GLA UNIVERSITY

PROJECT ON

WEB TECHNOLOGY

Submitted By:

1. Krishna Saraswat
2. Krishna Yadav
3. Lalit Kumar Badhotiya
4. Manushvi Pratap Singh
5. Harsh Sharma

Submitted To:

Mr. Shivank Chauhan Sir
(Subject Faculty)

PROJECT - 1

TO-DO LIST APP

Introduction:-

Welcome to our to-do list app, designed to simplify your daily tasks and boost productivity. With an intuitive interface and seamless functionality, our app allows you to easily create, organize, and manage your tasks in one place. Whether you're a busy professional, a student with deadlines, or simply someone who loves staying organized, our app is your go-to solution. Say goodbye to scattered sticky notes and hello to efficient task management with our to-do list app.

Project Overview:-

A streamlined to-do list program designed for efficient task management, featuring intuitive user interface and customizable prioritization.

Technology Used:-

HTML (HyperText Markup Language):-

HTML provides the structure and content of your web page. In the case of a to-do list app, HTML will define the layout of your app, including the input fields, buttons, and the area where the list of tasks will be displayed.

HTML Code:-

```html
<html>
<head>
    <title>Document</title>
    <script src="https://kit.fontawesome.com/f30fac2c61.js" crossorigin="anonymous"></script>
    <link href="https://shorturl.at/tEJYZ" rel="stylesheet">
    <link rel="stylesheet" href="Todo.css">
</head>
<body>
   <div class="container">
    <h1>ToDo App</h1>
    <div class="inputs">
        <input type="text" placeholder="Enter Your Task" id="inp">
        <button onclick="Add()">Add</button>
    </div>
    <div class="text">
    </div>
   </div>
    <script src="Todo.js"></script>
</body>
</html>
```

CSS (Cascading Style Sheets):-

CSS is used to style the HTML elements, making the app visually appealing and easy to use. You can define styles for various elements such as fonts, colors, sizes, margins, and padding.

CSS Code:-

```
body{
    background-color: rgb(4, 29, 29);
    font-family: 'Poppins', sans-serif;}
 .container{
    width: 360px;
    padding: 33px;
    text-align: center;
    background-color: #fff;
    margin-left: 35%;
    margin-top: 100px;}
 input{
    width: 270px;
    padding: 8px;
    border: 2px solid rgb(6, 145, 145);
    outline: none;
    font-size: 19px;}
 button{
    width: 60px;
    height: 41px;
    background-color: rgb(4, 29, 29);
    color: white;
    border: none;
    cursor: pointer;

    outline: none;
    border-radius: 2px;
    font-size: 19px;}
 .text ul{
    position: relative;
    background-color: rgb(5, 31, 31);
    color: white;
    font-size: 19px;
    padding: 7px;
    text-align: left;}
 .text ul i{
    position: absolute;
    right: 10px;
    cursor: pointer;}
```

JS (JavaScript):-

JavaScript adds interactivity and functionality to the app. It allows users to add tasks, mark tasks as completed, delete tasks, and perform other actions without reloading the page.By combining HTML, CSS, and JavaScript, you can create a functional and visually appealing to-do list app that allows users to manage their tasks effectively.

Java Script Code:-

```
let inputs = document.getElementById("inp");
let text = document.querySelector(".text");
function Add(){
    if(inputs.value == ""){
        alert("Please Enter Task")
    }else{
        let newEle = document.createElement("ul");
        newEle.innerHTML=`${inputs.value} <i class="fa-solid fa-trash"></i>`;
        text.appendChild(newEle);
        inputs.value="";
        newEle.querySelector("i").addEventListener("click" , remove);
        function remove(){
            newEle.remove()
        }
    }
}
```

Features:-

Efficient task categorization and prioritization with reminders and seamless syncing across devices.

Design and Implementation:-

A minimalist design with intuitive task input and quick access features, implemented with cross-platform compatibility for seamless usability.

Challenged Faced:-

Balancing simplicity with functionality and addressing user interface clutter while ensuring comprehensive task management.

Future Enhancement:-

Integration of AI for smart task suggestions and predictive scheduling, along with enhanced collaboration features for team productivity.

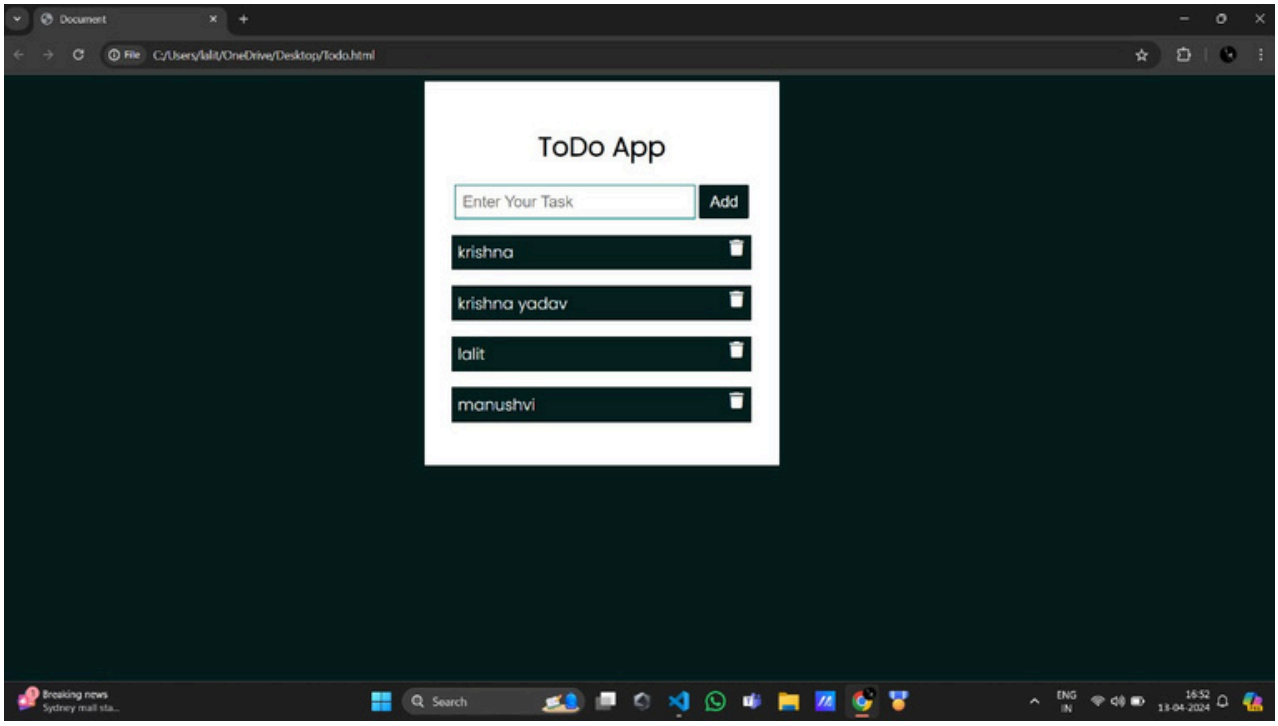
Conclusion:-

To-do lists streamline task management, offering users a structured approach to prioritize and accomplish goals efficiently, ultimately fostering productivity and peace of mind.

Reference:-

Task management and productivity tools like To do list and Microsoft To Do offer effective ways to organize daily tasks and optimize time management.

Output:-



PROJECT - 2

WEATHER APP

Introduction:-

Welcome to our weather app, your personal meteorologist in the palm of your hand. With our app, you can effortlessly stay informed about the current weather conditions and forecasts for any location worldwide. Whether you're planning a weekend getaway, preparing for your daily commute, or just curious about the weather outside, our app provides you with accurate and up-to-date information. Featuring a sleek design and user-friendly interface, accessing weather updates has never been easier. Say goodbye to surprises and hello to weather confidence with our weather app.

Project Overview:-

A weather app providing real-time forecasts and customizable alerts, offering users detailed weather information for optimal planning and preparedness.

Technology Used:-

HTML (HyperText Markup Language):-

Begin by designing the structure of your weather app using HTML. This includes elements for displaying the current weather conditions, forecasts, location search, and any other features you plan to include. You'll likely have sections for the current weather, hourly forecast, daily forecast, and possibly additional information such as radar maps or weather alerts.

HTML Code:-

```
<html>
<head>
<linkrel="stylesheet"href="https://fonts.googleapis.com/icon?family=Material+Icons"/>
<title>Weather App</title>
<link rel="stylesheet" href="weather.css" />></head>
<body>
<div class="weather-app">
<form class="search-form" action="">
<inputclass="city-input"
type="text"
placeholder="Enter City Name"/>
<button class="search-btn" type="submit">
<i class="material-icons">search</i>
</button></form>
<div class="city-date-section">
<h2 class="city">Delhi</h2>
<p class="date">7 Jan 2024</p></div>
<div class="temperature-info">
<div class="description">
<i class="material-icons">sunny</i>
<span class="description-text">Sunny</span></div>
<div class="temp">20°</div></div>
<div class="additional-info">
<div class="wind-info">
<i class="material-icons">air</i>rt<div>
<h3 class="wind-speed">4 KM/H</h3>
<p class="wind-label">Wind</p></div></div>
<div class="humidity-info">
<i class="material-icons">water_drop</i><div>
<h3 class="humidity">45%</h3>
<p class="humidity-label">humidity</p>
</div></div>
<div class="visibility-info">
<i class="material-icons">visibility</i><div>
<h3 class="visibility-distance">4 KM/H</h3>
<p class="visibility">Visibility</p>
</div></div></div></div>
<script src="weather.js"></script></body></html>
```

CSS (Cascading Style Sheets):-

Use CSS to style your HTML elements and make your weather app visually appealing and user-friendly. Consider using a responsive design approach to ensure your app looks good on various screen sizes, from mobile devices to desktop computers. Use appropriate fonts, colors, and layout techniques to create a clean and intuitive interface.

CSS Code:-

```
* {margin: 0;
    padding: 0;
    box-sizing: border-box;}
body {
    background-color: hsl(204, 86%, 23%);
    font-family: Arial, Helvetica, sans-serif;
    height: 100vh;}
.weather-app {
    background-color: white;
    max-width: 600px;
    margin: 50px auto;
    padding: 30px;
    border-radius: 15px;}
.search-form {
    display: flex;
    align-items: center;}
input[type="text"] {
    flex: 1;
    padding: 20px;
    border: 2px solid #3498db;
    border-radius: 15px;
    font-size: 1rem;
    height: 3.5rem;}
.search-btn {
    background-color: #3498db;
    border: none;
    padding: 15px;
    border-radius: 15px;
    cursor: pointer;
    color: white;
    margin-left: 10px;
    transition: backgroung-color 0.3s ease;}
.search-btn:hover {
    background-color: #2980b9;}
```

```
.city-date-section {
    margin-top: 30px;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    gap: 10px;}
.city-date-section .city {
    font-size: 2rem;
    opacity: 0.9;}
.city-date-section .date {
    background-color: #c3eaf4;
    padding: 5px 10px;
    border-radius: 15px;}
.temperature-info {
    margin-top: 20px;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;}
.temperature-info .temp {
    font-size: 10rem;}
.temperature-info .description {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px;}
.temperature-info .description i {
    font-size: 1.5rem;
    opacity: 0.8;}
.temperature-info .description span {
    font-size: 1.5rem;}
.additional-info {
    margin-top: 20px;
    display: flex;
    justify-content: space-around;}
.wind-info,
.humidity-info,
```

```
.visibility-info {
    background-color: #ecf0f1;
    width: 150px;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    gap: 15px;
    border-radius: 15px;
    padding: 30px 20px;
    box-shadow: rgba(0, 0, 0, 0.24) 0px 3px 8px;}
.additional-info .material-icons {
    font-size: 3rem;
    opacity: 0.8;}
.wind-info div,
.humidity-info div,
.visibility-info div {
    text-align: center;}
```

JS (JavaScript):-

In a weather app built with JavaScript, the code interacts with APIs to retrieve weather data and then dynamically updates the user interface to display this information. Through JavaScript, the app fetches data such as current conditions, hourly forecasts, and daily forecasts from the weather API.

