

What's New in StoreKit

Session 303

Pete Hare, App Store Engineer
Ross LeBeau, App Store Engineer

What's New in StoreKit



NEW

What's New in StoreKit



NEW

Promoting in-app purchases

What's New in StoreKit



NEW

Promoting in-app purchases

Server-to-server subscription notifications

What's New in StoreKit



NEW

Promoting in-app purchases

Server-to-server subscription notifications

Detailed subscription status information

What's New in StoreKit



NEW

Promoting in-app purchases

Server-to-server subscription notifications

Detailed subscription status information

Responding to reviews

What's New in StoreKit



NEW

Promoting in-app purchases

Server-to-server subscription notifications

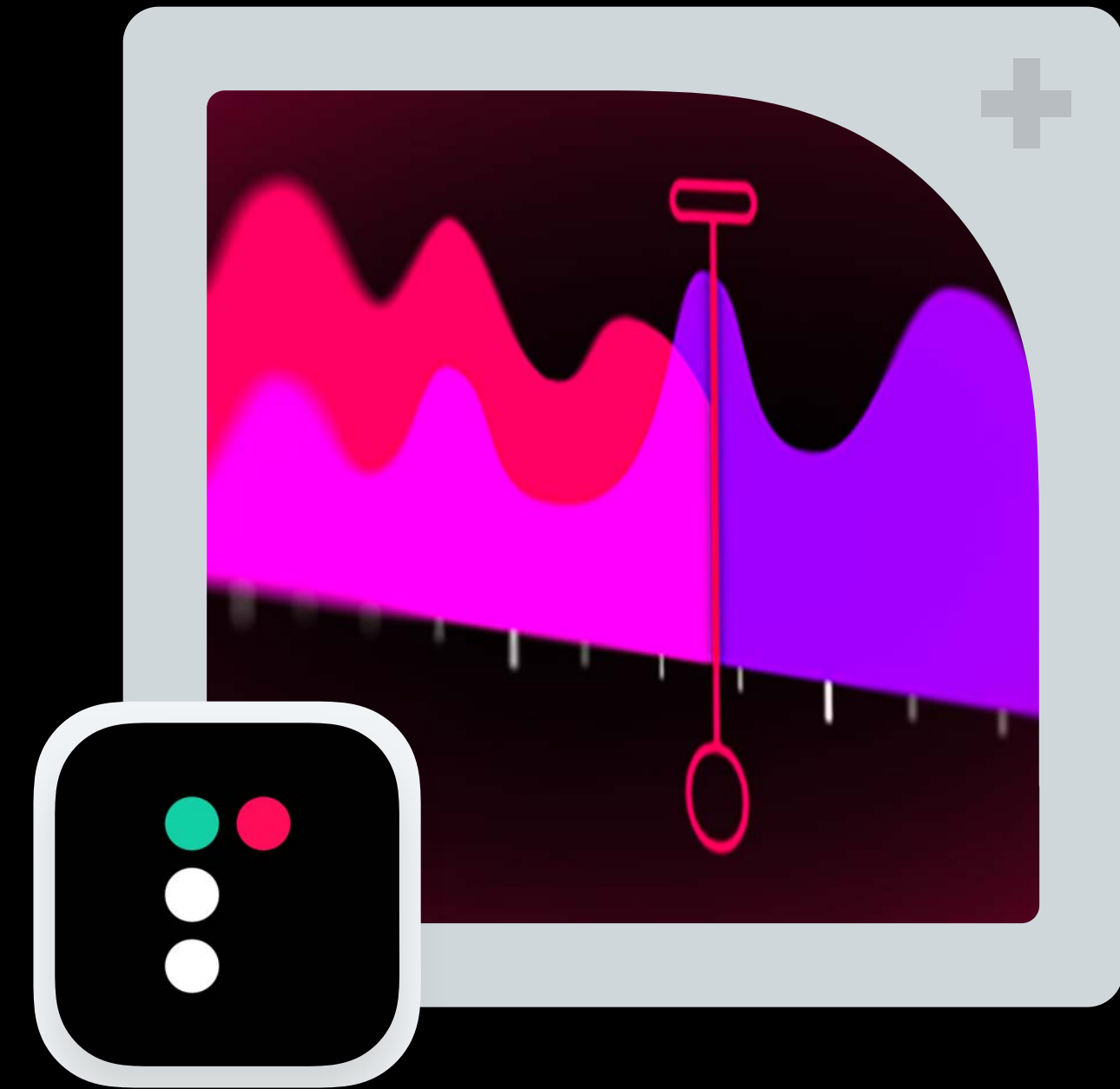
Detailed subscription status information

Responding to reviews

Asking for ratings and reviews

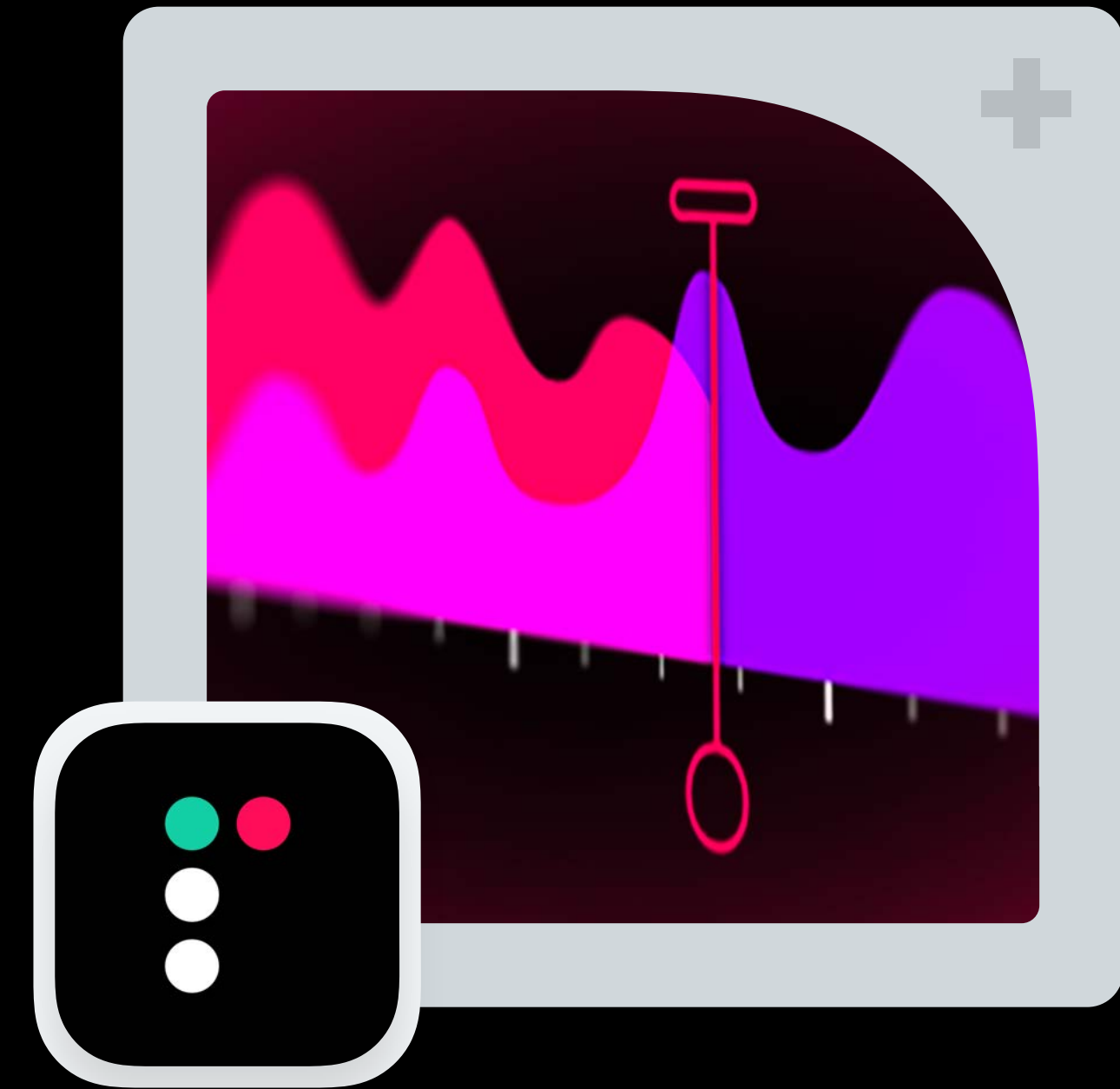
Review of In-App Purchases

In-App Purchase Overview



In-App Purchase Overview

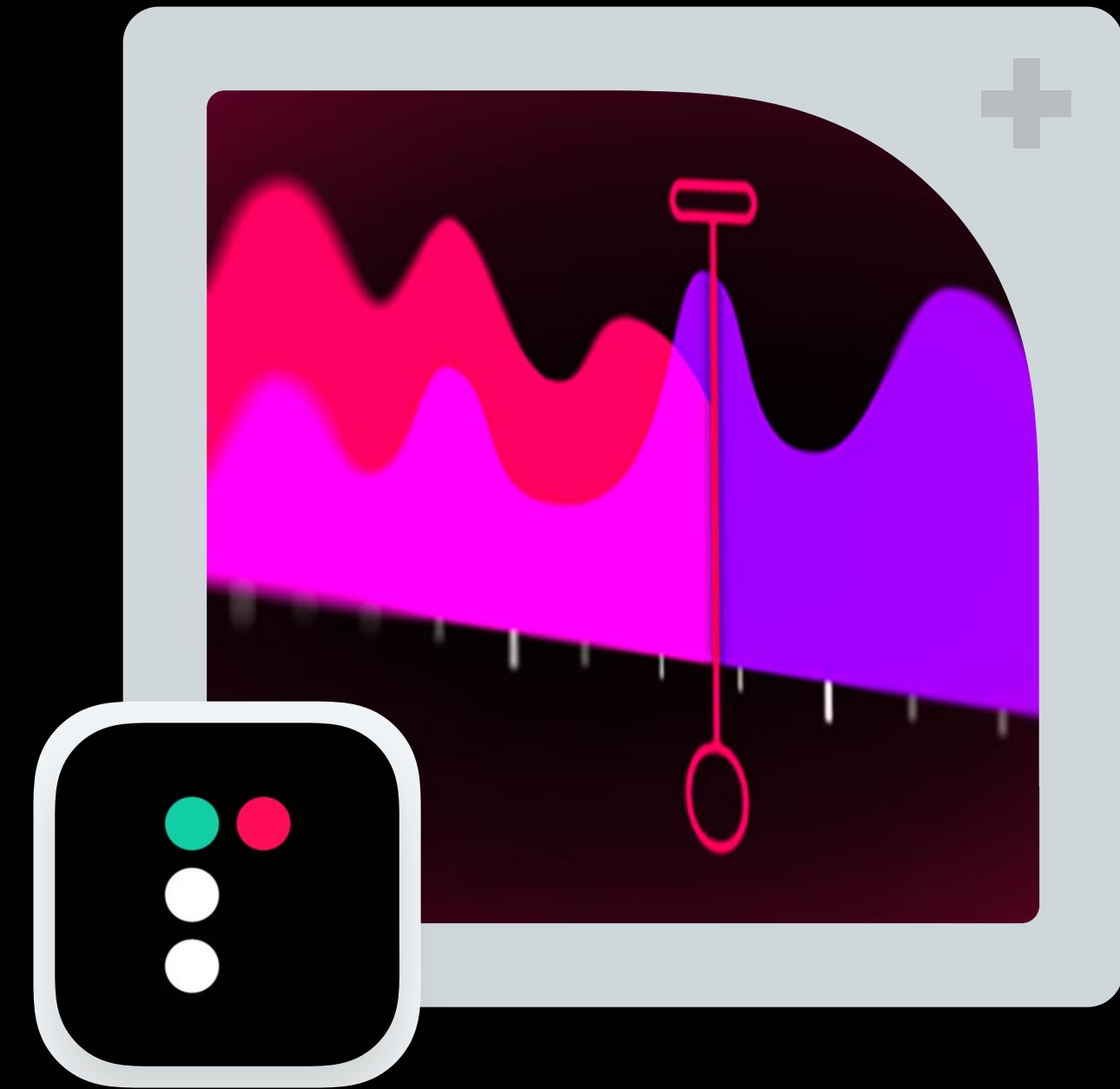
Digital content or service bought in-app



