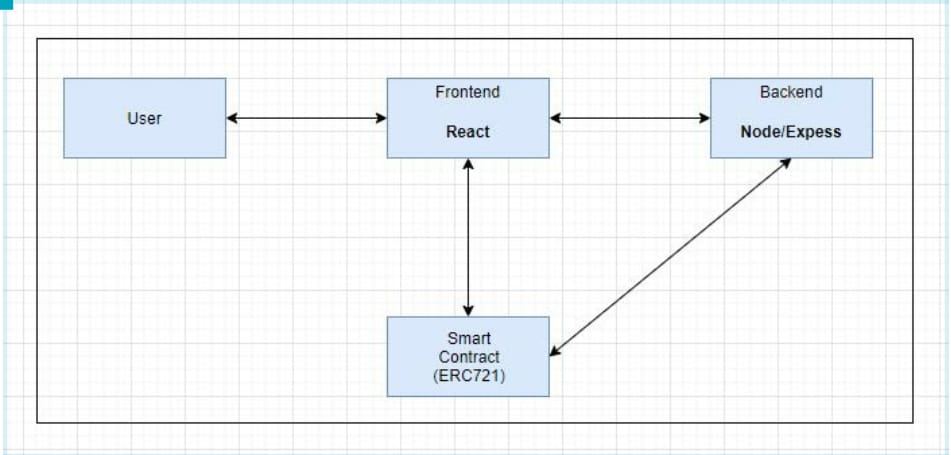
Architectural Documentation

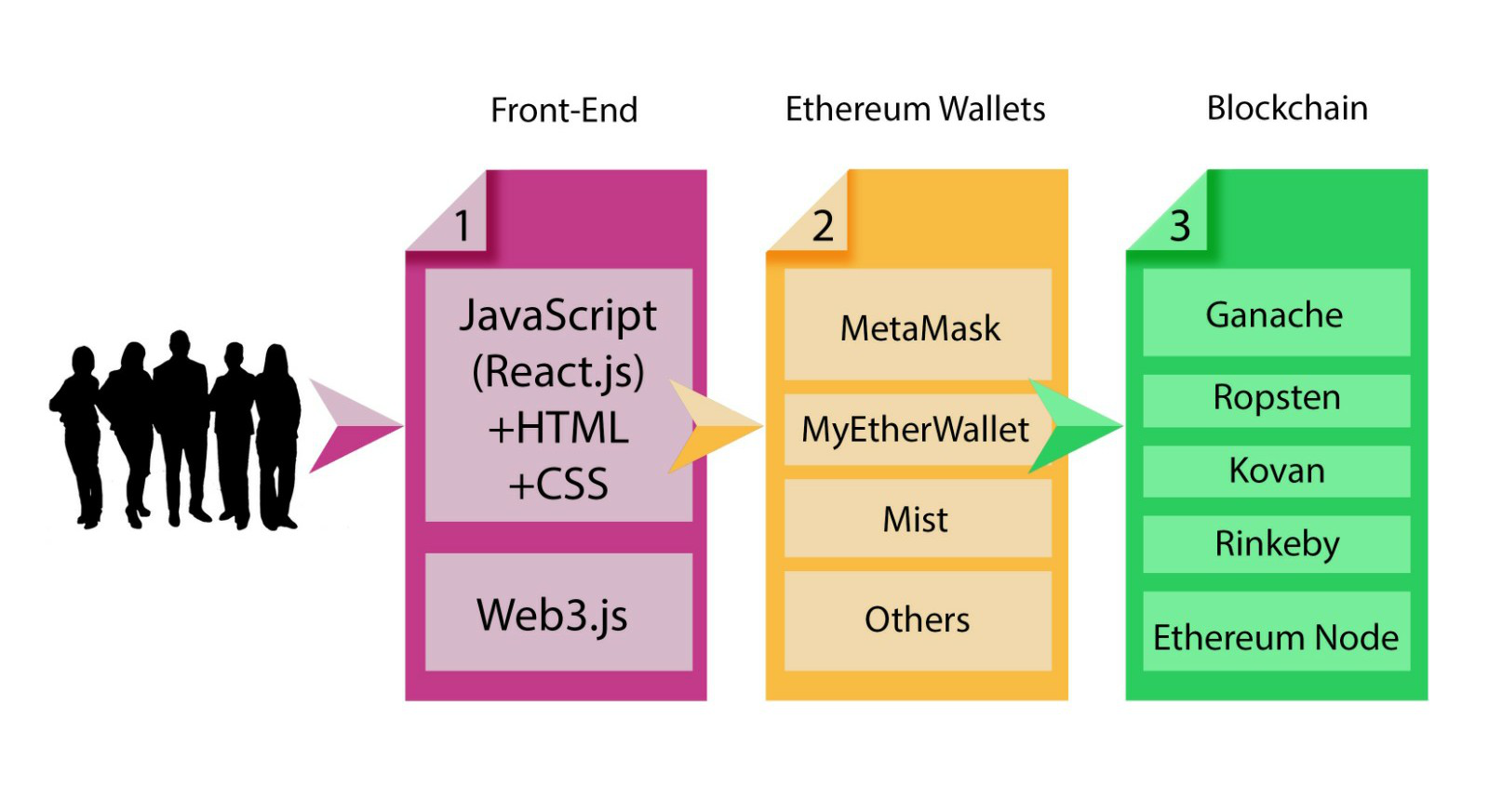
# Digital Art Platform:

The decentralized digital art platform provides the artist with a place to sell their arts, and thereby a buyer finds a marketplace. It has the following features:

* It allows artists to easily sell their product without a third party, with no hidden fee during the transaction
* All art trading history is available in the blockchain; it is transparent
* The user can browse digital arts in the art gallery
* Each piece of art shows its product information
* The product owner can resell the art with the desired price
* Once a deal has been completed, the ownership of the art will be transferred, and the buyer will pay the seller for the art

# Technical Architecture:



The web app is built with reactJS and web3 while the backend is built using NodeJS and Express.The smart contract is deployed using Truffle and Ganache.The smart contract functionalities have been tested using truffle suite. The generated abi is stored in the backend. ReactJS uses the abi from backend to interact with the smart contract. Web3 module is used in the react app to call the functionalities of the smart contract.

The above figure shows the techstack visual representation of the project.The frontend uses ReactJS along with HTML and CSS for building the app and web3 for interacting with the smart contract functionalities. Next phase would be the Ethereum wallets such as Metamask and MyEtherWallet. A Metamask browser extension is highly recommended for using this app. The smart contract is tested compatible to be deployable in most of the test nets some prominent ones being Ropsten and Rinkby. The entire app was built and tested using the local blockchain.

**Data**

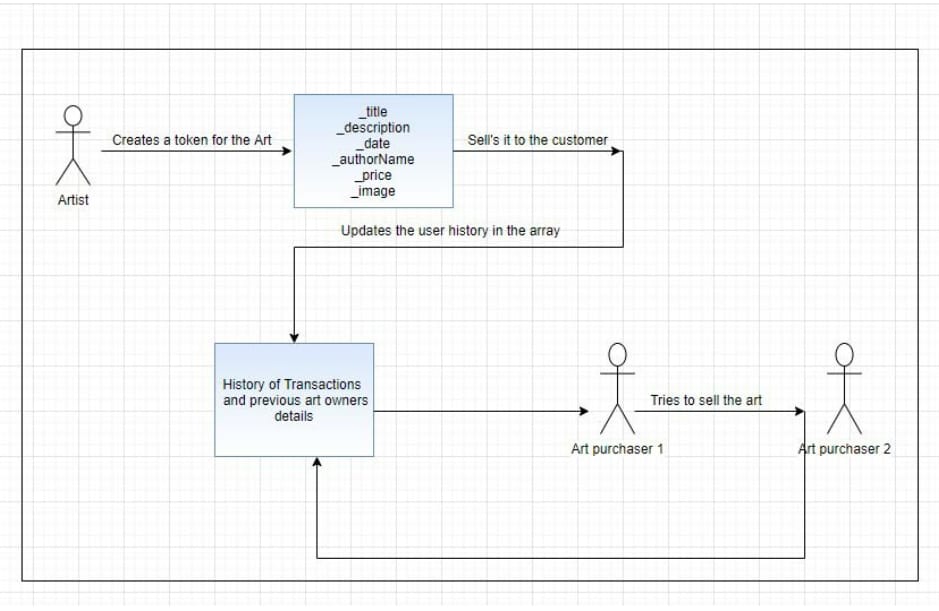
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Structure** | **Purpose** |
| Name, Symbol | String | n/a | Token Information |
| ownedTokensCount | mapping | ( address => uint ) | No. of tokens owned by owner |
| tokenOwner | mapping | ( uint => address ) | TokenID corresponding to owner |
| operatorApproval | mapping | ( address => mapping (    address => bool ) | Enable or disable third party (operator) to manage assets |
| Art | struct | { id, title, description, date, price, author, owner, status etc } | Each art has its own token attributes. |
| ArtTxn | struct | {id,price,seller,buyer,txnDate,status} | Keeps the information record of transaction history |
| status | Struct field | 0 or 1 | For sale (or) sold |

