Commünitäs Testing Approach

Testing an application not only ensures its quality and maximizes the test coverage, but it also minimizes the risks that are associated with malfunctions & attacks. With Blockchain technology, we have specific testing issues like smart contracts, blocks or DAO testings, that are even more important, due to the immutability nature of it and even more if economic resources (tokens, pools, fund locking, etc.) are involved.

Lack of security testing leads to developing Blockchain applications that are vulnerable to attacks at network level, user level and mining level. Failing to test for performance and load testing gives little or no insight into how the Blockchain application performs in both production as well as under specific workloads and network conditions. Along with standard testing, tools and the best practices in place, testing for block size and chain size are also essential. Without proper validation of block-size and chain-size leads to failure of Blockchain applications.

Also, we must ensure an accessible and easy to use interface for everybody, in order to allow the widest audience possible without discrimination. Also, an easy-to-use and quick-to-learn interface is required to make sure that anyone, regardless of age, religion, studies, etc. could use our platform.

With all of this in mind, we will do the following testing in our platform, to ensure an optimal experience for our users, organized in four parts:

- Software Tests
    - Technical Tests
    - Security Tests
- Product Tests
    - Product Tests
    - Marketing Tests

Let’s define, one by one, all of them and their specific testing issues and tools.

Software Testing KPIs and Metrics

The technical & security tests share almost the same (software) testing KPIs and metrics. We will cover them together and we will only specify the differences in each section, showing and explaining the more common ones at the beginning of this section. These shared KPI & metrics are:

Nº of Active Bugs: Helps identify the status of a defect (also called bug) -new, open, or fixed- and allows to take the necessary steps to rectify it. These are measured based on the threshold set by the team and are tagged for immediate action if they are above the threshold.

Nº of Automated Tests: While monitoring and analyzing the key performance indicators, it is important for the test manager to identify the automated tests. Through tricky, it allows the team to track the number of automated tests, which can help catch/detect the critical and high priority Bugs introduced in the software delivery stream.

Percentage of Covered Requirements: With the assistance of this key performance indicator the team can track the percentage of requirements covered by at least one test.

Nº of Passed Tests: The percentage of passed tests is evaluated/measured by the team by monitoring the execution of every last configuration within a test. This helps the team in understanding how effective the test configurations are in detecting and trapping the Bugs during the process of testing.

Nº of Test Instances Executed: This key performance indicator is related to the velocity of the test execution plan and is used by the team to highlight the percentage of the total instances available in a test set. However, this KPI does not offer an insight into the quality of the build.

Nº Test Executed: Once the test instances are determined the team moves ahead and monitors the different types of test execution, such as manual, automates, etc. Just like test instances executed, this is also a velocity KPI.

Nº Bugs Fixed Per Day: By evaluating this KPI the test manager is able to keep a track of the number of Bugs fixed on a daily basis as well as the efforts invested by the team to rectify these Bugs and issues. Moreover, it allows them to see the progress of the project as well as the testing activities.

Direct Coverage: This KPI helps to perform a manual or automated coverage of a feature or component and ensures that all features and their functions are completely and thoroughly tested. If a component is not tested during a particular sprint, it will be considered incomplete and will not be moved until it is tested.

Percentage of Critical & Escaped Bugs: The percentage of critical and escaped bugs is an important KPI that needs the attention of software testers. It ensures that the team and their testing efforts are focused on rectifying the critical issues and bugs in the product, which in turn helps them ensure the quality of the entire testing process as well as the product.

Time to Test: The focus of this key performance indicator is to help the software testing team measure the time that a feature takes to move from the stage of “testing” to “done”. It offers assistance in calculating the effectiveness as well as the efficiency of the testers and understanding the complexity of the feature under test.

Defect Resolution Time: Defect resolution time is used to measure the time it takes for the team to find the bugs in the software and to verify and validate the fix. Apart from this, it also keeps a track of the resolution time, while measuring and qualifying the tester’s responsibility and ownership for their bugs. In short, from tracking the bugs and making sure the bugs are fixed the way they were supposed to, to closing out the issue in a reasonable time, this KPI ensures it all.