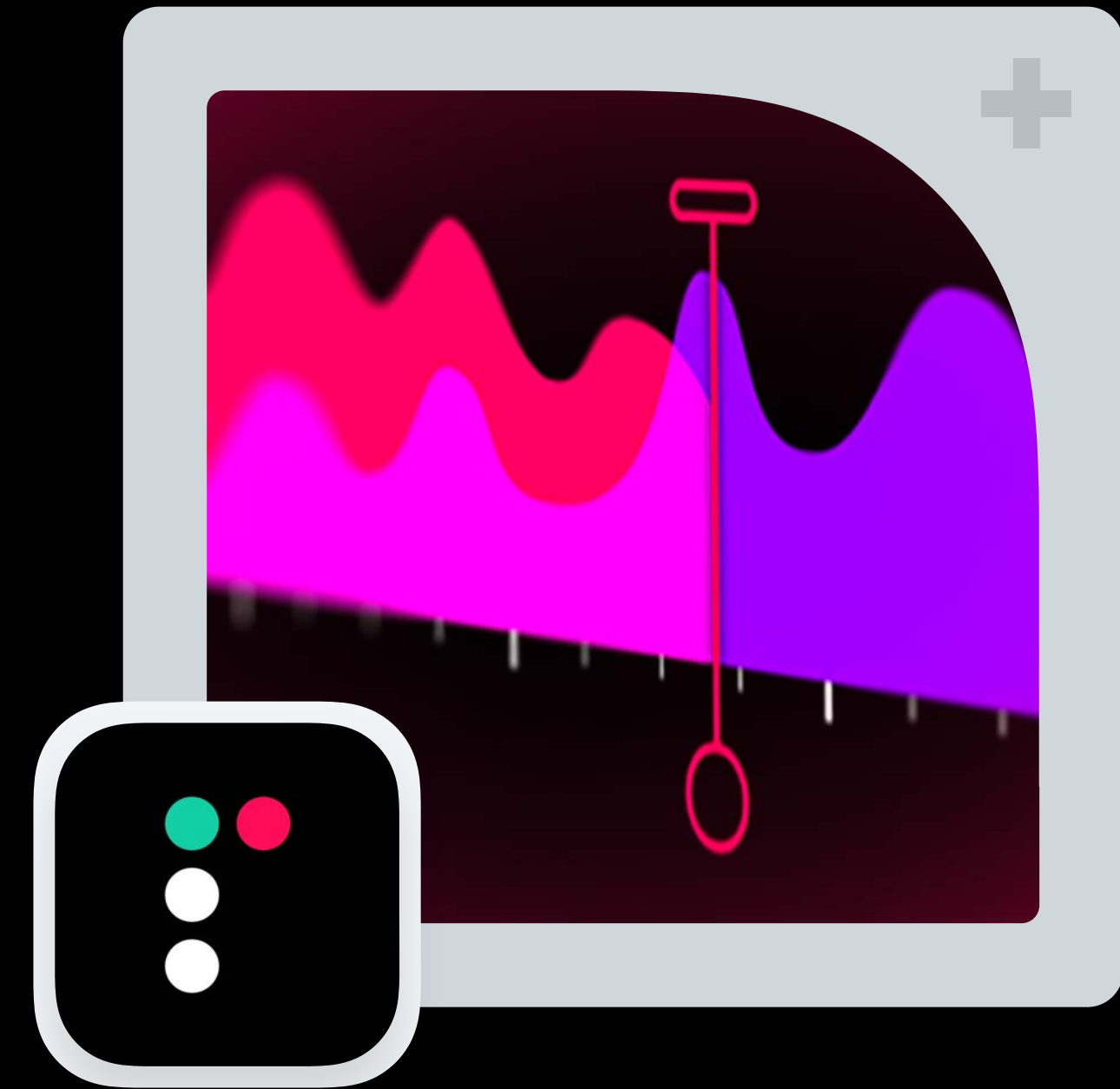
In-App Purchase Overview

Digital content or service bought in-app

Not for physical goods

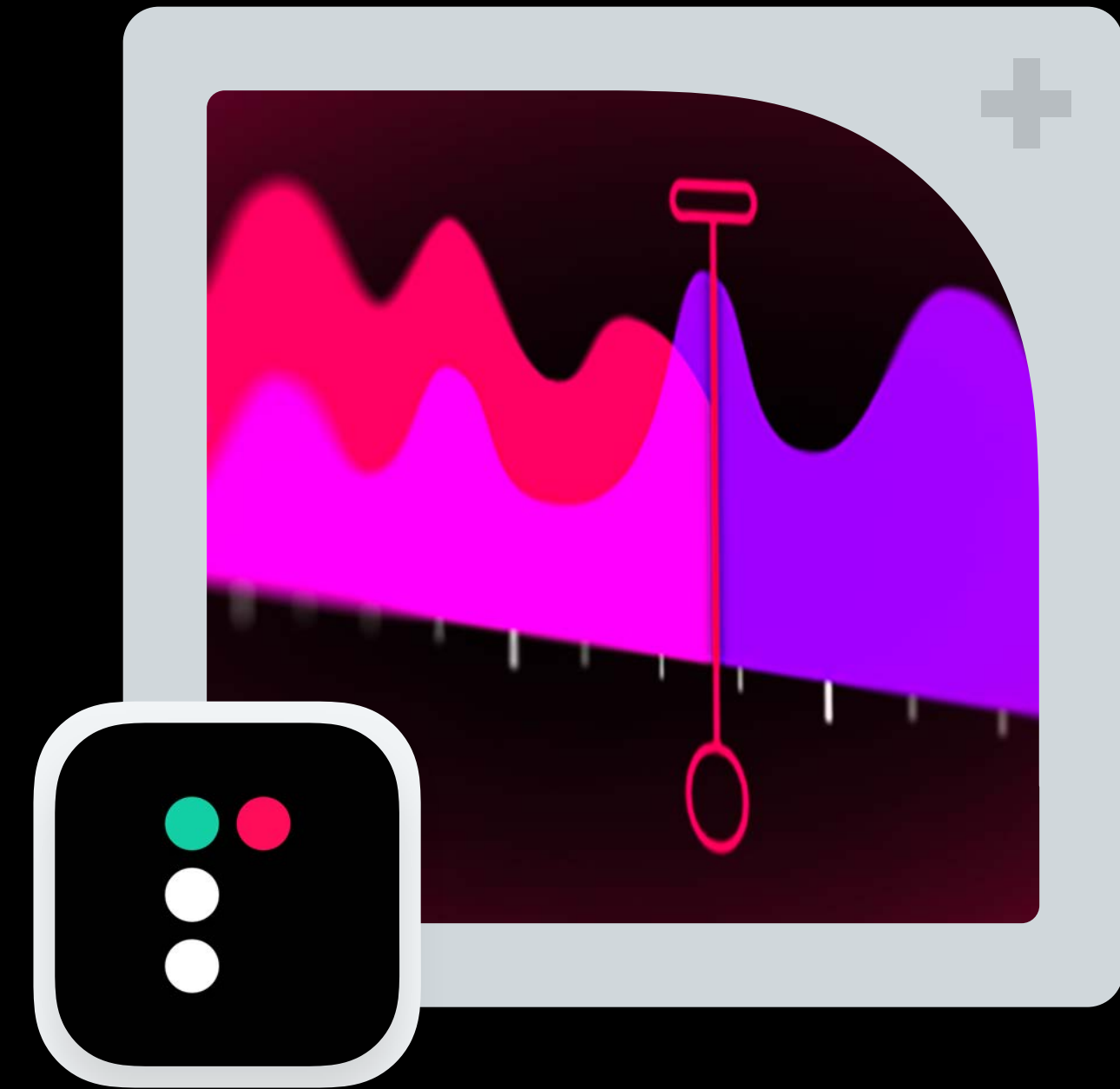


Types of In-App Purchases



Types of In-App Purchases

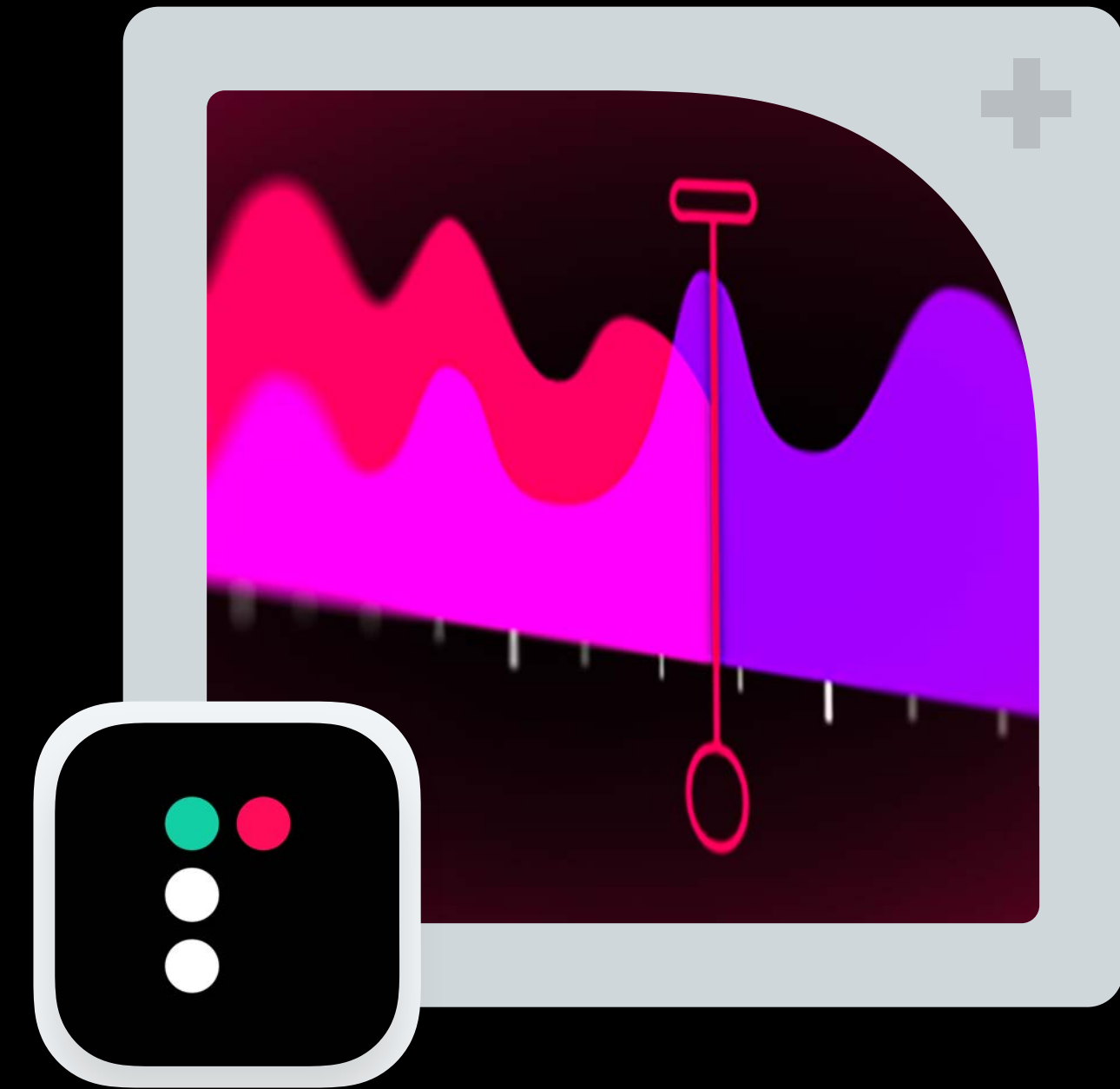
Consumable products



Types of In-App Purchases

Consumable products

Non-consumable products

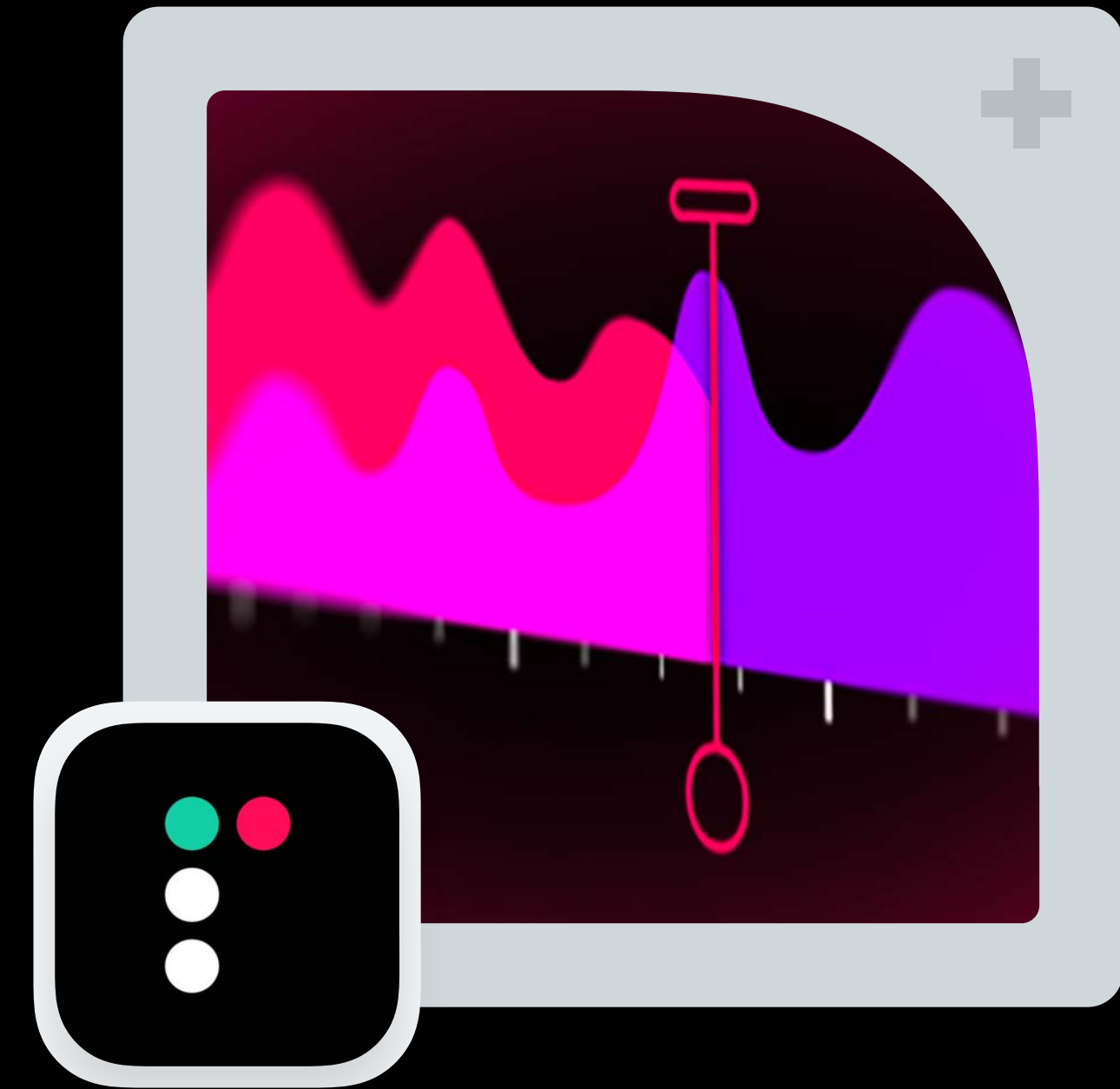


Types of In-App Purchases

Consumable products

Non-consumable products

Non-renewing subscriptions



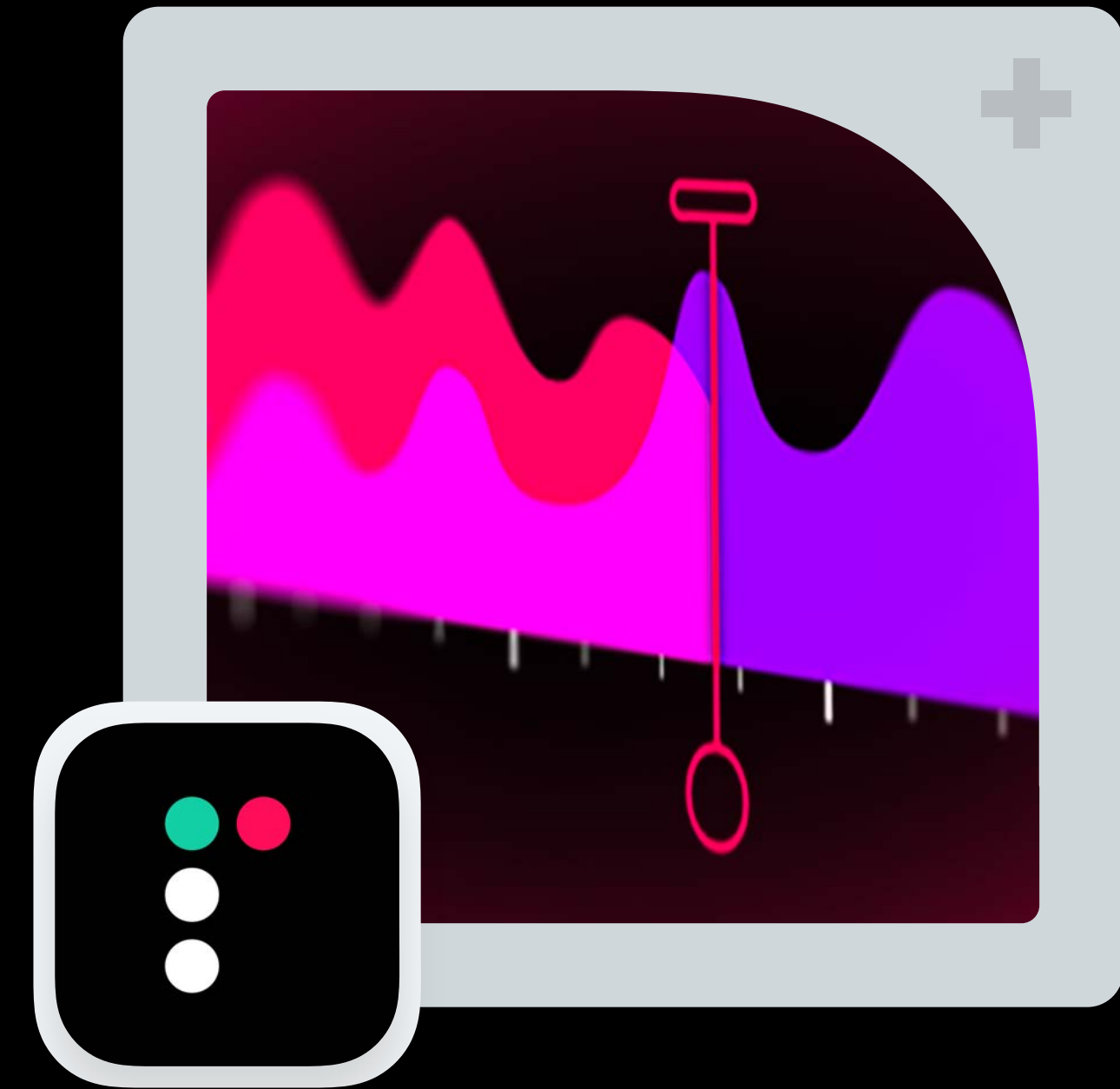
Types of In-App Purchases

Consumable products

Non-consumable products

Non-renewing subscriptions

Auto-renewable subscriptions



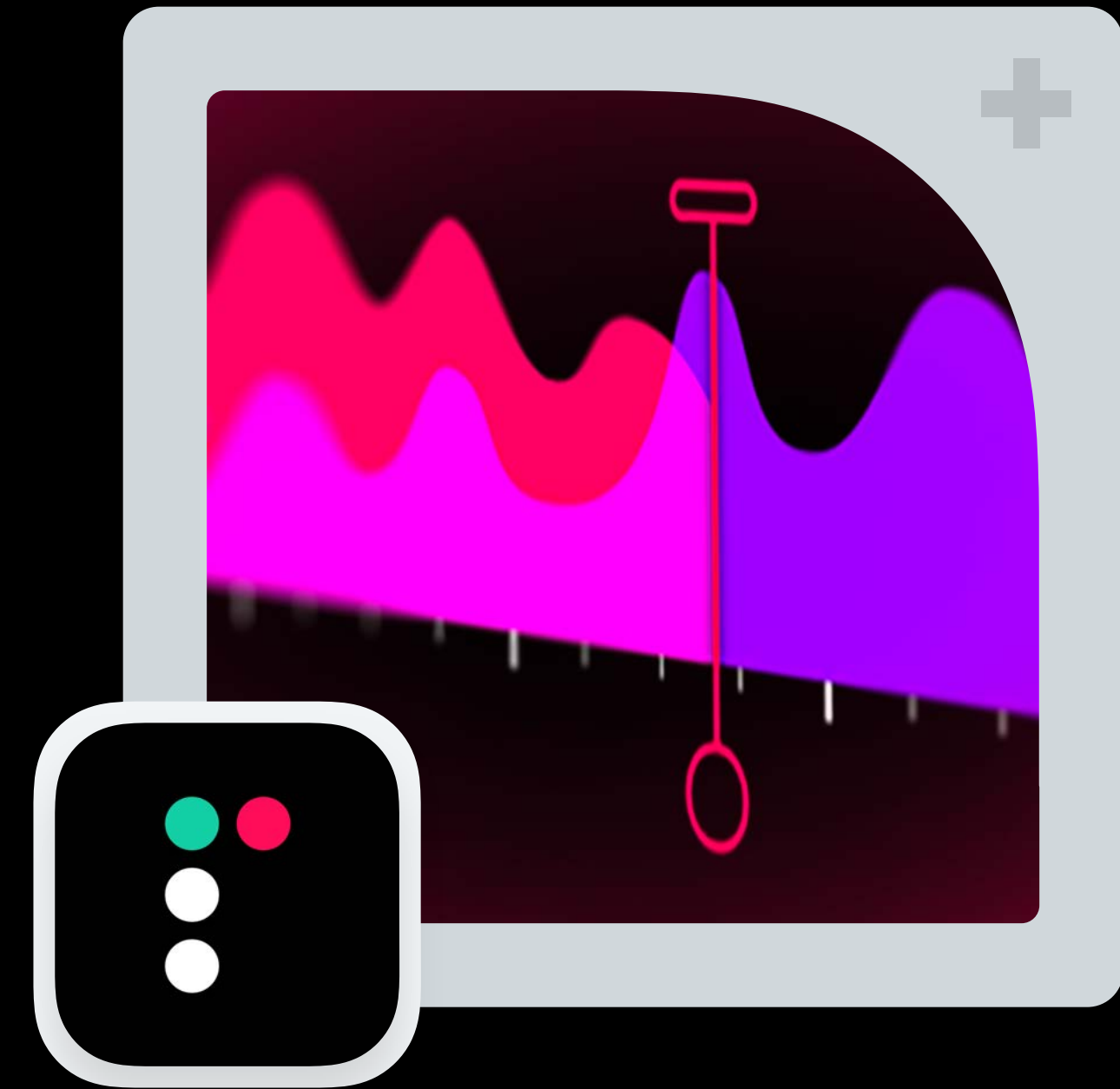
Types of In-App Purchases

Consumable products

Non-consumable products

Non-renewing subscriptions

Auto-renewable subscriptions



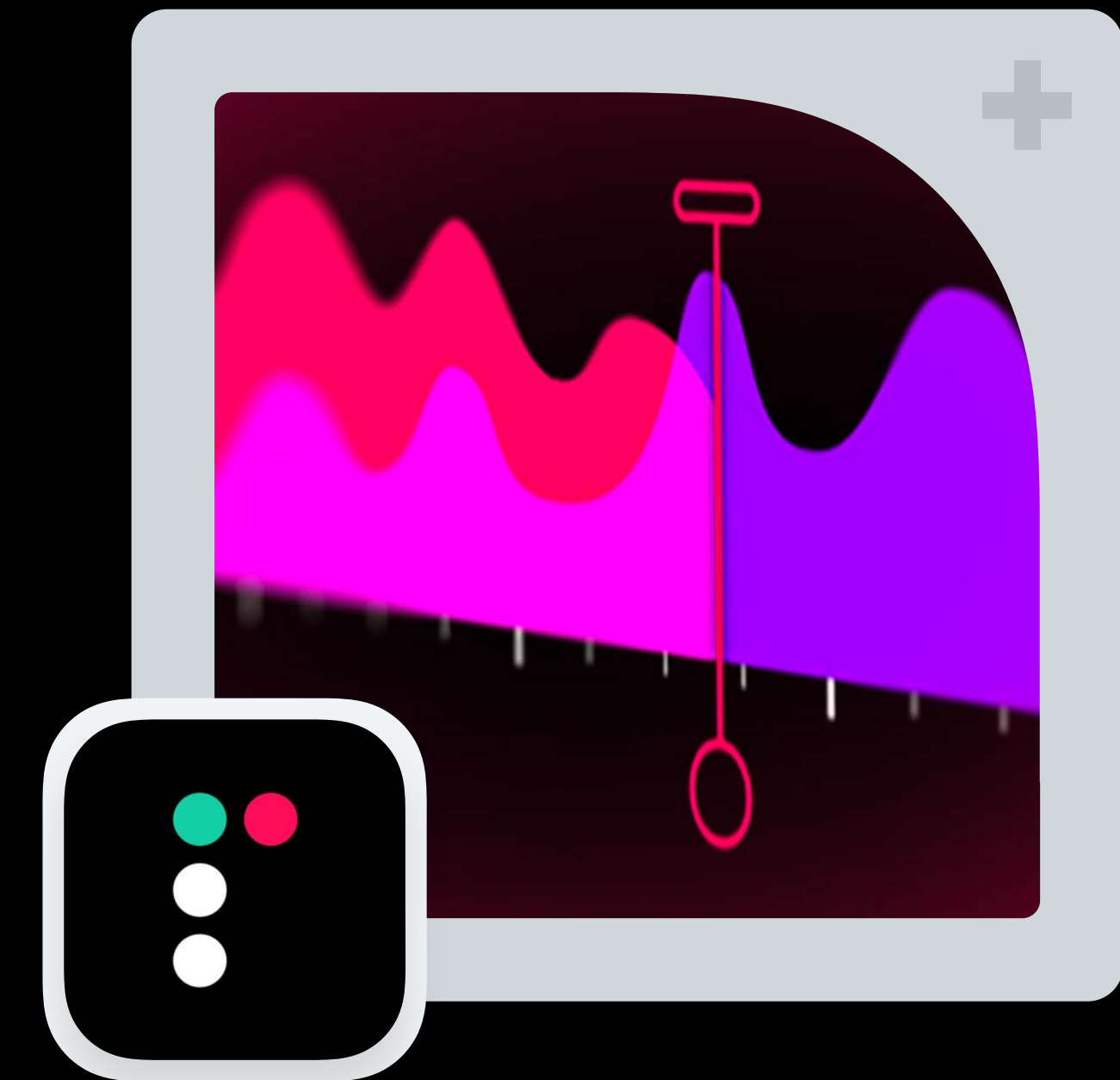
Types of In-App Purchases

Consumable products

Non-consumable products

Non-renewing subscriptions

Auto-renewable subscriptions



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



In-App Purchase Process



Load In-App Identifiers

Options for storing the list of product identifiers

Load In-App Identifiers

Options for storing the list of product identifiers

After setting up product identifiers in iTunes Connect

Load In-App Identifiers

Options for storing the list of product identifiers

After setting up product identifiers in iTunes Connect

Baked into your app

```
let identifiers = ["com.myCompany.myApp.product1",  
                  "com.myCompany.myApp.product2"]
```

Load In-App Identifiers

Options for storing the list of product identifiers

After setting up product identifiers in iTunes Connect

Baked into your app

```
let identifiers = ["com.myCompany.myApp.product1",  
                  "com.myCompany.myApp.product2"]
```

Or fetch from your server

```
let identifiers = remoteIdentifiers()
```

In-App Purchase Process




```
// Fetch Product Info
```

```
let request = SKProductsRequest(productIdentifiers: identifierSet)
```

```
request.delegate = self
```

```
request.start()
```

```
// Fetch Product Info
```

```
let request = SKProductsRequest(productIdentifiers: identifierSet)
```

```
request.delegate = self
```

```
request.start()
```

```
// Fetch Product Info
```

```
let request = SKProductsRequest(productIdentifiers: identifierSet)
```

```
request.delegate = self
```

```
request.start()
```

```
// Fetch Product Info
```

```
let request = SKProductsRequest(productIdentifiers: identifierSet)
```

```
request.delegate = self
```

```
request.start()
```

```
// Fetch Product Info
```

```
let request = SKProductsRequest(productIdentifiers: identifierSet)
```

```
request.delegate = self
```

```
request.start()
```

```
// Fetch Product Info

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        // Localized title and description
        product.localizedTitle
        product.localizedDescription
        // Price and locale
        product.price
        product.priceLocale
        // Content size and version (hosted)
        product.downloadContentLengths
        product.downloadContentVersion
    }
}
```

```
// Fetch Product Info

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        // Localized title and description
        product.localizedTitle
        product.localizedDescription
        // Price and locale
        product.price
        product.priceLocale
        // Content size and version (hosted)
        product.downloadContentLengths
        product.downloadContentVersion
    }
}
```

```
// Fetch Product Info

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        // Localized title and description
        product.localizedTitle
        product.localizedDescription
        // Price and locale
        product.price
        product.priceLocale
        // Content size and version (hosted)
        product.downloadContentLengths
        product.downloadContentVersion
    }
}
```



```
// Fetch Product Info

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        // Localized title and description
        product.localizedTitle
        product.localizedDescription
        // Price and locale
        product.price
        product.priceLocale
        // Content size and version (hosted)
        product.downloadContentLengths
        product.downloadContentVersion
    }
}
```

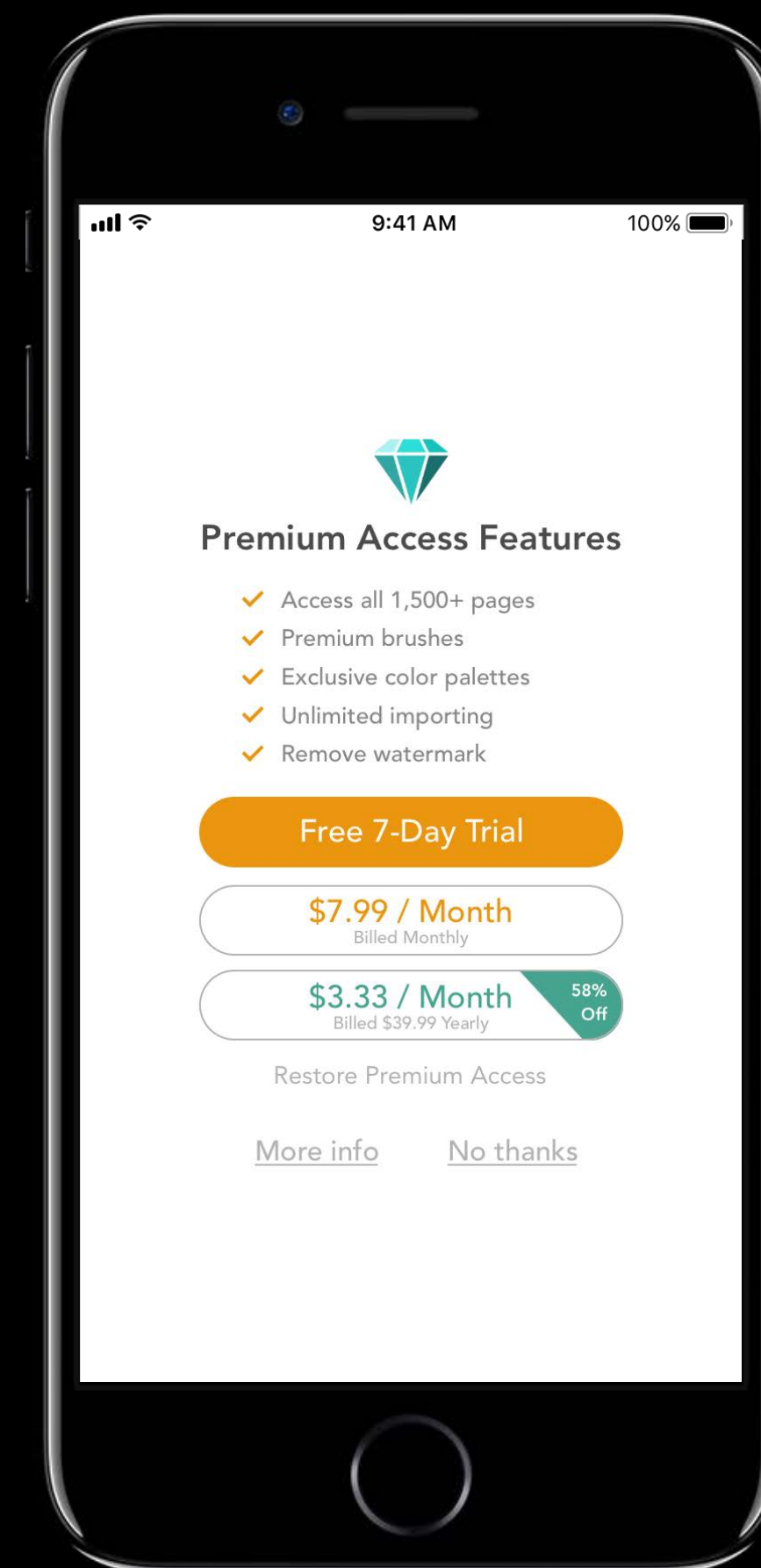
```
// Fetch Product Info

func productsRequest(_ request: SKProductsRequest, didReceive response: SKProductsResponse) {
    for product in response.products {
        // Localized title and description
        product.localizedTitle
        product.localizedDescription
        // Price and locale
        product.price
        product.priceLocale
        // Content size and version (hosted)
        product.downloadContentLengths
        product.downloadContentVersion
    }
}
```

