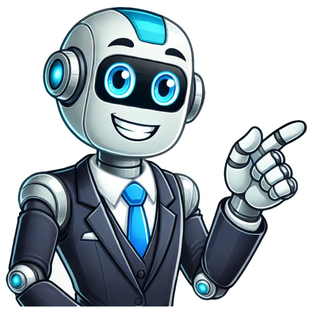
Click Here



Amazing support from humans to humans. During the 14-day trial period, you can invite your team and test the application in a production-like environment. If you have any questions, you can always contact our support team. We are here to help. Content In the software product development process, software testing plays an important role as it ensures and upholds the software product quality. The various phases of this software testing are Unit testing, Integration testing, System testing, User Acceptance testing and interestingly, the last phase of software testing, which is named UAT or User Acceptance Testing. It has a significant role in this entire automation software testing process. The last stage of the software testing process is known as UAT. This user acceptance testing process plays an important and crucial role as it validates whether all the business requirements have been fulfilled or not before releasing the actual product. Further, UAT testing can also be defined as the user methodology where the developed software is tested by the business user to validate if the software is working according to the specifications defined. This type of testing is also known as beta testing, application testing or more commonly end-user testing. The User Acceptance Testing process is taken up before planning to release the software into the market. This step ensures that the software is complete according to the functional specifications defined by the product owner, especially in e-commerce applications. It involves exposing the product or feature to real-life situations and confirming its performance from the viewpoint of end users. E-commerce User Acceptance Testing aims to verify that the system fulfils business requirements, operates as planned, and provides an excellent user experience. It is highly essential for the product owner to take up this process else businesses suffer from significant losses due to many post-release issues. Usually, this process is taken up as the last phase wherein intended users or rather the customer verify if the functionality as outlined in the user story or requirements documents exists. Specifically, the UAT process is taken up with a separate UAT test plan and in a separate testing environment with typically enabling production-like data set up to make the client/end-user clearly understand the developed software. Why UAT is essential for your business? UAT is an essential step to be taken up by the business owner / intended user / product-owner / client / stakeholder to have a first-hand understanding of software/product developed. The business user verifies and validates the software system developed before it is actually moved to the production environment. The various stakeholders involved in the UAT process include business analyst, QA lead or Test Manager, requirements specialist (if any), and the business or product owner. Primarily, UAT ensures if the developed system can effectively be used to support the business's day-to-day operations and works as per the user stories laid down. Typically, the product owner verifies if the solution works in full swing without any defects and confirms whether it meets their needs or not. If any bugs are identified, then the QA team should necessarily fix them up and re-test them. Hence, it is an essential step for software product development as businesses tend to incur significant losses if post-production issues occur. Who Performs UAT? – Client – End users Challenges of User Acceptance Testing (UAT) User Acceptance Testing (UAT) is a critical phase in the software development lifecycle where end users test the system to ensure it meets their requirement and function as intended. While UAT is important for validating the system's usability, several challenges can be associated with it. Some of the common challenges of UAT include: Requirement ambiguity: One of the significant challenges is the ambiguity or incomplete understanding of user requirements. If the requirements are not well-defined or documented, it becomes challenging to perform effective UAT as users may have different interpretations or expectations. Limited user availability: Getting user involvement and availability during UAT can be challenging. Users are often busy with their regular responsibilities and may not have sufficient time to dedicate to testing. This can lead to delays in the UAT process or inadequate coverage of test scenarios. Limited Scope: It refers to the uncontrolled expansion of project requirements or changes during development. If there are frequent changes to the system functionality, it can be challenging to keep up with the changes and align UAT efforts accordingly. This can result in delays and additional efforts required for testing. Test environment setup: Setting up a realistic test environment that mirrors the production environment can be complex. It may involve configuring multiple systems, databases, network setups, and third-party integrations. If the test environment is not accurately replicated, it can affect the validity and reliability of UAT results. Test data availability: UAT requires representative and meaningful test data that simulate real-world scenarios. Acquiring or generating appropriate test data covering various use and edge cases can be challenging. Inadequate or unrealistic test data can lead to incomplete testing and may not uncover potential issues. Communication and coordination: Effective communication and coordination between developers, testers, and users are crucial for successful UAT. It can be challenging to ensure clear and timely communication, especially if there are multiple stakeholders involved. Miscommunication or lack of collaboration can lead to misunderstandings, delays, and inefficient testing. Defect resolution delays: When users identify defects or issues during UAT methodology, the resolution process becomes critical. Delays in resolving reported defects can impact the overall testing timeline and user satisfaction. Timely communication, prioritization, and coordination between the development and testing teams are essential to address issues promptly. User expertise and training: Users participating in UAT may have varying levels of technical expertise and may require training or guidance to effectively perform testing activities. Ensuring that users understand the testing process, have access to necessary documentation, and receive adequate training can be challenging, especially when dealing with a large user base. Resistance to change: UAT can sometimes highlight gaps or inconsistencies in the system, which may require changes or modifications. Resistance to change from end users or stakeholders can hinder the smooth execution of UAT and the subsequent implementation of necessary improvements. What are the benefits of UAT for Businesses? • Validates whether business requirements are met as defined in the user stories • UAT helps in identifying the bugs that are left during the integration testing • Reduces the risk of finding defects post-production • Helps to identify the behaviour of the developed application as visualized by the product owner or the end-user who actually uses the software • UAT ensures the developed application functions well and meets the defined business conditions as defined in the requirements document Pre-requisites Businesses Must Know Before Performing UAT Testing Following are the entry criteria for User Acceptance Testing: • Check if the application code has been fully developed • Make sure if unit testing, integration testing, system testing, and acceptance testing has all been done by the QA teams • Check and verify if regression testing has been completed and no major defects have been identified • Check whether all reported defects have been fixed and re-tested • Ensure Traceability matrix should also be made available • Ensure an effective UAT environment must be made ready • Ensure a Sign-off mail from the system testing team is received • Ensure to have a UAT checklist before starting the UAT process User Acceptance Testing Phases 1. Planning Phase of UAT: Assigning a UAT Test Manager to oversee the entire process of UAT is essentially done in this stage. Proper planning and execution strategy are outlined here. Identification of critical resources is done and preparation of a critical resource plan is done. 2. Preparation of UAT Test Data, Scenarios and Test Environment: UAT readiness is ensured in this phase as the UAT test environment is set up, preparation of test management plan along with test data, interfaces, data, authorization along with scenario readiness is done here. 3. UAT Test Scheduling and Management: Proper action plans with UAT priorities are done in this phase. A triage process is kept in place to prioritize the assessments of defects blocking if any. An effective mechanism to track test scenarios and test scripts based on the requirements defined is taken up. 4. UAT Testing Execution and Defect Management: This is an important phase and proper identification of priority defects is taken up and more focus is placed on performing root cause analysis assessments. A trial run of UAT processes is done to validate execution and defects assignment and assessment is taken up for proper and quick resolution. 