NAME: SIDDHARTH SHEOKAND

REG NO.: 22BCE0662

ASSIGNMENT -3

CODE FOR Q1:-

Exp1.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial scale=1.0" />
    <title>Night Sky Animation</title>
  </head>
  <body>
    <canvas id="canvas" width="600" height="400"></canvas>
    <!--<script src="exp1.js"></script>-->
    <script src="exp2.js"></script>
  </body>
</html>
```

Exp2.js

```javascript
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");

const width = canvas.width;
const height = canvas.height;

// Night sky gradient function
function drawNightSky() {
  const gradient = ctx.createLinearGradient(0, 0, width, height);
  gradient.addColorStop(0, "#1b2430"); // Dark blue at the top
  gradient.addColorStop(0.5, "#2c3e50"); // Darker blue in the middle
  gradient.addColorStop(1, "#344e8c"); // Darkest blue at the bottom
  ctx.fillStyle = gradient;
  ctx.fillRect(0, 0, width, height);
}

// Function to draw the moon with a slight glow
function drawMoon(x, y, radius) {
  ctx.beginPath();
  ctx.arc(x, y, radius, Math.PI, Math.PI * 2, false); // Draw crescent moon
shape
  ctx.fillStyle = "white";
  ctx.fill();
```

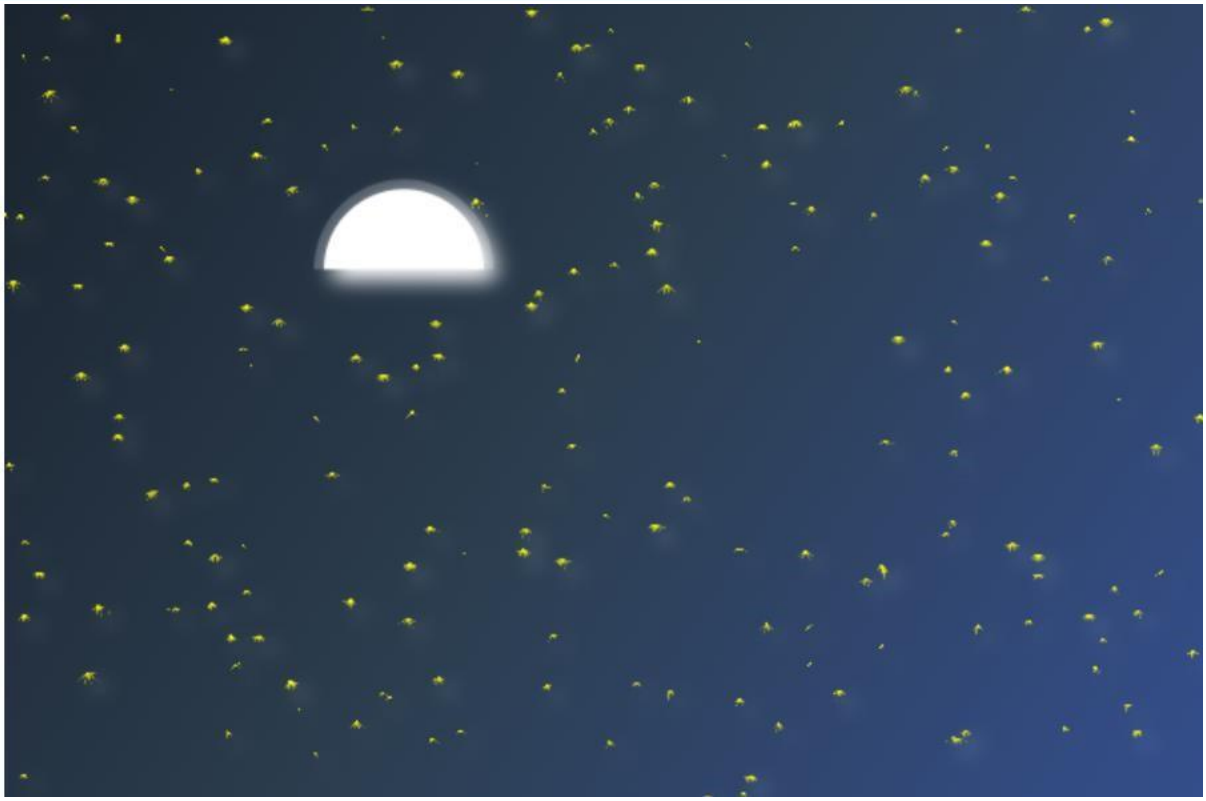
```
  // Create a slight glow effect
  ctx.shadowColor = "white";
  ctx.shadowBlur = 10;
  ctx.shadowOffsetX = 5;
  ctx.shadowOffsetY = 5;
  ctx.beginPath();
  ctx.arc(x, y, radius + 5, Math.PI, Math.PI * 2, false);
  ctx.fillStyle = "rgba(255, 255, 255, 0.2)"; // Transparent white
  ctx.fill();
  ctx.shadowColor = "none"; // Reset shadow
}