In-App Purchase Process

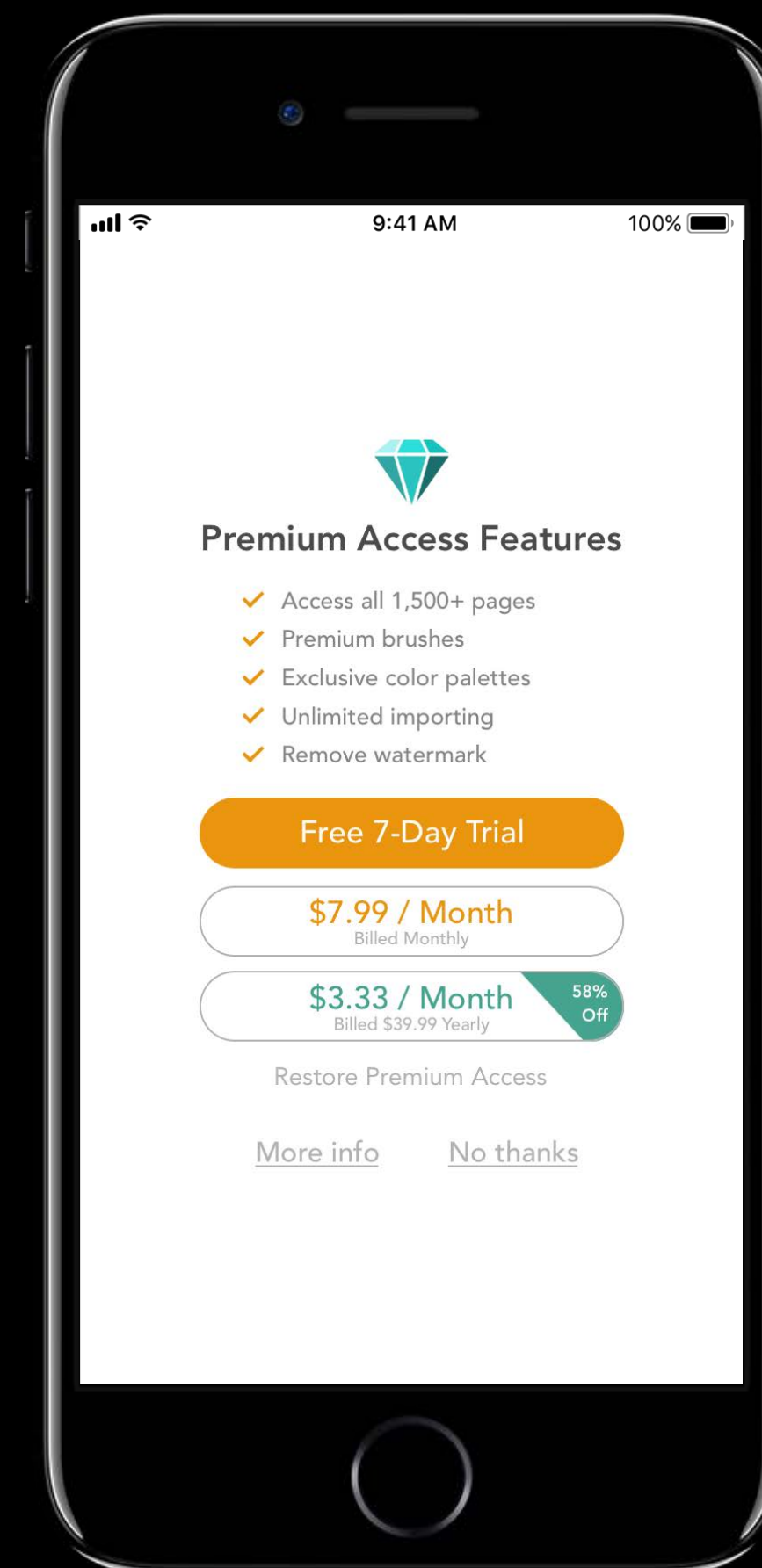


In-App Purchase UI



In-App Purchase UI

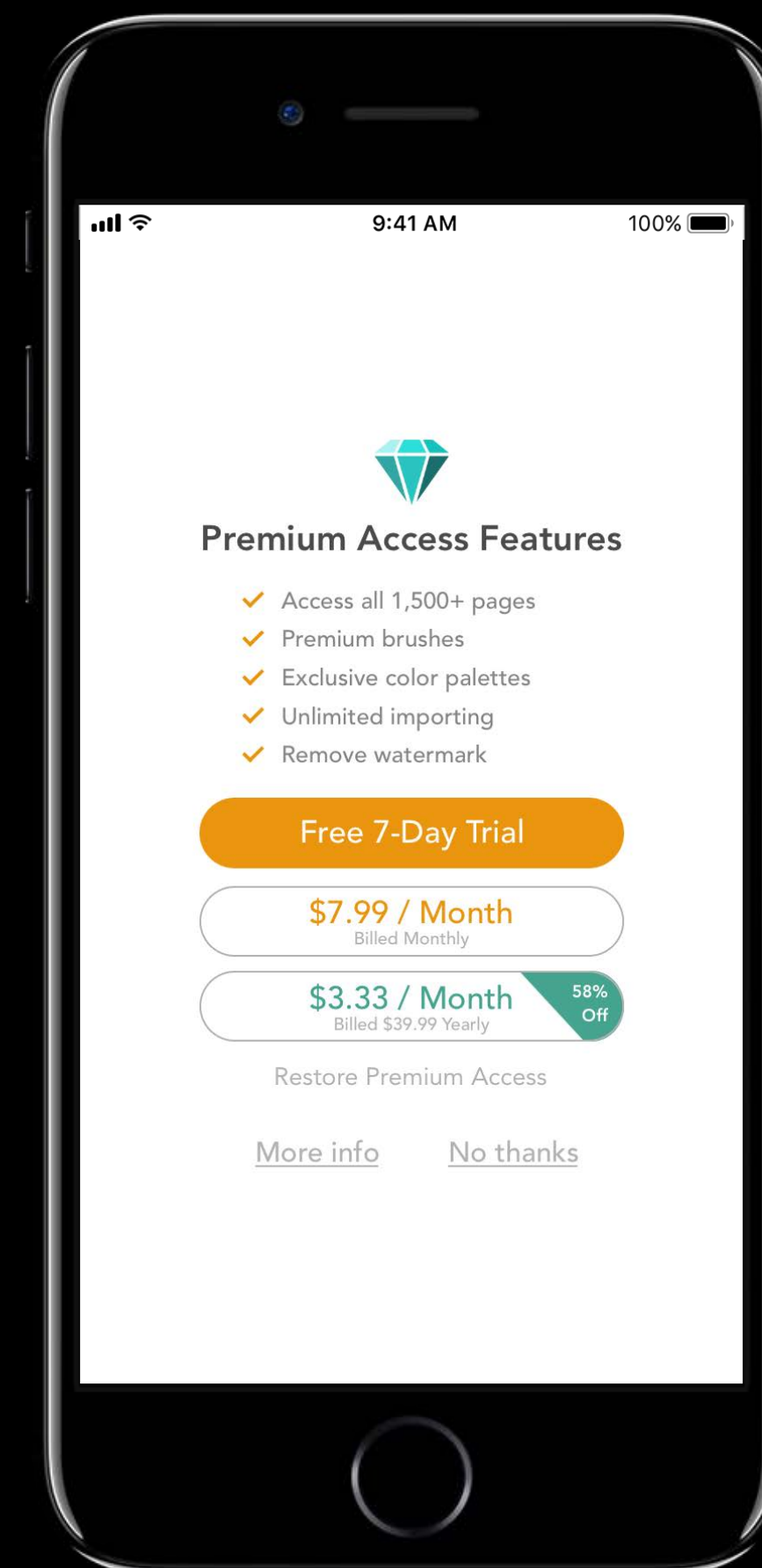
Up to the application



In-App Purchase UI

Up to the application

Can have a large effect on sales

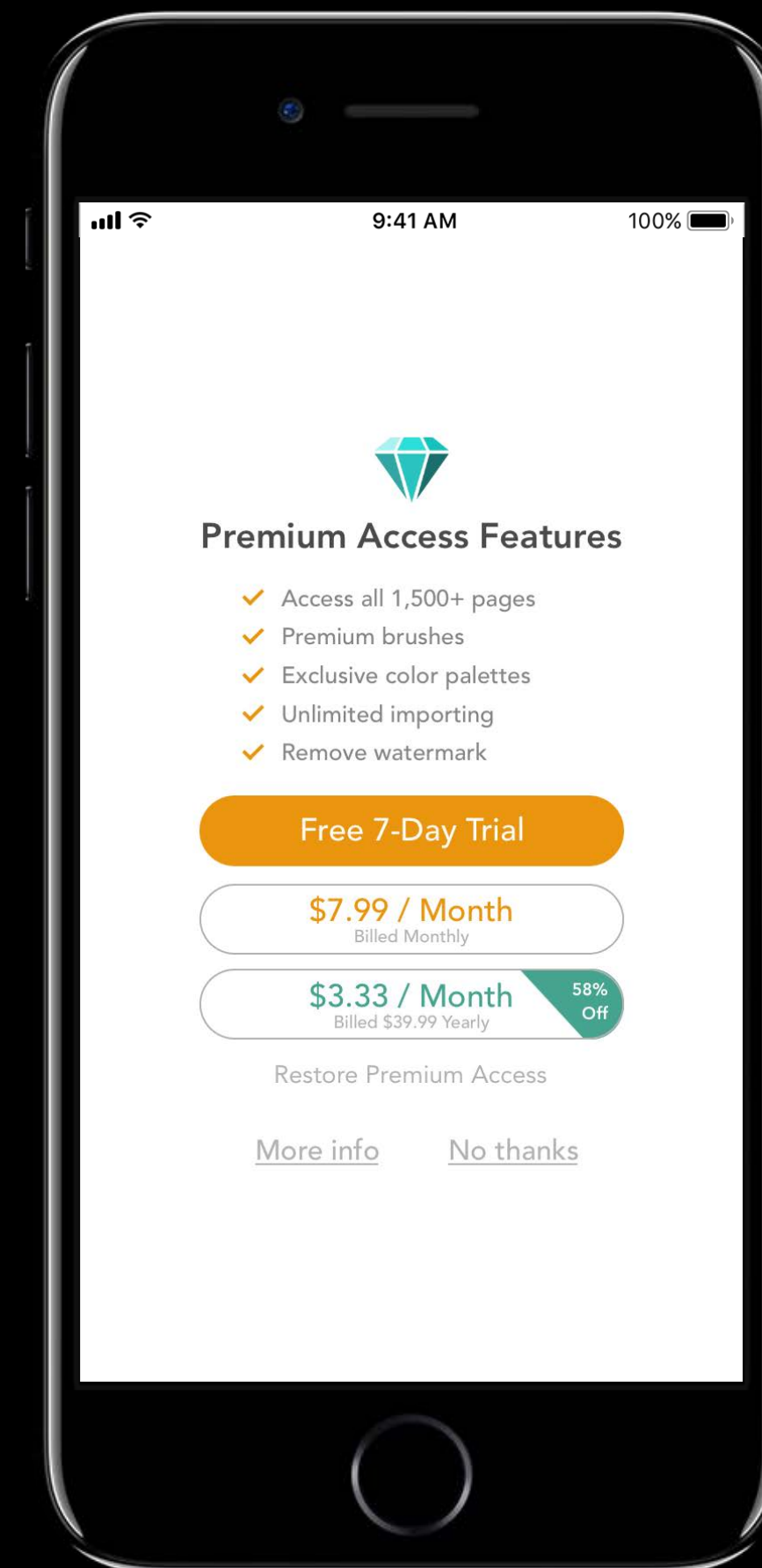


In-App Purchase UI

Up to the application

Can have a large effect on sales

<https://developer.apple.com/in-app-purchase/>



In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```


In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

In-App Purchase UI

Formatting the product price

```
let formatter = NumberFormatter()
formatter.numberStyle = .currency
formatter.locale = product.priceLocale // Not the device locale!
let formattedString = formatter.string(from: product.price)
```

Do not perform currency conversion

In-App Purchase Process



```
// Requesting a Payment
```

```
let payment = SKPayment(product: product)
```

```
SKPaymentQueue.default().add(payment)
```



```
// Requesting a Payment
```

```
let payment = SKPayment(product: product)
```

```
SKPaymentQueue.default().add(payment)
```

```
// Requesting a Payment
```

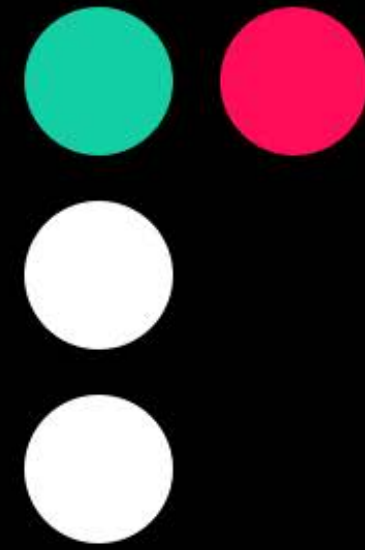
```
let payment = SKPayment(product: product)
```

```
SKPaymentQueue.default().add(payment)
```

```
// Requesting a Payment
```

```
let payment = SKPayment(product: product)
```

```
SKPaymentQueue.default().add(payment)
```



Pacemaker®

App Store

Cancel



REVERB
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE

\$1.99



Buy with Touch ID

App Store

Cancel



REVERB
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE

\$1.99



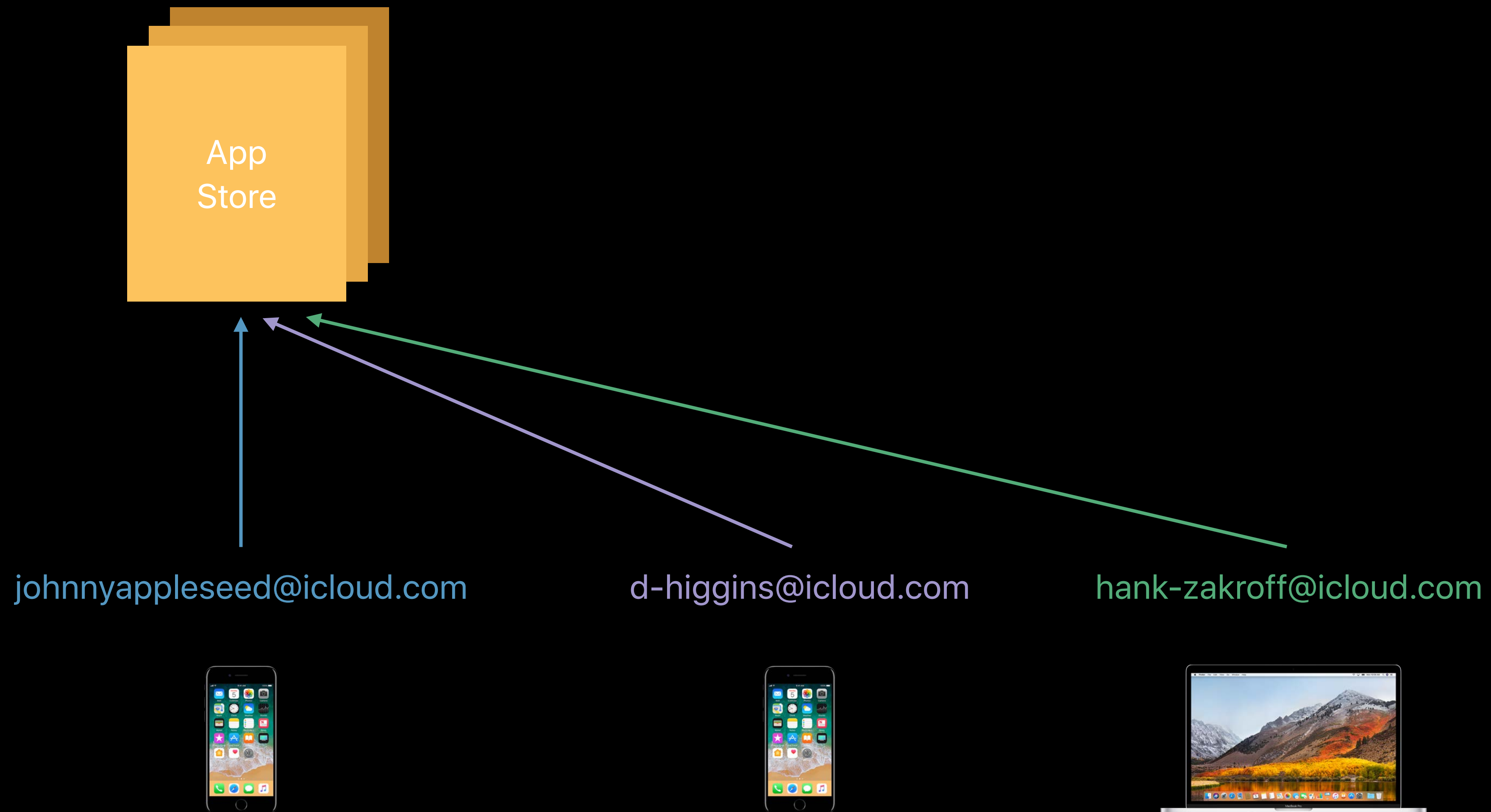
Done

Detecting Irregular Activity

Suspicious activity during payment process

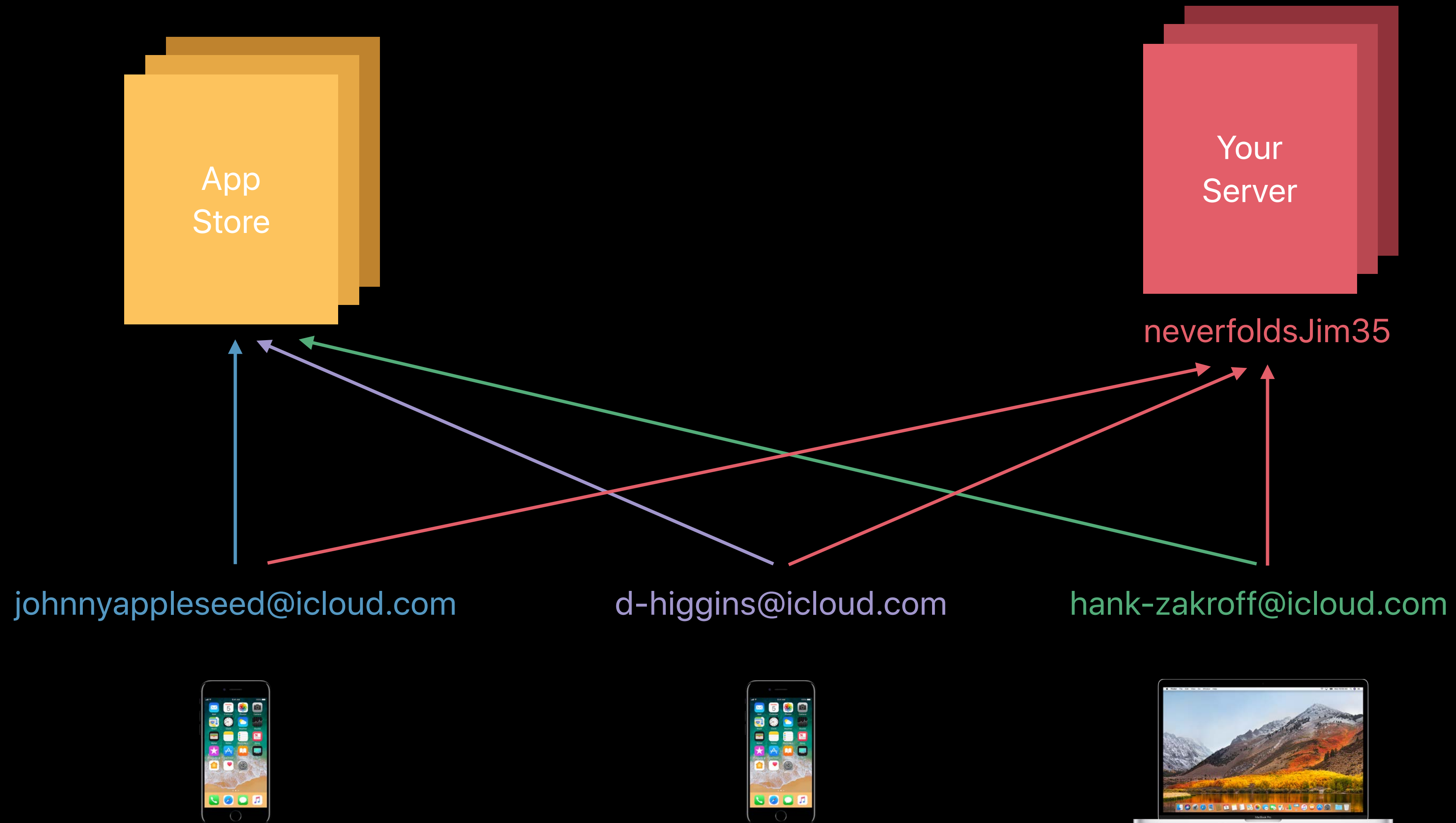
Detecting Irregular Activity

Suspicious activity during payment process



Detecting Irregular Activity

Suspicious activity during payment process



Detecting Irregular Activity

Provide an account identifier

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for your user's account

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID
- Don't provide the actual account name

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID
- Don't provide the actual account name
- Don't provide the password

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID
- Don't provide the actual account name
- Don't provide the password
- We suggest using a hash of the account name

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID
- Don't provide the actual account name
- Don't provide the password
- We suggest using a hash of the account name

```
let payment = SKPayment(product: product)
payment.applicationUsername = hash(yourCustomerAccountName)
SKPaymentQueue.default().add(payment)
```

Detecting Irregular Activity

Provide an account identifier

For applications with their own account management

Provide an **opaque identifier** for **your user's account**

- Don't send us the user's Apple ID
- Don't provide the actual account name
- Don't provide the password
- We suggest using a hash of the account name

```
let payment = SKPayment(product: product)
payment.applicationUsername = hash(yourCustomerAccountName)
SKPaymentQueue.default().add(payment)
```

In-App Purchase Process



```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }
}
```

```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }
}
```

```
// Start Observing the Payment Queue

import UIKit
import StoreKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, SKPaymentTransactionObserver {

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {
        SKPaymentQueue.default().add(self)
        return true
    }
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        }
    }
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        }
    }
}
```



```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        }
    }
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        }
    }
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        }
    }
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        case .deferred:
            // Allow the user to continue to use the app
            // It may be some time before the transaction is updated
            // Do not get stuck in a modal "Purchasing..." state!
        }
    }
}
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        case .deferred:
            // Allow the user to continue to use the app
            // It may be some time before the transaction is updated
            // Do not get stuck in a modal "Purchasing..." state!
        }
    }
}
}
```

```
// Handle SKPaymentQueueObserver Events

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, updatedTransactions transactions:
    [SKPaymentTransaction]) {
    for transaction in transactions {
        switch transaction.transactionState {
        case .purchased:
            // Validate the purchase
        case .deferred:
            // Allow the user to continue to use the app
            // It may be some time before the transaction is updated
            // Do not get stuck in a modal "Purchasing..." state!
        }
    }
}
}
```

Testing Deferred Transactions

Testing Deferred Transactions

Create a mutable payment

Testing Deferred Transactions

Create a mutable payment

Set the `simulatesAskToBuyInSandbox` flag

Testing Deferred Transactions

Create a mutable payment

Set the `simulatesAskToBuyInSandbox` flag

```
let payment = SKMutablePayment(product: product)
payment.simulatesAskToBuyInSandbox = true
SKPaymentQueue.default().add(payment)
```

Handling Errors

Handling Errors

Not all errors are equal

Handling Errors

Not all errors are equal

Check the error code

Handling Errors

Not all errors are equal

Check the error code

- Don't show an error alert unless necessary

Handling Errors

Not all errors are equal

Check the error code

- Don't show an error alert unless necessary
- User canceling a payment will result in an error

Handling Errors

Not all errors are equal

Check the error code

- Don't show an error alert unless necessary
- User canceling a payment will result in an error

Let StoreKit handle the transaction flow as much as possible

Handling Errors

Not all errors are equal

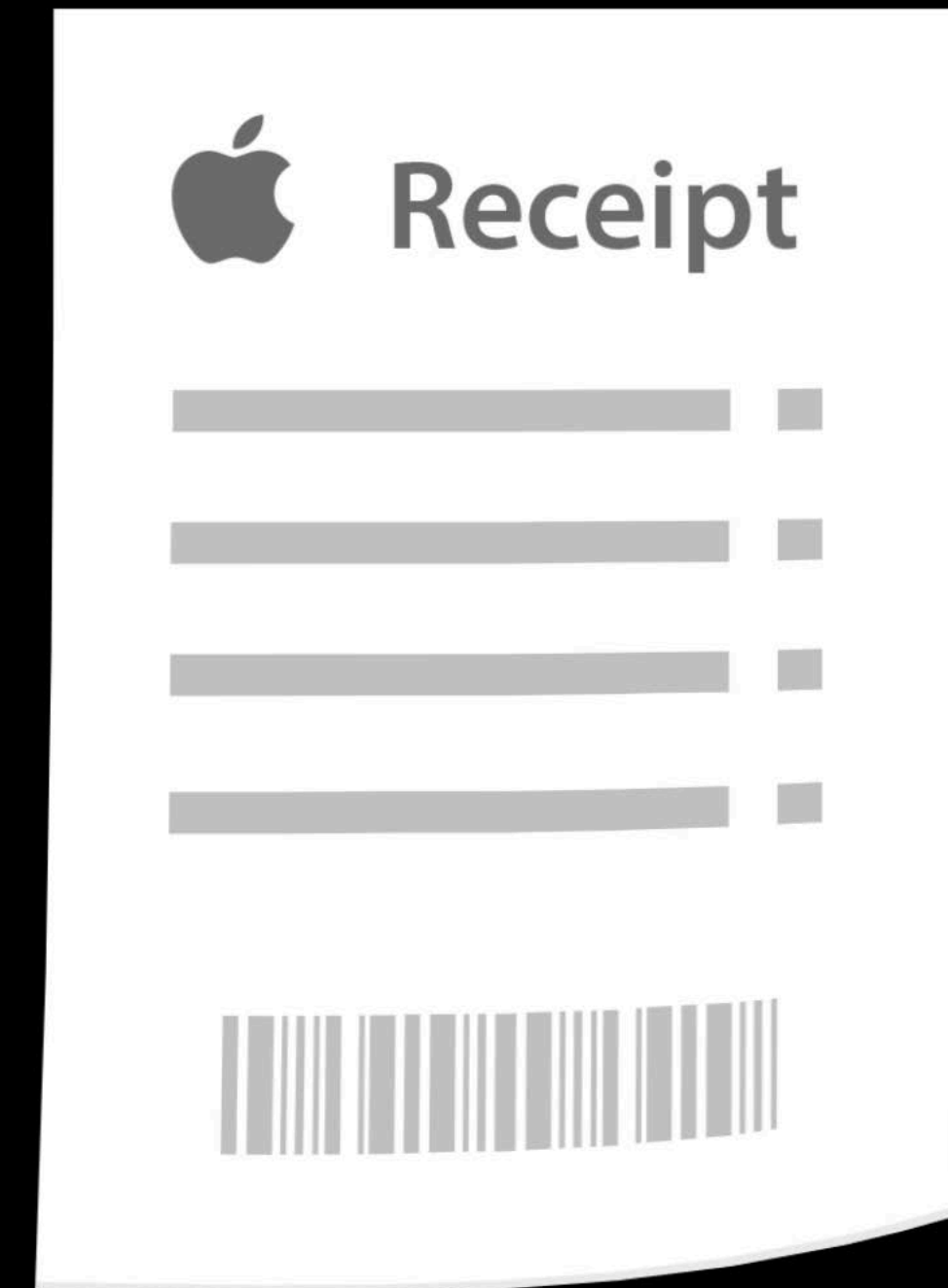
Check the error code

- Don't show an error alert unless necessary
- User canceling a payment will result in an error