5. UAT, Sign-off, and Reporting: In this final phase of UAT, accurate defect and testing status reports and defect reporting is generated from the test management system. Finally, a sign-off when all bugs have been fixed indicates the acceptance of the software. This final phase ensures and validates that the application developed meets the user requirements and is ready to be moved to production. What are UAT best practices? Following are the 5 UAT best practices: 1. Find Users: User identification is the first step. Selecting a group of users (who are interested in the product) is essential for conducting UAT. These users are none other than the end-users of the product. These users can be from the company or from outside or both. 2. Document test cases: There has to be a methodical plan for UAT testers to perform UAT productively. As these testers will be going through all the features of the product, their feedback in terms of how they want the product to function should be documented. 3. Prepare the environment: A test environment is crucial for the success of UAT. Those users who've been selected to perform UAT must have the credentials and data to carry out testing effectively. 4. Remote testing counts: As in certain scenarios, if the users are located at remote locations, it would be wise to schedule a common date and time to carry out UAT. Prior planning in terms of the technicalities involved in performing the testing activity should be clearly informed so that the users are well aware of all the implications. 5. Prioritize to detect and solve bugs: When the test goes live, there will be issues pertaining to addressing bugs. Let the user track and properly list all the bugs that have been encountered. The concerned team responsible for fixing bugs must prioritize the bugs and then solve it accordingly in a systematic manner. Types of UAT Businesses Should Know Alpha and Beta Testing: Alpha and Beta testing are the approaches in UAT that are practiced before releasing the software to the customers. Alpha testing is practiced in the early development phase with the help of internal teams. Before releasing the software to the customers, the potential user groups conduct this testing in the development environment. The feedback received from the alpha testers helps the development teams to fix the bugs and improve the overall quality of the product. Beta Testing takes place in the customer environment by involving a group of customers for testing. The feedback from this testing group helps in enhancing the quality of the software. Both these testing processes are done before the actual application or software is released to production. Regulation Acceptance Testing (RAT): RAT, also known as Compliance Acceptance testing, is the method to validate if the developed application complies with the rules and regulations as set by the governing agencies at the governmental level. This is a very important testing practice as the application is tested for its compliance towards a set of guidelines given by the prevailing authority. Contract Acceptance Testing (CAT): In this type of UAT approach, the software developed is tested against certain critical criteria. These criteria and specifications have been earlier defined and mentioned while the contract is made. Business Acceptance Testing (BAT): BAT is a user acceptance testing type that helps to validate whether the developed software satisfies the requirements and specifications that are set by the business. Operational Acceptance Testing (OAT): OAT is also known as Production Acceptance Testing is one of the UAT testing types that helps in assuring whether there is a proper workflow for the software, i.e. its reliability, compatibility, stability, etc. Black Box Testing: In this type of UAT, the testing teams are allowed to analyze a few functionalities of the application without knowing the internal code structure. Though black box testing is categorized under functional testing, but it is a part of UAT as it uses the same principles that the user is unaware of the codebase. This testing ensures that the software meets the business requirements. Conclusion Businesses are in need of quality software along with faster releases today to beat the competition. Software testing is an essential component of the software product development and essentially User Acceptance testing should be taken up by the business or the product owner. User Acceptance Testing helps in validating whether the developed software is functioning according to the requirements specified and if it meets all the user stories as defined. Leverage next-gen independent software testing service providers that ensure the quality software is delivered with careful UAT test plans and guides businesses through to arrive at an effective User Acceptance Testing process. With this UAT approach, businesses can be rest assured of a product that effectively meets their requirements and is all ready to be moved to production. Talk to our experts for more information on UAT testing FREE $0 ₹0 Per user per month 5 user maximum 5 projects 1K test cases 1 GB file space 5 custom fields 5 field sets 100 runs/project Custom run statuses Webhooks 0 API calls/hour Audit Trail Two-Factor Authentication Single Sign-On Team $9 ₹450+ GST Per user per month 5 user minimum 50 projects 50K test cases 5 GB/user file space 15 custom fields 5 field sets 250 runs/project Custom run statuses 2 Webhooks 1200 API calls/hour Audit Trail Two-Factor Authentication Single Sign-On Popular Business $15 ₹750+ GST Per user per month 5 user minimum 100 projects 100K test cases 15 GB/user file space 25 custom fields 10 field sets 500 runs/project Custom run statuses 3 Webhooks 2400 API calls/hour Audit Trail Two-Factor Authentication Single Sign-On Enterprise $29 ₹1,450+ GST Per user per month 5 user minimum 250 projects 250K test cases 30 GB/user file space 50 custom fields 15 field sets 1000 runs/project Custom run statuses 5 Webhooks 4800 API calls/hour Audit Trail Two-Factor Authentication Single Sign-On ServicesTravel ExpertiseInsightsCompanyTo find bugs and logical mistakes in the software, it is essential to engage in quality assurance activities. However, QA testing won't tell you if the product is aligned with business objectives and can perform required tasks in real-world scenarios. So, to verify the development team is building the right solution for the actual end users, conducting user acceptance testing is vital.User acceptance testing explainedUser acceptance testing (UAT) checks whether a product is right for end users. The word testing makes many people think it's the same as QA, but that's not exactly correct. Let's see what these and other related terms mean and how they are different.In the testing process, it's important to distinguish between validation and verification. Verification refers to general QA processes aimed at checking technical aspects of a product to ensure it actually works. It is carried out by QA engineers and testers throughout the development process. Their goal is to identify and fix bugs, test functionality, check performance, and validate security and integration. QA is based on technical specifications, and defects are reported to developers for resolution. The QA team decides when the software is stable enough to move forward. Read more about Quality Assurance in our whitepaper.Validation is a different name for UAT, and is conducted to confirm that the product corresponds with business requirements and meets the needs of the end user. It is performed by end users, business stakeholders, or product owners to ensure the software meets real-world business needs. Instead of focusing on technical correctness, UAT validates whether the software works as expected in real-life scenarios. If any issues arise, they are reported to business decision-makers, who determine whether the product is ready for release.Validation and verification activities in terms of overall product testingIn summary, QA ensures the product is functional and bug-free, while UAT ensures it is usable and meets business objectives. Both are necessary to deliver high-quality software that satisfies users.SIT (system integration testing) and UAT both happen late in software testing, but serve different purposes and are performed by different teams.SIT happens first to ensure that different system components or modules work together as expected. Developers or QA testers conduct it to identify defects in data flows and interactions between individual elements or modules — APIs, databases, frontend, etc. For example, SIT can include testing if a payment gateway integrates correctly with an e-commerce website. UAT is conducted after SIT to ensure the software meets business requirements and is ready for end-users.Some SIT and UAT test cases might look similar (e.g., verifying a feature works), but their perspective differs. For example, in SIT, testers check if a login interface properly communicates with a database; In UAT, users check if the login process follows the expected flow.Both testing stages use environments similar to production, so people assume they serve the same purpose. However, SIT is more controlled, while UAT mimics real-world usage.For more info on how system integration works, check out our article. the actual users of an existing product, users of a previous version of a product, stakeholders involved in the development of the product, and/or business analysts as end-user specialists. This enables the development team to fix most of the usability problems, bugs, and unexpected issues concerning functionality, system design, business requirements, etc.The main purpose of acceptance testing is to validate that the product corresponds with the user needs (defined at the product discovery stage) and is ready for launch. According