Successful Sprint Count Ratio: Though a software testing metric, this is also used by software testers as a KPI, once all the successful sprint statistics are collected. It helps them calculate the percentage of successful sprints, with the assistance of the following formula:

Successful Sprint Count Ratio: (Successful Sprint / Total Number of Sprints) x 100

Quality Ratio: Based on the passed or failed rates of all the tests executed by the software testers, the quality ratio, is used as both a software testing metrics as well as a KPI. The formula used for this is:

Quality Ratio: (Successful Tests Cases / Total Number of Test Cases) x 100

Test Case Quality: A software testing metric and a KPI, test case quality, helps evaluate and score the written test cases according to the defined criteria. It ensures that all the test cases are examined either by producing quality test case scenarios or with the assistance of sampling. Moreover, to ensure the quality of the test cases, certain factors should be considered by the team, such as:

- They should be written for finding faults and bugs.
- Test & requirements coverage should be fully established.
- The areas affected by the bugs should be identified and mentioned clearly.
- Test data should be provided accurately and should cover all the possible situations.
- It should also cover the success and failure scenarios.
- Expected results should be written in a correct and clear format.

Furthermore, as we saw in the previous section, a Blockchain project has its own specific metrics to track, different from the typical software development one.

Blockchain Specific KPI & Metrics

Total Transactions: The total of transactions performed by our platform and stored in the blockchain.

Market Capitalization: The total value of the platform assets, transactions, pools, addresses, and all the tokens.

Mempool Total Value: The total amount of money locked in the platform pools.

Connected Wallets: The number of wallets/addresses connected to the platform.

Finally, it's also important to track the health of the project itself (viewed as a software development project) and it's community. To do so, we will also track the following metrics each month.

Number of Developers: Active developers working in the project open-source repository. It shows the quality and activity of the project.

Number of API calls: The total amount of API calls to interact with the platform.

Commits: The total amount of work submitted to our repository.

App downloads: Number of downloads generated by our mobile app (if present).

Communities builded: The total number of organizations created in the platform.

Technical Tests:

Testing a blockchain is vital to ensure the bugs in a decentralized ledger are eliminated. This ultimately protects a business against the negative consequences of blockchain poor operation.

- White box testing:

  White box testing is testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. The testing can be done at system, integration and unit levels of software development.

  - Part 1 : Source Code Audit
    Understand the source code of the application, find security issues, prevent attacks & improve the workflow of the application. As the `Communitas` platform will be an open-source project, this could be done by everyone (increasing transparency and reliability), but we will create `bug bounties programs` in the Beta development part to attract developers & find early bugs & malfunctions in our code.

  - Part 2: (Unit) Tests cases
    Develop little automated tests (Unit tests) for each critical process or function to assert that given a known input, the desired result is always the output. The aim is to test each part of the software by separating it. It checks that components are fulfilling functionalities or not. This kind of testing will be performed by the platform developers itself.

  Tools:

  Ethereum Tester: Open-source testing library with manageable API support for various testing requirements.

  Embark: A testing framework that focuses on developing decentralized applications (dApps) that run on various systems or nodes. It has integrations with Ethereum blockchain, IPFS, and decentralized communication platforms such as Whisper and Orbit.

- Black box testing:

  Once the platform is ready for the launch, we will allocate a 1-week bounty program to allow (ethical) hackers & developers to perform a penetration test to find

vulnerabilities in `Communitas`. This penetration test will focus on aspects like:

- Cross-Site Request Forgery (*CSRF*)
- Privilege Escalations
- Cross-Site Scripting (*XSS*)
- Code Executions
- SQL Injection
- Authentication Bypasses
- Server-Side Request Forgery (*SSRF*)
- Data Leaks

Vulnerability severity will be judged by the standard model on the Internet: OWASP model

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

We will define the bug bounty payout in the future, depending on the severity risk. The tools used in this section will be determined by each participant depending on their skills & objectives.

- User Testing:

With this methodology, the goal is to test the platform/app at the user level (UI/UX) before launching it, to detect `human-problems` like the look-and-feel of the interface, sensations, difficulty of use of the platform, texts, images, page flow, etc. This type of testing usually have five different stages:

- Planning: We will define the goals in advance.
- Participant Recruitment: From TryMyUI and network (see below).
- Test: Users do the proper test with the specified scope & guides.
- Collect and evaluate feedback: List all found bugs/problems. Collect all the feedback & suggestions.
- Closure: Solve all the bugs, translate feedback/suggestions to the next product development phase.

This will be done by contracting a professional usability testing services provide and do from 20 to 100 users reviews with it, to get feedback. As they define (from TryMyUI.com website):

> *"Watch and listen as users navigate your platform/app and describe what's going through their minds. You'll receive a screencap video including user interactions like mouse movements, taps, swipes, and keypresses; real-time audio captured from the user's microphone, and written responses to a post-test survey"*

Also, we will use our social and familiar network to add almost 30 other non-expert user feedback. Resulting in at least, 50 human user testing video records and reports.