Let StoreKit handle the transaction flow as much as possible

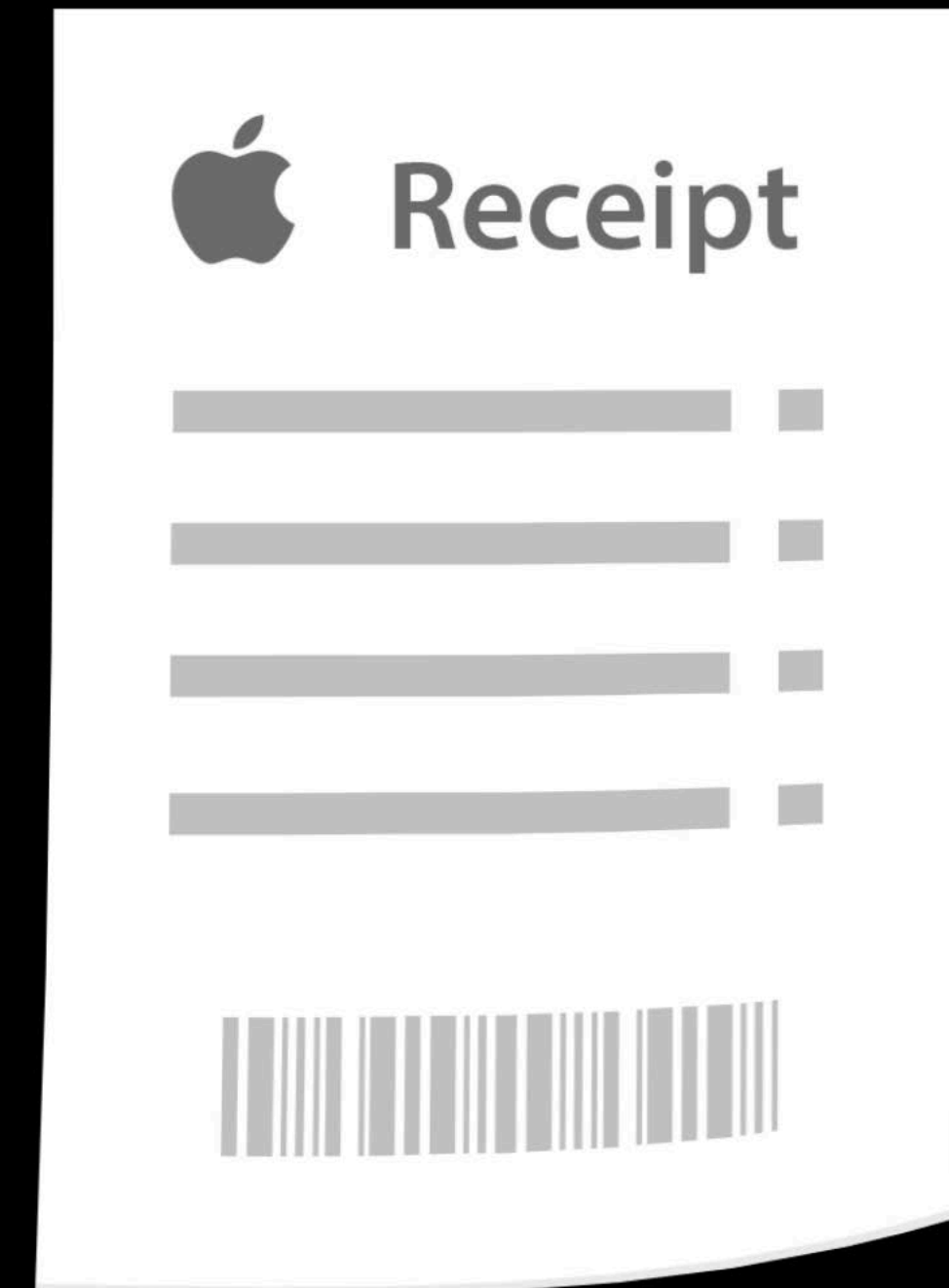
- Including asking for confirmation for purchase

The Receipt



The Receipt

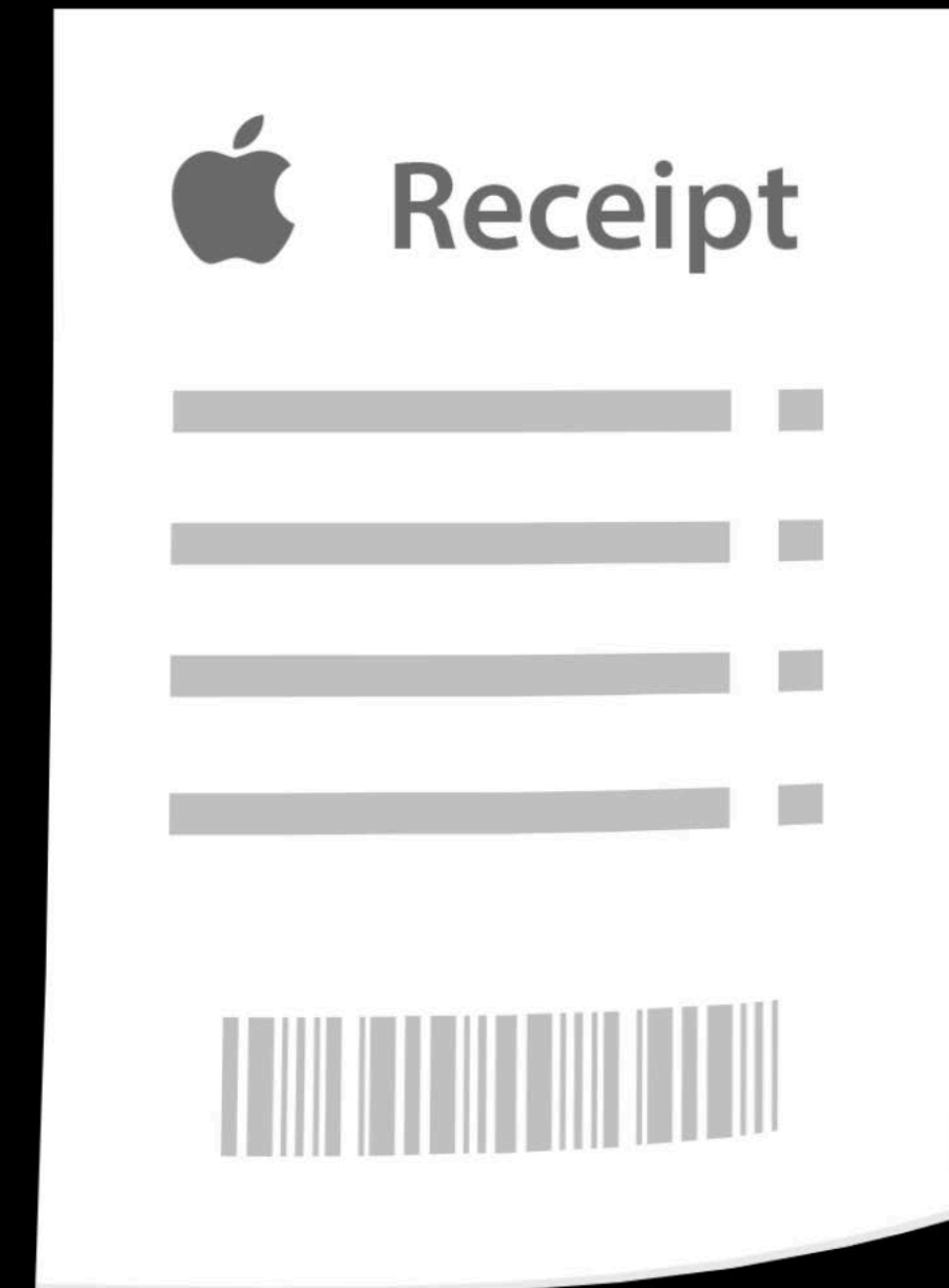
Trusted record of app and in-app purchases



The Receipt

Trusted record of app and in-app purchases

Stored on device

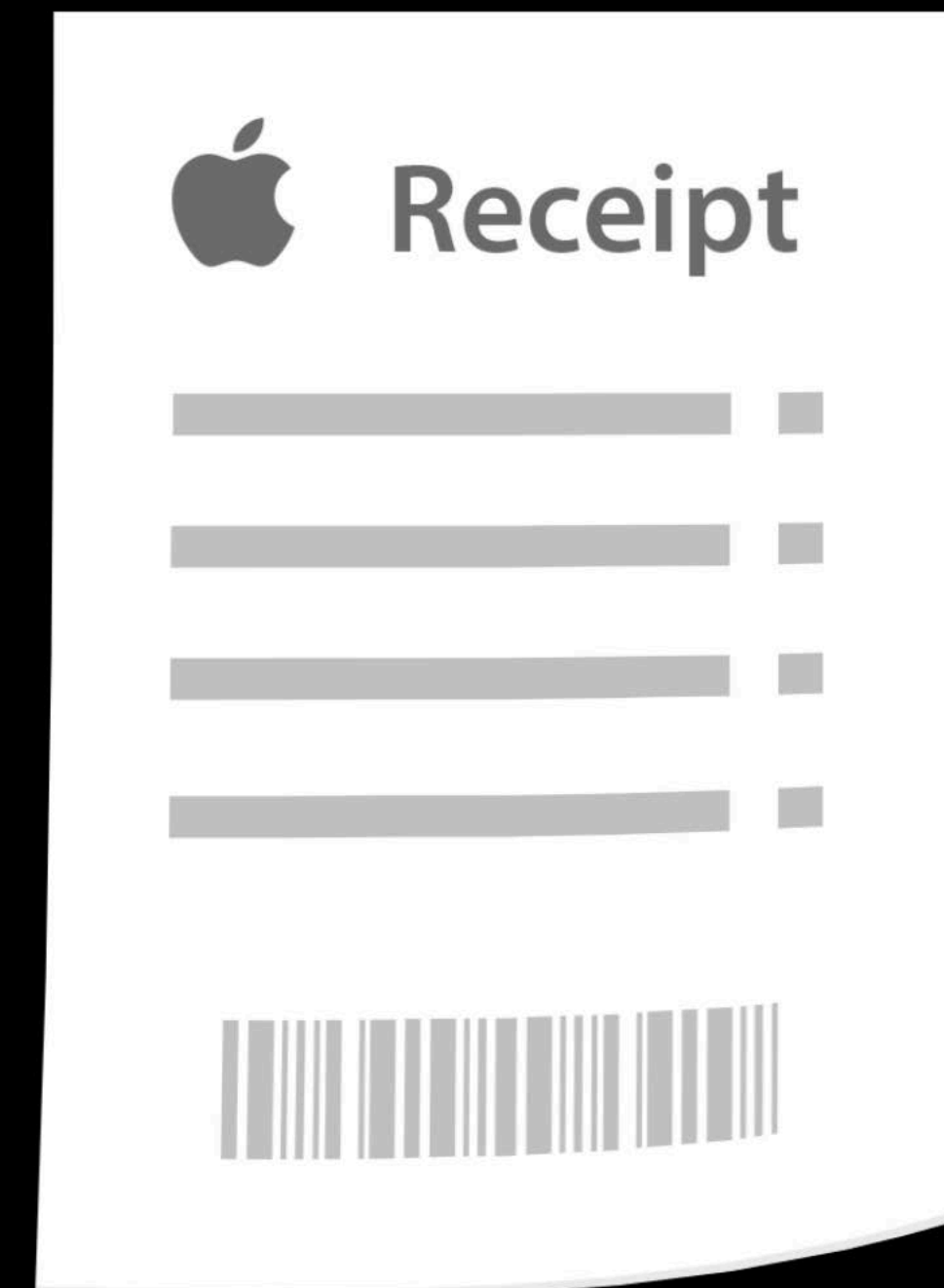


The Receipt

Trusted record of app and in-app purchases

Stored on device

Issued by the App Store



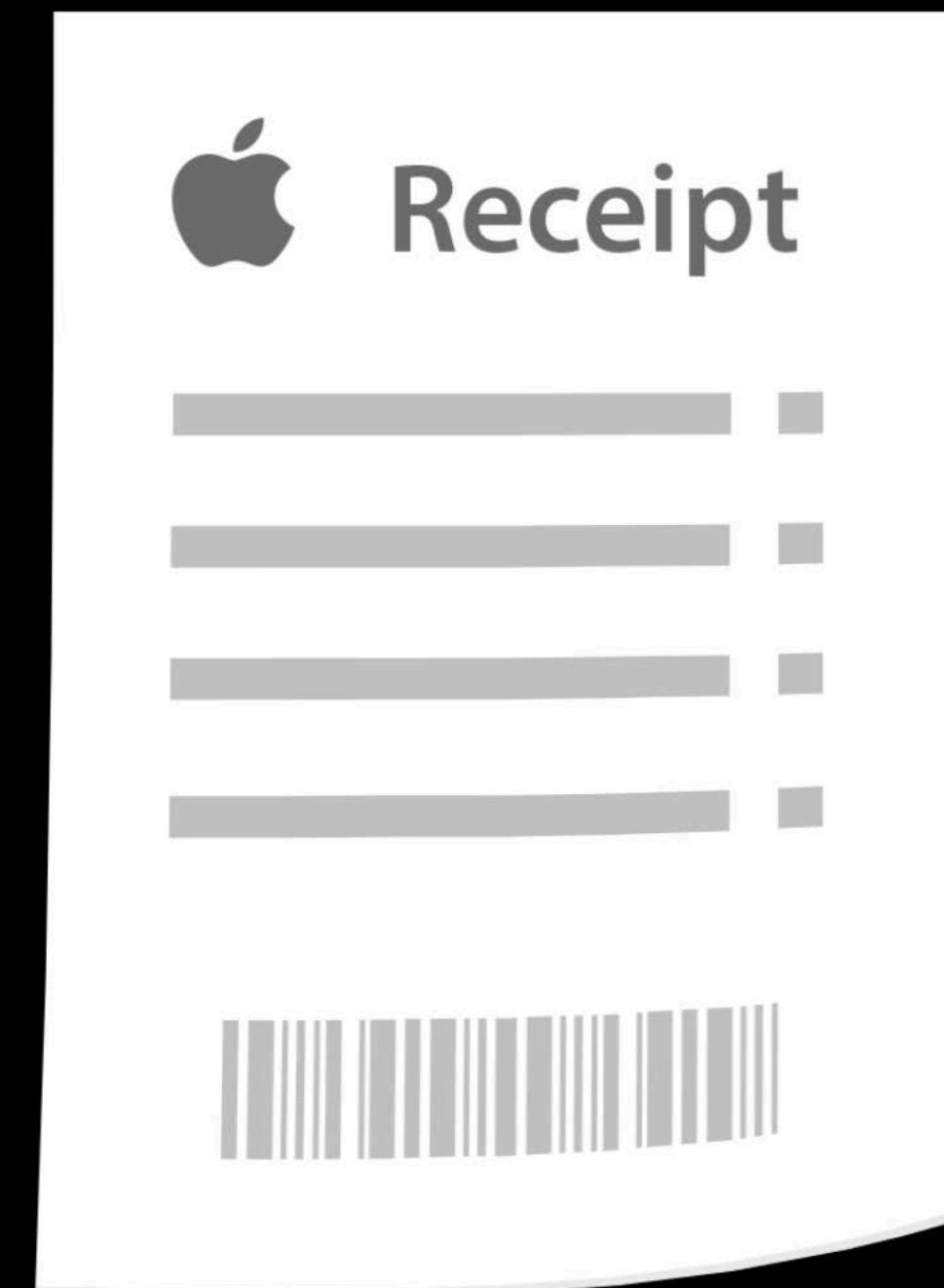
The Receipt

Trusted record of app and in-app purchases

Stored on device

Issued by the App Store

Signed and verifiable



The Receipt

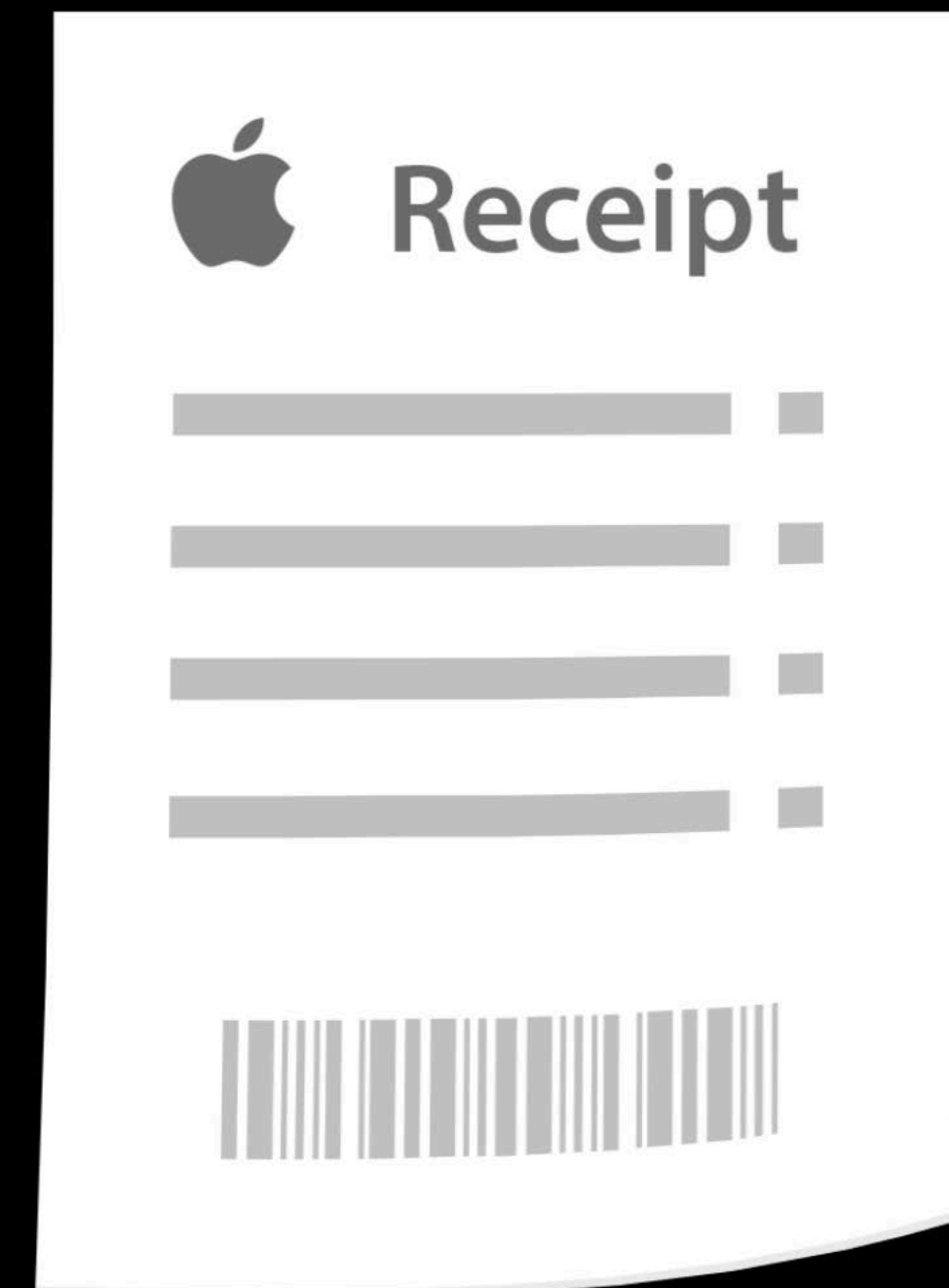
Trusted record of app and in-app purchases

Stored on device

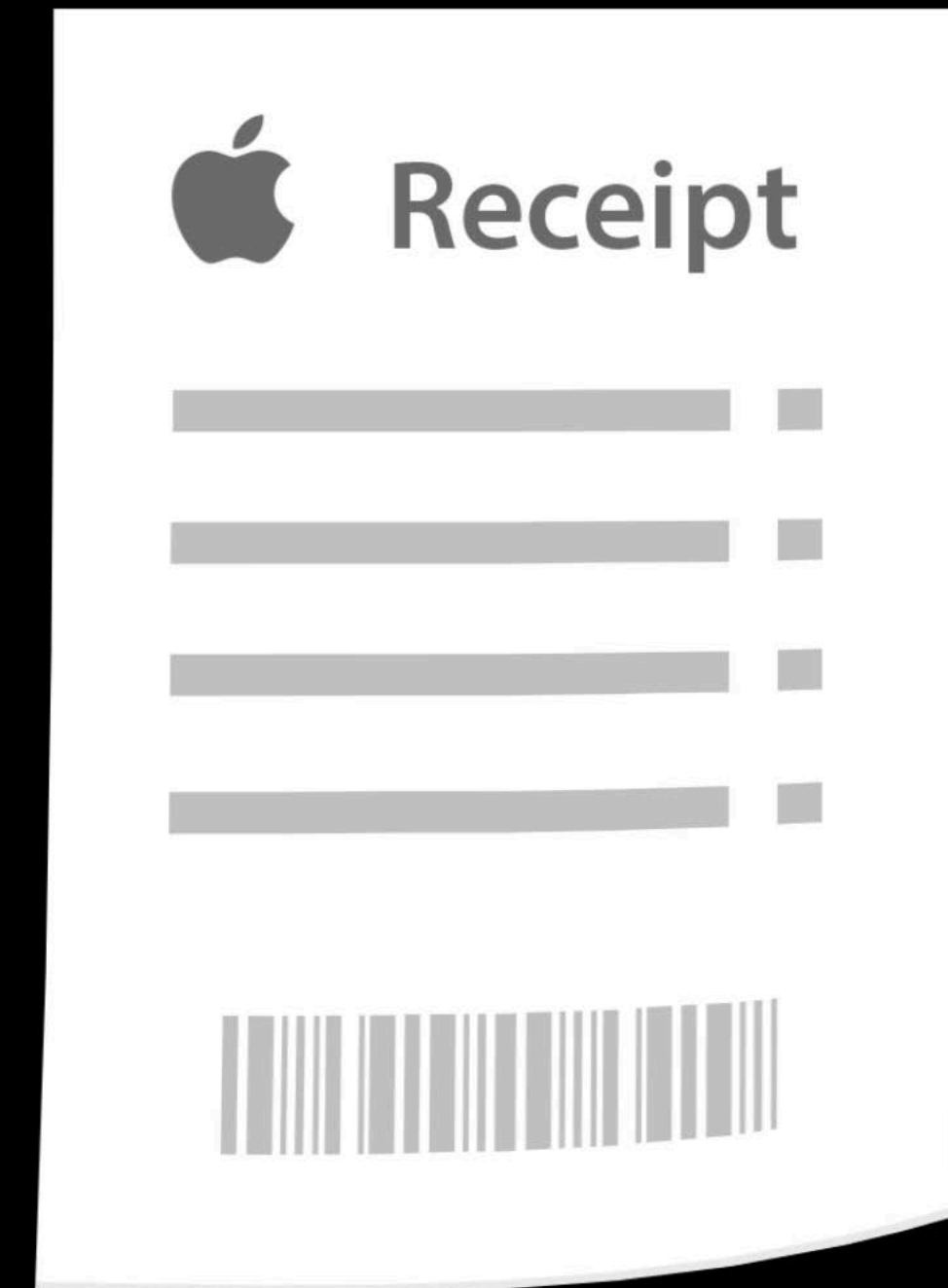
Issued by the App Store

Signed and verifiable

For your app, on that device only



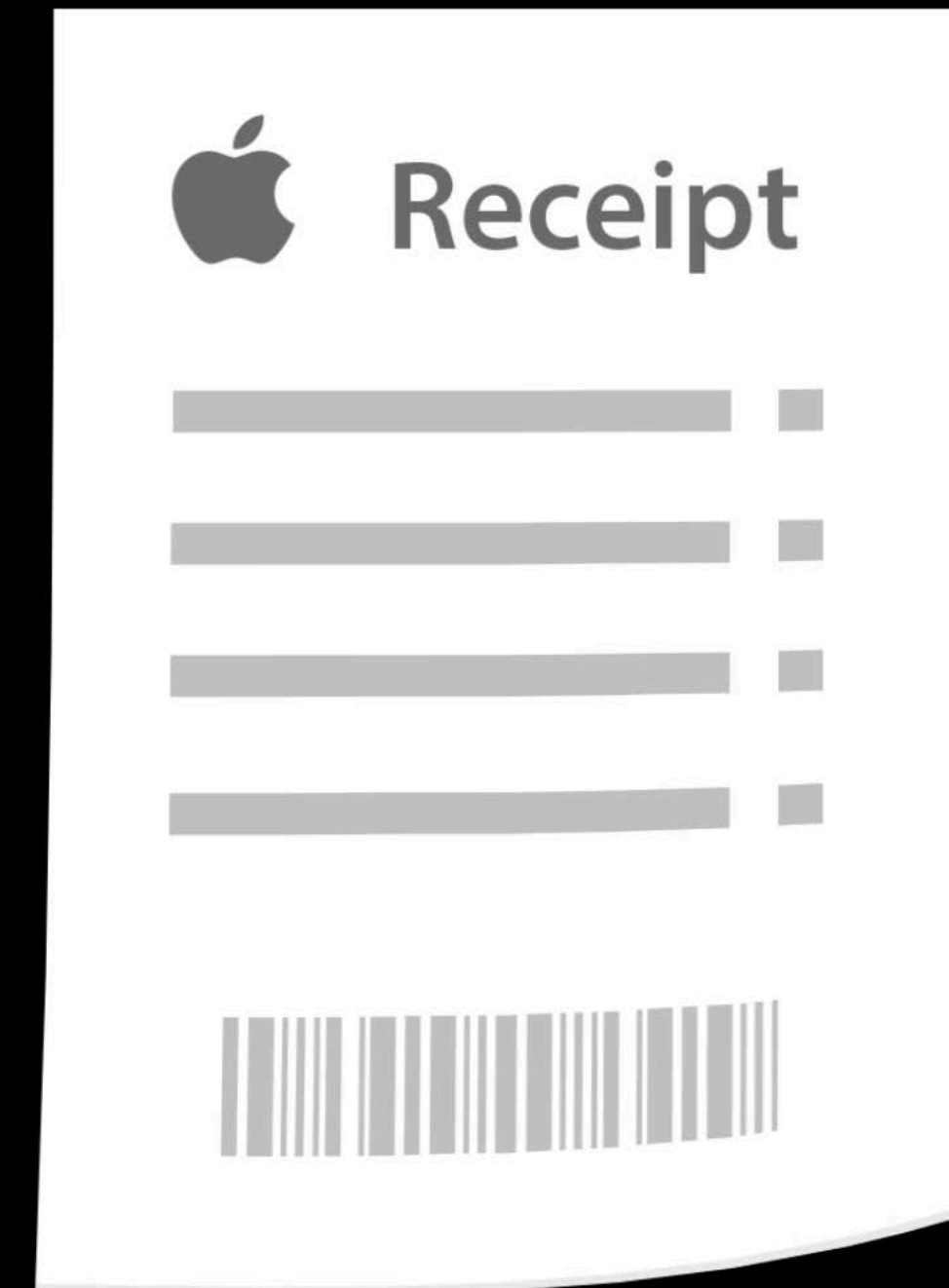
Receipt Validation



Receipt Validation

On-device validation

- Unlock features and content within the app



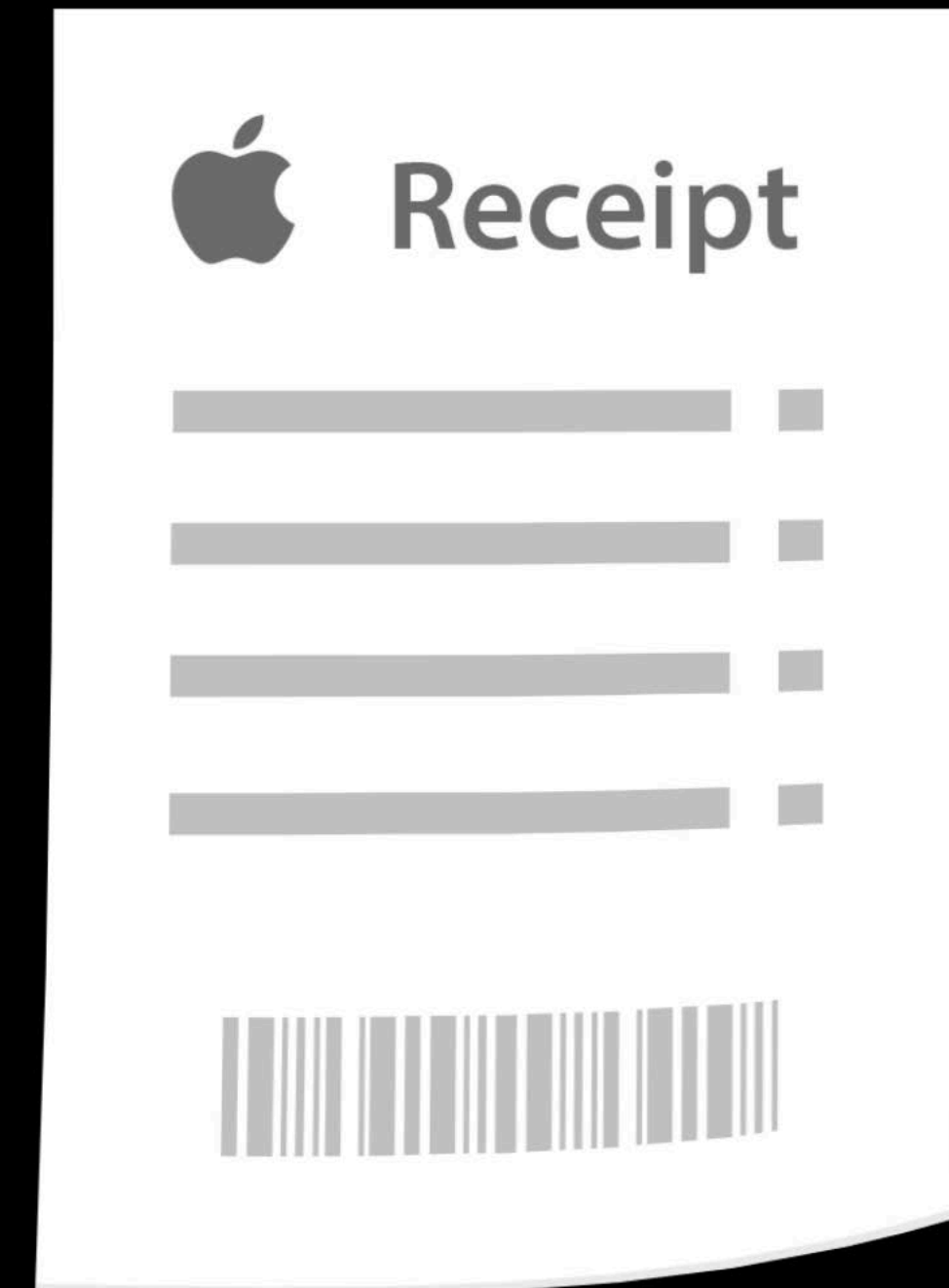
Receipt Validation

On-device validation

- Unlock features and content within the app

Server-to-server validation

- Restrict access to downloadable content
- Used often for subscriptions



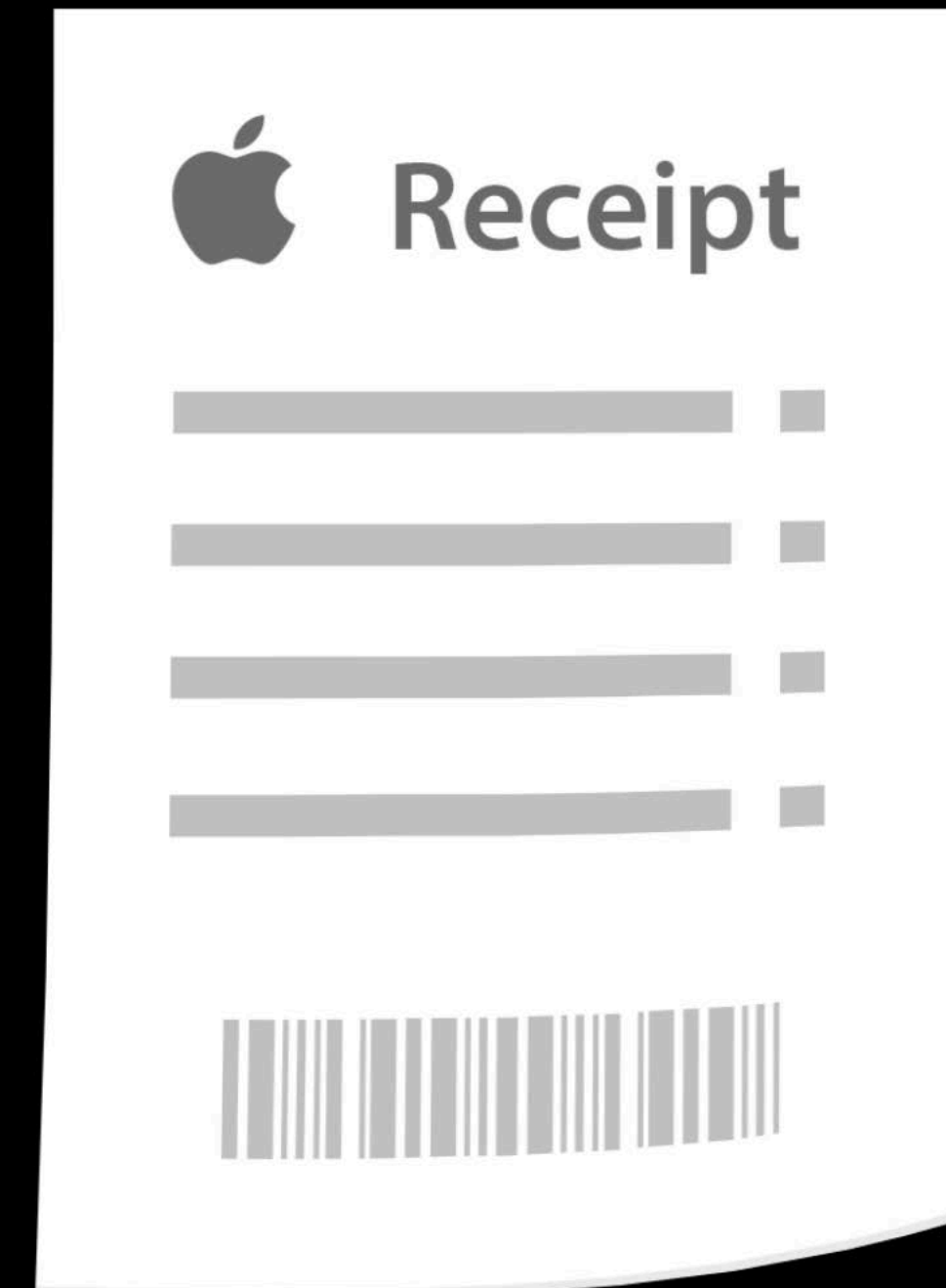
Receipt Validation

On-device validation

- Unlock features and content within the app

Server-to-server validation

- Restrict access to downloadable content
- Used often for subscriptions



In-App Purchase Process



Unlock Content

Unlock Content

Unlock functionality in your app

Unlock Content

Unlock functionality in your app

Download additional content

Downloading Content

Downloading Content

Apple-hosted content

Downloading Content

Apple-hosted content

- On-demand resources

Downloading Content

Apple-hosted content

- On-demand resources
- Hosted in-app purchase content

Downloading Content

Apple-hosted content

- On-demand resources
- Hosted in-app purchase content

Self-hosted content

Downloading Content

Apple-hosted content

- On-demand resources
- Hosted in-app purchase content

Self-hosted content

- Use background downloads with `NSURLSession`

In-App Purchase Process



Finish the Transaction

Finish the Transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Finish the Transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes **all** auto-renewable subscription transactions

Finish the Transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes **all** auto-renewable subscription transactions

Otherwise, the payment will stay in the queue

Finish the Transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes **all** auto-renewable subscription transactions

Otherwise, the payment will stay in the queue

Subscription billing retry depends on up-to-date information about transaction

Finish the Transaction

Finish all transactions once content is unlocked

- If downloading hosted content, wait until after the download completes

Includes **all** auto-renewable subscription transactions

Otherwise, the payment will stay in the queue

Subscription billing retry depends on up-to-date information about transaction

```
SKPaymentQueue.default().finishTransaction(transaction)
```

In-App Purchase Process



App Review

App Review

You must have a Restore button

App Review

You must have a Restore button

Restore and Purchase must be separate buttons

App Review

You must have a Restore button

Restore and Purchase must be separate buttons

Not just as a “backup” tool

App Review

You must have a Restore button

Restore and Purchase must be separate buttons

Not just as a “backup” tool

- Users with multiple devices

Restore Completed Transactions

Only restores transactions for

- Non-consumables
- Auto-renewable subscriptions

For consumables and non-renewing subscriptions

- You must persist the state!

Restore Completed Transactions

Restore Completed Transactions

```
SKPaymentQueue.default().restoreCompletedTransactions()
```

Restore Completed Transactions

```
SKPaymentQueue.default().restoreCompletedTransactions()
```

Observe the queue

```
// Additional callbacks in SKPaymentTransactionObserver  
func paymentQueueRestoreCompletedTransactionsFinished(_ queue: SKPaymentQueue) {}  
func paymentQueue(_ queue: SKPaymentQueue,  
    restoreCompletedTransactionsFailedWithError error: NSError) {}
```

Inspect the receipt and unlock content and features accordingly

Implementing In-App Purchases

Implementing In-App Purchases

Always observe the Payment Queue

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Display pricing using the product's price locale

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Display pricing using the product's price locale

Use the receipt to validate your purchases

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Display pricing using the product's price locale

Use the receipt to validate your purchases

Make the content available

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Display pricing using the product's price locale

Use the receipt to validate your purchases

Make the content available

Finish the transaction

Implementing In-App Purchases

Always observe the Payment Queue

Fetch localized product information from the App Store

Display pricing using the product's price locale

Use the receipt to validate your purchases

Make the content available

Finish the transaction

Allow the user to restore completed transactions

Promoting In-App Purchases

Ross LeBeau, App Store Engineer

Promoting In-App Purchases

Promoting In-App Purchases

Discoverable

Promoting In-App Purchases

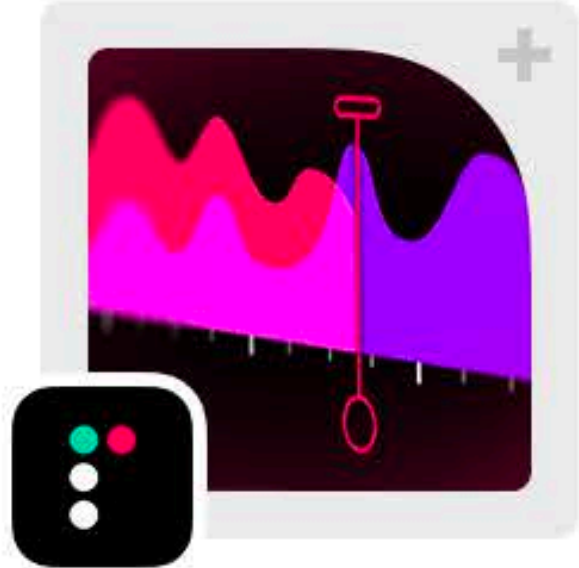
Discoverable

- App page

Eat better, sleep more, and track your progress over time with HealthKit.

Make Your Own Summer Jam

Add-ons for music creation



Beatskip
for [Pacemaker](#)


Skip beats to create jumps in your tracks.

\$1.99

Yes, You Can Cook That!

Recipies and ingredients at your doorstep

[See All](#)

 Blue Apron

Promoting In-App Purchases


Discoverable

- App page
- Editorial features

Eat better, sleep more, and track your progress over time with HealthKit.

Make Your Own Summer Jam

Add-ons for music creation



Beatskip
for [Pacemaker](#)


Skip beats to create jumps in your tracks.

\$1.99

Yes, You Can Cook That!

Recipies and ingredients at your doorstep

[See All](#)

 Blue Apron

Promoting In-App Purchases

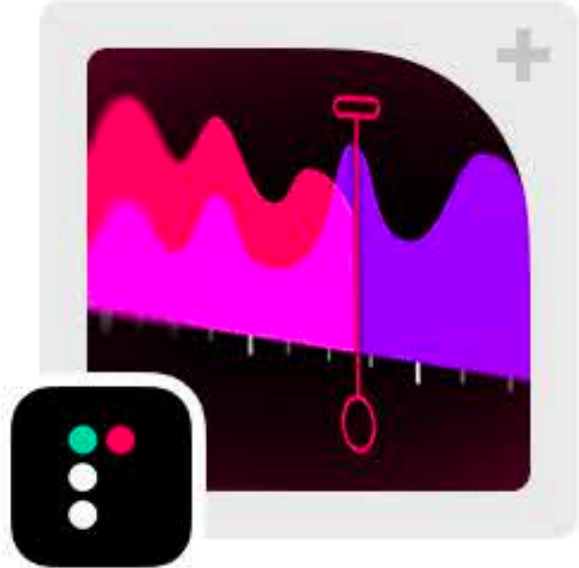
Discoverable

- App page
- Editorial features
- Search results

Eat better, sleep more, and track your progress over time with HealthKit.

Make Your Own Summer Jam

Add-ons for music creation



Beatskip
for [Pacemaker](#)


Skip beats to create jumps in your tracks.

\$1.99

Yes, You Can Cook That!

Recipies and ingredients at your doorstep

[See All](#)

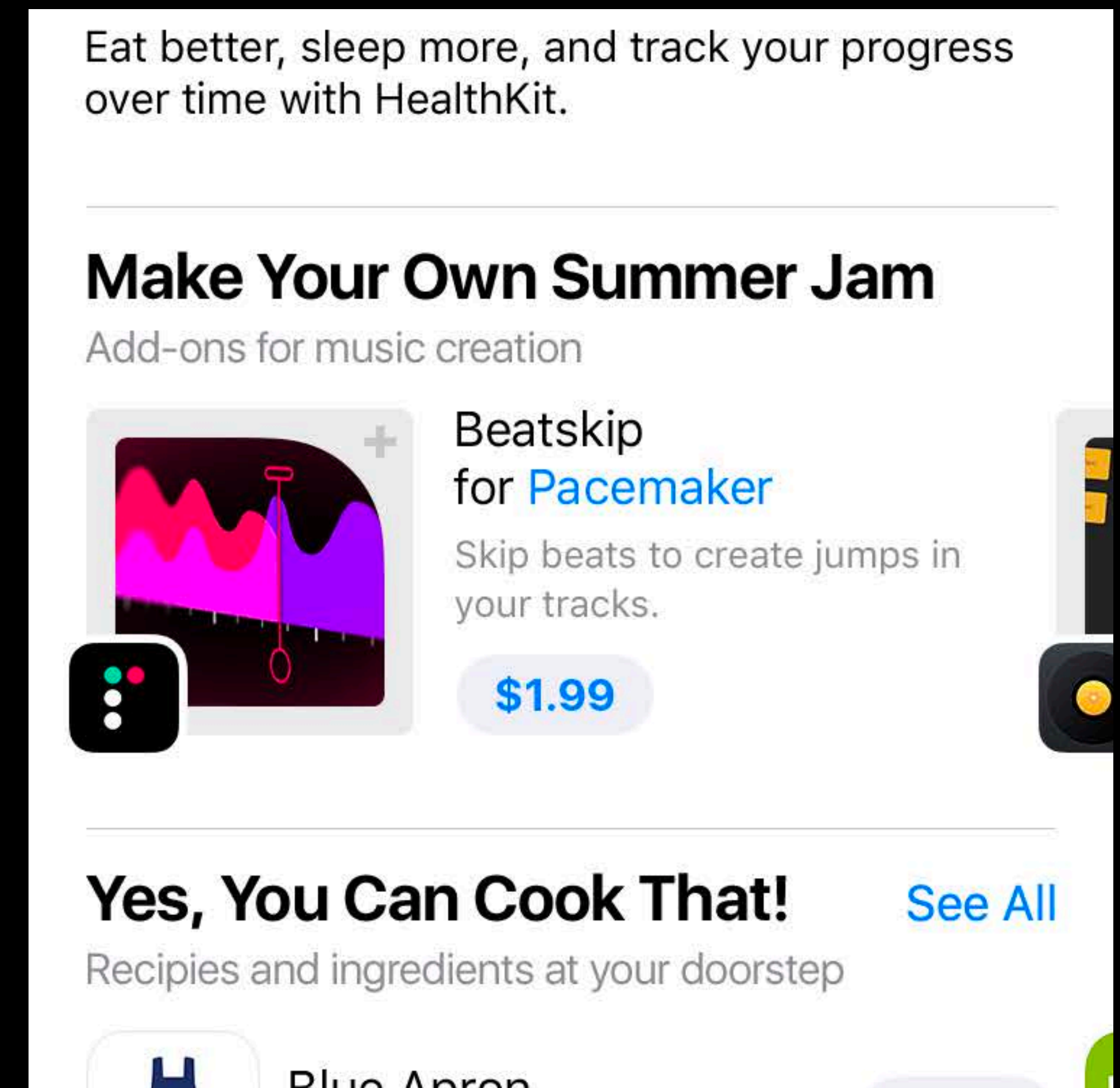
 Blue Apron

Promoting In-App Purchases

Discoverable

- App page
- Editorial features
- Search results

Start purchase on the App Store



Implementation Overview

Implementation Overview

Required

- Set up in iTunes Connect
- Handle info from App Store

Implementation Overview

Required

- Set up in iTunes Connect
- Handle info from App Store

Optional

- Order and visibility

Implementation Overview

Required

- Set up in iTunes Connect
- Handle info from App Store

Optional

- Order and visibility



9:41 AM

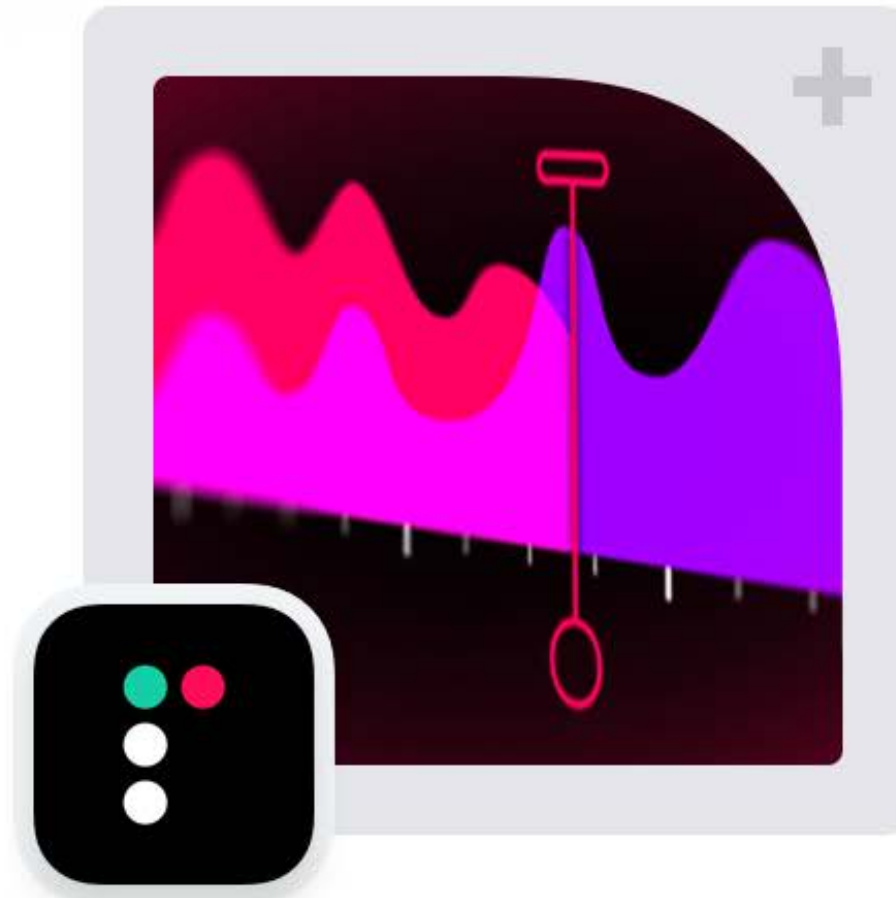
100%

TUESDAY, JUNE 6

Today



IN-APP PURCHASE



Reverb for Pacemaker

Size matters—control the size
of the room

\$1.99



Today



Games



Apps



Updates



Search



9:41 AM

100%

TUESDAY, JUNE 6

Today



IN-APP PURCHASE



Reverb for **Pacemaker**

Size matters—control the size
of the room

\$1.99



Today



Games



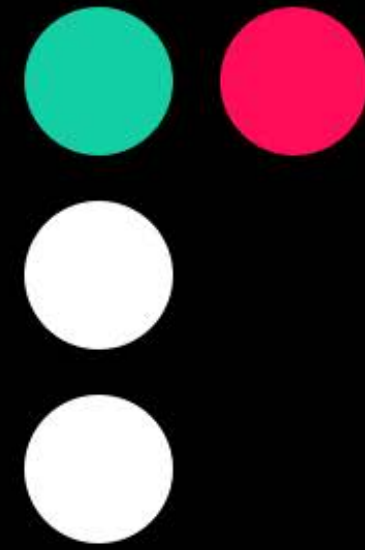
Apps



Updates



Search



Pacemaker®

App Store

Cancel



REVERB
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE

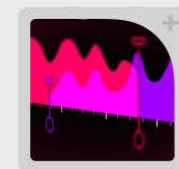
\$1.99



Buy with Touch ID

App Store

Cancel



REVERB
PACEMAKER
IN-APP PURCHASE

RATING 4+

ACCOUNT J.APPLESEED@ICLOUD.COM

PAY APP STORE

\$1.99



Done

```
// Continuing a Transaction from the App Store

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, shouldAddStorePayment payment: SKPayment,
                  forProduct product: SKProduct) -> Bool {
    return true
}
```

```
// Continuing a Transaction from the App Store

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, shouldAddStorePayment payment: SKPayment,
                  forProduct product: SKProduct) -> Bool {
    return true
}
```



```
// Deferring or Stopping a Transaction

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, shouldAddStorePayment payment: SKPayment,
                 forProduct product: SKProduct) -> Bool {
    // Hold on to the payment
    return false
}

SKPaymentQueue.default().add(savedPayment)
```

```
// Deferring or Stopping a Transaction
```

```
// MARK: - SKPaymentTransactionObserver
```

```
func paymentQueue(_ queue: SKPaymentQueue, shouldAddStorePayment payment: SKPayment,  
                 forProduct product: SKProduct) -> Bool {  
    // Hold on to the payment  
    return false  
}
```

```
SKPaymentQueue.default().add(savedPayment)
```

```
// Deferring or Stopping a Transaction

// MARK: - SKPaymentTransactionObserver

func paymentQueue(_ queue: SKPaymentQueue, shouldAddStorePayment payment: SKPayment,
                  forProduct product: SKProduct) -> Bool {
    // Hold on to the payment
    return false
}
```

```
SKPaymentQueue.default().add(savedPayment)
```

Testing Purchases

Protocol	<code>itms-services://</code>	
Parameters	<code>"action"</code>	<code>"purchaseIntent"</code>
	<code>"bundleId"</code>	<code>com.example.app</code>
	<code>"productIdentifier"</code>	<code>product_name</code>

Testing Purchases

Protocol	<code>itms-services://</code>	
Parameters	<code>"action"</code>	<code>"purchaseIntent"</code>
	<code>"bundleId"</code>	<code>com.example.app</code>
	<code>"productIdentifier"</code>	<code>product_name</code>

```
itms-services://?action=purchaseIntent&bundleId=com.example.app&productIdentifier=product_name
```

Order and Visibility

Order and Visibility

Defaults in iTunes Connect

Order and Visibility

Defaults in iTunes Connect

Override on device

Order and Visibility

Defaults in iTunes Connect

Override on device

Not synced

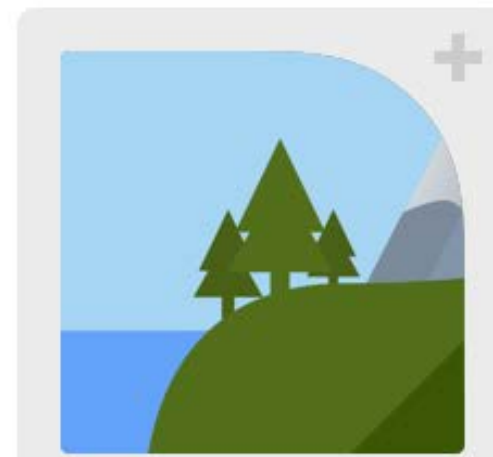
Visibility



Pro Subscription

Detailed topography and satellite imagery.