Java Script Code:-

```
const apiKey = '8d6454a89dff871786a0307b0dbebbee'
async function fetchWeatherData(city) {
try {const response = await fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}`);
if (!response.ok) {
throw new Error("Unable to fetch weather data");}
const data = await response.json();
console.log(data);
updateWeatherUI(data);} catch (error) {console.error(error);}}
const cityElement = document.querySelector(".city");
const temperature = document.querySelector(".temp");
const windSpeed = document.querySelector(".wind-speed");
const humidity = document.querySelector(".humidity");
const visibility = document.querySelector(".visibility-distance");
const descriptionText = document.querySelector(".description-text");
const date = document.querySelector(".date");
const descriptionIcon = document.querySelector(".description i")
function updateWeatherUI(data) {cityElement.textContent = data.name;
    temperature.textContent = `${Math.round(data.main.temp)}`;
    windSpeed.textContent = `${data.wind.speed} km/h`;
    humidity.textContent = `${data.main.humidity}%`;
    visibility.textContent = `${data.visibility / 1000} km`;
    descriptionText.textContent = data.weather[0].description;
    const currentDate = new Date();
    date.textContent = currentDate.toDateString();
    const weatherIconName = getWeatherIconName(data.weather[0].main);
    descriptionIcon.innerHTML = `<i class="material-icons">${weatherIconName}</i>`;}
const formElement = document.querySelector(".search-form");
const inputElement = document.querySelector(".city-input");
formElement.addEventListener("submit", function (e) {
e.preventDefault();
const city = inputElement.value;
if (city !== "") {fetchWeatherData(city);
inputElement.value = "";}});
function getWeatherIconName(weatherCondition) {
const iconMap = { Clear: "wb_sunny", Clouds: "wb_cloudy",
Rain: "umbrella",Thunderstorm: "flash_on",
Drizzle: "grain",Snow: "ac_unit",
Mist: "cloud",Smoke: "cloud",
Haze: "cloud",Fog: "cloud",};
return iconMap[weatherCondition] || "help";}
```

Features:-

A user-friendly interface displaying current conditions, hourly forecasts, and extended outlooks, with geolocation for accurate local weather updates.

Design and Implementation:-

A minimalist design showcasing weather data fetched from APIs, implemented with responsive UI elements for intuitive navigation across platforms.

Challenged Faced:-

Integrating multiple APIs for accurate and reliable weather data while optimizing performance and minimizing data usage for mobile users.

Future Enhancements:-

Implementing machine learning algorithms for more accurate weather predictions and integrating augmented reality features for immersive weather experiences.

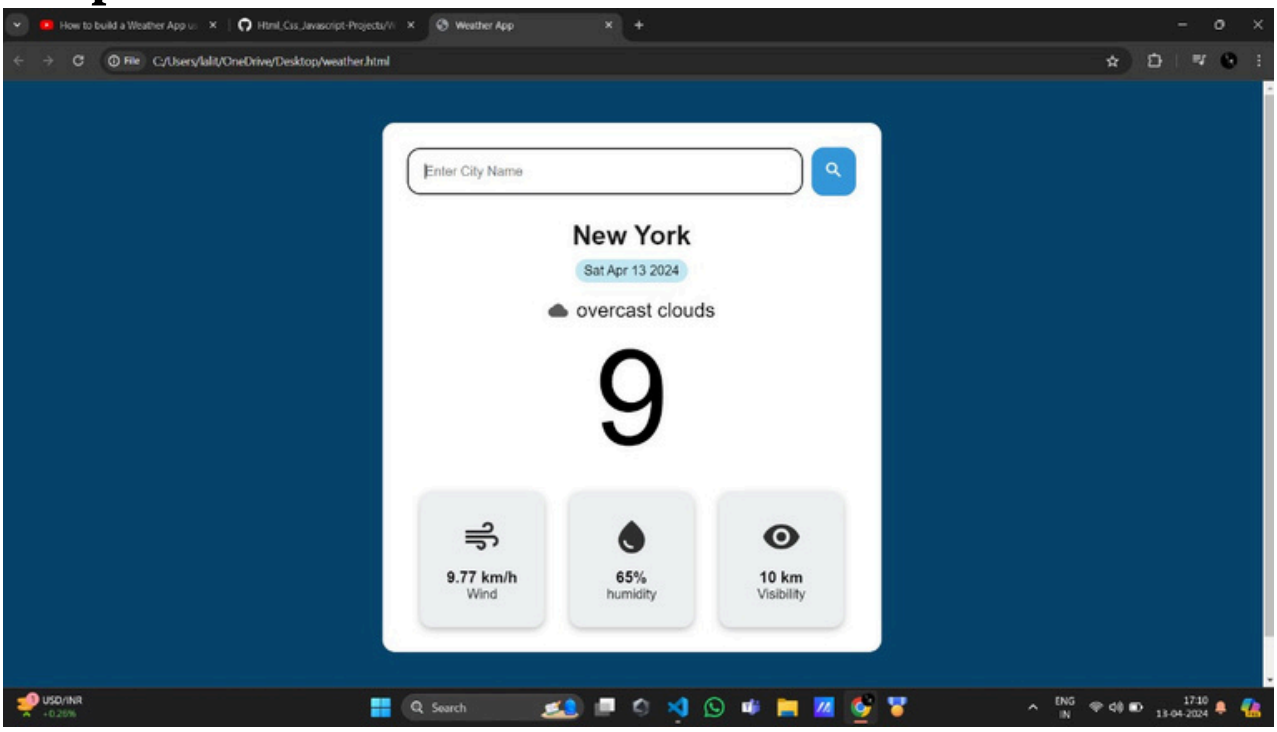
Conclusion:-

Weather apps offer indispensable tools for users to plan activities and stay informed about current and future weather conditions, enhancing daily decision-making and safety.

Reference:-

Popular weather apps such as AccuWeather and The Weather Channel provide comprehensive forecasts and real-time updates, trusted by millions worldwide for accurate weather information.

Output:-



PROJECT - 3

IMAGE SLIDER

Introduction:-

Welcome to our image slider app, where visual storytelling meets seamless functionality. Whether you're showcasing products, sharing memories, or adding visual flair to your website, our app is the perfect tool for creating captivating slideshows. With easy-to-use controls and customizable features, you can effortlessly navigate through images, add animations, and create dynamic presentations that engage your audience. Say goodbye to static content and hello to dynamic storytelling with our image slider app. Elevate your website or project with stunning visuals that leave a lasting impression.

Project Overview:-

An interactive image slider showcasing smooth transitions between pictures, designed with user-friendly controls and adaptable for diverse screen sizes.

Technology Used:-

HTML (HyperText Markup Language):-

HTML provides the structure and content of web pages. In the context of an image slider, HTML defines the elements that compose the slider, such as the container for the slides, individual slide elements (usually <img> tags for images), and navigation controls. Each <img> tag represents an image in the slider, with its src attribute specifying the image file and alt attribute providing alternative text for accessibility.

Code:-

```html
<!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>Image Slider JavaScript</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.1/css/all.min.css" />
    <script src="script.js" defer></script>
  </head>
  <body>
    <section class="wrapper">
      <i class="fa-solid fa-arrow-left button" id="prev"></i>
      <div class="image-container">
        <div class="carousel">
          <img src="https://cdn.pixabay.com/photo/2016/10/20/18/35/earth-1756274_1280.jpg" alt="" />
          <img src="https://img.freepik.com/free-photo/ultra-detailed-nebula-abstract-wallpaper-4_1562-749.jpg" alt="" />
          <img src="https://img.freepik.com/free-photo/night-sky-glows-with-iridescent-deep-space-generative-ai_188544-11285.jpg" alt="" />
          <img src="https://assets-global.website-files.com/6494abaf6ca6459fd9d4a0cc/64c187d695b640780c3b9cec_64b6ee06f7cbaaafe5c0f807_spacewalk.jpeg" alt="" />
        </div>
        <i class="fa-solid fa-arrow-right button" id="next"></i>
      </div>
    </section>
  </body>
</html>
```

CSS (Cascading Style Sheets):-

Creating an image slider using CSS focuses on styling and layout, as CSS alone doesn't handle dynamic functionality. The HTML structure defines the slider container and images. CSS styles the container, images, and navigation controls, utilizing transitions for smooth image transitions.

Code:-

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #343f4f;
}
.wrapper {
  display: flex;
  max-width: 650px;
  width: 100%;
  height: 400px;
  background: #fff;
  align-items: center;
  justify-content: center;
  position: relative;
  border-radius: 12px;
  box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1)
}
```

```css
.wrapper i.button {
  position: absolute;
  top: 50%;
  transform: translateY(-50%);
  height: 36px;
  width: 36px;
  background-color: #343f4f;
  border-radius: 50%;
  text-align: center;
  line-height: 36px;
  color: #fff;
  font-size: 15px;
  transition: all 0.3s linear;
  z-index: 100;
  cursor: pointer;
}
i.button:active {
  transform: scale(0.94) translateY(-50%);
}
i#prev {
  left: 25px;
}
i#next {
  right: 25px;
}
.image-container {
  height: 320px;
  max-width: 500px;
  width: 100%;
  overflow: hidden;
}
```

```css
.image-container .carousel {
  display: flex;
  height: 100%;
  width: 100%;
  transition: all 0.4s ease;
}
.carousel img {
  height: 100%;
  width: 100%;
  border-radius: 18px;
  border: 10px solid #fff;
  object-fit: cover;
}
```

JS (JavaScript):-

Constructing an image slider with JavaScript involves managing dynamic actions like image transitions, user navigation, and automated slideshows. JavaScript code orchestrates the switching of images, often triggered by user interactions or a timer. It also governs functionalities like next/previous buttons or pagination dots, ensuring seamless user control. Additionally, JavaScript facilitates responsive behavior and accessibility features, making the slider interactive and user-friendly across various devices and contexts.

Code:-

```javascript
const wrapper = document.querySelector(".wrapper"),
  carousel = document.querySelector(".carousel"),
  images = document.querySelectorAll("img"),
  buttons = document.querySelectorAll(".button");

let imageIndex = 1,
  intervalId;


const autoSlide = () => {

  intervalId = setInterval(() => slideImage(++imageIndex), 2000);
};

autoSlide();


const slideImage = () => {

  imageIndex = imageIndex === images.length ? 0 : imageIndex < 0 ? images.length - 1 : imageIndex;

  carousel.style.transform = `translate(-${imageIndex * 100}%)`;
};


const updateClick = (e) => {

  clearInterval(intervalId);

  imageIndex += e.target.id === "next" ? 1 : -1;
  slideImage(imageIndex);

  autoSlide();
};


buttons.forEach((button) => button.addEventListener("click", updateClick));


wrapper.addEventListener("mouseover", () => clearInterval(intervalId));
wrapper.addEventListener("mouseleave", autoSlide);
```

Features:-

A versatile image slider with customizable transition effects and navigation options, supporting both automatic and manual slide advancement.

Design and Implementation:-

An image slider with CSS for styling and JavaScript for interactivity, employing event listeners for seamless transition handling.

Challenged Faced:-

Addressing cross-browser compatibility issues and optimizing performance for smooth transitions, especially with large image files.

Future Enhancements:-

Incorporating touch gesture support for mobile devices and implementing lazy loading techniques to enhance performance and user experience.

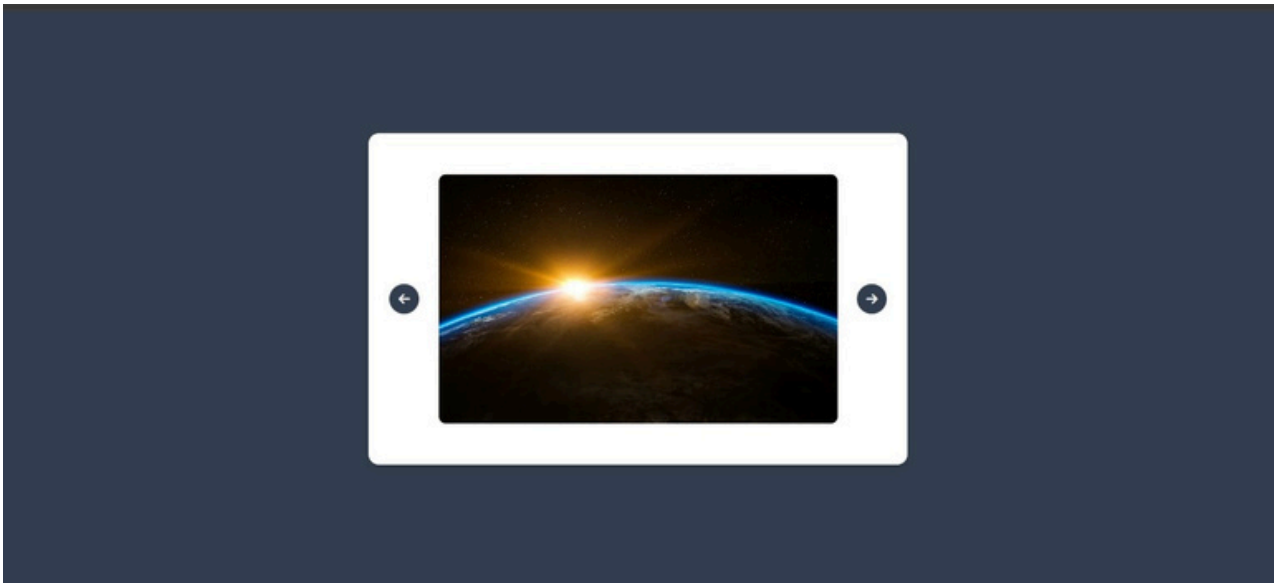
Conclusion:-

Image sliders provide an engaging way to showcase visual content, enhancing website aesthetics and user interaction while offering flexibility for content display.

Reference:-

Slick and Swiper are popular JavaScript libraries commonly used for creating responsive and feature-rich image sliders in web development projects.

Output:-



PROJECT - 4

QUIZ GAME

Introduction:-

Introducing "Quiz Quest: The Ultimate Trivia Adventure"! Step into a world where curiosity reigns supreme and knowledge is your greatest asset. Whether you're a trivia novice or a seasoned master, Quiz Quest offers an immersive and thrilling experience for all. Embark on a journey through a diverse array of categories, from history and science to entertainment and sports. Challenge your friends, family, or even strangers from around the globe as you compete to become the ultimate quiz champion. With engaging gameplay, captivating visuals, and a vast database of questions, Quiz Quest promises endless hours of entertainment and enlightenment. Are you ready to test your wits and conquer the quest for knowledge? Join us on Quiz Quest and let the adventure begin!

Project Overview:-

A dynamic quiz game project featuring multiple-choice questions, score tracking, and interactive feedback for an engaging user experience.

Technology Used:-

HTML(HyperText Markup Language):-

To develop an engaging and effective quiz game using HTML, one must grasp the foundational principles of HTML theory to structure the content on the web page. Additionally, a solid understanding of user interface (UI) theory is crucial for designing a visually appealing and intuitive interface, incorporating concepts such as layout, typography, and color theory.

Code:-

