IS4302 Blockchain and Distributed Ledger Technologies

AY 2023/2024 Semester 2
Group 4
Project Title: bETH

Architecture and Design Document

Name	Matriculation Number
Tan Jing Wen Sarah	AXXXXXXX
Pang Yong Jie	AXXXXXXX
Tyler Wang Yao	AXXXXXXX
Tang Wei Shawn	AXXXXXXX

Link to Code: https://github.com/sarahjingwen/bETH/

Table of Contents

1. Introduction

bETH is a decentralised betting protocol and platform built on the Ethereum blockchain, utilising smart contracts to offer a secure and transparent betting experience. bETH allows users to place bets via transferring Ether onto the bETH smart contract representing, and to be paid out a portion of the total prize pool if they win.

bETH is paired with an end-user website to interact with the blockchain via UI. Users may place bets but also initiate their own bets via the website.

2. Features

2.1 Dynamic Odds

2.1.1 Overview

In traditional betting, the odds are determined at the point of the bet placement, and the agreed upon odds will be used to finalise the payout. In our dynamic odds system however, there will be no guarantee of the odds until the betting time is over.

Some features of this include:

- **Lower fees** and a larger part of the prize pool to be allocated to prize winners.
    - Traditional betting providers have large bet spreads to prevent losses in the event that the odds fluctuate unfavourably, or in case a large bet were to be placed.
    - The large bet spread means that the bettors are not getting a fair value for their bets.
    - Dynamic odds poses zero risk to betting providers in terms of loss of capital.
- **Transparency in prize pool** computation.
    - Blockchain transactions are transparent and verifiable.
    - Total prize pool is computed based on the smart contract code and not manually accessible.
    - Transparent fees collection and gas payments built into the code.
- **Hassle-free peer-to-peer betting**
    - Complexities of handling all transactions are encapsulated at the smart contract level, and end users who wish to initiate bets can simply use UI.

2.1.2 How Our Design Enable this Feature

Dynamic odds is based on the ratio of two betting sides (for / against) which changes until the bets are closed. All payouts are allocated from the initial prize pool from all bettors. Since the

betting provider does not require any capital input, there is no payment required from the betting provider, reducing any risk of loss of capital for the betting providers.

Through a view function, people can view the current odds and decide if they would like to place their bets. All fees are also made transparent and are fixed for the bettors to view. When a bet concludes, anyone can trace the funds in full transparency.

2.2 High Revenue for Developer: Commission Fees

2.2.1 Overview

The main source of revenue for the developers and bet creators will be the commission fees involved in each bet. The fees will be structured as such:

- Fixed transaction fees associated with each bet that is linked to gas fees.
    - Main purpose is to pay for gas fees to prevent a situation where paying out a bettor will cause the contract to have a net loss if the gas fee is higher than the transaction fee.
    - Betting creators cannot change this value as it is fixed upon creation of the smart contract.
    - Transaction Fee is fixed per bet placed and applies only to winners, regardless of bet value as the purpose of the transaction fee is to cover the payout gas fee.
- Commission fee of 2% of total prize pool
    - Rake is computed before prize pool is distributed to the winning bettors
    - 1% commission fees goes to bet initiator
    - 1% commission fees goes to the platform creators (us)

2.2.2 How Our Design Enable this Feature

The prize pool is obtained from all the Ether received. We will then calculate the estimated transaction fee through Solidity. Currently, we check the current gas fees price multiplied by the estimated gas limit of 30 million, this is then further multiplied by the transaction fee, if the developers want to charge more than the supposed transaction fee as a buffer. From this prize pool, 1% will go to the platform creators, another 1% will go to the bet initiator, and the remaining prize pool will be split amongst the winners based on the percentage of their bet.

2.3 Social Peer-to-Peer Bets Handling

2.3.1 Overview

bETH will allow users to have public bets where anyone can participate and private bets where a closed group of people will be allowed to participate that is determined by the bet initiator:

- Bets can be open to public for a social betting
- Friends can bet on outcomes of events using the platform
- The Website can facilitate the social aspect of betting, by implementing features such as user public profile, bet creation history, browse available bets, etc.

