start();

        $_SESSION["username"] = $username;


        header("Location: main.php");
        exit();
    } else {
        // Display error message if login fails
        echo "Invalid username or password";
    }
}

$conn->close();
```

File Name: Style1.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}

section {
  position: relative;
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100vh;
  overflow: hidden;
}

section .bg {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  object-fit: cover;
  pointer-events: none;
}

section .trees {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  object-fit: cover;
  z-index: 100;
  pointer-events: none;
}

section .girl {
  position: absolute;
  scale: 0.65;
  pointer-events: none;
  animation: animateGirl 10s linear infinite;
}

@keyframes animateGirl {
  0% {
    transform: translateX(calc(100% + 100vw));
  }
  50% {
    transform: translateX(calc(-100% - 100vw));
  }
  50.01% {
    transform: translateX(calc(-100% - 100vw)) rotateY(180deg);
  }
  100% {
    transform: translateX(calc(100% + 100vw)) rotateY(180deg);
  }
}

.login {
  position: relative;
  padding: 60px;
  background: rgba(255, 255, 255, 0.25);
  backdrop-filter: blur(15px);
  border: 1px solid fff;
  border-bottom: 1px solid rgba(255, 255, 255, 0.5);
  border-right: 1px solid rgba(255, 255, 255, 0.5);
  border-radius: 20px;
  width: 500px;
  display: flex;
  flex-direction: column;
  gap: 30px;
  box-shadow: 0 25px 50px rgba(0, 0, 0, 0.1);
}

.login h2 {
  position: relative;
  width: 100%;
  text-align: center;
  font-size: 2.5em;
  font-weight: 600;
  color: 8f2c24;
  margin-bottom: 10px;
}

.login .inputBox {
  position: relative;
```

```
}

.login .inputBox input {
  position: relative;
  width: 100%;
  padding: 15px 20px;
  outline: none;
  font-size: 1.25em;
  color: 8f2c24;
  border-radius: 5px;
  background: fff;
  border: none;
  margin-bottom: 30px;
}

.login .inputBox ::placeholder {
  color: 8f2c24;
}

.login .inputBox btn {
  position: relative;
  border: none;
  outline: none;
  background: 8f2c24;
  color: fff;
  cursor: pointer;
  font-size: 1.25em;
  font-weight: 500;
  transition: 0.5s;
}

.login .inputBox btn:hover {
  background: d64c42;
}

.login .group {
  display: flex;
  justify-content: space-between;
}
```

```
.login .group a {
  font-size: 1.25em;
  color: 8f2c24;
  font-weight: 500;
  text-decoration: none;
}

.login .group a:nth-child(2) {
  text-decoration: underline;
}

.leaves {
  position: absolute;
  width: 100%;
  height: 100vh;
  overflow: hidden;
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1;
  pointer-events: none;
}

.leaves .set {
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  pointer-events: none;
}

.leaves .set div {
  position: absolute;
  display: block;
}

.leaves .set div:nth-child(1) {
  left: 20%;
  animation: animate 20s linear infinite;
```

```
}

.leaves .set div:nth-child(2) {
  left: 50%;
  animation: animate 14s linear infinite;
}

.leaves .set div:nth-child(3) {
  left: 70%;
  animation: animate 12s linear infinite;
}

.leaves .set div:nth-child(4) {
  left: 5%;
  animation: animate 15s linear infinite;
}

.leaves .set div:nth-child(5) {
  left: 85%;
  animation: animate 18s linear infinite;
}

.leaves .set div:nth-child(6) {
  left: 90%;
  animation: animate 12s linear infinite;
}

.leaves .set div:nth-child(7) {
  left: 15%;
  animation: animate 14s linear infinite;
}

.leaves .set div:nth-child(8) {
  left: 60%;
  animation: animate 15s linear infinite;
}

@keyframes animate {
  0% {
    opacity: 0;
    top: -10%;
    transform: translateX(20px) rotate(0deg);
  }
  10% {
    opacity: 1;
  }
  20% {
    transform: translateX(-20px) rotate(45deg);
  }
  40% {
    transform: translateX(-20px) rotate(90deg);
  }
  60% {
    transform: translateX(20px) rotate(180deg);
  }
  80% {
    transform: translateX(-20px) rotate(45deg);
  }
  100% {
    top: 110%;
    transform: translateX(20px) rotate(225deg);
  }
}
```

Explanation:

HTML:

1. The HTML document starts with the `<!DOCTYPE html>` declaration, specifying the document type and language.
2. Within the `<head>` section, there's a meta tag specifying the character encoding and a title for the page.
3. It includes a link to an external stylesheet named "style1.css".
4. Inside the `<body>` tag, there's a `<section>` element containing the login form and background images.
5. Background images, including the girl image and trees image, are placed within the `<section>` element.
6. The login form is structured using `<div>` elements with appropriate classes.
7. The form has fields for username and password, along with a submit button.
8. The form action is set to "loginn.php", indicating that the form data will be submitted to a PHP script for processing.

9. There’s also a link for signing up which redirects to “16ndex.php”.

CSS:

1. The CSS code resets some default styling properties using the universal selector `*`.
2. The `<section>` element is styled to occupy the entire viewport with flexbox alignment.
3. Background images are positioned absolutely to cover the entire section.
4. The girl image is animated to move horizontally across the screen using keyframes.
5. The login form is styled with a background color, border, border radius, and padding to create a card-like appearance.
6. Input fields are styled for width, padding, border, and background color.
7. Placeholder text inside input fields is styled.
8. The login button is styled with background color, text color, and cursor pointer.
9. Hover effect is added to the login button using a transition effect.
10. The link for signup is styled with font size, color, and underline.
11. Leaves animation is applied using keyframes, with individual leaves positioned at different points on the screen and animated to simulate falling motion.

PHP

This PHP code establishes a connection to a MySQL database, processes form data submitted via POST request, and handles user authentication based on the entered username and password. Here's an explanation of each section:

1. Database Connection:
   - It defines variables for database connection such as server name, username, password, and database name.
   - It then attempts to establish a connection to the MySQL database using the `mysqli` class.

2. Handling Form Submission:
   - It checks if the form is submitted using `$_SERVER["REQUEST_METHOD"]` to ensure that it's a POST request.
   - If the form is submitted, it retrieves the entered username and password from the `$_POST` superglobal.

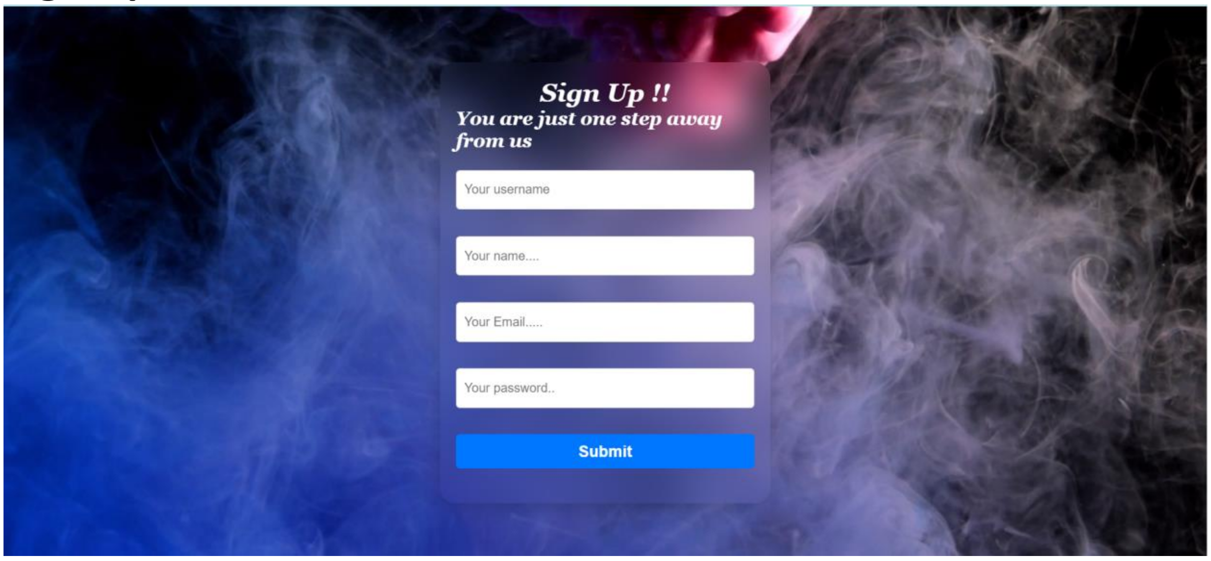
3. Query Execution:
   - It constructs an SQL query to check if there's any row in the `contact_messages` table where the username and password match the submitted values.
   - It then executes this query using `$conn->query($sql)` to fetch the result.

