September 29 2023

Game Testing

Tutorial: A Comprehensive Guide With Best Practices And Examples

Delve into the importance of game testing to identify and resolve bugs, glitches, and performance issues and ensure a seamless player experience.

OVERVIEW

Game testing, also known as video game testing, evaluates and assesses video games to ensure they meet specific quality standards before being released to the public.

It is the process where team testing efforts are focused solely on the gaming applications, as expectations from each element in the game application are very high for someone who plays a game regularly.

Therefore, testing needs to be extensively

planned and executed perfectly to meet that expectation. Surprisingly, a gaming application is much more than characters and controls, making game testing a difficult but exciting area.

What is Game Testing?

The most straightforward explanation of game testing is that it is the process of testing a gaming application. However, things may be more complex than they may be, for, say, a static simple web application.

A game is much more complex than a web application, with many elements networked together to work in sync with each other. For instance, if the network is slower than expected, one cannot drop down the quality of the rendered web page to make things faster.

So, we skip this setting in web development. However, in game applications, gamers do make use of resolutions based on several reasons. They may vary from the screen's ability to view a particular resolution, slower network connection, or to save bandwidth.

Game testing ensures that all the elements in a game application are bug-free and work as

intended on the target devices. If we consider it to be a mix of many types of web applications where one focuses on UI, one on video elements, one on graphics, etc., we might not be entirely wrong.

This makes us conclude one thing even before starting the exploration of these elements, i.e., it cannot be conducted similarly to other types of testing. Game testing needs particular attention, and there are a lot of reasons to support this statement.

Why Game Testing Needs Special Attention?

The game development journey taken over the last fifty years is nothing less than remarkable, from games like Space Invaders in 1978 to high-end games like Red Dead Redemption 2. They have built up billion-dollar organizations from the ground up, and sometimes, just a single game has been their lucky charm. This tells us how much an organization has at stake when a new game or version of an existing game is released to the users.

When a tester is testing a game application

before release, the following gets affected by his work:

- **Business reputation:** Business reputation can easily top the list of what game app quality means for an organization. Businesses can make their name in the market by quality, which depends on sound development and testing skills.

  This is important because one organization rarely focuses on a single franchise of games over a long period, say 20 years. They try to experiment and follow the trend as well. For a new organization trying its hands for the first time, a lot of expense can go into marketing, gameplay reviews, building trust, etc.

  However, all of these things are optional when the organization has already made its mark through one game, as gamers are loyal to such aspects, and if they do not play regularly, they will try it once and experience it.
- **Business growth:** A game application with good quality grows with word of mouth in this age of social media. PUBG mobile, a game whose popularity needs no introduction, has made over $10 billion

dollars through a single game, according to SensorTower.

If the game remains as popular as it is, the revenue will only increase over time. But all this depends on a bug-free application. The number of users starts to decline if the application is buggy, and therefore, a huge loss of revenue can be observed. This can also be irreversible due to loss of trust.

- **Exploring bugs:** The main motive of testing is exploring bugs in the application. While the result of how a game looks and feels to the gamer depends on many other teams, a bug is exclusively the responsibility of the testing team. It helps close any loopholes and lets the functionalities work as intended.
- **Test functionalities:** Developers will test the basic functionalities as they develop the features and write unit test cases for them. However, we cannot rely on that as the testing is far from what could guarantee our desired quality. For instance, load testing is something that developers do not do but is extremely important. Testers explore the functionalities in technical and non-technical ways and ensure the game features work as expected.

- **Minimize risks:** Brainstorming, analyzing, and documenting risks is one of the primary jobs of a game tester. With this, they minimize risks in a game application, which can prevent situations such as crashing the app, working abruptly, etc.
- **Identify areas of improvement:** Along with determining the application's functionalities, we also need to explore areas where improvements can be made. They may be working exactly the same as written in the requirements. However, they can be improved in areas such as dense pixelated UI or transitions over navigation.
- **Ensures user satisfaction:** One of the important benefits witnessed in game testing is ensuring user satisfaction. A game application will only succeed if it is bug-free, works seamlessly, and can excite the user every time they operate the application.

  This is an important skill to attain by a tester who can act as a user and ensure that the application will engage the user and keep him going. This is not required in any other application testing except when the application is a game.

Other than these game application-specific factors, we can also assume the generalized

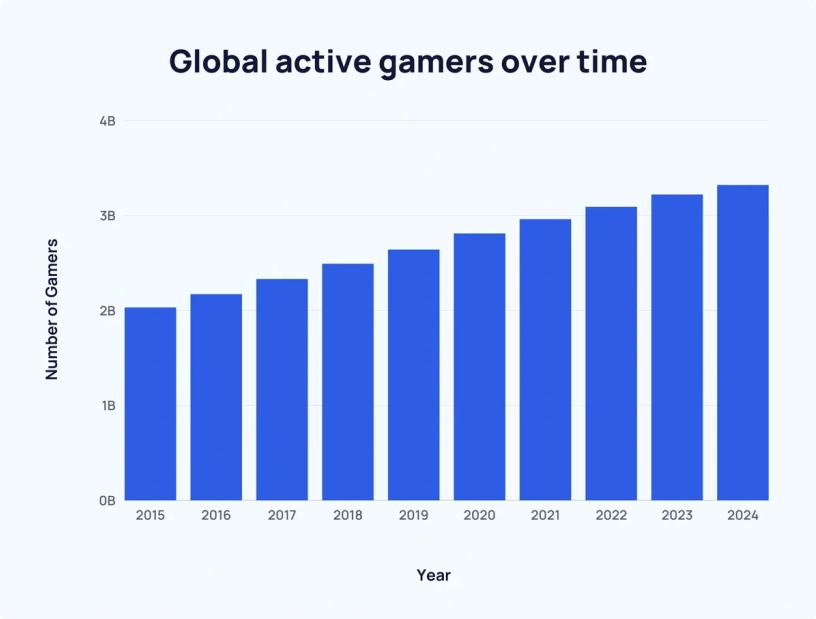
factors in the testing department. For example, checking test coverage, analyzing usability criteria, etc.

Types of Games

When we talk about game types, we can divide this category into two parts:

- The game genre
- The game platform

The first category is the game genre. It can be an arcade application, a racing application, etc. With the increase in the number of game application users (especially mobile game users), organizations try to differentiate themselves from other competitors and constantly invent new categories.