// Function to draw a star
function drawStar(x, y, radius, points) {
  const angle = Math.PI / points;
  let innerRadius = radius * 0.33;
  let cornerX, cornerY;

  ctx.beginPath();
  ctx.moveTo(x, y);
  for (let i = 0; i < points; i++) {
    cornerX = x + radius * Math.cos(i * angle);
    cornerY = y + radius * Math.sin(i * angle);
    ctx.lineTo(cornerX, cornerY);

    innerRadius = radius * 0.33;
    cornerX = x + innerRadius * Math.cos((i + 0.5) * angle);
    cornerY = y + innerRadius * Math.sin((i + 0.5) * angle);
    ctx.lineTo(cornerX, cornerY);
  }
  ctx.closePath();
  ctx.fillStyle = "yellow";
  ctx.fill();
}

// Function to draw multiple stars
function drawStars(numStars) {
  for (let i = 0; i < numStars; i++) {
    const x = Math.random() * width;
    const y = Math.random() * height;
    const radius = Math.random() * 3 + 1; // Random star size
    const points = Math.floor(Math.random() * 10) + 5; // Random number of
points

    drawStar(x, y, radius, points);
  }
}
```

```
// Function to clear the canvas
function clearCanvas() {
  ctx.clearRect(0, 0, width, height);
}

// Function to draw the scene
function drawScene() {
  clearCanvas();
  drawNightSky();
  drawStars(200); // Adjust the number of stars as desired
  drawMoon(width / 3, height / 3, 40); // Place moon in upper left
}

function animate() {
  drawScene();
  requestAnimationFrame(animate);
}
animate();

const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");

const width = canvas.width;
const height = canvas.height;

// Day sky gradient function
function drawDaySky() {
  const gradient = ctx.createLinearGradient(0, 0, width, height);
  gradient.addColorStop(0, "#c3e8f3"); // Light blue at the top
  gradient.addColorStop(0.5, "#d0e6f0"); // Lighter blue in the middle
  gradient.addColorStop(1, "#e0effe"); // Lightest blue at the bottom
  ctx.fillStyle = gradient;
  ctx.fillRect(0, 0, width, height);
}

// Function to draw the sun
function drawSun(x, y, radius) {
  ctx.beginPath();
  ctx.arc(x, y, radius, 0, Math.PI * 2);
  ctx.fillStyle = "yellow";
  ctx.fill();
}

// Function to draw a cloud (optional)
function drawCloud(x, y, size) {
  ctx.beginPath();
  ctx.moveTo(x, y);
```

```
  ctx.bezierCurveTo(
    x + size,
    y - size,
    x + size * 2,
    y,
    x + size * 2.5,
    y + size
  );
  ctx.bezierCurveTo(
    x + size * 3,
    y + size * 2,
    x + size * 2,
    y + size * 3,
    x + size,
    y + size * 2
  );
  ctx.bezierCurveTo(
    x,
    y + size,
    x - size,
    y + size * 2,
    x - size * 1.5,
    y + size
  );
  ctx.closePath();
  ctx.fillStyle = "#fff"; // White for clouds
  ctx.fill();
}