KPIs and Metrics: User global valuation, User-specific valuation (usability, desirability, usefulness, credibility), user feedback.

Tool: TryMyUI

- Accessibility Testing:

We will ensure that the platform is usable by everybody, including people with disabilities like hearing, color blindness, old age or other disadvantaged groups. We will do this testing process in two ways:

  - Automated testing:
    We will use a wide range of the awesome tools provided by W3C organization provides to ensure the WCAG 2.0 standards for accessibility. This includes, at least:
    - A11Y color contrast accessibility Validator: a tool that displays the color contrast issues of a web page per WCAG Guidelines. The results display color combinations that fail the contrast checkpoints and provide specific recommendations on how to fix the issue to become compliant.
    - ARC Toolkit: a set of accessibility tools that aids developers in identifying accessibility problems and features for WCAG 2.0, WCAG 2.1, EN 301 549, and Section 508.
    - CSS HTML Validator: offers one-click HTML, CSS, SEO, spelling, accessibility, mobility (Android/iOS), JavaScript linting, and link checking.
    - TAW: A online tool for determining the accessibility of a web. This tool analyzes the web site in accordance with W3C web accessibility guidelines and shows accessibilities issues.

- User testing:
  We will do 20%* of beta testing with people with disabilities.
  **(about 20% of the population has disability issues)*

KPIs and Metrics: Total Issues Found, Total Issues Resolved, General User testing feedback, TAW Score, Accessibility Index

Security Tests:

The entire premise of using a blockchain is to let everyone, usually people who do not trust each other, share valuable information in a secure manner. The blockchain is assumed to be secure because the records are secured through cryptography. All the participants have their own private keys assigned to each transaction they make and acts as a personal digital signature.

- Smart Contract Testing:

  As the smarts contracts (and for an extension, DAOs) are the operational core of our platform, we should have a testing approach to simulate all possible expected and unexpected conditions of the contract. Testing will look for all business logic combinations and appropriate execution of all the transactions (tokens, votes, issues, pools, etc.) in the context of a dynamically changing and expanding network. We will focus on these activities:

  - Smart contract deployment and configurations using custom build processes and scripts
  - Smart contract interface with the Blockchain, data, and rules processing
  - Performance and operational stability in the network post-deployment
  - Business logic as defined in the requirement and associated use cases

  We will use a wide range of standard and commonly used tools in the Blockchain ecosystem like:

  - **Ganache**: A world-class development environment, testing framework and asset pipeline for blockchains with built-in smart contracts compilation, linking, deployment, and binary management.
  - **Chai Assertion Library**: Chai is a BDD / TDD assertion library for Node and the browser that can be delightfully paired with any javascript testing framework.
  - **Remix-IDE**: Remix is a browser-based compiler and IDE that enables users to build Ethereum contracts with Solidity language and to debug transactions.
  - **Populus:** This framework has the testing functionality of Ethereum embedded in the form of a set of features for test contract deployment. It's developed around the py.test framework. Hence, it is relatively easy to implement.
  - **Truffle:** A lot of testing features, such as automated contract testing. The framework holds capabilities beyond just testing functionality within the Blockchain application.

- DAO testing:

  DAOs are central to the implementation of business rules and legal conditions; a vulnerable DAO in our platform could lead to an insecure system due to inherent immutability. DAO is a smart contract that positions itself as an organization, all processes of which are described by code working in a blockchain environment, while it is not a legal entity and is managed collectively by all its investors. Thus, there is a high risk of loss of data, disputes, and reputational damage. As DAO is a subset of smart contracts, we will use the same testing tools like smart contracts, plus some specific ones:

  - **Manticore:** is a symbolic execution tool for analysis of binaries and smart contracts. Manticore enables human-assisted analysis and the automatic detection of vulnerabilities.
  - **MythX**: a powerful security analysis service that finds Solidity vulnerabilities in your Ethereum smart contract code during your development life cycle.

- Integration Testing:

  Integration testing involves bringing up nodes locally and testing invariants about them by starting flows and inspecting their state.

  - **Corda**: Corda is a blockchain-inspired, open-source distributed ledger platform that does integration tests, writing flow tests and loading tests.