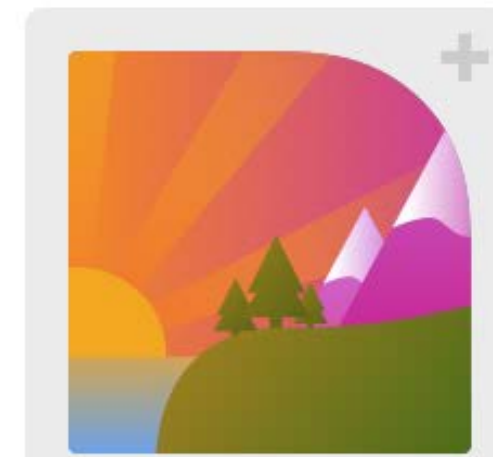
\$35.99



Fishing Hot Spots

Find hidden streams and lakes for hundreds of top fishing spots.

\$1.99

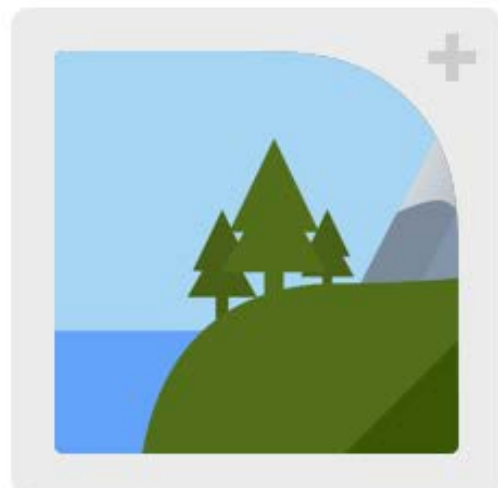


Hidden Beaches

Get to the most isolated beaches away from all the tourists.

\$1.99

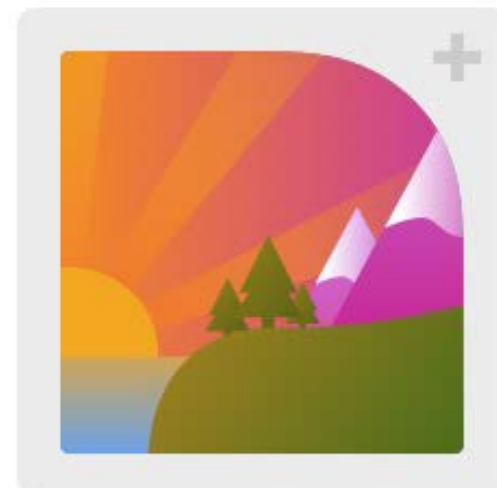
Visibility



Fishing Hot Spots

Find hidden streams and lakes for hundreds of top fishing spots.

\$1.99



Hidden Beaches

Get to the most isolated beaches away from all the tourists.

\$1.99

```
// Updating Visibility Override of a Promoted In-App Purchase

// Fetch Product Info for Pro Subscription

let storePromotionController = SKProductStorePromotionController.default()
storePromotionController.update(storePromotionVisibility: .hide, forProduct: proSubscription,
    completionHandler: { (error: Error?) in
        // Complete
    })
```

```
// Updating Visibility Override of a Promoted In-App Purchase
```

```
// Fetch Product Info for Pro Subscription
```

```
let storePromotionController = SKProductStorePromotionController.default()  
storePromotionController.update(storePromotionVisibility: .hide, forProduct: proSubscription,  
    completionHandler: { (error: Error?) in  
        // Complete  
    })
```

```
// Updating Visibility Override of a Promoted In-App Purchase
```

```
// Fetch Product Info for Pro Subscription
```

```
let storePromotionController = SKProductStorePromotionController.default()
```

```
storePromotionController.update(storePromotionVisibility: .hide, forProduct: proSubscription,  
    completionHandler: { (error: Error?) in  
        // Complete  
    })
```

```
// Reading Visibility Override of a Promoted In-App Purchase

// Fetch Product Info for Hidden Beaches pack

let storePromotionController = SKProductStorePromotionController.default()
storePromotionController.fetchStorePromotionVisibility(forProduct: hiddenBeaches,
    completionHandler: { (visibility: SKProductStorePromotionVisibility, error: Error?) in
        // visibility == .default
    })
```

```
// Reading Visibility Override of a Promoted In-App Purchase
```

```
// Fetch Product Info for Hidden Beaches pack
```

```
let storePromotionController = SKProductStorePromotionController.default()  
storePromotionController.fetchStorePromotionVisibility(forProduct: hiddenBeaches,  
    completionHandler: { (visibility: SKProductStorePromotionVisibility, error: Error?) in  
        // visibility == .default  
    })
```

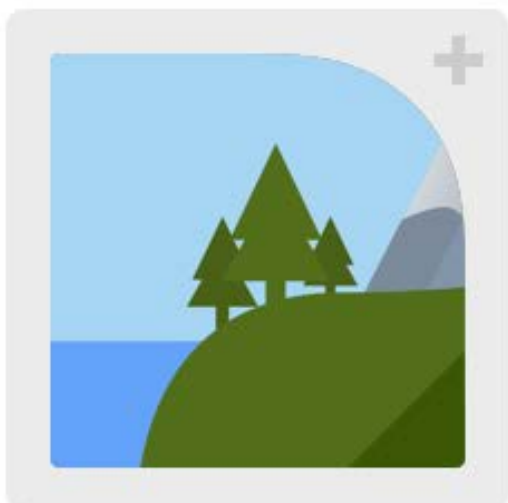

Order



Pro Subscription

Detailed topography and satellite imagery.

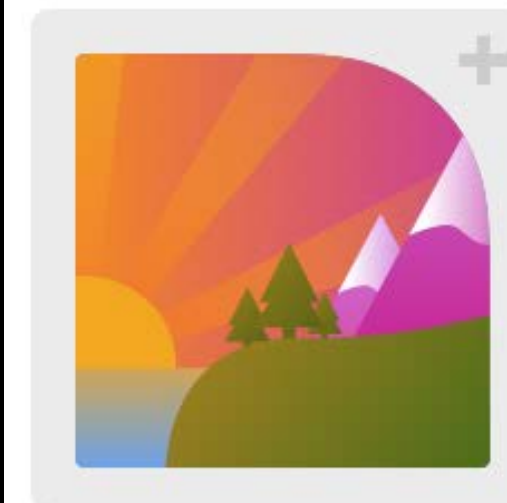
\$35.99



Fishing Hot Spots

Find hidden streams and lakes for hundreds of top fishing spots.

\$1.99

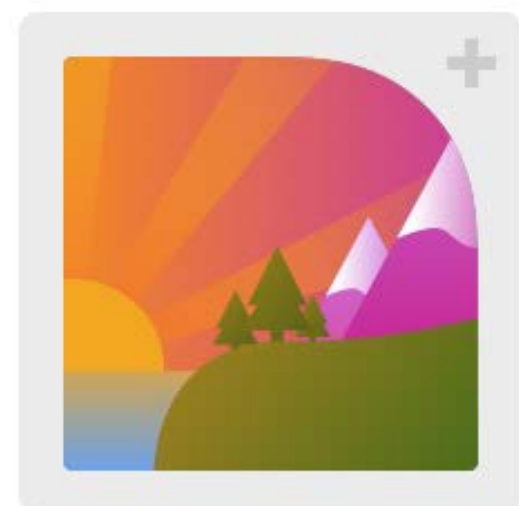


Hidden Beaches

Get to the most isolated beaches away from all the tourists.

\$1.99

Order



Hidden Beaches

Get to the most isolated beaches away from all the tourists.

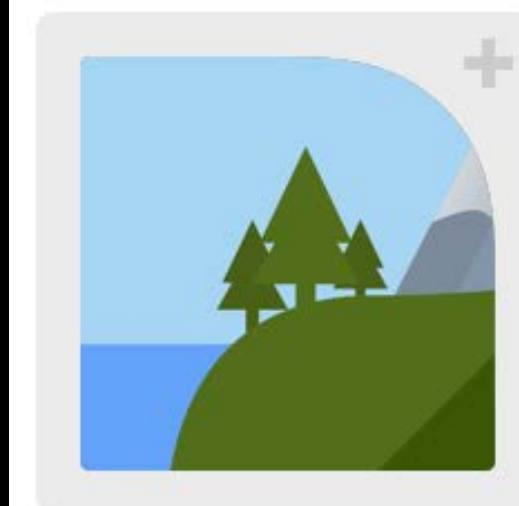
\$1.99



Pro Subscription

Detailed topography and satellite imagery.

\$35.99



Fishing Hot Spots

Find hidden streams and lakes for hundreds of top fishing spots.

\$1.99

```
// Updating Order Override of Promoted In-App Purchases

// Fetch Product Info for Pro Subscription, Fishing Hot Spots, and Hidden Beaches

let storePromotionController = SKProductStorePromotionController.default()
let newProductsOrder = [hiddenBeaches, proSubscription, fishingHotSpots]
storePromotionController.updateStorePromotionOrder(newProductsOrder,
    completionHandler: { (error: Error?) in
        // Complete
    })
```

```
// Updating Order Override of Promoted In-App Purchases

// Fetch Product Info for Pro Subscription, Fishing Hot Spots, and Hidden Beaches

let storePromotionController = SKProductStorePromotionController.default()
let newProductsOrder = [hiddenBeaches, proSubscription, fishingHotSpots]
storePromotionController.updateStorePromotionOrder(newProductsOrder,
    completionHandler: { (error: Error?) in
        // Complete
    })
```

```
// Updating Order Override of Promoted In-App Purchases

// Fetch Product Info for Pro Subscription, Fishing Hot Spots, and Hidden Beaches

let storePromotionController = SKProductStorePromotionController.default()
let newProductsOrder = [hiddenBeaches, proSubscription, fishingHotSpots]
storePromotionController.updateStorePromotionOrder(newProductsOrder,
    completionHandler: { (error: Error?) in
        // Complete
    })
```

```
// Reading Order Override of Promoted In-App Purchases

let storePromotionController = SKProductStorePromotionController.default()
storePromotionController.fetchStorePromotionOrder(completionHandler: {
    (products: [SKProduct], error: Error?) in
        // products == [hiddenBeaches, proSubscription, fishingHotSpots]
    })
```

```
// Reading Order Override of Promoted In-App Purchases
```

```
let storePromotionController = SKProductStorePromotionController.default()  
storePromotionController.fetchStorePromotionOrder(completionHandler: {  
    (products: [SKProduct], error: Error?) in  
        // products == [hiddenBeaches, proSubscription, fishingHotSpots]  
    })
```

Promoting In-App Purchases

Promoting In-App Purchases

Discoverable in App Store

Promoting In-App Purchases

Discoverable in App Store

Set up in iTunes Connect

Promoting In-App Purchases

Discoverable in App Store

Set up in iTunes Connect

Start purchase in App Store

Promoting In-App Purchases

Discoverable in App Store

Set up in iTunes Connect

Start purchase in App Store

Handle in app via `SKPaymentTransactionObserver`

Promoting In-App Purchases

Discoverable in App Store

Set up in iTunes Connect

Start purchase in App Store

Handle in app via `SKPaymentTransactionObserver`

Optional—order and visibility

Ratings, Reviews, and Responses

Pete Hare, App Store Engineer



9:41 AM

100%

[← Games](#)



Alto's Adventure

A serene snowboarding odyssey

\$0.99



4.7, 7.8K Ratings



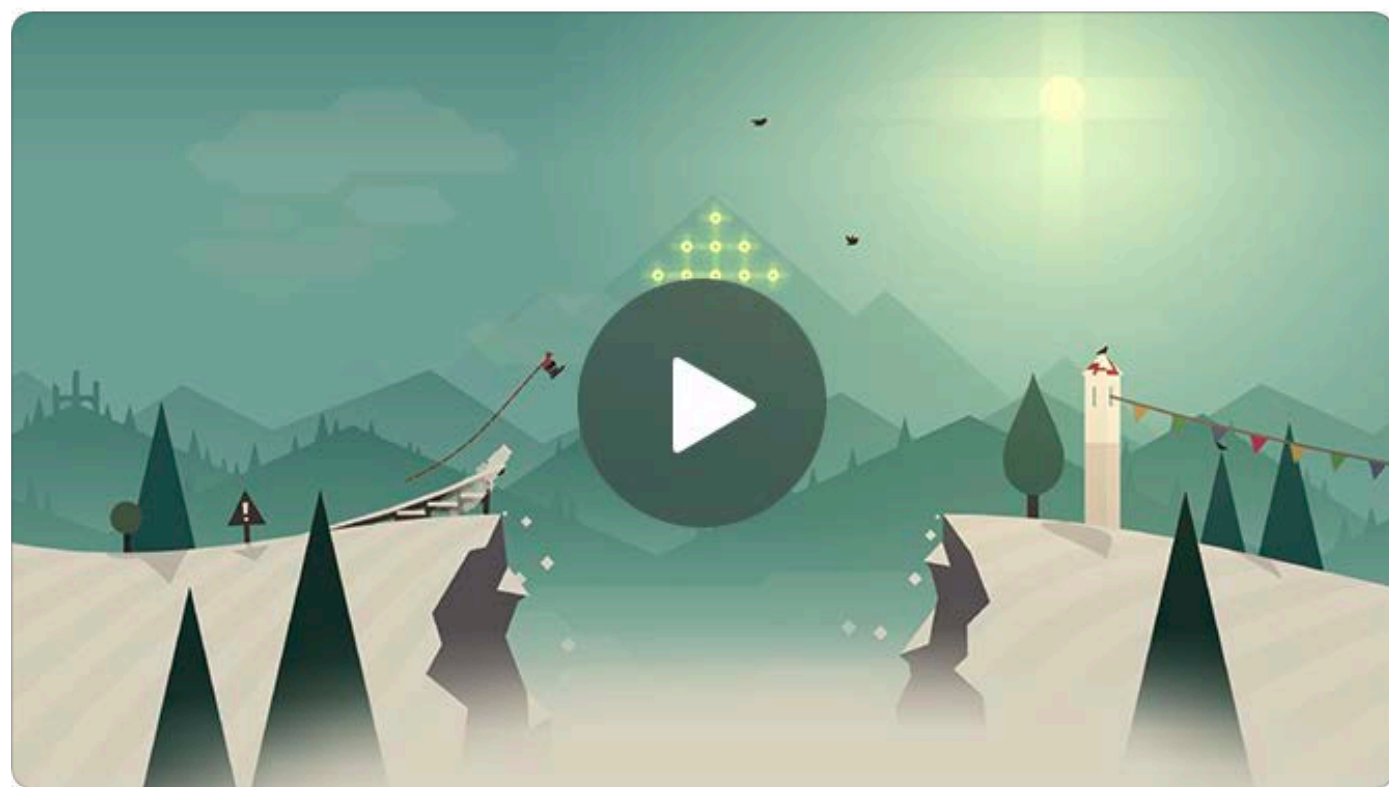
Game

#44

Action

9+

Age Rating



Offers iPad and Apple TV Apps



Above the placid ivory snow lies a sleepy mountain village, brimming with the promise of adventure. Join Alto and his friends as they [more](#)

Developer



Today



Games



Apps



Updates



Search



9:41 AM

100%

[← Games](#)

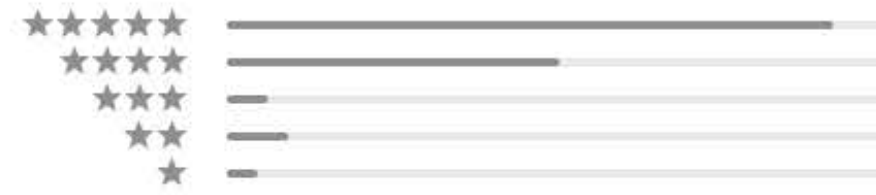


\$0.99

Ratings & Reviews

[See All](#)

4.7



out of 5

7,867 Ratings

Truly Amazing Game

1d ago



CuriousFox25

This game is wonderful and highly enjoyable for the entire family! I love how easy it is to pick up and play wherever you are. If you need an infinite runner to play with the kids, this is the one to get. After a long day, the colorful visuals and cheery music always put me in a good mood [more](#)



Editors' Choice



In this gorgeous twist on the infinite runner, you guide an agile snowboarder down a never ending mountain, pulling [more](#)



Today



Games



Apps



Updates



Search



9:41 AM

100%

[← Games](#)

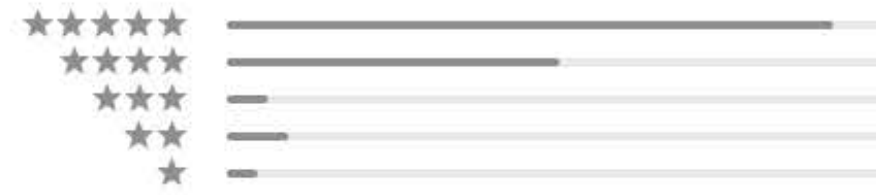


\$0.99

Ratings & Reviews

[See All](#)

4.7



out of 5

7,867 Ratings

Impossible to Stop Playing

2d ago



PhilSchiller

This game rocks! Ever since I unlocked the wingsuit I haven't been able to put it down... even during our ET meetings.

Developer Response

1h ago

Just make sure to keep the volume off!



Editors' Choice



In this gorgeous twist on the infinite runner, you guide an agile snowboarder down a never ending mountain, pulling [more](#)



Today



Games



Apps



Updates



Search



9:41 AM

100%

[← Games](#)

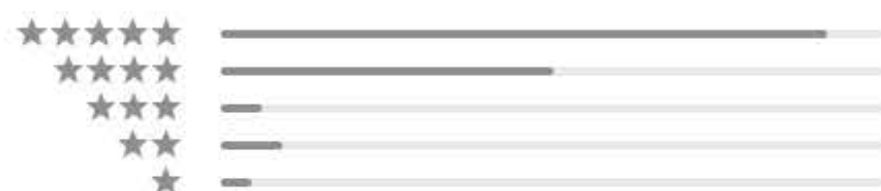


\$0.99

Ratings & Reviews

[See All](#)

4.7



out of 5

7,867 Ratings

A Fun Challenge

4d ago



F1racer09

I can't even count how many hours I've spent playing this awesome game. The visuals are stunning and doing backflips is so addictive. But it's not easy—I had to replay one level multiple times before moving on. I'm getting better each time I play, but I still have a lot to learn.



Editors' Choice



In this gorgeous twist on the infinite runner, you guide an agile snowboarder down a never ending mountain, pulling [more](#)



Today



Games



Apps



Updates



Search

Ratings, Reviews, and Responses

What's new



NEW

Ratings, Reviews, and Responses

What's new

NEW

Reset your rating

Ratings, Reviews, and Responses

What's new



NEW

Reset your rating

Respond to reviews

Ratings, Reviews, and Responses

What's new



NEW

Reset your rating

Respond to reviews

Ask for ratings and reviews via `SKStoreReviewController`

Ratings, Reviews, and Responses

What's new



NEW

Reset your rating

Respond to reviews

Ask for ratings and reviews via `SKStoreReviewController`

Deep link to write review in the App Store

Ratings, Reviews, and Responses

What's new



NEW

Reset your rating

Respond to reviews

Ask for ratings and reviews via `SKStoreReviewController`

Deep link to write review in the App Store

Helpfulness and Report a Concern on iOS

Helpful



Not Helpful



Report a Concern





9:41 AM

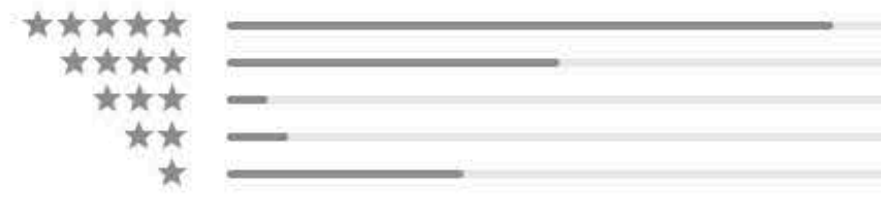
100%

[← Alto's Adventure](#)

Ratings & Reviews

4.7

out of 5



24.467 Ratings

Tap to



Write a review

[Support](#)

Helpful

[Useful](#)

Thanks for your feedback.

Tru

ago



015

This game is wonderful and highly enjoyable for the entire family! I love how easy it is to pick up and play wherever you are. If you need an infinite runner to play with the kids, this is the one to get. After a long day, the colorful visuals and cheery music always puts me in a good mood.

Impossible to stop playing

4d ago



51



9:41 AM

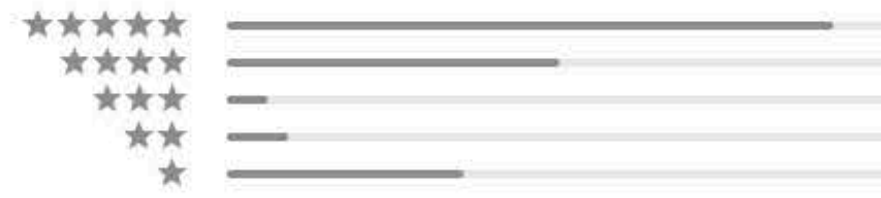
100%

[← Alto's Adventure](#)

Ratings & Reviews

4.7

out of 5



24.467 Ratings

Tap to



Write a review

Support

Reported

Helpful

Thanks for your feedback.

Tru

ago



015

This game is wonderful and highly enjoyable for the entire family! I love how easy it is to pick up and play wherever you are. If you need an infinite runner to play with the kids, this is the one to get. After a long day, the colorful visuals and cheery music always puts me in a good mood.