Source

With over ten main game genres and countless subgenres, discussing this category could result in a rant with little value.

The second category, however, can summarize all these genres and all those apps to give us more insights about their behavior and testing approach.

The first thing a team decides before development begins is the platform for which the application needs to be developed. It is a crucial decision because of different frameworks, technologies, and developers are often different for different platforms. All these parameters expand even further when we develop a game application.

- **Mobile:** A mobile game application targets

devices that run mobile operating systems. These can extend to:

- **Android** - Targets the Android operating system. For example, Half-Life 2.
- **iOS** - Targets iOS operating system. For example, Oceanhorn 2.
- **Windows Phone:** Targets Windows Phone OS. For example, Asphalt 8: Airborne.

These are the three major mobile operating systems. There are multiple other systems for mobile as well, but the market share is meager, which is reflected in their game development process.

- **Desktop:** Desktop game applications target operating systems that run on a desktop. These come with high-end specifications (such as graphics) rather than a mobile device.

  They have multiple components attached to them, such as a keyboard for key-related functions and a mouse for cursor-related functions. This makes the testing of a desktop game highly complex and challenging.

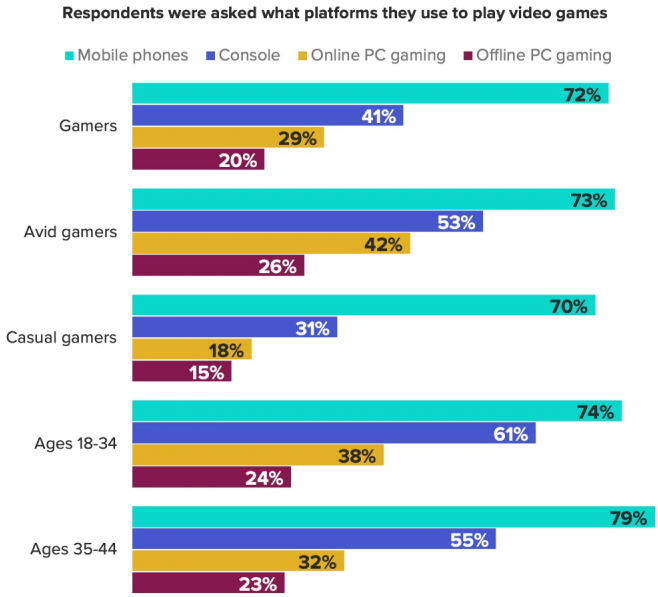
- **Windows OS:** Targets Windows operating system. For instance, Age of Sail 2.
- **macOS:** Targets Mac operating system. For example, Cap'n Magneto.
- **Linux:** Targets the Linux operating system and its multiple distributions. For instance, DOTA 2.

As with mobile, desktop operating systems are also large in number, but the above three cover more than 90% of the market.

- **Consoles:** A video game console is developed especially for playing gaming applications, which generally have very high specifications, and unlike a mobile device or a desktop, a video game console is not multi-purpose. The only job it performs is to process the game application and output it through video channels on the output device, which in most cases is a television or a monitor.

  Manufacturing and distributing the console to the world is a highly high-cost-intensive job. Due to this, console manufacturers are limited, and whatever products are in the market are priced

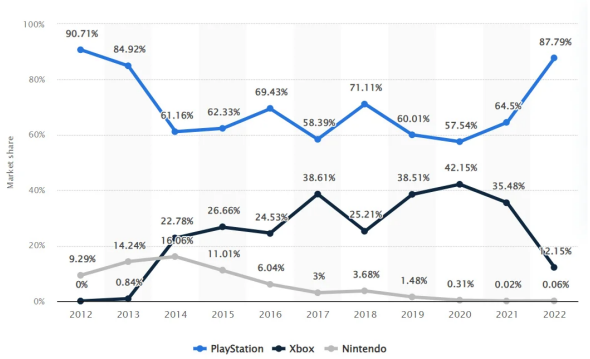
towards a high point. This restricts the majority of people from buying video consoles. However, we still find that 3 out of 5 people play games on video consoles.



Source

Such a huge demand is catered by the following consoles:

- **PlayStation:** Targets Sony's console, which covers over 87% of the global market share.
- **Xbox:** Targets Microsoft's console, which covers around 12% of the global market share.
- **Nintendo:** Targets Nintendo's gaming console, which covers around 0.06% of the global market.



Source

Currently, these three players have captured the complete market of video consoles, and as a tester, testing on these platforms should be enough.

- **Virtual Reality:** VR games enable gamers to experience 3D surroundings virtually. These games work in two ways:
    - The game is installed on a mobile device, and the device is connected to the VR box.
    - The game is installed directly on the VR box.

  Both approaches require additional development work, especially when installing directly on the VR due to operating system issues. The games supported in VR include

Half-Life: Alyx, Assetto Corsa, etc.

These are the actual physical machines that need their respective supported games to run. However, we can also achieve the same using virtual machines that let another operating system run on top of a different operating system. This method works, but with a couple of glitches:

- Applications start to lag because of multiple kernel and CPU calls as they are converted to the virtual box.
- Gamers can misuse this and start to play a mobile game on a desktop for better rankings. Therefore, organizations keep targeting such behavior and impose restrictive codes.

Using the real physical machine for which the game application is intended to run is recommended to make the most use of the features.

Game vs. Software

Tester

A software tester who works on game testing is called a game tester. However, vice versa is not true. There are specific skills that a software tester needs to acquire for game testing.

A gaming application needs constant engagement from the user; otherwise, losing them is risky. We cannot wait to optimize this behavior after the release, as the risk is too high. In such scenarios, only a tester with an eye for gaming applications can help. He needs to be interested in games with a past gaming experience that can make him a perfect candidate for this job. A software tester might test the documented functionalities perfectly, but a game tester can test the "*game*" as it is intended to be "*played,*" not "*developed.*"

The second most important quality in a game tester is their knowledge of existing games in the same genre and platform. While releasing a game, we must be aware of our competitors and their offerings. This helps us take advantage and gain an edge over them.

This quality of a game tester will also help point out specific bugs or issues that can only be discovered by playing and not through automation testing, as we do through usability

testing in software. For an organization starting to develop games, it is vital to hire people with these qualities; otherwise, the organizations need to hire freelancers to do this job, which is always costly and not secure.

