Table 1

1	Form testing		
	1/	registration	
		User with exact data exist in the system	
		User with exact data Dont exist in the system	
		User that were blocked in the system can't make registration again	
	2/	autorization	
		User exist in the sys with exact login and password	
		User doesnt exisct in the sys with exact login and password	
		User exist but the Password is incorrect	
		User is exist but was blocked by moderation	
		Validization of all fileds	
	3/	validization of all requirement fileds	
		Max Min symbols	
		Spec symbols. What symbols are accepted	
		Required fileds *	
		* asteriks is visile for req fileds	
		System doesnt show error massage for not req fields when they are empty	
	4/	feedback forms	
	5/	link for user agrement	
2	Search		
		Results are found and exicst / or not	
		Correct message about empty result	
		Empty search request	
		emodzy search / pic	
3	Fields	Test the terms of use and frequently asked questions: they should be clear and accessible to the user.	
		Number fields: dont process the letters. And show error massge	Mandatory input
		Fractions 1.1 or 1,1 how this numbers are validizing	Handle only whitespace
		Negatie numbers	Using spaces in text (before, after, and inside; spaces at the beginning and end of the line must be truncated when saving)
		division by 0 Деление на ноль корректно обрабатывается.	Numeric data format (if allowed): negative, fractional with a dot and comma, with a dot and a comma 123.123.123,00)
		Max lenght of the field. The data can be cut out	Entering tags and scripts (validation should be carried out only in the user part. The entered tags should be displayed in the same form in which they were entered)
		Spec symbols for every filed	Use of special characters (entered characters should be displayed in the same form in which they were entered, unless the input of special characters is prohibited by the requirements of the application)
		Text doesnt cross the filed borders	Ability to edit entered values
		Make sure a confirmation message is displayed for update and delete operations.	Correct distribution of text across lines (automatic line break)
		Make sure the values are displayed in the correct currency.	Unique data (for example, the uniqueness of the login)
		Test the numeric fields: they should not accept letters, in which case an appropriate error message should be displayed.	Automatic positioning of the cursor in the first input field when opening a form
		Make sure leap years are validated correctly and do not cause calculation errors.	

		Test the timeout functionality.	
		Test that division by zero is calculated correctly.	
4	pop-up field		
		Max symbols	
		confirmation messages displayed for update and delete operations.	
		Correct selection of the page background (if the background of the page changes when the pop-up appears, then when the page scale is changed, the background should fill the entire page - the size of the changed background corresponds to the page size)	
		Fixed pop-up position (dynamic change of position) in the case of using a scroll	
		Messages about input error	
5	Filters		Testing Tables
		Test the functionality of the filtration (ascending, descending, newest)	When several pages appear, there are buttons Forward, Back, To the first, To the last page (pagination)
		Ajust filters by all req	Checking sorts (+ checking sorting by default)
		Задать фильтры, по которым нет выдачи.	Filtration check (if possible)
		Filters by category / subcategory.	Updating table values after adding / changing / deleting data
		Filters with search radius.	Single / multiple selection of multiple values
		Data in drop-down lists.	
		Test all the data in the dropdown lists: they should be in chronological order.	Radio buttons
6	buttons	Test the functionality of the available buttons.	Functionality (on / off)
		Absence of calling the same action again when pressing the button several times Отсутствие вызова одного и того же действия повторно при нажатии на кнопку несколько раз	There can be no less than 2 radio buttons
		Unavailable buttons are not hidden, but blocked Недоступные кнопки не скрыты, а заблокированы	By default, one radio button should be enabled
		Pressing the space between closely spaced buttons should not trigger an action Нажатие на пространство между близко расположенными кнопками не должно приводить к действию	More than 1 radio button cannot be enabled
7	Наличие favicon.		When switching to the next page and returning back, the selected radio button should not be reset
8	error	Checking the handling of various errors (page not found, timeout, server error, etc.).	
9	downloads	Test that all downloaded documents open correctly.	Mandatory file selection
10		The user can download / attach / upload files / media (pictures, videos, etc.). And also remove these files from attachments. Make sure that files go to the server only after clicking the appropriate button	
		Formats of uploading files	
		Size restrictions	
		If there are no images, there should be a corresponding thumbnail, or the image should not be displayed at all	
		Size control (height / width), must be resized	
		Loading executable files (EXE, PHP, JSP etc.). Renamed EXE	
11		Test the email functionality of the system	
12	Caches, cookies and sessions		
		User deleted cach of the browser	
		User deleted cookies while were on the website	
		See what happens if the user deletes cookies after visiting the site.	

		Test a site with disabled cookies	
		Test a site with enabled cookies	
		Verify the cookie is encrypted before being written to the user’s machine	
		Check the security aspects when removing the cookies	
		If the cookies have a duration of action, then it is tested whether they are active in the specified period of time.	
13	DevTools	Elements tab	All errors in HTML with picker tool. All styles CSS are loaded. Html + CSS Parsing and showing a good pic for costumers. Change the language (Internationalization)
		Toggle device ToolBar	For Mobile devices too. Throttling. Offline connection. Screenshot. Responsive dimensions: Web page adjusts when you move it. Adaptive design: Does not move. Point centering.
		Console Tab	Can save the logs for bugs.
		Sources Tab	Files from server. JAVASCRIPT
		Network	HTPP request to server. +Look with Postman send requests All; Headers Preview Response Timing
		Performance	Clear cash, cookies. Page response time graph
		App	Local Storage Session storage Indexed DB Web SQL Cookies Trust token Cache Cache storage
		Lighthouse	Gererate reports about performance
		AdBlock	Turn off Adblock
14			
		Test that if the functionality fails, the user is redirected to a custom error page.	
		Test the terms of use and frequently asked questions: they should be clear and accessible to the user.	
		Test that Java Script works correctly in different browsers (IE, Firefox, Chrome, Safari, Opera).	
		Ability to change the browser window	
15	Links	Outbound links	
		Internal links correctness	
		There are no links leading to the same page	
		The links that are used to send e-mails to site admins	
		If there are pages that not referenced	
		There are no broken links	
16	HTML/CSS validation	HTML syntax errors	
		Verify the site is available for search machines	
		Verify your web page has an accurate site map in both XML and HTML format	
17	Menu		
		Making the appropriate jump when selecting a menu item	

		Visual difference at the time of working on a specific tab (highlight, underline)	
		Scrolling testing.	
		No scrolling if the text fits on the page without scrolling.	
		Corresponding text changes when using a scroll.	Other
		The ability to change the position of the scroll using the mouse, Page up / down, Home / End buttons.	The user must be informed about the actions taking place in the system by means of messages about the successful completion of the operation.
			For irreversible actions, such as deletion, there must be confirmation messages
			404 Error (following an incorrect URL should lead to a 404 error page, not just a Page cannot be found. The 404 error page should be implemented in the overall design of the application under test)
		Testing calendars.	Tab order (top to bottom left to right). Fields available for reading and and assigned should be skipped
		If you can enter a date manually, you need to check the different formats	Logo should be a link to the main page
		Input consistency checks (dates in the future, etc.)	Focus on the button for performing actions (data entry -> pressing Enter -> the action was performed)
		Leap year check	Checking Breadcrumbs ("Breadcrumbs" is a navigation element that is a sign of ease of use of the site as a whole and movement through its structure)
			Decrease / increase page scale (elements should be redistributed accordingly, while maintaining proportions)
			Displaying flash elements when the flash player is disabled / not installed in the browser (the user should be prompted to download and install the latest version of the flash player; an alternative image should be displayed in place of the flash object)
			Performance check: sending emails, notifications (both to the admin and to the user), unless the absence of letters is a specific project
	EMAIL		
	Empty filed email		
	Email min characters		
	Email max characters		
	Email with numbers in the name of account		
	Email with numbers in the domain part		
	- hyphen in the email		
	-hyphen in the domain part		
	. Period in the name of email		
	No period in the email (error)		
	More than max characters		
	No @ in the email		
	Space in the email		
	Empty input		
	No domain name		
		Login Password validation	
		Fill in the field with correct login and correct pass. Expected: Logged in successfully. Log out. Clear cache and cookies (open / close browser).	
		Leave both fields blank. Attempt to sign in. Expected: Error message.	

		Leave the login field empty. Attempt to sign in. Expected: Error message.	
		Leave the password blank. Attempt to sign in. Expected: Error message.	
		Enter the correct login and incorrect pass. Expected: Error message.	
		Enter incorrect login, but correct pass. Expected: Error message.	
		Enter incorrect login and incorrect pass. Expected: Error message.	
		Fill in the login field, enter the correct pass, and enter the correct login in the password field. Expected: Error message.	
		Enter login <script> alert (123) </script> and correct pass. Expected: Error message.	
		Fill in the login SQL query field (‘or’ a ’=‘ a ’; DROP TABLE user; SELECT * FROM blog WHERE code LIKE‘ a% ’;) - the structure of the request depends on the DB.	
		Fill in the login script field (<script> alert (“Hello, world!”) </alert>, <script> document.getElementByID (“…”). Disabled = true </script>)	
		Fill in the login html-tags (<form action = ”http://live.hh.ru”> <input type = ”submit”> </form>)	
		Fill in the login field with a complex sequence of characters like “♣”, “” '~! @ # $% ^ & * ()?>,. / \ <] [/ * <! - “”, “$ {code}”; - >	
		Fill in the login field with a text consisting of only spaces;	
		Fill in the login field with the correct login, starting with multiple spaces, and the correct pass. Expected: Error message or auto-truncate whitespace.	
		Fill in the login field with the correct login followed by a few spaces, and the correct pass. Expected: Error message or auto-truncate whitespace.	
		Enter the correct login and correct pass. Click on the "Back" button in the browser. Expected: or The page should be expired, or see the same fields. If the second, enter login and pass in the fields again. Go. Signed in?	
		Enter the correct login. Specify pass using letters of DIFFERENT case.	
		Enter login using different case letters. Specify correct pass.	
		Check the limitation on the length of login and password when registering? Enter qqweqweqweqweqweqweqweqweqweqweqweqweqweqwe / qqweqweqweqweqweqweqweqweqweqweqweqweqweqwe	
		Fill in the login / pass field Aa! @ # $% ^ & * () -_ + = `~ / \,.?> <| B / PaSSword! @ # $% ^ & * () -_ + =` ~ / \,.?> <| Are there any restrictions on valid characters?	
		Open the first browser. Log in as a valid user. Open a second browser. Log in with the same valid user. Log out in the first browser. Switch to the second browser. Do something that only a logged in user can do.	
		Open a browser. Enter valid data in the fields. Click on the Login button. Disconnect internet. Get “page unavailable”. Connect the internet back. Go on the website. Expected: not logged in.	
		Is the account / IP blocked for the one who enters the wrong pass n times?	
		Set focus to login field. Enter text. Press the Tab key on your keyboard. Expected: Focus moves to the password field. Enter text. Press the Tab key on your keyboard. Expected: focus moves to the “remember me” checkbox. Press the Space button on the keyboard. Expected: a check mark has appeared. Press the Tab key on your keyboard. Expected: Focus moves to the Login button. Press the Enter button on the keyboard. Expected: the process has started.	
		Check for ‘Remember me on this computer’. Fill in the fields with valid data. Check Remember me. Login. Close the browser. Open browser. Open the site page. Expected: login is not required to login.	
		Enter the correct login and correct pass. Copy the received url and paste it into another browser. Expected: It should not display the user’s welcome page.	

1			
2			
3			
	Testing field list Тестирование полей со списком.		
	Sort alphabetically or by meaning		
	If the values go beyond the boundaries of the list, and there is no possibility of increasing the size of the list, then display of hints (tooltips) is required		
	Selecting a list item by pressing the corresponding first letter on the keyboard		
	Ability to select multiple values for a combo box		
	The ability to enter values manually (if the application allows it)		

4			
5			
	(check boxes).		
6	Functionality (on / off)		
	Mandatory selection of at least one checkbox		
	The presence of an additional checkbox that displays / removes all checkboxes if there are more than 10 checkboxes		
	When switching to the next page and returning back, the selected radio button should not be reset		
7			
8			
9			
10			
11			
12			

13			
14			
15			
16			
17			

Functional Testing Mobile	
1. Installation / removal / rolling of versions	
2. Launching the application (displaying Splash Screen)	**GUI**
3. The performance of the main functionality of the application	Text boxes, buttons, menus and icons function accurately
3.1 Authorization (by phone number / via social networks / e-mail)	Push notifications render correctly and appear at the right intervals
3.2 Registration (by phone number / via social networks / e-mail)	Any transactions or purchases should happen seamlessly
3.3 Onboarding new users	Elements need to be usable size for user to press
3.4 Validation of required fields	No empty windows in app
3.5 Navigation between application sections	Multiple press on buttons
3.6 Editing data in the user profile	Cheking native gestures in app
3.7 Checking payment	work with situations with no space to work for app
3.8 Testing Filters	elements size
no space for installation	ammount of the information within page
supporting functions check - 3G, SD card	diff sizes of the screen adaptations
install or transfer app on sd card	portraiit / landscape orientation
compliance of the rules - Apple hig, google material design	Responsivness to the impact Graphic, sounds, tactile
3.9 Bonuses	Messages about errors
4. Correct display of errors	Colors scale
5. Working with files (sending / receiving / viewing)	RETINa and other simple windows. Retina Display is a brand name used by Apple for its series of IPS LCD and OLED displays that have a higher pixel density than traditional Apple displays.
6. Testing timeouts	Screen size and touch interface
7. Testing stubs (no internet connection / no, for example, goods, etc.)	convenient size of buttons so that you don't have to look for it on the screen and hit it the third time
8. Testing pop-ups, alerts	element response speed (high; the pressed key should be visually different)
9. Testing WebView	
10. Scroll / swipe elements	
11. Testing PUSH notifications https://software-testing.ru/library/testing/mobile-testing/3595-push-notifications-in-mobile-apps	
12. Minimizing / expanding the application	
13. Different types of connections (cellular / Wi-Fi)	
14. Screen orientation (landscape / portrait)	
15. Dark / light themes	Version OS they cant be installed on unsupported OS
16. Advertising in the app	Compilance of screens in diff apps
17. Sharing content in the social. the network	Incoming calls, message, notifications
18. Application work in the background	Turn off device, eject battery
19. Page pagination пагинация	Standby mode(with password)
20. Privacy policies and other links to documents	Changing of orientation in standby mode
21. Cross platform	On/Off of the network, GPS, Avia mode, bluetooth
Interruption Testing:	On/Off Sd cards, keyboard, garniture
This form of mobile testing checks how an application responds when faced with an unexpected interruption. Depending on the nature of the interruption, the application should pause and then return to its original state, or even react in a particular way. Obviously, the kind of interruptions will differ on the basis of the application under test, but some common interruptions that should be considered while testing are:	Price matching and content
Incoming or phone call when an application is running	
Incoming message or SMS when an application is running	
Low battery when an application is running	use diff functionalities of the device (Back button on Android)
The device plugged in or out of charging when an application is running	Translation errors
Device shutting down when an application is running	Check date format

