Problem Description:

You are tasked with creating a **Multi-Utility NFT Contract** that has the following functionalities:

---

1. Phased Minting with Merkle Proofs

- Implement a phased minting process:
    - **Phase 1:** Whitelisted users (verified with a Merkle root) can mint for free.
    - **Phase 2:** Selected users (verified via a new Merkle root) can mint at a discounted price.
    - **Phase 3:** Open minting for everyone at the full price.
- Create an ERC20 and use it as the payment token.

---

2. Discount Verification with Signatures

- Users in Phase 2 (who qualify for discounts) must provide a valid signature from the contract owner to mint with the discounted price.

---

3. Vesting Integration

- After minting is done, lock the minting fee in a linear vesting schedule for one year on Sablier and allow only the owner to claim the vested tokens.

Security Requirements:

- Ensure the contract is protected from common and novel security flaws like reentrancy, missing input validation, signature malleability, reusable signatures etc.

Events:

- Emit events for minting.

Deliverables:

1. **Smart Contract Implementation**
    - Write the Solidity contract using best practices (version >=0.8.0).
    - Use OpenZeppelin libraries for ERC721, ERC20, and Merkle proof verification.

2. **Foundry Test Suite**
   Write tests to cover:
        - Phased minting with Merkle proofs and signature validation.
        - Vesting functionality and withdrawal logic.
        - Edge cases like invalid Merkle proofs, invalid signatures

   You can employ the Branching Tree Technique (BTT) to improve test coverage.

3. **Readme File**
    - Briefly explain the contract design, features, and testing approach.

Constraints:

- The contract must compile successfully using `solc` >=0.8.0.
- The test suite must achieve 80%+ coverage using Foundry.

Example Input and Output:

Input (Phase 1 Minting):

```
mint(phase1MerkleProof);
```

Expected Output:

- User mints an NFT for free if the proof is valid.

Input (Phase 2 Minting):

```
mintWithDiscount(signature, phase2MerkleProof);
```

Expected Output:

- User mints an NFT at a discounted price if the signature and proof are valid.

Submission:

Once completed, reply to this email with a GitHub repository link and other details.