- Security Testing Report:

  Security validation at a transaction, block, and network level. We will cover:

  - Identity/Access testing for users and administrators
  - Data Integrity testing to assess the ease of modifying data and assess its impact on application behavior
  - Encryption security and safety standards to prevent any misuse and misappropriation

- Performance Testing:

  Performance testing verifies the performance and the latency within the Blockchain network. Performance testing in Blockchain includes identifying performance bottlenecks, defining the metrics for tuning the system and gauging the scalability of the application

- Security Audit:

  Last, but not least, we will contract an external security audit and due diligence. It will be run by an external professional agency and one of the most experienced teams in the Blockchain community, Consensys that already did a lot of public audits for top Blockchain projects.

  Tool: Consensys Diligence

Metrics & KPIs:

Passed Test Cases Percentage, Failed Test Cases Percentage, Fixed Bugs Percentage (bugs fixed / bugs reported), Accepted Bugs Percentage, Critical Bugs Percentage (critical bugs / total bugs reported), Average time for a development team to repair bugs, Number of tests run per time period, Test review efficiency, Number of bugs per test hour, Total Number of Bugs found, Total of issues reported, Total Performance Issues, Total Bugs found in Smart Contracts, Defect Severity Index, Test Coverage, Active Bugs, Number of Automated Tests, Bugs Fixed Per Day.

Product Tests:

The goal in this section of tests is to understand and improve the final product and make it more usable and attractive for the final users. Testing it as soon as possible (in the design/mockup/prototype part) is a key concept to iterate and improve the product in earlier stages to be able to present a final usable product.

Also, prepare the production product to be able to recollect a wide range of metrics to be able to have a clear vision of what's happening on it.

- Wizard Of Oz:

  We will use the Wizard of Oz methodology at the beginning of the platform and until we reach a critical amount of users, to manually review all the DAO's created by the users and we will search for legal problems, help with setting the organization up and give mentorship. When we have around +5K monthly users, we plan to launch the 'legal advisor & mentor' role within the platform, where users could use their expertise to assess the DAO creation and get rewards for this work.

- Mockups:

  We built wireframes & mockups from day-0 to start imagining and refining the platform interface and user experience. We work in an agile way: design, review, build, improve & iterate. With this methodology, we expect to deliver high-quality, easy-to-use layouts & pages.

  Tools*: MockFlow, InDesign*

- Live Product:

  One of the key points of the product will be the community, and we want to interact with them with real-time feedback, knowing which parts they love and which they don't like. This will be done with standard internet tools to track user clicks, user flows, funnels, user retention, time spent in the platform, heatmaps & more. This will help us to develop in a lean way, adapting and pivoting depending on our community needs.

  Metrics & KPIs: Visitors, App Downloads, Number of Created DAOs, Number of Pooled $, Number of Votings, Number of Issues, Number of Resolutions, Number of Members, Average Members per DAO, Average Pooled $ per DAO

  Tools: HotJar, Google Analytics, Survey Monkey, Intern tools (self-developed)

Marketing Tests:

A stunning product could become a total failure due to the lack of good distribution channels and marketing strategies. Communicate the right message, focusing on the right people who need it, requires hard researching and refining work. We will spread our word by different channels and each one requires a specific perspective but with a global vision and message. Our principal distribution channels testing approach would be:

- Landing Page:

  This will be the main entry point for our platform, and almost all other marketing channels (ads, email campaigns, search engine marketing, social networks, video campaigns, referrals, etc.) will point all the traffic towards this landing page.

  Having this in mind, it's critical to optimize and improve the landing page itself in different ways:

  - **A/B testing**: Testing different versions of our platform to different user groups (starting with the wireframing/prototyping stage). This will help us to redefine and improve our message & visuals from the beginning. Every time we deploy new features or made changes in copy text, layouts or images we will test its performance with A/B testing tools.

    Tools: Optimizely, Unbounce, Google Analytics

  - **Live Product**: In the same way we will have metrics and different tracking tools in the live product, we will have it too for the landing page (visitors, retention, time spent, rebounds, etc.).

    Tools: Google Analytics

  - **Heat Map**: to visually understand what the users "see" and which parts of the landing work and which don't.

    Tools: Hotjar

  Our final goal is to have a clear, concise action-oriented headline, explaining the product offer in a clear and easy-to-understand manner.