4. Authentication:
   - If the query returns any rows (`$result->num_rows > 0`), it indicates that the username and password combination is correct.
   - In such a case, it initiates a session using `session_start()` and assigns the username to a session variable `$_SESSION["username"]`.
   - It then redirects the user to a success page named `main.php` using `header("Location: main.php")`. The `exit()` function ensures that further code execution is stopped.

5. Handling Failed Authentication:
   - If the query returns no rows, it means that the provided username or password is incorrect.
   - In this case, it echoes "Invalid username or password" to inform the user about the authentication failure.

6. Closing Database Connection:
   - Finally, it closes the database connection using `$conn->close()` to free up resources.

Sign-up:



File Name: indexx.php

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Contact Messages </title>
   <link rel="stylesheet" href="style.css">
</head>

<body>
   <video autoplay muted loop id="myVideo">
      <source src="bg.mp4" type="video/mp4">
   </video>

   <div class="container">
      <div class="label">
         <h1> Sign Up !!</h1>
         <h2> You are just one step away from us</h2>
      </div>
      <form action="process_form.php" id="contactForm" method="POST">
   <div class="alert">Your message sent</div>
   <div class="alert1">Your message not sent</div>

   <div class="inputBox">
      <input type="text" id="username" name="username" placeholder="Your
username">
   </div>

   <div class="inputBox">
      <input type="text" id="name" name="name" placeholder="Your
name....">
   </div>

   <div class="inputBox">
      <input type="email" id="emailid" name="email" placeholder="Your
Email.....">
   </div>
```

```
  <div class="inputBox">
    <input type="password" id="password" name="password"
placeholder="Your password..">
  </div>

  <div class="inputBox">
    <button type="submit">Submit</button>
  </div>
</form>

  </div>
</body>

</html>
```

File Name: process_form.php

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Check if the required fields are set
  if (isset($_POST["username"], $_POST["name"], $_POST["email"],
$_POST["password"])) {
    // Get the form data
    $username = trim($_POST["username"]);
    $name = trim($_POST["name"]);
    $email = trim($_POST["email"]);
    $password = trim($_POST["password"]);

    // Validate the form data (e.g., check if fields are not empty)

    $servername = "localhost";
    $db_username = "root";
    $db_password = "";
    $db_name = "contact_form_db";

    // Create a connection
    $conn = new mysqli($servername, $db_username, $db_password,
$db_name);
```

```
        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        // Insert data into the database
        $sql = "INSERT INTO contact_messages (name,email,
PASSWORD,username) VALUES ('$name', '$email', '$password', '$username')";
        if ($conn->query($sql) === TRUE) {
            header("Location: index.php");
            exit();
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }

        $conn->close();
    } else {
        echo "One or more required fields are missing";
    }
} else {
    echo "Form submission method not allowed";
}
?>
```

File Name: style.css

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
myVideo {
    position: fixed;
    right: 0;
    bottom: 0;
    min-width: 100%;
    min-height: 100%;
    overflow: hidden;
  }

body{
    width: auto;
    height: auto;
    display: flex;
    justify-content: center;
    align-items: center;
}
.label{
    color: FFF;
    align-items: center;
    display: flex;
    justify-content: center;
    flex-direction: column;
    font-family: Georgia, 'Times New Roman', Times, serif;
    font-style: italic;
}

.container{
    z-index: 1;
    margin: 70px;
    width: 60vh;
    height: 80vh;
    padding: 20px;
    border-radius: 20px;
    box-shadow:  0px 5px 25px rgba(0,0,0,0.2);
    display: flex;
    justify-content:center;
    align-items: center;
    flex-direction: column;
    backdrop-filter: blur(20px);
}

.container form{
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
    flex-direction: column;
}

.inputBox{
    width: 100%;
    margin: 10px 0;
    border-radius: 5px;
    overflow: hidden;
}

.inputBox input[type="text"], .inputBox input[type="email"],.inputBox
textarea[type="textarea"]{
    width: 100%;
    height: 50px;
    border-radius: 5px;
    border: none;
    outline: none;
    overflow: hidden;
    border: 1px solid rgba(0,0,0,0.2);
    padding: 0px 10px;
    font-size: 16px;
    color: 444;
}

.inputBox password{
    width: 100%;
    height: 50px;
    border-radius: 5px;
    border: none;
    outline: none;
    overflow: hidden;
    border: 1px solid rgba(0,0,0,0.2);
    padding: 0px 10px;
    font-size: 16px;
    color: 444;
}

.inputBox button{
    width: 100%;
    padding: 10px 20px;
    border: none;
    outline: none;
    background: rgb(0, 119, 255);
    color: FFF;
```

```
    font-size: 20px;
    font-weight: bold;
    cursor: pointer;

}

.inputBox button:hover{
    background: rgb(0, 17, 255);
    transition: all 0.3s ease;
}

::placeholder{
    font-size: 16px;
}

.alert{
    width: 100%;
    background: rgb(0, 255, 106);
    padding: 10px 20px;
    border-radius: 5px;
    text-align: center;
    font-size: 18px;
    font-weight: 900;
    display: none;
}
.alert1{
    width: 100%;
    background: rgb(255, 4, 0);
    padding: 10px 20px;
    border-radius: 5px;
    text-align: center;
    font-size: 18px;
    font-weight: 900;
    display: none;
}
@keyframes AnimateBlink{
    0%{
        background: 0ef;
    }
    100%{
        background: 0ef;
    }
}
```

Explanation:

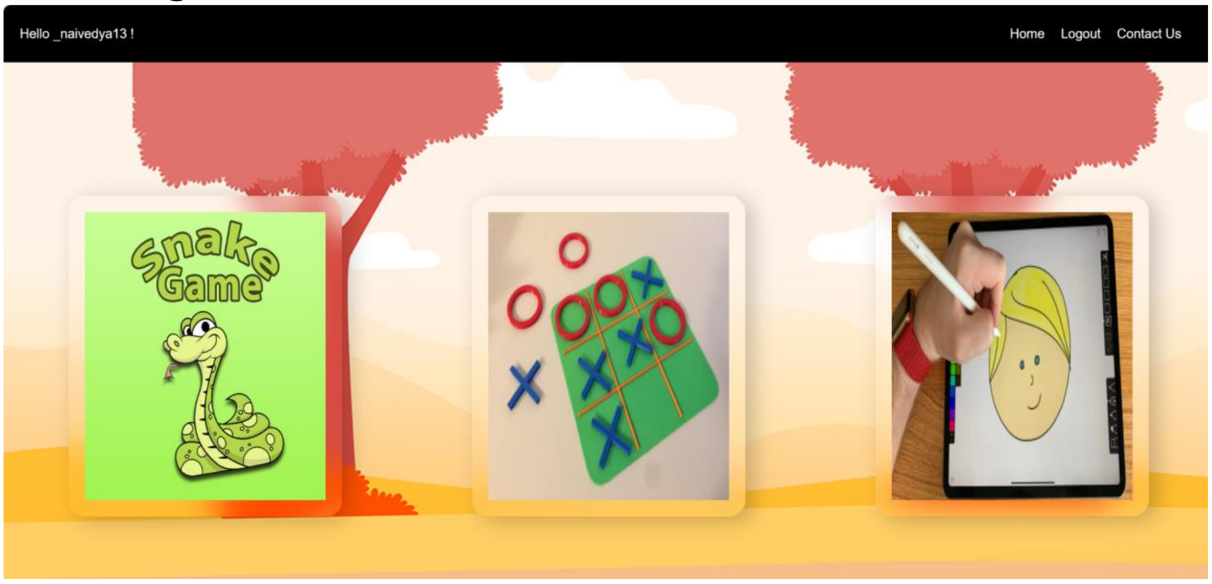
This set of files constitutes a simple signup form with validation and submission handling using PHP and styling with CSS. Let's break down each file:

1. indexx.php:
   - This file contains the HTML structure of the signup form.
   - It includes a background video element (`<video>`) with a source file (`bg.mp4`).
   - Inside the `<div class="container">`, there's a form with fields for username, name, email, and password.
   - Each input field is wrapped inside a div with the class `inputBox`.
   - There's a submit button at the end of the form.
   - This file references an external stylesheet named `style.css`.

2. process_form.php:
   - This PHP script handles form submission.
   - It checks if the request method is POST and if required fields (username, name, email, password) are set.
   - If the required fields are set, it retrieves the form data using `$_POST`.
   - It then establishes a connection to a MySQL database and inserts the form data into a table named `contact_messages`.
   - If the insertion is successful, it redirects the user to `index.php`.
   - Otherwise, it outputs an error message.
   - This script ensures that form submission method is POST and handles database interaction securely.

3. style.css:
   - This CSS file contains styles for the signup form and its elements.
   - It resets default styles using `*` selector.
   - The background video (`myVideo`) is styled to cover the entire viewport.
   - The body is styled to center the container vertically and horizontally.
   - Styles for the container (`div.container`), form, input fields, and submit button are defined.
   - There are specific styles for input fields, buttons, and placeholders.
   - Error and success messages (`div.alert`, `div.alert1`) are styled with background colors.
   - Additionally, there's an animation (`@keyframes AnimateBlink`) defined, but it's not applied to any element in the provided HTML.

Main Page:



File Name: main.php

