Solidity developer tech assignment

Problem:
Your task is to create a staking contract.
Users stake different quantities of ERC-20 token named “TKN”. Assume that an external caller would periodically transfer reward TKNs to a staking smart contract (no need to implement this logic). Rewards are proportionally distributed based on staked TKN.
Contract caller can:
- Stake
- **Unstake (would be a plus if caller can unstake part of stake)**
- See how many tokens each user can unstake

Main info:
- Use Solidity as a main language (version higher than 0.4.0)
- Cover contract with unit tests. You can use Truffle or Hardhat

My Word:

I think this assignment is really worth for getting notice the ability of solidity developer.
To be honest, I have read many similar projects and papers to find the best solution to be passes in this step.
Actually, during that period, I noticed that the essay ***Scalable Reward Distribution on the Ethereum Blockchain*** by Batog et al is not suit one to meet all requirement from this assignment.

- Main issue is that

The paper by Bogdan et al. assumes that stake size for each user doesn’t change once it is set(stakeholder deposit tokens or ether into staking pool). According to this assumption, stakeholder should wait to collect rewards and then can unstake his assets including rewards from staking pool. However, it’s model can not accepted in real world, actually.
That’s why there’s requirement to change staked size anytime and unstake some of them. Right?
Yes, I strongly agree with this requirement and I reached out the fact that I have to modify the algorithm in this essay a little bit.
That’s why I named staking contract as **StakeV2**

- What I did

The total amount of reward of stakeholder j is

$$total\_reward_j = \sum_t reward_{j,t} = \sum_t stake_{j,t} \frac{reward_t}{T_t}$$

This can be rewritten like following:

$$total\_reward_j = stake_{j,n} \sum_{t=0}^{n} \frac{reward_t}{T_t} - \sum_{t=1}^{n} ((stake_{j,t} - stake_{j,t-1}) \sum_{t=0}^{t-1} \frac{reward_t}{T_t})$$

※ according to this equation:

$$\sum_{i=0}^{n} a_i b_i = a_n \sum_{j=0}^{n} b_j - \sum_{i=1}^{n} ((a_i - a_{i-1}) \sum_{j=0}^{i-1} b_j)$$

I proved this by myself and if you mind, it’s okay to skip following part

$$b_i = \sum_{j=0}^{i} b_j - \sum_{j=0}^{i-1} b_j$$

Substitue b_i on the LHS of above equation

$$\sum_{i=0}^{n} a_i b_i = \sum_{i=0}^{n} a_i\left(\sum_{j=0}^{i} b_j - \sum_{j=0}^{i-1} b_j\right) = \sum_{i=0}^{n}\left(a_i\sum_{j=0}^{i} b_j - a_i\sum_{j=0}^{i-1} b_j\right)$$

$$=> \ \sum_{i=0}^{n}\left(a_i\sum_{j=0}^{i} b_j\right) \ - \ \sum_{i=0}^{n}\left(a_i\sum_{j=0}^{i-1} b_j\right) \qquad *$$

Here:

$$\sum_{i=0}^{n}\left(a_i\sum_{j=0}^{i-1} b_j\right) = \sum_{i=1}^{n}\left(a_i\sum_{j=0}^{i-1} b_j\right)$$

because when $i = 0$, $\boldsymbol{a_i\sum_{j=0}^{i-1} b_j}$ doesn't have value.

Also according to

$$\sum_{i=0}^{n} a_i = a_n + \sum_{i=1}^{n} a_{i-1}$$

The **equation *** should be

$$a_n\sum_{j=0}^{n} b_j \ + \ \sum_{i=1}^{n}\left(a_{i-1}\sum_{j=0}^{i-1} b_j\right) - \sum_{i=1}^{n}\left(a_i\sum_{j=0}^{i-1} b_j\right)$$

$$=> \ a_n\sum_{j=0}^{n} b_j \ - \ \sum_{i=1}^{n}\left(\sum_{j=0}^{i-1} b_j\,(a_i - a_{i-1})\right)$$

Or

$$\boldsymbol{\sum_{i=0}^{n} a_i b_i = a_n\sum_{j=0}^{n} b_j - \sum_{i=1}^{n}\left((a_i - a_{i-1})\sum_{j=0}^{i-1} b_j\right)}$$

Then, let's define

$$\text{reward\_per\_token}_{\text{t}} = \sum_{k=0}^{t}\frac{reward_k}{T_k}$$

$$\triangle\ \text{stake}_{\text{j, t}} = \text{stake}_{\text{j, t}} - \text{stake}_{\text{j, t}-1}$$

