Data Model

```
Participant
- address string (PK)
- team int (FK)
```

```
Team
- id (PK)
- participants []string
- participant_shares []int
- balance int
- withdraw_votes int
```

Backend

smart contracts

```
Payment.sol
- release
```

```
TeamBounties.sol
- createTeam
- payTeam
- voteWithdraw
- withdraw
- addParticipant
- proposeParticipant
```

```
Contract deployed at
address xxxx by core
dev team
```

postgres table(s)

```
Participants
- address string (PK)
- belongs_in []int
- total int
```

```
Teams
- address string (PK)
- name string
- addresses
- total int
```

python API

```
insert/update (address,
team_id, total)
```

Note that the RDB table is completely optional.
the real source of truth will be held in the smart contract.
But and RDB means lower latency and better user experience for users who are browsing teams.

UI (scenarios)

Create a team:
1. User connects metamask account, fills in form (addresses, shares, team name).
2. Check the form is correct
3. Using the connected account, deploy Payments.sol and record the new address of the smart contract.
4. Call createTeam function on the already deployed TeamBounties. contract with the relevant arguments.
5. Call API to insert new team and new participants!

Connected accounts pays the gas fees.

Pay a team:
1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and transfers funds from his/her wallet to TeamBounties contract (using payTeam function)

Vote to withdraw:

Once a team has been paid, the balance of TeamBounties contract has increased, but participants have not been paid. To receive payment, participants must vote. A participants vote is as powerful as its stake. Once enough participants have voted to unlock balance that we have >50% majority, the balance can be paid to the Payments contract, where the participants will be able to cash out.

1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and clicks on vote button

Withdraw:
1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and clicks on Withdraw button

Release:
1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and clicks on Release button
3. Call API and update participants and teams tables!

Add Participant:
1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and clicks on add participant (providing an address). The user that is adding the participant needs to be part of the team. If that user has >50% of stake, the new participant will be automatically added. Else other participants from the team need to vote for new participant to officially join
3. Call API and update participants and teams tables!

A single address can only be in the process of joining a team one team at a time!

Vote for new participant:
1. Scroll through the list of teams, or search your team by name or address.
2. Once team is found, user connects metamask wallet and scans the list of pending participants (if any). For the ones they approve, click on vote. Once a participant has joined a team they are able to vote for every open candidature.

3. Call API and update participants and teams tables!