```
<?php
session_start();

// Check if the user is logged in
if (isset($_SESSION['username'])) {
    $userName = $_SESSION['username'];
} else {
    // Redirect to the login page if not logged in
    header("Location: index.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
    <link rel="stylesheet" href="main_design.css">
</head>

<body>
    <video autoplay muted loop id="myVideo">
```

```
        <source src="bg1.mp4" type="video/mp4">
    </video>
    <nav class="nav">
        <i class="uil uil-bars navOpenBtn"></i>
        <?php
        // Assuming $userName contains the user's name fetched from the
database
        echo 'Hello   ' . $userName . ' !';
        ?>
        <ul class="nav-links">
            <i class="uil uil-times navCloseBtn"></i>
            <li><a href="main.php">Home</a></li>
            <li><a href="index.php">Logout</a></li>
            <li><a href="https://forms.gle/z7wqfihe5uqwEvNZ7">Contact
Us</a></li>
        </ul>
        <i class="uil uil-search search-icon" id="searchIcon"></i>
    </nav>
    <div class="body1">
        <div class="container1">
            <a href="snake.php">
            <img id="img1"
src="https://img1.ugamezone.com/201901/2019/0130/f6/c/303093/original.jp
g" />
            </a>
        </div>
        <div class="container2">
            <a href="tic_tac_toe.php">
                <img id="img2"
src="https://i.etsystatic.com/18390435/r/il/afc0b9/2230101434/il_fullxfull.223
0101434_tsnw.jpg" />
            </a>
        </div>
        <div class="container3">
            <a href="paint.php">
                <img id="img3" src="https://9to5mac.com/wp-
content/uploads/sites/6/2018/12/Linea-sketch-iPad-Pro-drawing-
app.jpeg?quality=82&strip=all&w=1600" />
            </a>
        </div>
```

```
  </div>
</body>

</html>
```

File Name: main_design.css

```
img1 {
  height: 100%;
  width: 100%;
}
img2 {
  height: 100%;
  width: 100%;
}
img3 {
  height: 100%;
  width: 100%;
}
.body1 {
  justify-content: center;
  display: flex;
  flex-direction: row;
}
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background: url("bg.jpg");
}

.nav {
  position: fixed; /* Fix the navigation bar position */
  top: 0; /* Place it at the top of the viewport */
  left: 0;
  right: 0;
  z-index: 9999; /* Set a high z-index value */
  background-color: black;
  color: fff;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 20px;
  backdrop-filter: blur(40px);
}

.nav-links {
  z-index: 1;
  list-style: none;
  padding: 0;
  display: flex;
  justify-content: flex-end; /* Align links to the right */
  flex-grow: 1; /* Allow links to take remaining space */
}

.nav-links li {
  margin-right: 20px;
}

.nav-links a {
  text-decoration: none;
  color: fff;
}

.container1 {
  z-index: 1;
  margin: 80px;
  width: 100vh;
  height: 50vh;
  padding: 20px;
  border-radius: 20px;
  box-shadow: 10px 5px 25px rgba(0, 0, 0, 0.2);
  /* display: flex;
    flex-direction: row; /* Display items in a row */
  justify-content: center; /* Center items horizontally */
  align-items: center; /* Center items vertically */
  backdrop-filter: blur(20px);
}
.container2 {
  z-index: 1;
  margin: 80px;
  width: 100vh;
  height: 50vh;
  padding: 20px;
  border-radius: 20px;
  box-shadow: 10px 5px 25px rgba(0, 0, 0, 0.2);
  /*display: flex;
    flex-direction: row;  Display items in a row */
  justify-content: center; /* Center items horizontally */
  align-items: center; /* Center items vertically */
  backdrop-filter: blur(20px);
}
.container3 {
  z-index: 1;
  margin: 80px;
  width: 100vh;
  height: 50vh;
  padding: 20px;
  border-radius: 20px;
  box-shadow: 10px 5px 25px rgba(0, 0, 0, 0.2);
  /*display: flex;
    flex-direction: row;  Display items in a row */
  justify-content: center; /* Center items horizontally */
  align-items: center; /* Center items vertically */
  backdrop-filter: blur(20px);
}
```

Explaination:

The provided PHP file (`main.php`) is a webpage that serves as the main page of the application. It includes dynamic PHP code to check if the user is logged in and display personalized content accordingly. Below is an explanation of the PHP and CSS code in this file:

1. PHP Section:
   - `session_start()`: This function starts a new or resumes an existing session.
   - `isset($_SESSION['username'])`: This checks if the 'username' key is set in the session, indicating that the user is logged in.
   - If the user is logged in, their username is stored in the `$userName` variable.
   - If the user is not logged in, they are redirected to the login page (`index.php`).

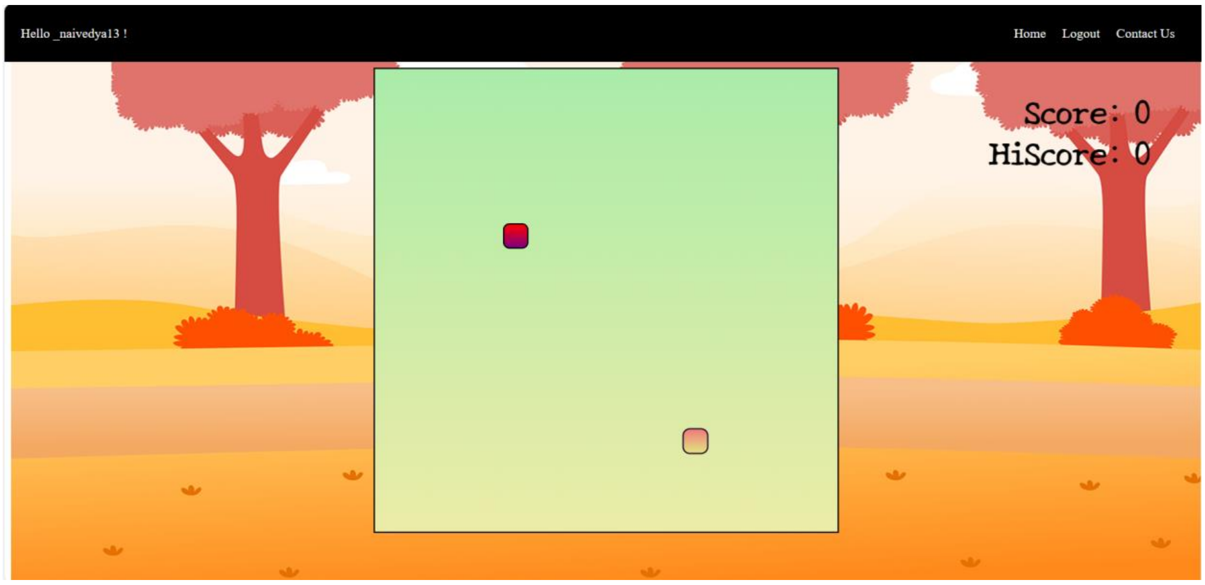
- This PHP section ensures that only authenticated users can access the main page.

2. HTML Section:
   - The HTML structure consists of a video background (`<video>`) and a navigation bar (`<nav>`).
   - The navigation bar displays the user's name and links to different sections of the website, such as Home, Logout, and Contact Us.
   - The body contains three containers (`div.container1`, `div.container2`, `div.container3`), each representing a different section or feature of the website.
   - Each container contains an image (`<img>`) wrapped in an anchor (`<a>`) tag, linking to a specific page or feature.

3. CSS Section (`main_design.css`):
   - Styles the images (`img1`, `img2`, `img3`) within the containers to ensure they fill the container size.
   - Defines the layout for the main content area (`div.body1`) using flexbox, arranging the containers in a row.
   - Sets general styles for the body, navigation bar (`nav`), navigation links (`ul.nav-links`), and individual containers (`div.container1`, `div.container2`, `div.container3`).
   - Uses CSS properties like `margin`, `padding`, `border-radius`, `box-shadow`, `justify-content`, `align-items`, and `backdrop-filter` for styling.

Games:

File Name: snake.php



