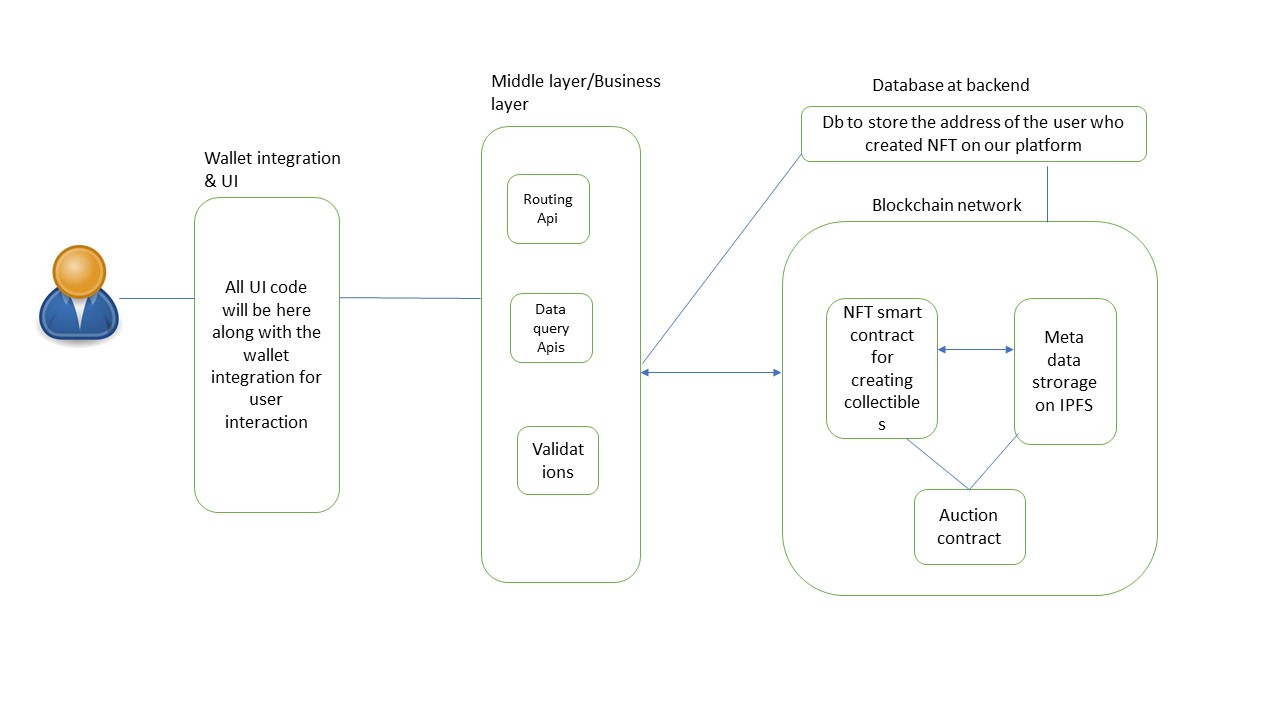
**NFT marketplace implementation Architecture**



**Technology stack**

**React** - Frontend

**Node.JS** - Middle layer/Business Layer logics

**IPFS-** Metadata/tokenURI

**MongoDB/Postgress** - For additional data storage

**Blockchain** - backend smart contract implementation

Detail Flow and references

1. Our frontend layer will contain all the UI code along with the wallet integration part
2. All the business logic, Api routing and validations will be performed at middle layer
3. Backend will contain the implementation of smart contracts on Blockchain for NFT creation and interaction
4. We may need to use additional DB for storing some additional data for the dashboard

**Functionalities**

1.User will be allow to connect with wallet like metamask/walletconnect

2.User can create NFT with metadata and put out for sale/Acution

3.We can store the metadata in the form of tokenURI in IPFS

4.we might need to use the graphQL to query the onchain data from blockchain

**Backend implementaion Needs**

1. **Token Minter (ERC-721/ERC-1155)**
   * Use Opensea/Rarible as references
   * Can also allow users to create highly programmed ERC-721s/1155’s using this minter (standard/advanced interfaces)?
     + E.g. [Dynamic NFTs](https://blog.chain.link/build-deploy-and-sell-your-own-dynamic-nft/)
2. **Escrow Contracts with below features**
   * Auction, reserve price, buy now, make an offer, etc. functionalities
   * Royalty and fee split features to be baked into these contracts (assuming primary market would require a different set of contracts that secondary market)
   * Support for the following ERC-20 tokens: USDC, DAI, and ETH
3. **Meta Transactions implementation**
   * See OpenSea & OpenZeppelin github can be used as reference for meta transaction
   * See “meta transactions” link below
     + [Biconomy](https://docs.matic.network/docs/develop/metatransactions/metatransactions-biconomy)
     + [Gas Station Network](https://docs.matic.network/docs/develop/metatransactions/metatransactions-gsn)
4. **Wallet Integration/Creation**
   * Metamask/Fortmatic
     + See Metamask’s [Custom Networks API post](https://consensys.net/blog/metamask/connect-users-to-layer-2-networks-with-the-metamask-custom-networks-api/?utm_content=156337458&utm_medium=social&utm_source=twitter&hss_channel=tw-3278906401)

Technical references for smart contracts

* [OpenZeppelin Developer Portal](https://docs.openzeppelin.com/contracts/3.x/)
  + [ERC-721](https://docs.openzeppelin.com/contracts/3.x/erc721)
  + [ERC-1155](https://docs.openzeppelin.com/contracts/3.x/erc1155)
  + [Gas Station Network (Meta Transactions)](https://docs.openzeppelin.com/contracts/3.x/gsn)
  + [Payment](https://docs.openzeppelin.com/contracts/3.x/api/payment)
  + [Presets](https://docs.openzeppelin.com/contracts/3.x/api/presets)

**Reference Githubs (what we could fork and build off of)**

* [Superare Github](https://github.com/dutch8596/pixura-contracts)
  + Standouts
    - [Creator Royalty](https://github.com/dutch8596/pixura-contracts/blob/master/contracts/build/IERC721CreatorRoyalty.json)
      * Creator royalty contract to guide fee split functionality on primary and secondary market
    - [ERC-721 Token Creator](https://github.com/dutch8596/pixura-contracts/blob/master/contracts/build/IERC721TokenCreator.json)
    - [SendValueorEscrow Contract](https://github.com/dutch8596/pixura-contracts/blob/master/contracts/build/SendValueOrEscrow.json)
    - [SuperRare Market Auction](https://github.com/dutch8596/pixura-contracts/blob/master/contracts/build/SuperRareMarketAuctionV2.json)
* [OpenSea Github](https://github.com/ProjectOpenSea)
  + Standouts
    - [Meta Transactions Contracts](https://github.com/ProjectOpenSea/meta-transactions) - enable gasless transactions