OS upgrade occurring when an application is running	Checking diff time zones
Loss and restoration of the network while an application is running. Interruption testing ensures that an app handles interruptions without failure or anomaly. When being used by real users, every app will have to operate along with other device functions. This means that every app will have to be optimized to deal with these device functions while running at all times.	Check that new functionality works on older versions of OS
	Updates checking (Saving profile data, autorization)
Localization Testing:	transition between screens
This approach to mobile app testing is meant to test features that are dependent on the geographical location of an app. Since most apps seek to appeal to a global user base, they include localized features for the convenience of users. These features can vary from enabling different languages, enabling commerce in local currency to adherence with local laws and regulations. Localization testing checks these features to ensure that they are activated and functioning in the right locations. Customers always prefer apps with UI elements aligned with their culture, language, and device accessibility. They expect their experience to be adjusted to their localized needs and preferences. AppAnnie's research confirms that fully localized apps do better in the global market. Localization testing is also one of the most challenging mobile app testing types since most QA teams lack adequate access to test coverage and resources necessary for its implementation.	Chancelaration of the requests if they not finished
1. All elements in the application are translated into the needed language	
2. The texts are protected inside the application and the user in the application settings can set the required language	
3. The texts depend on the language in the system settings	
4. Texts come from the server	
5. Correct display of date formats (YEAR - MONTH - DAY or DAY - MONTH - YEAR.)	
6. Correct display of time depending on the time zone	
in another language, there must be enough room for text on the screen	
dates must match the format of the specified region	
temporary settings must be respected	
Memory Leak Testing:	Use developers setting - imitation of bad connection, dont save the action

A memory leak refers to a situation in which the app fails to return the memory it has acquired for temporary use in order to function. The available memory for the app drains, and the app cannot function. If an app is frequently used or opened, a small memory leak can result in its termination. Memory leaks emerge from programming bugs, so every app needs to be tested for this issue. Memory leak testing is done by running an app on multiple devices. By doing so, testers can check the app performance on devices with different memory capabilities, and optimize the app to function effectively on each configuration.	Deinstallation - delete all user data
can be checked with Instruments (standard MacOS application). Can be no more than 30MB for a 2g iPhone / iPod, about 70MB for all devices up to the 2nd iPad	
pay attention to windows with a lot of information, during a long stay of the user in the application	
Usability Testing:	
Also known as user experience testing, this checks an app for user-friendliness. Basically, it checks ease of use and intuitiveness, aiming to provide a seamless user experience that is free of bugs and anomalies. Since the success of an app depends on the appeal of its end-to-end user experience, it is best to do usability tests with actual customers on real devices. This is the best way to understand the preferences of the target audience. Conversely, one can have skilled testers running user scenarios that mirror the behavior of actual end-users. A few pointers to keep in mind during usability tests:	
Smooth, visually appealing layout and design	
A high degree of intuitiveness	
Quick response time – Most users prefer apps that launch within 2-3 seconds after tapping the icon.	
Correct display of elements on devices with different screen resolutions	
All fonts meet the requirements	
All texts are correctly aligned	
All error messages are correct, without spelling and grammatical errors	
Correct screen titles	
There are placeholders in the search lines	
Inactive elements are displayed in gray	
Links to documents lead to the corresponding section on the site	s
Animation between transitions	
Correct return to the previous screen	
Supports basic gestures when working with touch screens (swipe back, etc.)	
Pixel-perfect	
Checking the operation of applications on retina screens and various OS versions	
correct display of various elements on retina / non-retina screens	

installing the application on the correct OS version	
check the installation for all possible devices	
various functions on devices: absence / presence of a camera (ipad) (autofocus), absence / presence of GPS	
Performance Testing:	
It is important to test how an application performs under various conditions. This is where performance testing comes in. It puts the device under various forms of pressure so that it does not malfunction in non-optimal conditions. A few things that performance testing should verify:	High load of the central processor
Device performance:	Out of memory
Installation and log-in time, battery consumption, memory consumption, etc.	Loading the battery
Network performance:	Refusals
Delays, errors, pauses in receiving digital information or rendering network-activated features	Low network bandwidth
API/Server performance:	A large number of user interactions with the application (this may require simulating real network conditions)
Speed and formation of data transfer from back-end to front-end	Application load time
Recovery capabilities:	Processing requests
Built-in back-up and recovery functions that can save or recover user data in the event of data loss.	Data caching
Security Testing:	The consumption of resources by the application (for example, battery consumption)

App users are becoming increasingly conscious of issues surrounding data security. Online privacy and confidentiality of personal data are major concerns for most netizens – 70% report being concerned that their personal information will be shared without permission. In fact, 81% of users say they would uninstall an app and switch vendors because of security concerns. Needless to say, security testing is imperative to the success of an app. Since almost every app requires some kind of personal information to run, tests must be conducted to fortify them, in order to provide confidentiality of data. QAs must thoroughly check that the application is able to defend its users from having their information leaked or hacked in. This is especially true of financial apps	
1. Testing permissions (access to camera / microphone / gallery / etc.)	
2. User data (passwords) are not transmitted in clear text	
3. In the fields with password entry and password confirmation, the data is hidden by asterisks	
Compatibility	
Compatibility testing is used to ensure that your application is compatible with other OS versions, various shells and third-party services, and device hardware.	
1. Correct display of geo	
2. Information about transactions (checks, etc.)	
3. Various payment methods (Google Pay, Apple Pay)	
4. Testing sensors (illumination, device temperature, gyroscope, etc.)	
5. Testing interruptions (incoming call / SMS / push / alarm clock / Do not disturb mode, etc.)	
6. Connecting external devices (memory card / headphones, etc.)	
Checking the type of purchases (recoverable, not recoverable)	

verification of compliance with the actual / declared value of the application	
verification of the restoration of the purchase regardless of the device, but with the link to the account	
restoration of purchases	
Save of the purchases with app update	
Checking energy consumption	
you need to check how much your powerful application drains the battery of the device. Most likely, the user will delete it if it causes the mobile phone to be charged too often.	
These are the main points and features of mobile application testing that you should pay attention to when testing mobile applications. Each device is individual due to the parameters and configurations set by the user. But nevertheless, you should adhere to checking the above points on any device.	
Checking the work of the feedback	
content upload messages / progress	
network access error messages	
presence of messages when trying to delete important information	
the presence of a screen / message at the end of the process / game (Game over screen)	
Checking for updates	

checking various ways to install updates (wifi, bluetooth, usb)	
checking the work of the installed changes, the places where they were made	
make sure that updates are supported by older operating systems so that elements that work well on the new system do not crash on older versions.	
Checking the Application Response to External Interrupts	
incoming / outgoing sms, mms, calls	
battery discharge / removal	
network disconnect / wifi	
connecting cable, card, charging	
Advertising in the mobile application	
ads should not overlap application control buttons	
ads should have an available close button, because most often the user does not look for it, but simply deletes the application with the ends	

Functional Testing Mobile	
1. Installation / removal / rolling of versions	
2. Launching the application (displaying Splash Screen)	
3. The performance of the main functionality of the application	
3.1 Authorization (by phone number / via social networks / e-mail)	
3.2 Registration (by phone number / via social networks / e-mail)	
3.3 Onboarding new users	
3.4 Validation of required fields	
3.5 Navigation between application sections	
3.6 Editing data in the user profile	
3.7 Checking payment	
3.8 Testing Filters	
no space for installation	
supporting functions check - 3G, SD card	
install or transfer app on sd card	
compliance of the rules - Apple hig, google material design	
3.9 Bonuses	
4. Correct display of errors	
5. Working with files (sending / receiving / viewing)	
6. Testing timeouts	
7. Testing stubs (no internet connection / no, for example, goods, etc.)	
8. Testing pop-ups, alerts	
9. Testing WebView	
10. Scroll / swipe elements	
11. Testing PUSH notifications https://software-testing.ru/library/testing/mobile-testing/3595-push-notifications-in-mobile-apps	
12. Minimizing / expanding the application	
13. Different types of connections (cellular / Wi-Fi)	
14. Screen orientation (landscape / portrait)	
15. Dark / light themes	
16. Advertising in the app	
17. Sharing content in the social. the network	
18. Application work in the background	
19. Page pagination пагинация	
20. Privacy policies and other links to documents	
21. Cross platform	
Interruption Testing:	
This form of mobile testing checks how an application responds when faced with an unexpected interruption. Depending on the nature of the interruption, the application should pause and then return to its original state, or even react in a particular way. Obviously, the kind of interruptions will differ on the basis of the application under test, but some common interruptions that should be considered while testing are:	
Incoming or phone call when an application is running	
Incoming message or SMS when an application is running	
Low battery when an application is running	
The device plugged in or out of charging when an application is running	
Device shutting down when an application is running	

OS upgrade occurring when an application is running	
Loss and restoration of the network while an application is running. Interruption testing ensures that an app handles interruptions without failure or anomaly. When being used by real users, every app will have to operate along with other device functions. This means that every app will have to be optimized to deal with these device functions while running at all times.	
Localization Testing:	
This approach to mobile app testing is meant to test features that are dependent on the geographical location of an app. Since most apps seek to appeal to a global user base, they include localized features for the convenience of users. These features can vary from enabling different languages, enabling commerce in local currency to adherence with local laws and regulations. Localization testing checks these features to ensure that they are activated and functioning in the right locations. Customers always prefer apps with UI elements aligned with their culture, language, and device accessibility. They expect their experience to be adjusted to their localized needs and preferences. AppAnnie's research confirms that fully localized apps do better in the global market. Localization testing is also one of the most challenging mobile app testing types since most QA teams lack adequate access to test coverage and resources necessary for its implementation.	
1. All elements in the application are translated into the needed language	
2. The texts are protected inside the application and the user in the application settings can set the required language	
3. The texts depend on the language in the system settings	
4. Texts come from the server	
5. Correct display of date formats (YEAR - MONTH - DAY or DAY - MONTH - YEAR.)	
6. Correct display of time depending on the time zone	
in another language, there must be enough room for text on the screen	
dates must match the format of the specified region	
temporary settings must be respected	
Memory Leak Testing:	

A memory leak refers to a situation in which the app fails to return the memory it has acquired for temporary use in order to function. The available memory for the app drains, and the app cannot function. If an app is frequently used or opened, a small memory leak can result in its termination. Memory leaks emerge from programming bugs, so every app needs to be tested for this issue. Memory leak testing is done by running an app on multiple devices. By doing so, testers can check the app performance on devices with different memory capabilities, and optimize the app to function effectively on each configuration.	
can be checked with Instruments (standard MacOS application). Can be no more than 30MB for a 2g iPhone / iPod, about 70MB for all devices up to the 2nd iPad	
pay attention to windows with a lot of information, during a long stay of the user in the application	
Usability Testing:	
Also known as user experience testing, this checks an app for user-friendliness. Basically, it checks ease of use and intuitiveness, aiming to provide a seamless user experience that is free of bugs and anomalies. Since the success of an app depends on the appeal of its end-to-end user experience, it is best to do usability tests with actual customers on real devices. This is the best way to understand the preferences of the target audience. Conversely, one can have skilled testers running user scenarios that mirror the behavior of actual end-users. A few pointers to keep in mind during usability tests:	
Smooth, visually appealing layout and design	
A high degree of intuitiveness	
Quick response time – Most users prefer apps that launch within 2-3 seconds after tapping the icon.	
Correct display of elements on devices with different screen resolutions	
All fonts meet the requirements	
All texts are correctly aligned	
All error messages are correct, without spelling and grammatical errors	
Correct screen titles	
There are placeholders in the search lines	
Inactive elements are displayed in gray	
Links to documents lead to the corresponding section on the site	
Animation between transitions	
Correct return to the previous screen	
Supports basic gestures when working with touch screens (swipe back, etc.)	
Pixel-perfect	
Checking the operation of applications on retina screens and various OS versions	
correct display of various elements on retina / non-retina screens	

installing the application on the correct OS version	2. Check list iOS App Store Review Guidelines, https://developer.apple.com/app-store/review/guidelines/
check the installation for all possible devices	1. Safety
various functions on devices: absence / presence of a camera (ipad) (autofocus), absence / presence of GPS	**1.1 Objectionable ContentApps should not include content that is offensive, insensitive, upsetting, intended to disgust, in exceptionally poor taste, or just plain creepy. Examples of such content include:**
	1.3 Kids CategoryThe Kids Category is a great way for people to easily find apps that are designed for children. If you want to participate in the Kids Category, you should focus on creating a great experience specifically for younger users.
Performance Testing:	**1.4 Physical Harm**
It is important to test how an application performs under various conditions. This is where performance testing comes in. It puts the device under various forms of pressure so that it does not malfunction in non-optimal conditions. A few things that performance testing should verify:	**1.5 Developer InformationPeople need to know how to reach you with questions and support issues. Make sure your app and its Support URL include an easy way to contact you; this is particularly important for apps that may be used in the classroom. Failure to include accurate and up-to-date contact information not only frustrates customers, but may violate the law in some countries. Also ensure that Wallet passes include valid contact information from the issuer and are signed with a dedicated certificate assigned to the brand or trademark owner of the pass.**
Device performance:	**1.6 Data SecurityApps should implement appropriate security measures to ensure proper handling of user information collected pursuant to the Apple Developer Program License Agreement and these Guidelines (see Guideline 5.1 for more information)**
Installation and log-in time, battery consumption, memory consumption, etc.	**1.7 Reporting Criminal Activity must involve local law enforcement, and can only be offered in countries where such involvement is active.**
Network performance:	**2. Performance**
Delays, errors, pauses in receiving digital information or rendering network-activated features	**2.1 App CompletenessSubmissions to App Review, including apps you make available for pre-order, should be final versions with all necessary metadata and fully functional URLs included; placeholder text, empty websites, and other temporary content should be scrubbed before submission. Make sure your app has been tested on-device for bugs and stability before you submit it, and include demo account info (and turn on your back-end service!) if your app includes a login. If you offer in-app purchases in your app, make sure they are complete, up-to-date, and visible to the reviewer, or that you explain why not in your review notes. Please don't treat App Review as a software testing service. We will reject incomplete app bundles and binaries that crash or exhibit obvious technical problems.**
API/Server performance:	**2.2 Beta TestingDemos, betas, and trial versions of your app don't belong on the App Store – use TestFlight instead.**
Speed and formation of data transfer from back-end to front-end	**2.3 Accurate MetadataCustomers should know what they're getting when they download or buy your app, so make sure all your app metadata, including privacy information, your app description, screenshots, and previews accurately reflect the app's core experience and remember to keep them up-to-date with new versions.**
Recovery capabilities:	**2.4 Hardware Compatibility**
Built-in back-up and recovery functions that can save or recover user data in the event of data loss.	**2.4.1 To ensure people get the most out of your app, iPhone apps should run on iPad whenever possible. We encourage you to consider building universal apps so customers can use them on all of their devices. Learn more about Universal apps.**
Security Testing:	**2.4.2 Design your app to use power efficiently and be used in a way that does not risk damage to the device. Apps should not rapidly drain battery, generate excessive heat, or put unnecessary strain on device resources.**

