PART I

Our code could be enhanced by using Moralis features for blockchain data interaction.

```
import Moralis from 'moralis';

// Initialize Moralis
await Moralis.start({
  apiKey: process.env.API_KEY });
// Array of addresses
const addressList = process.env.ADDRESS
Const start = process.env.START
Const end = process.env.END
try {
  //
  for (const address of addressList) {
        const response = await Moralis.EvmApi.token.getTokenTransfers({
        chain: "0x1",
        order: "DESC",
        fromBlock: start,
        toBlock: end,
        address: address     });
        console.log(`Transfers for address ${address}:`, response.raw);
  }
} catch (e) {
  console.error(e);
}
```

And there are some actions that can be done concerning :

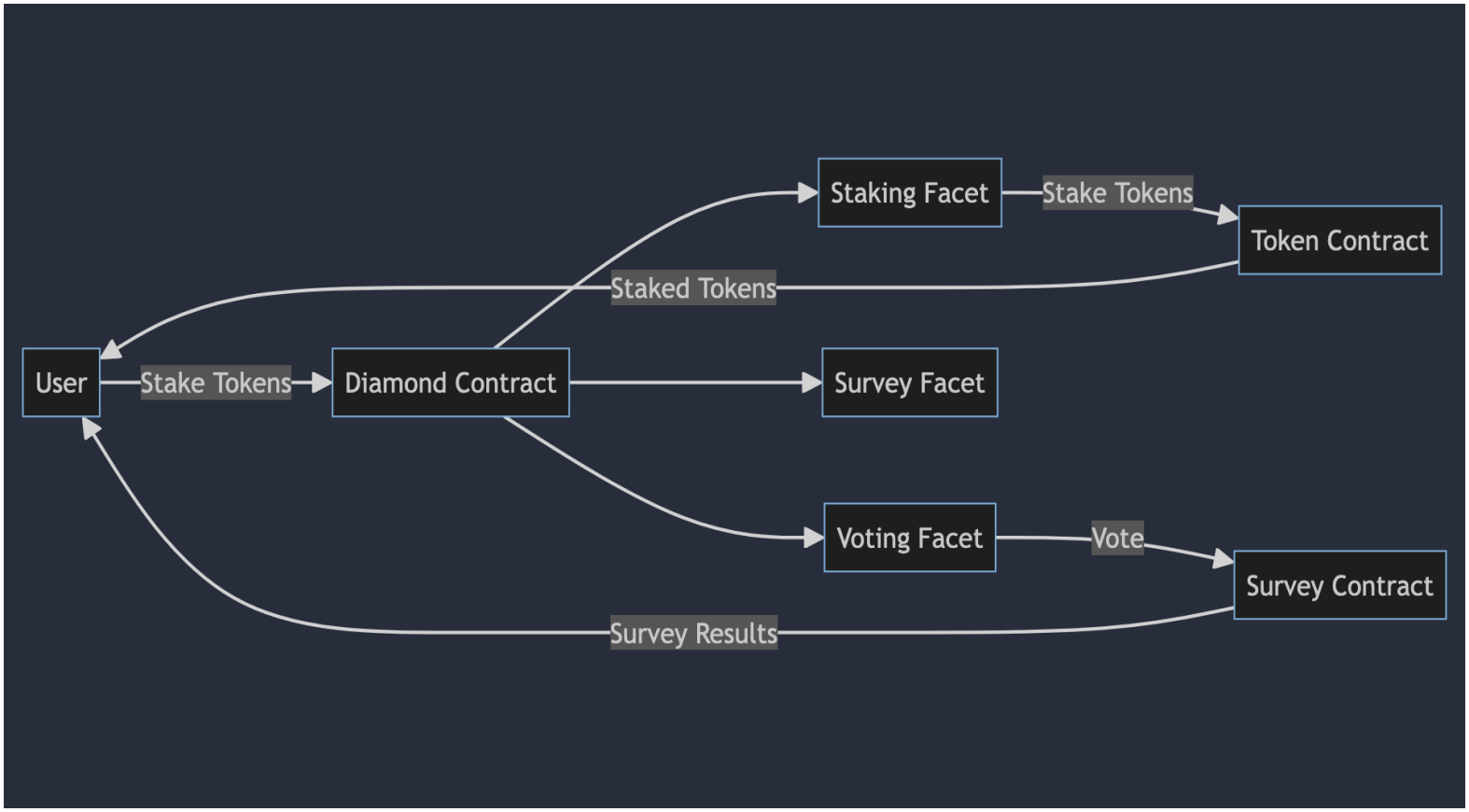
1- Security: Validate API responses and secure the Moralis API key.

2- Scalability: Implement batch processing and caching to handle large data volumes efficiently.

3- Maintainability: Write modular code with robust error handling for better code management.

4- Speed: Use batch process and concurrency to avoid blocks logs limitation

PART II



We should use the Diamond Design Pattern(EIP2535) which specifically addresses the challenges of contract size limitations and facilitates modularity and upgradability in smart contract development.

It could structure it as follows:

- **Diamond Contract:** This will be the main contract that users interact with. It allows users to stake tokens and vote in surveys. It will delegate function calls to different facets.
- **Staking Facet:** This facet is responsible for handling all functionality related to token staking, such as accepting tokens, tracking staked tokens, and handling un-staking.
- **Voting Facet:** This facet is responsible for managing the voting process. It ensures that only users who have staked the required token amount can vote in a survey.
- **Survey Facet:** This facet creates and manages surveys. It links surveys to tokens and ensures only those who have staked the corresponding tokens can participate.
- **Token Contract:** This contract is in charge of the token economics and interactions.

For security:

1-The Diamond contract should only allow authorized facets to make changes.

2- A multi-signature wallet or a decentralized autonomous organization (DAO) could be used to manage the upgrade process, ensuring no single entity has full control.

For maintainability and upgrades :

The Diamond pattern allows for efficient upgradeability because facets can be added, removed, or replaced without changing the main Diamond contract, preserving data consistency.

Conclusion:

Possible improvements could be implementing a reward system for staking and voting, adding additional security measures like a pause mechanism in case of emergencies, and implementing gas optimization practices to minimize transaction costs.