**PROJECT PROPOSAL:**

Hi @arslyte

So here's a breakdown of the project PREX.finance. We are still contemplating whether to proceed with the execution and would first like to determine whether our tech resources are sufficient to pull this project to completion, would like to hear your thoughts and capacity

There are **2** main components to PREX

1. Farm component

A clone of a PCS-like farm, as close to https://fullsail.finance. The idea is that users will stake their tokens or LPs to earn PREX tokens. These tokens could be either swapped on PCS for stablecoins or they could be used to make predictions/place bets on the second component of the project

2. Prediction Exchange component

On this component, users make a prediction on pricing of various crypto coins. Let's take BTC as an example, but this will apply for any coin we decide to make markets on.

Each market will take place over 2 hours. For example, we take the market that takes place from 2-4pm. The whole flow will happen as follows: At 2 pm, a snapshot of BTC price is taken, say $60,000. So between 2-3pm, users will make a prediction about whether the price at 4pm will be higher or lower than $60,000. At 3pm, no more bets will be accepted. At 4pm, when price is revealed, it will be determined whether the winners are those who bet above, or those that bet below.

And here is how the winnings work. Let me explain first the concept of Bonus Pot. Every day, the Farm component of PREX will generate a certain amount fees from user staking that goes to a Bonus Pot. For instance, yesterday a total of 2400 PREX tokens were generated for the Bonus Pot. These 2400 PREX will be distributed evenly across the Bonus Pots for EACH of the 24 daily markets. In this case, 100 PREX will be in the Bonus Pot for each market. 80% of the bonus pot will go to the winners. 10% of the bonus will go to the losers. 10% will go to PREX Reserves Fund for buybacks and future developments.

Going back to our example of the market created for 2-4pm. Assuming 100 PREX in the Bonus Pot. Assume 10 people bet 100 PREX each that the price of BTC at 4pm will be above $60,000. Only 1 person bet 100 PREX that the price of BTC will be below.

At 4pm, if price is above, the outcome will be as follows:

Each of the 10 winners will take 10% of the Bonus Pot of 80 PREX in addition to 10% of the 100 PREX from the loser, so gross winnings of each winner will be 100 (original bet) + 8 (share of winner's pot) + 10 (share of loser losses) = 118. PREX.finance will take a cut of 5% from from the winnings.

The loser in this scenario, loses his 100 PREX bet but being the only person in the pool, receives 100% of the loser pot of 10. So even though he lost, he still retrieved 10 as he was the only person in the pool, so he only effectively lost 90.

Exact mechanism and percentages might have to be tweaked here and there, but that's the general idea, hope that is clear enough.

The Farm component I believe can be easily cloned from PCS/Fullsail, and should not take too much time.

The Prediction Exchange should be the time-consuming one.

That said, can you please study the project carefully and provide me with a breakdown of the timeline and the difficulty level for a project like this?

Of course feel free to let me know if you have any questions or suggestions to make this better.

Here is a sneak peak at the UI which is not finalized yet: https://xd.adobe.com/view/8ebd0b24-2318-494e-a1a0-270403d1d761-dbee/?fullscreen

Awaiting your comments, please take your time to read through and digest.

Thank you.

**REQUIREMENTS GATHERING**

* A ERC20 token, PREX
* A PancakeSwap-like Farm, for yielding PREX
* A betting platform, lottery for PREX
* The bet is whether a specific coin will be higher or lower than a price at a specific point in time
* The bet is revealed and distributed 2 hours after that specific point in time
* The wagerable duration is 1 hour long, afterwards no more bets can be placed in this round
* 24 betting rounds per day, guaranteeing 1 is running 24/7.

**DRAFT OF CONTRACT(S)**

PERSISTENCE = 30

markets: Market[]

* Market:
  1. name: string // name of token
  2. tokenAddr: address
  3. token: ERC20
  4. pots: Pot[PERSISTENCE] // PERSISTENCE hours of pot data persistence,

// refresh is required before exhaustion of persistence time

* 1. openPotId: uint8 // index of the open Pot

// i.e. the pot which players are currently betting into

* 1. lastRefreshedTime: uint
* Pot:
  1. marketId: uint // index of the Market which this Pot is a part of
  2. locked: Boolean // indicates whether Pot funds were distributed; yes: false, no: true
  3. closedAfter: uint // timestamp, indicates whether Pot is bet-able, now<closedAfter
  4. sealedAfter: uint // timestamp, indicates whether Pot is sealed, now>=sealedAfter
  5. lower: map(address, Prediction)
  6. higher: map(address, Prediction)
  7. lowerPool: uint
  8. higherPool: uint
  9. bonusPool: uint
* Prediction:
  1. predictor: address
  2. wager: uint
* predict:
  + 1. New Pot
  + now>=closedAfter:
    - pots[openPotId].locked=true
    - require(pots[openPotId+1 OR 0].locked==false)
    - openPotId<PERSISTENCE: openPotId+=1
    - openPotId>=PERSISTENCE: openPotId=0
    - pots[openPotId].closedAfter=previousClosedAfter+(now-previousClosedAfter)/hours + 1
    - pots[openPotId].sealedAfter=pots[openPotId].closedAfter + 1
  + 2. Existing Pot
* refresh:
  + for market in markets:
    - for pot in market.pots:
      * if pot.locked:
        + distribute pot
        + zero out
        + unlock

**ALTERNATIVELY**

* Pot
  + open: uint // valuation
  + close: uint // valuation
  + lower: map(address, prediction)
  + higher: map(address, prediction)
  + lowerPool: uint
  + higherPool: uint
  + bonusPool: uint
  + wagerableFrom: timestamp
  + wagerableTill: timestamp
  + claimableFrom: timestamp
* Prediction
  + predictor: address
  + wager: uint
  + lastModifiedTime: uint
* spawnPot(wagerableFrom = now, wagerDuration, idleDuration)
  + Add new Pot
  + Initialize all parameters except ‘close’
    - How to query valuation ‘open’?
* dropPot
  + claimPot
  + Delete Pot
* claimPot(id)
  + Check all conditions
  + Allocate funds
  + Lock the pot
* resetPot():
  + Reset all parameters given \_wagerableFrom, …
  + Initialize all parameters except ‘close’
  + How to query valuation ‘open’?
* predict
  + Check all conditions
  + Cast a wager into a Pot
* distribute
  + Allocates bonuses to each pot

Notes.  
wagerableFrom = \_wagerableFrom  
wagerableTill = wagerableFrom + \_wagerDuration  
claimableFrom = wagerableTill + \_idleDuration  
Ex. Assume Higher Won  
fees = lowerPool \* 5 / 100 // FEES\_RATE = 5  
prizePool = lowerPool – fees

**Winners:**winnerBonusCut = bonusPool \* 80 / 100(prizePool + winnerBonusCut) \* prediction.wager / higherPool

**Losers:**   
loserBonusCut = bonusPool \* 10 / 100  
If loserBonusCut > lowerPool:   
 Gain: (loserBonusCut – lowerPool) \* prediction.wager / lowerPool   
Else:  
 Loss: (lowerPool – loserBonusCut) \* prediction.wager / lowerPool