to an Origsoft survey on UAT usage, over 75 percent of respondents said they conduct multiple cycles of end-user testing with 57 percent claiming the poor quality of the product as a cause.So, here are the main reasons why UAT is important and should be a part of your development.Ensure correspondence with business requirements. As we already mentioned, UAT is done to verify that the product operates in the real-world circumstances as required and allows end users to solve targeted problems. If you skip UAT, you might miss out on some critical flaws or system malfunctions that will inevitably cause user dissatisfaction.Adjust initial requirements. Sometimes, as end users test the product, they can come up with valuable thoughts on how to improve the tested software. Getting such feedback will allow you to adjust your requirements and get a better result for your customers.Avoid losses. First, UAT will allow your team to improve the product cheaper and much more easily (that mostly concerns the Agile model though). Second, we all know stories about product failures because of poor functionality and usability. UAT provides you with real-world user feedback and makes it far less likely to have losses caused by an unsuccessful product launch.In any case, UAT requires organization and preparation work to make it effective. If you want to ensure your product's validity, consider the following steps in conducting user acceptance testing.Analyzing product requirements is the first step of UAT planning. The primary source of input information would be the software requirements specification as it includes the complete scope of business and functional requirements.Business requirements are high-level objectives of your organization that communicate business needs. For example,"customers should be able to use multiple payment methods."Learn more about documenting business requirementsFunctional requirements bridge a technical solution with the business requirements. So, a functional requirement might sound like "implement PayPal, Visa, Mastercard, and Payoneer acceptance at checkout."The overview of these requirements will tell you exactly what you should test, whether the implemented solutions work for the users and solve problems for the business. Functional requirements can be translated into test cases, considering the success criteria of business requirements. And that will help you form an overall testing strategy. Consider engaging your business analysts, QA engineers, or product owners for requirement analysis.Acceptance testing may take place at different stages of the project, depending on the methodology you are using, but typically it's conducted at the end of the development cycle before release. As two most popular project management methodologies in software development are Waterfall and Agile, we will look at the user acceptance testing process within those models.A Waterfall model is a traditional project management methodology based on a step-by-step development of the product.The stages don't intersect, meaning there is no simultaneous designing and design testing, or development and testing. The whole process is strictly documented and intended to deliver a fully functional application at the end of development without iterations.User acceptance stage within the Waterfall modelUser acceptance testing in Waterfall takes place at the final stage of development, right before the launch.It can be conducted only after the system is considered code and function ready, after achieving the following benchmarks. Product business requirements have been met. The code base is finished. QA activities (system, integration, unit testing) have been completed. Bugs revealed during the QA stage have been fixed. Minor visual issues are in an acceptable range. User acceptance environment (UAT manager, tools for testing, test scenarios, etc.) is created. In the Waterfall model, user acceptance testing is the definitive point that demonstrates software readiness. If a product meets user acceptance criteria, it means the product is ready for production. UAT activities, in that case, are for completing the system check, including its functionality, usability, and bugs. But still, the primary goal is to ensure that the product corresponds with the initial requirements and end-user needs.Agile project management isn't as straightforward as Waterfall. It's based on iterating each development stage until the product reaches the required quality and functionality. Iterations allow for dynamic change in requirements, as Agile doesn't focus on creating much documentation. And that, in turn, enables engineers to quickly respond to customer feedback.User acceptance testing in Agile modelThe image shows the Agile product development cycle with iterations. The main difference between UAT in Waterfall and in Agile is that UAT is carried out multiple times (often within each stage) and its results may impact the initial requirements, as it delivers instant feedback on what works the best.The checkpoints for starting end-user testing in an Agile project are formed business requirements, UX/system documentation, testing material (interactive mock-ups, high-fidelity prototypes, demos), and user acceptance environment. In Agile, UAT is an integral part of overall testing activities, so it may take different forms and use different tools. For example, these can be tests on functional and non-functional requirements or early-stage testing to validate assumptions made during the planning stage. At the end of each iteration, acceptance testing produces deliverables that are used to modify requirements, system architecture, UX style guides, etc.Before starting user acceptance testing, it's essential to establish clear documentation to ensure alignment between all stakeholders. Well-structured UAT documents provide a roadmap for the testing process, defining objectives, responsibilities, and expectations.Below are the key documents that should be created for UAT.UAT plan. Creating a UAT test plan will help you keep everybody aligned with the same objectives and vision. Being the main document, it includes all the information concerning what will be tested, by whom, and how. To cover all the organizational and processual aspects of UAT, you have to detail the testing strategy and entry/exit criteria.End-user testing strategy. Your testing strategy should cover product description, testing objectives, testing scope, standards, testing types, testers/roles, process curators (managers), reviewers, reporting standards, and outcomes. Entry criteria. These are the conditions that establish the software readiness to be tested. They are set at the earliest stage of planning by the development team, QA, business analysts, and stakeholders.Exit or acceptance criteria. These are the conditions dictating that the software is valid for users. Matching acceptance criteria would be the final stage of your UAT.Acceptance criteria explainedTest scenarios and test scripts. Test scenarios are hypothetical situations that users may encounter when interacting with your product, while test scripts are formal, detailed descriptions of the actions needed to perform UAT. Their aim is to guide your testers through possible system usage problems.Basically, a test scenario should convey a simple idea of what will be tested. An example of a scenario is "check shopping cart functionality." Each user scenario is connected with one or two requirements or user stories. They are written to validate that the system is usable, checking the end-to-end operations with real data.Test scripts often contain a sequence of specific steps to take. They give exact instructions on what testers should do, as well as describe expected results. For example, a script can suggest clicking the "Ok" button that has to be followed by a confirmation popup.To write good test scripts and scenarios for user acceptance testing, consider involving end users in approval to include all the possible use cases, both common and uncommon. Also, consider writing them in plain language, avoiding complicated phrasing or overly techy explanations.Test cases. A test case is a set of specific actions that are taken to test and verify a particular system behavior, feature, or functionality. Test cases are more detailed units that have to correspond with all the test scenarios. Most often you will rely on your user stories and business use cases to write efficient test cases. Examples are: Check an unregistered user adding the product to the shopping cart. Check shopping cart filtering. Check the "continue shopping" button. Test cases are efficient when there is a clear purpose stated, and users are able to understand what they should do to complete it. The user guide to a test case may look like this: Open the application. Add any product to a shopping cart. Authentication is not needed. Proceed to the shopping cart. You may also include expected results in the test case so that the user is aware of what is going to happen:The product will appear in a shopping cart.The system will ask you to authorize as a registered user.As we mentioned earlier, testers can be recruited from your existing user base. Depending on the project specifics, those can be subject matter experts, real-world users of the product, stakeholders, business analysts, or a product owner. You can also use crowd-sourcing platforms to search for testers or hire a freelance specialist.Consider creating a social media message or even a landing page to attract an audience. Your potential testers should not necessarily be