```
<?php
session_start();

// Check if the user is logged in
if (isset($_SESSION['username'])) {
    $userName = $_SESSION['username'];
} else {
    // Redirect to the login page if not logged in
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Snake Game</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <nav class="nav">
        <i class="uil uil-bars navOpenBtn"></i>
        <?php
        // Assuming $userName contains the user's name fetched from the
database
        echo 'Hello   ' . $userName . ' !';
        ?>
        <ul class="nav-links">
           <i class="uil uil-times navCloseBtn"></i>
           <li><a href="main.php">Home</a></li>
           <li><a href="index.php">Logout</a></li>
           <li><a href="https://forms.gle/z7wqfihe5uqwEvNZ7">Contact
Us</a></li>
        </ul>
        <i class="uil uil-search search-icon" id="searchIcon"></i>
    </nav>
    <div class="body">
        <div id="scoreBox">Score: 0</div>
        <div id="hiscoreBox">HiScore: 0</div>
        <div id="board"></div>
    </div>
</body>
<script src="js/index.js"></script>
</html>
```

File Name: style.css

```
@import
url('https://fonts.googleapis.com/css2?family=New+Tegomin&display=swap');
.body{
    background: url("../bg.jpg");
    min-height: 100vh;
    background-size: 100vw 100vh;
    background-repeat: no-repeat;
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden;
}
.nav {
    position: fixed; /* Fix the navigation bar position */
    top: 0; /* Place it at the top of the viewport */
    left: 0;
    right: 0;
    z-index: 9999; /* Set a high z-index value */
    background-color: black;
    color: fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 20px;
    backdrop-filter: blur(40px);
}
.nav-links {
    z-index: 1;
    list-style: none;
    padding: 0;
    display: flex;
    justify-content: flex-end; /* Align links to the right */
    flex-grow: 1; /* Allow links to take remaining space */
  }

  .nav-links li {
    margin-right: 20px;
  }

  .nav-links a {
    text-decoration: none;
    color: fff;
  }
board{
    background: linear-gradient(rgb(170, 236, 170), rgb(236, 236, 167));
    width: 80vmin;
    height: 80vmin;
    border: 2px solid black;
    display: grid;
    grid-template-rows: repeat(18, 1fr);
    grid-template-columns: repeat(18, 1fr);
}

scoreBox{
    position: absolute;
    top: 100px;
    right: 70px;
    font-size: 39px;
    font-weight: bold;
    font-family: 'New Tegomin', serif;
```

```
}

hiscoreBox{
   position: absolute;
   top: 150px;
   right: 70px;
   font-size: 39px;
   font-weight: bold;
   font-family: 'New Tegomin', serif;
}

.head{
   background: linear-gradient(rgb(240, 124, 124), rgb(228, 228, 129));
   border: 2px solid rgb(34, 4, 34);
   transform: scale(1.02);
   border-radius: 9px;
}

.snake{
   background-color: purple;
   border: .25vmin solid white;
   border-radius: 12px;
}

.food{
   background: linear-gradient(red, purple);
   border: .25vmin solid black;
   border-radius: 8px;
}
```

File Name: index.js

```
// Game Constants & Variables
let inputDir = {x: 0, y: 0};
const foodSound = new Audio('music/food.mp3');
const gameOverSound = new Audio('music/gameover.mp3');
const moveSound = new Audio('music/move.mp3');
const musicSound = new Audio('music/music.mp3');
let speed = 8;
let score = 0;
let lastPaintTime = 0;
let snakeArr = [
    {x: 13, y: 15}
];

food = {x: 6, y: 7};

function main(ctime) {
    window.requestAnimationFrame(main);
    // console.log(ctime)
    if((ctime - lastPaintTime)/1000 < 1/speed){
        return;
    }
    lastPaintTime = ctime;
    gameEngine();
}
function isCollide(snake) {
    // If you bump into yourself
    for (let i = 1; i < snakeArr.length; i++) {
        if(snake[i].x === snake[0].x && snake[i].y === snake[0].y){
            return true;
        }
    }
    // If you bump into the wall
    if(snake[0].x >= 18 || snake[0].x <=0 || snake[0].y >= 18 || snake[0].y <=0){
        return true;
    }

    return false;
}
function gameEngine(){
    // Part 1: Updating the snake array & Food
    if(isCollide(snakeArr)){
        gameOverSound.play();
        musicSound.pause();
        score = 0;
        inputDir =  {x: 0, y: 0};
        alert("Game Over. Press any key to play again!");
        snakeArr = [{x: 13, y: 15}];
        musicSound.play();
    }
    // If you have eaten the food, increment the score and regenerate the food
    if(snakeArr[0].y === food.y && snakeArr[0].x ===food.x){
        foodSound.play();
        score += 1;
        if(score>hiscoreval){
            hiscoreval = score;
            localStorage.setItem("hiscore", JSON.stringify(hiscoreval));
            hiscoreBox.innerHTML = "HiScore: " + hiscoreval;
        }
        scoreBox.innerHTML = "Score: " + score;
        snakeArr.unshift({x: snakeArr[0].x + inputDir.x, y: snakeArr[0].y +
inputDir.y});
        let a = 2;
        let b = 16;
        food = {x: Math.round(a + (b-a)* Math.random()), y: Math.round(a + (b-a)*
Math.random())}
    }

    // Moving the snake
    for (let i = snakeArr.length - 2; i>=0; i--) {
        snakeArr[i+1] = {...snakeArr[i]};
    }

    snakeArr[0].x += inputDir.x;
    snakeArr[0].y += inputDir.y;

    // Part 2: Display the snake and Food
    // Display the snake
    board.innerHTML = "";
    snakeArr.forEach((e, index)=>{
        snakeElement = document.createElement('div');
        snakeElement.style.gridRowStart = e.y;
        snakeElement.style.gridColumnStart = e.x;

        if(index === 0){
            snakeElement.classList.add('head');
        }
        else{
            snakeElement.classList.add('snake');
        }
        board.appendChild(snakeElement);
    });
    // Display the food
    foodElement = document.createElement('div');
    foodElement.style.gridRowStart = food.y;
    foodElement.style.gridColumnStart = food.x;
    foodElement.classList.add('food')
    board.appendChild(foodElement);
}
// Main logic starts here
musicSound.play();
let hiscore = localStorage.getItem("hiscore");
if(hiscore === null){
    hiscoreval = 0;
    localStorage.setItem("hiscore", JSON.stringify(hiscoreval))
}
else{
    hiscoreval = JSON.parse(hiscore);
    hiscoreBox.innerHTML = "HiScore: " + hiscore;
}

window.requestAnimationFrame(main);
window.addEventListener('keydown', e =>{
    inputDir = {x: 0, y: 1} // Start the game
    moveSound.play();
    switch (e.key) {
        case "ArrowUp":
            console.log("ArrowUp");
            inputDir.x = 0;
            inputDir.y = -1;
            break;

        case "ArrowDown":
            console.log("ArrowDown");
            inputDir.x = 0;
            inputDir.y = 1;
            break;

        case "ArrowLeft":
            console.log("ArrowLeft");
            inputDir.x = -1;
            inputDir.y = 0;
            break;

        case "ArrowRight":
            console.log("ArrowRight");
            inputDir.x = 1;
            inputDir.y = 0;
            break;
        default:
            break;
    }

