Task No: 3 (Polling System Smart Contract) CodeAlpha internship {Blockchain)

I'll help create a personal portfolio that includes the polling system smart contract project as one of the showcased projects. Below is a complete solution with HTML, CSS, JavaScript, and the Solidity smart contract code.

Personal Portfolio

HTML (index.html)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Portfolio</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <!-- Navigation -->
    <nav>
        <ul>
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#projects">Projects</a></li>
            <li><a href="#resume">Resume</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>

    <!-- Home Section -->
    <section id="home">
        <h1>Welcome to My Portfolio</h1>
        <p>Hi, I'm [Your Name], a passionate developer skilled in web development and blockchain technology.</p>
        <button id="theme-toggle">Toggle Theme</button>
    </section>

    <!-- About Section -->
    <section id="about">
        <h2>About Me</h2>
        <p>I specialize in creating dynamic web applications and smart contracts. My expertise includes HTML, CSS, JavaScript, and Solidity.</
    </section>

    <!-- Projects Section -->
    <section id="projects">
        <h2>Projects</h2>
        <div class="project">
            <h3>Polling System Smart Contract</h3>
            <p>A decentralized polling system built with Solidity, allowing users to create polls, vote, and view results with time-based rest
            <button onclick="showCode()">View Code</button>
            <pre id="code-snippet" style="display: none;">
pragma solidity ^0.8.0;

contract PollingSystem {
    struct Poll {
        string question;
        string[] options;
        mapping(uint => uint) votes; // Option index => vote count
        mapping(address => bool) hasVoted;
        uint endTime;
        address creator;
        bool isActive;
    }

    mapping(uint => Poll) public polls;
    uint public pollCount;

    event PollCreated(uint pollId, string question, address creator);
    event Voted(uint pollId, uint option, address voter);
    event PollEnded(uint pollId, uint winningOption);

    modifier onlyCreator(uint _pollId) {
        require(msg.sender == polls[_pollId].creator, "Not the creator");
        _;
    }

    function createPoll(string memory _question, string[] memory _options, uint _duration) public {
        require(_options.length >= 2, "Need at least 2 options");
        Poll storage newPoll = polls[pollCount];
        newPoll.question = _question;
        newPoll.options = _options;
        newPoll.endTime = block.timestamp + _duration;
        newPoll.creator = msg.sender;
        newPoll.isActive = true;

        emit PollCreated(pollCount, _question, msg.sender);
        pollCount++;
    }

    function vote(uint _pollId, uint _option) public {
        Poll storage poll = polls[_pollId];
        require(poll.isActive, "Poll is not active");
        require(block.timestamp <= poll.endTime, "Poll has ended");
        require(_option < poll.options.length, "Invalid option");
        require(!poll.hasVoted[msg.sender], "Already voted");

        poll.votes[_option]++;
        poll.hasVoted[msg.sender] = true;

        emit Voted(_pollId, _option, msg.sender);
    }

    function endPoll(uint _pollId) public onlyCreator(_pollId) {
        Poll storage poll = polls[_pollId];
        require(poll.isActive, "Poll already ended");
        require(block.timestamp > poll.endTime, "Poll duration not over");

        poll.isActive = false;
        uint winningOption = getWinningOption(_pollId);

        emit PollEnded(_pollId, winningOption);
    }

    function getWinningOption(uint _pollId) public view returns (uint) {
        Poll storage poll = polls[_pollId];
        uint maxVotes = 0;
        uint winningOption = 0;

        for (uint i = 0; i < poll.options.length; i++) {
            if (poll.votes[i] > maxVotes) {
                maxVotes = poll.votes[i];
                winningOption = i;
            }
        }
        return winningOption;
    }

    function getPollDetails(uint _pollId) public view returns (string memory, string[] memory, uint, bool) {
        Poll storage poll = polls[_pollId];
        return (poll.question, poll.options, poll.endTime, poll.isActive);
    }
}
            </pre>
        </div>
        <div class="project">
            <h3>Personal Portfolio</h3>
            <p>A responsive portfolio website built with HTML, CSS, and JavaScript.</p>
        </div>
    </section>