```html
<html>
<head>
<title>Document</title>
<link rel="stylesheet" href="quizgame.css" />
</head>
<body>
<section class="main">
<div class="container">
<div class="col">
<h3 id="questionBox">
1) Lorem ipsum dolor sit amet, consectetur adipisicing elit Debitis?
</h3>
</div>
<div class="col box">
<input name="option" type="radio" id="first" value="a" required>
<label for="first">Testing 1</label>
</div>
<div class="col box">
<input name="option" type="radio" id="second" value="b" required>
<label for="second">Testing 2</label>
</div>
<div class="col box">
<input name="option" type="radio" id="third" value="c" required>
<label for="third">Testing 3</label>
</div>
<div class="col box">
<input name="option" type="radio" id="fourth" value="d" required>
<label for="fourth">Testing 4</label></div>
<button id="submit">Submit</button></div>
</section>
<script src="quizgame.js"></script>
</body>
</html>
```

CSS (Cascading Style Sheets):-

In crafting a quiz game with CSS, mastering CSS theory is pivotal for shaping a captivating and user-friendly interface. CSS layout methodologies like flexbox and grid empower developers to arrange elements with precision, ensuring clarity in presenting questions and choices.

Code:-

```
* {padding: 0;
    margin: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;}
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@100&display=swap');
.main {
    width: 100%;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: rgb(24, 128, 207);}
.container {
    width: 35rem;
    box-shadow: 0px 0px 5px grey;
    display: flex;
    background-color: white;
    border-radius: 10px;
    overflow: hidden;
    flex-direction: column;}
.col {text-align: justify;
    padding: 15px;
    width: 95%;}
#submit {
    width: 100%;
    background-color: rgb(47, 8, 73);
    transition: 0.5s;
    color: white;
    outline: none;
    border: none;
    font-size: 25px;
    display: block;
    padding: 10px;
    cursor: pointer;}
#submit:hover {
    background-color: rgb(34, 6, 53);}
.box {box-shadow: 0px -1px 1px grey;
    width: 100%;}
```

JS (Java Script):-

In the context of a quiz game, JavaScript theory is indispensable for creating dynamic interactions, managing game logic, and ensuring a seamless user experience. Leveraging JavaScript's event-driven paradigm, developers can handle user inputs, such as selecting answers and progressing through questions, with ease.

Code:-

```
const quizData = [{question: "Which of the following is a client site language?",
a: "Java",b: "C",c: "Python",d: "JavaScript",
correct: "d",},
{ question: "What does HTML stand for?",
a: "Hypertext Markup Language",b: "Cascading Style Sheet",
c: "Jason Object Notation",d: "Helicopters Terminals Motorboats Lamborginis",
correct: "a",},
{question: "What year was JavaScript launched?",
a: "1996",b: "1995",c: "1994",d: "none of the above",correct: "b",},
{ question: "What does CSS stands for?",a: "Hypertext Markup Language",
b: "Cascading Style Sheet",
c: "Jason Object Notation",
d: "Helicopters Terminals Motorboats Lamborginis",
correct: "b",}];let index = 0;let correct = 0,incorrect = 0,
total = quizData.length;
let questionBox = document.getElementById("questionBox");
let allInputs = document.querySelectorAll("input[type='radio']")
const loadQuestion = () => {
if (total === index) {return quizEnd()}reset()
const data = quizData[index]
questionBox.innerHTML = `${index + 1}) ${data.question}`
allInputs[0].nextElementSibling.innerText = data.a
allInputs[1].nextElementSibling.innerText = data.b
allInputs[2].nextElementSibling.innerText = data.c
allInputs[3].nextElementSibling.innerText = data.d}
document.querySelector("#submit").addEventListener(
"click",function() {const data = quizData[index]
const ans = getAnswer()
if (ans === data.correct) {correct++;} else {incorrect++;}index++;loadQuestion()})
const getAnswer = () => {let ans;allInputs.forEach(
(inputEl) => {if (inputEl.checked)
{ans = inputEl.value; }})
return ans;}
const reset = () => {allInputs.forEach(
(inputEl) => {inputEl.checked = false;})}
const quizEnd = () => {
document.getElementsByClassName("container")[0].innerHTML = `<div class="col">
<h3 class="w-100"> Hii, you've scored ${correct} / ${total} </h3></div>`}
loadQuestion(index);
```

Features:-

Interactive quiz gameplay with timed questions and randomized options, accompanied by a scoring system and immediate feedback for user engagement.

Design and Implementation:-

Designed with HTML, CSS, and JavaScript for a responsive user interface and interactive functionalities, leveraging DOM manipulation for dynamic question rendering and result tracking.

Challenges Faced:-

Managing complex logic for question generation and user responses while ensuring smooth user experience across various devices and screen sizes.

Future Enhancements:-

Implementing multiplayer functionality for competitive gaming experiences and integrating machine learning algorithms for adaptive difficulty levels.

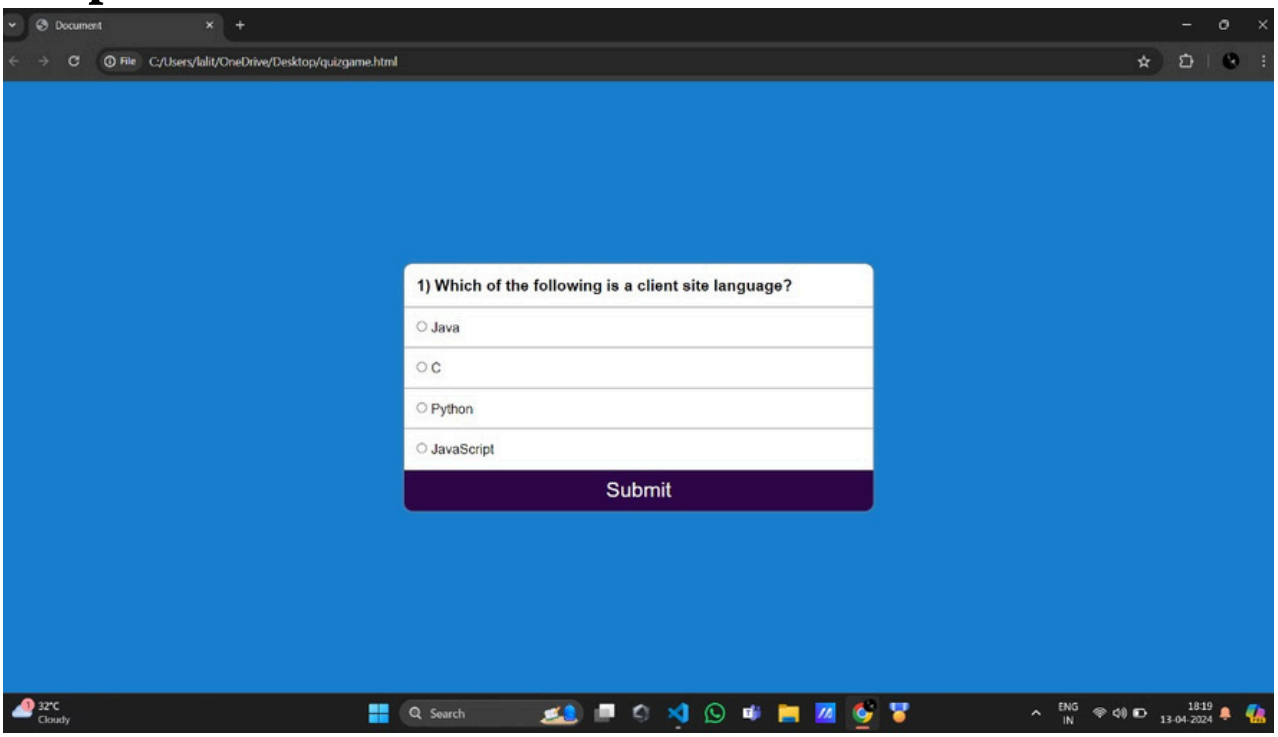
Conclusion:-

Quiz games provide an entertaining and educational platform for users to test their knowledge while fostering engagement and learning in an interactive setting.

References:-

Popular JavaScript libraries like React-Quiz-Component and Quizizz API offer comprehensive solutions for building dynamic and interactive quiz games in web development projects.

Output:-



PROJECT - 5

PORTFOLIO WEBSITE

Introduction:-

A portfolio webpage serves as a digital showcase of an individual's or company's work, skills, and achievements. It typically includes sections such as an about me/about us, projects, skills, resume, and contact information. These webpages are designed to impress potential clients, employers, or collaborators by presenting a curated selection of the creator's best work, demonstrating their expertise and capabilities in a visually appealing and organized manner. Portfolio websites are essential tools for professionals in various fields, including design, development, writing, photography, and more, as they provide a platform to showcase their talents and attract opportunities for collaboration and employment.

Project Overview:-

A portfolio webpage project presents an individual's or company's work and skills in a visually appealing format, aiming to impress potential clients or employers with a curated showcase of achievements and capabilities.

Technology Used:-

HTML (HyperText Markup Language):-

HTML is utilized to structure the content of a portfolio webpage, defining elements like headings, paragraphs, images, and links for presenting information about the creator's work, skills, and achievements.

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=divice-width, initial-scale=1.0">
    <title>Personal Portfolio</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div id="header">

    <div class="container">
        <nav>
            <img src="C:\Users\yadhu\Downloads\portfolio-high-resolution-logo-removebg-preview.png" class="logo">
            <ul>
                <li><a href="#header">Home</a></li>
                <li><a href="#about">About</a></li>
                <li><a href="#services">Services</a></li>
                <li><a href="#about">Portfolio</a></li>
                <li><a href="https://wa.me/qr/36LB7QOQUJFZI1">Contact</a></li>
            </ul>
        </nav>
        <div class="header-text">
            <p><u>UI/UX Designer </u> </p>
            <h1>Hi, I'm <span>Krishna <br> Yadav</span> From <span>भारत</span></h1>

        </div>
    </div>
</div>
```

```html
    <div id="services">
        <div class="container">
            <h1 class="sub-title">My Services</h1>
            <div class="services-list">
                <div>
                    <h2>Web Design</h2>
                    <p>Web design involves creating attractive, user-friendly web pages and web-based apps. User experience design, user interface design, and visual design all play key
roles in the web design process. Web design teams take both form and functionality into account, balancing strong visuals with sound usability
                    </p>
                    <a href="index.html">learn more</a>
                </div>
                <div>
                    <h2>UX/UI Design</h2>
                    <p>Content in UX/UI design refers to the language that appears in a user interface1. Content design, also known as UX writing, is essentially designing with words
                    </p>
                    <a href="https://www.coursera.org/">learn more</a>
                </div>
                <div>
                    <h2>App Design</h2>
                    <p>Defining the layout of the app
                        Designing UI elements such as buttons, icons, text fields, and checkboxes (to name a few)
                        Designing the app's navigation system, including menus and navigation bars
                        Making choices about the app's typography, colour scheme, and imagery
                        Designing the app's interactive properties—what actions can the user take with the app? ...
                    </p>
                    <a href="https://clearbridgemobile.com">learn more</a>
            </div>
        </div>
    </div>
</div>


</div>
    <div id="about">
        <div class="container">
            <div class="row">
                <div class="about-col-1">
                    <img src="">
                </div>
                <div class="about-col-2">
                    <h1 class="sub-title">About Me</h1>

                    <div class="tab-titles">

                        <p class="tab-links active-link" onclick="opentab('skills')">Skills</p>
                        <p class="tab-links active-link" onclick="opentab('experience')">Experience</p>
                        <p class="tab-links active-link" onclick="opentab('education')">Education</p>
                    </div>
                    <div class="tab-contents active-tab" id="skills">
                        <ul>
                            <li><span>UI/UX</span><br>Designing Web/App interfaces</li>
                            <li><span>Web Development</span><br> Web App Development</li>
                            <li><span>App Development</span><br>Building Android/IOS Apps </li>
                        </ul>
                    </div>
                    <div class="tab-contents" id="experience">
                        <ul>
                            <li><span>2021 - Current</span><br>UX/UI Design Training at GLA UNIVERSITY</li>
                            <li><span>2019 - 2021</span><br>Team lead at StarApp LLC.</li>
                            <li><span>2017 - 2019</span><br>UX/UI Design Executive at Coin Digital Ltd.</li>
                            <li><span>2016 - 2017</span><br>Internship at Google Cloud.</li>
                        </ul>
                    </div>
                    <div class="tab-contents" id="education">
                        <ul>
                            <li><span>2016</span><br>UX/UI Design Training at GLA UNIVERSITY</li>
                            <li><span>Web Development</span><br>MCA from MIT Bangalore.</li>
                            <li><span>App Development</span><br>BCA from GLA UNIVERSITY Mathura.</li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

CSS (Cascading Style Sheet):-

CSS styles the HTML elements of a portfolio webpage, defining properties such as layout, typography, colors, and spacing to create a visually appealing and cohesive design that enhances the presentation of the creator's work and skills.

Code:-