App users are becoming increasingly conscious of issues surrounding data security. Online privacy and confidentiality of personal data are major concerns for most netizens – 70% report being concerned that their personal information will be shared without permission. In fact, 81% of users say they would uninstall an app and switch vendors because of security concerns. Needless to say, security testing is imperative to the success of an app. Since almost every app requires some kind of personal information to run, tests must be conducted to fortify them, in order to provide confidentiality of data. QAs must thoroughly check that the application is able to defend its users from having their information leaked or hacked in. This is especially true of financial apps	**2.4.4 Apps should never suggest or require a restart of the device or modifications to system settings unrelated to the core functionality of the app. For example, don't encourage users to turn off Wi-Fi, disable security features, etc.**
1. Testing permissions (access to camera / microphone / gallery / etc.)	**2.5 Software Requirements**
2. User data (passwords) are not transmitted in clear text	**2.5.1 Apps may only use public APIs and must run on the currently shipping OS. Learn more about public APIs. Keep your apps up-to-date and make sure you phase out any deprecated features, frameworks or technologies that will no longer be supported in future versions of an OS.**
3. In the fields with password entry and password confirmation, the data is hidden by asterisks	**2.5.2 Apps should be self-contained in their bundles, and may not read or write data outside the designated container area, nor may they download, install, or execute code which introduces or changes features or functionality of the app, including other apps.**
Compatibility	**2.5.3 Apps that transmit viruses, files, computer code, or programs that may harm or disrupt the normal operation of the operating system and/or hardware features, including Push Notifications and Game Center, will be rejected. 2.5.4 Multitasking apps may only use background services for their intended purposes: VoIP, audio playback, location, task completion, local notifications, etc. If your app uses location background mode, include a reminder that doing so may dramatically decrease battery life.**
Compatibility testing is used to ensure that your application is compatible with other OS versions, various shells and third-party services, and device hardware.	**2.5.5 Apps must be fully functional on IPv6-only networks.**
1. Correct display of geo	**2.5.6 Apps that browse the web must use the appropriate WebKit framework and WebKit Javascript.**
2. Information about transactions (checks, etc.)	**2.5.7 Video streaming content over a cellular network longer than 10 minutes must use HTTP Live Streaming and include a baseline 192 kbps HTTP Live stream.**
3. Various payment methods (Google Pay, Apple Pay)	**2.5.8 Apps that create alternate desktop/home screen environments or simulate multi-app widget experiences will be rejected.**
4. Testing sensors (illumination, device temperature, gyroscope, etc.)	**2.5.9 Apps that alter or disable the functions of standard switches, such as the Volume Up/Down and Ring/Silent switches, or other native user interface elements or behaviors will be rejected. For example, apps should not block links out to other apps or other features that users would expect to work a certain way. Learn more about proper handling of links.**
5. Testing interruptions (incoming call / SMS / push / alarm clock / Do not disturb mode, etc.)	**2.5.10 Apps should not be submitted with empty ad banners or test advertisements.**
6. Connecting external devices (memory card / headphones, etc.)	**2.5.11 SiriKit and Shortcuts**
	2.5.12 Apps using CallKit or including an SMS Fraud Extension should only block phone numbers that are confirmed spam. Apps that include call-, SMS-, and MMS- blocking functionality or spam identification must clearly identify these features in their marketing text and explain the criteria for their blocked and spam lists.
	2.5.13 Apps using facial recognition for account authentication must use LocalAuthentication (and not ARKit or other facial recognition technology) where possible, and must use an alternate authentication method for users under 13 years old.
	2.5.14 Apps must request explicit user consent and provide a clear visual and/or audible indication when recording, logging, or otherwise making a record of user activity. This includes any use of the device camera, microphone, screen recordings, or other user inputs.
Checking the type of purchases (recoverable, not recoverable)	**2.5.15 Apps that enable users to view and select files should include items from the Files app and the user's iCloud documents.**

verification of compliance with the actual / declared value of the application	**2.5.16 App Clips, widgets, extensions, and notifications should be related to the content and functionality of your app. Additionally, all App Clip features and functionality must be included in the main app binary. App Clips cannot contain advertising.**
verification of the restoration of the purchase regardless of the device, but with the link to the account	**3. Business**
restoration of purchases	**3.1 Payments**
Save of the purchases with app update	**3.1.1 In-App Purchase:**
Checking energy consumption	**If you want to unlock features or functionality within your app, (by way of example: subscriptions, in-game currencies, game levels, access to premium content, or unlocking a full version), you must use in-app purchase. Apps may not use their own mechanisms to unlock content or functionality, such as license keys, augmented reality markers, QR codes, etc. Apps and their metadata may not include buttons, external links, or other calls to action that direct customers to purchasing mechanisms other than in-app purchase.**
you need to check how much your powerful application drains the battery of the device. Most likely, the user will delete it if it causes the mobile phone to be charged too often.	**Apps may use in-app purchase currencies to enable customers to “tip” the developer or digital content providers in the app.**
These are the main points and features of mobile application testing that you should pay attention to when testing mobile applications. Each device is individual due to the parameters and configurations set by the user. But nevertheless, you should adhere to checking the above points on any device.	**Any credits or in-game currencies purchased via in-app purchase may not expire, and you should make sure you have a restore mechanism for any restorable in-app purchases.**
	Apps may enable gifting of items that are eligible for in-app purchase to others. Such gifts may only be refunded to the original purchaser and may not be exchanged.
	Apps distributed via the Mac App Store may host plug-ins or extensions that are enabled with mechanisms other than the App Store.
	Apps offering “loot boxes” or other mechanisms that provide randomized virtual items for purchase must disclose the odds of receiving each type of item to customers prior to purchase.
	Digital gift cards, certificates, vouchers, and coupons which can be redeemed for digital goods or services can only be sold in your app using in-app purchase. Physical gift cards that are sold within an app and then mailed to customers may use payment methods other than in-app purchase.
	Non-subscription apps may offer a free time-based trial period before presenting a full unlock option by setting up a Non-Consumable IAP item at Price Tier 0 that follows the naming convention: “XX-day Trial.” Prior to the start of the trial, your app must clearly identify its duration, the content or services that will no longer be accessible when the trial ends, and any downstream charges the user would need to pay for full functionality. Learn more about managing content access and the duration of the trial period using Receipts and Device Check.
Checking the work of the feedback	**3.1.2 Subscriptions: Apps may offer auto-renewing in-app purchase subscriptions, regardless of category on the App Store. When incorporating auto-renewable subscriptions into your app, be sure to follow the guidelines below.**
content upload messages / progress	**3.1.2(a) Permissible uses: If you offer an auto-renewing subscription, you must provide ongoing value to the customer, and the subscription period must last at least seven days and be available across all of the user’s devices.**
network access error messages	**3.1.2(b) Upgrades and Downgrades: Users should have a seamless upgrade/ downgrade experience and should not be able to inadvertently subscribe to multiple variations of the same thing.**
presence of messages when trying to delete important information	**3.1.2(c) Subscription Information: Before asking a customer to subscribe, you should clearly describe what the user will get for the price. How many issues per month? How much cloud storage? What kind of access to your service? Ensure you clearly communicate the requirements described in Schedule 2 of the Apple Developer Program License Agreement, found in Agreements, Tax, and Banking.**
the presence of a screen / message at the end of the process / game (Game over screen)	**3.1.3 Other Purchase Methods: The following apps may use purchase methods other than in-app purchase. Apps in this section cannot, within the app, encourage users to use a purchasing method other than in-app purchase. Developers can send communications outside of the app to their user base about purchasing methods other than in-app purchase.**
Checking for updates	**3.1.3(a) “Reader” Apps: Apps may allow a user to access previously purchased content or content subscriptions (specifically: magazines, newspapers, books, audio, music, and video). Reader apps may offer account creation for free tiers, and account management functionality for existing customers.**

checking various ways to install updates (wifi, bluetooth, usb)	**3.1.3(b) Multiplatform Services: Apps that operate across multiple platforms may allow users to access content, subscriptions, or features they have acquired in your app on other platforms or your web site, including consumable items in multi-platform games, provided those items are also available as in-app purchases within the app.**
checking the work of the installed changes, the places where they were made	**3.1.3(c) Enterprise Services: If your app is only sold directly by you to organizations or groups for their employees or students (for example professional databases and classroom management tools), you may allow enterprise users to access previously-purchased content or subscriptions. Consumer, single user, or family sales must use in-app purchase.**
make sure that updates are supported by older operating systems so that elements that work well on the new system do not crash on older versions.	**3.1.3(d) Person-to-Person Services: If your app enables the purchase of real-time person-to-person services between two individuals (for example tutoring students, medical consultations, real estate tours, or fitness training), you may use purchase methods other than in-app purchase to collect those payments. One-to-few and one-to-many real-time services must use in-app purchase.**
Checking the Application Response to External Interrupts	**3.1.3(e) Goods and Services Outside of the App: If your app enables people to purchase physical goods or services that will be consumed outside of the app, you must use purchase methods other than in-app purchase to collect those payments, such as Apple Pay or traditional credit card entry.**
incoming / outgoing sms, mms, calls	**3.1.3(f) Free Stand-alone Apps: Free apps acting as a stand-alone companion to a paid web based tool (eg. VOIP, Cloud Storage, Email Services, Web Hosting) do not need to use in-app purchase, provided there is no purchasing inside the app, or calls to action for purchase outside of the app.**
battery discharge / removal	***3.1.4 Hardware-Specific Content: In limited circumstances, such as when features are dependent upon specific hardware to function, the app may unlock that functionality without using in-app purchase. App features that work in combination with an approved physical product (such as a toy) on an optional basis may unlock functionality without using in-app purchase, provided that an in-app purchase option is available as well. You may not, however, require users to purchase unrelated products or engage in advertising or marketing activities to unlock app functionality.***
network disconnect / wifi	***3.1.5 Cryptocurrencies:***
connecting cable, card, charging	***(i) Wallets: Apps may facilitate virtual currency storage, provided they are offered by developers enrolled as an organization.***
Advertising in the mobile application	***(ii) Mining: Apps may not mine for cryptocurrencies unless the processing is performed off device (e.g. cloud-based mining).***
ads should not overlap application control buttons	***(iii) Exchanges: Apps may facilitate transactions or transmissions of cryptocurrency on an approved exchange, provided they are offered by the exchange itself.***
ads should have an available close button, because most often the user does not look for it, but simply deletes the application with the ends	***(iv) Initial Coin Offerings: Apps facilitating Initial Coin Offerings ("ICOs"), cryptocurrency futures trading, and other crypto-securities or quasi-securities trading must come from established banks, securities firms, futures commission merchants ("FCM"), or other approved financial institutions and must comply with all applicable law.***
	(v) Cryptocurrency apps may not offer currency for completing tasks, such as downloading other apps, encouraging other users to download, posting to social networks, etc.
	3.1.6 Apple Pay: Apps using Apple Pay must provide all material purchase information to the user prior to sale of any good or service and must use Apple Pay branding and user interface elements correctly, as described in the Apple Pay Marketing Guidelines and Human Interface Guidelines. Apps using Apple Pay to offer recurring payments must, at a minimum, disclose the following information:
	The length of the renewal term and the fact that it will continue until canceled
	What will be provided during each period
	The actual charges that will be billed to the customer
	How to cancel
	3.1.7 Advertising: Display advertising should be limited to your main app binary, and should not be included in extensions, App Clips, widgets, notifications, keyboards, watchOS apps, etc. Ads displayed in an app must be appropriate for the app's age rating, allow the user to see all information used to target them for that ad (without requiring the user to leave the app), and may not engage in targeted or behavioral advertising based on sensitive user data such as health/medical data (HealthKit APIs), school and classroom data (ClassKit), or from kids, etc. Interstitial ads or ads that interrupt or block the user experience must clearly indicate that they are an ad, must not manipulate or trick users into tapping into them, and must provide easily accessible and visible close/skip buttons large enough for people to easily dismiss the ad.
	3.2 Other Business Model IssuesThe lists below are not exhaustive, and your submission may trigger a change or update to our policies, but here are some additional dos and don'ts to keep in mind:
	4. Design

	Apple customers place a high value on products that are simple, refined, innovative, and easy to use, and that's what we want to see on the App Store.
	4.1 CopycatsCome up with your own ideas. We know you have them, so make yours come to life. Don't simply copy the latest popular app on the App Store, or make some minor changes to another app's name or UI and pass it off as your own. In addition to risking an intellectual property infringement claim
	4.2 Minimum FunctionalityYour app should include features, content, and UI that elevate it beyond a repackaged website. If your app is not particularly useful, unique, or "app-like," it doesn't belong on the App Store. If your App doesn't provide some sort of lasting entertainment value or adequate utility, it may not be accepted. Apps that are simply a song or movie should be submitted to the iTunes Store. Apps that are simply a book or game guide should be submitted to the Apple Books Store.
	4.2.1 Apps using ARKit should provide rich and integrated augmented reality experiences; merely dropping a model into an AR view or replaying animation is not enough.
	4.2.2 Other than catalogs, apps shouldn't primarily be marketing materials, advertisements, web clippings, content aggregators, or a collection of links.
	4.2.3
	(i) Your app should work on its own without requiring installation of another app to function.
	(ii) Make sure you include sufficient content in the binary for the app to function at launch.
	(iii) If your app needs to download additional resources in order to function on initial launch, disclose the size of the download and prompt users before doing so.
	4.2.4 Apple Watch apps that appear to be a watch face are confusing, because people will expect them to work with device features such as swipes, notifications, and third-party complications. Creative ways of expressing time as an app interface is great (say, a tide clock for surfers), but if your app comes too close to resembling a watch face, we will reject it.
	4.2.5 Apps that are primarily iCloud and iCloud Drive file managers need to include additional app functionality to be approved.
	4.2.6 Apps created from a commercialized template or app generation service will be rejected unless they are submitted directly by the provider of the app's content.
	4.2.7 Remote Desktop Clients: If your remote desktop app acts as a mirror of specific software or services rather than a generic mirror of the host device, it must comply with the following:
	(a) The app must only connect to a user-owned host device that is a personal computer or dedicated game console owned by the user, and both the host device and client must be connected on a local and LAN-based network.
	(b) Any software or services appearing in the client are fully executed on the host device, rendered on the screen of the host device, and may not use APIs or platform features beyond what is required to stream the Remote Desktop.
	(c) All account creation and management must be initiated from the host device.
	(d) The UI appearing on the client does not resemble an iOS or App Store view, does not provide a store-like interface, or include the ability to browse, select, or purchase software not already owned or licensed by the user. For the sake of clarity, transactions taking place within mirrored software do not need to use in-app purchase, provided the transactions are processed on the host device.
	(e) Thin clients for cloud-based apps are not appropriate for the App Store.
	4.3 SpamDon't create multiple Bundle IDs of the same app. If your app has different versions for specific locations, sports teams, universities, etc., consider submitting a single app and provide the variations using in-app purchase. Also avoid piling on to a category that is already saturated; the App Store has enough fart, burp, flashlight, fortune telling, dating, drinking games, and Kama Sutra apps, etc. already.
	4.4 ExtensionsApps hosting or containing extensions must comply with the App Extension Programming Guide or the Safari App Extensions Guide and should include some functionality, such as help screens and settings interfaces where possible. You should clearly and accurately disclose what extensions are made available in the app's marketing text, and the extensions may not include marketing, advertising, or in-app purchases.
	4.4.1 Keyboard extensions have some additional rules.
	4.4.2 Safari extensions must run on the current version of Safari on macOS. They may not interfere with System or Safari UI elements and must never include malicious or misleading content or code. Violating this rule will lead to removal from the Apple Developer Program. Safari extensions should not claim access to more websites than strictly necessary to function.