**Functions**

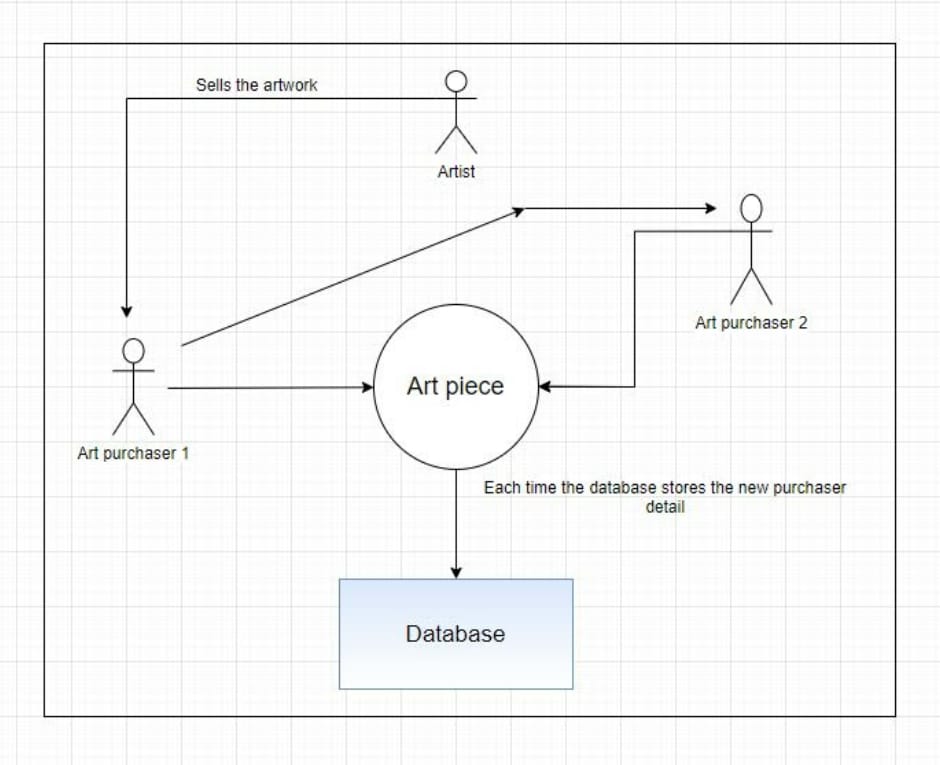
**Functions:**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| *ERC721 functions ()* | Used to transfer ownership, check balance, get approved, check owner, checks if the address is approved for operating, etc. |
| *CreateTokenAndSellArt()* | Used to publish and sell the artwork by authors |
| *buyArt()* | Used to place an order for digital art, verifies all the conditions and then updates the ownership info. |
| *resellArt()* | Used when art owner wants resell the art piece at their own price |
| *findArt()* | Used to find art details by passing token Id |
| *findAllPendingArt()* | Used to find all art pieces for sale in the gallery |
| *getArtAllTxn()* | Used to retrieve transaction history and details |