2.3.2 How Our Design Enable this Feature

All bets have an integer variable linked to Group_ID. There is a createGroup function where anyone can create a group consisting of a list of addresses, where every group has an ID.

To create a public bet, the bet initiator can leave the Group_ID to be 0. Otherwise, the bet initiator creates the bet linked to a Group_ID. Whenever someone tries to participate in a bet and the Group_ID is not 0, a lookup will be performed to check if the address is approved by checking if that address is located in the group based on the Group_ID.

2.4 Betting on Non-traditional Betting Events

2.4.1 Overview

Traditional online betting providers rarely provide betting services for entertaining events or meme bets, but there is a demand for this.

- COVID-19 daily cases betting
- Epstein list betting going viral

2.4.2 How Our Design Enable this Feature

Since bets can be created by anyone, the community can come up with eccentric betting events and bETH will be able to take care of the logistics and distribution of funds safely. For a bet to be confirmed, an administrator will select the result of the bet. After 24 hours, the bet winnings will be distributed. This delay is to prevent any human errors on the administrator's end, where it can be changed if needed during the 24 hour delay period.

2.5 Anonymous Betting

2.5.1 Overview

There may be social stigma associated with gambling and betting, and there is a need for anonymity for bettors. Traditional online casinos and betting providers may require identity verification and this poses a danger to the information being leaked, should the user account or the casino gets hacked.

It is advantageous for people to bet anonymously as it would prevent their personal activities from being scrutinised or cause social tensions.

2.5.2 How Our Design Enable this Feature

To interact with bETH, a user requires a wallet containing ethereum. Wallets are anonymous and people can create bets or place bets anonymously by interacting with our smart contract. This provides our users with anonymity and prevents any social stigma or tracking.

3. Justification

bETH is disruptive to traditional betting services such as Singapore Pools or 3rd party websites like Stake. Due to our transparent and low fees, the winnings to spending ratio will be higher than traditional sites. The social betting feature also enables users to place bets with their friends casually on our platform.

Another advantage of bETH is that the smart contracts are transparent and immutable. Funds that are deposited for a bet will not be lost to any third party risks. On the other hand, for other online casinos, the funds may be stored in their wallet instead and users may face the risk of the company closing down or getting their accounts locked due to regulations. bETH will only hold deposited funds for the agreed period of time before distributing it out assuming the bet was successful.

4. Limitations

Currently, bETH has a few limitations in its initial stage. As of now, the admin has to manually approve the final results of bets. This method of approving bets is not easily scalable as the administrator could be overwhelmed with a high number of bets. Furthermore, the administrator can face attacks where someone creates a lot of bets, causing a denial of service if the administrator is unable to process all bets, leading to financial loss if bets are not processed.

Another limitation would be the ethereum network's gas fees. During times of high ETH gas fees, users may have to pay more contract fees to place bets. Similarly, the transaction fees that bETH charges the customer must be high enough for the admin to cover individual transaction fees. These extra fees may deter users from placing bets.

Lastly, bETH may be subjected to individual countries' regulations and rulings. These regulations are dependent on individual countries and their view on online gambling. Stricter countries may block bETH's platform to prevent users from gambling. In the event that this occurs, bETH may have lesser revenue due to regulations.

5. System Architecture

5.1 Use Case Descriptions

Table 1: Create Group Use Case Description

Use Case	Create Group	
Scenario	A user wishes to create a group of addresses for the private group to place bets.	
Brief Description	User sends a list of addresses to the smart contract. bETH stores the group and returns a group ID for future creation of bets.	
Actors	User	
Pre-conditions	NIL	
Post-conditions	bETH stores the list of addresses. bETH returns the group ID to the user.	
Flow of Events	**Actor**	**bETH**
	1. User sends a list of addresses to the smart contract.	1.1. bETH generates the new group ID using the index of the groups array. 1.2. bETH stores the list of new addresses in the array of groups. 1.3. bETH returns the new group ID to represent the successful addition of the new group.
Exception Conditions	NIL	