});
```

Explanation:

The provided PHP file (`snake.php`) is a part of a web application that implements a simple snake game. Below is an explanation of the PHP, HTML, CSS, and JavaScript code in this file:

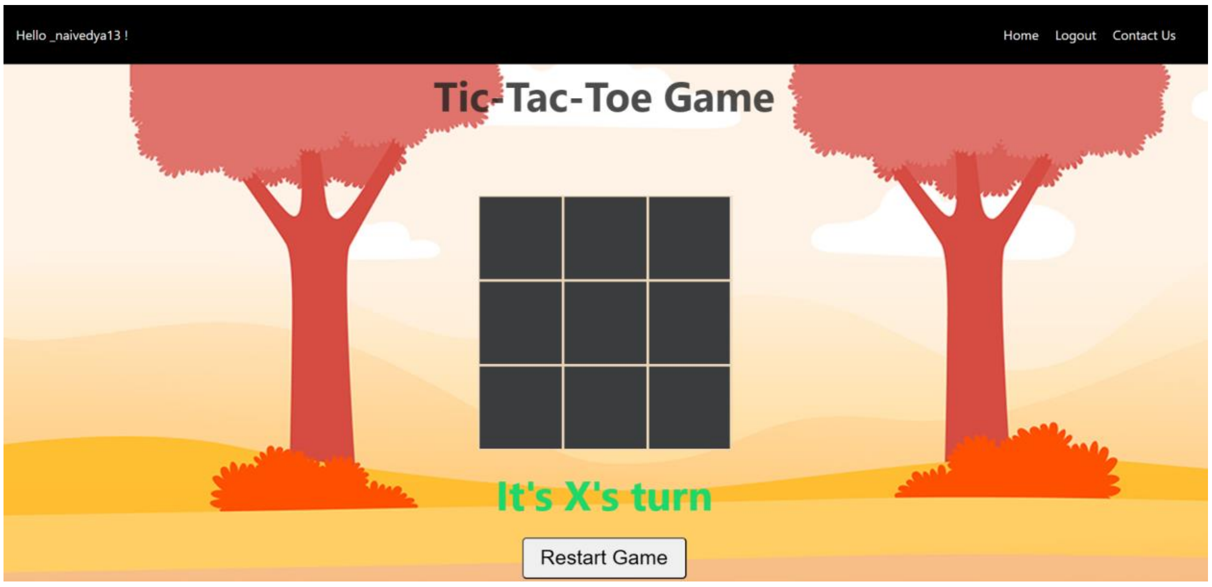
1. PHP Section:
   - The PHP section starts with `session_start()` to begin or resume a session.
   - It checks if the user is logged in by verifying the existence of the 'username' session variable.
   - If the user is not logged in, they are redirected to the login page (`login.php`) using `header("Location: login.php")`.
   - The `exit()` function is called to stop the execution of PHP code after the redirection.

2. HTML Section:
   - The HTML structure includes a navigation bar (`<nav>`) and a main container (`<div class="body">`) to hold the game canvas.
   - Inside the main container, there are two elements to display the current score (`<div id="scoreBox">`) and the highest score (`<div id="hiscoreBox">`).
   - The game canvas itself is represented by a `<div>` element with the id "board".

3. CSS Section (`style.css`):
  - Styles the background, navigation bar, and other elements for visual presentation.
  - Defines the layout and appearance of the game board and score display.

4. JavaScript Section (`index.js`):
  - Defines game constants and variables, such as the snake's initial position, food position, speed, score, and sounds.
  - The `main()` function is the main game loop that runs continuously using `window.requestAnimationFrame()`.
  - The `isCollide()` function checks for collisions between the snake and the walls or itself.
  - The `gameEngine()` function updates the game state, checks for collisions, handles scoring, and renders the game elements.
  - The game logic includes moving the snake, handling food consumption, updating the score, and checking for game over conditions.
  - Event listeners are added to handle keyboard input for controlling the snake's direction.

Tic-Tac-Toe:



File Name: tic_tac_toe.php

```
<?php
session_start();

// Check if the user is logged in
```

```
if (isset($_SESSION['username'])) {
   $userName = $_SESSION['username'];
} else {
   // Redirect to the login page if not logged in
   header("Location: login.php");
   exit();
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>X/O</title>
   <link rel="stylesheet" href="tic.css">
</head>
<body>
   <nav class="nav">
      <i class="uil uil-bars navOpenBtn"></i>
      <?php
      // Assuming $userName contains the user's name fetched from the
database
      echo 'Hello   ' . $userName . ' !';
      ?>
      <ul class="nav-links">
         <i class="uil uil-times navCloseBtn"></i>
         <li><a href="main.php">Home</a></li>
         <li><a href="index.php">Logout</a></li>
         <li><a href="https://forms.gle/z7wqfihe5uqwEvNZ7">Contact
Us</a></li>
      </ul>
      <i class="uil uil-search search-icon" id="searchIcon"></i>
   </nav>
   <section>
      <h1 class="game--title">Tic-Tac-Toe Game</h1>
      <div class="game--container">
         <div data-cell-index="0" class="cell"></div>
         <div data-cell-index="1" class="cell"></div>
         <div data-cell-index="2" class="cell"></div>
         <div data-cell-index="3" class="cell"></div>
```

```
          <div data-cell-index="4" class="cell"></div>
          <div data-cell-index="5" class="cell"></div>
          <div data-cell-index="6" class="cell"></div>
          <div data-cell-index="7" class="cell"></div>
          <div data-cell-index="8" class="cell"></div>
      </div>
      <h2 class="game--status"></h2>
      <button class="game--restart">Restart Game</button>
   </section>
   <script src="tic.js"></script>
</body>
</html>
```

File Name: tic.css

```
body {
   background : url("bg.jpg");
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
   background-color: e4e4e4;
}
.nav {
   position: fixed; /* Fix the navigation bar position */
   top: 0; /* Place it at the top of the viewport */
   left: 0;
   right: 0;
   z-index: 9999; /* Set a high z-index value */
   background-color: black;
   color: fff;
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 10px 20px;
   backdrop-filter: blur(40px);
}
.nav-links {
   z-index: 1;
   list-style: none;
   padding: 0;
   display: flex;
   justify-content: flex-end; /* Align links to the right */
   flex-grow: 1; /* Allow links to take remaining space */
```

```
  }

  .nav-links li {
    margin-right: 20px;
  }

  .nav-links a {
    text-decoration: none;
    color: fff;
  }
section {
    text-align: center;
}

.cell {
    font-family: "Permanent Marker", cursive;
    width: 100px;
    height: 100px;
    border: 2px solid ecd7ba;
    cursor: pointer;
    line-height: 100px;
    font-size: 60px;
}

.game--title {
    font-size: 50px;
    color: 1f1f1f;
    margin: 90px auto;
    opacity: .8;
}

.game--container {
    display: grid;
    grid-template-columns: repeat(3, auto);
    width: 306px;
    margin: 10px auto;
    background-color: 3a3c3e;
    color: 04c0b2;

}
```

```css
.game--status {
   font-size: 50px;
   color: 1ed86c;
   margin: 20px auto;
}

.game--restart {
   width: 200px;
   border-radius: 5px;
   height: 50px;
   font-size: 25px;
   color: 000000;

}

.game--restart:hover {
   background-color: 1ed86c;
}
```

File Name: tic.js

```javascript
const statusDisplay = document.querySelector('.game--status');

let gameActive = true;
let currentPlayer = "X";
let gameState = ["", "", "", "", "", "", "", "", ""];

const winningMessage = () => `Player ${currentPlayer} has won!`;
const drawMessage = () => `Game ended in a draw!`;
const currentPlayerTurn = () => `It's ${currentPlayer}'s turn`;

statusDisplay.innerHTML = currentPlayerTurn();

const winningConditions = [
   [0, 1, 2],
   [3, 4, 5],
   [6, 7, 8],
   [0, 3, 6],
   [1, 4, 7],
   [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
];

function handleCellPlayed(clickedCell, clickedCellIndex) {
    gameState[clickedCellIndex] = currentPlayer;
    clickedCell.innerHTML = currentPlayer;
}

function handlePlayerChange() {
    currentPlayer = currentPlayer === "X" ? "O" : "X";
    statusDisplay.innerHTML = currentPlayerTurn();
}

function handleResultValidation() {
    let roundWon = false;
    for (let i = 0; i < winningConditions.length; i++) {
        const winCondition = winningConditions[i];
        let a = gameState[winCondition[0]];
        let b = gameState[winCondition[1]];
        let c = gameState[winCondition[2]];
        if (a === '' || b === '' || c === '') {
            continue;
        }
        if (a === b && b === c) {
            roundWon = true;
            break;
        }
    }

    if (roundWon) {
        statusDisplay.innerHTML = winningMessage();
        gameActive = false;
        return;
    }

    let roundDraw = !gameState.includes("");
    if (roundDraw) {
        statusDisplay.innerHTML = drawMessage();
        gameActive = false;
        return;
    }