  Metrics & KPIs: Audience, User Retention, Avg. Session Length, TTFB, Error rate, Bounce Rate, Top pages, Conversion Rate, Bounce Rate, Click Path, Average Page Depth, New Visitors

- A/B Tests:

  We will run different A/B tests or split-tests to optimize the different campaigns in different channels like social network posts & messages, landing page (see above), video campaigns, copy messages, Ads, etc. This will be important to refine our message, make it clear and easy-to-understand to everybody.

  We will test subjective variables like:

  - **Social content:** Post's title, readability, description, and word count factor into how our audience engages with social content.
  - **Layout:** While these variables may vary by the distribution channel, we can still test the layout of posts or landing pages. For example, Facebook offers multiple ad formats, Instagram a more visual platform and with Twitter, we could target expert users.
  - **Imagery:** Images contribute – and deter – from engagement and conversion. Does a product image, or the application of a project shot perform better? Do promoted videos or gifs get more action than static images? Testing the media we use in a campaign can help us select the most impactful visuals.

- Video Campaign:

  Video is an exciting way to reach and engage with our audience and potential users. We will create a series of videos to explain, in different languages, how our platform works, the best use cases of using the DAO template creation, and other useful videos explaining key concepts of the platform as Voting Mechanism, Agreement, Disputes, Money Pooling & Interests, How to create a new community (DAO) and much more content.

  Another useful way to use video is to record how a task is performed in our platform, with screen-recording tools, in order to show how to interact in a tutorial way, explaining step-by-step different use cases of the functionality and revealing the best way to develop the full powers of our platform.

  We consider that good easy-to-understand communication could make the difference to increase confidence to non-expert users.

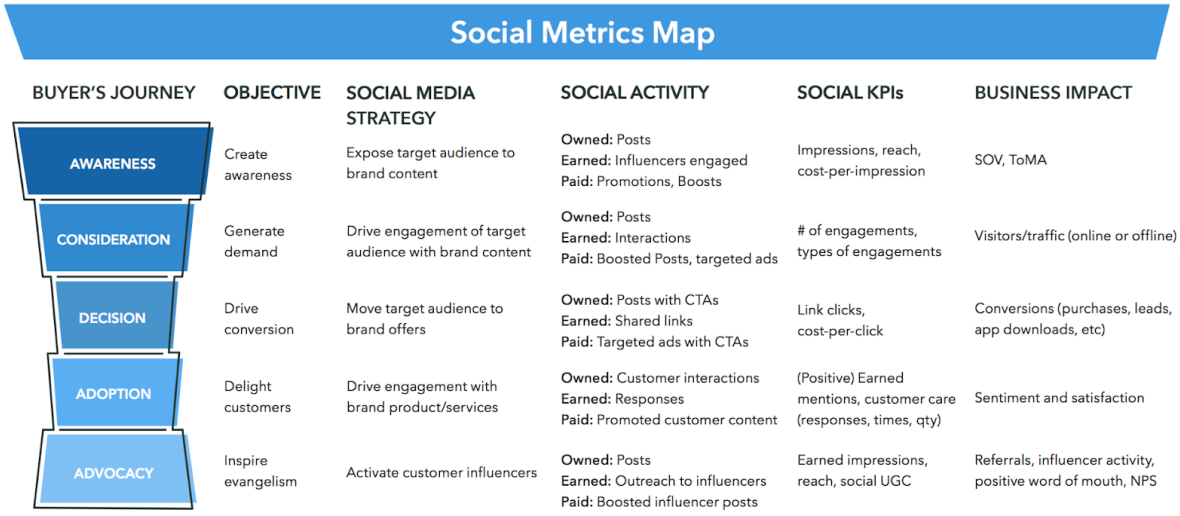
  Metrics & KPIs: Unique Views, Plays, Audience Size, Num. of interactions, Num. of shares, Mentions

  Tools: YouTube, RecordMyScreen, Adobe After Effects, Sony Vegas Pro, MMS

- Social:

  We will generate a content strategy for each principal social network: Facebook, Twitter, Instagram, Youtube & LinkedIn (considering TikTok). This doesn't mean exclusive content for each platform rather than specific strategies like content type and tone (text & images for Twitter, Facebook & Linkedin, images for Instagram & videos for Youtube) and iterates with the testing tools to detect which content is most engaging and drives more conversions.

  We will use a social metric map *(image 1)* to help us prioritize the most important aspects of our platform and measure our progress:



  Another key aspect to increase our reach is by doing strategic partnerships with influencers & other communities with similar interests: democratic organizations, human-centered organizations, circular economy, sustainability, co-operatives, NGOs, etc.