```css
*{
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    box-sizing: border-box;
}
body{
    background: #030303;
    color: #fff;
}
#header{
    width: 100%;
    height: 100vh;
    background-image:url();
    background-size:cover;
    background-position:center;
    ;
}
.container{
    padding: 10px 10%;
}
nav{
    display: flex;
    align-items: center;
    justify-content: space-between;
    flex-wrap: wrap;
}
.logo{
    width: 300px;
    height: px;
}
nav ul li{
    display: inline-block;
    list-style: none;
    margin: 10px 20px;
```

```css
nav ul li a{
    color: #fff;
    text-decoration: none;
    font-size:18px;
    position: relative;
}
nav ul li a::after{
    content: '';
    width: 0;
    height: 3px;
    background: #ff004f;
    position: absolute;
    left: 0;
    bottom: -6px;
    transition: 0.5s;
}
nav ul li a:hover::after{
    width: 100%;
}
.header-text{
    margin-top: 20%;
    font-size: 30px;
}
.header-text h1{
    font-size: 60px;
    margin-top: 20px;
}
.header-text h1 span{
    color: #ff004f;
}
/*--------about------*/
#about{
    padding: 80px 0;
    color: #ababab;
}
.row{
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
}
```

```css
.about-col-1{
    flex-basis: 35%;
}
.about-col-1 img{
    width: 100%;
    border-radius: 15px;
}
.about-col-2{
    flex-basis: 60%;
}
.sub-title{
    font-size: 60px;
    font-weight: 600;
    color: #fff;
}
.tab-titles{
    display: flex;
    margin: 20px 0 40px;
}
.tab-links{
    margin-right: 50px;
    font-size: 18px;
    font-weight: 500;
    cursor: pointer;
    position: relative;
}
.tab-links::after{
    content:'';
    width: 0;
    height: 3px;
    background: #ff004f;
    position:absolute;
    left: 0;
    bottom: -8px;
    transition: 0.5s;
}
.tab-links.active-link::after{
    width: 50%;
}
```

```
}
.tab-contents ul li{
    list-style: none;
    margin: 10px 0;
}
.tab-content ul li span{
    color: #b64a6c;
    font-size: 14px;
}
.tab-contents{
    display: none;
}
.tab-contents.active-tab{
    display: block;
}
/*---------------services------------*/
#services{
    padding: 30px 0;
}
.services-list{
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px,1fr));
    grid-gap: 40px;
    margin-top: 50px;
}
```

JS (Java Script):-

JavaScript enhances interactivity on a portfolio webpage, enabling features like dynamic content loading, form validation, and animations to create a more engaging and user-friendly experience for visitors interacting with the site.

Code:-

```
var tablinks = document.getElementsByClassName("tab-links");
var tabcontents = document.getElementsByClassName("tab-contents");

function opentab(tabname){
    for(tablink of tablinks){
        tablink.classList.remove("active-link");
    }
    for(tabcontent of tabcontants){
        tabcontant.classList.remove("active-tab");
    }
    event.currentTarget.classList.add("active-link");
    document.getElementById(tabname).classList.add("active-tab");
}
```

Features:-

A portfolio webpage typically includes sections such as an about me/about us, projects showcase, skills presentation, and contact information, providing visitors with a comprehensive overview of the creator's abilities and achievements. Additionally, it may incorporate responsive design elements, visual enhancements, and interactive features to captivate and engage users effectively.

Design and Implementation:-

The portfolio webpage design emphasizes user-friendly navigation and visually appealing layouts, implemented with HTML, CSS, and JavaScript for content presentation and interactive features, ensuring an engaging and professional user experience.

Challenged Faced:-

Balancing creativity with professionalism in design choices and ensuring cross-browser compatibility and responsiveness across various devices pose common challenges in portfolio webpage development.

Future Enhancements :-

Future enhancements may include integrating multimedia elements like videos and interactive prototypes, as well as incorporating dynamic content updates and integrating with APIs for real-time data display, enhancing the user experience and functionality of the portfolio webpage.

Conclusion:-

A portfolio webpage serves as a powerful tool for showcasing skills and accomplishments, leaving a lasting impression on visitors and opening doors to new opportunities in the professional world.

References:-

Professional portfolio platforms like Behance and Dribbble offer inspiration and templates for designing engaging portfolio webpages, while web development resources like W3Schools and MDN provide tutorials and documentation for implementing HTML, CSS, and JavaScript techniques.

Output:-

PROJECT - 6

EXPENSE TRACKER

Introduction:-

Introducing an expense tracker involves highlighting its significance in managing finances efficiently. An expense tracker serves as a digital companion, empowering users to meticulously monitor their spending habits, income streams, and overall financial health. By systematically recording every transaction, categorizing expenditures, and analyzing patterns, individuals and businesses gain invaluable insights into their financial behaviors. This visibility enables informed decision-making, facilitates budgeting endeavors, and fosters accountability towards financial goals. With features ranging from transaction recording and categorization to budgeting tools and insightful analytics, expense trackers streamline the financial management process and promote fiscal responsibility. Whether utilized for personal finance management or organizational accounting, an expense tracker emerges as an indispensable asset in navigating the complex landscape of financial stewardship, ultimately paving the way for greater financial stability and prosperity.

Project Overview:-

An expense tracker project facilitating easy recording and categorization of expenses, with features for budget management and insightful expenditure analysis.

Technology Used:-

HTML (HyperText Markup Language):-

In developing an expense tracker, HTML theory plays a crucial role in shaping the user interface. Semantic markup, like <header>, <nav>, <main>, and <footer>, structures the layout logically, while forms and inputs such as <input>, <select>, and <button> capture user input for recording transactions and managing financial data. Tables organize tabular data like transaction histories, and hyperlinks facilitate seamless navigation between sections.

Code:-

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>Expense Tracker</title>
    <link rel="stylesheet" href="expense.css" />
  </head>
  <body>
    <h2>Expense Tracker</h2>
    <div class="container">
      <h4>Your Balance</h4>
      <h1 id="balance">$0.00</h1>
      <div class="inc-exp-container">
        <div>
```

```html
        <h4>Income</h4>
        <p id="money-plus" class="money-plus">
          +$0.00
        </p>
      </div>
      <div>
        <h4>Expense</h4>
        <p id="money-minus" class="money-minus">
          -$0.00
        </p>
      </div>
    </div>

    <h3>History</h3>
    <ul id="list" class="list">
    </ul>
    <h3>Add New Transition</h3>
    <form id="form">
```

```html
            <div class="form-control">
                <label for="text">Text</label>
                <input type="text" id="text" placeholder="Enter Text...."/>
            </div>
            <div class="form-control">
                <label for="amount">Amount <br> (negative - expense ,positive - income )</label>
                <input type="number" id="amount" placeholder="Enter amount...">
            </div>
            <button class="btn">All transaction</button>
        </form>
    </div>
    <script src="expense.js"></script>
  </body>
</html>
```

CSS (Cascading Style Sheet):-

In an expense tracker application, CSS serves as the stylistic backbone, enhancing the visual appeal and user experience. Through CSS, developers can customize elements like colors, fonts, spacing, and layout, creating a cohesive and engaging interface. Additionally, CSS enables the creation of responsive designs, ensuring usability across different devices.

Code:-

```css
:root {
  --box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12),
    0 1px 2px rgba(0, 0, 0, 0.24);
}

* {
  box-sizing: border-box;
}

body {
  background-color: #f7f7f7;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont,
    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell,
    "Open Sans", "Helvetica Neue", sans-serif;
}

.container {
  margin: 30px auto;
  width: 350px;
}

h1 {
  letter-spacing: 1px;
  margin: 0;
}

h3 {
  border-bottom: 1px solid #bbb;
  padding-bottom: 10px;
  margin: 40px 0 10px;
}

h4 {
  margin: 0;
  text-transform: uppercase;
}

.inc-exp-container {
  background-color: #fff;
  box-shadow: var(--box--shadow);
  padding: 20px;
  display: flex;
  justify-content: center;
  margin: 20px 0;
}

.inc-exp-container > div {
```

```css
.inc-exp-container > div {
  flex: 1;
  text-align: center;
}

.inc-exp-container > div:first-of-type {
  border-right: 1px solid #dedede;
}

.money {
  font-size: 2;
  letter-spacing: 1px;
  margin: 5px 0;
}
.money-plus {
  color: #2ecc71;
}

.money-minus {
  color: #c0392b;
}

label {
  display: inline-block;
  margin: 10px 0;
}

input[type="text"],
input[type="number"] {
  border: 1px solid #dedede;
  border-radius: 2px;
  display: block;
  font-size: 16px;
  padding: 10px;
  width: 100%;
}

.btn {
  cursor: pointer;
  background-color: #9c88ff;
  box-shadow: var(--box-shadow);
  color: #fff;
  border: 0;
  display: block;
  font-size: 16px;
  margin: 10px 0 30px;
  padding: 10px;
  width: 100%;
}

.btn:focus,
.delete-btn:focus {
  outline: 0;
}
```

```css
.list {
  list-style-type: none;
  padding: 0;
  margin-bottom: 40px;
}

.list li {
  background-color: #fff;
  box-shadow: var(--box--shadow);
  color: #333;
  display: flex;
  justify-content: space-between;
  position: relative;
  padding: 10px;
  margin: 10px 0;
}

.list li.plus {
  border-right: 5px solid #2ecc71;
}

.list li.minus {
  border-right: 5px solid #c0392b;
}

.delete-btn {
  cursor: pointer;
  background-color: #e74c3c;
  border: 0;
  color: #fff;
  font-size: 20px;
  line-height: 20px;
  padding: 2px 5px;
  position: absolute;
  top: 50%;
  left: 0;
  transform: translate(-100%, -50%);
  opacity: 0;
  transition: opacity 0.3s ease;
}

.list li:hover .delete-btn {
  opacity: 1;
}
```

JS (Java Script):-

JavaScript is essential in an expense tracker application for creating dynamic and interactive functionalities. With JavaScript, developers can implement features such as real-time calculation of expenses, validation of user inputs, and dynamic updates to the UI based on user actions.