tech-savvy or familiar with software testing processes. However, the ones who already have or will use your product (or maybe a similar one) will be good candidates for your UAT since in that case you can avoid deep onboarding and QA team involvement.There are specific instruments on the market that are designed for end-user testing. Most popular tools offer testing management functionality like reporting, task overviews, and testing documentation templates. Here are some examples of software that can be used to support your UAT activities.Testing dashboard in SpiraTestUsersnap is a popular platform that allows users to mark bugs right on the screen, leave comments and suggestions, and share feedback. There are a lot of similar instruments such as Userback and UserTesting.FitNesse is an open source, wiki-powered framework for acceptance test automation. It allows all stakeholders to easily create, edit, and run tests, leaving early feedback. Users enter specially formatted inputs to automatically generate tests that are immediately executed by the system. Then the output is returned and highlighted depending on whether it matches the expected result or not. This collaboration platform has a mild learning curve and is popular among Agile teams.Bugwolf is another instrument for conducting UAT. Besides testing environment and bug reporting, it offers gamification and competition features to motivate and engage users. You will also find built-in payment options if you are going to conduct end-user testing online.Well-known project management tools like Jira or Trello also have functionality for performing UAT.To get the most out of end-user testing, start with the training. Your testers and UAT manager are responsible for that. Consider structuring your training process to include the following aspects.Introduce users to the testing process, its objectives, and tools.Provide them with reporting standards and guidelines.Ensure users understand test cases properly, providing support if needed.Give them access to the testing environment.Most often end-user testing can be done on the user's side and online, meaning you won't have to supply your testers with the hardware. More complicated projects or confidential data may require gathering a dedicated team of user testers at your office. It's also important to appoint a manager who will provide documentation, tools, and support.Once you have your test scenarios and test cases, you are good to go with the tests. To support your end users during the process and get the required results, ensure that users have a clear understanding of what actions each test case requires. Keep in mind that theyare not professional testers. During the test, provide real or close to real data to the users, avoiding sample content or dummy buttons. Any misconception may get them stuck at the test case.Another important aspect is having your developers ready to fix anything that goes wrong. Your testing environment can shut down or there can be errors preventing users from testing. They should be able to access required functionality at each stage of testing, whether it's an interactive design or a functional app, to perform each test case included in the test plan.During your UAT activities, you will get tons of data from testers. Your QA team will have to analyze it. The data is collected via user reports submitted manually or via a specific tool. Additionally, you can conduct interviews with separate users to get more insight into the test cases they performed and what they think of them.To evaluate system readiness, consider measuring the percentage of tests passed/failed/fixed.Test tracking dashboard in PanayaThere are also a few more points that should be considered.System stability. Stability can be determined by the number of unexpected errors met during the UAT.Testing coverage. Coverage is measured by the number of test scenarios/cases written and their ratio to the overall finished tests. You may also match your UAT testing results with the customer journey map to understand which part of functionality was left unchecked.System usability. This can be calculated by the number of tests not passed because the user didn't find a way to do it. But the overall UX is tested during usability testing, which is conducted as a separate activity.Contract/requirement compliance. Requirement compliance is checked after all the end-user tests are finished. It ensures that the software corresponds to the initial requirements/contract scope, even after changes brought by user acceptance.After executing UAT, all defects must be documented with relevant comments and passed to the development team. They, in turn, must make adjustments to the code to address the issues revealed by UAT.Once you fix the bugs, retest them to make sure everything is working properly. When the acceptance criteria are reached and approved by reviewers, the final acceptance decision is made about production readiness of the product.As we mentioned earlier, UAT testing is different from other QA activities because it's performed not only by tech specialists. It's important to engage actual end users in this process. Involving QA professionals and business analysts is also necessary, as is close collaboration with the project manager and development team.The responsibilities of the UAT team can differ depending on the company and project needs, but here is an example of the role distribution you can consider.Business program manager. This is the person that coordinates and oversees the entire project, aligning it with business objectives. Before the UAT stage, the program manager should generate a program delivery plan and business requirements document to support testing activities. He/she is also responsible for reviewing and approving the test plan and test strategy.During UAT, the program manager monitors test activities execution and ensures completion on schedule and budget. Afterwards, he/she reviews the test report and decides on deployment to production.UAT test lead/manager. The test lead's responsibility is to accurately plan and organize UAT. For that, close cooperation with the project manager is required.The test lead gathers and analyzes all the business and functional requirements, which are then used to develop the necessary documentation, i.e., test strategy, test plan, test scenarios, etc. Additionally, at the preparation stage, he/she works with the test team, assigning test scenarios to team members and organizing training to make sure testers understand the UAT procedure. The test lead also prepares and manages necessary resources and loads essential test data in test tools.Throughout the UAT, the lead coordinates the execution of test cases, making sure all test results are documented. He/she also tracks test progress, collects metrics, and creates/maintains a test report.UAT test team members. The main task of the test team is to execute tests in accordance with the provided schedule and instructions. Testers should create test logs and report on defects and incidents. They also typically participate in retesting activities (if needed).The project manager, as the person responsible for successful project completion, has to monitor testing activities, provide organizational support, and report on progress. He/she would also act as a mediator between the testing team, developers, customers, and any other possible stakeholders.Summing up the UAT guidelines we presented above, we've developed a checklist to help you organize your testing activities and not miss out on anything important.Initiating the UAT project.Verify with your development team that all product components are ready for testing. Document any issues that could not be addressed before the UAT.Identify key stakeholders.Choose a team leader responsible for the project, including paperwork.Discuss and agree on the project structure, UAT team, and UAT documentation.Thoroughly discuss testing procedures and create an initial UAT plan.Create your UAT team and make sure you have testers from each market segment and/or each group of stakeholders. Be certain all participation-related documentation is complete and signed (nondisclosure, participation agreement, etc.).Communicate the testing strategy and schedule to the team. Make sure every member understands roles, procedures, and responsibilities.Make sure all business requirements are captured and communicated to the UAT team.Discuss and agree on entry and exit criteria.Prepare all the business documentation: test plan, test scenarios, test cases, etc.Communicate business objectives and acceptance/exit criteria of the system.Agree on reporting standards.Conduct the needed training on the system and auxiliary tools. Make sure testers understand how to report incidents.Gather and prepare all the necessary resources for UAT activities. Book space if needed.Prepare and test the environment, test management tools, devices, servers, feedback channels, issue tracking, content delivery, etc.Make sure you have all the logins, security access has been set up, and test data has been loaded. Monitor how procedures are carried out and make sure reports are submitted timely and accurately. Create and maintain a test summary report. Analyze output information by measuring the percentage of tests that passed/failed as well as categorizing defects by severity.Identify status against acceptance criteria.Prepare a final UAT report and present it to stakeholders together with the estimated time and effort required to meet acceptance criteria and recommendations for release.Testing procedures