	4.4.3 StickersStickers are a great way to make Messages more dynamic and fun, letting people express themselves in clever, funny, meaningful ways. Whether your app contains a sticker extension or you're creating free-standing sticker packs, its content shouldn't offend users, create a negative experience, or violate the law.
	4.5 Apple Sites and Services
	4.5.1 Apps may use approved Apple RSS feeds such as the iTunes Store RSS feed, but may not scrape any information from Apple sites (e.g. apple.com, the iTunes Store, App Store, App Store Connect, developer portal, etc.) or create rankings using this information.
	4.5.2 Apple Music
	4.5.3 Do not use Apple Services to spam, phish, or send unsolicited messages to customers, including Game Center, Push Notifications, etc. Do not attempt to reverse lookup, trace, relate, associate, mine, harvest, or otherwise exploit Player IDs, aliases, or other information obtained through Game Center, or you will be removed from the Apple Developer Program.
	4.5.4 Push Notifications must not be required for the app to function, and should not be used to send sensitive personal or confidential information. Push Notifications should not be used for promotions or direct marketing purposes unless customers have explicitly opted in to receive them via consent language displayed in your app's UI, and you provide a method in your app for a user to opt out from receiving such messages. Abuse of these services may result in revocation of your privileges.
	4.5.5 Only use Game Center Player IDs in a manner approved by the Game Center terms and do not display them in the app or to any third party.
	4.5.6 Apps may use Unicode characters that render as Apple emoji in their app and app metadata. Apple emoji may not be used on other platforms or embedded directly in your app binary.
	4.6 Alternate App IconsApps may display customized icons, for example, to reflect a sports team preference, provided that each change is initiated by the user and the app includes settings to revert to the original icon.
	4.7 HTML5 Games, Bots, etc.Apps may contain or run code that is not embedded in the binary (e.g. HTML5-based games, bots, etc.), as long as code distribution isn't the main purpose of the app, the code is not offered in a store or store-like interface, and provided that the software adheres to the additional rules that follow in 4.7.1 and 4.7.2. These additional rules are important to preserve the experience that App Store customers expect, and to help ensure user safety.
	4.7.1 Software offered under this rule must:
	be free or purchased using in-app purchase;
	only use capabilities available in a standard WebKit view (e.g. it must open and run natively in Safari without modifications or additional software); and use WebKit and JavaScript Core to run third-party software and should not attempt to extend or expose native platform APIs to third-party software;
	be offered by developers that have joined the Apple Developer Program and signed the Apple Developer Program License Agreement;
	not provide access to real money gaming, lotteries, or charitable donations;
	adhere to the terms of these App Store Review Guidelines (e.g. do not include objectionable content); and
	not offer digital goods or services for sale.
	4.7.2 Upon request, you must provide an index of software and metadata available in your app. It must include Apple Developer Program Team IDs for the providers of the software along with a URL which App Review can use to confirm that the software complies with the requirements above.
	4.8 Sign in with AppleApps that use a third-party or social login service (such as Facebook Login, Google Sign-In, Sign in with Twitter, Sign In with LinkedIn, Login with Amazon, or WeChat Login) to set up or authenticate the user's primary account with the app must also offer Sign in with Apple as an equivalent option. A user's primary account is the account they establish with your app for the purposes of identifying themselves, signing in, and accessing your features and associated services.
	Sign in with Apple is not required if:
	Your app exclusively uses your company's own account setup and sign-in systems.
	Your app is an education, enterprise, or business app that requires the user to sign in with an existing education or enterprise account.
	Your app uses a government or industry-backed citizen identification system or electronic ID to authenticate users.
	Your app is a client for a specific third-party service and users are required to sign in to their mail, social media, or other third-party account directly to access their content.

	4.9 Streaming gamesStreaming games are permitted so long as they adhere to all guidelines—for example, each game update must be submitted for review, developers must provide appropriate metadata for search, games must use in-app purchase to unlock features or functionality, etc. Of course, there is always the open Internet and web browser apps to reach all users outside of the App Store.
	4.9.1 Each streaming game must be submitted to the App Store as an individual app so that it has an App Store product page, appears in charts and search, has user ratings and review, can be managed with ScreenTime and other parental control apps, appears on the user's device, etc.
	4.9.2 Streaming game services may offer a catalog app on the App Store to help users sign up for the service and find the games on the App Store, provided that the app adheres to all guidelines, including offering users the option to pay for a subscription with in-app purchase and use Sign in with Apple. All the games included in the catalog app must link to an individual App Store product page.
	5. Legal
	5.1 PrivacyProtecting user privacy is paramount in the Apple ecosystem, and you should use care when handling personal data to ensure you've complied with privacy best practices, applicable laws, and the terms of the Apple Developer Program License Agreement, not to mention customer expectations. More particularly:
	5.1.1 Data Collection and Storage
	(i) Privacy Policies: All apps must include a link to their privacy policy in the App Store Connect metadata field and within the app in an easily accessible manner. The privacy policy must clearly and explicitly:
	Identify what data, if any, the app/service collects, how it collects that data, and all uses of that data.
	(ii) Permission: Apps that collect user or usage data must secure user consent for the collection, even if such data is considered to be anonymous at the time of or immediately following collection. Paid functionality must not be dependent on or require a user to grant access to this data.
	(iii) Data Minimization: Apps should only request access to data relevant to the core functionality of the app and should only collect and use data that is required to accomplish the relevant task. Where possible, use the out-of-process picker or a share sheet rather than requesting full access to protected resources like Photos or Contacts.
	(iv) Access: Apps must respect the user's permission settings and not attempt to manipulate, trick, or force people to consent to unnecessary data access. For example, apps that include the ability to post photos to a social network must not also require microphone access before allowing the user to upload photos. Where possible, provide alternative solutions for users who don't grant consent. For example, if a user declines to share Location, offer the ability to manually enter an address.
	(v) Account Sign-In: If your app doesn't include significant account-based features, let people use it without a login. If your app supports account creation, you must also offer account deletion within the app. Apps may not require users to enter personal information to function, except when directly relevant to the core functionality of the app or required by law. If your core app functionality is not related to a specific social network (e.g. Facebook, WeChat, Weibo, Twitter, etc.), you must provide access without a login or via another mechanism. Pulling basic profile information, sharing to the social network, or inviting friends to use the app are not considered core app functionality. The app must also include a mechanism to revoke social network credentials and disable data access between the app and social network from within the app. An app may not store credentials or tokens to social networks off of the device and may only use such credentials or tokens to directly connect to the social network from the app itself while the app is in use.
	(vi) Developers that use their apps to surreptitiously discover passwords or other private data will be removed from the Apple Developer Program.
	(vii) SafariViewController must be used to visibly present information to users; the controller may not be hidden or obscured by other views or layers. Additionally, an app may not use SafariViewController to track users without their knowledge and consent.
	(viii) Apps that compile personal information from any source that is not directly from the user or without the user's explicit consent, even public databases, are not permitted on the App Store.
	(ix) Apps that provide services in highly-regulated fields (such as banking and financial services, healthcare, gambling, legal cannabis use, and air travel) or that require sensitive user information should be submitted by a legal entity that provides the services, and not by an individual developer. Apps that facilitate the legal sale of cannabis must be geo-restricted to the corresponding legal jurisdiction.

	(x) Apps may request basic contact information (such as name and email address) so long as the request is optional for the user, features and services are not conditional on providing the information, and it complies with all other provisions of these guidelines, including limitations on collecting information from kids.
	5.1.2 Data Use and Sharing
	(i) Unless otherwise permitted by law, you may not use, transmit, or share someone's personal data without first obtaining their permission. You must provide access to information about how and where the data will be used. Data collected from apps may only be shared with third parties to improve the app or serve advertising (in compliance with the Apple Developer Program License Agreement). You must receive explicit permission from users via the App Tracking Transparency APIs to track their activity. Learn more about tracking. Apps that share user data without user consent or otherwise complying with data privacy laws may be removed from sale and may result in your removal from the Apple Developer Program.
	(ii) Data collected for one purpose may not be repurposed without further consent unless otherwise explicitly permitted by law.
	(iii) Apps should not attempt to surreptitiously build a user profile based on collected data and may not attempt, facilitate, or encourage others to identify anonymous users or reconstruct user profiles based on data collected from Apple-provided APIs or any data that you say has been collected in an "anonymized," "aggregated," or otherwise non-identifiable way.
	(iv) Do not use information from Contacts, Photos, or other APIs that access user data to build a contact database for your own use or for sale/ distribution to third parties, and don't collect information about which other apps are installed on a user's device for the purposes of analytics or advertising/marketing.
	(v) Do not contact people using information collected via a user's Contacts or Photos, except at the explicit initiative of that user on an individualized basis; do not include a Select All option or default the selection of all contacts. You must provide the user with a clear description of how the message will appear to the recipient before sending it (e.g. What will the message say? Who will appear to be the sender?).
	(vi) Data gathered from the HomeKit API, HealthKit, Clinical Health Records API, MovementDisorder APIs, ClassKit or from depth and/or facial mapping tools (e.g. ARKit, Camera APIs, or Photo APIs) may not be used for marketing, advertising or use-based data mining, including by third parties. Learn more about best practices for implementing CallKit, HealthKit, ClassKit, and ARKit.
	(vii) Apps using Apple Pay may only share user data acquired via Apple Pay with third parties to facilitate or improve delivery of goods and services.
	5.1.3 Health and Health ResearchHealth, fitness, and medical data are especially sensitive and apps in this space have some additional rules to make sure customer privacy is protected:
	(i) Apps may not use or disclose to third parties data gathered in the health, fitness, and medical research context—including from the Clinical Health Records API, HealthKit API, Motion and Fitness, MovementDisorderAPIs, or health-related human subject research—for advertising, marketing, or other use-based data mining purposes other than improving health management, or for the purpose of health research, and then only with permission. Apps may, however, use a user's health or fitness data to provide a benefit directly to that user (such as a reduced insurance premium), provided that the app is submitted by the entity providing the benefit, and the data is not be shared with a third party. You must disclose the specific health data that you are collecting from the device.
	(ii) Apps must not write false or inaccurate data into HealthKit or any other medical research or health management apps, and may not store personal health information in iCloud.
	(iii) Apps conducting health-related human subject research must obtain consent from participants or, in the case of minors, their parent or guardian. Such consent must include the (a) nature, purpose, and duration of the research; (b) procedures, risks, and benefits to the participant; (c) information about confidentiality and handling of data (including any sharing with third parties); (d) a point of contact for participant questions; and (e) the withdrawal process.
	(iv) Apps conducting health-related human subject research must secure approval from an independent ethics review board. Proof of such approval must be provided upon request.
	5.1.4 KidsFor many reasons, it is critical to use care when dealing with personal data from kids, and we encourage you to carefully review all the requirements for complying with laws like the Children's Online Privacy Protection Act ("COPPA"), the European Union's General Data Protection Regulation ("GDPR"), and any other applicable regulations or laws.

