**Executive Summary**

The main target of our project is using Ethereum as an efficient way to handle online pet shops. We want a decentralized application (DApp) that takes Ethereum addresses and associates them with online pet shop operations. Based on the petShopTutorial, we will modify the website's structure and write the smart contract and front-end logic for its use.

For an online application for pet stores use Web3-based client-server systems to host their services, this can create mistrust between the parties involved in transactions because the pet store has centralized control over data processing and storage. To address this issue, decentralized applications (DApps) can be used. DApps run on a peer-to-peer network of computers, which is the blockchain, rather than a single hosting server. They function like regular apps, but they are not controlled by a single entity. Operational data is encrypted and recorded on a transparent public blockchain, such as Ethereum or EOS. MetaMask Wallet with a Web3 browser can be used to control the token while using the DApp. DApps do not have single points of failure like hosting servers, so they are not subject to downtime or limits. They are also resistant to modification and censorship by central authorities. By using the Ethereum blockchain, pet filtering and returning information can be published in a transparent and trustworthy manner. User privacy can be maintained using an Ethereum-based identity verification wallet that does not contain personal information.

The modified pet shop DApp leverages smart contracts to allow users to perform features using their Ethereum wallet. Based on the petShopTutorial, features are being added.

1. *Show the owner of the pet (possibly none or 1)*

By clicking a specific button, the current owner of the pet will be shown.

* updated src/js/app.js -> modified markAdopted(), handleAdopt(), initContract()
* updated contracts/Pets.sol for new data struct to support new functions: added create(), idExists(), idFind(); modified adopt(), getPets()

*2. Up/down vote a pet*

By clicking the voting button, the user can up/down vote a pet, the current vote number will be shown in time.

* updated src/js/app.js :added markVotes(), handleUpVote(), handleDownVote(); modified init(), bindEvents()
* updated contracts/Pets.sol -> added upVote(), downVote()

*3. Return a pet*

* updated contracts/Pets.sol to support pet return: added returnPet()
* updated src/js/app.js : modified init(), bindEvents(), markAdopted() to support the behavior of "Return" button, added handleReturnPet() to return the pet for a fee
* updated index.html to support "Return" button

*4. Adoption histories*

*5. Track the number of adopted pets and served customers*

* updated contracts/Pets.sol to support pet return -> added addCust() to trac customers, trackPet() and trackCust() to get how many custumers have been served and how many pets adopted
* updated src/js/app.js -> add markPets(), markCusts()to support the behavior, modified markVotes() handleAdopt() to trigger the behaviour
* updated index.html to support information display