Table 2: View Group Use Case Description

Use Case	View Group	
Scenario	User wishes to view the addresses in the group, by the group ID	
Brief Description	User calls on the viewGroup function and inputs the group ID. bETH returns the list of addresses stored in the group.	
Actors	User	
Pre-conditions	User has previously created the group with the group ID.	
Post-conditions	NIL	
Flow of Events:	**Actor**	**bETH**
	1. User sends the group ID to the smart contract calling the view group function.	1.1. bETH retrieves the list of addresses associated with the group ID in the array of groups. 1.2. bETH returns the list of addresses to the user.
Exception Conditions	NIL	

Table 3: Create Bet Use Case Description

Use Case	Create Bet
Scenario	User wants to create a bet
Brief Description	User creates a bet by giving the descriptions of the bet, and descriptions of each side of the bet. User also defines the opening and closing dates of the bet. User optionally sends the group ID of a group to make it a private bet. bETH creates the bet and returns the betID.
Actors	User
Pre-conditions	NIL

Post-conditions	A bet is created and stored within bETH.	
Flow of Events	**Actor**	**bETH**
	1. User sends bETH the details of the new bet to be created, including: name of bet, description of one side of bet, description of other side of bet, minimum bet, opening date, closing date, and optionally the group ID.	1.1. bETH creates a new bet using the information provided, and sets the bet creator as the address of the user creating the bet. 1.2. bETH returns the bet ID to confirm the successful creation of the bet.
Exception Conditions	NIL	

Table 4: Place Bet Use Case Description

Use Case	Place Bet	
Scenario	User wishes to place a bet by sending ETH to a side of the bet.	
Brief Description	User sends ETH to the payable function on the bet, choosing the amount of ETH to send and the side of the bet that they wish to bet on. bETH records the bet.	
Actors	User	
Pre-conditions	A bet with the bet ID is already created. If the bet is a private bet, the user's address should be in the list of the group's addresses. The user should place the bet after the opening time and before the closing time.	
Post-conditions	bETH records the user's bet.	
Flow of Events	**Actor**	**bETH**
	1. User sends ETH to the contract, defining which bet they wish to bet on and	1.1. bETH checks that the bet is available for betting

	which side of the bet they wish to take.	1.1a. If the bet is a private bet, the sender's address should be in the group's list of addresses. 1.1b. The bet should still be open; the current time should be after the opening time and before the closing time of the bet. 1.2. bETH records the user's bet and stores the user's address and amount in the bet, with the corresponding side of the bet.
Exception Conditions	Bet is closed. Bet is private and the user's address is not in the list of addresses of the group.	

Table 5: View Current Odds Use Case Description

Use Case	View Current Odds	
Scenario	User wishes to view the current odds of the bet.	
Brief Description	bETH calculates the live odds and sends it to the user.	
Actors	User	
Pre-conditions	Bet with the bet ID is created.	
Post-conditions	NIL	
Flow of Events	**Actor**	**bETH**
	1. User sends the bet ID to check on the bet's live odds.	1.1. bETH calculates the live odds of the bet corresponding

		to the bet ID by taking total bets on each side and returning the odds of side1/side2. Since Solidity is unable to output fractions, the output is a signed 64.64-bit fixed point number which can then be processed by the frontend to output floating points.
Exception Conditions	NIL	

Table 6: Payout Use Case Description

Use Case	Payout	
Scenario	Admin verifies the results of the bet, and chooses a side of the bet to be picked as the winner, so that the winning bets will be paid out.	
Brief Description	Admin triggers the payout function, picking a side as the winning side. System updates the results of the bet and puts the payout function on a 24h wait before bets are paid out.	
Actors	Admin	
Pre-conditions	Bet has closed; current time is after closing time.	
Post-conditions	Payout date of the bet is set to 24 hours after the current time. A bet result is updated for the bet, according to the result that the admin entered. At the end of the payout phase, the boolean “completed” for the bet is set to true to true.	
Flow of Events	**Actor**	**bETH**
	1. Admin inputs the betId and result to initiate the payout execution.	1.1. bETH calculates the payout time and triggers a payment delay event for 24 hours.