Aspect	Game Tester	Software Tester
Primary Focus	Testing video games for quality and gameplay.	Testing software applications for functionality and performance.
Skill Set	• Interest and experience in gaming. • Knowledge of gaming genres and platforms. • Ability to play games as intended. • Familiarity with existing	• Strong understanding of software development. • Testing documented functionalities. • Focus on functional and non-functional aspects of software.

	games in the same genre and platform.	
Testing Approach	Tests games as they are meant to be played.	Tests software based on specifications and requirements.
User Engagement	Requires constant engagement as players.	May not always require constant user engagement.
Competitive Awareness	Needs to be aware of competitors and their offerings in the gaming industry.	Competitive awareness may not be as critical for software testing.
Bug Discovery	Can discover specific bugs and issues through gameplay that automation testing may miss.	Relies more on structured testing methods and automation for bug discovery.
Cost and Hiring	Important to hire individuals with gaming experience and skills.	Typically, in-house testers with

Considerations	Freelancers may be needed if in-house expertise is lacking.	software testing.

Game vs. Software Testing

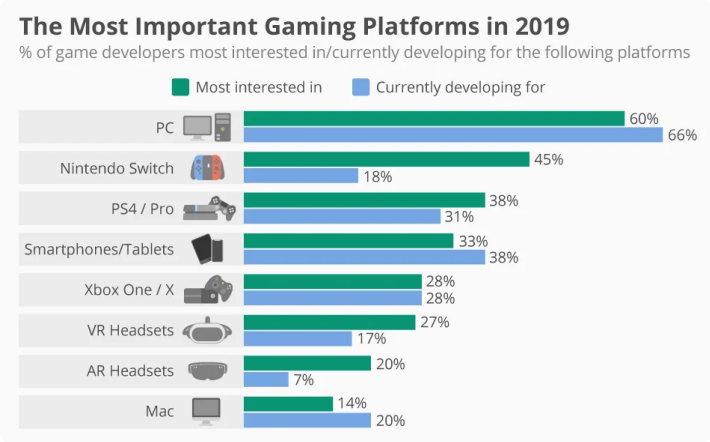
As we need a different skill set for testing a game application, so is the case when the testing approach is finalized. Considering game testing similar to software testing can turn the release strategies upside down with many bugs and possibly the application's early death.

To compare the stark differences between the two, we can take the following parameters that give us the clearest view for better understanding:

- **Complexity:** Game testing is highly complex. The absence of set standards and protocols and still managing the quality of the application is the reason behind this challenging affair. The testers need to focus on minor details that might not worry a user on a web application but can certainly do for a gamer. Such a complex environment sets apart game testing from software testing.

- **Platforms:** Game applications are unique in a way that a complete physical device or a complete operating system can be developed just for them. For instance, PlayStation is a device just to run game applications. This system works well because a completely separate system keeps the costs to a minimum while providing the maximum configurations.

  This is hard to achieve on a laptop or desktop. Due to this, when we start to test games, a long platform queue awaits us, demanding the repetition of the same experience on all of them. This is also reflected in the survey where developers are asked about their priorities when it comes to game development.



Source

Naturally, a single tester cannot test all these platforms. This creates a demand to hire more testers with the same skill set and

coordinate if there is a bug to be resolved on all of the platforms. Finding like-minded testers for the same game genre and experts in a platform is a challenging job in itself.

- **Requirements and planning:** The initial stages of software testing start with designing and planning the steps according to the requirements. It takes considerable time and effort, but keeping track of testing and the developed functionalities is vital. In a way, all the subsequent phases refer to planning throughout the testing cycle.

  This is not true with game testing. If we start planning a game test, the application quality will be inferior. It happens because of a user's behavior with a game application that is always random, unlike a web application where developers can easily control the flow. Requirements are gathered at a minimal rate to understand the critical functionalities, but planning is kept to a minimum in game testing.

- **Automation testing scope:** Automation testing uses test scripts to execute tests on the applications automatically. This can be integrated into pipelines, frameworks,

regression suites, or kept as a standalone function. Software testing uses automation testing on a large scale and always aims to cover as much testing as possible through this method. But this can be made possible only because we know the actions we need to perform and can convert the same using test scripts.

When it comes to game testing, the actions are random in nature. We cannot imitate most functions and, therefore, depend heavily on manual testing to do the heavy jobs.

- **Team size:** A game has a lot of components to test, and we need different types of testers to perform those testing tasks. This makes the organization hire a lot of people. Such is not the case with software testing for other applications. In software testing, even if the team is large, they do not work together on a single domain and keep their work isolated.

  With a larger team working together on overlapping areas, excellent coordination is required to avoid missing crucial bugs. In addition, we also need to have management overheads that can take up a lot of time and not leave much for the

actual test execution. While manually achieving this feat could be challenging, cloud-based platforms/tools can provide a centralized system to showcase all the information and activities done by team members. It helps keep everyone's work in check.

- **Testing approach:** Software testing works on a planned approach with references to designs and requirements already documented before the process begins. Therefore, we get a laid down path to follow, and the testing approach can be termed *"straightforward"* or "*planned*."

  Game testing does not have a plan like software testing. Game testers might not follow the process as they do today for any version in the future. It is unpredictable, and this makes it perfect for exploring crucial bugs. Due to this nature, game testing is termed "*random*" or "*unplanned*."

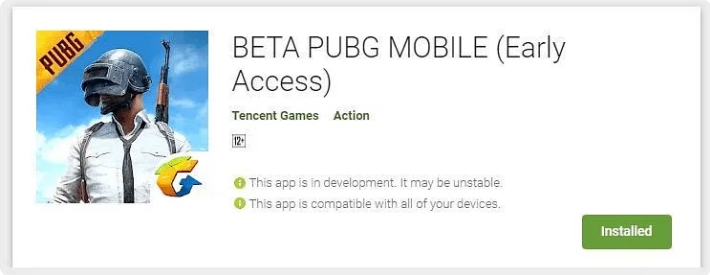
- **Release approach:** Software testers can rely on their automation tests, designs, and plans to guarantee a successful release. When all these things are green, they generally provide a "*ready for release*" signal

in the test reports , which is enough to release the application to all users. After all this, even if we get a bug after release, it is easier to fix and does not impact the complete application in most cases.