	Apps may ask for birthdate and parental contact information only for the purpose of complying with these statutes, but must include some useful functionality or entertainment value regardless of a person's age.
	Apps intended primarily for kids should not include third-party analytics or third-party advertising. This provides a safer experience for kids. In limited cases, third-party analytics and third-party advertising may be permitted provided that the services adhere to the same terms set forth in Guideline 1.3.
	Moreover, apps in the Kids Category or those that collect, transmit, or have the capability to share personal information (e.g. name, address, email, location, photos, videos, drawings, the ability to chat, other personal data, or persistent identifiers used in combination with any of the above) from a minor must include a privacy policy and must comply with all applicable children's privacy statutes. For the sake of clarity, the parental gate requirement for the Kid's Category is generally not the same as securing parental consent to collect personal data under these privacy statutes.
	As a reminder, Guideline 2.3.8 requires that use of terms like "For Kids" and "For Children" in app metadata is reserved for the Kids Category. Apps not in the Kids Category cannot include any terms in app name, subtitle, icon, screenshots or description that imply the main audience for the app is children.
	5.1.5 Location ServicesUse Location services in your app only when it is directly relevant to the features and services provided by the app. Location-based APIs shouldn't be used to provide emergency services or autonomous control over vehicles, aircraft, and other devices, except for small devices such as lightweight drones and toys, or remote control car alarm systems, etc. Ensure that you notify and obtain consent before collecting, transmitting, or using location data. If your app uses location services, be sure to explain the purpose in your app; refer to the Human Interface Guidelines for best practices for doing so.
	5.2 Intellectual PropertyMake sure your app only includes content that you created or that you have a license to use. Your app may be removed if you've stepped over the line and used content without permission. Of course, this also means someone else's app may be removed if they've "borrowed" from your work. If you believe your intellectual property has been infringed by another developer on the App Store, submit a claim via our web form. Laws differ in different countries, but at the very least, make sure to avoid the following common errors:
	5.2.1 Generally: Don't use protected third-party material such as trademarks, copyrighted works, or patented ideas in your app without permission, and don't include misleading, false, or copycat representations, names, or metadata in your app bundle or developer name. Apps should be submitted by the person or legal entity that owns or has licensed the intellectual property and other relevant rights.
	5.2.2 Third-Party Sites/Services: If your app uses, accesses, monetizes access to, or displays content from a third-party service, ensure that you are specifically permitted to do so under the service's terms of use. Authorization must be provided upon request.
	5.2.3 Audio/Video Downloading: Apps should not facilitate illegal file sharing or include the ability to save, convert, or download media from third-party sources (e.g. Apple Music, YouTube, SoundCloud, Vimeo, etc.) without explicit authorization from those sources.
	5.2.4 Apple Endorsements: Don't suggest or imply that Apple is a source or supplier of the App, or that Apple endorses any particular representation regarding quality or functionality.
	5.2.5 Apple Products: Don't create an app that appears confusingly similar to an existing Apple product, interface (e.g. Finder), app (such as the App Store, iTunes Store, or Messages) or advertising theme. The Human Interface Guidelines have more information on how to use Activity rings.
	5.3 Gaming, Gambling, and LotteriesGaming, gambling, and lotteries can be tricky to manage and tend to be one of the most regulated offerings on the App Store. Only include this functionality if you've fully vetted your legal obligations everywhere you make your app available and are prepared for extra time during the review process. Some things to keep in mind:
	5.4 VPN AppsApps offering VPN services must utilize the NEVPNManager API and may only be offered by developers enrolled as an organization. You must make a clear declaration of what user data will be collected and how it will be used on an app screen prior to any user action to purchase or otherwise use the service.
	5.5 Mobile Device Management Mobile Device Management Apps that offer Mobile Device Management (MDM) services must request this capability from Apple. Such apps may only be offered by commercial enterprises, educational institutions, or government agencies, and in limited cases, companies using MDM for parental control services or device security.

	5.6 Developer Code of ConductPlease treat everyone with respect, whether in your responses to App Store reviews, customer support requests, or when communicating with Apple, including your responses in Resolution Center. Do not engage in harassment of any kind, discriminatory practices, intimidation, bullying, and don't encourage others to engage in any of the above. Repeated manipulative or misleading behavior or other fraudulent conduct will lead to your removal from the Apple Developer Program.
	5. Check list design Android- https://developer.android.com/docs/quality-guidelines/core-app-quality
	Area
	Navigation
	Notifications
	UI and Graphics
	Visual quality

	Accessibility
	Your app should implement the expected functional behavior.
	Audio
	Media
	Sharing
	Background Service
	Your app should provide the performance, stability, compatibility, and responsiveness expected by users.
	Stability
	Performance
	SDK

	Battery
	Your app should handle user data and personal information safely, with the appropriate level of permission.
	In addition to this checklist, applications published on the Google Play Store must also follow the User Data policies to protect users' privacy.
	Permissions
	Data & Files
	Identity
	App Components

	Networking
	WebViews
	Execution
	Cryptography
	Be sure that your apps can be published on Google Play.
	Policies
	App Details Page
	User Support

Functional Testing Mobile		
1. Installation / removal / rolling of versions		
2. Launching the application (displaying Splash Screen)		
3. The performance of the main functionality of the application		
3.1 Authorization (by phone number / via social networks / e-mail)		
3.2 Registration (by phone number / via social networks / e-mail)		
3.3 Onboarding new users		
3.4 Validation of required fields		
3.5 Navigation between application sections		
3.6 Editing data in the user profile		
3.7 Checking payment		
3.8 Testing Filters		
no space for installation		
supporting functions check - 3G, SD card		
install or transfer app on sd card		
compliance of the rules - Apple hig, google material design		
3.9 Bonuses		
4. Correct display of errors		
5. Working with files (sending / receiving / viewing)		
6. Testing timeouts		
7. Testing stubs (no internet connection / no, for example, goods, etc.)		
8. Testing pop-ups, alerts		
9. Testing WebView		
10. Scroll / swipe elements		
11. Testing PUSH notifications https://software-testing.ru/library/testing/mobile-testing/3595-push-notifications-in-mobile-apps		
12. Minimizing / expanding the application		
13. Different types of connections (cellular / Wi-Fi)		
14. Screen orientation (landscape / portrait)		
15. Dark / light themes		
16. Advertising in the app		
17. Sharing content in the social. the network		
18. Application work in the background		
19. Page pagination пагинация		
20. Privacy policies and other links to documents		
21. Cross platform		
Interruption Testing:		
This form of mobile testing checks how an application responds when faced with an unexpected interruption. Depending on the nature of the interruption, the application should pause and then return to its original state, or even react in a particular way. Obviously, the kind of interruptions will differ on the basis of the application under test, but some common interruptions that should be considered while testing are:		
Incoming or phone call when an application is running		
Incoming message or SMS when an application is running		
Low battery when an application is running		
The device plugged in or out of charging when an application is running		
Device shutting down when an application is running		

OS upgrade occurring when an application is running		
Loss and restoration of the network while an application is running. Interruption testing ensures that an app handles interruptions without failure or anomaly. When being used by real users, every app will have to operate along with other device functions. This means that every app will have to be optimized to deal with these device functions while running at all times.		
Localization Testing:		
This approach to mobile app testing is meant to test features that are dependent on the geographical location of an app. Since most apps seek to appeal to a global user base, they include localized features for the convenience of users. These features can vary from enabling different languages, enabling commerce in local currency to adherence with local laws and regulations. Localization testing checks these features to ensure that they are activated and functioning in the right locations. Customers always prefer apps with UI elements aligned with their culture, language, and device accessibility. They expect their experience to be adjusted to their localized needs and preferences. AppAnnie's research confirms that fully localized apps do better in the global market. Localization testing is also one of the most challenging mobile app testing types since most QA teams lack adequate access to test coverage and resources necessary for its implementation.		
1. All elements in the application are translated into the needed language		
2. The texts are protected inside the application and the user in the application settings can set the required language		
3. The texts depend on the language in the system settings		
4. Texts come from the server		
5. Correct display of date formats (YEAR - MONTH - DAY or DAY - MONTH - YEAR.)		
6. Correct display of time depending on the time zone		
in another language, there must be enough room for text on the screen		
dates must match the format of the specified region		
temporary settings must be respected		
Memory Leak Testing:		

A memory leak refers to a situation in which the app fails to return the memory it has acquired for temporary use in order to function. The available memory for the app drains, and the app cannot function. If an app is frequently used or opened, a small memory leak can result in its termination. Memory leaks emerge from programming bugs, so every app needs to be tested for this issue. Memory leak testing is done by running an app on multiple devices. By doing so, testers can check the app performance on devices with different memory capabilities, and optimize the app to function effectively on each configuration.		
can be checked with Instruments (standard MacOS application). Can be no more than 30MB for a 2g iPhone / iPod, about 70MB for all devices up to the 2nd iPad		
pay attention to windows with a lot of information, during a long stay of the user in the application		
Usability Testing:		
Also known as user experience testing, this checks an app for user-friendliness. Basically, it checks ease of use and intuitiveness, aiming to provide a seamless user experience that is free of bugs and anomalies. Since the success of an app depends on the appeal of its end-to-end user experience, it is best to do usability tests with actual customers on real devices. This is the best way to understand the preferences of the target audience. Conversely, one can have skilled testers running user scenarios that mirror the behavior of actual end-users. A few pointers to keep in mind during usability tests:		
Smooth, visually appealing layout and design		
A high degree of intuitiveness		
Quick response time – Most users prefer apps that launch within 2-3 seconds after tapping the icon.		
Correct display of elements on devices with different screen resolutions		
All fonts meet the requirements		
All texts are correctly aligned		
All error messages are correct, without spelling and grammatical errors		
Correct screen titles		
There are placeholders in the search lines		
Inactive elements are displayed in gray		
Links to documents lead to the corresponding section on the site		
Animation between transitions		
Correct return to the previous screen		
Supports basic gestures when working with touch screens (swipe back, etc.)		
Pixel-perfect		
Checking the operation of applications on retina screens and various OS versions		
correct display of various elements on retina / non-retina screens		

installing the application on the correct OS version	🟨	🟨
check the installation for all possible devices		
various functions on devices: absence / presence of a camera (ipad) (autofocus), absence / presence of GPS		
Performance Testing:		
It is important to test how an application performs under various conditions. This is where performance testing comes in. It puts the device under various forms of pressure so that it does not malfunction in non-optimal conditions. A few things that performance testing should verify:		
Device performance:		
Installation and log-in time, battery consumption, memory consumption, etc.		
Network performance:		
Delays, errors, pauses in receiving digital information or rendering network-activated features		
API/Server performance:		
Speed and formation of data transfer from back-end to front-end		
Recovery capabilities:		
Built-in back-up and recovery functions that can save or recover user data in the event of data loss.		
Security Testing:		

App users are becoming increasingly conscious of issues surrounding data security. Online privacy and confidentiality of personal data are major concerns for most netizens – 70% report being concerned that their personal information will be shared without permission. In fact, 81% of users say they would uninstall an app and switch vendors because of security concerns. Needless to say, security testing is imperative to the success of an app. Since almost every app requires some kind of personal information to run, tests must be conducted to fortify them, in order to provide confidentiality of data. QAs must thoroughly check that the application is able to defend its users from having their information leaked or hacked in. This is especially true of financial apps		
1. Testing permissions (access to camera / microphone / gallery / etc.)		
2. User data (passwords) are not transmitted in clear text		
3. In the fields with password entry and password confirmation, the data is hidden by asterisks		
Compatibility		
Compatibility testing is used to ensure that your application is compatible with other OS versions, various shells and third-party services, and device hardware.		
1. Correct display of geo		
2. Information about transactions (checks, etc.)		
3. Various payment methods (Google Pay, Apple Pay)		
4. Testing sensors (illumination, device temperature, gyroscope, etc.)		
5. Testing interruptions (incoming call / SMS / push / alarm clock / Do not disturb mode, etc.)		
6. Connecting external devices (memory card / headphones, etc.)		
Checking the type of purchases (recoverable, not recoverable)		

verification of compliance with the actual / declared value of the application		
verification of the restoration of the purchase regardless of the device, but with the link to the account		
restoration of purchases		
Save of the purchases with app update		
Checking energy consumption		
you need to check how much your powerful application drains the battery of the device. Most likely, the user will delete it if it causes the mobile phone to be charged too often.		
These are the main points and features of mobile application testing that you should pay attention to when testing mobile applications. Each device is individual due to the parameters and configurations set by the user. But nevertheless, you should adhere to checking the above points on any device.		
Checking the work of the feedback		
content upload messages / progress		
network access error messages		
presence of messages when trying to delete important information		
the presence of a screen / message at the end of the process / game (Game over screen)		
Checking for updates		

checking various ways to install updates (wifi, bluetooth, usb)		
checking the work of the installed changes, the places where they were made		
make sure that updates are supported by older operating systems so that elements that work well on the new system do not crash on older versions.		
Checking the Application Response to External Interrupts		
incoming / outgoing sms, mms, calls		
battery discharge / removal		
network disconnect / wifi		
connecting cable, card, charging		
Advertising in the mobile application		
ads should not overlap application control buttons		
ads should have an available close button, because most often the user does not look for it, but simply deletes the application with the ends		

	ID	Tests
	VX-N1	CR-3
	VX-N2	CR-3
	VX-N3	CR-1 CR-3 CR-5
	VX-S1	CR-9
	VX-S2	CR-9
	VX-U1	CR-5
	VX-U2	CR-5
	VX-U3	CR-5
	VX-V1	CR-all
	VX-V2	CR-all
	VX-V3	CR-all

	VX-A1	CR-all
	VX-A2	CR-all
	VX-A3	CR-all
	FN-A1	CR-1 CR-8
	FN-A2	CR-1 CR-2 CR-8
	FN-A3	CR-0
	FN-A4	CR-0
	FN-A5	CR-0
	FN-M1	CR-0 CR-6 CR-8
	FN-M2	CR-0
	FN-M3	CR-0
	FN-S1	CR-0
	FN-B1	CR-6
	PS-S1	CR-all SD-1
	PS-P1	CR-all SD-1
	PS-P2	CR-all SD-1
	PS-P3	PM-1
	PS-T1	CR-0
	PS-T2	SP-1
	PS-T3	SP-1

	PS-T4	SP-2 SP-3
	PS-T5	SP-3
	PS-T6	SP-2
	PS-B1	BA-1
	SC-P1	SC-4
	SC-P2	
	SC-P3	CR-0
	SC-P4	CR-0
	SC-P5	CR-0
	SC-DF1	SC-1
	SC-DF2	SC-10
	SC-DF3	
	SC-ID1	CR-0
	SC-ID2	CR-0
	SC-ID3	CR-0
	SC-AC1	SC-5
	SC-AC2	CR-0 SC-4

	SC-AC3	SC-3
	SC-N1	SC-9
	SC-N2	SC-6
	SC-N3	
	SC-W1	SC-6
	SC-W2	SC-7
	SC-E1	
	SC-C1	
	GP-P1	GP-all
	GP-P2	GP-1
	GP-D1	GP-1 GP-2
	GP-D2	GP-1
	GP-D3	GP-1
	GP-X1	GP-1

Functional Testing Mobile	
1. Installation / removal / rolling of versions	
2. Launching the application (displaying Splash Screen)	
3. The performance of the main functionality of the application	
3.1 Authorization (by phone number / via social networks / e-mail)	
3.2 Registration (by phone number / via social networks / e-mail)	
3.3 Onboarding new users	
3.4 Validation of required fields	
3.5 Navigation between application sections	
3.6 Editing data in the user profile	
3.7 Checking payment	
3.8 Testing Filters	
no space for installation	
supporting functions check - 3G, SD card	
install or transfer app on sd card	
compliance of the rules - Apple hig, google material design	
3.9 Bonuses	
4. Correct display of errors	
5. Working with files (sending / receiving / viewing)	
6. Testing timeouts	
7. Testing stubs (no internet connection / no, for example, goods, etc.)	
8. Testing pop-ups, alerts	
9. Testing WebView	
10. Scroll / swipe elements	
11. Testing PUSH notifications https://software-testing.ru/library/testing/mobile-testing/3595-push-notifications-in-mobile-apps	
12. Minimizing / expanding the application	
13. Different types of connections (cellular / Wi-Fi)	
14. Screen orientation (landscape / portrait)	
15. Dark / light themes	
16. Advertising in the app	
17. Sharing content in the social. the network	
18. Application work in the background	
19. Page pagination пагинация	
20. Privacy policies and other links to documents	
21. Cross platform	
Interruption Testing:	
This form of mobile testing checks how an application responds when faced with an unexpected interruption. Depending on the nature of the interruption, the application should pause and then return to its original state, or even react in a particular way. Obviously, the kind of interruptions will differ on the basis of the application under test, but some common interruptions that should be considered while testing are:	
Incoming or phone call when an application is running	
Incoming message or SMS when an application is running	
Low battery when an application is running	
The device plugged in or out of charging when an application is running	
Device shutting down when an application is running	