		1.2. After 24 hours, the payout function is triggered. 1.3. The smart contract pays the admin and the bet initiator a fixed percentage of the total prize pool. 1.4. System calls the Ethereum network to check for the current average gas fees for transactions. 1.5. After subtraction of the transaction fee, the remaining ether is split among the winners iteratively. 1.6. “Completed” attribute of the bet is set to true.
Exception Conditions	NIL	

Table 7: Change Results Use Case Description

Use Case	Change Results
Scenario	Admin changes the result in the case when admin wrongly keys in results wrongly at first.
Brief Description	Admin realises that the original results are entered wrongly. Results are changed to the opposite boolean.
Actors	Admin
Pre-conditions	A result has been determined for the bet. Current time is before the payout date. Bet attribute “completed” is false.
Post-conditions	Results of the bet with the corresponding betID has its results changed.

Flow of Events	Actor	bETH
	1. Admin inputs the betId.	1.1. bETH switches the boolean value of the results.
Exception Conditions	NIL	

5.2 Sequence Diagrams

Sequence diagrams represent the functional interactions between the users, frontend, function host (backend), and the bETH smart contract. Our team has omitted the actual frontend development, but in actual implementations of bETH, the contract creator (admin) would likely present a frontend web interface for easy access to the protocol, where non-tech savvy users are able to create and place bets without the need to access the smart contracts themselves.

The sequence diagrams use a function host (e.g. AWS Lambda, Azure Functions), and omit backend-database interactions that would likely be implemented in real world scenarios. The team focussed on the on-chain interactions, and strategically left out the off-chain processes like user and account management for clarity of the documentation which describes the bETH protocol.