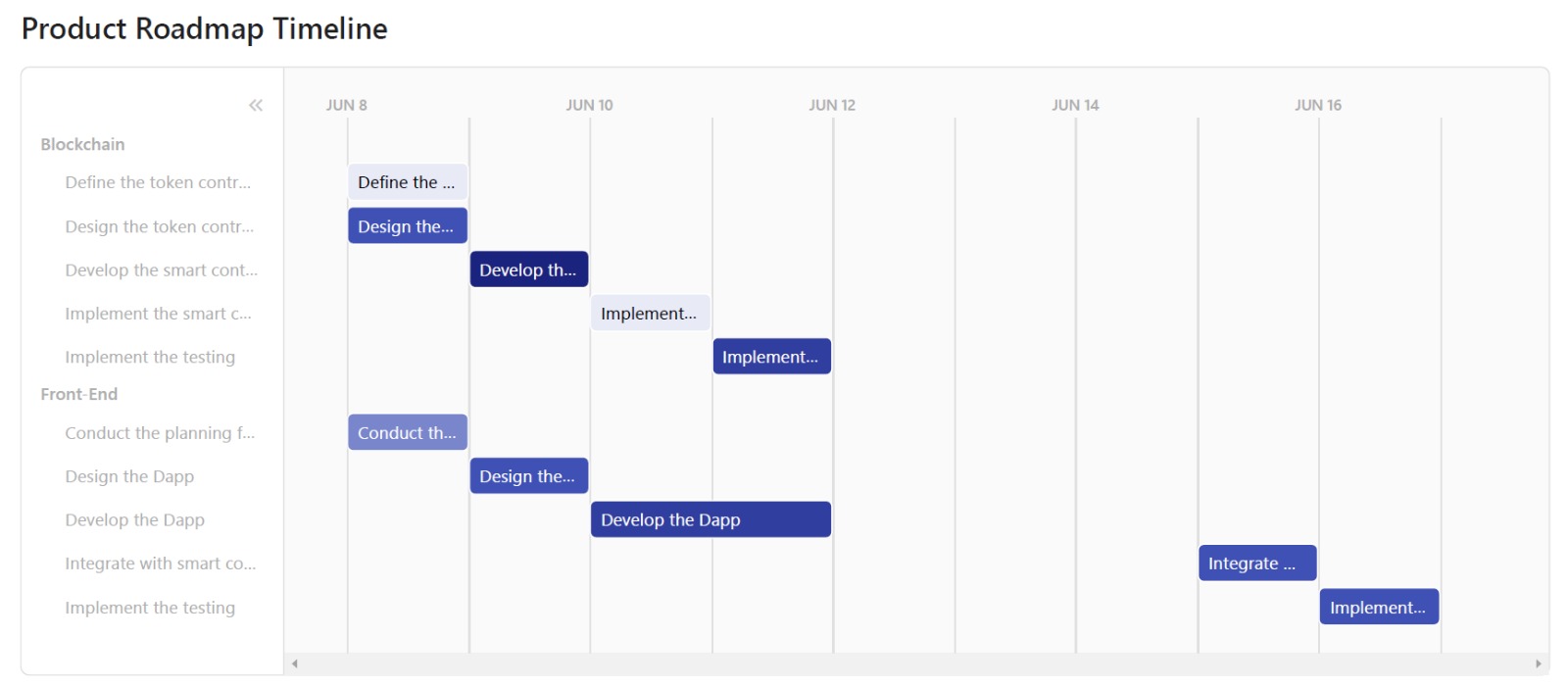
# Flow Diagram:

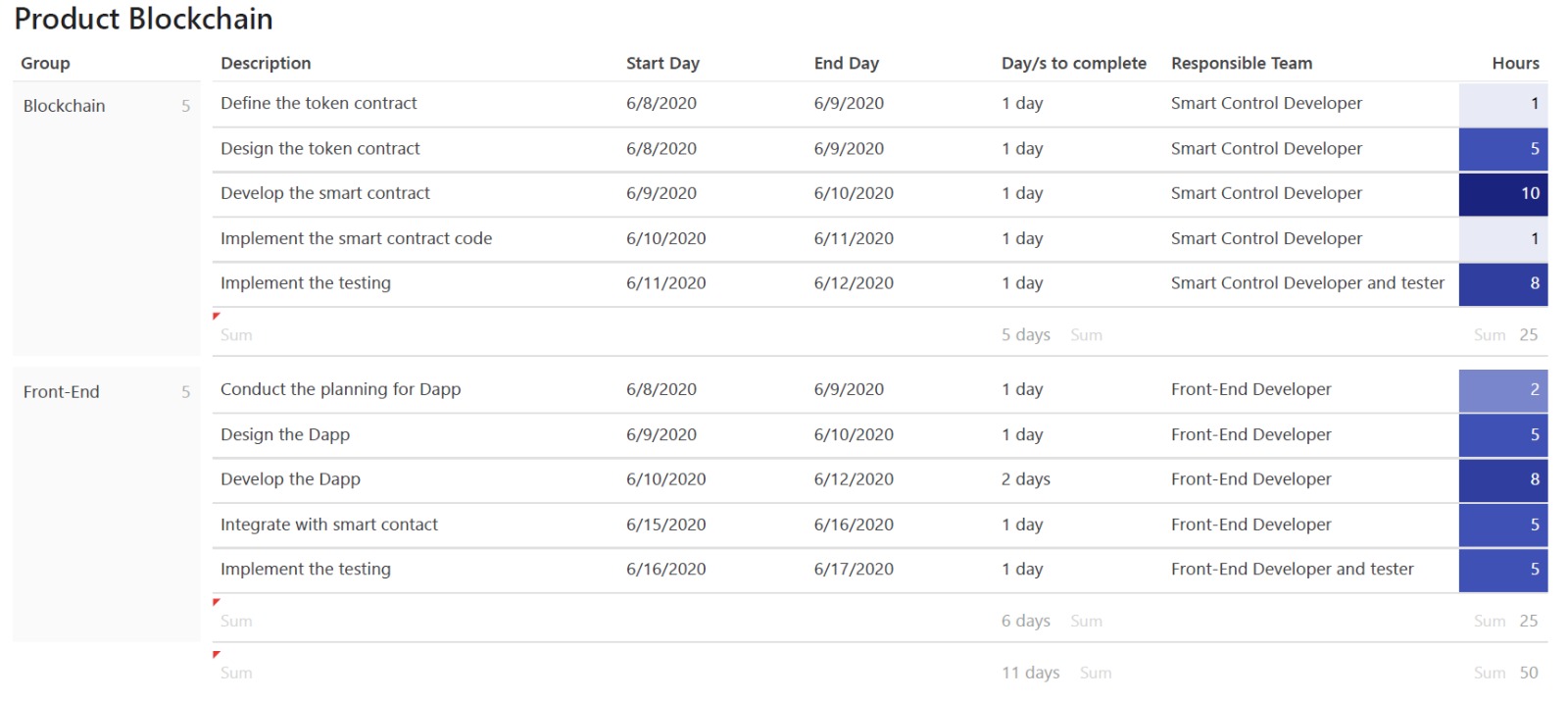


# State Diagram:

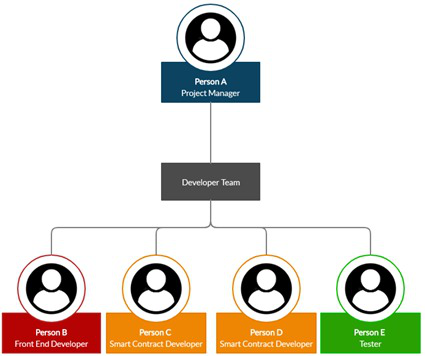


# Project Plan:





# Staffing model:



# Cost Estimation:

