DUMPSARENA

Monetize Android Applications

Android AND-403

Version Demo

Total Demo Questions: 11

Total Premium Questions: 111

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

QUESTION NO: 1

Approximately how much time it takes for an application that is published to be available to the users?

A. It will be available within minutes to users.

B. It takes around one week to be available to the users.

C. As soon as Google Play finishes the application review process.

D. It takes a day for an application to be available to the user.

ANSWER: A

Explanation:

It will be available within minutes to users. Once a release has been successfully published and made available in Google Play, its availability is generally propagated quickly, so users can normally find and install the application within minutes. This refers to the post-publication availability of the release, rather than the time required to prepare a release, submit it for review when review applies, or wait for any developer-selected publishing controls. Google Play allows developers to publish releases immediately after they are ready, and published changes are then distributed through Play's systems. The precise experience for an individual user can still depend on factors such as device compatibility, country or region targeting, track eligibility, and temporary Play Store caching, but these do not change the expected approximate availability stated in the question. Google's Play Console documentation describes publishing as the action that makes changes live on Google Play, while its release guidance explains how releases are made available to eligible users. See Publish app changes with managed publishing and Prepare and roll out a release.

QUESTION NO: 2

Which of the following should an application do when processing a completed Google Play purchase? (Choose three)

A. Verify the purchase before granting the entitlement.

B. Grant the purchased content or access.

C. Acknowledge or consume the purchase as appropriate.

D. Grant the item when the purchase screen is opened.

ANSWER: A B C

Explanation:

The app should verify the purchase, grant the corresponding entitlement, and acknowledge or consume the purchase according to its product type. Verification, preferably with a secure backend for valuable entitlements, helps ensure that the purchase is valid and associated with the expected product. The entitlement should be granted only after the app has processed the purchase successfully.

A non-consumable product or subscription purchase must be acknowledged, while a consumable product is consumed after delivery. The application should not grant items merely because the user opens the purchase screen, because no completed transaction has been confirmed at that point. See Google Play Billing security guidance and Google Play Billing integration.

QUESTION NO: 3

Which of the following can only be performed before an application release and not after? (Choose two)

A. Create a product list to be purchased through in-app billing.

B. Remove log messages.

C. Build a signed release of your application.

D. Create a Google Wallet merchant account.

ANSWER: B C

Explanation:

Remove log messages. is a pre-release activity because logging intended for development and debugging should be reviewed and removed or appropriately disabled before distributing the production build. Release preparation includes ensuring that the delivered artifact does not expose unnecessary diagnostic information or incur avoidable logging overhead. This work must be reflected in the application package that is built and submitted for release; it cannot retroactively alter an already published version.

Build a signed release of your application. must also occur before release. Android application packages or app bundles distributed through Google Play must be signed so that Android can verify the app's identity and establish a trusted update path. The signing configuration and resulting release artifact are part of the publishing process, and a published build cannot be transformed into a signed release afterward. Any later correction requires building, signing, and publishing a new version with an incremented version code. Android's release guidance describes preparing a release build and signing it before distribution, while its app-signing documentation explains the role of signing in installation and updates. See Prepare your app for release and Sign your app.

QUESTION NO: 4

Which of the following is not a proper method to publish your Android application?

A. Releasing to an application marketplace.

B. Sending it to through mail.

C. Uploading it to website.

D. Sending it through e-mail.

ANSWER: B

Explanation:

Sending it to through mail. is the intended incorrect choice because it does not describe a usable electronic distribution channel for an Android application package. An Android app must be delivered as an installable digital artifact, such as an APK for direct distribution or an Android App Bundle submitted to an app store for device-specific APK generation. Physical mail cannot itself provide users with an installable application or the updates, compatibility handling, and security controls associated with normal Android distribution.

Android's supported publishing workflow centers on digitally uploading an app release to a distribution service, most commonly Google Play. Developers may also distribute APKs directly through controlled digital channels when appropriate, provided users can obtain the file and explicitly allow installation from that source. Google documents the Play publishing process and the release artifacts used for app distribution in its Publish your app documentation. It also explains that installations from sources outside Google Play require the user to permit that source under the device's security settings in its Android Help guidance for unknown apps. Therefore, the malformed/physical-mail method is not a proper way to publish an Android application.

QUESTION NO: 5

Which advertisement format is designed to give the user an in-application reward after viewing or interacting with an ad?

A. Banner ad

B. Rewarded ad

C. Native ad

D. App open ad

ANSWER: B

Explanation:

A rewarded ad is designed to offer the user an in-app reward, such as extra game currency, an additional life, or access to a feature, after the user chooses to engage with the ad. The reward should be clearly described before the user starts the ad experience, and the application should grant it only after receiving the appropriate reward callback from the advertising SDK.

Unlike a banner ad, a rewarded ad is not intended to occupy a persistent section of the screen. Unlike an interstitial ad, its monetization model is based on a user voluntarily choosing to receive a defined reward. See Google Mobile Ads SDK rewarded ads documentation.

QUESTION NO: 6

Which of the following is a requirement to use In-App Billing service on Google Play? (Choose three)

A. A Google Wallet merchant account.

B. A publisher account.

C. A published application.

D. Google Play License Verification Library.

ANSWER: A B C

Explanation:

In-App Billing through Google Play requires a Google Play developer (publisher) account because the application and its products are configured and managed in Play Console. The developer account establishes the identity authorized to distribute the app and use Google Play's monetization services. A Google Wallet merchant account, now generally represented by the payments profile associated with the developer account, is required to receive proceeds from paid in-app products and subscriptions. This payment setup supplies the financial and tax information Google needs to process developer payouts. A published application is also required in the traditional In-App Billing workflow because products are created for a specific app package that has been uploaded to Google Play; the app's release and product configuration are tied to its Play Console entry. Current testing workflows can use test tracks, but the application must still be created and uploaded in Play Console before billing products can be configured and tested. Google's current documentation describes creating a Play Console developer account, configuring a payments profile for monetization, and integrating the billing system for apps distributed through Google Play. See Google Play Console Help: Set up a payments profile and Android Developers: Integrate Google Play Billing.

QUESTION NO: 7

In which of the following sections on Google Play Developer Console you can see an application's license key?

A. APK

B. Store Listing

C. In-app Products

D. Services and APIs

ANSWER: D

Explanation:

Services and APIs is correct. In the Google Play Developer Console layout used for this material, an app's Google Play Licensing public key was displayed under the app's Services and APIs area, in the licensing section. This key is the Base64-encoded RSA public key associated with the application; it is copied into the app or licensing-verification implementation so

that license responses obtained from Google Play can be verified cryptographically. The key identifies the publisher-side licensing configuration for that particular app and is not a signing key or a value generated in the APK itself. Google's licensing documentation describes retrieving the application's public license key from Play Console as part of configuring the Licensing service, then embedding that value where the app's license-checking code can use it. In newer Play Console navigation, Google has reorganized some menus, but the underlying purpose and app-specific public licensing key remain the same. See the Android developer guide for setting up Google Play Licensing and the overview of the Google Play Licensing service.

QUESTION NO: 8

Which of the following is the correct place of your application to use the license key verification?

A. Java Classes.

B. AndroidManifest.xml file

C. Layout resource file of the main activity.

D. It is not used in the application at all.

ANSWER: A

Explanation:

Java Classes. is correct because Google Play license verification is implemented as application runtime code. The application integrates the Google Play Licensing Library, creates a license-checking component, supplies its licensing public key, and initiates a check through Java or Kotlin code in an app component such as an activity, service, or another appropriate application class. The verification workflow must execute while the app runs so that it can contact the Google Play licensing service, receive the signed response, validate that response, and apply the app's chosen policy for licensed, unlicensed, or retry conditions. The public key used by the app is obtained from the Google Play Console and is used by the client-side licensing code to verify the integrity and origin of the licensing response. Therefore, Java classes are the appropriate location for the licensing logic and its callbacks. Google's licensing overview describes this client-library integration and the runtime license-check request flow: Google Play Licensing Overview. The library's implementation guidance, including the use of a license checker and policy in application code, is also documented at Setting Up Google Play Licensing.

QUESTION NO: 9

Which of the following are drawbacks for publicly publishing your application through e-mail? (Choose three)

A. The user must enable installation from "Unknown Source" on the Android device.

B. The application only reaches a selected set of users that are specified in the email by the developer prior to sending an email.

C. Any updated version of the application should be re-attached and sent to the users. There is no automatic updating.

D. It is a quick way to send the application to a limited number of users.

ANSWER: A B C

Explanation:

Distribution by e-mail is an informal APK-sharing method rather than a managed application-distribution channel. The user must enable installation from "Unknown Source" on the Android device because an APK received as an attachment is outside the trusted app-store installation flow. On current Android versions, this permission is granted to the particular app acting as the installer, such as an e-mail client or file manager, under the "install unknown apps" setting. This creates an additional setup and trust step for recipients before installation can proceed.

The application only reaches a selected set of users that are specified in the email by the developer prior to sending an email. E-mail delivery depends on the developer maintaining recipient lists and recipients receiving, opening, and manually installing the attachment; it does not provide public discovery, store listing, or broad availability. Any updated version of the application should be re-attached and sent to the users. There is no automatic updating. Each release therefore requires a new distribution message and a user-initiated installation, making version management and timely adoption difficult. Android's documentation describes the security controls for installs from unknown sources and recommends managed distribution mechanisms for application delivery. See Android Developers: Unknown sources and Android Developers: Distribute your app.

QUESTION NO: 10

Which of the following are correct about consuming a one-time product purchase? (Choose two)

A. It allows the user to purchase the consumable item again.

B. It marks a consumable purchase as processed.

C. It permanently removes a user's subscription.

D. It should be used for a non-consumable premium-feature unlock.

ANSWER: A B

Explanation:

Consuming a purchase is appropriate for a consumable one-time product, such as virtual currency, extra lives, or a single-use hint. When the application consumes the purchase, Google Play records that the item has been used and permits the user to buy that product again. Consumption also completes the required post-purchase processing for that consumable item.

A non-consumable entitlement, such as permanently removing advertisements, should not be consumed because the user must retain access after the purchase. Subscriptions are recurring products and are not consumed through the one-time-product consumption flow. See Google Play one-time purchase lifecycle.

QUESTION NO: 11

Which of the following is the correct Android API on which the In-app billing API is supported on?

A. API 2.1 or higher

B. API 2.2 or higher

C. API 1.5 or higher

D. API 16 or higher

ANSWER: B

Explanation:

API 2.2 or higher is correct in the context of the legacy Google Play In-app Billing API covered by this certification material. Android 2.2 corresponds to API level 8, which was the minimum platform version required for an application to use the original In-app Billing service supplied through Google Play. An app targeting this billing integration declared the billing permission in its manifest and communicated with the Play Store billing service, while devices running Android 2.2 or later provided the required platform support. This requirement concerns device compatibility for the original service, rather than the app's chosen target SDK version. Google later replaced the original service integration with the Google Play Billing Library, whose current releases have newer minimum-SDK requirements. Therefore, when interpreting this exam question, the relevant historical API requirement is Android 2.2 (API level 8) or later. Android's platform-version reference identifies Android 2.2 as API level 8: Android SDK Platform release notes. For current implementation guidance and the evolution to the modern billing client, see Integrate the Google Play Billing Library.