5.2.1 Create/View Group

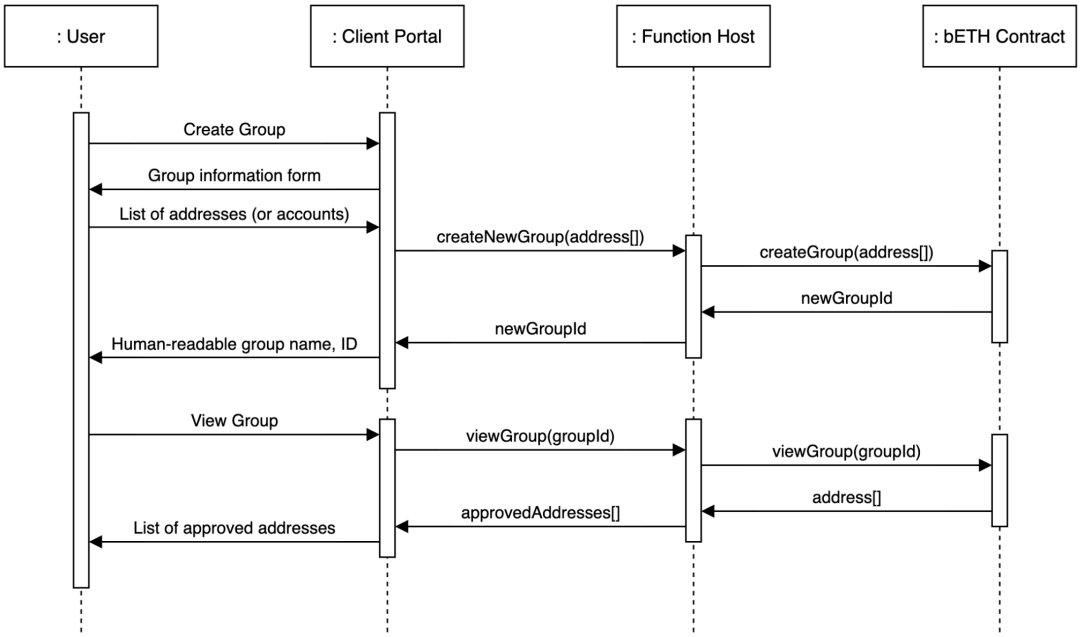

Figure 1: Sequence Diagram for Create/View Group

5.2.2 Create Bet

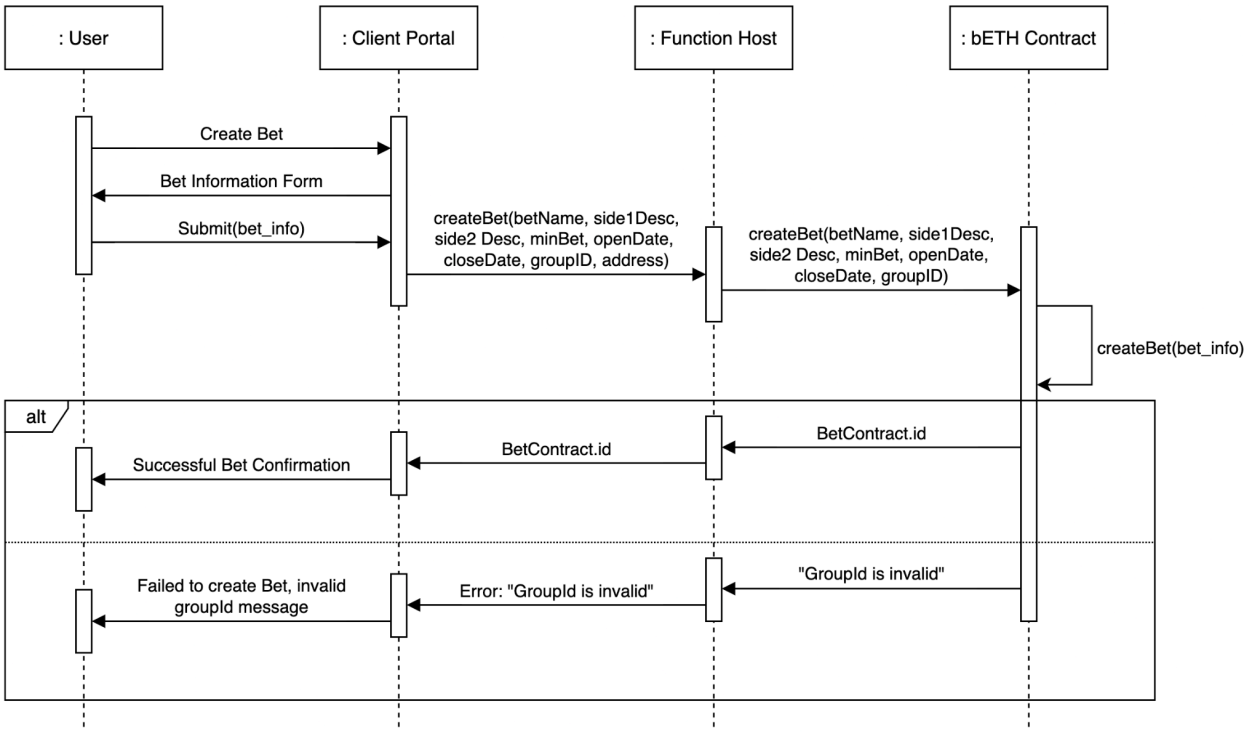

Figure 2: Sequence Diagram for Create Bet

5.2.3 Place Bet

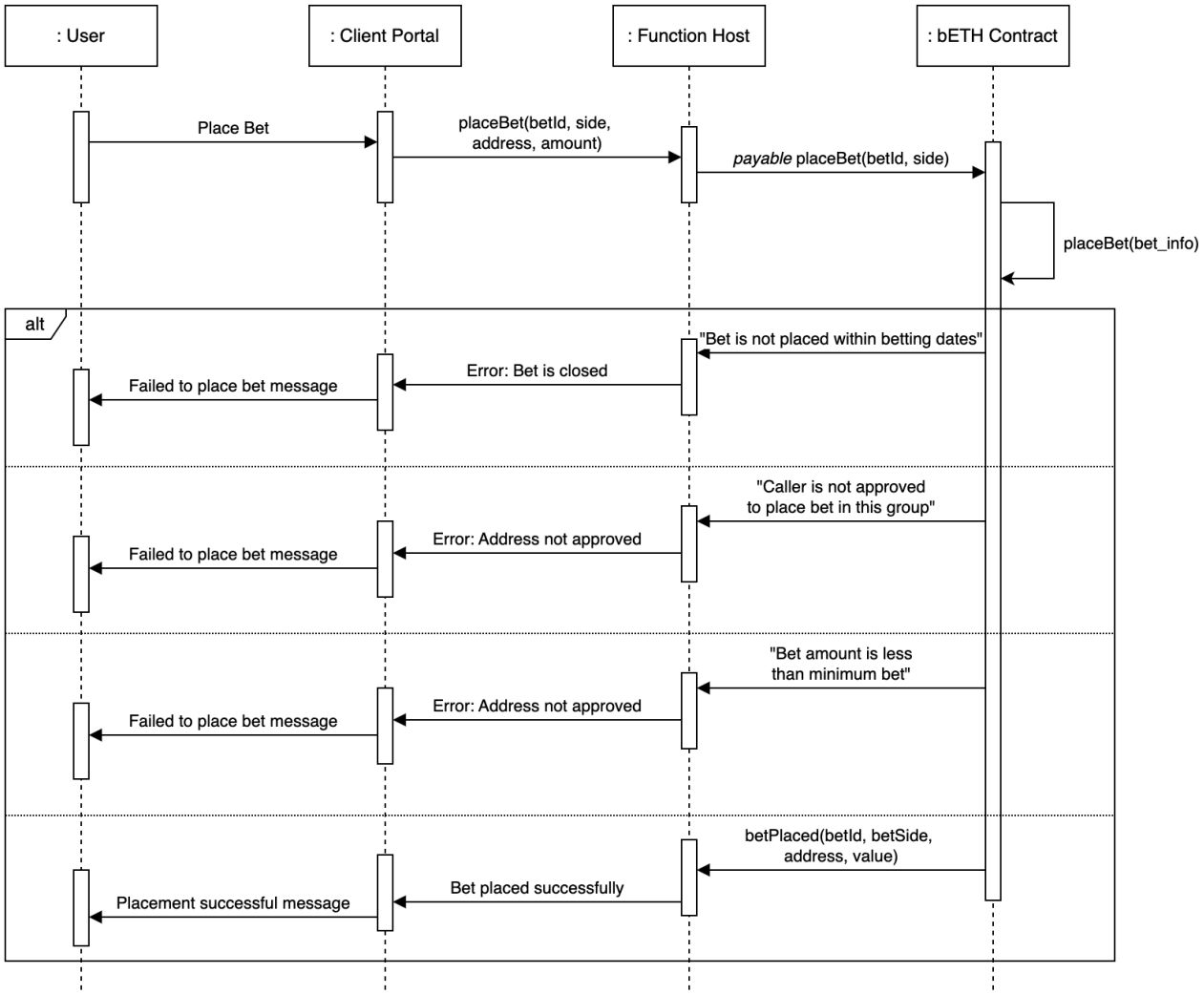

Figure 3: Sequence Diagram for Place Bet

5.2.4 View Odds

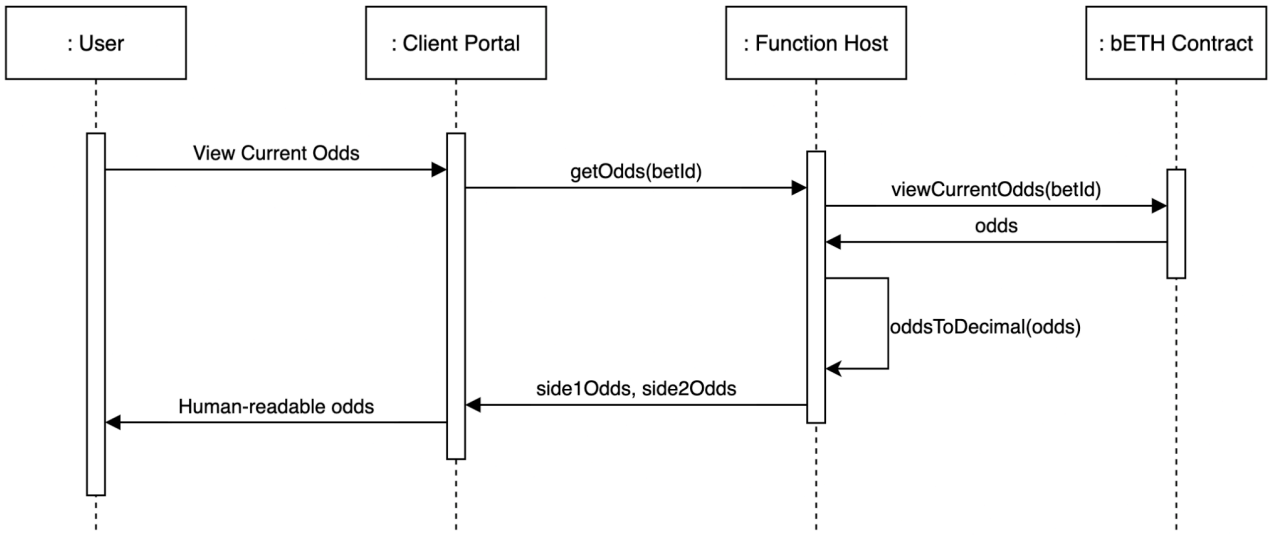

Figure 4: Sequence Diagram for View Odds

5.2.5 Payout

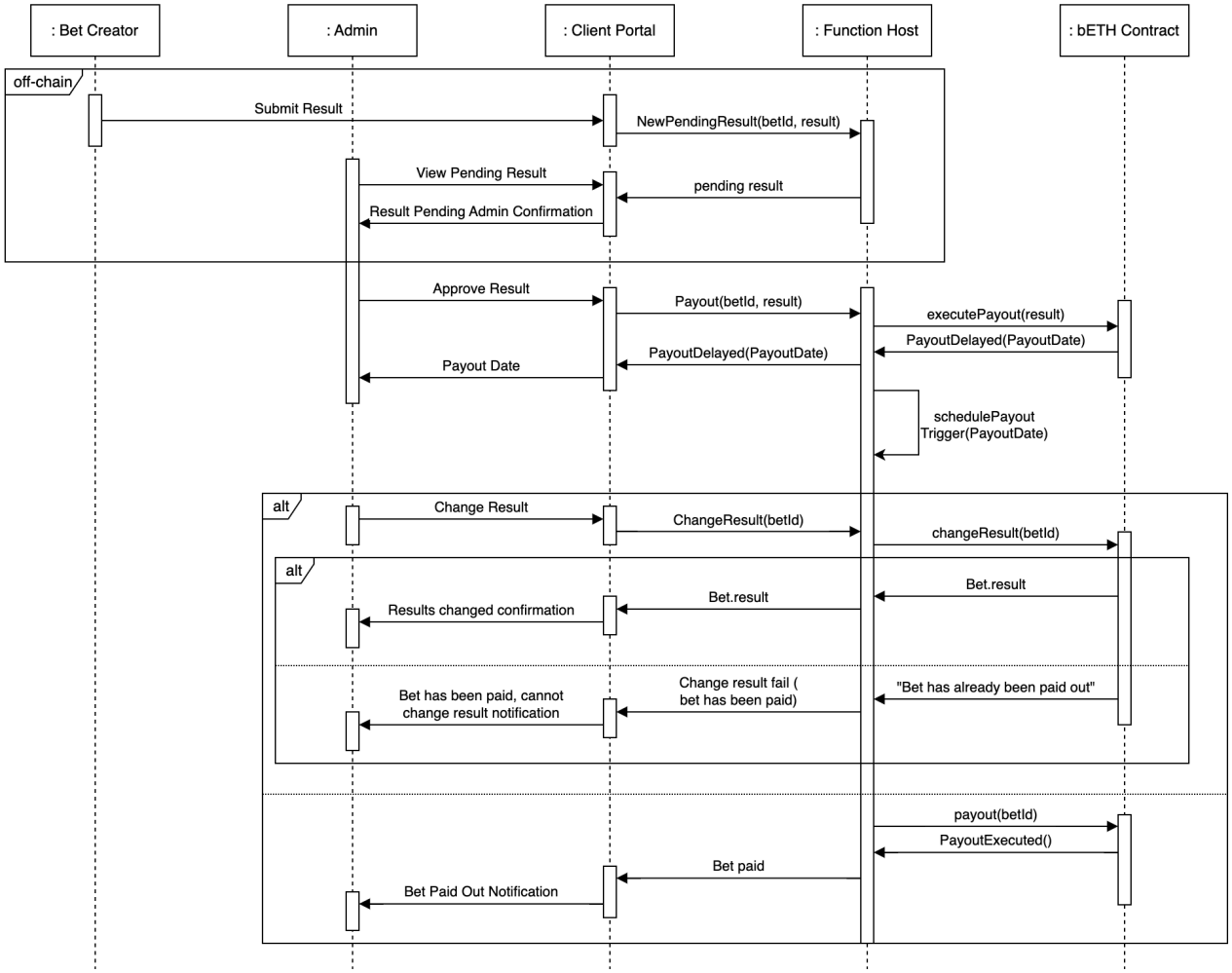

Figure 5: Sequence Diagram for Payout

5.3 Class Structure

bETH Contract

uint8 **commissionFeeBetCreator:** Commission fee percentage for the person who created the bet.
uint8 **commissionFeeAdmin**: Commission fee percentage for the admin.
uint8 **transactionFee:** Transaction fee percentage for winner payouts.
address **admin:** Address of admin.

Bet Object

string **betName**: Name of the bet.
string **side1Description**: Description of the meaning of voting for the bet.