  Metrics & KPIs (per network): Number of Impressions, Reach, Nº of Engagements, Link clicks, Audience Growth, Targeted traffic to the Landing page, Mentions, Nº (and quality) of partnerships

  Tools: Facebook pixels, Google Analytics, Twitter Analytics, Instagram Analytics, Youtube analytics, Sprout.

- Publicity and (unconventional) PR:

  Publicity and paid ads are a keystone on the traditional marketing campaigns, but we will try to reduce the paid ads and convert it to more valuable assets for our community like Bug Bounty programs, Hackathons, Contests, Grants & Giveaways. In this way, we will be reducing the direct ads campaigns on Google, Facebook or

Twitter (to name a few) almost to zero and focus our ads on other platforms.

Our plan is to run either direct advertising (paid ads) allocating only 10-15% of the total PR expenses & indirect advertising campaigns (not direct paid need, that could go viral on local news, blogs, social networks, etc.) allocating 85-90% of the total PR expenses.

This indirect campaigns could be:

- Running regular hackathons to improve the DAOs templates & attract developers and users with prizes and recognition.
- Host regional contests around our platform to extend country-specific problems with DAOs. Either on-site with local sponsors & prizes or online ones.
- Social networks giveaways.
- Yearly contest for electing the best Year DAOs by categories
- Grants for the best communities build with Commünitäs

And once set up, we will promote them with local media, targeting blogs, email newsletters, social networks, and paid ads. Doing this way, we will either build a strong community around our platform, select and promote best uses cases, reward participants & developers that help improve our platform and retro-alimenting the Commünitäs ecosystem in a positive way, attracting new users with the best strategy: a sincere word-of-mouth recommendation.

Metrics & KPIs:

- <u>For Hackathons, Contests, Giveaways, Grants:</u>

  The number of run hackathons, Number of participating developers, Number of issues/projects developed, Number of regional contests, Total Number of Participants, Number of Voters in the Yearly Contest, Number of Grants, Total amount of $ per Grants, Reach, Mentions, Number of Sponsors, Total Amount given by Sponsors.

- <u>For Ads:</u>

  Conversion per Impression, Click Through Rate, Conversion per Impression, Click Through Rate, Return of Investment

Reference:

https://en.wikipedia.org/wiki/Web_analytics
https://en.wikipedia.org/wiki/A/B_testing
https://mijingo.com/blog/show/topic/web-performance-testing
https://www.dappros.com/201811/blockchain-developer-roadmap/
https://moz.com/learn/seo/social-media-measuring-testing-video
https://adparlor.com/blog/optimize-your-message-how-to-ab-test-on-social/
https://moz.com/learn/seo/ranking-visibility
https://www.lsb.com/blog/social-media-testing-5-steps/
https://www.guru99.com/software-testing-metrics-complete-tutorial.html
https://www.advancedwebranking.com/blog/step-by-step-test-your-paid-search-ads/
https://sproutsocial.com/insights/guides/social-media-testing/
https://www.thinksys.com/qa-testing/software-testing-metrics-kpis/
https://www.trufflesuite.com
https://en.wikipedia.org/wiki/Performance_indicator
https://web3js.readthedocs.io
https://www.w3.org/standards/webdesign/accessibility
https://www.w3.org/WAI/
https://www.w3.org/WAI/standards-guidelines/
https://diligence.consensys.net/
https://www.hotjar.com/
https://unbounce.com/
https://optimizely.com
https://mockflow.com/
https://www.w3.org/WAI/ER/tools/
https://www.softwaretestinghelp.com/how-perform-software-product-testing/
https://dev.to/smartym/how-to-test-ethereum-smart-contracts-audit-best-practices-53kg
https://www.danielefavi.com/blog/testing-a-smart-contract/
https://www.lsb.com/blog/social-media-testing-5-steps/
https://www.inflectra.com/ideas/topic/testing-methodologies.aspx
https://www.magicblockchainqa.com/our-services/
https://neilpatel.com/blog/beginners-guide-ab-testing-ppc/
https://sudonull.com/post/73628-Testing-Ethereum-smart-contracts-using-the-DAO-as-an-example
https://testguild.com/blockchain-testing-tools/
https://www.scnsoft.com/blog/tools-for-testing-blockchain-applications
https://www.a1qa.com/blog/review-of-tools-to-test-blockchain-based-apps/
https://en.wikipedia.org/wiki/Black-box_testing