Impossible to stop playing

4d ago



51

Ratings, Reviews, and Responses

Responding to reviews

I loved this game



Jan 30

Comcup

I have never written a review before, but this game is so good. I just want more levels. I played through this game at [More](#)

Developer Response

Feb 14

Thanks for making us your first review! Keep checking for updates, there's n [More](#)

Awesome game



Feb 2

De Maximus

If you like unusual challenge of ripping a paper up to save the cute little guys, this game is fun to be stumped, save the [More](#)

Developer Response

Mar 16

Glad you had fun getting stumped. We're big fans of unusual challenges, too. E [More](#)



9:41 AM

100%

[← All Inboxes \(2\)](#)



From: [Apple](#) >

[Hide](#)

To: [Kelly Westover](#) >

You've received a developer response to your review of TripGuides

Today at 9:41 AM



App Store

Dear Johnny,

App Co. responded to your review of



TripGuides
App Co.

"Glad you're enjoying the game though I'm sorry you're having problems on that level, we'll be issuing an update soon to fix it. In the meantime we suggest you go level 3 manually via Load Game in the Main Menu. Sorry again!"

Do you want to [update your review?](#)

You can also [email the developer.](#)





9:41 AM

100%

< All Inboxes (2)



From: [Apple](#) >

Hide

To: [Kelly Westover](#) >

You've received a developer response to your review of TripGuides

Today at 9:41 AM



App Store

Dear Johnny,

App Co. responded to your review of



TripGuides
App Co.

"Glad you're enjoying the game though I'm sorry you're having problems on that level, we'll be issuing an update soon to fix it. In the meantime we suggest you go level 3 manually via Load Game in the Main Menu. Sorry again!"

Do you want

[update your review](#)

You can also [email the developer](#).





9:41 AM

100%

Cancel

Edit Review

Send



Tap a Star to Rate

Great improvements!

This game has improved a LOT. There were some problems with the earlier version, but since updating I've had no issues. Thanks for listening!

I

The

This

1

2

3

4

5

6

7

8

9

0

-

/

:

;

(

)

\$

&

@

"

#+=

.

,

?

!

'

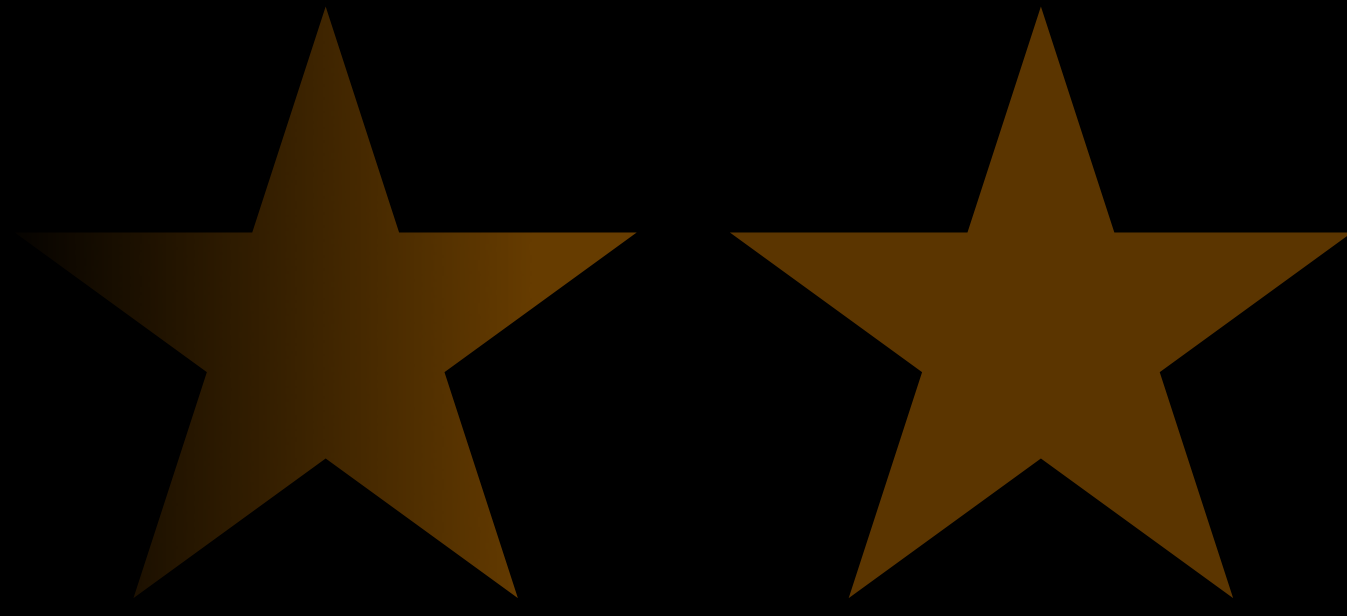


ABC



space

return





Average increase of 1.5 stars per review

Ratings, Reviews, and Responses

Responding to reviews

More information

<https://developer.apple.com/app-store/responding-to-reviews>

Ratings, Reviews, and Responses

Responding to reviews

More information

<https://developer.apple.com/app-store/responding-to-reviews>

Ratings, Reviews, and Responses

Receiving reviews

Ratings, Reviews, and Responses

Receiving reviews

Prompt for review with `SKStoreReviewController`

Ratings, Reviews, and Responses

Receiving reviews

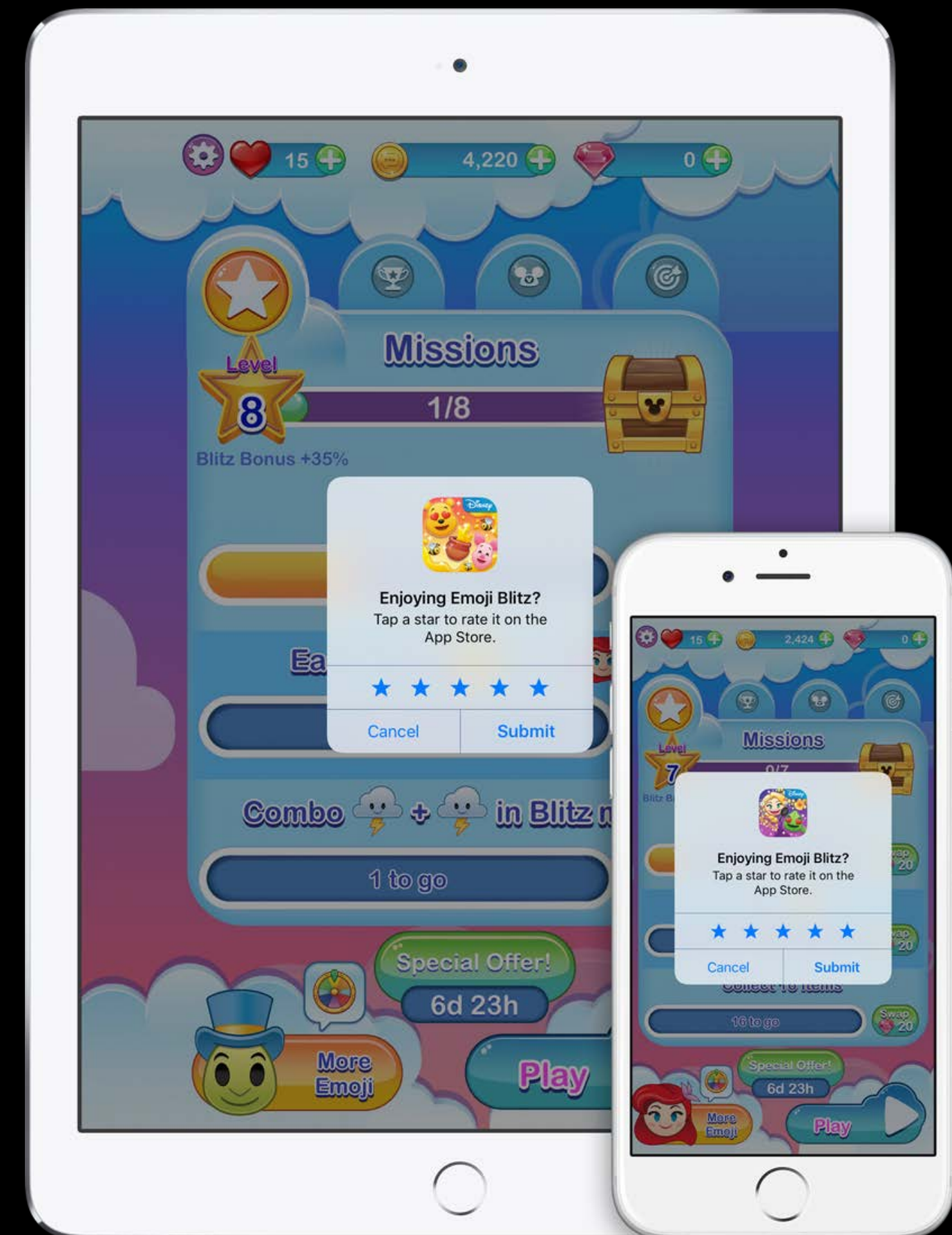
Prompt for review with `SKStoreReviewController`

Deep link to review in App Store

Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

NEW

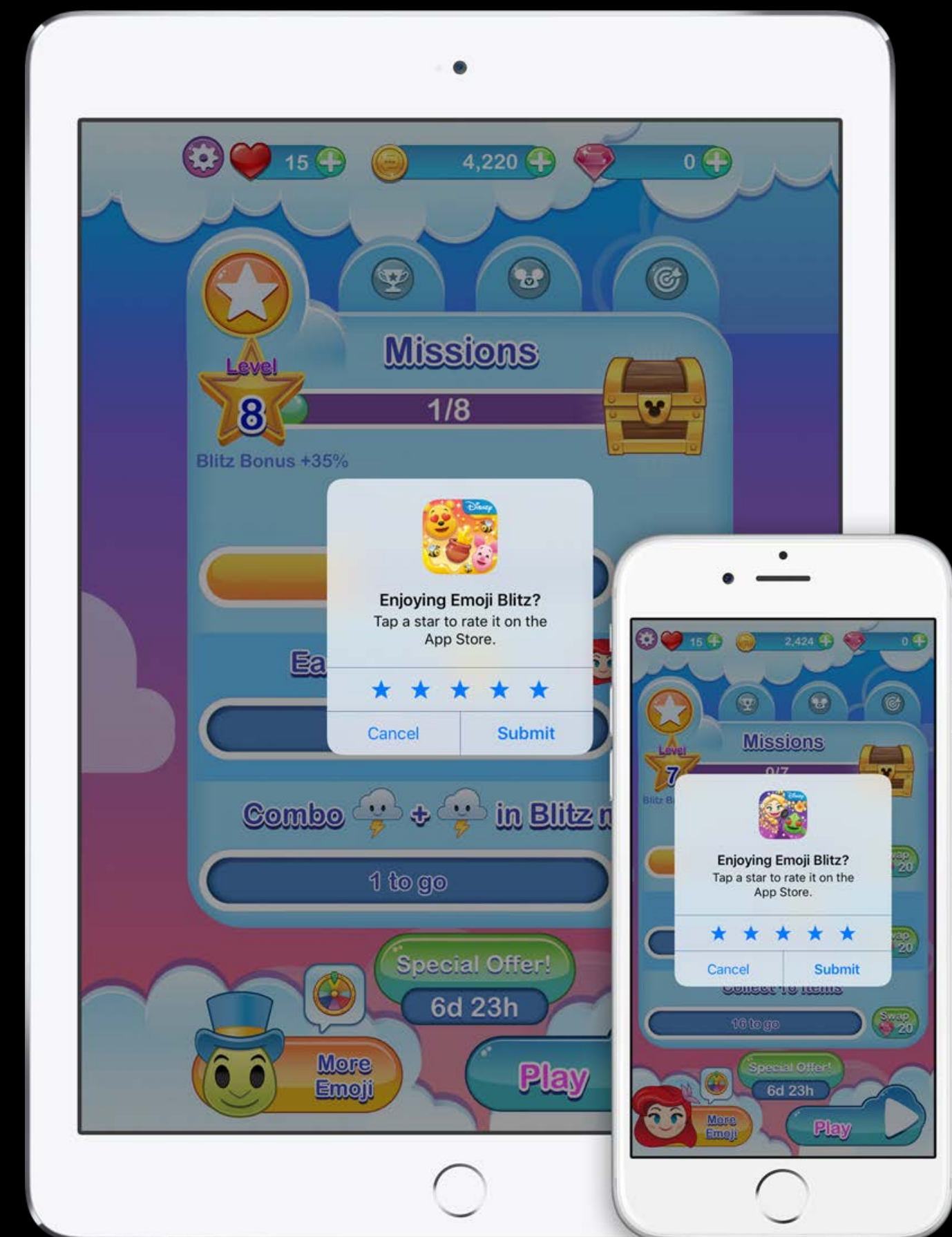


Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

NEW

Introduced in iOS 10.3



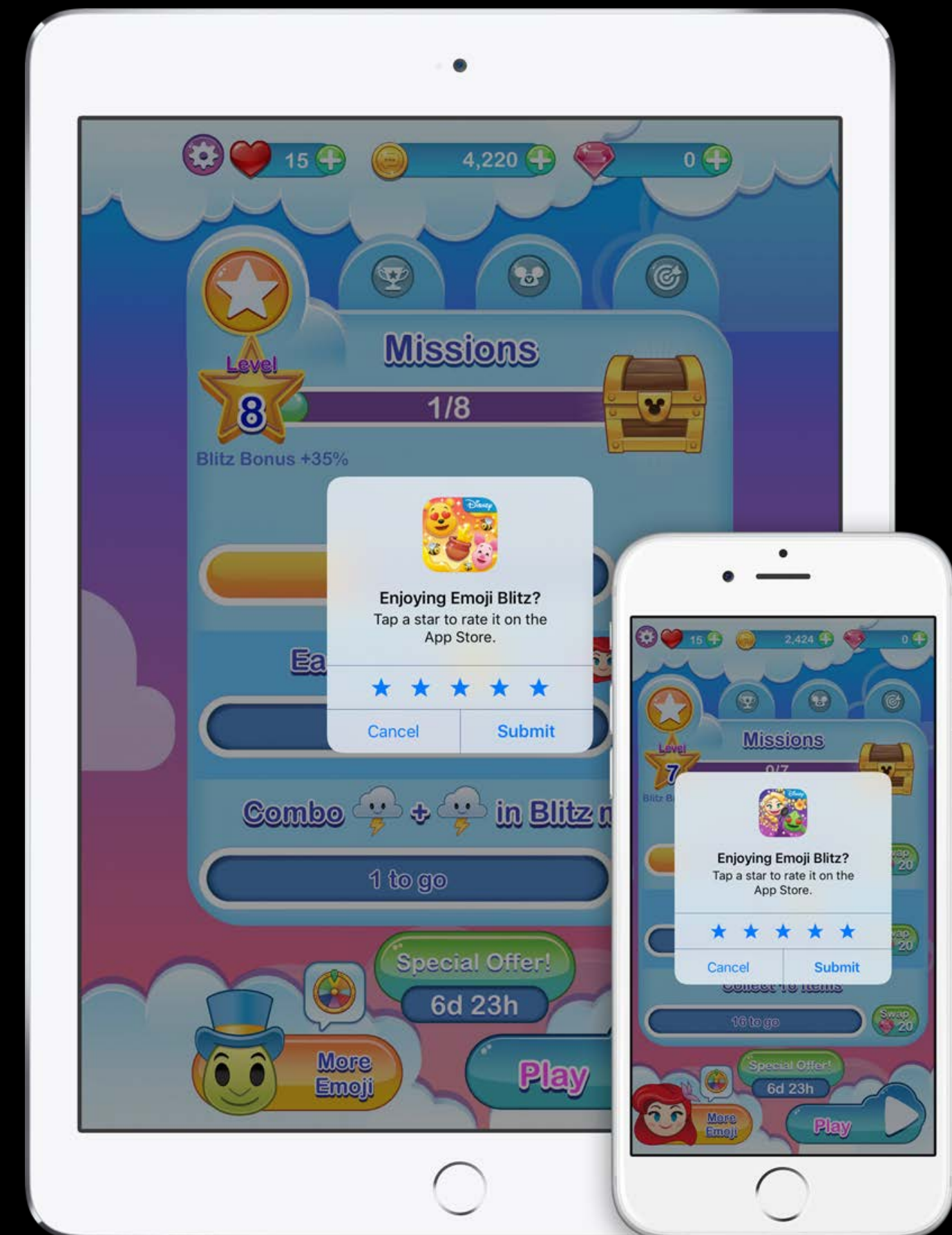
Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

NEW

Introduced in iOS 10.3

Quick way to request a rating/review



Ratings, Reviews, and Responses

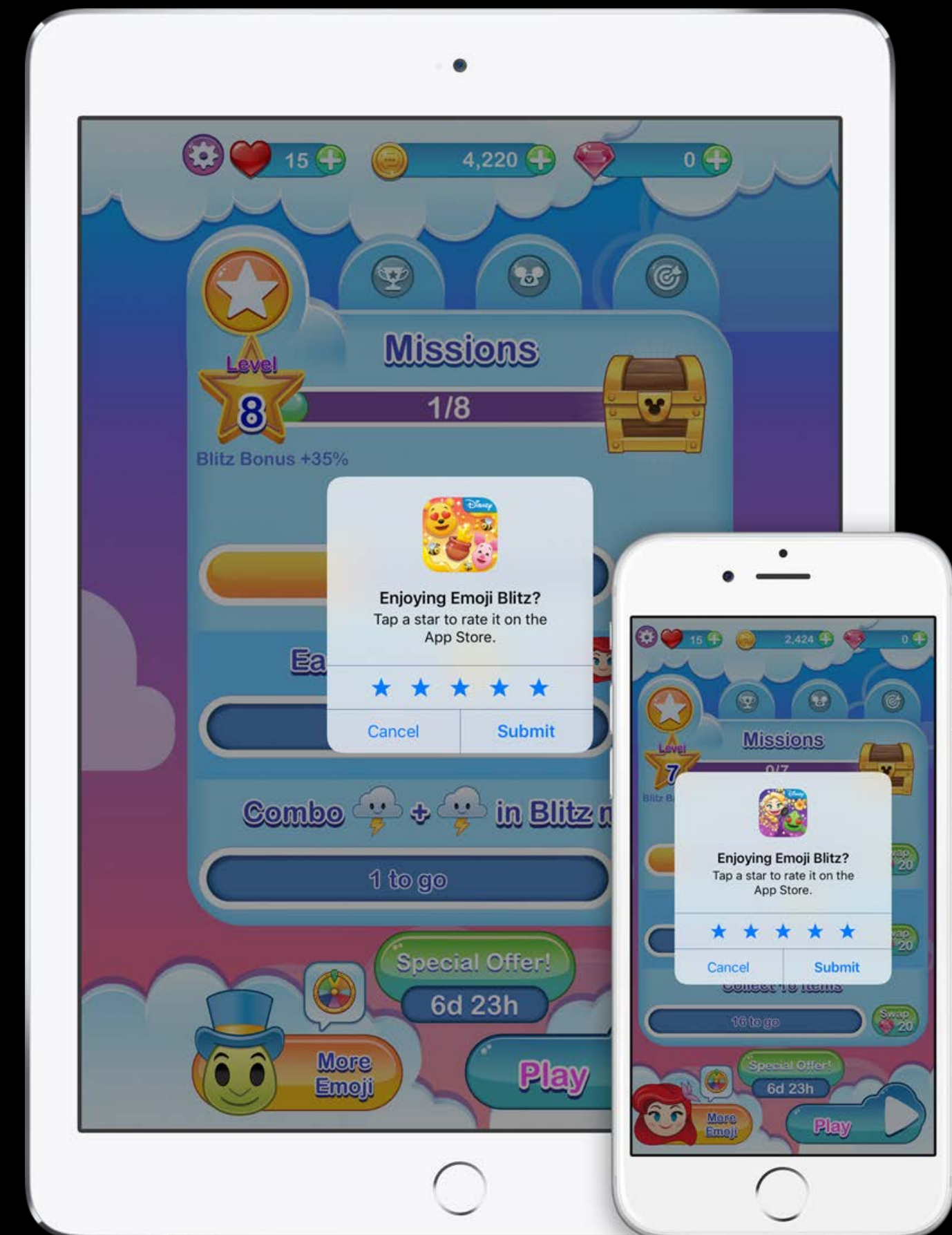
Asking for ratings and reviews with SKStoreReviewController