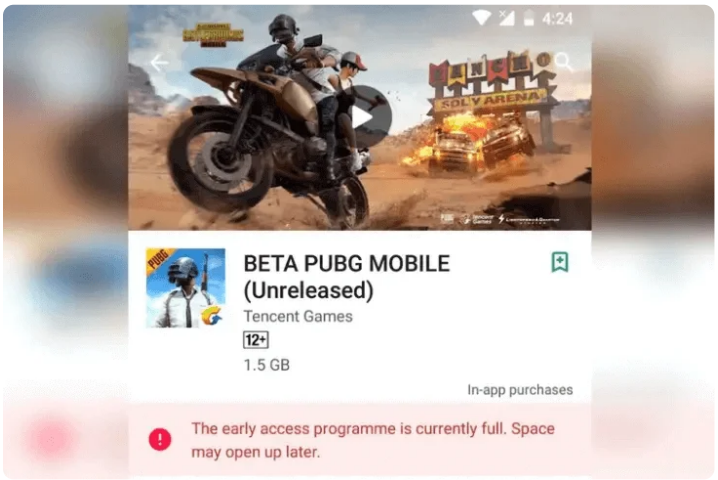
Game testing follows a different approach for release due to the magnitude of risks involved. A game application cannot afford a bug as the word can spread quickly in the gaming community, and users may never return due to a bad experience.

On top of it, game development costs much more than, for instance, a web application. No organization can, therefore, afford to start from scratch multiple times. All this makes us release the game application in phases to explore bugs not caught during the testing cycle. For this, organizations prefer beta testing.

Beta testing is a part of the testing cycle where a normal user acts as a tester intentionally or unintentionally. For this, applications put down a beta sign over the application, marking it unstable for global use at the current stage.



Developers also release a notification to gather the testers as quickly as possible for early releases. Once the beta application is installed, it is up to the core testing team on how they would like to use the gamer's experience. They can collect logs and reports or contact the gamer for their experience and pain points. However, beta releases are restricted in number, and only a few people are allowed access.



Once complete, the early access is restricted through a notification, as shown in the above image. Beta testing explores

bugs that only a gamer can point out. We can also witness beta access programs for other applications, such as web applications or fintech native applications. However, they are rare due to the minimum risks involved, and the focus is more on maximum sign-ups.

- **Difficulty:** Game testing is easier to perform because of the randomness of the testing process. Without any fixed path, game testing becomes a more "*fun*" phase rather than restricted to documents. Testers complete this phase as users provide suggestions, and the feedback cycle is more open.

  Software testing might not be so flexible when it comes to software application testing. Since they are restricted, any suggestions and feedback may revert back to a "*not a requirement*" answer, which is fine because it may ripple into other possible changes in the requirements that no customer would want. This makes software testing a bit more complex, even with lower complexity.

- **Tester's knowledge:** Finally, we can

compare software and game testing based on a tester's knowledge required to conduct each.

Software testing is primarily based on theoretical knowledge. When we learn software testing, we learn about designing, coverage calculations, writing scripts using predefined methods, etc.

Game testing, on the other hand, is much more practical, and no institute teaches it explicitly as it lacks theory. When a tester starts testing a game, he just has a premise in hand, but things can quickly take a turn, which is expected and recommended for better game quality.

Therefore, it might take a lot of time for someone new to game testing to understand how to approach each element and explore bugs without any predefined scripts.

These differences help us understand the complexity a game testing environment brings with it and how it cannot be clubbed together with software testing. Leaving aside software testing now, let's dive a bit deeper into game testing and how we start and follow the approach towards it.

Life Cycle of Game Testing

The Game Testing Life Cycle is a part of the Game Development Life Cycle, which is divided as follows:

- Pitching
- Preparation
- Implementation
- Testing
- Pre-production release (Beta)
- Deployment

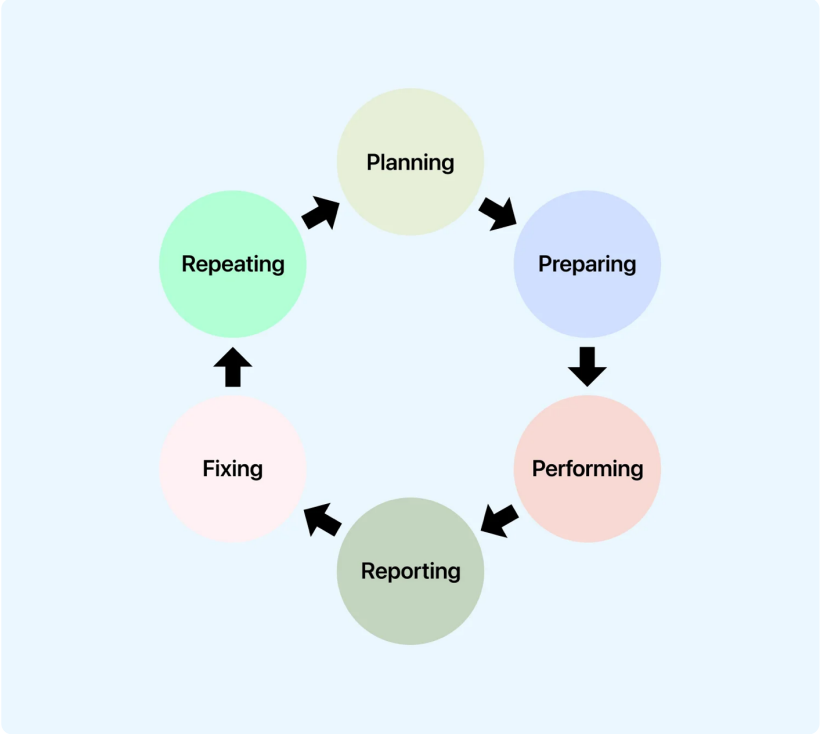
The testing part here is where the Game Testing Life Cycle begins. This life cycle is completed in six phases:

Planning

The first phase is planning the game testing. This phase is not as strict as software testing's planning phase. Here, we make a rough sketch of the basic features, what areas are improved, how they are improved, and what previous bugs have been fixed. Unlike software testing, we do not plan every step that we need to follow during the testing phase.

Preparation

In the next phase, we prepare testing environments, devices, specifications, and everything else required for game testing. If something needs to be reported, the documents or a bug list is prepared to verify any fixes.



Performing

In this phase, the testers start the testing process with methods described in this post. They act as gamers while always looking out for bugs and defects in the application. The testers try to explore the application and even document their playing experience.

Reporting

All our findings and exploration of the application, including verifying existing bugs, need to be reported in test reports created in this phase.

Fixing

As we discussed, game testing is a process of exploring the application without following any strict plans. Due to this, we may find a lot of improvements that cannot be documented as bugs as we would in software testing.