Code:-

```javascript
//1
const balance = document.getElementById(
  "balance"
);
const money_plus = document.getElementById(
  "money-plus"
);
const money_minus = document.getElementById(
  "money-minus"
);
const list = document.getElementById("list");
const form = document.getElementById("form");
const text = document.getElementById("text");
const amount = document.getElementById("amount");

const localStorageTransactions = JSON.parse(localStorage.getItem('transactions'));

let transactions = localStorage.getItem('transactions') !== null ? localStorageTransactions : [];

function addTransaction(e){
  e.preventDefault();
  if(text.value.trim() === '' || amount.value.trim() === ''){
    alert('please add text and amount')
  }else{
    const transaction = {
      id:generateID(),
      text:text.value,
      amount:+amount.value
    }
```

```javascript
    transactions.push(transaction);

    addTransactionDOM(transaction);
    updateValues();
    updateLocalStorage();
    text.value='';
    amount.value='';
  }
}
function generateID(){
  return Math.floor(Math.random()*100000000);
}
function addTransactionDOM(transaction) {
  const sign = transaction.amount < 0 ? "-" : "+";
  const item = document.createElement("li");
  item.classList.add(
    transaction.amount < 0 ? "minus" : "plus"
  );

  item.innerHTML = `
    ${transaction.text} <span>${sign}${Math.abs(
    transaction.amount
  )}</span>
    <button class="delete-btn" onclick="removeTransaction(${transaction.id})">x</button>
    `;
```

```javascript
function addTransactionDOM(transaction) {
  list.appendChild(item);
}
function updateValues() {
  const amounts = transactions.map(
    (transaction) => transaction.amount
  );
  const total = amounts
    .reduce((acc, item) => (acc += item), 0)
    .toFixed(2);
  const income = amounts
    .filter((item) => item > 0)
    .reduce((acc, item) => (acc += item), 0)
    .toFixed(2);
  const expense =
    (amounts
      .filter((item) => item < 0)
      .reduce((acc, item) => (acc += item), 0) *
    -1).toFixed(2);

    balance.innerText=`$${total}`;
```

```javascript
    money_plus.innerText = `$${income}`;
    money_minus.innerText = `$${expense}`;
}
function removeTransaction(id){
  transactions = transactions.filter(transaction => transaction.id !== id);
  updateLocalStorage();
  Init();
}
function updateLocalStorage(){
  localStorage.setItem('transactions',JSON.stringify(transactions));
}
function Init() {
  list.innerHTML = "";
  transactions.forEach(addTransactionDOM);
  updateValues();
}
Init();
form.addEventListener('submit',addTransaction);
```

Features:-

Efficient expense logging with customizable categories and budget tracking functionalities, coupled with visualizations for expense analysis and financial planning.

Design and Implementation:-

Designed with a user-friendly interface using HTML/CSS, and implemented with JavaScript for dynamic expense entry and storage utilizing local storage or database integration.

Challenges Faced:-

Managing data security and privacy concerns while integrating with financial APIs for real-time transaction syncing, and ensuring cross-platform compatibility for seamless user experience.

Future Enhancements :-

Implementing machine learning algorithms for expense pattern analysis and predictive budgeting, alongside mobile app development for on-the-go expense tracking.

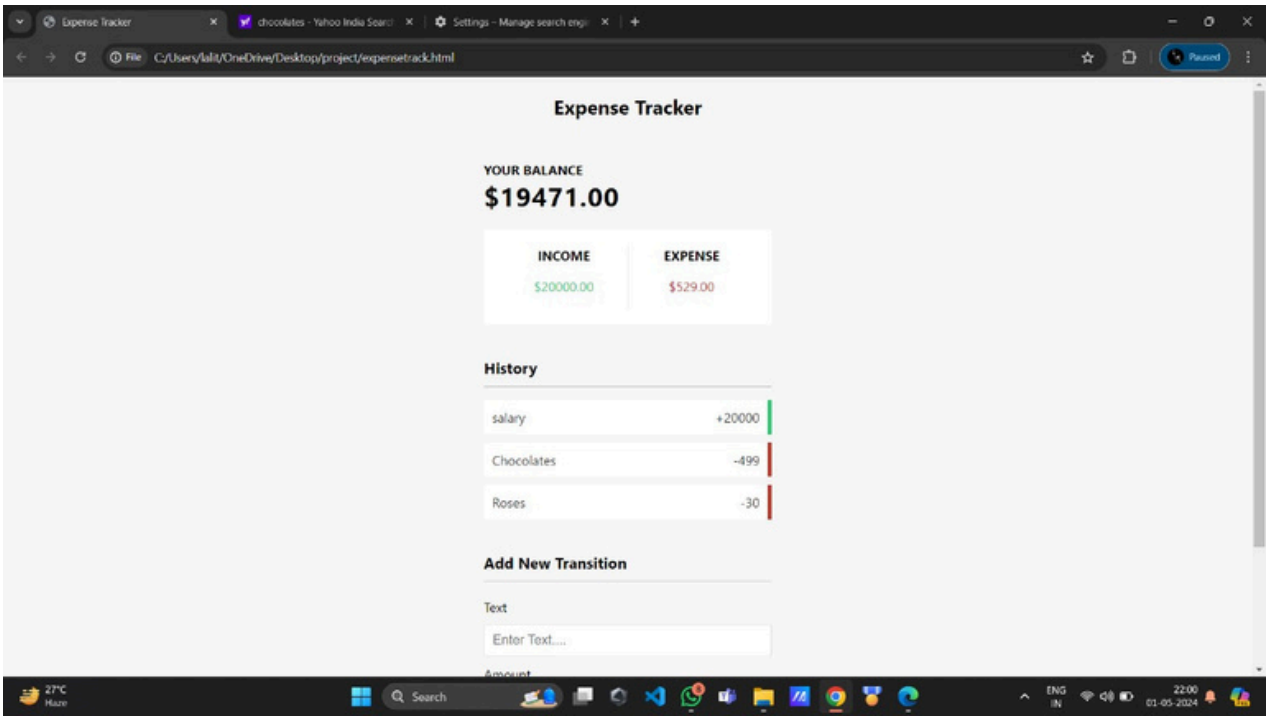
Conclusion:-

Expense trackers provide invaluable assistance in financial management, fostering better budgeting habits and financial awareness for improved financial health.

References:-

Popular expense tracker applications like Mint and Expensify offer comprehensive solutions for personal and business finance management, trusted by millions worldwide for efficient expense tracking and budgeting.

Output:-



PROJECT - 7

MEMORY GAME

Introduction:-

The Memory Game program is a digital adaptation of the classic card-matching game, offering players an entertaining way to test and enhance their memory skills. Players are presented with a grid of facedown cards, each containing unique symbols, and must flip them over to find matching pairs. With each successful match, cards remain face up, revealing more symbols as players race against the clock to uncover all pairs. Developed using languages like JavaScript, Python, or Java, along with HTML and CSS for intuitive interfaces, the program provides a seamless and enjoyable gaming experience for players of all ages, fostering cognitive stimulation and enjoyment.

Project Overview:-

A classic memory game project featuring card matching gameplay, customizable difficulty levels, and interactive animations for a fun and challenging user experience.

Technology Used:-

HTML (HyperText Markup Language):-

In an HTML-based memory game program, HTML is used to structure the game's interface. Elements like are employed to represent cards, buttons, and the game board. Each card may have attributes like data-card to distinguish them.

Code:-

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Memory Game</title>
    <link
      href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
      rel="stylesheet"
    />
    <link rel="stylesheet" href="memory game.css" />
  </head>
  <body>
    <div class="wrapper">
      <div class="stats-container">
        <div id="moves-count"></div>
        <div id="time"></div>
      </div>
      <div class="game-container"></div>
      <button id="stop" class="hide">Stop Game</button>
    </div>
    <div class="controls-container">
      <p id="result"></p>
      <button id="start">Start Game</button>
    </div>
    <script src="memory game.js"></script>
  </body>
</html>
```

CSS (Cascading Style Sheet):-

In a memory game program, CSS (Cascading Style Sheets) is utilized to define the visual appearance and layout of game elements. This includes styling for cards, the game board, buttons, and any other interface components.

Code:-

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}
body {
  background-color: #f4c531;
}
.wrapper {
  box-sizing: content-box;
  width: 26.87em;
  padding: 2.5em 3em;
  background-color: #ffffff;
  position: absolute;
  transform: translate(-50%, -50%);
  left: 50%;
  top: 50%;
  border-radius: 0.6em;
  box-shadow: 0 0.9em 2.8em rgba(86, 66, 0, 0.2);
}
.game-container {
  position: relative;
  width: 100%;
  display: grid;
  gap: 0.6em;
}
.stats-container {
  text-align: right;
  margin-bottom: 1.2em;
}
.stats-container span {
  font-weight: 600;
}
.card-container {
  position: relative;
  width: 6.25em;
  height: 6.25em;
  cursor: pointer;
}
.card-before,
.card-after {
  position: absolute;
  border-radius: 5px;
  width: 100%;
  height: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  border: 4px solid #000000;
  transition: transform 0.7s ease-out;
  backface-visibility: hidden;
}
.card-before {
  background-color: #f4c531;
```

```
.card-before {
  font-size: 2.8em;
  font-weight: 600;
}
.card-after {
  background-color: #ffffff;
  transform: rotateY(180deg);
}
.card-container.flipped .card-before {
  transform: rotateY(180deg);
}
.card-container.flipped .card-after {
  transform: rotateY(0deg);
}
.controls-container {
  position: absolute;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  width: 100%;
  height: 100%;
  background-color: #f4c531;
  top: 0;
}
button {
  border: none;
  border-radius: 0.3em;
  padding: 1em 1.5em;
  cursor: pointer;
}
#stop {
  font-size: 1.1em;
  display: block;
  margin: 1.1em auto 0 auto;
  background-color: #000000;
  color: #ffffff;
}
.controls-container button {
  font-size: 1.3em;
  box-shadow: 0 0.6em 2em rgba(86, 66, 0, 0.2);
}
.hide {
  display: none;
}
#result {
  text-align: center;
}
#result h2 {
  font-size: 2.5em;
}
#result h4 {
  font-size: 1.8em;
  margin: 0.6em 0 1em 0;
```

JS (Java Script):-

In a memory game program, JavaScript is crucial for implementing game logic and interactivity. JavaScript handles events like card clicks, manages the game state (e.g., which cards are flipped), and determines whether pairs match. It also controls timing, such as tracking the player's progress and managing timers.

