1. **Make a comparative table of functional, non-functional, and change-related types of testing.**

| **Functional**  **Types of Testing** | **What is**  **Being Checked** | **When**  **Applicable** | **Restrictions** | **Peculiarities** |
| --- | --- | --- | --- | --- |
| **Functional Testing** | - If a feature or functionality is doing “what should do”. Essentially, testing if the output of a specific action is the expected result. | - Both during software development or after doing changes to the software (either to correct defects or launch new features).  - Applicable in all testing level phases (unit, integration, system and acceptance). | - Requirements need to be laid out.  - Testers need to have a good understanding of the requirements and software system. | - |
| **GUI**  **Testing** | - The interface according to the requirements (size, font color, consistent behavior). | - Right after development implements the software interface | - Time consuming due to the effect that we have to test different interface scenarios in different devices | - |
| **Security and Access Control Testing** | - The security of the system (risks associated with hacker attacks, viruses, and access to confidential information). | - Before launching the software system into production | - | - |
| **Interoperability**  **Testing** | - The ability of the application to interact with other systems or components. It also covers compatibility testing (testing the same system in different software versions) | - During the integration testing level phase | - | - |

| **Non-functional Types of Testing** | **What is**  **Being Checked** | **When**  **Applicable** | **Restrictions** | **Peculiarities** |
| --- | --- | --- | --- | --- |
| **Performance Testing** | - Evaluates how a system performs under a particular workload. There are different types of tests that can be performed and help testing reliability, speed, and scalability of the software. | - Close launching the software system into production  - Or during system and acceptance testing level phases | - These testing types can be overlooked depending on budget and resources available | - It can be impacted by external dependencies and other third-party products that can affect the testing |
| **Installation Testing** | - Test if the installation / configuration of a software system is successful. It also covers testing updates or uninstalling the software | - During system and acceptance testing level phases | - | - Testing may need to happen in a wide range of devices and may result in overlooking problems in certain devices |
| **Usability**  **Testing** | - Test to confirm the level of convenience, clarity, and user experience of the software system | - During acceptance testing level phases | - Requires a considerable sample of different users to test different user experiences | - Usability is subjective and is dependent on the user preferences, so it can produce unreliable results |
| **Failover and Recovery Testing** | - Tests if the product can successfully recover from failures from software defects, faulty hardware or other problems | - During system  testing level phases | - Difficult to recreate or come up with failure scenarios to properly test | - |
| **Configuration Testing** | - Test that a software system can still operate under different system configurations (operating systems, network configurations, etc.) | - During system or acceptance  testing level phases  - Right before releasing the software product |  |  |

| **Change-related Types of Testing** | **What is**  **Being Checked** | **When**  **Applicable** | **Restrictions** | **Peculiarities** |
| --- | --- | --- | --- | --- |
| **Smoke**  **Testing** | - Short cycle of tests that confirm the software is still performing it’s main functions after a change in the code (new or to fix a defect) | - Smoke test is always applicable after changes to the code happen | - It’s not an exhaustive test of the features, it’s just designed to cover the main functionalities of the software | - These tests are designed to be quick and give the team immediate feedback on the software performance |
| **Regression Testing** | - Aims to check that after changes are done into the code software all the overall functions are still working as expected. | - Every time there is considerable update or change to the software code: bug fixes, code improvements, or new features. | -This is a broader test aimed to validate the entire software system.  - Can be very time consuming for teams, so it’s dependent on time and resources available | - Regression Tests can be functional and non-functional  - Regression Tests are one of tests that teams try to automate the most |
| **Re-test** | - Confirming that a defect has been fixed by re-testing the same steps that identified the defect issue in the first place | - Whenever a defect is fixed from the software code | - Test cases and test results need to be well documented, so that the re-test can be as accurate as possible | - |
| **Build Verification Testing** | - Test if a created build version meets  the quality criteria and is stable before it is released for further testing | - Always applicable when a new build is created | - It’s not an exhaustive test of the features, it’s designed to cover the main functionalities of the build | - |
| **Sanity**  **Testing** | - It verifies that recent code changes have not impacted specific functionalities. Compared to regression testing, it is a more narrow and quick test | - Typically executed after each build or minor update or change in the software code |  | - Usually performed manually  - Part of regression testing, but focus on specific functionalities |

1. **Explain the difference between regression and retesting (5 sentences).**

Regression Testing and Re-test are both changed-related types of testing, but have different goals. Re-test happens after a defect is fixed, and consists in confirming that the defect has been successfully taken care of and the functionality is working as expected. Regression Testing aims to check if previously existing functionalities are still working as expected after changes are made in the software (bugs are fixed, new features launched, code changed).

1. **Do you think it is possible to perform only functional testing for a product without checking non-functional requirements?**

**If yes - in what cases?**

**If not, why not?**

**Support your answer.**

I believe it would technically be possible to only perform functional tests in very particular cases. Since, functional tests ensure that the software meets its functional requirements, in other words the software is doing what is expected to do, by doing it so the software is going to be functional. This could ultimately be an approach used in scenarios where there is a lack of resources (developers/testers to support our development and testing phase), or where there are either budget or time limitations. Also, it could be part of the strategy, in an early phase, to give priority and focus exclusively on functional testing, hence making sure that we have built a functional software. After that, then focus on the non-functional requirements.