Therefore, in the game testing, we include a feedback cycle for teaming up with the developers to discuss the bugs, improvements, and everything else reported during the previous phases. The team then decides the path to take and works on those feedback accordingly.

Repeating

Repeat all the processes from Step 1 (Planning) to re-verify the improvements, bug fixes, and everything else on the updated build to make sure we release a high-quality application.

These phases can be further broken down into more sub-phases, but the cycle's gist should remain the same.

What is Playtesting and its importance?

Software testing has a method called usability testing that involves hiring users to operate the application and ask them questions, take feedback, or analyze their actions. This process when done on a game application is called "*Playtesting,*" and the testing type is called a "*Playtest*".

Playtesting is a process of managing quality control of the game application by hiring gamers from across the globe who find defects in the application. This is similar to what the tester would do, but in playtesting, we involve professional gamers who do not play as gamers but just to point out the bugs.

They explore each corner of the application and bring their gaming experience with it, which helps not only in finding defects but also in providing feedback to include elements that only a gamer can provide.

Playtesting cannot be done at the end of development. Things become too complex, and time to deliver is very little to make any major change. However, if defects could be explored at a very early stage, we can do it in minimum time. Due to this, playtesting is conducted at various levels, which are discussed below.

Levels of Playtesting

Playtesting can be conducted at four different levels:

- **Gross playtesting:** Gross playtesting refers to testing the foundational elements of the game application. For example, the basic gameplay accuracy, faults in gameplay, and similar issues. The game is not fully developed yet, and the main aim of gross playtesting is to build strong foundations to avoid bug rectification at a later stage, which is very expensive.
- **In-house playtesting:** In-house playtesting is the method where an organization's

testers are included to test the game. These testers require special skills, which we discussed in the game tester vs. software tester section.

- **Blind playtesting:** The challenge with always including professional testers is that they may assume many things as they have immense experience in the area. For instance, they may assume where audio controls will be present, where customization options are, or which icon represents what function.

  A similar thing happens with software testers as their technical eye may overlook certain things as they already know them. Blind playtesting, therefore, targets normal users without any game experience to understand the new user's behavior. Testers record their actions and understand if they need to move things for a better gameplay experience, especially for new users. Organizations can club blind playtesting with beta testing as well.

changes. The focus is to guarantee a final confirmation from the users before the release.

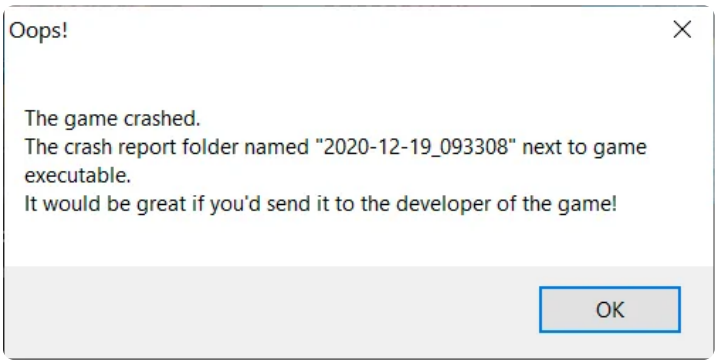
Once all four levels are completed, the game application is released to the end users on the platform it is built for.

Types of Expected Bugs in Game Testing

While functional and non-functional issues continue to be a part of game testing (like software testing), we can expect a new range of bugs in this scenario. Having a basic knowledge about them beforehand will help us explore critical bugs outside of game functionalities.

- **Crashing:** A game application is an extensive application from a developer's point of view. It contains a long list of files with multiple functions linked together, working in sync for a good experience. However, things do not work out as expected often, and there is no way to check each line of code, its impact, links with other codes, execution details, etc.

With these many things, no organization would want to show a page like this to their users:



A sample crash scenario on a Windows PC game

Such bugs are labeled as "*Crashed*" or "*Hanged*," determining a stage beyond which the execution of the game stops automatically. In most cases, the user is required to start from the beginning, downgrading the user's game experience.

- **Object perfection and Physics:** While game applications do incline toward the fantasy side, developers need to follow the rules of Physics for most of the objects. This is because each user fills in certain information based on their real-world experience.

  For instance, a box is expected to have six sides as it is a cube. Or a character should put right leg after left leg while walking.

Such bugs require careful analysis of each object placed on a screen during gameplay. It certainly makes things harder for a tester but, in the end, results in a perfect game application as well.

- **Sound-related bugs:** Another essential element of a game application is the sound effects. It is extremely rare to find a game application with no sound. Games as basic as Flappy Bird also try to add some audible elements for user engagement.

  It has become so essential that every game user does expect this function on any game they download from the store. Therefore, for a game developer, it becomes crucial to put audible elements in various places. As a game tester, it becomes essential to test those elements as they enhance the overall game experience.

- **Graphical bugs:** Something that we rarely talk about on a web application or generic mobile native applications is the graphical issues. Web developers believe in vector images that can be scaled to any amount without getting pixelated. There are no concerns about graphics for development or testers.

On the other hand, the first thing any experienced game user will notice is the graphics, and the downside of which is that they can catch bugs very easily due to that. The following image shows two images of the same scene where one has slightly inferior graphics than the other:

Source

The recent advancements in game engines have produced extremely high graphics in video games. For instance, the following is a screenshot of Starfield created on Creation Engine.

For graphics that look like a real-world scene, testers should ensure that the main characters and all the elements on the screen complement the graphics intended by the developers.

- **Text-related bugs:** Game applications contain a lot of textual information. While some information given as settings is common, much of the print goes out for narratives, storytelling, and much more. Now, it is important to note that even if web applications have a ton more information in written text, they can be corrected by us even if spelled a little wrong.

  For instance, an address can be understood by the user easily. However, game applications have stories and narratives, which makes the textual elements similar to subtitles that can be spoken and need to be understood to understand the game application. Their relevance is, therefore, more than what

we see on a web application.

- **Functional defects:** These are the expected bugs that revolve around the functional elements of the game application. For example, the features of games, gameplay elements, features directly related to the controls, etc.
- **In-game purchases and payment issues:** To gain maximum application downloads, organizations (especially the ones with mobile game applications) launch their application for free use.

  But they need to pay the bills and keep the organization running. For this, they need a medium to earn, which generally comes from in-app purchases, either for ad-free experiences or customizable elements.

