NovaMint Document

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract Beachsunset is ERC721, ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIdCounter;
    string public constant collectionName = "Beach sunset";
      // Description: sunset at beach\n     string public constant defaultAssetName =
"mother.webp";
    uint256 public constant creationTimestamp = 1749639046;

    constructor(address initialOwner)
        ERC721("Beach sunset", "BEAC") Ownable(initialOwner) {}

    function mintNFT(address recipient, string memory tokenURI) public onlyOwner returns
(uint256) {
        _tokenIdCounter.increment();
        uint256 newItemId = _tokenIdCounter.current();
        _safeMint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);
        return newItemId;
    }
        function _update(address to, uint256 tokenId, address auth) internal
override(ERC721, ERC721URIStorage) returns (address) { return super._update(to, tokenId,
auth); }
    function _increaseBalance(address account, uint128 amount) internal override(ERC721,
ERC721URIStorage) { super._increaseBalance(account, amount); }
     function tokenURI(uint256 tokenId) public view override(ERC721, ERC721URIStorage)
returns (string memory) { return super.tokenURI(tokenId); }
        function supportsInterface(bytes4 interfaceId) public view override(ERC721,
ERC721URIStorage) returns (bool) { return super.supportsInterface(interfaceId); }
       function _burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {
super._burn(tokenId); }
}
```