// Function to draw multiple clouds (optional)
function drawClouds(numClouds) {
  for (let i = 0; i < numClouds; i++) {
    const x = Math.random() * width;
    const y = (Math.random() * height) / 3; // Limit clouds to top third
    const size = Math.random() * 30 + 10; // Random cloud size

    drawCloud(x, y, size);
  }
}

// Function to clear the canvas
function clearCanvas() {
  ctx.clearRect(0, 0, width, height);
}

// Function to draw the scene
function drawScene() {
  clearCanvas();
```

```
  drawDaySky();
  drawSun(width / 3, height / 3, 40); // Place sun in upper left
  drawClouds(3); // Adjust the number of clouds (optional)
}

function animate() {
  drawScene();
  requestAnimationFrame(animate);
}
animate();
```

OUTPUT:



CODE FOR Q2:-

Index.html

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

```
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial scale=1" />
    <title>React & Tailwind CSS Starter Pack</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
 This HTML file is a template.
 If you open it directly in the browser, you will see an empty page.
 You can add webfonts, meta tags, or analytics to this file.
 The build step will place the bundled scripts into the <body> tag.
 To begin the development, run `npm start` or `yarn start`.
 To create a production bundle, use `npm run build` or `yarn
build`.
 -->
  </body>
</html>
```

Index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import "./index.css";
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.js: Main-File All React-Code And it’s Components

```
import React, { Component } from "react";
import "./App.css";
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      caloriesConsumed: 0,
      dailyGoal: 2000,
      foodLog: [],
      mealName: "",
      mealCalories: "",
      mealLogged: false,
    };
  }
  handleLogCalories = () => {
    const { mealName, mealCalories } = this.state;
    if (!mealName || !mealCalories) {
      alert("Please enter both meal name and calories!");
      return;
    }
    const calories = parseInt(mealCalories);
    if (isNaN(calories)) {
      alert("Please enter a valid number for calories!");
      return;
    }
    this.setState((prevState) => ({
      caloriesConsumed: prevState.caloriesConsumed + calories,
      foodLog: [
        ...prevState.foodLog,
        { id: Date.now(), name: mealName, calories, timestamp: new Date() },
      ],
      mealName: "",
      mealCalories: "",
      mealLogged: true,
    }));
    setTimeout(() => {
      this.setState({ mealLogged: false });
    }, 3000);
  };
  handleDeleteMeal = (mealId) => {
    this.setState((prevState) => ({
      foodLog: prevState.foodLog.filter((entry) => entry.id !== mealId),
      caloriesConsumed:
        prevState.caloriesConsumed -
        prevState.foodLog.find((entry) => entry.id === mealId).calories,
```

```
    }));
  };
  render() {
    const {
      caloriesConsumed,
      dailyGoal,
      foodLog,
      mealName,
      mealCalories,
      mealLogged,
    } = this.state;
    const progress = (caloriesConsumed / dailyGoal) * 100;
    return (
      <div className="App">
        <h1>Calorie Tracker</h1>
        <div className="goal-form">
          <label htmlFor="daily-goal">Set Daily Calorie Goal:</label>
          <input
            type="number"
            id="daily-goal"
            value={dailyGoal}
            onChange={(e) =>
              this.setState({ dailyGoal: parseInt(e.target.value) })
            }
          />
        </div>
        <p>Today's Calorie Intake: {caloriesConsumed} calories</p>
        <div className="progress-bar">
          <div className="progress" style={{ width: `${progress}%` }}></div>
        </div>
        <div>
          <input
            type="text"
            placeholder="Enter meal name"
            value={mealName}
            onChange={(e) => this.setState({ mealName: e.target.value })}
          />
          <input
            type="text"
            placeholder="Enter calories"
            value={mealCalories}
            onChange={(e) => this.setState({ mealCalories: e.target.value })}
          />
          <br></br>
          <button onClick={this.handleLogCalories}>Log Meal</button>
        </div>
        {mealLogged && (
          <p className="success-message">Meal Logged Successfully!</p>
```