A snapshot of in-app purchases in a game application

Since this is the only way to earn for the businesses, testers need to make sure that these functions work smoothly and that the purchases always go through unless interrupted by the user himself. Such bugs are critical and should be escalated as soon as they are found.

- **State of game bugs:** Finally, ensuring the game is saved when the user wants to (if not automatically) and resumes from the same state is important. This is a common feature because most games run from point A to point B rather than starting from the same point each time.

  A basic puzzle-solving game, too, should start from the same level at which the user left it last solved. A bug in this application section could frustrate the user and prove costly for the organization.

Covering these types of bugs in game testing is enough to ensure a high-quality game application.

Game Testing

Techniques

Game testers try to follow various techniques to explore the bugs in the game applications. Different techniques focus on other areas of the app and help improve the overall quality of the application. The most popular techniques followed by game testers are listed below.

- **Combinatorial testing technique:** A popular approach to testing something that includes more than one option is combinatorial testing. In this testing technique, we make multiple combinations of the options available to us and try each to verify their working with the game application. For instance, aspect ratio and resolution options are available in a game application that can be mixed, and the final effect can be checked.

  The only issue for a tester while working on combinatorial testing techniques is the number of options modern game applications have. When we start making the list of the combinations, we realize how hard it is and how much time it demands from a tester.

  To tackle this problem, a game tester may look for solutions on the Internet that

contain a few websites that take the inputs and generate an Excel sheet with all the combinations. Unlike software applications where data driven testing can take over for the next step, this part needs to be conducted manually in the game application.

- **Load testing technique:** Load testing technique ensures the application's current version can withstand specific loads that our application may face. The process of load testing and the thresholds depends on the type of game application.

  For instance, many independent organizations conduct competitions for shooter game multiplayer applications at a very large scale. During the competition, a sudden surge in traffic is expected not only because of players but also from broadcasters and other viewers. A predefined threshold value of load helps such events to be conducted smoothly and efficiently run the application on average days.

- **Ad hoc testing technique:** In ad hoc testing, testers don't follow any pattern or procedure. Unlike other testing, we are not starting from state A and going to state B with an expectation to verify the

functionality. Here, we just explore the application randomly, due to which sometimes ad hoc testing is also referred to as "*random testing*."

- **Cleanroom testing technique:** This technique uses mathematical models and statistical techniques to theoretically define the current game application as one with zero defects. This technique is theoretical, and the existing theories and mathematics models help certify the game application or gain certification from international standard organizations and tag it as "*reliable*" with proof.
- **Playtesting technique:** The tester plays the game as a user in this testing technique. The main aim is to focus on the non-functional aspects of the game application, including the excitement factor, the engagement factor, the difficulty factor, etc. This technique can also be done using freelancers or independent game application users.
- **Regression testing technique:** The elements that can be checked through automation are kept in the regression testing bundle. These tests are run each time a new code is pushed to check the basic functionalities of the application and

whether or not it affects any existing code. It helps keep the codebase stable and prevents major bugs from reaching users.

Along with these techniques, other techniques complement them too and depend on the tester if they wish to adopt in the existing cycle. However, the above-listed techniques are certainly enough to test the quality of the application successfully.

Game Testing KPIs

Key performance indicators are used by testers to analyze the areas that are weak and need more improvements. These KPIs can be measured either before or after installation. Before installing game testing KPIs, we look at the load a game app can take, the average bootup time, frame rate analysis, etc.

However, since the environment inside the organization is close to optimum, the after-install indicators (that come directly from the real-world environment) are much more helpful than the before-install indicators.

Therefore, this section will focus on the same with measurements that help testers optimize app quality.

- **App crashes per n users:** The most important KPI we need to analyze is how often our application crashes in the real world. This directly affects the business, revenue, and reputation, and therefore, an app crash report is often directly sent to the developers/testers.

  The testers can analyze the crashes per n users where n can depend on the sensitivity the organization can withstand. An increasing number over time is a matter of concern and should be raised with appropriate teams.

- **Bug reports per n users:** Many users send bug reports when they feel something is wrong. Since this number is always less, we should always provide special attention to the bug a user has sent by spending so much effort. A game application is complex, so bugs are expected out of it. However, testers should monitor the number of bugs we are getting over a particular period and the rate of increase, if any.

- **Average game running time:** The average game running time represents the average time one user operates on the application. This factor directly relates to the engagement and excitement factor we tested through beta testing, freelance

gamers, internal testers, and other methods. But this is the real account of how engaging our game is, and a decreasing average running time needs deeper analysis from different teams.

- **Average money earned:** The average money earned shows the business growth and how we attract users toward additional paid offerings. An increasing graph is what every organization aims for as it contributes towards the business revenue.

  However, a decreasing graph can point towards an existing bug that possibly no one had reported. A deeper analysis can lead us to the root cause and again analyze this key performance indicator in the future to understand if the cause was the same as we thought or if something else is in the way.

These four KPIs indicate the current situation of our application in the real world and its probable future if it continues to be the same way. Leaving these KPIs unattended can make the application unpredictable, which could lead to the consumption of more resources that no organization would want.

Game Testing Tools

As a game developer, it's crucial to thoroughly test your video game to ensure it runs smoothly and can handle many players without crashing.

Remember, your video game can suddenly become popular worldwide and attract many players. That's why you must ensure your game can handle all those players without issues. And this is where game testing tools come in handy. Let's look at some popular ones that can help ensure your game is top-notch.

- **LambdaTest:** LambdaTest is a cloud-based testing platform to test game applications. It is an invaluable asset in your game testing journey, offering various features tailored to enhance your testing efforts.

  With LambdaTest, you gain access to a real device cloud with access to 3000+ browsers, devices, and platforms. This enables you to conduct comprehensive game testing on real Android and iOS devices, ensuring your game performs flawlessly in real-world scenarios.

  Additionally, LambdaTest provides the flexibility to test your games on a virtual testing cloud, encompassing Android

Emulators and iOS Simulators. This virtual environment allows you to simulate various device configurations, further expanding your testing possibilities.

By leveraging LambdaTest, you can evaluate your video games, identifying and addressing issues that could impact gameplay quality. This thorough testing process ultimately results in a higher-quality gaming experience for your players.