Code:-

```javascript
const moves = document.getElementById("moves-count");
const timeValue = document.getElementById("time");
const startButton = document.getElementById("start");
const stopButton = document.getElementById("stop");
const gameContainer = document.querySelector(".game-container");
const result = document.getElementById("result");
const controls = document.querySelector(".controls-container");
let cards;
let interval;
let firstCard = false;
let secondCard = false

const items = [
  { name: "bee", image: "bee.png" },
  { name: "crocodile", image: "crocodile.png" },
  { name: "macaw", image: "macaw.png" },
  { name: "gorilla", image: "gorilla.png" },
  { name: "tiger", image: "tiger.png" },
  { name: "monkey", image: "monkey.png" },
  { name: "chameleon", image: "chameleon.png" },
  { name: "piranha", image: "piranha.png" },
  { name: "anaconda", image: "anaconda.png" },
  { name: "sloth", image: "sloth.png" },
  { name: "cockatoo", image: "cockatoo.png" },
  { name: "toucan", image: "toucan.png" },
];


let seconds = 0,
  minutes = 0;

let movesCount = 0,
  winCount = 0;


const timeGenerator = () => {
  seconds += 1;

  if (seconds >= 60) {
    minutes += 1;
    seconds = 0;
  }
  let secondsValue = seconds < 10 ? `0${seconds}` : seconds;
  let minutesValue = minutes < 10 ? `0${minutes}` : minutes;
  timeValue.innerHTML = `<span>Time:</span>${minutesValue}:${secondsValue}`;
};
const movesCounter = () => {
  movesCount += 1;
  moves.innerHTML = `<span>Moves:</span>${movesCount}`;
};
const generateRandom = (size = 4) => {
  let tempArray = [...items];
  let cardValues = [];
  size = (size * size) / 2;
  for (let i = 0; i < size; i++) {
```

```javascript
const generateRandom = (size = 4) => {
    const randomIndex = Math.floor(Math.random() * tempArray.length);
    cardValues.push(tempArray[randomIndex]);
    tempArray.splice(randomIndex, 1);
  }
  return cardValues;
};

const matrixGenerator = (cardValues, size = 4) => {
  gameContainer.innerHTML = "";
  cardValues = [...cardValues, ...cardValues];
  cardValues.sort(() => Math.random() - 0.5);
  for (let i = 0; i < size * size; i++) {
    gameContainer.innerHTML += `
     <div class="card-container" data-card-value="${cardValues[i].name}">
        <div class="card-before">?</div>
        <div class="card-after">
        <img src="${cardValues[i].image}" class="image"/></div>
     </div>
     `;
  }
  gameContainer.style.gridTemplateColumns = `repeat(${size},auto)`;
  cards = document.querySelectorAll(".card-container");
  cards.forEach((card) => {
    card.addEventListener("click", () => {
      if (!card.classList.contains("matched")) {
        card.classList.add("flipped");
        if (!firstCard) {
          firstCard = card;
          firstCardValue = card.getAttribute("data-card-value");
        } else {
          movesCounter();
          secondCard = card;
          let secondCardValue = card.getAttribute("data-card-value");
          if (firstCardValue == secondCardValue) {
            firstCard.classList.add("matched");
            secondCard.classList.add("matched");
            firstCard = false;
            winCount += 1;
            if (winCount == Math.floor(cardValues.length / 2)) {
              result.innerHTML = `<h2>You Won</h2>
            <h4>Moves: ${movesCount}</h4>`;
              stopGame();
            }
          } else {
            let [tempFirst, tempSecond] = [firstCard, secondCard];
            firstCard = false;
            secondCard = false;
            let delay = setTimeout(() => {
              tempFirst.classList.remove("flipped");
              tempSecond.classList.remove("flipped");
            }, 900);
          }
        }
      }
```

```javascript
const matrixGenerator = (cardValues, size = 4) => {
  cards.forEach((card) => {
    card.addEventListener("click", () => {
    });
  });
};
startButton.addEventListener("click", () => {
  movesCount = 0;
  seconds = 0;
  minutes = 0;
  controls.classList.add("hide");
  stopButton.classList.remove("hide");
  startButton.lassList.add("hide");
  interval = setInterval(timeGenerator, 1000);
  moves.innerHTML = `<span>Moves:</span> ${movesCount}
  initializer();
});
stopButton.addEventListener(
  "click",
  (stopGame = () => {
    controls.classList.remove("hide");
    stopButton.classList.add("hide");
    startButton.classList.remove("hide");
    clearInterval(interval);
  })
);
```

```javascript
const initializer = () => {
  result.innerText = "";
  winCount = 0;
  let cardValues = generateRandom();
  console.log(cardValues);
  matrixGenerator(cardValues);
};
```

Features:-

Interactive card flipping with randomized positions, coupled with score tracking and timer for challenging gameplay and skill improvement.

Design and Implementation:-

Designed with HTML, CSS, and JavaScript for card layout and interactivity, utilizing event listeners for game logic and state management.

Challenged Faced:-

Managing complex game logic for card matching and state management while optimizing performance for smooth user experience across devices.

Future Enhancement :-

Integrating multiplayer functionality for competitive gaming experiences and implementing various themes and difficulty levels for enhanced gameplay options.

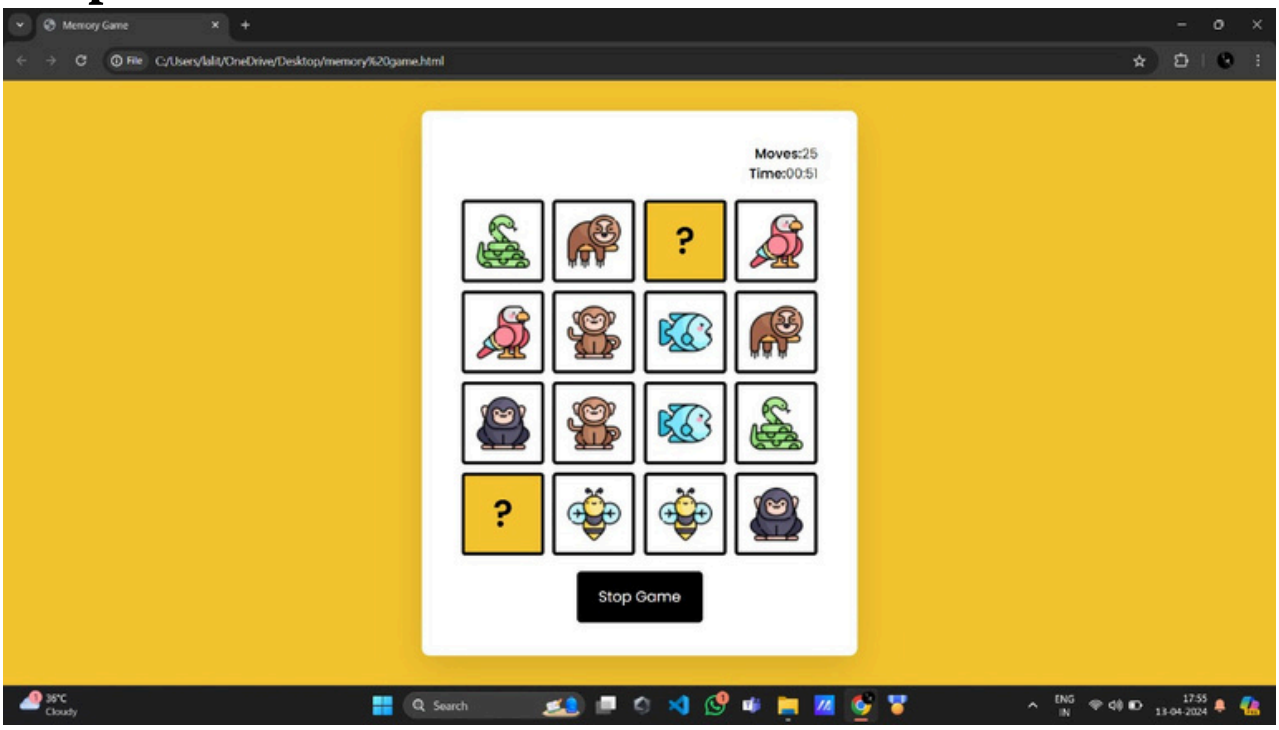
Conclusion:-

Memory games provide engaging cognitive exercises, fostering memory retention and concentration skills in an entertaining and interactive format.

References:-

JavaScript libraries like Phaser and CreateJS offer robust frameworks for developing memory games, providing extensive features for graphics rendering and game logic implementation.

Output:-



PROJECT - 8

CURRENCY CONVERTER

Introduction:-

A Currency Converter program facilitates the conversion of one currency to another based on current exchange rates. It serves as a practical tool for individuals and businesses engaged in international transactions, travel, or investment. Through user-friendly interfaces, users input the amount in one currency and select the desired conversion currency, with the program displaying the equivalent amount. These programs often fetch real-time exchange rates from reliable sources like financial institutions or online APIs to ensure accuracy. Currency Converter programs streamline financial processes, providing convenience and efficiency in managing diverse currency transactions.

Project Overview:-

A currency converter project facilitating real-time exchange rate retrieval and accurate currency conversion, featuring user-friendly interface and customizable currency pairs.

Technology Used:-

HTML (HyperText Markup Language):-

In a Currency Converter program, HTML is used to structure the user interface. It defines elements such as input fields for entering the amount to convert and dropdown menus for selecting the currencies. HTML also includes buttons for initiating the conversion process and displaying the result. Additionally, labels and placeholders provide guidance to users, enhancing the usability of the interface. Through HTML, the Currency Converter program creates a clear and intuitive layout for users to interact with and perform currency conversions seamlessly.

Code:-

```html
<!DOCTYPE MDN Reference
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Currency Converter App in JavaScript | CodingNepal</title>
    <link rel="stylesheet" href="money.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>
  </head>
  <body>
    <div class="wrapper">
      <header>Currency Converter</header>
      <form action="#">
        <div class="amount">
          <p>Enter Amount</p>
          <input type="text" value="1">
        </div>
        <div class="drop-list">
          <div class="from">
            <p>From</p>
            <div class="select-box">
              <img src="https://flagcdn.com/48x36/us.png" alt="flag">
              <select> <!-- Options tag are inserted from JavaScript --> </select>
            </div>
          </div>
```

```html
          </div>
          <div class="exchange-rate">Getting exchange rate...</div>
          <button>Get Exchange Rate</button>
        </form>
      </div>

      <script src="country list.js"></script>
      <script src="money.js"></script>

    </body>
</html>
```

CSS (Cascading Style Sheet):-

In a Currency Converter program, CSS (Cascading Style Sheets) is utilized to style and customize the appearance of the user interface elements defined in HTML.

Code:-

```css
/* Import Google Font - Poppins */
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=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;
  padding: 0 10px;
  background: #675AFE;
}
::selection{
  color: #fff;
  background: #675AFE;
}
.wrapper{
  width: 370px;
  padding: 30px;
  border-radius: 7px;
  background: #fff;
  box-shadow: 7px 7px 20px rgba(0, 0, 0, 0.05);
}
.wrapper header{
  font-size: 28px;
  font-weight: 500;
  text-align: center;
}
.wrapper form{
  margin: 40px 0 20px 0;
}
form :where(input, select, button){
  width: 100%;
  outline: none;
  border-radius: 5px;
  border: none;
}
form p{
  font-size: 18px;
  margin-bottom: 5px;
}
form input{
  height: 50px;
  font-size: 17px;
  padding: 0 15px;
  border: 1px solid #999;
}
form input:focus{
  padding: 0 14px;
  border: 2px solid #675AFE;
}
form .drop-list{
  display: flex;
  margin-top: 20px;
  align-items: center;
  justify-content: space-between;
}
```

```css
.drop-list .select-box{
  display: flex;
  width: 115px;
  height: 45px;
  align-items: center;
  border-radius: 5px;
  justify-content: center;
  border: 1px solid #999;
}
.select-box img{
  max-width: 21px;
}
.select-box select{
  width: auto;
  font-size: 16px;
  background: none;
  margin: 0 -5px 0 5px;
}
.select-box select::-webkit-scrollbar{
  width: 8px;
}
.select-box select::-webkit-scrollbar-track{
  background: #fff;
}
.select-box select::-webkit-scrollbar-thumb{
  background: #888;
  border-radius: 8px;
  border-right: 2px solid #ffffff;
}
.drop-list .icon{
  cursor: pointer;
  margin-top: 30px;
  font-size: 22px;
}
form .exchange-rate{
  font-size: 17px;
  margin: 20px 0 30px;
}
form button{
  height: 52px;
  color: #fff;
  font-size: 17px;
  cursor: pointer;
  background: #675AFE;
  transition: 0.3s ease;
}
form button:hover{
  background: #4534fe;
}
```

JS (Java Script):-

In a Currency Converter program, JavaScript handles the conversion logic and dynamic updates of currency exchange rates. It retrieves data from external APIs or sources to fetch the latest exchange rates and performs calculations based on user input.

Code:-

```javascript
const droplist = document.querySelectorAll("form select"),
fromCurrency = document.querySelector(".from select"),
toCurrency = document.querySelector(".to select"),
getButton = document.querySelector("form button");

for (let i = 0; i < droplist.length; i++) {
    for(let currency_code in country_list){
        // selecting USD by default as FROM currency and NPR as TO currency
        let selected = i == 0 ? currency_code == "USD" ? "selected" : "" : currency_code == "NPR" ? "selected" : "";
        // creating option tag with passing currency code as a text and value
        let optionTag = `<option value="${currency_code}" ${selected}>${currency_code}</option>`;
        // inserting options tag inside select tag
        droplist[i].insertAdjacentHTML("beforeend", optionTag);
    }
    droplist[i].addEventListener("change", e =>{
        loadFlag(e.target); // calling loadFlag with passing target element as an argument
    });
}

function loadFlag(element){
    for(let code in country_list){
        if(code == element.value){ // if currency code of country list is equal to option value
            let imgTag = element.parentElement.querySelector("img"); // selecting img tag of particular drop list
            // passing country code of a selected currency code in a img url
            imgTag.src = `https://flagcdn.com/48x36/${country_list[code].toLowerCase()}.png`;
        }
    }
}

window.addEventListener("load", ()=>{
    getExchangeRate();
});

getButton.addEventListener("click", e =>{
    e.preventDefault(); //preventing form from submitting
    getExchangeRate();
});

const exchangeIcon = document.querySelector("form .icon");
exchangeIcon.addEventListener("click", ()=>{
    let tempCode = fromCurrency.value; // temporary currency code of FROM drop list
    fromCurrency.value = toCurrency.value; // passing TO currency code to FROM currency code
    toCurrency.value = tempCode; // passing temporary currency code to TO currency code
    loadFlag(fromCurrency); // calling loadFlag with passing select element (fromCurrency) of FROM
    loadFlag(toCurrency); // calling loadFlag with passing select element (toCurrency) of TO
    getExchangeRate(); // calling getExchangeRate
})

function getExchangeRate(){
    const amount = document.querySelector("form input");
    const exchangeRateTxt = document.querySelector("form .exchange-rate");
    let amountVal = amount.value;
    // if user don't enter any value or enter 0 then we'll put 1 value by default in the input field
    if(amountVal == "" || amountVal == "0"){
        amount.value = "1";
        amountVal = 1;
    }
    exchangeRateTxt.innerText = "Getting exchange rate...";
    let url = `https://v6.exchangerate-api.com/v6/YOUR-API-KEY/latest/${fromCurrency.value}`;
    // fetching api response and returning it with parsing into js obj and in another then method receiving that obj
    fetch(url).then(response => response.json()).then(result =>{
        let exchangeRate = result.conversion_rates[toCurrency.value]; // getting user selected TO currency rate
        let totalExRate = (amountVal * exchangeRate).toFixed(2); // multiplying user entered value with selected TO currency rate
        exchangeRateTxt.innerText = `${amountVal} ${fromCurrency.value} = ${totalExRate} ${toCurrency.value}`;
    }).catch(() =>{ // if user is offline or any other error occured while fetching data then catch function will run
        exchangeRateTxt.innerText = "Something went wrong";
    });
}
```

Feature:-

Real-time currency exchange rates updated from reliable sources, supporting conversion between multiple currencies with precision and ease.

