**Enhancing Your DApp: Minting, Approvals, and User Authentication**

**Objective:**

- Build upon your basic ERC-20 token DApp by adding more advanced features.
- Implement token minting and approval mechanisms.
- Introduce user authentication using MetaMask integration.

**Prerequisites:**

- Completion of previous assignments, specifically Assignment 4: ERC-20 Token Creation and Interaction.
- Familiarity with basic HTML, CSS, and JavaScript.
- Basic understanding of a frontend framework (e.g., React, Vue.js, or Svelte) is recommended.

**Instructions:**

1. **Enhance the User Interface:**
    - Add new elements to your existing DApp frontend:
        - A form to input the amount of new tokens to mint.
        - A button to trigger the minting process.
        - A form to input the spender address and the amount of tokens to approve.
        - A button to trigger the approval process.
        - A display to show the user's connected MetaMask address (after authentication).
2. **Implement Minting:**
    - Add a new function in your JavaScript code (or framework component) to handle the minting process.
    - Use the `mint` function from your ERC-20 contract to create new tokens and assign them to the connected user's address.
    - Update the token balance display after a successful mint.
3. **Implement Approvals:**
    - Add a new function to handle the approval process.
    - Use the `approve` function from your ERC-20 contract to allow a specified address (spender) to spend a certain amount of the user's tokens.
    - Provide feedback to the user about the approval status.
4. **User Authentication (MetaMask Integration):**
    - When the DApp loads, check if MetaMask is installed and prompt the user to connect their wallet if it's not already connected.
    - Once connected, display the user's MetaMask address on the interface.
    - Restrict access to minting and approval functions to the owner of the contract (the address that deployed it).

    - You can use MetaMask's `ethereum.request({ method: 'eth_requestAccounts' })` to request the user's accounts and verify ownership.

**Deliverables:**

- Submit your updated HTML, CSS, and JavaScript code files.
- Update your README file with instructions on how to run the enhanced DApp.
- Provide screenshots or a video demonstrating the new features (minting, approvals, authentication).
- Answer the following questions in your report:
    - What security considerations did you take into account when implementing minting and approvals?
    - How did you handle user authentication and restrict access to certain functions?
    - What are some potential use cases for the approval mechanism in real-world applications?

**Grading Rubric:**

- **Functionality:** Do the minting and approval functions work correctly? Is user authentication implemented effectively?
- **User Interface:** Is the interface clear, intuitive, and visually appealing, incorporating the new features seamlessly?
- **Code Quality:** Is the code well-structured, readable, and commented, reflecting good programming practices?
- **Error Handling:** Does the DApp handle potential errors gracefully and provide informative feedback to the user?
- **Security:** Are there any vulnerabilities in the minting or approval processes? Is user authentication robust?
- **Creativity and Innovation:** Did the student add any additional features or creative elements to the DApp?