ANUJA PANDEY

1. List down the things you cannot test on an Emulator but on Real Devices?

   Ans- The real device allows the testers to test almost all the real-time scenarios which can be tested for the mobile applications. These devices 0are operated using fingers and simulate real-life usage. Emulators cannot be used in such a situation.

   The real device allows the testers to test even usability issues like the look and feel of the application, color resolution of the screen, whether the picture is bright or not under both day and night conditions but The emulator/simulator is not able to properly emulate/simulate the exact color display of the devices when the real device is in sunlight or in black.

   The emulator/simulators are not able to simulate the battery issues. Real world devices can easily perform the same.

   The emulator/simulators are not able to simulate the incoming interrupts for SMS as well as the incoming calls. Real world devices can easily simulate incoming interrupts.

   The memory available at the emulator/simulator tends to be far more than the real devices so this may create misconception for the users who would be using the same validations whereas real device will give the actual image of the phone hence would be more useful in testing.

2. What All Major Networks to Be Considered While Performing Application Testing?

   Ans- Application should be tested on 4G, 3G, 2G and WIFI. 2G is a slower network, its good if we verify our application on slower network also to track our application performance.

3. What are the google core quality guidelines for Mobile Apps, Explain guidelines for Functionality?

   Ans- Android users expect high-quality apps. App quality directly influences the long-term success of the app in terms of installs, user ratings and reviews, engagement, and user retention. We assess the core aspects of quality in our app, through a compact set of quality criteria and associated tests. All Android apps should meet these criteria.

   Before publishing any apps, test them against these criteria to ensure that they function well on many devices, meets Android standards for navigation and design, and are prepared for promotional opportunities in the Google Play store.

   The guidelines for Functionality are:

1. Permissions-

The app requests only the absolute minimum permissions that it needs to support core functionality.

The app does not request permissions to access sensitive data (such as Contacts or the System Log) or services that can cost the user money (such as the Dialer or SMS), unless related to a core capability of the app.

2. Install Location-

The app functions normally when installed on SD card (if supported by app).Supporting installation to SD card is recommended for most large apps (10MB+).

3. Audio-

Audio does not play when the screen is off, unless this is a core feature (for example, the app is a music player).

Audio does not play behind the lock screen, unless this is a core feature.

Audio does not play on the home screen or over another app, unless this is a core feature.

Audio resumes when the app returns to the foreground, or indicates to the user that playback is in a paused state.

4. UI and Graphics-

The app supports both landscape and portrait orientations (if possible).

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.

Minor letterboxing to compensate for small variations in screen geometry is acceptable.

5. User/App States-

The app should not leave any services running when the app is in the background, unless related to a core capability of the app.

For example, the app should not leave services running to maintain a network connection for notifications, to maintain a Bluetooth connection, or to keep the GPS powered-on.

The app correctly preserves and restores user or app state.

4. Write down test cases for Push notification.

   Ans- Test Cases for Push Notifications are:

   Verify the notification received even if the app is open.

   Verify the notification received even if the app is closed.

   Verify the notification received even if the device is locked.

   Verify the notification received even if another app is in use.

   Verify the notification is clickable & opens up the notification and redirect to the app.

   Verify the notification should get removed from notification bar once the notification is opened.

   Verify repeated notification multiple times and check notification is in the order.

   Verify the recent notification should always be on the top in a stack.

5. What do you understand by Device and OS fragmentation in mobile application testing?

   Ans- Mobile device fragmentation is a phenomenon that occurs when some mobile users are running older versions of an operating system, while other users are running newer versions.

   Mobile device fragmentation can be a problem for software developers who must create different versions of the same app in order to make sure it works correctly with different versions of a given OS.

   It can also be a problem for IT departments because different operating versions have different capabilities, which can make them harder to manage and secure.

   Mobile device fragmentation is often associated with Android, Google’s mobile OS. It is not as much of an issue with iOS devices.

   2. OS Fragmentation

OS Fragmentation arises due to different versions of OS. Both Android & iOS has around 10+ different versions available.

Developers have to make sure that their app is running on all different versions of OS(whether it is Android, iOS or any other OS).

There should be seamless user experience regardless of the OS installed in device.

6. What is deep linking? What do you understand by deferred deep linking?

   Ans- In the context of mobile apps, deep linking consists of using a uniform resource identifier (URI) that links to a specific location within a mobile app rather than simply launching the app.

   Deferred deep linking is one aspect of mobile deep linking. It describes the principle of deep linking into an app that is not yet installed. In this case, deep linking will be "deferred" until the application will be installed by the user. This implies that clicking (or otherwise invoking) the deep link causes:

   An app store to open (Google Play/IOS or Windows App Store depending on the user's device) to enable the user to install the app

   Once the app is installed, the link is invoked with its original URL and parameters so that the newly installed app can handle the invocation.