    handlePlayerChange();
}

function handleCellClick(clickedCellEvent) {
    const clickedCell = clickedCellEvent.target;
    const clickedCellIndex = parseInt(clickedCell.getAttribute('data-cell-index'));

    if (gameState[clickedCellIndex] !== "" || !gameActive) {
        return;
    }

    handleCellPlayed(clickedCell, clickedCellIndex);
    handleResultValidation();
}

function handleRestartGame() {
    gameActive = true;
    currentPlayer = "X";
    gameState = ["", "", "", "", "", "", "", "", ""];
    statusDisplay.innerHTML = currentPlayerTurn();
    document.querySelectorAll('.cell').forEach(cell => cell.innerHTML = "");
}

document.querySelectorAll('.cell').forEach(cell => cell.addEventListener('click',
handleCellClick));
document.querySelector('.game--restart').addEventListener('click',
handleRestartGame);
```

Explanation:

1. tic_tac_toe.php:
  - PHP Section:
    - Starts a session to maintain user authentication.
    - Checks if the user is logged in. If not, redirects to the login page.
  - HTML Section:
    - Contains the structure of the Tic-Tac-Toe game.
    - Includes navigation, game title, game board, game status, and restart button.

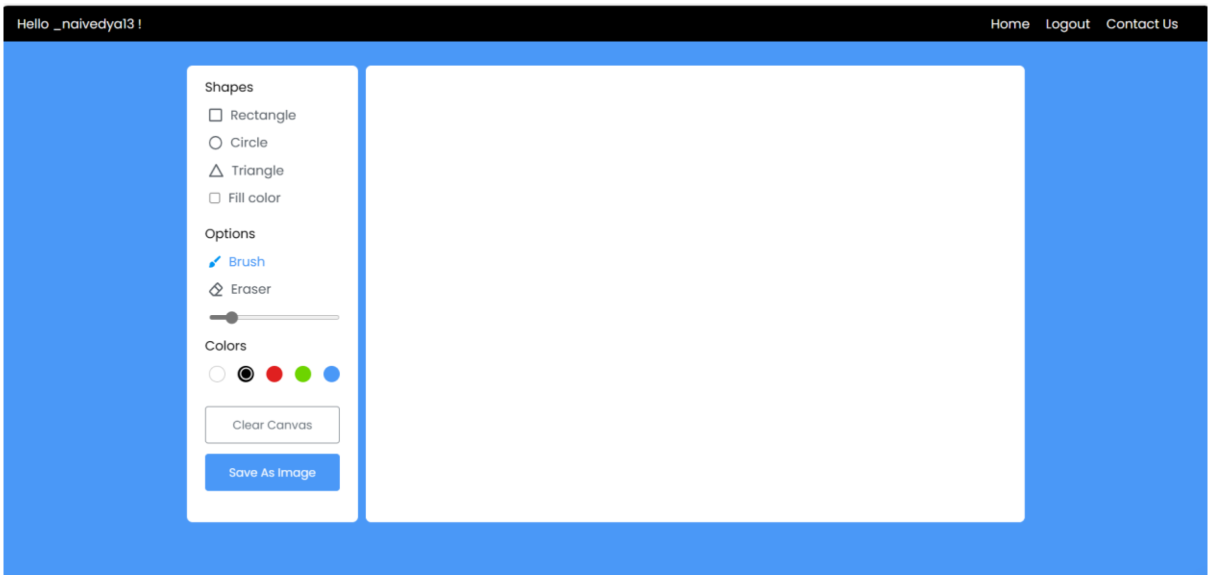
  - CSS Section:
    - Styles the elements of the Tic-Tac-Toe game, such as layout, navigation, game title, game board, game status, and restart button.
  - JavaScript Section:
    - Manages the behavior and interactions of the Tic-Tac-Toe game.
    - Handles cell clicks, cell played, player change, result validation (winning conditions and draw), and game restart.

2. tic.css:
  - Defines styles for the Tic-Tac-Toe game, such as layout, navigation bar, game title, game board, game status, and restart button.

3. tic.js:
  - Implements the logic and functionality of the Tic-Tac-Toe game.
  - Tracks the game state, current player, and game status.
  - Handles cell clicks, cell played, player change, result validation (winning conditions and draw), and game restart.

Overall, these files work together to create a simple Tic-Tac-Toe game where two players can take turns clicking cells on the board until one player wins or the game ends in a draw. The PHP section ensures that only authenticated users can access the game.

Paint:



File Name: paint.php

```
<?php
session_start();

// Check if the user is logged in
if (isset($_SESSION['username'])) {
    $userName = $_SESSION['username'];
} else {
    // Redirect to the login page if not logged in
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Drawing App </title>
    <link rel="stylesheet" href="paint.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="paint.js" defer></script>
  </head>
  <body>
    <nav class="nav">
      <i class="uil uil-bars navOpenBtn"></i>
      <?php
      // Assuming $userName contains the user's name fetched from the
database
      echo 'Hello   ' . $userName . ' !';
      ?>
      <ul class="nav-links">
        <i class="uil uil-times navCloseBtn"></i>
        <li><a href="main.php">Home</a></li>
        <li><a href="index.php">Logout</a></li>
        <li><a href="https://forms.gle/z7wqfihe5uqwEvNZ7">Contact
Us</a></li>
      </ul>
      <i class="uil uil-search search-icon" id="searchIcon"></i>
  </nav>
    <div class="container">
      <section class="tools-board">
        <div class="row">
          <label class="title">Shapes</label>
          <ul class="options">
            <li class="option tool" id="rectangle">
              <img src="icons/rectangle.svg" alt="">
              <span>Rectangle</span>
            </li>
            <li class="option tool" id="circle">
              <img src="icons/circle.svg" alt="">
              <span>Circle</span>
            </li>
            <li class="option tool" id="triangle">
              <img src="icons/triangle.svg" alt="">
              <span>Triangle</span>
            </li>
            <li class="option">
              <input type="checkbox" id="fill-color">
              <label for="fill-color">Fill color</label>
            </li>
          </ul>
        </div>
        <div class="row">
          <label class="title">Options</label>
          <ul class="options">
            <li class="option active tool" id="brush">
              <img src="icons/brush.svg" alt="">
              <span>Brush</span>
            </li>
            <li class="option tool" id="eraser">
              <img src="icons/eraser.svg" alt="">
              <span>Eraser</span>
            </li>
            <li class="option">
              <input type="range" id="size-slider" min="1" max="30" value="5">
            </li>
          </ul>
        </div>
        <div class="row colors">
          <label class="title">Colors</label>
          <ul class="options">
            <li class="option"></li>
            <li class="option selected"></li>
            <li class="option"></li>
            <li class="option"></li>
            <li class="option">
              <input type="color" id="color-picker" value="4A98F7">
            </li>
          </ul>
        </div>
        <div class="row buttons">
          <button class="clear-canvas">Clear Canvas</button>
          <button class="save-img">Save As Image</button>
        </div>
      </section>
      <section class="drawing-board">
        <canvas></canvas>
      </section>
    </div>

  </body>
</html>
```

File Name: paint.css

```
/* Import Google font - Poppins */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&dis
play=swap');
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background: 4A98F7;
}
.nav {
  position: fixed; /* Fix the navigation bar position */
  top: 0; /* Place it at the top of the viewport */
  left: 0;
  right: 0;
  z-index: 9999; /* Set a high z-index value */
  background-color: black;
  color: fff;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 20px;
  backdrop-filter: blur(40px);
}
.nav-links {
  z-index: 1;
  list-style: none;
  padding: 0;
  display: flex;
  justify-content: flex-end; /* Align links to the right */
  flex-grow: 1; /* Allow links to take remaining space */
}
```

```
.nav-links li {
  margin-right: 20px;
}

.nav-links a {
  text-decoration: none;
  color: fff;
}
.container{
  display: flex;
  width: 100%;
  gap: 10px;
  padding: 10px;
  max-width: 1050px;
}
section{
  background: fff;
  border-radius: 7px;
}
.tools-board{
  width: 210px;
  padding: 15px 22px 0;
}
.tools-board .row{
  margin-bottom: 20px;
}
.row .options{
  list-style: none;
  margin: 10px 0 0 5px;
}
.row .options .option{
  display: flex;
  cursor: pointer;
  align-items: center;
  margin-bottom: 10px;
}
.option:is(:hover, .active) img{
  filter: invert(17%) sepia(90%) saturate(3000%) hue-rotate(900deg)
brightness(100%) contrast(100%);
}
.option :where(span, label){
  color: 5A6168;
  cursor: pointer;
  padding-left: 10px;
}
.option:is(:hover, .active) :where(span, label){
  color: 4A98F7;
}
.option fill-color{
  cursor: pointer;
  height: 14px;
  width: 14px;
}
fill-color:checked ~ label{
  color: 4A98F7;
}
.option size-slider{
  width: 100%;
  height: 5px;
  margin-top: 10px;
}
.colors .options{
  display: flex;
  justify-content: space-between;
}
.colors .option{
  height: 20px;
  width: 20px;
  border-radius: 50%;
  margin-top: 3px;
  position: relative;
}
.colors .option:nth-child(1){
  background-color: fff;
  border: 1px solid bfbfbf;
}
.colors .option:nth-child(2){
  background-color: 000;
}
```