OS upgrade occurring when an application is running	
Loss and restoration of the network while an application is running. Interruption testing ensures that an app handles interruptions without failure or anomaly. When being used by real users, every app will have to operate along with other device functions. This means that every app will have to be optimized to deal with these device functions while running at all times.	
Localization Testing:	
This approach to mobile app testing is meant to test features that are dependent on the geographical location of an app. Since most apps seek to appeal to a global user base, they include localized features for the convenience of users. These features can vary from enabling different languages, enabling commerce in local currency to adherence with local laws and regulations. Localization testing checks these features to ensure that they are activated and functioning in the right locations. Customers always prefer apps with UI elements aligned with their culture, language, and device accessibility. They expect their experience to be adjusted to their localized needs and preferences. AppAnnie's research confirms that fully localized apps do better in the global market. Localization testing is also one of the most challenging mobile app testing types since most QA teams lack adequate access to test coverage and resources necessary for its implementation.	
1. All elements in the application are translated into the needed language	
2. The texts are protected inside the application and the user in the application settings can set the required language	
3. The texts depend on the language in the system settings	
4. Texts come from the server	
5. Correct display of date formats (YEAR - MONTH - DAY or DAY - MONTH - YEAR.)	
6. Correct display of time depending on the time zone	
in another language, there must be enough room for text on the screen	
dates must match the format of the specified region	
temporary settings must be respected	
Memory Leak Testing:	

A memory leak refers to a situation in which the app fails to return the memory it has acquired for temporary use in order to function. The available memory for the app drains, and the app cannot function. If an app is frequently used or opened, a small memory leak can result in its termination. Memory leaks emerge from programming bugs, so every app needs to be tested for this issue. Memory leak testing is done by running an app on multiple devices. By doing so, testers can check the app performance on devices with different memory capabilities, and optimize the app to function effectively on each configuration.	
can be checked with Instruments (standard MacOS application). Can be no more than 30MB for a 2g iPhone / iPod, about 70MB for all devices up to the 2nd iPad	
pay attention to windows with a lot of information, during a long stay of the user in the application	
Usability Testing:	
Also known as user experience testing, this checks an app for user-friendliness. Basically, it checks ease of use and intuitiveness, aiming to provide a seamless user experience that is free of bugs and anomalies. Since the success of an app depends on the appeal of its end-to-end user experience, it is best to do usability tests with actual customers on real devices. This is the best way to understand the preferences of the target audience. Conversely, one can have skilled testers running user scenarios that mirror the behavior of actual end-users. A few pointers to keep in mind during usability tests:	
Smooth, visually appealing layout and design	
A high degree of intuitiveness	
Quick response time – Most users prefer apps that launch within 2-3 seconds after tapping the icon.	
Correct display of elements on devices with different screen resolutions	
All fonts meet the requirements	
All texts are correctly aligned	
All error messages are correct, without spelling and grammatical errors	
Correct screen titles	
There are placeholders in the search lines	
Inactive elements are displayed in gray	
Links to documents lead to the corresponding section on the site	
Animation between transitions	
Correct return to the previous screen	
Supports basic gestures when working with touch screens (swipe back, etc.)	
Pixel-perfect	
Checking the operation of applications on retina screens and various OS versions	
correct display of various elements on retina / non-retina screens	

installing the application on the correct OS version	🟧
check the installation for all possible devices	
various functions on devices: absence / presence of a camera (ipad) (autofocus), absence / presence of GPS	
Performance Testing:	
It is important to test how an application performs under various conditions. This is where performance testing comes in. It puts the device under various forms of pressure so that it does not malfunction in non-optimal conditions. A few things that performance testing should verify:	
Device performance:	
Installation and log-in time, battery consumption, memory consumption, etc.	
Network performance:	
Delays, errors, pauses in receiving digital information or rendering network-activated features	
API/Server performance:	
Speed and formation of data transfer from back-end to front-end	
Recovery capabilities:	
Built-in back-up and recovery functions that can save or recover user data in the event of data loss.	
Security Testing:	

App users are becoming increasingly conscious of issues surrounding data security. Online privacy and confidentiality of personal data are major concerns for most netizens – 70% report being concerned that their personal information will be shared without permission. In fact, 81% of users say they would uninstall an app and switch vendors because of security concerns. Needless to say, security testing is imperative to the success of an app. Since almost every app requires some kind of personal information to run, tests must be conducted to fortify them, in order to provide confidentiality of data. QAs must thoroughly check that the application is able to defend its users from having their information leaked or hacked in. This is especially true of financial apps	
1. Testing permissions (access to camera / microphone / gallery / etc.)	
2. User data (passwords) are not transmitted in clear text	
3. In the fields with password entry and password confirmation, the data is hidden by asterisks	
Compatibility	
Compatibility testing is used to ensure that your application is compatible with other OS versions, various shells and third-party services, and device hardware.	
1. Correct display of geo	
2. Information about transactions (checks, etc.)	
3. Various payment methods (Google Pay, Apple Pay)	
4. Testing sensors (illumination, device temperature, gyroscope, etc.)	
5. Testing interruptions (incoming call / SMS / push / alarm clock / Do not disturb mode, etc.)	
6. Connecting external devices (memory card / headphones, etc.)	
Checking the type of purchases (recoverable, not recoverable)	

verification of compliance with the actual / declared value of the application	
verification of the restoration of the purchase regardless of the device, but with the link to the account	
restoration of purchases	
Save of the purchases with app update	
Checking energy consumption	
you need to check how much your powerful application drains the battery of the device. Most likely, the user will delete it if it causes the mobile phone to be charged too often.	
These are the main points and features of mobile application testing that you should pay attention to when testing mobile applications. Each device is individual due to the parameters and configurations set by the user. But nevertheless, you should adhere to checking the above points on any device.	
Checking the work of the feedback	
content upload messages / progress	
network access error messages	
presence of messages when trying to delete important information	
the presence of a screen / message at the end of the process / game (Game over screen)	
Checking for updates	

checking various ways to install updates (wifi, bluetooth, usb)	
checking the work of the installed changes, the places where they were made	
make sure that updates are supported by older operating systems so that elements that work well on the new system do not crash on older versions.	
Checking the Application Response to External Interrupts	
incoming / outgoing sms, mms, calls	
battery discharge / removal	
network disconnect / wifi	
connecting cable, card, charging	
Advertising in the mobile application	
ads should not overlap application control buttons	
ads should have an available close button, because most often the user does not look for it, but simply deletes the application with the ends	

	Description
	The app supports standard Back button navigation and does not make use of any custom, on-screen "Back button" prompts.
	The app supports gesture navigation for going back / going to the home screen.
	The app correctly preserves and restores user or app state. The app preserves user or app state when leaving the foreground and prevents accidental data loss due to back-navigation and other state changes. When returning to the foreground, the app should restore the preserved state and any significant stateful transaction that was pending. Examples include: changes to editable fields, game progress, menus, videos, and other sections of the app or game. When the app is resumed from the Recents app switcher, the app returns the user to the exact state in which it was last used. When the app is resumed after the device wakes from the sleep (locked) state, the app returns the user to the exact state in which it was last used. When the app is relaunched from Home or All Apps, it should do one of the following, depending on how much time has passed since it was last used:If the app was last used a short time ago (minutes), restore the app state as close as possible to its previous state. If more time has passed since the app was last used, try to restore the app as close as possible to its previous state; or start it from its home screen or some other default state.
	Notifications follow Material Design guidelines. In particular: Notifications are not used for cross-promotion or advertising another product, as this is strictly prohibited by the Play Store. Notification channels are defined according to best practices, rather than serving all notifications from one channel. Selecting the correct notification priority. Multiple notifications are stacked into a single notification group, where possible. Set timeouts for notifications where appropriate. Notifications are persistent only if related to ongoing events, such as music playback or a phone call. For more information, see the Functionality section.
	For messaging apps, social apps and conversations: Use the MessagingStyle notifications for conversations. Support the direct reply action. Support conversation shortcuts, and implement best practices for getting the best direct share ranking. Support bubbles.
	The app supports both landscape and portrait orientations (if possible) and folding / unfolding.Orientations expose largely the same features and actions and preserve functional parity. Minor changes in content or views are acceptable.
	The app uses the whole screen in both orientations and does not letterbox to account for orientation changes, including folding and unfolding.Minor letterboxing to compensate for small variations in screen geometry is acceptable.
	The app correctly handles rapid transitions between display orientations and device folding / unfolding without rendering problems or losing state.
	The app displays graphics, text, images, and other UI elements without noticeable distortion, blurring, or pixelation. The app should use vector drawables where possible. The app provides high-quality graphics for all targeted screen sizes and form factors. No aliasing at the edges of menus, buttons, and other UI elements is visible.
	The app displays text and text blocks in an acceptable manner for each of the app’s supported languages. Composition is acceptable in all supported form factors. No cut-off letters or words are visible. No improper word wraps within buttons or icons are visible. There is sufficient spacing between text and surrounding elements.
	The app’s content, and all web contents referred to by the app, support dark theme.

	Touch targets should be at least 48dp in size. Learn more.
	The app's text and foreground content should maintain a high enough color contrast ratio with its background: 3.0:1 for large text / graphics 4.5:1 for small text (text smaller than 18pt, or if the text is bold and smaller than 14pt) Learn more about color and contrast.
	Describe each UI element, except for TextView, using contentDescription.
	Audio resumes when the app returns to the foreground, or indicates to the user that playback is in a paused state.
	If audio playback is a core feature, the app should support background playback.
	When the user initiates audio playback, the app should do one of the following within one second: Start playing the audio. Provide a visual indicator that the audio data is being prepared.
	The app should request audio focus when audio starts playing and abandon audio focus when playback stops.
	The app should handle other apps' requests for audio focus. For example, an app might reduce playback volume when another app plays speech.
	If the app plays audio in the background, it must create a Notification styled with MediaStyle.
	If the app plays video, it should support picture-in-picture playback.
	If the app encodes video, it should do so using the HEVC video compression standard.
	The app should use the Android Sharesheet when sharing content. It can suggest targets that are unavailable to custom solutions.
	The app avoids running unnecessarily long services in the background. To ensure the smooth running of the user's device, the system applies various restrictions on background services. These are not considered good uses of background services:Maintaining a network connection for notifications Maintaining a Bluetooth connection Keeping the GPS powered-on Learn how to choose the right solution for your work.
	The app does not crash or block the UI thread causing ANR (Android Not Responding") errors. Utilize Google Play's pre-launch report to identify potential stability issues. After deployment, pay attention to the Android Vitals page in the Google Play developer console.
	The app loads quickly or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load.
	Apps should render frames every 16ms to achieve 60 frames per second. Developers can use the Profile HWUI rendering option in testing. If there are issues, tools are available to help diagnose slow rendering.
	With StrictMode enabled (see StrictMode Testing, below), no red flashes (performance warnings from StrictMode) are visible when testing the app. Any red flashes indicate bad behaviors regarding storage, network access, or memory leaks.
	The app runs on the latest public version of the Android platform without crashing or severely impacting core functionality.
	The app targets the latest Android SDK needed to align with Google Play requirements by setting the targetSdk value.
	The app is built with the latest Android SDK by setting the compileSdk value.

	Any Google or third-party SDKs used are up-to-date. Any improvements to these SDKs, such as stability, compatibility, or security, should be available to users in a timely manner.For Google SDKs, consider using SDKs powered by Google Play services, when available. These SDKs are backward compatible, receive automatic updates, reduce your app package size, and make efficient use of on-device resources. The developer is accountable for the entire app's codebase, inclusive of any third-party SDKs used.
	The app does not use non-SDK interfaces.
	No debug libraries are included in the production app. This can cause performance as well as security issues.
	The app properly supports the power management features that were introduced in Android 6.0 (Doze and App Standby). In the case where core functionality is disrupted by power management, only qualified apps may request an exemption. See Support for other use cases in Doze and App Standby.During development, developers can test app standby and doze behavior using these ADB commands. In terms of battery usage, developers can use the Android Studio energy profiler or the Battery Historian tool, combined with planned background work, to diagnose unexpected battery use.
	The app requests only the absolute minimum number of permissions that it needs to support its use case at hand. For some permissions such as location, use coarse location in place of fine location if possible.
	The app requests permission to access sensitive data (such as SMS, Call Log, or Location) or services that cost money (such as Dialer or SMS) only when directly related to the core use cases of the apps. Implications related to these permissions should be prominently disclosed to the user. Depending on how you are using the permissions, there might be an alternative way to fulfill your app's use case without relying on access to sensitive information. For example, instead of requesting permissions related to a user's contacts, it may be more appropriate to request access by using an implicit intent.
	The app requests runtime permissions in context, when the functionality is requested, rather than upfront during app startup.
	The app clearly conveys why certain permissions are needed or follow the recommended flow to explain why it needs a permission.
	The app should gracefully degrade when users deny or revoke a permission. The app should not prevent the user from accessing the app altogether.
	All sensitive data is stored in the app's internal storage.
	No personal or sensitive user data is logged to the system log or an app-specific log.
	The app does not use any non-resettable hardware IDs, such as the IMEI, for identification purposes.
	The app provides hints to autofill account credentials and other sensitive information, such as credit card info, physical address, and phone number.
	Integrate One Tap for Android for a seamless sign in experience.
	The app supports biometric authentication to protect financial transactions or sensitive information, such as important user documents.
	The app sets the android:exported attribute explicitly for all activities, services, broadcast receivers, and especially content providers. Only application components that share data with other apps, or components that should be invoked by other apps, are exported.
	All intents and broadcasts follow best practices: Use explicit intents if the destination application is well defined. Use Intents to defer permissions to a different app that already has the permission. Share data securely across apps. Intents that contain a payload are verified before use. If you need to pass an Intent to another app, so that the receiving app can invoke and expect a callback in the calling app, do not include a nested intent in the extras. Use a PendingIntent. When setting up your PendingIntents, explicitly set the immutable flag, where applicable.