might differ from company to company. Here are a couple of other downloadable UAT checklists that might fit your needs as well: Checklist 1, Checklist 2.We've mentioned a few important documents that have to be created for proper UAT planning and execution. There are different ways to write them, but we offer some templates that may come in handy.A UAT test plan outlines the overall strategy, scope, objectives, and roles for user acceptance testing.Purpose of the UATScope (what features/processes are being tested)Assumptions and risks2. Roles and ResponsibilitiesWho will be involved (business users, testers, developers)Responsibilities of each teamEntry and exit criteriaTest scenarios and test casesAcceptance criteria4. Test Environment & DataDescription of the UAT environmentTest data setup5. Test Execution & ReportingHow test results will be loggedDefect tracking processWho approves UAT completionAcceptance or rejection criteriaLuckily, you don't have to create your test strategy and plan from scratch. There are plenty of templates available that you can download and customize to suit your test project.A test scenario is a broad test case that describes a feature or function to be tested.A UAT test case document outlines specific test scenarios, steps, expected results, and actual outcomes.Here are some sources where you can download A test report summarizes test execution, defects found, and whether the software is ready for release. Below are templates and examples for creating structured test reports for UAT, system testing, or regression testing.Project Name:Test Report Date:Tested By:Test Environment:Testing Type: (UAT, Functional, Regression, etc.)Total test cases: 50Passed: 40Failed: 5Blocked: 5In progress: 2Defect ID: BUG-101Module: LoginSeverity: HighStatus: OpenDescription: Users unable to log in with the correct credentials4. Test Findings & RecommendationsMajor issues that need fixing before release.Any recommendations (e.g., retesting required, additional UAT needed).Approved By:Date:Decision: (Go/No-Go)You can get more report templates here:Just like at any other stage of the development cycle, certain challenges might arise during UAT activities. Here are some of them – together with ideas on how to be prepared:poor planning – sufficient time has to be scheduled for UAT, otherwise the results will be incomplete;wrong testing environment – for UAT, the environment should be different from the one used for functional testing;inappropriate tester selection – testers have to be chosen from the corresponding consumer group which is the main target audience of the product;lack of training – testers have to be well-trained prior to conducting UAT, otherwise it won't be effective; andinefficient communication – a seamless communication process between teams has to be set up to avoid gaps, misunderstandings, and delays.Keeping these challenges and our guidelines in mind, you'll be able to conduct efficient and productive user acceptance testing.

User Acceptance Testing (UAT), or application testing, is the final stage of any software development or change request lifecycle before go-live. UAT meaning the final stage of any development process to determine that the software does what it was designed to do in real-world situations. Actual users test the software to determine if it does what it was designed to do in real-world situations, validating changes made and assessing adherence to their organization's business requirements. The main purpose of acceptance testing is to validate end-to-end business flow. The definition and purpose of User Acceptance Testing (UAT). The detailed steps involved in performing UAT in software. Prerequisites necessary for conducting successful UAT. Common challenges faced during UAT and how to overcome them. Best practices for managing the UAT process effectively. The importance of thorough documentation during UAT. How to document UAT test case outcomes. Strategies to enhance the effectiveness of your UAT process. How Panaya can simplify and accelerate UAT testing for all stakeholders. Performing User Acceptance Testing (UAT) involves several steps that ensure that a change, update, or new feature meets the requirements of the end-user. The UAT testing process begins with preparing test scenarios and cases and finding suitable participants. The testing takes place in a separate testing environment. The first step in UAT is to prepare test scenarios and cases and identify suitable participants. The testing takes place in a separate testing environment and may span multiple sessions over several days. The process starts with instructing participants on how to correctly perform the tasks, filling in the test scenario forms, and providing feedback. End-users can then complete the test cases while company employees monitor progress and assist with any questions or issues that may arise during the testing process. Before performing a User Acceptance Testing (UAT), certain prerequisites must be met. These include: Business requirements provided to the testing team. Completed system, integration, and unit testing with no high or medium defects or showstoppers Regression testing completed not to affect the software. UAT environment available and ready User Acceptance Testing (UAT) presents several challenges that organizations must address to achieve a successful testing outcome. One significant challenge is the potential damage to reputation caused by rework and retesting. If organizations continue to rely on outdated tools like Excel or traditional test management tools, it can lead to difficulties coordinating with business users and lack of visibility into the testing status. Besides, coordinating with globally dispersed business users can become costly and time-consuming. Other common UAT challenges include engaging non-technical business users, facilitating collaboration between users, reducing idle time, improving visibility and control over the testing process, and ensuring high-quality test evidence without disrupting user productivity, increasing location costs, or interfering with business users' availability. The key to successful UAT is adopting industry best practices, including five steps that take you through the process from start to finish. 1. Knowledge Gathering for Test Planning Always begin by gathering the information required to create a comprehensive test. Your list of questions for the relevant stakeholders must include: Which business processes should be tested? What sequence of actions must be taken for a representative test? What are the guidelines for selecting test data? What are the intended results of the changes made? Which UAT team is responsible for testing? Generally, the entire process requires a significant degree of collaboration between the integration manager, the different functional leads, and the relevant business process owners. 2. UAT Scoping Not all business processes must be tested. Some can be safely ignored. It would be best if you never began UAT until you've defined the scope of your project. You'll find that it tends to balloon pretty quickly. Unless you scope in advance, it can become challenging to decide on the fly what is critical for the success of your test. 3. UAT Design Once you understand the scope of your UAT test, you can move on to design. This includes mapping and assigning different steps to various business users and setting a timeline. As time passes and you have more use cases to reference, this step will become much more manageable. 4. UAT Execution With your UAT process clearly defined, you can now begin testing, address any defects and decide if you should move ahead to production or not. To make this step optimally efficient, you'll need flawless communication and balance between testers and developers, focusing on documentation (see below for a deeper dive into this issue), progress reporting, and defect management. 5. Business Objective Confirmation Once execution is over, and as many defects as possible are resolved, it is time to sign off on UAT and go live. The sign-off approval indicates that the change meets business requirements and is ready for deployment. Documentation of your UAT testing strategy and the overall plan is indispensable to the outcome of your current and future tests. This documentation should include information about out-of-scope situations that are worth testing, the expectations for the test, general agreements about the standards for passing, how to carry out the test, the owners and participants involved, the scope of work, and the venue used. Additionally, it's important to note any successful past use cases, including details about the test structures, management, and outcomes. This documentation will provide a clear and comprehensive guide for the entire testing process. Document your tests and their results with traceable and annotated records that are easy to access and use. (But please don't mistake Excel sheets for 'easy to use'.) Here are some examples of the kinds of data to include in your outcome documentation: Acceptance criteria Business impact (e.g., high, medium, low) Business requirement Comments Date executed Expected outcomes Name of tester Pass/Fail Test case name and number Test case steps defined Your testers should independently record their own UAT results, which should be reviewed daily. In this way, issues can be identified early and addressed immediately. With the right agile UAT tools in place, you can tackle those challenges and take the best practices we talked about to the next level. You'll be reducing the time and