    <!-- Resume Section -->
    <section id="resume">
        <h2>Resume</h2>
        <p><strong>Education:</strong> [Your Degree, University, Year]</p>
        <p><strong>Skills:</strong> HTML, CSS, JavaScript, Solidity, Web3</p>
        <p><strong>Experience:</strong> [Your Experience Details]</p>
        <a href="resume.pdf" download>Download Resume</a>
    </section>

    <!-- Contact Section -->
    <section id="contact">
        <h2>Contact</h2>
        <p>Email: [your.email@example.com]</p>
        <p>LinkedIn: [Your LinkedIn Profile]</p>
        <p>GitHub: [Your GitHub Profile]</p>
    </section>
    <script src="script.js"></script>
</body>
</html>
```

CSS (styles.css)

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

body {
    background-color: #f4f4f4;
    color: #333;
    transition: background-color 0.3s, color 0.3s;
}

body.dark {
    background-color: #222;
    color: #fff;
}

nav {
    background-color: #007BFF;
    padding: 1rem;
    position: fixed;
    width: 100%;
    top: 0;
}

nav ul {
    list-style: none;
    display: flex;
    justify-content: center;
}

nav ul li {
    margin: 0 1.5rem;
}

nav ul li a {
    color: white;
    text-decoration: none;
    font-weight: bold;
}

nav ul li a:hover {
    color: #ddd;
}

section {
    padding: 4rem 2rem;
    margin-top: 60px;
    text-align: center;
}

h1, h2 {
    margin-bottom: 1rem;
}

.project {
    background-color: #fff;
    padding: 1.5rem;
    margin: 1rem auto;
    max-width: 600px;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.dark .project {
    background-color: #333;
}

button {
    padding: 0.5rem 1rem;
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

pre {
    background-color: #f8f8f8;
    padding: 1rem;
    border-radius: 5px;
    overflow-x: auto;
    text-align: left;
    margin-top: 1rem;
}

.dark pre {
    background-color: #444;
}

a {
    color: #007BFF;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}
```

JavaScript (script.js)

```javascript
// Theme Toggle
const themeToggle = document.getElementById('theme-toggle');
themeToggle.addEventListener('click', () => {
    document.body.classList.toggle('dark');
    themeToggle.textContent = document.body.classList.contains('dark') ? 'Light Theme' : 'Dark Theme';
});

// Show/Hide Code Snippet
function showCode() {
    const codeSnippet = document.getElementById('code-snippet');
    codeSnippet.style.display = codeSnippet.style.display === 'none' ? 'block' : 'none';
}
```

Polling System Smart Contract Explanation

The Solidity smart contract (`PollingSystem`) included in the portfolio allows:

- **Poll Creation**: Users can create polls with a question, multiple options, and a duration (`createPoll`).
- **Voting**: Users can vote once per poll, with checks to prevent double-voting (`vote`).
- **Time Restrictions**: Polls end after the specified duration, enforced by `block.timestamp` (`endPoll`).
- **Vote Storage**: Uses a `mapping` to track votes per option and voter status.
- **Winner Determination**: Automatically calculates the winning option (`getWinningOption`).
- **Access Control**: Only the poll creator can end a poll (`onlyCreator` modifier).
- **Events**: Emits events for poll creation, voting, and poll ending for transparency.

Key features:

- **Data Structures**: Uses `struct` for polls, `mapping` for votes and voter tracking, and arrays for options.
- **Security**: Includes checks for valid options, active polls, and voting eligibility.
- **Time Functions**: Leverages `block.timestamp` for time-based restrictions.

How to Use

1. **Portfolio**:
   - Save the HTML, CSS, and JavaScript files in a directory.
   - Replace placeholders (e.g., `[Your Name]`, `[your.email@example.com]`) with your details.
   - Add a `resume.pdf` file in the same directory for the resume download link.
   - Open `index.html` in a browser to view the portfolio.
   - The portfolio is responsive, includes a dark/light theme toggle, and shows the Solidity code for the polling system when the "View Code" button is clicked.
2. **Smart Contract**:
   - Deploy the Solidity contract on an Ethereum-compatible blockchain (e.g., using Remix IDE).
   - Interact with the contract to create polls, vote, and end polls.
   - The contract code is displayed in the portfolio for reference.

This solution provides a clean, interactive portfolio showcasing your skills and the polling system project, with a focus on simplicity and functionality. Let me know if you need further customization!