However, it would be completely unrecommended to launch a software product without looking at non-functional requirements. Non-functional requirements cover crucial properties of the system such as usability, security and reliability that will be crucial in providing the best experience to the users. By launching a product overlooking non-functional requirements we are compromising the experience and perspective the user has of our software product.

1. **How do you see the need for smoke testing? Is it always appropriate?**

Smoke tests are a quick tests cycle that confirm the software main features are running and performing as expected after building or making changes to the code (defects fixed or new feature build).

Smoke tests are mainly appropriate right after a new software build is made (and is about to be launched) or after a new software code is deployed to production to make sure everything is running as expected.

However, I would say running a smoke test is always appropriate whenever there is a software code update.

**5. You are the founder of a startup planning to launch a mobile application for sharing cat photos on iOS and Android devices. Users can upload photos of cats but they cannot upload photos of other animals/people/objects. Users can add friends and leave likes and comments.**

**Write 5 functional test cases that would test the application.**

| **Test Case 1:** Upload a cat photo on CatSharing APP | |
| --- | --- |
| **Preconditions:**   * Tester has App installed in the test device * Tester has WIFI connection in the test device * Cat photo available in the device | |
| **Steps** | **Expected Results** |
| 1. User opens CatSharing App on test device | CatSharing App Opens |
| 2. Users signs in using email and password | User profile appears |
| 3. User clicks on the “Upload” button in the bottom menu | New frame to upload photos from device appears |
| 4. Select a cat photo from your device photo gallery |  |
| 5. Click on “Publish” button on the bottom of the screen frame | A green pop-up with the message “Photo Uploaded” appears |

| **Test Case 2:** Upload a dog photo on CatSharing APP | |
| --- | --- |
| **Preconditions:**   * Tester the has App installed in the test device * Tester has WIFI connection in the test device * Dog photo available in the device | |
| **Steps** | **Expected Results** |
| 1. User opens CatSharing App on test device | CatSharing App Opens |
| 2. Users signs in using email and password | User profile appears |
| 3. User clicks on the “Upload” button in the bottom menu | New frame to upload photos from device appears |
| 4. Select a dog photo from your device photo gallery |  |
| 5. Click on “Publish” button on the bottom of the screen frame | A red pop-up with the message “Sorry, you can only upload Cat photos!” appears |

| **Test Case 3:** Make a friend request on CatSharing APP | |
| --- | --- |
| **Preconditions:**   * Tester the has App installed in the test device * Tester has WIFI connection in the test device | |
| **Steps** | **Expected Results** |
| 1. User opens CatSharing App on test device | CatSharing App Opens |
| 2. Users signs in using email and password | User profile appears |
| 3. User clicks on the search icon button in the top menu | Search bar expands |
| 4. Users writes the friend username in the search bar | Friend username and photo icon appears |
| 5. Click on the friend photo icon displayed in the screen | Private profile of the friend appears |
| 4. Click on the “Add as Friend” button next to the username | A message “Pending Acceptance” appears in the button |

| **Test Case 4:** Comment a photo from another friend on CatSharing APP | |
| --- | --- |
| **Preconditions:**   * Tester the has App installed in the test device * Tester has WIFI connection in the test device * Ensure the test environment is set up with at least one user account, having a minimum of one friend added and two photos uploaded, to validate friend and photo-related functionalities. | |
| **Steps** | **Expected Results** |
| 1. User opens CatSharing App on test device | CatSharing App Opens |
| 2. Users signs in using email and password | User profile appears |
| 3. User clicks on the search icon button in the top menu | Search bar expands |
| 4. Users writes the friend username in the search bar | Friend username and photo icon appears |
| 5. Click on the friend photo icon displayed in the screen | Profile of the friend with uploaded photos appears |
| 6. Click on the first cat photo in the friend profile | Cat photo frame expands |
| 7. Write “Cute Photo!” on the description field below the cat photo | Comment is added below the friend cat photo |

| **Test Case 4:** Like a photo from another friend on CatSharing APP | |
| --- | --- |
| **Preconditions:**   * Tester the has App installed in the test device * Tester has WIFI connection in the test device * Ensure the test environment is set up with at least one user account, having a minimum of one friend added and one photo uploaded, to validate friend and photo-related functionalities. | |
| **Steps** | **Expected Results** |
| 1. User opens CatSharing App on test device | CatSharing App Opens |
| 2. Users signs in using email and password | User profile appears |
| 3. User clicks on the search icon button in the top menu | Search bar expands |
| 4. Users writes the friend username in the search bar | Friend username and photo icon appears |
| 5. Click on the friend photo icon displayed in the screen | Profile of the friend with uploaded photos appears |
| 6. Click on the first cat photo in the friend profile | Cat photo frame expands |
| 7. Click on the white thumbs up icon below the cat photo | Thumbs up icon changes color from white to yellow |

**6. Write what non-functional requirements you would like to apply to your startup's product.**

**Describe the tests that would check them (3-5 examples).**

* The user can download successfully the App from a Store | **Installation Testing**
* A user can load the App in less than 1 minute | **Performance Testing: Performance and Load Testing**
* The user can upload a cat photo in 5 seconds | **Usability Testing**
* The App is responsive for both IOS and Android systems | **Performance Testing**