effort needed for UAT processes by up to 50%. Engaging both your functional and business users on a standardized platform from the start is key to ensuring tests reflect actual end-to-end business processes. The right solution will offer collaborative technologies to coordinate among cross-functional, globally-dispersed users, and will be intuitive enough to ensure business users are comfortable with the process. When it comes to scoping your project, you can't get very far without input from your business users. Yet getting them to list all the important information you need in spreadsheets can be incredibly exhausting. The right test management solution would be able to guide users through this process, intuitively. Moreover, instead of having to rescope each project from scratch, test plans can be repurposed so users can get started immediately. Copying and pasting screenshots of test results into Word or Excel is very time-consuming and prone to human error. Optimize your UAT testing with automated documentation, workflow, and defect management. The right tool will help you with exploratory testing and be able to document tests using a recorder for playback as needed, accelerating the process and reducing the back-and-forth between the software development and testing teams. When you start off with a business-process-centric approach, it's much easier to track processes throughout the test lifecycle. Instead of relying on unmanageable and unreliable Excel sheets, leverage real-time dashboards to help you track multiple test cycles at both the test and business process level. You'll be able to monitor defects and manage overdue tests with built-in notifications to proactively reassign tests or send reminders to relevant stakeholders. UAT workflows often feel like running a relay race blindfolded to your key users. There are so many dependencies they are simply unaware of as they wait their turn in a waterfall-type workflow. This is anything but agile UAT. Instead, you can relieve dependency bottlenecks with embedded workflow automation features – even in a multi-step, multi-tester business process. Notifications, for example, can let a user know when it's their turn to test within the business process (a 'Time to Test' alert) or when a defect is resolved and ready for retesting (a 'Retest' notification), and a 'Close' notification informs developers of test or retest success. Globally dispersed key users are bound to have time-zone and communication issues that can make their whole testing experience even more unpleasant than it usually is. The right defect management tool can sidestep these problems and reduce the time wasted on ineffective communication between testing and development teams, automatically alerting developers to errors during testing and attaching the steps that produced them. When a defect is found, all other tests affected by it can be automatically identified, and testers can be warned or blocked from proceeding until the defect is resolved. Does all of this sound complicated? It doesn't have to be. A smart test management solution will help simplify your UAT cycles. Panaya Test Dynamix, our top-ranked end-to-end test management platform, includes features like test notifications, collaborative communications, automated documentation, and simple defect reporting. It allows end-users to quickly complete their part in the UAT cycle and return to business. The result is greater adoption, better ROI, fewer bottlenecks, real-time visibility, and zero risk at go-live. As an end-to-end testing solution that mirrors actual business processes, Panaya Test Dynamix provides those benefits and more, streamlining UAT and accelerating business process testing by 85%. User acceptance testing need no longer be a battle. Download the eBook How to Simplify UAT Testing and learn how to: Automate more elements in your user acceptance testing Incentivize key testers with ease-of-use Gain business users' confidence and promote adoption "We loved Panaya TDx for its collaboration features. It is a user friendly, cloud-based solution that offers easily repeatable test scenarios between similar projects. We would recommend it to any other organization running SAP." Pam Brown | Senior Director, ERP Business Process Organization, Bruker Read the Full Case Study > UAT is a crucial final stage in the software development lifecycle, ensuring the software meets real-world business requirements. Proper planning and scoping are essential to a successful UAT process. Effective UAT requires coordination between technical teams and end-users, and overcoming challenges like dispersed teams and outdated tools. Best practices include gathering knowledge, scoping correctly, designing, executing, and confirming business objectives. Comprehensive documentation is vital for current and future UAT processes. Automating documentation and defect management can significantly enhance UAT efficiency. Tools like Panaya Test Dynamix can simplify UAT, providing real-time visibility, better ROI, and streamlined testing processes. Start Simplifying Your Tests Now – Get The Guide >> What is UAT? User acceptance testing (UAT) is the safeguard against unfinished, ineffective, or faulty software products before go-live. Well-designed, high-quality UAT tests are thorough and reflect user requirements accurately, identifying problems that would go unnoticed in integration or unit tests. Finally, UAT tests provide a macro-level overview of the system's completion. What is the purpose of UAT? User acceptance testing validates the testing done at the end of the development cycle. The software may undergo other testing phases and be completely functional but might still not meet its requirements for the end users. UAT safeguards against faulty, ineffective, or unfinished software products being released. To be effective, UAT should be thorough and reflect user requirements so that the software will not be released with bugs. These issues can be costly and potentially damaging to the software vendor's reputation. Why Run UAT? The software can be completely functional from a technical standpoint and still fail because requirements are not clearly defined or effectively communicated to developers (a widespread problem with evolving projects). In other cases, new code that appeared effective in every virtual deployment model may have been inadequately tested for a dynamic real-world environment.User acceptance testing (UAT) is the safeguard against unfinished, ineffective or faulty software products reaching rollout. It achieves that goal by answering the question, "Have we produced what customers want?"Well-designed, high-quality UAT tests are thorough and reflect user requirements accurately, identifying problems that would go unnoticed in integration or unit tests. Finally, UAT tests provide a macro-level overview of the system's completion.For example, a purchase order interface might be improved to include a new subfield for online customers. A unit test could confirm that the change was executed and integrated correctly. But it will take a user acceptance test to ensure that the revised order functions correctly throughout the purchase-to-pay process, across multiple departments, from initial creation and approval, through receipt and invoicing to accounting. When should UAT be run? UAT is one of the most critical phases of software development and change implementation. It should be run after unit testing so that development teams are satisfied that their code works as expected and after successful QA testing, whether automated, manual, or both. Then, just before the move on to production, the business users put it to the test. Their practical understanding of how the software fits into real-world scenarios can reveal hidden vulnerabilities and ensure the final product meets your organization's business requirements. They are the last word. Who performs UAT? Business users are the actual UAT performers. Key business users are needed for their business know-how on all phases of the UAT cycle – planning, execution, and evaluation and executing UAT before go-live. Functional experts who oversee the technical side of software development play an important role in shaping UAT cycles and interpreting the results. Stakeholders, Business Analysts, and Software Testing professionals can perform these tests, but UAT imposes unique challenges, such as multiple users involved in testing each business process. The right testing management solution should enable project managers and testing managers to easily assign different steps of a single business process to multiple functional experts or key users, who can then run the tests in an automatic workflow. What is the difference between System Testing vs. User Acceptance Testing? System testing is performed by testers and developers, while end-users and clients perform User Acceptance Testing. System testing is responsible for testing the interfaces between the components and interactions to various systems parts like hardware, software, and interfaces among systems. In system testing, individual units must be integrated first in separate builds; however, the whole design is examined in user acceptance testing. User Acceptance Testing (UAT) is a important phase in software testing where the software is tested in a real-world scenario by end-users to ensure it meets their requirements and functions as expected. Unlike other forms of testing, UAT focuses on validating the software's user-friendliness, functionality, and performance from the user's perspective. This testing phase helps in identifying any discrepancies between the expected and actual outcomes, ensuring that the software is ready for deployment.What is user acceptance testing (UAT)?User Acceptance Testing (UAT) serves the purpose of ensuring that the software meets the business requirements and is ready for deployment by validating its functionality in a real-world environment. It allows end-users to test the software to ensure it meets their needs and operates as expected, helping to identify and fix any issues before the final release. UAT is crucial for quality assurance and customer satisfaction, as it ensures that the software is user-friendly, reliable, and meets all specified criteria.Acceptance criteria attributes for UATCompletenessAccuracyUser-friendlinessPerformanceReliabilitySecurityScalabilityCompatibilityWhat is the purpose of UAT?The purpose of User Acceptance Testing (UAT) is to identify bugs in software, systems, and networks that may cause problems for users. UAT ensures that software can handle real-world tasks and perform to development specifications. Users are allowed to interact with the software before its official release to see if any features were overlooked or if any bugs exist.Basic guidelines of UAT: When doing UAT, it is important to test with users in different locations, not just on different devices. It is better

to have one person in a separate location rather than testing with people who are already online together. Also, using email is often more reliable than social media, especially for private communication, like delivering services through apps like Signal.Who performs UAT?User Acceptance Testing (UAT) is typically performed by end-users or clients who will ultimately use the software in their daily operations. These individuals represent the target audience for the software and are responsible for validating whether the software meets their requirements and expectations before it is deployed.Types of User Acceptance TestingBelow are the 5 types of user acceptance testing:Types of User Acceptance Testing UAT1. Beta User Acceptance TestingBeta UAT means that users who have completed one or more rounds of tests will be shown a popup stating if they are accepted for testing by the new version of Angular2 (a beta release). The application is tested in a natural environment.It reduces risks, and failures, and improves the quality of the product through customer feedback.2. Black Box Testing In black box testing, end-users or testers evaluate specific functionalities of the software without knowing the internal workings or code structure. Testers focus on how well the software performs its intended tasks from a user perspective, checking inputs and outputs against expected outcomes. This type of testing ensures that the software meets user requirements without requiring knowledge of its underlying technical details.3. Operational Acceptance Testing (OAT)Operational Acceptance Testing (OAT) is a software testing technique that evaluates a software application's operational readiness before release or production. The goal of operational acceptance testing is to ensure system and component compliance as well as the smooth operation of the system in its Standard Operating Environment (SOE). OAT Testing (Operational Acceptance Testing) is also known as Operational Readiness Testing (ORT) or Operational Testing. These test cases guarantee there are work processes set up to permit the product or framework to be utilized. This ought to incorporate work processes for reinforcement plans, client preparation, and different support cycles and security checks.4. Contract Acceptance TestingContract Acceptance Testing refers to the process of testing developed software against predefined and agreed-upon criteria and specifications. When the project team agrees on the contract, they define the relevant criteria and specifications for acceptance.Contract acceptance testing involves testing the software against specific criteria and specifications outlined in the project contract or agreement.This type of UAT ensures that the delivered software aligns with the agreed-upon terms and conditions between the client and the development team. 5. Regulation Acceptance TestingRegulation AT is generally called Compliance AT.This sort of affirmation testing is done to guarantee the thing dismisses no rules and rules that are set by the regulating associations of the particular country where the thing is being conveyed. Generally, things that are available from one side of the planet to the other should go through this testing type considering the way that different countries have different standards and rules set by discrete directing associations.6. Alpha User Acceptance TestingAlpha UAT means that your user is tested before they get a hold of the product, so if you're testing users' usage patterns we recommend running an alpha test to ensure it can pass all acceptance tests before the beta gets deployed into production. It enables more rapid validation in early adopters/testers which allows fast adjustments as the software progresses through development with each release cycle toward feature maturity.It ensures that there is no opportunity for bugs or exploits once security updates become available based upon adoption levels achieved by products launched later during their life cycle such should be done at least six months after launch.How to perform User Acceptance Testing (UAT)Each type of UAT serves a specific purpose in validating the software's functionality, usability, and compliance before it is deployed for production use. Here are the steps of Execute User Acceptance TestsSteps to Execute UAT Tests1. Requirements AnalysisThis step involves analyses of business requirements. The following documents will be considered and studied thoroughly to identify and develop test scenarios:Business Use Cases.Business Requirements Document (BRD).System Requirements Specification (SRS).Process Flow Diagrams. 2. UAT Test Plan CreationIn this step, a test place is created that will help to outline the test strategy that will be used to verify and ensure that the software meets the expected business requirements. The test plan includes entry criteria, exit criteria, test scenarios, and a test case approach.3. Identify Test ScenariosThis step involves identifying the test scenarios will respect to the business requirements and creating test cases listing the clear test steps. The test cases should cover the UAT test scenarios. 4. Create UAT Test CasesCreate UAT test cases in this step that cover most of the test scenarios. Business use cases are the input here to create test cases. 5. Prepare Test DataIt is considered a best practice to use live data for UAT testing, UAT testers should be familiar with database flow.6. Test RunThis step involves executing the test cases and reporting the bugs if there are any. Re-test the software once the bugs are fixed. In this step, test management tools can be used for test case execution. 7. Confirm Business ObjectivesIn this step, the UAT testers need to sign off the mail after the UAT testing to ensure that the product is good to go for production. Deliverables here are Test Plan, UAT Test Scenarios, Test Cases, Results Log, and Defect Log.Challenges of User Acceptance Testing (UAT)Challenges of carrying out User Acceptance Testing include:Misreporting Activities: The use and misuse/misreporting activities by potential users can be extremely challenging to control. For example, this issue may arise when a company is not equipped with appropriate information systems. Proper Example to demonstrate: Provide an example project to test the validity and reliability, or at least performance, aspects - such as time complexity, resource usage per user, etc. Proper Evaluation: Evaluating how this information is handled by users after a successful acceptance Test needs to be done using common programming tools that provide adequate input data including HTML formatted examples with optional inputs for feedback before/after each iteration.Usability: The tester’s job is critical in UAT since they must demonstrate the usability of your product by simulating real-life scenarios. They must also gather information on how your users interact with your product. Proper Balance: In addition to inviting users, IT professionals must balance user input and expectations with costs and constraints. For example, some companies limit the number of users per computer during their beta tests. This limits both costs and data collection. Limitations of Actions Performed by User: There are also limitations on what actions each user can perform within the program— for example, some programs have an expiration date so that companies do not waste valuable data on unappealing customers.Need for User Acceptance TestingUsability: There is a need for User Acceptance Testing in Software Testing for any product because the software test process relies on users to get used. Feedback Mechanism: The best way to reach users and ensure their confidence is to introduce feedback via usability testing using tools.Non-Feasible Cause: There are some reasons why this may not be feasible: A lack – there are many ways one can go about providing user acceptance tests within applications that have complex features - ex "Safari" has been tested extensively by testers since its debut 2 decades ago with little change. The use case will become simple very quickly with proper testing methods. Documentation and Communication: Software testers are looking for good documentation about how code works so that they can verify