- **Appium:** Appium is a test automation framework to test your mobile application. It's a user-friendly tool for testing games, especially those with lots of graphics, like video games. The best part is that Appium is cross-platform compatible and works on Android and iOS devices. This makes testing games quicker and more effective.
- **Hatchet:** Hatchet is a cloud-based testing tool that allows you to test your game on various devices. It offers automated testing and helps developers identify compatibility issues across different hardware configurations. This helps developers find and fix glitches and bugs during the early stages of development, ultimately improving the overall performance and video game quality.

- **Valgrind:** Valgrind is a video game testing tool that can significantly enhance your quality assurance needs for game applications. This is particularly helpful when your video game is developed using C or C++ programming languages. It facilitates robust profiling and debugging within applications developed in these languages. Developers leverage Valgrind to identify and rectify memory leaks and other intricate low-level issues within their video games.
- **GameBench:** GameBench stands as a prominent tool in the world of video games and similar applications. Its primary role revolves around collecting, analyzing, and improving performance data. Specifically designed for mobile games, GameBench offers a complete solution for evaluating how well games work on various devices.

  This tool closely monitors essential aspects of video games, like how much memory they use, how quickly they drain your device's battery, and how smoothly they display images (known as frame rates).
- **BugSplat:** BugSplat is a powerful tool that helps you quickly find and fix issues while testing games. You can figure out what's causing issues that mess up the game and

make it unstable.

Using BugSplat, you can identify and work on the most critical issues within minutes, ensuring your game is back up and running smoothly. BugSplat doesn't just stop at finding issues; it also helps you understand why they happened, making it easier to solve them.

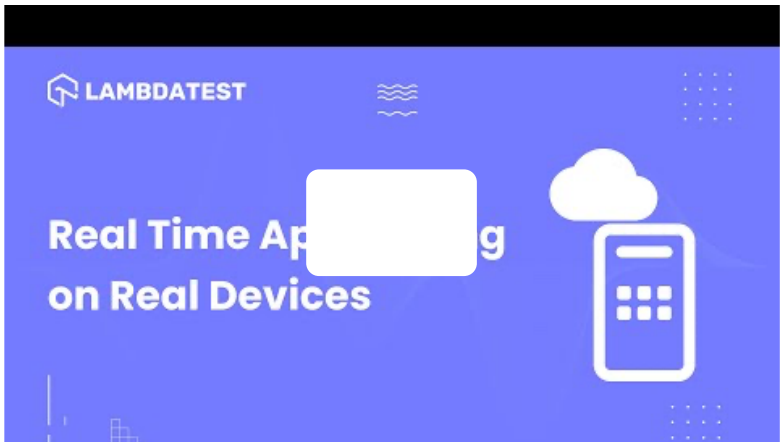
How to Perform Game Testing?

After going through all the theoretical analysis on game testing, the question is, how can we initiate the actual process and explore all the defects of a game? For this, we will use LambdaTest, an AI-powered test orchestration and execution platform, to perform manual and automated testing at scale.

As an introduction to LambdaTest, consider it as a companion in your game testing journey that asks for your minimum time and provides you with all the requirements to conduct it successfully. It provides an online device farm of

real browsers, devices, and platforms for game testing. Moreover, you can even test your games on a virtual testing cloud of Android Emulators and iOS Simulators.

You can also automate your game testing with LambdaTest using frameworks like Appium, Espresso, and XCUITest.



Subscribe to the LambdaTest YouTube Channel and stay updated with the latest tutorials around mobile app testing , Appium automation , and more.

As a cloud-based solution to all the testing problems, LambdaTest does not need heavy infrastructures and a network of systems to run the tests. It just requires a sign-up from your side, which is free for everyone.

Once sign-up is done, log in with your

credentials to open your LambdaTest Dashboard.

Since games can exist on an online URL or native mobile applications, you can choose the method by which you wish to test your game application. You can also upload apps from the Google Play Store, Apple App Store, and more.

For this demo, we will pick the native mobile game app and test it on real devices.

1. To start game testing, go to **Real Device > App Testing** from the left toolbar to get access to real mobile devices.

2. Here, you can upload your game app and select the device and OS on which you wish to run the test. Once the app is uploaded, hit the **Start** button.

The test session will start, and the application will be installed on the selected device specification.

From here, the testers can navigate through the game as they would on a physical device. They can also use third-party integrations or other popular tools for an enhanced experience from the same screen.

With that being mentioned, it is also true that game testing can be done on physical devices as well. But the problem is the number of devices you have to test in. Throughout this game testing tutorial, we focused on how risky it is to deploy a game application without proper testing. This results in multiple testing phases, and the testers must be sure about the performance of all the devices they release.

This is a cost-heavy task, and adding it with regular maintenance and

procurements, many organizations would hesitate to go forward. Hence, a cloud-based solution is an optimum choice where you don't need to worry about anything to set up, execute, or maintain from one point to another. It is lighter on the pocket, and with real devices on board, the experience is similar to a physical device.

Challenges in Game Testing

After exploring all the corners of game testing and its execution, we also need to prepare for the challenges we will face. Although what may be termed a "*challenge*" is personal to a team member and organization, the most frequently troubling ones are listed below.

- **Device range:** In the section "*Types of*

Games," we explored the range of devices a game is developed for. Depending on their resources, the organization can develop either of the options or a combination of them.

The challenge is the number of devices we encounter when we include desktops and mobile in the target devices. Hundreds of manufacturers keep releasing new versions of mobile devices, OS, browsers, desktops, and everything that needs an upgrade. This makes our target device list huge, and testing something as big as a game application on all of those is a challenge.

However, it is an easy way through for consoles and VRs as there are only a couple of devices, but by limiting the target devices, we also limit the number of users.

- **Minimum specs and maximum delivery:** Android applications demand huge resources from the device, especially primary and secondary memories. While it certainly depends on how a game application is built, a large part of RAM and space use is how a device uses them during the run. Currently, manufacturers provide

RAM as low as 4 GB, but it is insufficient. Testers are required to test the device RAM usage to determine if the application will lag or not in the documented devices. A simple RAM test is shown below:

A lot of the experts today recommend 8GB of RAM for use in Android, but that just restricts the number of users we can cater to. This is a huge challenge for a tester as developers often provide the minimum requirements based on the APIs and technologies they are using.

They pick it up from the official channels, which is theoretical in many ways. Having to measure RAM usage on various functions and accurately identifying the weak areas is a tough job.

- **Too many things to test:** Consider the following image taken from Call of Duty:

Modern Warfare for mobile:

Source

Take a few minutes and just note how many elements are there on this home page. The buttons we see are each one particular function that will take us to another screen and is a large functionality.

