**SOFTWARE DESIGN**

**FunixSwap**

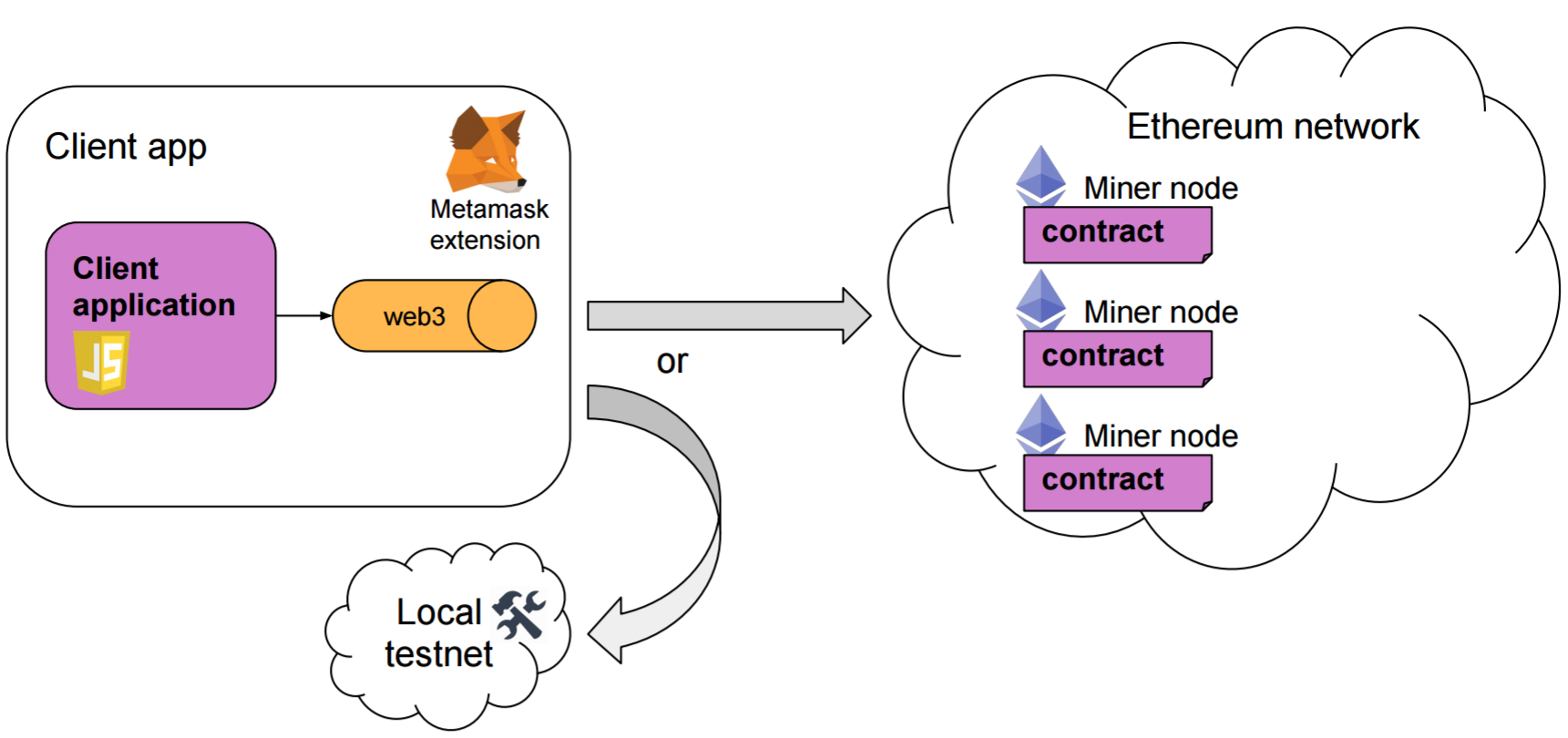
**1. Architecture**

/\*

*Visualize the high-level architecture by a set of modules/entities, connect those modules that have relation and label the connection.*

*For example:*

*The following sample diagram (source: internet) shows a high-level architecture of a generic decentralized application:*



\*/

*//TODO: draw a high-level architecture of FunixSwap to include the following entities:*

*- An Ethereum network where all FunixSwap smart contracts deployed on.*

*- A web browser where the front-end of FunixSwap is displaying*

*- The web3.js that will help font-end to interact with smart contracts*

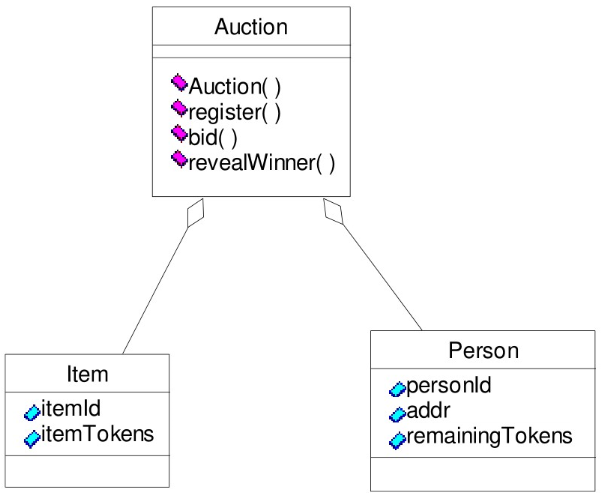
*- The Metamask*

**2. Class Design**

/\*

*We will focus on smart contract design. Remember the design of Chinese Auction that we have done in the Smart Contract course. We will need to do a simple class design for smart contracts here. We have a Reserve smart contract and an Exchange smart contract. For each one, try to define all necessary attributes and functions.*

*(sample of class diagram)*



\*/

//*TODO: Design the Reserve smart contract and Exchange smart contract and put these contracts to a class diagram.*

**3. Functional Design**

/\* *For each function in the smart contract, we need to define its signature, including the following:*

* *Function name*
* *Function purpose*
* *List of parameters (including name and data type for each parameter)*
* *Return type*

\*/

*//TODO: define function signature for all functions in the Reserve and Exchange smart contract*