We can rewrite

$$total\_reward_j = \text{stake}_{\text{j, n}} * \text{reward\_per\_token}_{\text{n}} - \sum_{\text{t}=1}^{\text{n}}\left(\triangle\ \text{stake}_{\text{j, t}} * \text{reward\_per\_token}_{\text{t}-1}\right)$$

If we define

$$\text{reward\_tally}_{\text{j,n}} = \sum_{\text{t}=1}^{\text{n}}\left(\triangle\ \text{stake}_{\text{j, t}} * \text{reward\_per\_token}_{\text{t}-1}\right)$$

Then we get

$$\boxed{total\_reward_j = \mathbf{stake_{j,\,n}} * \mathbf{reward\_per\_token_n} - \mathbf{reward\_tally_{j,n}}}$$

That’s what I modified to implement the requirement from this assignment.

In my contract, there are 3 main variables (stake, reward_per_token, reward_tally) and 3 main functions(deposit, distribute, withdraw).

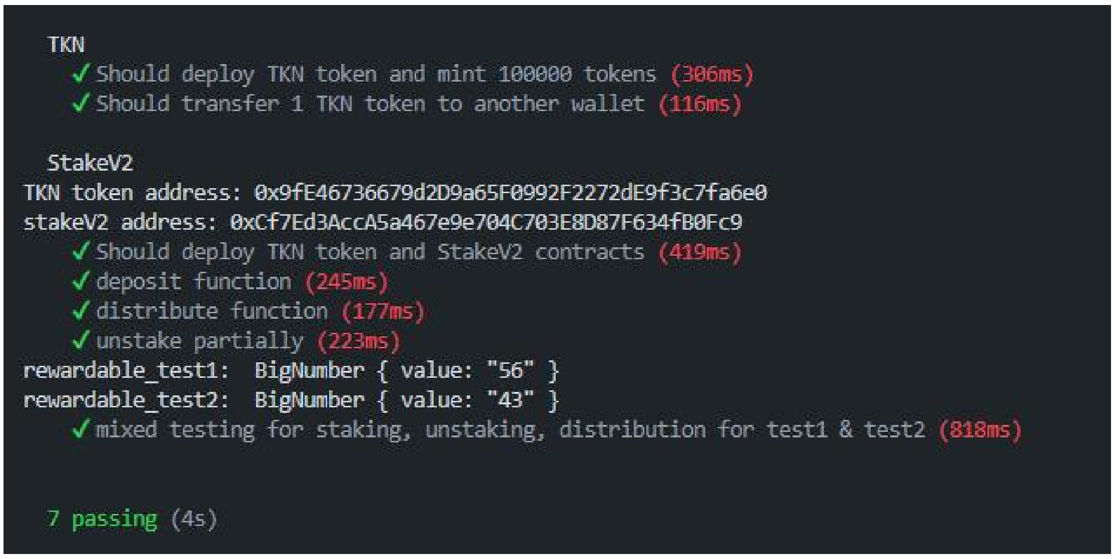
Modification actions should be taken like following:

Variable	Function
stake	deposit, withdraw_stake
reward_per_token	distribute
reword_tally	deposit, withdraw

-Unit-Testing

1. ERC20 Token Test
   Token delopy and transferring operation are tested
2. Staking Operation Test

- Tested Staking & ERC20Token contracts deploying and transfer functionality of TKN tokens in Staking contract
- Deposit functionality for test1 account.
- Distribution functionality for test1 account.
- Unstaking(partially) functionality for test1 account.(also should works fine for complete unstaking)
- Mixed Testing with test1 & test2 account for all kinds of functionality above(including showing unstakable assets)



```
  TKN
    ✓ Should deploy TKN token and mint 100000 tokens (306ms)
    ✓ Should transfer 1 TKN token to another wallet (116ms)

  StakeV2
TKN token address: 0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
stakeV2 address: 0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9
    ✓ Should deploy TKN token and StakeV2 contracts (419ms)
    ✓ deposit function (245ms)
    ✓ distribute function (177ms)
    ✓ unstake partially (223ms)
rewardable_test1:  BigNumber { value: "56" }
rewardable_test2:  BigNumber { value: "43" }
    ✓ mixed testing for staking, unstaking, distribution for test1 & test2 (818ms)


  7 passing (4s)
```

Thank you very much