whether what they are doing does exactly as promised; this ensures an honest test if required or just improves their ability to communicate on behalf but not through tests themselves. It also opens them up to trying something novel (like some new features) until there are problems presented by a lack of functionality. A few tools used for UAT are listed below:1. Marker.ioReport visual bugs straightforwardly into your devices, without leaving your site or web applicationIt lets users post messages, comments, and events to a "hub" hosted on Google Analytics, with an optional delay between updates which ensures only one message gets sent per second. This delays your data loss by eliminating any accidental user interactions that might interrupt their Web App flow. 2. FullStoryEnables clients to track and screen every client action. From snaps to page advances, everything is listed consequently. It allows you to visualize user acceptance and rejection through some graphs, similar in functionality to GraphPad but with a lot more flexibility.The data can be viewed either via interactive dashboards like Scrum or by drawing on individual parts of it that are then visualized along with actual user feedback using your favorite software It makes this kind of structured test much easier than one would typically think, perhaps even less frustrating.3. HotjarUncovers the internet-based conduct and voice of your clients. Hotjar provides you with the '10,000-foot view' of how to further develop your site's client experience and execution/transformation rates. This application runs a service that keeps track of an online database of people who have ever viewed your website. The following page summarizes what Hotjars offer and provides tips on creating websites using them. Also, It allows users to run tests from a command line and it does a great job at testing various features that may be added later on.4. CrazyEggA web-based device that screens individual pages from your site, providing you with a breakdown of where various guests have clicked and on what part of the screen.The user will need to build a class with all needed methods and return values along its arguments so that this can be easily tested by other developers or clients/users using different APIs like Selenium Server test suites. It comes in two flavors - one which builds on top of Mocha Test Suite i.e. WebDriver, and the other has just built upon MuleTest's framework but adds some custom features such as implementation through Sockets, etc.5. Qualaroo Allows users to easily test their Web Apps.Qualaroo is a Python library that allows users to easily test their Web Apps. Common data structures can be created in Python which allows us to directly run our tests against different server configurations using QA tools like RSpec and TDDRunner. 6. SentryA web interface that allows users to write acceptance tests on their ownIt's simple but effective and has been accepted into several national standards bodies such as ISO 9001 and ANSI X9-TRIAMS. Sentry provides a web interface that allows users to write acceptance tests and upload them by selecting an option on their dashboard from the toolbar menu with various test cases selected during setup. How to Perform User Acceptance TestingHere it the Proper Guidelines of the Acceptance testing which is Covering in detailStep 1: Understand Business Requirements and Goals Begin by reviewing the business requirements and goals to fully understand the project's objectives. Use key documents like Business Use Cases, Process Flow Diagrams, and System Requirements Specification (SRS) to help create test scenarios that align with the project's needs.Step 2: Create a UAT Plan with Assignments Develop a UAT plan that outlines the approach to ensure the application meets business requirements. Include entry and exit criteria, detailed test scenarios, and timelines. Make sure to assign specific tests to various parts of the software, simulating real user experiences. Use clear, simple language for instructions, keeping in mind that not all users are technical experts.Step 3: Identify Test Scenarios and Create Test Cases Identify key test scenarios based on business processes and design test cases with clear, actionable steps. Focus on functional aspects and UI/UX performance. For example, check if buttons, drop-downs, and sliders work properly, if overlapping elements interfere with scrolling, and if all CSS styles display correctly.Step 4: Prepare Test Data Use real, live data for testing, but make sure to scramble it for privacy and security. This is best handled by professional testers who understand how to replicate real user behavior.Step 5: Execute Tests and Track Results Run the test cases and document any issues you encounter. Once bugs are fixed, re-test to verify the issues are resolved. Pay close attention to feedback from testers, as they may use simple, non-technical language. Use feedback templates to organize and present responses clearly.Step 6: Confirm Business Goals Are Achieved Finally, ensure the project meets its business objectives. Verify that the application delivers what was initially promised and addresses the business needs effectively.Exit Criteria for User Acceptance TestingThere are some Exit Criteria required to be met for User Acceptance Testing. They include:Confidence: A high level of confidence that the proposed user has enough knowledge, experience, and skill set to perform at least one task effectively.Proper Execution: Where tests show users can contribute fully to existing tasks successfully using their expertise. All three terms represent different levels with each being less than 50% when compared to full-time professionals in this area. When you use these two criteria as input your goal is to gain support from others who have achieved similar results through other research methods instead of focusing on just learning how important it was once they got there. Lesser Defects: After analyzing the test results, project managers should be able to draw some conclusions based on what they've found. For example, if there are more errors during testing than expected, this can be taken as a positive sign. It shows that the program is easy to learn and use which is a necessary condition for successful implementation. In addition, this means that their project objectives are understandable and easily implemented by end users. In other words, their business process works satisfactorily. If there are fewer errors than expected, this can also be taken as a positive sign. It indicates that implementing certain security measures early in the development lifecycle will go a long way in reducing unexpected errors during testing.No Critical Defects: After drawing these conclusions, project managers should ensure that all critical defects found during testing are resolved within one month after launch. This allows them time to notify users about any lingering issues and rectify any critical bugs before releasing the final copy to end users. Doing so will increase the likelihood of satisfied users and increase early adopter interest in your product.Proper UAT: A well-coordinated UAT helps software developers identify problems early on in their projects- thereby reducing overall development costs and timeline delays. Hence, performing UAT effectively requires a lot of planning and commitment from every developer involved in a project. The developer performing the test should analyze each scenario carefully before moving on to the next test step. Doing so will help them produce a quality product that satisfies customer needs and meets project objectives at the earliest possible time.User Acceptance Testing vs System TestingHere are the Difference between User Acceptance Testing vs System TestingParametersUATSystem TestTesting MethodIt is based on User Convenience and Feedback. Testing is done based on meeting customer requirements.It is based on the System/Feature Milestone - Testing is done based on meeting system/specs requirements.UsageUAT is done by Clients, Stakeholders, and Testers.System testing is done by the Developer and Testers.TypesAlpha UAT, Beta UAT, Operational acceptance testing, Contract AT, Regulation AT. System and Integration Tests.Test CasesTest cases here include positive scenarios.True and False (+/-) Test cases.Testing MethodologyThe testing methodology of UAT is based on Functional Tests.Functional / Performance/ Other Tests.Order of ExecutionUAT is done after the System Test.The system Test is done first.Acceptance Testing vs V-ModelBelow is the difference between acceptance testing and V-Model:ParametersUATV-ModelTesting TypeIt is based on customer feedback.It is based on the Verification and Validation Model.DurationUAT is carried out after every iteration post-system test.In V-Model, testing happens at the end only.Test ScenariosTests Based on Customer Requirements and happen Feature by Feature / Module by Module.Happens whole as a software.PhaseUAT is carried out after the system test.For every development phase, there is a V-Model Test phase.Customer InteractionIn UAT, interaction with customers is done regularly.V-Model doesn't need customer interaction regularly.ConclusionIn conclusion, User Acceptance Testing (UAT) and the V-Model differ significantly in their approach to software testing. UAT focuses on customer feedback, occurs after each system test iteration, and tests features based on customer requirements with regular customer interaction. Each method has its unique benefits, making them suitable for different testing needs and project requirements.