Best practices for dApp testing automation.

Testing

Before deploying an Ethereum Smart Contract to any official network (mainnet or testnet), the Smart Contract Developer must verify the code quality of their developed Smart Contracts.

For high-quality dApp delivery, we recommend the following 3 phases:

- Unit Testing
- End-to-End
- End-User Testing

Unit Testing

The developer must ascert the correct functionality of the source code. This can be accomplished by creating tests that:

- Check data types of variables
- Verify the result you receive is the result expected
- Prove the logic of your program.

For accomplishing those purposes in a dApp development, we recommend the Truffle suite. Truffle's main goal is to provide a development environment for Ethereum developers. Truffle

includes the Ganache private blockchain, which allows the Ethereum developer to migrate their compiled Smart Contracts for no-cost exhaustive testing.
In addition to the Truffle suite, we recommend the following testing frameworks:

- Mocha
- Chai

Those 3 frameworks provide a simple and intuitive syntax for Smart Contract unit testing.

End-to-end testing

End-to-end testing refers to the testing of the complete dApp's flow.
It asserts correct functionality, optimal load times, optimal data response time and the dApp's high level functionality.

End-to-end tests can be executed in a dedicated server.

End-to-end testing is the bridge that connects QA and the developer team.

For End-to-end testing, we recommend the following frameworks:

- Selenium
- Protractor

Those End-to-end testing frameworks are one of the most widely-used globally.