```
.colors .option:nth-child(3){
  background-color: E02020;
}
.colors .option:nth-child(4){
  background-color: 6DD400;
}
.colors .option:nth-child(5){
  background-color: 4A98F7;
}
.colors .option.selected::before{
  position: absolute;
  content: "";
  top: 50%;
  left: 50%;
  height: 12px;
  width: 12px;
  background: inherit;
  border-radius: inherit;
  border: 2px solid fff;
  transform: translate(-50%, -50%);
}
.colors .option:first-child.selected::before{
  border-color: ccc;
}
.option color-picker{
  opacity: 0;
  cursor: pointer;
}
.buttons button{
  width: 100%;
  color: fff;
  border: none;
  outline: none;
  padding: 11px 0;
  font-size: 0.9rem;
  margin-bottom: 13px;
  background: none;
  border-radius: 4px;
  cursor: pointer;
}
```

```
.buttons .clear-canvas{
  color: 6C757D;
  border: 1px solid 6C757D;
  transition: all 0.3s ease;
}
.clear-canvas:hover{
  color: fff;
  background: 6C757D;
}
.buttons .save-img{
  background: 4A98F7;
  border: 1px solid 4A98F7;
}
.drawing-board{
  flex: 1;
  overflow: hidden;
}
.drawing-board canvas{
  width: 100%;
  height: 100%;
}
```

File Name: paint.js

```
const canvas = document.querySelector("canvas"),
toolBtns = document.querySelectorAll(".tool"),
fillColor = document.querySelector("fill-color"),
sizeSlider = document.querySelector("size-slider"),
colorBtns = document.querySelectorAll(".colors .option"),
colorPicker = document.querySelector("color-picker"),
clearCanvas = document.querySelector(".clear-canvas"),
saveImg = document.querySelector(".save-img"),
ctx = canvas.getContext("2d");

// global variables with default value
let prevMouseX, prevMouseY, snapshot,
isDrawing = false,
selectedTool = "brush",
brushWidth = 5,
selectedColor = "000";
```

```
const setCanvasBackground = () => {
   // setting whole canvas background to white, so the downloaded img
background will be white
   ctx.fillStyle = "fff";
   ctx.fillRect(0, 0, canvas.width, canvas.height);
   ctx.fillStyle = selectedColor; // setting fillstyle back to the selectedColor, it'll
be the brush color
}

window.addEventListener("load", () => {
   // setting canvas width/height.. offsetwidth/height returns viewable
width/height of an element
   canvas.width = canvas.offsetWidth;
   canvas.height = canvas.offsetHeight;
   setCanvasBackground();
});

const drawRect = (e) => {
   // if fillColor isn't checked draw a rect with border else draw rect with
background
   if(!fillColor.checked) {
      // creating circle according to the mouse pointer
      return ctx.strokeRect(e.offsetX, e.offsetY, prevMouseX - e.offsetX,
prevMouseY - e.offsetY);
   }
   ctx.fillRect(e.offsetX, e.offsetY, prevMouseX - e.offsetX, prevMouseY -
e.offsetY);
}

const drawCircle = (e) => {
   ctx.beginPath(); // creating new path to draw circle
   // getting radius for circle according to the mouse pointer
   let radius = Math.sqrt(Math.pow((prevMouseX - e.offsetX), 2) +
Math.pow((prevMouseY - e.offsetY), 2));
   ctx.arc(prevMouseX, prevMouseY, radius, 0, 2 * Math.PI); // creating circle
according to the mouse pointer
   fillColor.checked ? ctx.fill() : ctx.stroke(); // if fillColor is checked fill circle else
draw border circle
}
```

```
const drawTriangle = (e) => {
   ctx.beginPath(); // creating new path to draw circle
   ctx.moveTo(prevMouseX, prevMouseY); // moving triangle to the mouse
pointer
   ctx.lineTo(e.offsetX, e.offsetY); // creating first line according to the mouse
pointer
   ctx.lineTo(prevMouseX * 2 - e.offsetX, e.offsetY); // creating bottom line of
triangle
   ctx.closePath(); // closing path of a triangle so the third line draw
automatically
   fillColor.checked ? ctx.fill() : ctx.stroke(); // if fillColor is checked fill triangle
else draw border
}

const startDraw = (e) => {
   isDrawing = true;
   prevMouseX = e.offsetX; // passing current mouseX position as prevMouseX
value
   prevMouseY = e.offsetY; // passing current mouseY position as prevMouseY
value
   ctx.beginPath(); // creating new path to draw
   ctx.lineWidth = brushWidth; // passing brushSize as line width
   ctx.strokeStyle = selectedColor; // passing selectedColor as stroke style
   ctx.fillStyle = selectedColor; // passing selectedColor as fill style
   // copying canvas data & passing as snapshot value.. this avoids dragging the
image
   snapshot = ctx.getImageData(0, 0, canvas.width, canvas.height);
}

const drawing = (e) => {
   if(!isDrawing) return; // if isDrawing is false return from here
   ctx.putImageData(snapshot, 0, 0); // adding copied canvas data on to this
canvas

   if(selectedTool === "brush" || selectedTool === "eraser") {
      // if selected tool is eraser then set strokeStyle to white
      // to paint white color on to the existing canvas content else set the stroke
color to selected color
      ctx.strokeStyle = selectedTool === "eraser" ? "fff" : selectedColor;
```

```
        ctx.lineTo(e.offsetX, e.offsetY); // creating line according to the mouse
pointer
        ctx.stroke(); // drawing/filling line with color
    } else if(selectedTool === "rectangle"){
        drawRect(e);
    } else if(selectedTool === "circle"){
        drawCircle(e);
    } else {
        drawTriangle(e);
    }
}

toolBtns.forEach(btn => {
    btn.addEventListener("click", () => { // adding click event to all tool option
        // removing active class from the previous option and adding on current
clicked option
        document.querySelector(".options .active").classList.remove("active");
        btn.classList.add("active");
        selectedTool = btn.id;
    });
});

sizeSlider.addEventListener("change", () => brushWidth = sizeSlider.value); //
passing slider value as brushSize

colorBtns.forEach(btn => {
    btn.addEventListener("click", () => { // adding click event to all color button
        // removing selected class from the previous option and adding on current
clicked option
        document.querySelector(".options
.selected").classList.remove("selected");
        btn.classList.add("selected");
        // passing selected btn background color as selectedColor value
        selectedColor =
window.getComputedStyle(btn).getPropertyValue("background-color");
    });
});

colorPicker.addEventListener("change", () => {
    // passing picked color value from color picker to last color btn background
    colorPicker.parentElement.style.background = colorPicker.value;
    colorPicker.parentElement.click();
});

clearCanvas.addEventListener("click", () => {
    ctx.clearRect(0, 0, canvas.width, canvas.height); // clearing whole canvas
    setCanvasBackground();
});

saveImg.addEventListener("click", () => {
    const link = document.createElement("a"); // creating <a> element
    link.download = `${Date.now()}.jpg`; // passing current date as link download
value
    link.href = canvas.toDataURL(); // passing canvasData as link href value
    link.click(); // clicking link to download image
});