Design and Implementation :-

Designed with HTML, CSS, and JavaScript for user interface and functionality, utilizing API integration for live exchange rate data retrieval.

Challenged Faced:-

Ensuring data accuracy and reliability from external APIs while handling currency fluctuations and managing user input validation for seamless conversion.

Future Enhancements :-

Implementing advanced charting tools for historical exchange rate analysis and integrating cryptocurrency support for broader currency conversion capabilities.

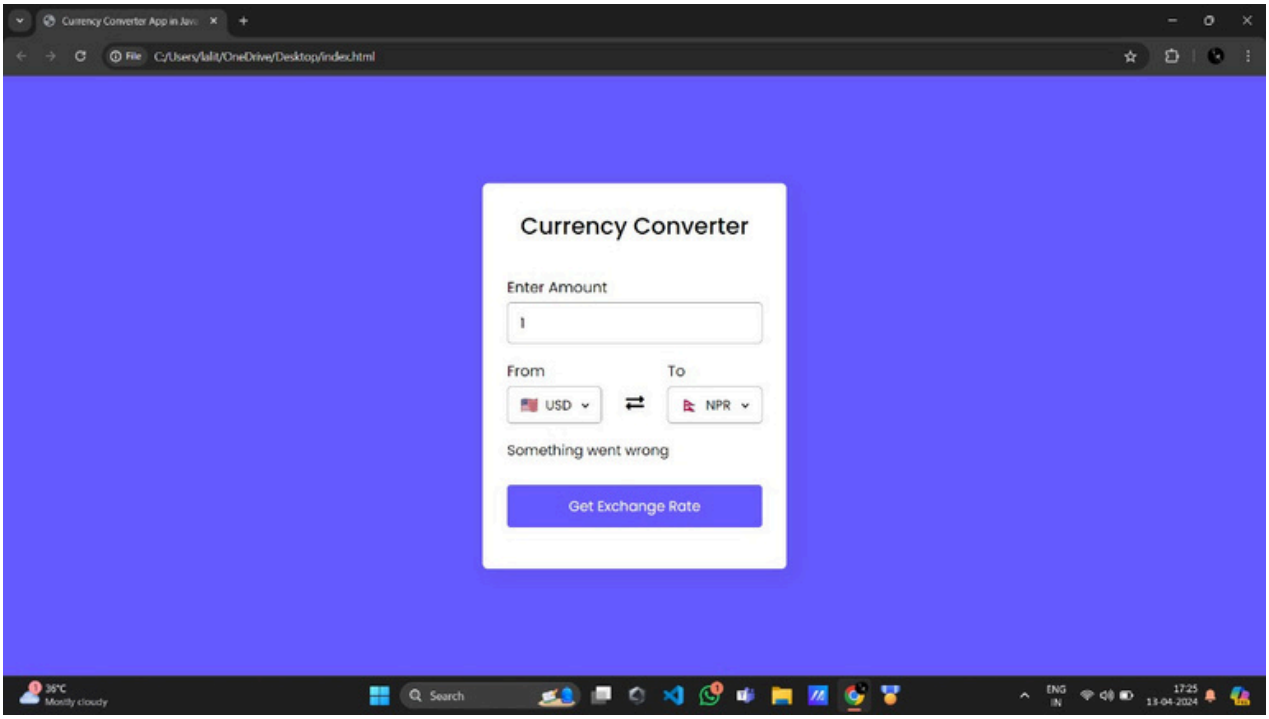
Conclusion:-

Currency converters offer vital tools for seamless international transactions, empowering users with accurate exchange rate information and facilitating effortless currency conversions.

References:-

Currency converter tools like XE and OANDA provide reliable exchange rate data and user-friendly interfaces, trusted by individuals and businesses for international financial transactions.

Output:-



PROJECT - 9

MUSIC PLAYER

Introduction:-

A music player program is software designed to play audio files on a computer or mobile device. It typically allows users to organize their music library, create playlists, and control playback features such as play, pause, skip, and repeat. Some advanced music players also offer features like equalization, visualizations, and integration with online streaming services.

Project Overview:-

A music player project enabling users to organize and play their music library, featuring customizable playlists, playback controls, and audio visualization for an immersive listening experience.

Technology Used:-

HTML (HyperText Markup Language):-

HTML is commonly used in music player programs to create the user interface (UI) elements that allow users to interact with the software. HTML is used to structure the layout of the player, including buttons for play, pause, skip, volume control, and playlist management. Additionally, HTML is utilized to display metadata such as song titles, artist names, and album artwork.

Code:-

```html
<!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>Music Player</title>
    <link rel="stylesheet" href="musicplayer.css">
    <script src="musicplayer.js" defer></script>
    <link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.0/css/all.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
  <div class="player">
      <div class="wrapper">
          <div class="details">
              <div class="now-playing">PLAYING x OF y</div>
              <div class="track-art"></div>
              <div class="track-name">Track Name</div>
              <div class="track-artist">Track Artist</div>
          </div>

          <div class="slider_container">
              <div class="current-time">00:00</div>
               <input type="range" min="1" max="100" value="0" class="seek_slider" onchange="seekTo()">
               <div class="total-duration">00:00</div>
          </div>

          <div class="slider_container">
              <i class="fa fa-volume-down"></i>
               <input type="range" min="1" max="100" value="99" class="volume_slider" onchange="setVolume()">
               <i class="fa fa-volume-up"></i>
          </div>

          <div class="buttons">
              <div class="random-track" onclick="randomTrack()">
                  <i class="fas fa-random fa-2x" title="random"></i>
              </div>
              <div class="prev-track" onclick="prevTrack()">
                  <i class="fa fa-step-backward fa-2x"></i>
              </div>
```

```html
              <div class="playpause-track" onclick="playpauseTrack()">
                  <i class="fa fa-play-circle fa-5x"></i>
              </div>
              <div class="next-track" onclick="nextTrack()">
                  <i class="fa fa-step-forward fa-2x"></i>
              </div>
              <div class="repeat-track" onclick="repeatTrack()">
                  <i class="fa fa-repeat fa-2x" title="repeat"></i>
              </div>
          </div>

          <div id="wave">
              <span class="stroke"></span>
              <span class="stroke"></span>
              <span class="stroke"></span>
              <span class="stroke"></span>
              <span class="stroke"></span>
              <span class="stroke"></span>
              <span class="stroke"></span>
          </div>
      </div>
  </div>
</body>
</html>
```

CSS (Cascading Style Sheet):-

CSS (Cascading Style Sheets) is essential in music player programs for styling the HTML elements that comprise the user interface. It's used to define the colors, fonts, sizes, and layout of the player's components, ensuring a visually appealing and cohesive design.

Code:-

```css
body {
    font-family: Arial, Helvetica, sans-serif;
    background: linear-gradient
    (to right, #9a1d86, #1595be);
    font-weight: bold;
    }

    .player{
      height: 95vh;
      display: flex;
      align-items: center;
      flex-direction: column;
      justify-content: center;
  }
    .wrapper{
      border: 1px solid transparent;
      padding: 30px;
      border-radius: 20px;
      background-color: #ddd;
      box-shadow: rgba(0, 0, 0, 0.3)
      0px 19px 38px, rgba(0, 0, 0, 0.22)
      0px 15px 12px;
    }

    .details {
    display: flex;
    align-items: center;
    flex-direction: column;
    justify-content: center;
    }

    .track-art {
    margin: 25px;
    height: 250px;
    width: 250px;
    border:2px solid #FFFAFA;
    background-size: cover;
    background-position: center;
    border-radius: 50%;
    -moz-box-shadow: 0px 6px 5px black;
    -webkit-box-shadow: 0px 6px 5px black;
    box-shadow: 0px 6px 5px black;
    -moz-border-radius:190px;
    -webkit-border-radius:190px;
    border-radius:190px;
    }

    .now-playing {
    font-size: 1rem;
    }

    .track-name {
    font-size: 2.5rem;
    }

    .track-artist {
    margin-top: 5px;
    font-size: 1.5rem;
    }

    .buttons {
    display: flex;
    flex-direction: row;
    align-items: center;
    margin-bottom: 30px;
    }
    .active{
        color: black;
    }
```

```css
.repeat-track,
.random-track,
.playpause-track,
.prev-track,
.next-track {
padding: 25px;
opacity: 0.8;
transition: opacity .2s;
}

.repeat-track:hover,
.random-track:hover,
.playpause-track:hover,
.prev-track:hover,
.next-track:hover {
opacity: 1.0;
}

.slider_container {
display: flex;
justify-content: center;
align-items: center;
}

.seek_slider, .volume_slider {
-webkit-appearance: none;
-moz-appearance: none;
appearance: none;
height: 5px;
background: #83A9FF;
-webkit-transition: .2s;
transition: opacity .2s;
}

.seek_slider::-webkit-slider-thumb,
.volume_slider::-webkit-slider-thumb {
-webkit-appearance: none;
-moz-appearance: none;
appearance: none;
width: 15px;
height: 15px;
background: white;
border: 3px solid #3774FF;
cursor: grab;
border-radius: 100%;
}

.seek_slider:hover,
.volume_slider:hover {
opacity: 1.0;
}

.seek_slider {
width: 60%;
}

.volume_slider {
width: 30%;
}

.current-time,
.total-duration {
padding: 10px;
}
```

```css
i.fa-volume-down,
i.fa-volume-up {
padding: 10px;
}

i,
i.fa-play-circle,
i.fa-pause-circle,
i.fa-step-forward,
i.fa-step-backward,
p {
cursor: pointer;
}
.randomActive{
    color: black;
}
.rotate {
    animation: rotation 8s infinite linear;
}
@keyframes rotation {
    from {
      transform: rotate(0deg);
    }
    to {
      transform: rotate(359deg);
    }
}
.loader {
  height: 70px;
  display: flex;
  justify-content: center;
  align-items: center;
}
.loader .stroke{
    background: #f1f1f1;
    height: 150%;
    width: 10px;
    border-radius: 50px;
    margin: 0 5px;
    animation: animate 1.4s linear infinite;
}
  @keyframes animate {
    50% {
      height: 20%;
      background: #4286f4;
    }

    100% {
      background: #4286f4;
      height: 100%;
    }
  }
  .stroke:nth-child(1){
      animation-delay: 0s;
  }
  .stroke:nth-child(2){
    animation-delay: 0.3s;
}
.stroke:nth-child(3){
    animation-delay: 0.6s;
}
.stroke:nth-child(4){
    animation-delay: 0.9s;
}
.stroke:nth-child(5){
    animation-delay: 0.6s;
```

JS (Java Script):-

JavaScript is a crucial component of music player programs as it enables interactivity and dynamic functionality. In a music player, JavaScript is used to handle user interactions such as clicking on play, pause, or skip buttons, as well as managing the playback of audio files.