It will contain multiple other functionalities also. Add this to the functionalities that will be active when we start the game including the complexities involved in the main character. It is obvious that testing a game application is overwhelming, and there are just too many pieces to consider. It is a challenge we cannot ignore or prevent but just have to deal with to ensure a high-quality game for the user.

- **Latency and network issues:** The latency in the network and other issues, such as weak

bandwidth that may arise due to the user's poor network, is a challenge to test for a tester. Such parameters are hard to mimic and provide inaccurate reports for developers to optimize algorithms.

Source

On other software, we can put a loader, and the user can just wait, latency as small as 500ms can potentially render the graphics a lot later, which is frustrating for the game user, especially in a multiplayer environment.

- **Finding perfect testers:** In this guide, we tried to find a balance of skills between a tester and a gamer required to test a game application perfectly. But it may seem more challenging than it is.

  A tester may be technically excellent but not have used a game application in a decade. Or we may have a hardcore game

app user who needs to learn about a testing framework. Such job requirements are hard to satisfy, and as a result, we get delayed testing.

- **Multiplayer testing:** A multiplayer game provides the same scenario to multiple real game users but from their point of view. For instance, as shown in the following image,

Source

Each player has their own game but is also part of a team and a common experience. Such a thing is hard to develop and test because of the accuracy and consistency required in this playing style.

If person A shoots someone, another person B should also see that on their screen simultaneously if there are no network lags. Or if person A is racing with person B, their position should be

accurately seen on both screens. Therefore, the number of scenarios is huge and even greater when the number of players who can play together increases.

Testing a multiplayer game for perfect accuracy is challenging because of the limited resources to prove the similarities between the two.

- **Experience testing:** Finally, the trickiest challenge in game application testing is to measure the excitement and engagement level of the application from a user's point of view. In general, organizations do everything they can because a game application would work only if it is engaging enough.

  They hire professional testers, approach normal users, and the testers themselves measure the engagement through their own experience. But all of this is still not enough just because the sample size is too small, and the experience of one could be hugely different than the experience of another user. When millions of users play the game, can counting the experience of a dozen people give us accurate measures? This is a challenging question to answer, and testers are never sure

about it.

Summary

Game testing is a complex and nondeterministic path that testers take without knowing what journey they are embarking on. As if the extraordinary number of functions a game possesses is not enough, the number of platforms that can process a game application has also increased over the years. All of this makes game testing a challenging but exciting atmosphere worth experiencing if you are engaged in games and have an eye for finding defects.

This guide uncluttered the complex net of game testing by exploring its strengths, weaknesses, and often hidden areas unless testing is started. Hope this game testing guide helps you for your next adventure with game applications to test them better.

About the Author

Harish Rajora, He is a computer science engineer. He loves to keep growing as the technological world grows. He feels there is no powerful tool than a computer to change the world in any way. Apart from his field of study,

he likes reading books a lot and write sometimes on Twitter.

Frequently asked questions

General

How does one become a game tester?

What do you mean by game testing?

Which tool is used for game testing?

Author's Profile

Harish Rajora

Harish Rajora, He is a computer science engineer. He loves to keep growing as the technological world grows. He feels there is no powerful tool than a computer to change the world in any way. Apart from his field of study, he likes reading books a lot and write sometimes on Twitter.

Hubs: 15

Reviewer's Profile

Salman Khan

Salman works as a Product Marketing Manager at LambdaTest. With over four years in the software testing domain, he brings a wealth of experience to his role of reviewing blogs, learning hubs, product updates, and documentation write-ups. Holding a Master's degree (M.Tech) in Computer Science, Salman's expertise extends to various areas including web development, software testing (including automation testing and mobile app testing), CSS, and more.

Did you find this page helpful?

More Hubs

14 Chapters

Automation Testing

The quintessential guide to Automation Testing. This is an in-depth resource covering all aspects of testing.

27 Chapters

Selenium Python

Want to learn automation testing with Python? Check out our step-by-step tutorial on Python with examples!

12 Chapters

Selenium Locators

Here we explore different types of Selenium locators and learn how they are used with different automation testing.

9 Chapters

Selenium NUnit

Isn't it better to use an effective test framework like NUnit on a powerful and scalable Selenium Grid?

10 Chapters

Selenium WebdriverIO

Through this guide, we will learn how to use WebdriverIO, a next-gen test automation framework for Node.js.

8 Chapters

Selenium C#

This tutorial will teach how to master Selenium, making your test automation more streamlined and efficient.

Real Device Cloud
Test Manager
Mobile App Testing
AI-Powered Testing
HyperExecute
Performance Testing
LT Browser
LT Debug
Local Page Testing
Automated Screenshots
Geo-Location Testing
Accessibility Testing
Responsive Testing
Localization Testing
Visual Regression Testing
Integrations
Test Analytics

Safari Browser Online
Microsoft Edge
Opera
Yandex
Mac OS
Mobile Devices
iOS Simulator
Android Emulator
Browser Emulator

Playwright Testing
Puppeteer Testing
Taiko Testing

Mobile App Automation

Appium Testing
Espresso Testing
XCUITest Testing

Newsletter
Webinars
Videos
FAQ
Web Technologies Compatibility
Automation Testing Advisor
Software Testing [Glossary]
Free Online Tools
Mobile Testing Advisor
Sitemap
Status

Community & Support
Partners
Open Source
Content Editorial Policy
Write for Us
Become an Affiliate
Terms of Service
Privacy Policy
Trust
Team
Contact Us

Appium Tutorial
Jest Tutorial
More Learning Hubs

What's New

Changelog
Webinar:Achieving Continuous Quality Through People, Process, Tools and Culture
Future of QA Survey Report
Test on iPhone 16
Test on Samsung Galaxy S24 Series
September'24 Updates
Coding Jag - Issue 211
Kayak [Case Study]

Professional Services

Python Asyncio [Blog]

No-Code Test Automation [Hub]

Appium 101 [Certification]

LambdaTest is #1 choice for SMBs and Enterprises across the globe.

Deliver unparalleled digital experience with our Next-Gen, AI-powered testing cloud platform. Ensure exceptional user experience across all devices and browsers.

LambdaTest has formal standards certification and comply in line with acts and regulations across the globe.

Start free with Google

Start free with Email

Cross Browser Testing Cloud Built With For Testers