	All components that share content between your apps use android:protectionLevel="signature" for custom permissions. This includes activities, services, broadcast receivers, and especially content providers.Apps should not rely on accessing a list of installed packages. The access has been restricted beginning in Android 11.
	All network traffic is sent over SSL.
	The application declares a network security configuration.
	If the application uses Google Play services, the security provider is initialized at application startup.
	Do not use setAllowUniversalAccessFromFileURLs() for accessing local content. Instead, use WebViewAssetLoader.
	WebViews should not use addJavaScriptInterface() with untrusted content.On Android 6.0 and above, use HTML message channels instead.
	The app does not dynamically load code from outside the app's APK. Developers should use Android App Bundles, which includes Play Feature Delivery and Play Asset Delivery.Starting August 2021, the use of Android App Bundles will become mandatory for all new apps in the Google Play store.
	The app uses strong, platform-provided cryptographic algorithms and a random number generator. Also, the app does not implement custom algorithms.
	The app strictly adheres to the terms of the Google Play Developer Content Policy and does not offer inappropriate content, does not use the intellectual property or brand of others, and so on.
	The app maturity level is set appropriately, based on the Content Rating Guidelines.
	The app's feature graphic follows the guidelines outlined in this support article. Make sure that: The app listing includes a high-quality feature graphic. The feature graphic does not contain device images, screenshots, or small text that will be illegible when scaled down and displayed on the smallest screen size that your app is targeting. The feature graphic does not resemble an advertisement.
	The app's screenshots and videos do not show or reference non-Android devices.
	The app's screenshots or videos do not represent the content and experience of your app in a misleading way.
	Common user-reported bugs in the Reviews tab of the Google Play page are addressed if they are reproducible and occur on many different devices. If a bug occurs on only a few devices, you should still address it if those devices are particularly popular or new.

Table 1

	GUI
1. Testing the text field.	
Field name (spelling, match with open module or page)	The content of the web page is correct No spelling and grammatical errors, all pages have correct titles.
Alignment of field names (alignment to the left or right (depending on the requirements of the application, indents, identical distances between the title and the field)	Alignment of pictures, fonts, texts. All fonts are compliant.
Correct position of the text inside the text, long text does not go beyond the boundaries of the field when typing	Informative errors, tips.
Unification of design (color, font, size (height / width), alignment of margins)	Tooltips exist for all fields.
The location of the entered text within the field (unification, bottom alignment, unless otherwise determined by the specific requirements of the application)	Indentation between fields, columns, rows and error messages.
	The buttons have a standard size and color.
2. Testing the button.	There are no broken links and images on the site.
Button name (spelling, matching with action)	Inactive fields are grayed out.
Clicking' effect (the appearance of the button should change when clicked, if this does not contradict the capabilities of the browser)	Check the site at different screen resolutions.
Hints name (match with the button name, spelling)	The scroll should only appear when required.

· Unification of design (color, font, size (height / width), highlight color, alignment). The check is carried out both for the button, as for the element, and for the button name	Displaying checkboxes and radio buttons, buttons must be accessible from the keyboard, and the user must be able to navigate the site using only the keyboard.
·	Displaying drop-down lists.
3. Testing radio buttons.	Long text is hidden under ellipsis.
Unified design for the entire application	Correct date selection.
Aligning the location of the radio button with the corresponding name	Placeholders in fields.
Alignment of radio button locations (along the edge)	The logo leads to the main page of the site.
	Transitions and navigation between pages and menu sections.
4. Testing check boxes.	
Unified design for the entire application	
Aligning the location of the checkbox with the appropriate name	
Correct display of a disabled checkbox	
5. Testing combo boxes.	
Spelling values	
Highlighting when selecting each of the values, when selecting several values at the same time	
· Unification of design (color, font, size (height / width), highlight color, alignment). Checking is carried out both for the field, as an element, and for the values, and their names	
6. Testing the menu.	
Tab highlighting on hover	
Change cursor on hover	
Clicking' effect, if this does not conflict with the capabilities of the browser	
If the work is currently in the selected tab, then in the menu it differs visually (highlighted, underlined)	
Matching names in case the menu is duplicated in several places	
7. Testing windows.	
The appearance of a scroll when decreasing (changing) the size of the browser window	

Preservation of the arrangement of elements when reducing (changing) the browser window, when changing the scale	
Matching the window title depending on the purpose of the page (for example, the window title should be Profile if the user is on a profile page)	
Spelling, naming syntax	
· Unification of names	
8. Testing scrolling.	
Unification of types and types of scrolls on all pages (if there is a custom scroll, it must be applied on all identical forms)	
9. Testing links.	
Unification of styles (in accordance with the site design)	
· Location of links (in accordance with the design of the site). For example, positioning all links to the left or right of the elements	
Names (unification, identity of the names of links of the same purpose, spelling, correspondence with the open module or page, capacity of the name of the link in the allotted block)	
Changing the appearance of the cursor when hovering over a link	
Change the appearance of the link on hover (underline)	
10. Testing tables.	
Unification of design for the whole application (color, font, size (height / width), alignment)	
Title (correspondence with the current module, spelling)	
Alignment of sorting icons in column names	
Alignment of column names, values within the table	
Correct display of long names (corresponding line breaks, abbreviated names (appearance ..., or abbreviated by word)	
Correct display of data after using sorting (column and column sizes are fixed, the text does not break the table structure)	
11. Testing pop-ups.	
Spelling, the syntax of the text located on the pop-up	
Pop-up display in the center of the page, window, form	

Alignment of the text presented on the pop-up	
Correct position of the text on the pop-up: the text should be within the pop-up, the long name should be on a new line, unless otherwise specified by the specific requirements of the application)	
12. Testing calendars.	
Unification of design for the whole application (color, font, size (height / width), alignment)	
• Display the calendar next to the field	
• Correct alignment of all elements and links in the calendar	
13. Testing fields for uploading files.	
• Unification of design for the entire application (color, font, size (height / width), alignment)	
• Alignment of names of uploaded files, thumbnails of files themselves	
14. Testing messages.	
• Message that there are no corresponding items (in tables, when searching, when switching to pages)	
• Spelling, message syntax	
• Correspondence of messages within the meaning depending on the performed action	
• Match the name of the requested action in the error message to the action that the user should take. For example, if you want to select a value from a list, the error message should say 'Please select' and not 'Please enter'	
• Unification of styles (color, size) for the entire application	
• Matching the colors to the message type (red for error messages, green for success messages), if these colors do not contradict the specific requirements of the application	
• Correspondence of field names in error messages and messages about successful completion of the operation with the names of fields, forms, tables, buttons, etc.	
• Correspondence of the order of displaying error messages in accordance with the order of the fields in which errors were found	
• The field containing the error should (preferably) be highlighted	

	WEB
1. Testing the text field.	All texts are correctly aligned.
Field name (spelling, match with open module or page)	All error messages are correct, without spelling or grammatical errors, and correspond to the title of the window.
Alignment of field names (alignment to the left or right (depending on the requirements of the application, indents, identical distances between the title and the field)	A link to the home page should be on every page of the site.
Correct position of the text inside the text, long text does not go beyond the boundaries of the field when typing	Check that there are no broken links and images on the site.
Unification of design (color, font, size (height / width), alignment of margins)	
The location of the entered text within the field (unification, bottom alignment, unless otherwise determined by the specific requirements of the application)	Check the site at different screen resolutions ((640 x 480, 600x800, etc.)
	Check that the user can use the system without annoyance.
2. Testing the button.	Check that TAB is working correctly.
Button name (spelling, matching with action)	The scroll bar should appear only when required.
Clicking' effect (the appearance of the button should change when clicked, if this does not contradict the capabilities of the browser)	If there are any error messages when submitting the form, it should contain the information provided by the user.
Hints name (match with the button name, spelling)	All fields (text, dropdown menus, radio buttons, etc.) and buttons must be keyboard accessible and the user must be able to navigate the site using only the keyboard.

· Unification of design (color, font, size (height / width), highlight color, alignment). The check is carried out both for the button, as for the element, and for the button name	Make sure that the data in the dropdowns is not clipped due to the size of the field, and check if the data is hardcoded or managed by an administrator.
·	
3. Testing radio buttons.	
Unified design for the entire application	
Aligning the location of the radio button with the corresponding name	
Alignment of radio button locations (along the edge)	
4. Testing check boxes.	
Unified design for the entire application	
Aligning the location of the checkbox with the appropriate name	
Correct display of a disabled checkbox	
5. Testing combo boxes.	
Spelling values	
Highlighting when selecting each of the values, when selecting several values at the same time	
· Unification of design (color, font, size (height / width), highlight color, alignment). Checking is carried out both for the field, as an element, and for the values, and their names	
6. Testing the menu.	
Tab highlighting on hover	
Change cursor on hover	
Clicking' effect, if this does not conflict with the capabilities of the browser	
If the work is currently in the selected tab, then in the menu it differs visually (highlighted, underlined)	
Matching names in case the menu is duplicated in several places	
7. Testing windows.	
The appearance of a scroll when decreasing (changing) the size of the browser window	

Preservation of the arrangement of elements when reducing (changing) the browser window, when changing the scale	
Matching the window title depending on the purpose of the page (for example, the window title should be Profile if the user is on a profile page)	
Spelling, naming syntax	
· Unification of names	
8. Testing scrolling.	
Unification of types and types of scrolls on all pages (if there is a custom scroll, it must be applied on all identical forms)	
9. Testing links.	
Unification of styles (in accordance with the site design)	
· Location of links (in accordance with the design of the site). For example, positioning all links to the left or right of the elements	
Names (unification, identity of the names of links of the same purpose, spelling, correspondence with the open module or page, capacity of the name of the link in the allotted block)	
Changing the appearance of the cursor when hovering over a link	
Change the appearance of the link on hover (underline)	
10. Testing tables.	
Unification of design for the whole application (color, font, size (height / width), alignment)	
Title (correspondence with the current module, spelling)	
Alignment of sorting icons in column names	
Alignment of column names, values within the table	
Correct display of long names (corresponding line breaks, abbreviated names (appearance ..., or abbreviated by word)	
Correct display of data after using sorting (column and column sizes are fixed, the text does not break the table structure)	
11. Testing pop-ups.	
Spelling, the syntax of the text located on the pop-up	
Pop-up display in the center of the page, window, form	

Alignment of the text presented on the pop-up	
Correct position of the text on the pop-up: the text should be within the pop-up, the long name should be on a new line, unless otherwise specified by the specific requirements of the application)	
12. Testing calendars.	
Unification of design for the whole application (color, font, size (height / width), alignment)	
• Display the calendar next to the field	
• Correct alignment of all elements and links in the calendar	
13. Testing fields for uploading files.	
• Unification of design for the entire application (color, font, size (height / width), alignment)	
• Alignment of names of uploaded files, thumbnails of files themselves	
14. Testing messages.	
• Message that there are no corresponding items (in tables, when searching, when switching to pages)	
• Spelling, message syntax	
• Correspondence of messages within the meaning depending on the performed action	
• Match the name of the requested action in the error message to the action that the user should take. For example, if you want to select a value from a list, the error message should say 'Please select' and not 'Please enter'	
• Unification of styles (color, size) for the entire application	
• Matching the colors to the message type (red for error messages, green for success messages), if these colors do not contradict the specific requirements of the application	
• Correspondence of field names in error messages and messages about successful completion of the operation with the names of fields, forms, tables, buttons, etc.	
• Correspondence of the order of displaying error messages in accordance with the order of the fields in which errors were found	
• The field containing the error should (preferably) be highlighted	

1. Testing the text field.		
Field name (spelling, match with open module or page)	Data types. Make sure that only valid values can be entered for certain data types, such as currency and dates.	
Alignment of field names (alignment to the left or right (depending on the requirements of the application, indents, identical distances between the title and the field)	Field width. If a specific text box is for a specific number of characters, specify in the user interface that the entered data should not exceed the character limit. (For example, a field that allows 50 characters in the application database should not allow users to enter more than 50 characters in the interface.)	
Correct position of the text inside the text, long text does not go beyond the boundaries of the field when typing	Navigation elements. Make sure all navigation buttons on the page work and redirect users to the correct page or screen.	
Unification of design (color, font, size (height / width), alignment of margins)	Progress indicators. Sometimes the application takes time to complete the assigned work, in such cases, use the progress indicator, it will help you understand that the work is still being done.	
The location of the entered text within the field (unification, bottom alignment, unless otherwise determined by the specific requirements of the application)	Input hints. In a drop-down menu with hundreds of items, when you enter the first letter, only those items that start with this letter should remain, so you save users from looking at a long footcloth of values.	
	Scroll tables. When the data from the table flows to the next page, the scrolling function should allow users to scroll through the data, but not touch all the headers.	
2. Testing the button.	Error logging. When a fatal error occurs on the system, ensure that the application writes the error details to a special log file for later review.	
Button name (spelling, matching with action)	Menu items and mode. Make sure the application only displays menu items that are available in a particular mode.	
Clicking' effect (the appearance of the button should change when clicked, if this does not contradict the capabilities of the browser)	Keyboard shortcuts. Check your keyboard shortcuts to see if they work correctly, regardless of browser, platform, or device.	
Hints name (match with the button name, spelling)	Action confirmation buttons. Make sure the user interface has a valid confirmation button every time the user wants to save or delete an item.	