```
        )}
        <FoodLog foodLog={foodLog} onDeleteMeal={this.handleDeleteMeal} />
      </div>
    );
  }
}
const FoodLog = ({ foodLog, onDeleteMeal }) => (
  <div>
    <h2>Food Log</h2>
    <ul>
      {foodLog.map((entry, index) => (
        <li key={entry.id}>
          {entry.timestamp.toISOString()} - {entry.name}: {entry.calories}
          calories
          <button onClick={() => onDeleteMeal(entry.id)}>Delete</button>
        </li>
      ))}
    </ul>
  </div>
);
export default App;
```

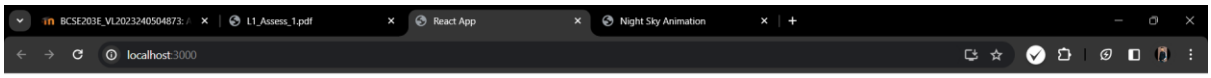
App.css: Used To Style React Components

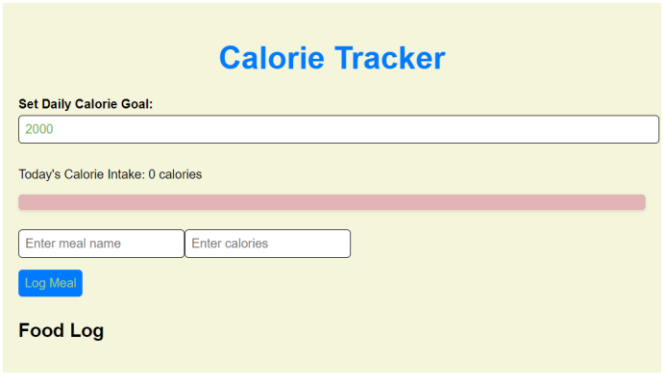
```
.App {
  max-width: 800px;
  margin: 0 auto;
  margin-top: 100px;
  padding: 20px;
  font-family: Arial, sans-serif;
  background-color: beige;
}
h1 {
  text-align: center;
  font-size: 40px;
  color: #007bff;
  font-weight: bolder;
}
.goal-form {
  margin-bottom: 20px;
}
.goal-form label {
  font-weight: bold;
}
.goal-form input {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
```

```
  margin-bottom: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 5px;
  color: #6bb543;
}
.progress-bar {
  background-color: #e1b5b5;
  height: 20px;
  border-radius: 5px;
  overflow: hidden;
  margin-bottom: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
.progress {
  background-color: #97cc7b;
  height: 100%;
  transition: width 0.3s ease-in-out;
}
input[type="text"],
input[type="number"],
button {
  margin-top: 5px;
  margin-bottom: 10px;
  padding: 8px;
  font-size: 16px;
  border: 1px solid black;
  border-radius: 5px;
  transition: border-color 0.3s ease-in-out;
}
button {
  background-color: #007bff;
  color: #a0c97d;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out;
}
button:hover {
  background-color: #0056b3;
}
.success-message {
  color: #6bb543;
  text-align: center;
  font-weight: bold;
}
.food-log {
  background-color: #ffffff;
```

```
  padding: 15px;
  border-radius: 10px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
.food-log ul {
  list-style: none;
  padding: 0;
}
.food-log li {
  border-bottom: 1px solid #ddd;
  padding: 10px 0;
  display: flex;
  justify-content: space-between;
}
.food-log li:last-child {
  border-bottom: none;
}
.food-log button {
  background-color: #dc3545;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out;
}
.food-log button:hover {
  background-color: #c82333;
}
```

OUTPUT:





React App Install