Code:-

```
let now_playing = document.querySelector('.now-playing');
let track_art = document.querySelector('.track-art');
let track_name = document.querySelector('.track-name');
let track_artist = document.querySelector('.track-artist');

let playpause_btn = document.querySelector('.playpause-track');
let next_btn = document.querySelector('.next-track');
let prev_btn = document.querySelector('.prev-track');

let seek_slider = document.querySelector('.seek_slider');
let volume_slider = document.querySelector('.volume_slider');
let curr_time = document.querySelector('.current-time');
let total_duration = document.querySelector('.total-duration');
let wave = document.getElementById('wave');
let randomIcon = document.querySelector('.fa-random');
let curr_track = document.createElement('audio');

let track_index = 0;
let isPlaying = false;
let isRandom = false;
let updateTimer;

const music_list = [
    {
        img : 'images/stay.png',
        name : 'Stay',
        artist : 'The Kid LAROI, Justin Bieber',
        music : 'music/stay.mp3'
    },
    {
        img : 'images/fallingdown.jpg',
        name : 'Falling Down',
        artist : 'Wid Cards',
        music : 'music/fallingdown.mp3'
    },
    {
        img : 'images/faded.png',
        name : 'Faded',
        artist : 'Alan Walker',
        music : 'music/Faded.mp3'
    },
    {
        img : 'images/ratherbe.jpg',
        name : 'Rather Be',
        artist : 'Clean Bandit',
        music : 'music/Rather Be.mp3'
    }
];

loadTrack(track_index);

function loadTrack(track_index){
    clearInterval(updateTimer);
    reset();

    curr_track.src = music_list[track_index].music;
    curr_track.load();

    track_art.style.backgroundImage = "url(" + music_list[track_index].img + ")";
    track_name.textContent = music_list[track_index].name;
    track_artist.textContent = music_list[track_index].artist;
    now_playing.textContent = "Playing music " + (track_index + 1) + " of " + music_list.length;

    updateTimer = setInterval(setUpdate, 1000);

    curr_track.addEventListener('ended', nextTrack);
    random_bg_color();
  loadTrack(track_index);
  playTrack();
}
function prevTrack(){
  if(track_index > 0){
      track_index -= 1;
  }else{
      track_index = music_list.length -1;
  }
  loadTrack(track_index);
  playTrack();
}
function seekTo(){
  let seekto = curr_track.duration * (seek_slider.value / 100);
  curr_track.currentTime = seekto;
}
function setVolume(){
  curr_track.volume = volume_slider.value / 100;
}
function setUpdate(){
  let seekPosition = 0;
  if(!isNaN(curr_track.duration)){
      seekPosition = curr_track.currentTime * (100 / curr_track.duration);
      seek_slider.value = seekPosition;
```

```
    let currentMinutes = Math.floor(curr_track.currentTime / 60);
    let currentSeconds = Math.floor(curr_track.currentTime - currentMinutes * 60);
    let durationMinutes = Math.floor(curr_track.duration / 60);
    let durationSeconds = Math.floor(curr_track.duration - durationMinutes * 60);

    if(currentSeconds < 10) {currentSeconds = "0" + currentSeconds; }
    if(durationSeconds < 10) { durationSeconds = "0" + durationSeconds; }
    if(currentMinutes < 10) {currentMinutes = "0" + currentMinutes; }
    if(durationMinutes < 10) { durationMinutes = "0" + durationMinutes; }

    curr_time.textContent = currentMinutes + ":" + currentSeconds;
    total_duration.textContent = durationMinutes + ":" + durationSeconds;
}
```

Features:-

Supports various audio formats with playlist management, equalizer settings, and seamless navigation through tracks for personalized listening experiences.

Design and Implementation :-

Designed with HTML/CSS for interface and JavaScript for functionality, utilizing audio APIs for playback control and data management.

Challenged Faced:-

Ensuring cross-browser compatibility for audio playback and managing large music libraries efficiently while optimizing performance for smooth user experience.

Future Enhancements :-

Implementing machine learning algorithms for personalized music recommendations and integrating social features for music sharing and collaborative playlists.

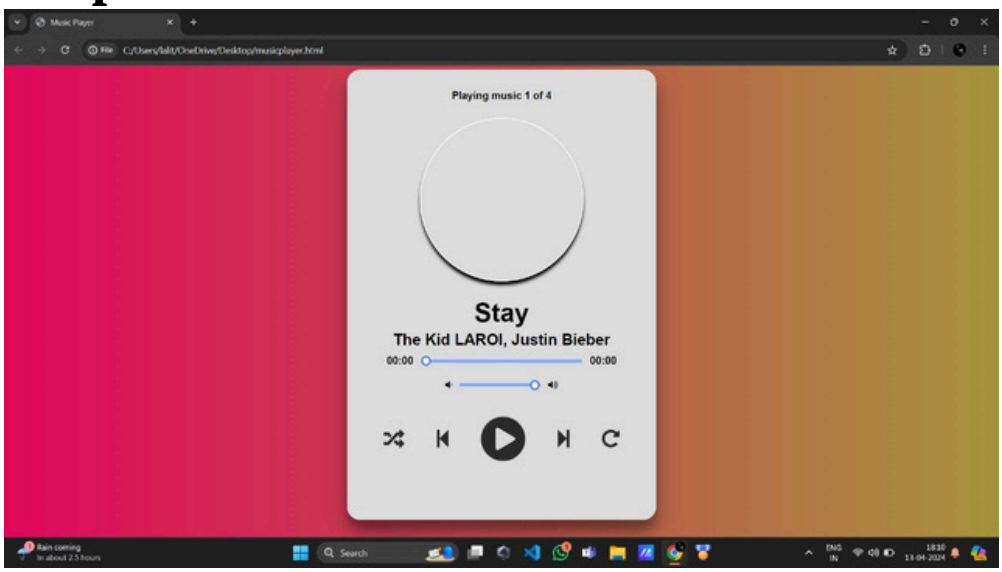
Conclusion:-

Music players serve as versatile platforms for organizing and enjoying diverse audio content, enriching user experiences and fostering musical exploration and enjoyment.

References:-

Popular music players like Winamp and VLC offer comprehensive features for audio playback, playlist management, and customization, trusted by millions worldwide for their reliability and versatility.

Output:-



PROJECT - 10

MOVIE RECOMMENDATION

Introduction:-

Movie recommendation systems are intelligent algorithms designed to suggest films to users based on their preferences, viewing history, and behavior. These systems analyze vast amounts of data, including movie genres, ratings, cast, and user interactions, to generate personalized recommendations. By leveraging machine learning and collaborative filtering techniques, they predict which movies a user might enjoy and present them with tailored suggestions. Ultimately, movie recommendation systems aim to enhance user satisfaction by helping viewers discover new films that align with their tastes and interests, making the movie-watching experience more enjoyable and engaging.

Project Overview:-

A movie recommendation project employing collaborative filtering algorithms to suggest personalized films based on user preferences and viewing history, enhancing the movie-watching experience with tailored suggestions.

Technology Used:-

HTML (HyperText Markup Language):-

In a movie recommendation project, HTML is employed to structure the layout of web pages, defining elements like buttons, forms, and containers. It creates the foundation for presenting movie suggestions, allowing users to navigate through recommendations and interact with the interface. Additionally, HTML incorporates links to external CSS stylesheets and JavaScript files for enhanced styling and functionality, ensuring a visually appealing and responsive user experience.

Code:-

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Movie Recommendation System</title>
    <link rel="stylesheet" href="stylee.css">
</head>
<body>
    <header>
        <h1>Movie Recommendations</h1>
    </header>
    <main>
        <section id="search-section">
            <input type="text" id="searchInput" placeholder="Enter genre (e.g., Action, Romance)">
            <button onclick="fetchMovies()">Get Recommendations</button>
        </section>
        <section id="results-section">

        </section>
    </main>
    <script src="scriptt.js"></script>
</body>
</html>
```

CSS (Cascading Style Sheet):-

CSS is utilized to style the HTML elements, defining properties such as color, font, layout, and responsiveness to enhance the visual presentation of movie recommendations. Selectors target specific elements or classes, applying custom styles to improve the user interface and ensure a cohesive design aesthetic across the application. Media queries enable adaptability to different screen sizes, optimizing the display for various devices and enhancing user experience.

Code:-

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-image: linear-gradient( 179.7deg,  rgba(248,126,126,1) -0.5%, rgba(251,206,143,1) 35.3%, rgba(184,252,233,1) 67.2%, rgba(118,162,229,1) 92.3% );
}
header {
    background-color: #20232a;
    color: #61dafb;
    padding: 10px 20px;
    text-align: center;}
h1 {
    margin: 0;
}
.movie-list {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    padding: 20px;
}
.movie {
    background-image: linear-gradient( 293.5deg,  rgba(181,149,208,1) 3.2%, rgba(251,148,207,1) 9.9%, rgba(181,149,208,1) 22.9%, rgba(251,148,207,1) 36.4%, rgba(181,149,208,1) 50.1%,
rgba(251,148,207,1) 61.1%, rgba(181,149,208,1) 71.2%, rgba(251,148,207,1) 84.2%, rgba(181,149,208,1) 92.2% );
    border: 1px solid #ddd;
    margin: 10px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    width: 300px;
}
.movie2 {
    background-image: radial-gradient( circle farthest-corner at 3.6% 5.5%,  rgba(255,150,197,1) 9.8%, rgba(255,190,94,1) 26.3%, rgba(246,255,94,1) 43.8%, rgba(113,255,129,1) 59.2%,
rgba(187,255,255,1) 74.6%, rgba(214,146,255,1) 89.4% );
    border: 1px solid #ddd;
    margin: 10px;
    padding: 20px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    width: 300px;}
h2 {
    color: #333;
}
```

JS (Java Script):-

JavaScript is employed to implement interactive features like dynamic content loading, user input handling, and AJAX requests for fetching movie data from APIs. Event listeners enable responses to user actions such as clicks or input changes, triggering actions like displaying movie details or updating recommendations in real-time. Additionally, JavaScript facilitates data manipulation and processing, allowing for personalized recommendation algorithms and enhancing the overall functionality of the movie recommendation system.

Code:-

```
const apiKey = '063b9c7527f084b77b36a00e91e16c01';  // You need to replace this with your TMDB API key
const apiUrl = 'https://api.themoviedb.org/3/search/movie?api_key=' + apiKey + '&query=';

function fetchMovies() {
    let input = document.getElementById('searchInput').value;
    if(input) {
        fetch(apiUrl + input)
        .then(response => response.json())
        .then(data => displayMovies(data.results))
        .catch(error => console.error('Error:', error));
    } else {
        alert("Please enter a movie genre to search.");
    }
}

function displayMovies(movies) {
    const resultsSection = document.getElementById('results-section');
    resultsSection.innerHTML = ''; // Clear previous results
    movies.forEach(movie => {
        const movieDiv = document.createElement('div');
        movieDiv.classList.add('movie');
        movieDiv.innerHTML = `
            <h3>${movie.title}</h3>
            <p>${movie.overview}</p>
        `;
        resultsSection.appendChild(movieDiv);
    });
}
```

Features:-

Personalized film suggestions based on user preferences and viewing history, enhancing the movie-watching experience with tailored recommendations.

Design and Implementation:-

Utilizing collaborative filtering algorithms for personalized movie suggestions, implemented with database integration and user interface design for seamless user experience.

Challenged Faced:-

Addressing data sparsity and cold start problems while ensuring algorithm scalability and relevance of recommendations amidst evolving user preferences.

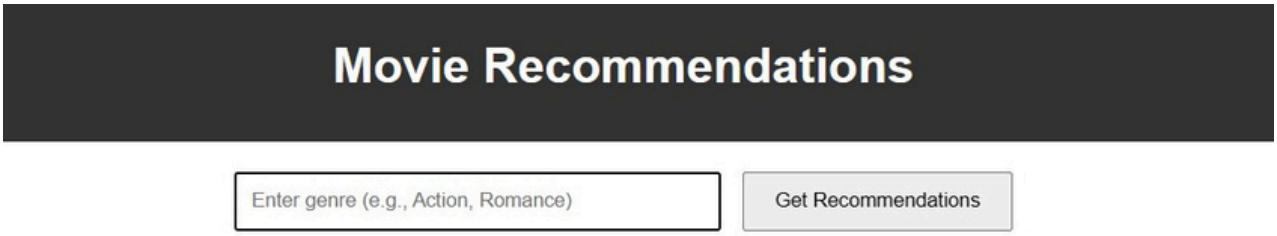
Future Enhancements:-

Incorporating deep learning models for more accurate user preference prediction and integrating social network data for enhanced recommendation diversity.

Conclusion:-

Movie recommendation systems optimize user satisfaction by offering personalized film suggestions, enriching the movie-watching experience with tailored content discovery.

References:-

Platforms like Netflix and Amazon Prime utilize sophisticated recommendation algorithms, enhancing user engagement by suggesting relevant movies based on viewing history and preferences.

Output:-