canvas.addEventListener("mousedown", startDraw);
canvas.addEventListener("mousemove", drawing);
canvas.addEventListener("mouseup", () => isDrawing = false);
```

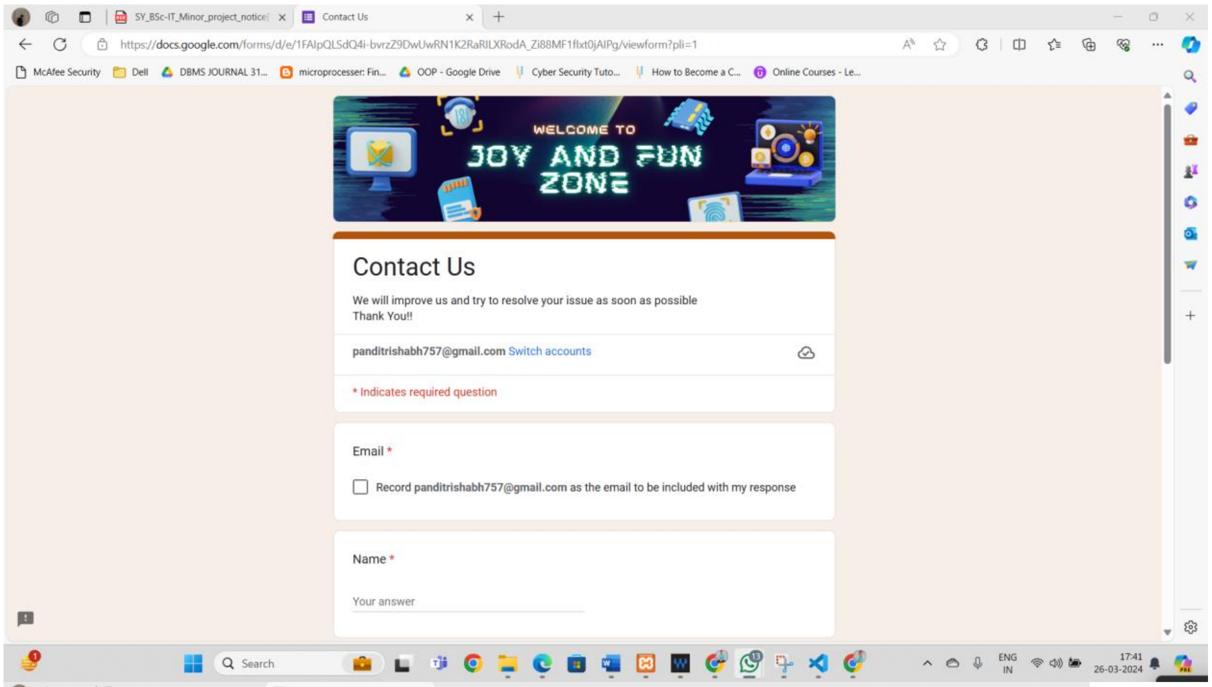
Explanation:

1. paint.php:
   - PHP Section:
     - Starts a session to maintain user authentication.
     - Checks if the user is logged in. If not, redirects to the login page.
   - HTML Section:
     - Contains the structure of the drawing application.
     - Includes navigation, tools, color palette, canvas, and buttons.
   - CSS Section:
     - Defines styles for the layout, navigation bar, tools, colors, buttons, and canvas.
   - JavaScript Section:
     - Handles the functionality of the drawing application, including selecting tools, colors, drawing shapes, clearing the canvas, and saving the drawing.

2. paint.css:
   - Styles the elements of the drawing application, such as layout, navigation, tools, colors, and canvas.

3. paint.js:
  - Manages the behavior and interactions of the drawing application.
  - Selects tools (brush, eraser, shapes), sets brush size, chooses colors, clears the canvas, and saves the drawing.
  - Implements drawing functionalities for freehand drawing, rectangles, circles, and triangles.
  - Listens for events such as mouse clicks, mouse movements, and slider changes to update the canvas accordingly.

Contact Form:



Contact Us

We will improve us and try to resolve your issue as soon as possible
Thank You!!

panditrishabh757@gmail.com Switch accounts

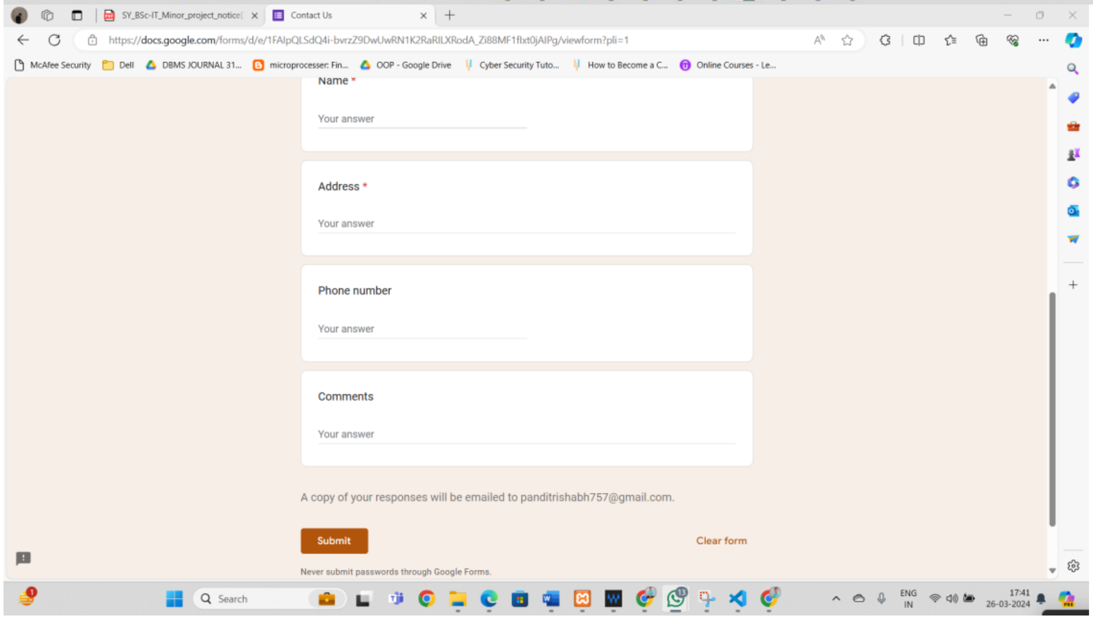
* Indicates required question

Email *

☐ Record panditrishabh757@gmail.com as the email to be included with my response

Name *

Your answer



Name *

Your answer

Address *

Your answer

Phone number

Your answer

Comments

Your answer

A copy of your responses will be emailed to panditrishabh757@gmail.com.

Submit

Clear form

Never submit passwords through Google Forms.

Contact Form

Chapter – 4
Conclusion

Snake Game (snake.php, style.css, index.js):

- Functionality: Implements a classic snake game where the player controls a snake to eat food and grow while avoiding collisions with walls and itself.

- Security: Ensures user authentication by checking if the user is logged in before allowing access to the game.

- Styling: Utilizes CSS for styling elements such as the navigation bar, game board, score display, and buttons, providing a visually appealing interface.

- Interactivity: Incorporates JavaScript to handle game logic, including snake movement, collision detection, score management, and keyboard input for controlling the snake.

- Persistence: Stores the highest score in the browser's local storage to maintain the player's progress across sessions.

- Enhancements: Could be further improved by adding features such as different levels of difficulty, sound effects toggle, and responsive design for mobile devices.

Tic-Tac-Toe Game (tic_tac_toe.php, tic.css, tic.js):

- Gameplay: Offers a two-player Tic-Tac-Toe game where players take turns marking cells to form a winning pattern or result in a draw.

- Authentication: Implements user authentication to restrict access to authenticated users only, ensuring a secure gaming experience.

- Design: Uses CSS for styling elements like the navigation bar, game title, game board, status display, and restart button, presenting a visually appealing layout.

- Functionality: JavaScript handles game logic, including cell clicks, result validation, player turns, and game restarts, providing an interactive gaming experience.

- Scalability: Allows for potential enhancements such as implementing an AI opponent, adding animations for winning cells, and integrating multiplayer functionality.

Paint App (paint.php, paint.css, paint.js):

- Features: Provides a simple drawing app with tools for drawing shapes (rectangle, circle, triangle), freehand drawing (brush), erasing, color selection, size adjustment, and canvas clearing.

- Security: Includes user authentication to ensure that only authenticated users can access the painting application.

- Design: Utilizes CSS for styling elements such as the navigation bar, tool options, color palette, canvas, and buttons, offering an intuitive user interface.

- Functionality: JavaScript manages the drawing functionality, including handling mouse events for drawing shapes and freehand strokes, color selection, size adjustment, canvas clearing, and image saving.

- Customization: Allows users to customize their drawing experience by selecting different drawing tools, colors, and brush sizes.

- Potential Enhancements: Could be enhanced with features like undo/redo functionality, layer management, image importing, and sharing options.

Overall, all three applications demonstrate a combination of functionality, security, design, and interactivity, providing users with engaging experiences tailored to different types of games and creative activities. Each application has room for further improvement and feature expansion based on user feedback and evolving requirements.