· Unification of design (color, font, size (height / width), highlight color, alignment). The check is carried out both for the button, as for the element, and for the button name	Compliance with the standards of graphical interfaces	
·	Design elements evaluation: layout, colors, fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links	
3. Testing radio buttons.	Testing with different screen resolutions	
Unified design for the entire application	Testing of localized versions: accuracy of translation (multilanguage, multicurrency), checking the length of names of interface elements, etc.	
Aligning the location of the radio button with the corresponding name	Testing the graphical user interface on target devices: smartphones and tablets.	
Alignment of radio button locations (along the edge)		
4. Testing check boxes.		
Unified design for the entire application		
Aligning the location of the checkbox with the appropriate name		
Correct display of a disabled checkbox		
5. Testing combo boxes.		
Spelling values		
Highlighting when selecting each of the values, when selecting several values at the same time		
· Unification of design (color, font, size (height / width), highlight color, alignment). Checking is carried out both for the field, as an element, and for the values, and their names		
6. Testing the menu.		
Tab highlighting on hover		
Change cursor on hover		
Clicking' effect, if this does not conflict with the capabilities of the browser		
If the work is currently in the selected tab, then in the menu it differs visually (highlighted, underlined)		
Matching names in case the menu is duplicated in several places		
7. Testing windows.		
The appearance of a scroll when decreasing (changing) the size of the browser window		

Preservation of the arrangement of elements when reducing (changing) the browser window, when changing the scale		
Matching the window title depending on the purpose of the page (for example, the window title should be Profile if the user is on a profile page)		
Spelling, naming syntax		
· Unification of names		
8. Testing scrolling.		
Unification of types and types of scrolls on all pages (if there is a custom scroll, it must be applied on all identical forms)		
9. Testing links.		
Unification of styles (in accordance with the site design)		
· Location of links (in accordance with the design of the site). For example, positioning all links to the left or right of the elements		
Names (unification, identity of the names of links of the same purpose, spelling, correspondence with the open module or page, capacity of the name of the link in the allotted block)		
Changing the appearance of the cursor when hovering over a link		
Change the appearance of the link on hover (underline)		
10. Testing tables.		
Unification of design for the whole application (color, font, size (height / width), alignment)		
Title (correspondence with the current module, spelling)		
Alignment of sorting icons in column names		
Alignment of column names, values within the table		
Correct display of long names (corresponding line breaks, abbreviated names (appearance ..., or abbreviated by word)		
Correct display of data after using sorting (column and column sizes are fixed, the text does not break the table structure)		
11. Testing pop-ups.		
Spelling, the syntax of the text located on the pop-up		
Pop-up display in the center of the page, window, form		

Alignment of the text presented on the pop-up		
Correct position of the text on the pop-up: the text should be within the pop-up, the long name should be on a new line, unless otherwise specified by the specific requirements of the application)		
12. Testing calendars.		
Unification of design for the whole application (color, font, size (height / width), alignment)		
• Display the calendar next to the field		
• Correct alignment of all elements and links in the calendar		
13. Testing fields for uploading files.		
• Unification of design for the entire application (color, font, size (height / width), alignment)		
• Alignment of names of uploaded files, thumbnails of files themselves		
14. Testing messages.		
• Message that there are no corresponding items (in tables, when searching, when switching to pages)		
• Spelling, message syntax		
• Correspondence of messages within the meaning depending on the performed action		
• Match the name of the requested action in the error message to the action that the user should take. For example, if you want to select a value from a list, the error message should say ‘Please select’ and not ‘Please enter’		
• Unification of styles (color, size) for the entire application		
• Matching the colors to the message type (red for error messages, green for success messages), if these colors do not contradict the specific requirements of the application		
• Correspondence of field names in error messages and messages about successful completion of the operation with the names of fields, forms, tables, buttons, etc.		
• Correspondence of the order of displaying error messages in accordance with the order of the fields in which errors were found		
• The field containing the error should (preferably) be highlighted		

1. Testing the text field.		
Field name (spelling, match with open module or page)		
Alignment of field names (alignment to the left or right (depending on the requirements of the application, indents, identical distances between the title and the field)		
Correct position of the text inside the text, long text does not go beyond the boundaries of the field when typing		
Unification of design (color, font, size (height / width), alignment of margins)		
The location of the entered text within the field (unification, bottom alignment, unless otherwise determined by the specific requirements of the application)		
2. Testing the button.		
Button name (spelling, matching with action)		
Clicking' effect (the appearance of the button should change when clicked, if this does not contradict the capabilities of the browser)		
Hints name (match with the button name, spelling)		

· Unification of design (color, font, size (height / width), highlight color, alignment). The check is carried out both for the button, as for the element, and for the button name		
·		
3. Testing radio buttons.		
Unified design for the entire application		
Aligning the location of the radio button with the corresponding name		
Alignment of radio button locations (along the edge)		
4. Testing check boxes.		
Unified design for the entire application		
Aligning the location of the checkbox with the appropriate name		
Correct display of a disabled checkbox		
5. Testing combo boxes.		
Spelling values		
Highlighting when selecting each of the values, when selecting several values at the same time		
· Unification of design (color, font, size (height / width), highlight color, alignment). Checking is carried out both for the field, as an element, and for the values, and their names		
6. Testing the menu.		
Tab highlighting on hover		
Change cursor on hover		
Clicking' effect, if this does not conflict with the capabilities of the browser		
If the work is currently in the selected tab, then in the menu it differs visually (highlighted, underlined)		
Matching names in case the menu is duplicated in several places		
7. Testing windows.		
The appearance of a scroll when decreasing (changing) the size of the browser window		

Preservation of the arrangement of elements when reducing (changing) the browser window, when changing the scale		
Matching the window title depending on the purpose of the page (for example, the window title should be Profile if the user is on a profile page)		
Spelling, naming syntax		
· Unification of names		
8. Testing scrolling.		
Unification of types and types of scrolls on all pages (if there is a custom scroll, it must be applied on all identical forms)		
9. Testing links.		
Unification of styles (in accordance with the site design)		
· Location of links (in accordance with the design of the site). For example, positioning all links to the left or right of the elements		
Names (unification, identity of the names of links of the same purpose, spelling, correspondence with the open module or page, capacity of the name of the link in the allotted block)		
Changing the appearance of the cursor when hovering over a link		
Change the appearance of the link on hover (underline)		
10. Testing tables.		
Unification of design for the whole application (color, font, size (height / width), alignment)		
Title (correspondence with the current module, spelling)		
Alignment of sorting icons in column names		
Alignment of column names, values within the table		
Correct display of long names (corresponding line breaks, abbreviated names (appearance ..., or abbreviated by word)		
Correct display of data after using sorting (column and column sizes are fixed, the text does not break the table structure)		
11. Testing pop-ups.		
Spelling, the syntax of the text located on the pop-up		
Pop-up display in the center of the page, window, form		

Alignment of the text presented on the pop-up		
Correct position of the text on the pop-up: the text should be within the pop-up, the long name should be on a new line, unless otherwise specified by the specific requirements of the application)		
12. Testing calendars.		
Unification of design for the whole application (color, font, size (height / width), alignment)		
• Display the calendar next to the field		
• Correct alignment of all elements and links in the calendar		
13. Testing fields for uploading files.		
• Unification of design for the entire application (color, font, size (height / width), alignment)		
• Alignment of names of uploaded files, thumbnails of files themselves		
14. Testing messages.		
• Message that there are no corresponding items (in tables, when searching, when switching to pages)		
• Spelling, message syntax		
• Correspondence of messages within the meaning depending on the performed action		
• Match the name of the requested action in the error message to the action that the user should take. For example, if you want to select a value from a list, the error message should say 'Please select' and not 'Please enter'		
• Unification of styles (color, size) for the entire application		
• Matching the colors to the message type (red for error messages, green for success messages), if these colors do not contradict the specific requirements of the application		
• Correspondence of field names in error messages and messages about successful completion of the operation with the names of fields, forms, tables, buttons, etc.		
• Correspondence of the order of displaying error messages in accordance with the order of the fields in which errors were found		
• The field containing the error should (preferably) be highlighted		

1	2	3	4	1	2
TC ID	**Title**	**Steps to reproduce**	**Expected result**	**Actual Result**	**Pass/ Fail**

Table 1

	The tester must understand functional requirements, business logic, main application scenario, and database design.
	The tester should understand the tables, triggers, storage procedures, display methods, and pointers used for the application.
	The tester must understand the logic of triggers, storage procedures, display methods, and pointers.
	The tester needs to understand which tables are affected when insert, update, and delete operations are performed in the application.
	Database Testing Scenarios:
	Check the name of the database: it must match the specification.
	Check tables, columns, column types and defaults to make sure they all match the specification.
	Check if the column is nullable.
	Check the primary and foreign key of each table.
	Check storage procedures.
	Test if the storage procedure is installed.
	Check the name of the storage procedure.
	Check the names of the parameters, their types and number.
	Check if the parameters are required or not.
	Test the storage procedure by deleting some parameters.
	Check the database if the output is zero - records with zero should be involved.
	Test the storage procedure with simple SQL queries.
	Make sure the procedure returns values.
	Verify the procedure by entering test data.
	Check the behavior of each flag in the table.
	Make sure the data is correctly saved to the database after every input.
	Check the data with every update, delete, and insert operation.
	Check the length of each field. The length on the backend and front end must match.
	Check the names of the QA, UAT and Sales databases. The names must be unique.
	Check the encrypted data in the database.
	Check the base size and response time for each request.
	Check the data displayed on the frontend and make sure it matches the backend.
	Check the integrity of the data by entering invalid values into the database.
	Check your triggers.
	Check the base size and response time for each request.
	Check the data displayed on the frontend and make sure it matches the backend.
	Check the encrypted data in the database.
	Check the integrity of the data by entering invalid values into the database.
	Check the names of the QA, UAT and Sales databases. The names must be unique.
	Check the length of each field. The length on the backend and front end must match.
	Check your triggers.
	Check the data with every update, delete, and insert operation.

	Check the database if the output is zero - records with zero should be involved.
	Check if the parameters are required or not.
	Make sure the procedure returns values.
	Check the name of the storage procedure.
	Check the behavior of each flag in the table.
	Check storage procedures.
	Check if the column is nullable.
	Make sure the data is correctly saved to the database after every input.
	Check the name of the database: it must match the specification.
	Check the primary and foreign key of each table.
	Check tables, columns, column types and defaults to make sure they all match the specification.
	Test if the storage procedure is installed.
	Test the storage procedure by deleting some parameters.
	Check the names of the parameters, their types and number.
	Test the storage procedure with simple SQL queries.
	Verify the procedure by entering test data.

Cross browser, platform		Testing site behavior at or beyond the limits of its anticipated workload (**Stress testing**)		
	Testing in various browsers (Firefox, Chrome, Safari - this is the minimum set): animation, layout, fonts, notifications, etc.	Testing site behavior at increasing workload (**Load testing**)		
	Testing in various OS versions: Windows, Mac, Linux.	Testing the ability to work within or just above the acceptable period **(Stability testing)**		
	Java Script code works in different browsers.	Testing of website performance by increasing the data volume in the database **(Volume testing)**		
	View on mobile devices.	Testing of website performance when multiple users login to it **(Concurrency testing)**		
	Determining the performance, stability and scalability of an application under different workloads.	Testing the behavior of your site when the additional workload is given continuously **(Endurance testing)**		
	Determine if the current architecture can support the application during peak loads.			
	Determining which configuration leads to the best performance.			
	Identifying the application and infrastructure bottleneck. Определение бутылочного горла приложения и инфраструктуры.			
	Determining if the response time has changed for the new version of the application.			
	Evaluation of product and / or hard wear to ensure that it will withstand the forecasted load.			
Mobile Stress Testing				
	Stress testing is aimed at determining the effectiveness of an application's performance under high load conditions. The stress test in this context is focused on mobile devices only.			
	1. High load of the central processor			
	2. Out of memory			
	3. Loading the battery			
	4. Refusals			
	5. Low network bandwidth			
	6. A large number of user interactions with the application (this may require simulating real network conditions)			
Performance testing				
	If a user installs an app and it doesn't appear quickly enough (for example, within three seconds), it can be removed in favor of another app. Time and resource consumption aspects are important success factors for an application, and performance testing is conducted to measure these aspects.			
	1. Application load time			
	2. Processing requests			
	3. Data caching			
	4. The consumption of resources by the application (for example, battery consumption)			
Integration Testing WEB		Integration testing is done to ensure that your application is compatible with third-party services.		
We check the work of third-party modules: payment, sharing, cards.				
Advertising (viewing, ad clicks, analytics).				
Metrics (page clicks, elements view, clicks).				
Test the site in different browsers (IE, Firefox, Chrome, Safari, Opera) and make sure the site displays correctly.				
Make sure the version of HTML you are using is compatible with the respective browser versions.				
Make sure the pictures are displayed correctly in different browsers.				
Make sure the fonts display correctly in different browsers.				
Make sure your Java Script code works in different browsers.				

Check out animated GIFs in different browsers.				
Security testing WEB				
Данная проверка нацелена на поиск недостатков и пробелов с точки зрения безопасности нашего приложения.	This check is aimed at finding flaws and gaps in terms of the security of our application.			
The user cannot log in: under the old password, is locked in the service, has reached the authorization limit, entered someone else's verification code.	Ensure the unauthorized access to secure pages is not possible Verify sessions are automatically killed after prolonged user inactivity Test SSL security functions All attempts at breaking, reporting errors etc. should be logged and stored in a separate file for further analysis. Check the captcha work using automatic scripts Ensure restricted files are not downloadable without appropriate access Ensure there is no login ability while entering wrong password or username			
Pages containing sensitive data (password, credit card number and CVC, security questions, etc.) are opened over HTTPS (SSL).	Verify sessions are automatically killed after prolonged user inactivity			
The password is hidden by asterisks on the pages: registration, "forgot password", "change password".	Test SSL security functions			
Correct display of error messages.	All attempts at breaking, reporting errors etc. should be logged and stored in a separate file for further analysis.	Make sure pages containing sensitive information (password, credit card number, security questions, etc.) are opened over HTTPS (SSL).		
Ending the session after logging out.	Check the captcha work using automatic scripts	Make sure important information (password, credit card number) is displayed encrypted.		
Access to closed sections of the site.	Ensure restricted files are not downloadable without appropriate access	Make sure that the rules for creating passwords are implemented on all authorization pages (registration, "forgot password" page, password change).		
SQL injection.	Ensure there is no login ability while entering wrong password or username	Make sure that if the password is changed, the user cannot log in with the old one.		
Cross-Site Scripting (XSS) vulnerabilities.		Make sure the error messages do not contain any sensitive information.		
HTML injection.		Make sure that if the user is logged out or the session is ended, he cannot use the site.		
Cookies must be stored encrypted.		Check access to private and open pages of the site directly without authorization.		
User roles and content access.		Make sure the View Source option is disabled and not visible to the user.		
		Make sure that the user account is locked out if he entered the password incorrectly several times.		
		Make sure the password is not stored in cookies.		
		Make sure that if any functionality does not work, the system does not display information about the application, server, or database. Instead, an appropriate error message is displayed.		
		Check the site for SQL Injection.		
		Check user rights and their roles. For example, a candidate should not be able to access the admin page.		
		Make sure important transactions are logged and information can be traced.		
		Make sure that session values are displayed encrypted in the address bar.		
		Make sure cookies are stored encrypted.		
		Test your application for brute force attack resistance		
Security testing Mobile				
This check is aimed at finding flaws and gaps in terms of application security.				

1. Testing permissions (access to camera / microphone / gallery / etc.)				
2. User data (passwords) are not transmitted in clear text				
3. In the fields with password entry and password confirmation, the data is hidden by asterisks				