string **side2Description**: Description of the meaning of voting against the bet.
uint256 **minBet**: Minimum amount for each bet.
address **betCreator**: Address of person who created the bet.
uint8 **currentParticipantsCount**: Total number of participants for the bet.
uint256 **groupId**: Group ID for private bets. For public bets, it will always be 0.
uint256 **openingDate**: Opening date for the bet.
uint256 **closingDate**: Closing date of the bet
uint256 **payoutDate**: Payout date of the bet. This date is set when the admin keys in the result and calls the payout function.
bool **completed**: Whether or not the betting event is completed.
bool **result**: Whether the result is for or against the bet.
uint256 **stakeSide1Bet**: Total stakes voted for the bet. This is to avoid repeated summation.
uint256 **stakeSide2Bet**: Total stakes voted against the bet. This is to avoid repeated summation.
address[] **side1BetsAddress**: List of addresses voted for the bet. This is so that we are able to iterate through the array later to pay the winners.
address[] **side2BetsAddress**: List of addresses voted against the bet. This is so that we are able to iterate through the array later to pay the winners.
mapping(address => uint256) **side1Bets**: Hashmap of address voted for the bet and their individual stakes.
mapping(address => uint256) **side2Bets**: Hashmap of address voting against the bet and their individual stakes.

6. Future Expansion

To further improve bETH in the future, an oracle can be introduced to replace the administrator's role of selecting the bet result. If a suitable oracle can be selected that is linked to the bet, the result of the bet can be automatically verified and the bet can be automatically closed without any need for a human administrator to select the bet result. This would allow bETH to scale well since it no longer has the problem of having a high volume of bets.

Another improvement that could be made would be to support more blockchain networks especially those that have lower fees. This could be networks like solana. Alternatively, we can also reduce the fees of ethereum through layer 2 solutions. To do so, we can also bridge the ethereum into a Layer 2 network such as Arbitrum. By doing so, we can transfer the ether through Arbitrum instead of the original Ethereum network, greatly reducing the contract fees, allowing us to reduce the overall transmission fee charged to the users.

Last but not least, We can also introduce a reputation system to gamify the project. This reputation system will be used to track those that initiate bets with high popularity, giving them a score for each high popularity bet. This system would incentivise users into creating more bets, boosting the usage of bETH. Users that have a high win rate will also be highlighted, similar to Bybit's leaderboard of top traders, creating more speculation and promoting the project.