NEW

Introduced in iOS 10.3

Quick way to request a rating/review

Will be required for all modal rating/
review prompts



Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

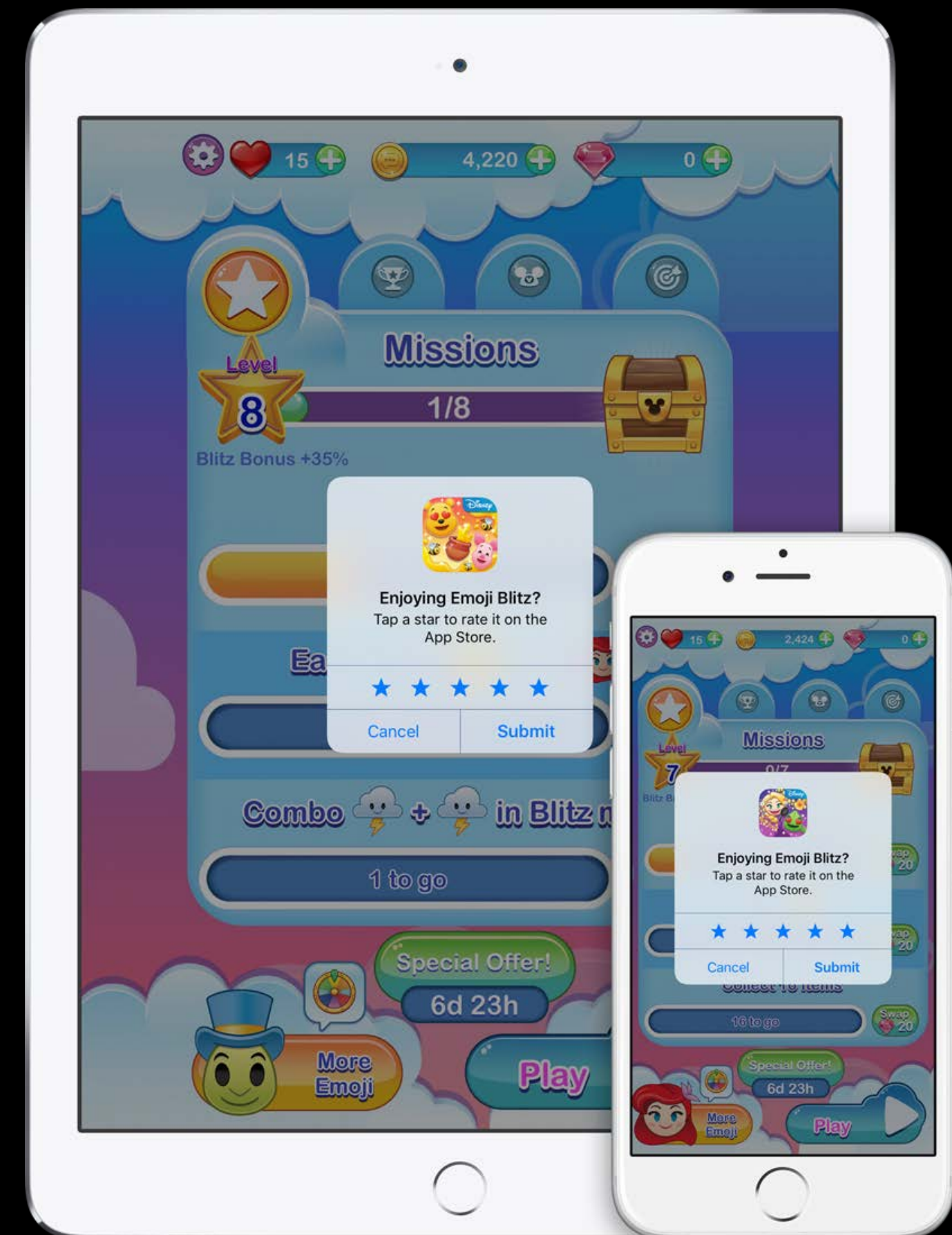
NEW

Introduced in iOS 10.3

Quick way to request a rating/review

Will be required for all modal rating/
review prompts

Restrictions in place



Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

NEW

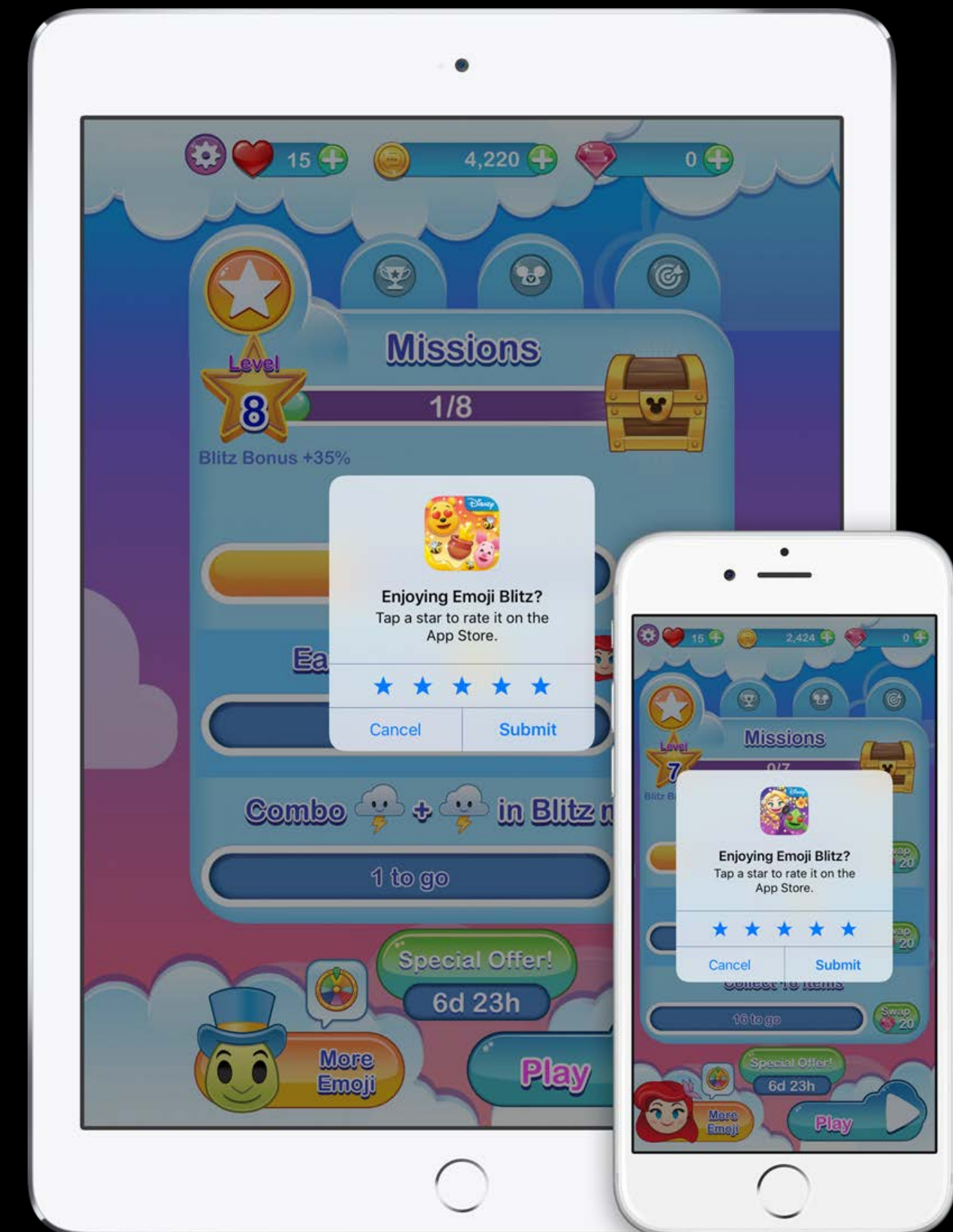
Introduced in iOS 10.3

Quick way to request a rating/review

Will be required for all modal rating/
review prompts

Restrictions in place

- Limited requests per device



Ratings, Reviews, and Responses

Asking for ratings and reviews with SKStoreReviewController

NEW

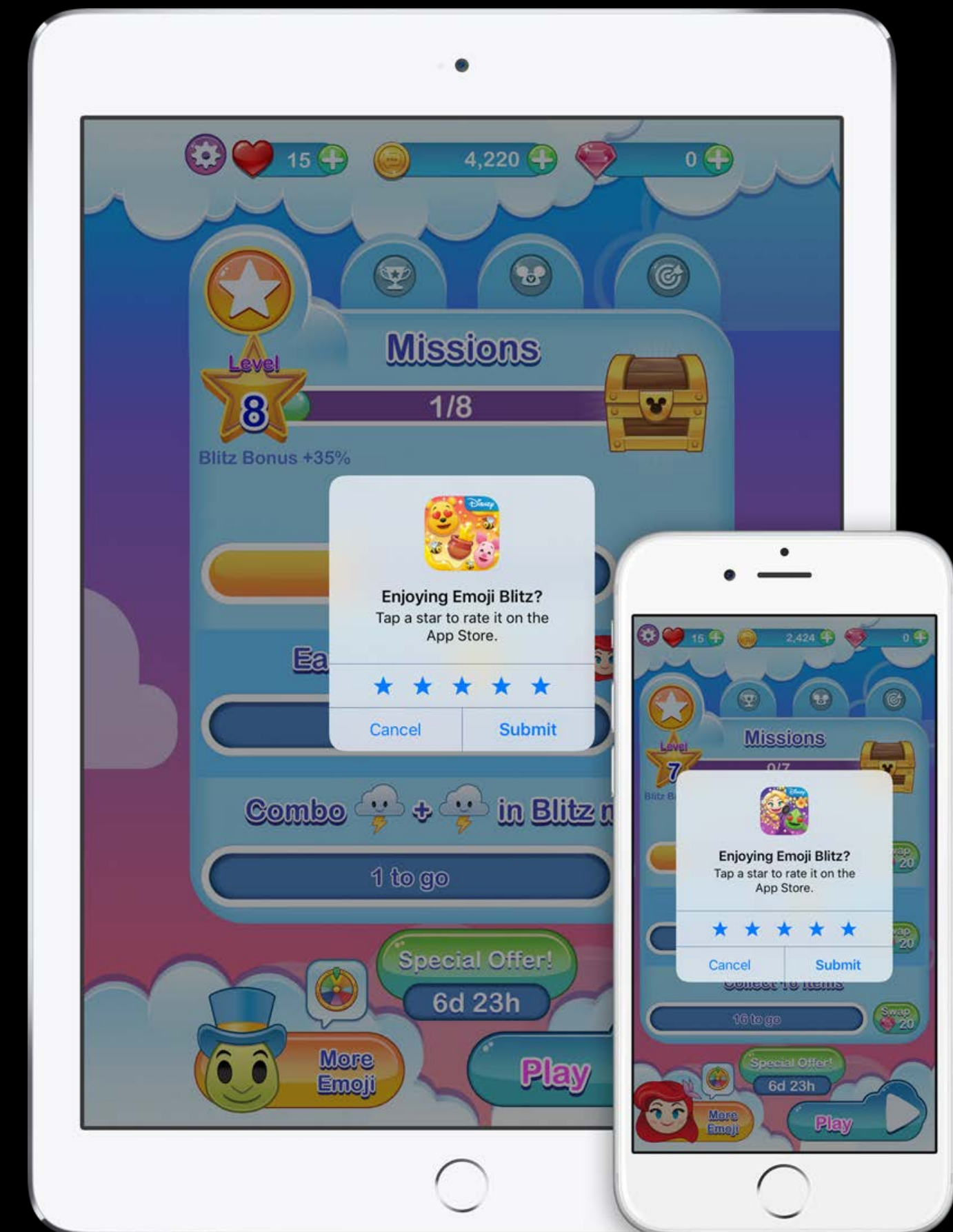
Introduced in iOS 10.3

Quick way to request a rating/review

Will be required for all modal rating/
review prompts

Restrictions in place

- Limited requests per device
- Can be disabled by user in Settings



Ratings, Reviews, and Responses

Asking for ratings and reviews with `SKStoreReviewController`

Ratings, Reviews, and Responses

Asking for ratings and reviews with `SKStoreReviewController`



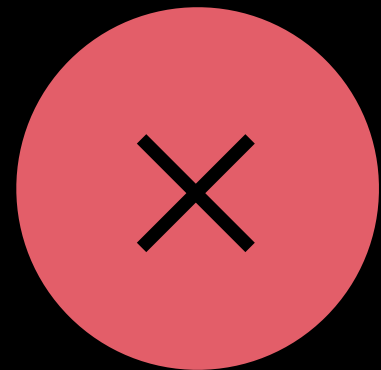
Present as a prompt after a user action

Ratings, Reviews, and Responses

Asking for ratings and reviews with `SKStoreReviewController`



Present as a prompt after a user action



Do not present from a button

May not present any UI due to restrictions


```
// Asking for Ratings and Reviews with SKStoreReviewController
```

```
if shouldPromptUser() {  
    SKStoreReviewController.requestReview()  
}
```

```
func shouldPromptUser() -> Bool {  
    // Local business rules  
}
```

```
// Asking for Ratings and Reviews with SKStoreReviewController
```

```
if shouldPromptUser() {  
    SKStoreReviewController.requestReview()  
}
```

```
func shouldPromptUser() -> Bool {  
    // Local business rules  
}
```

```
// Asking for Ratings and Reviews with SKStoreReviewController
```

```
if shouldPromptUser() {  
    SKStoreReviewController.requestReview()  
}
```

```
func shouldPromptUser() -> Bool {  
    // Local business rules  
}
```

```
// Asking for Ratings and Reviews with SKStoreReviewController
```

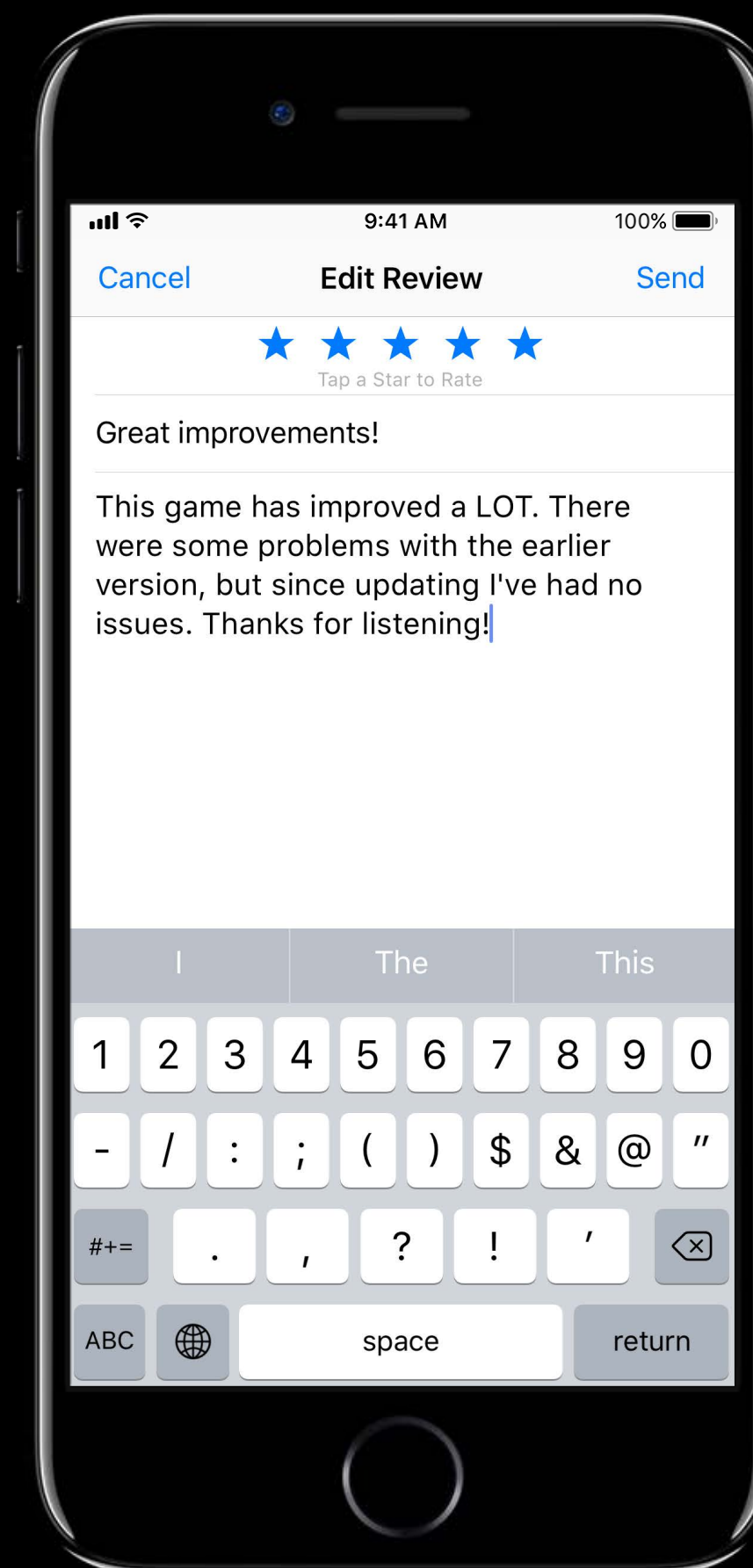
```
if shouldPromptUser() {  
    SKStoreReviewController.requestReview()  
}
```

```
func shouldPromptUser() -> Bool {  
    // Local business rules  
}
```

Ratings, Reviews, and Responses

Deep link to write a review in App Store

NEW

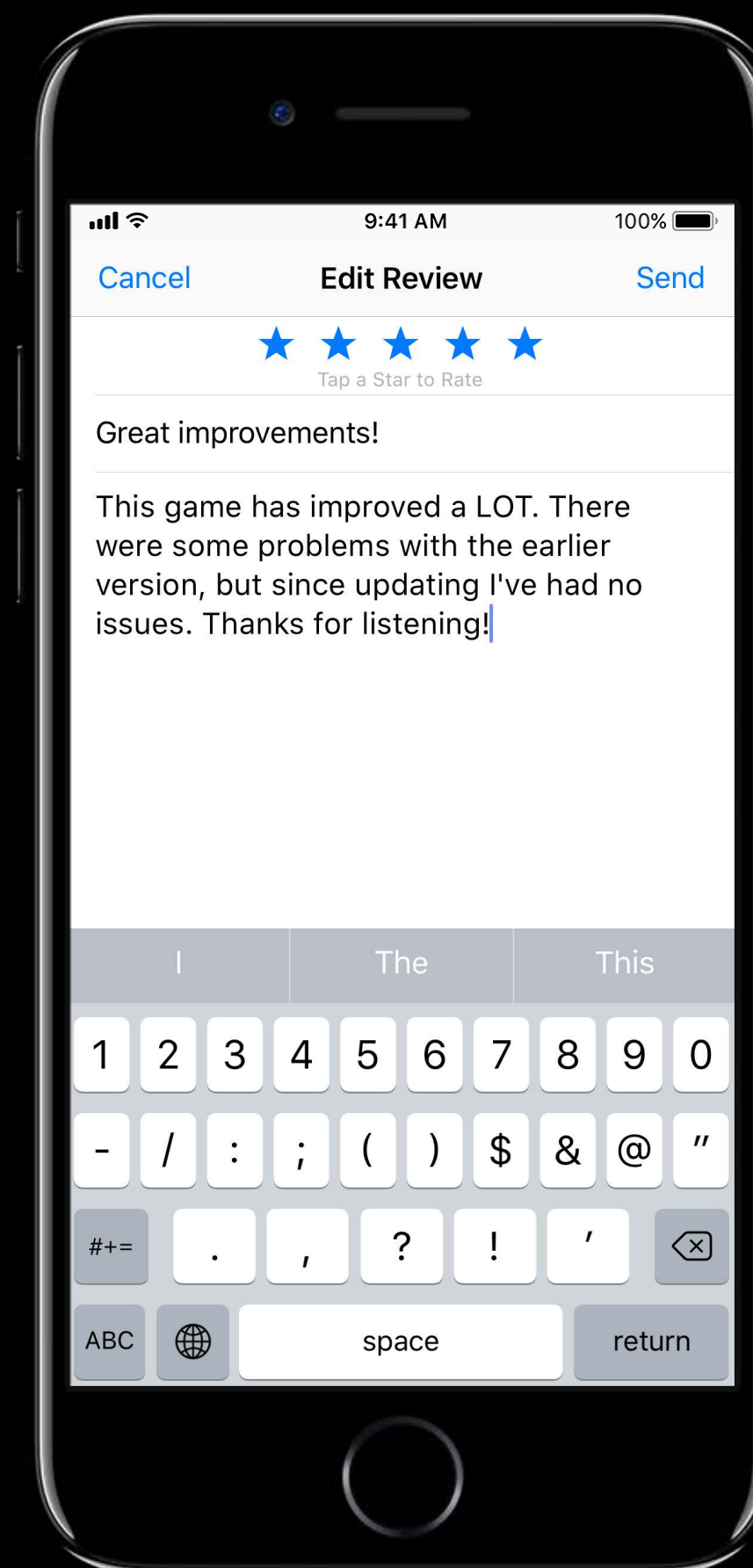


Ratings, Reviews, and Responses

Deep link to write a review in App Store

NEW

Introduced in iOS 10.3



Ratings, Reviews, and Responses

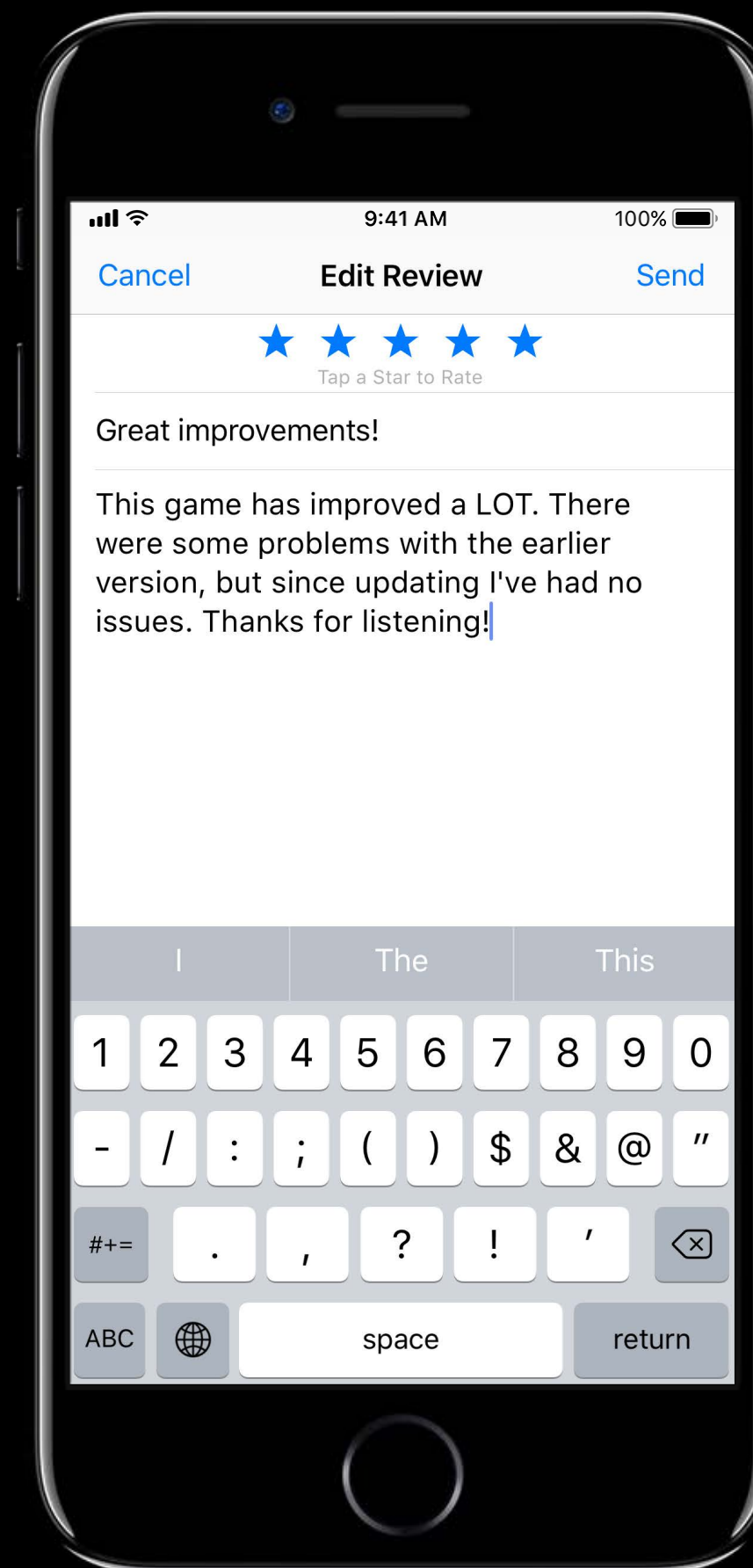
Deep link to write a review in App Store

NEW

Introduced in iOS 10.3

Link to open your app in the App Store

- Presents compose review from app page



Ratings, Reviews, and Responses

Deep link to write a review in App Store

NEW

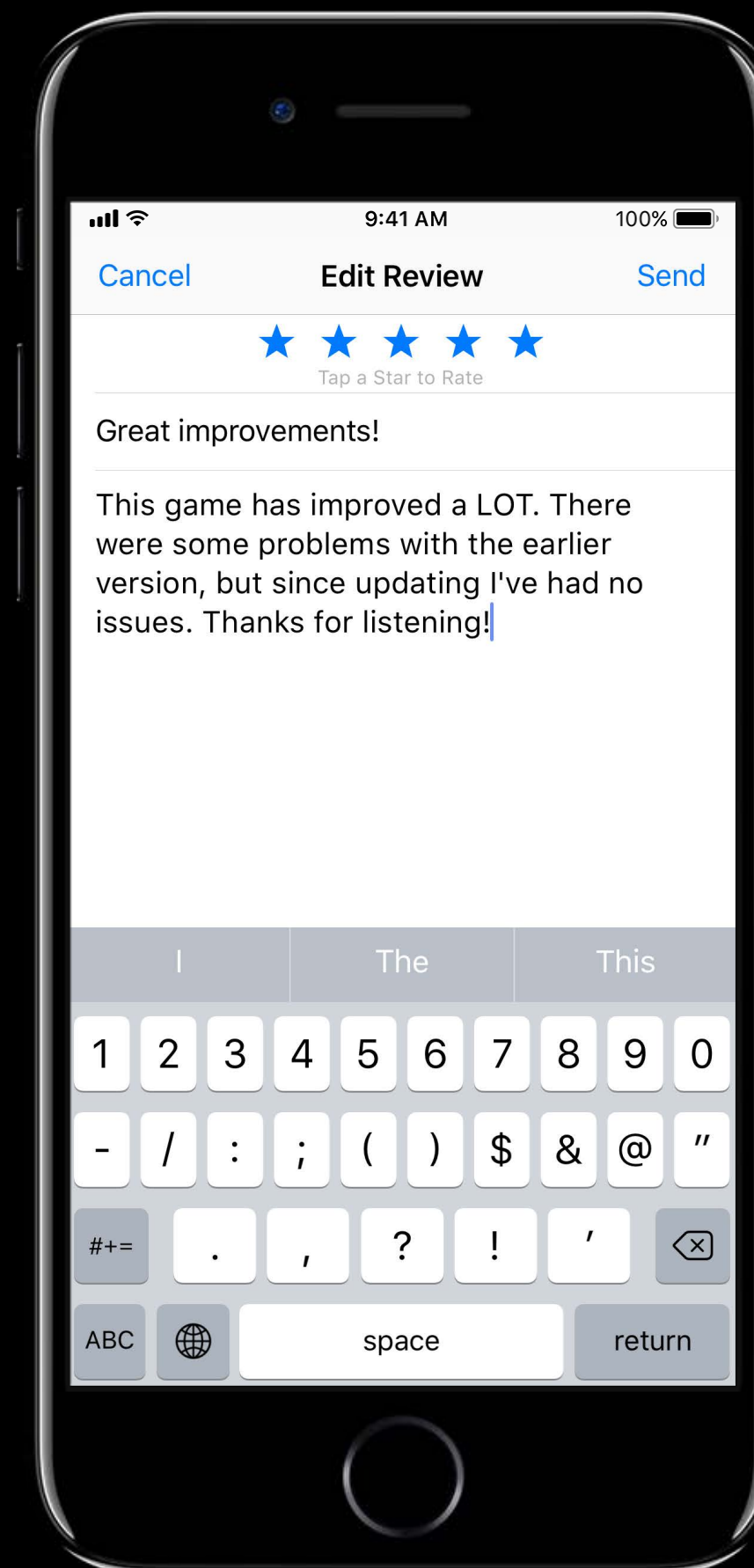
Introduced in iOS 10.3

Link to open your app in the App Store

- Presents compose review from app page

User initiated actions

- Button in settings



Ratings, Reviews, and Responses

Deep link to write a review in App Store

Ratings, Reviews, and Responses

Deep link to write a review in App Store



Use from an embedded button in your app

Such as a settings screen

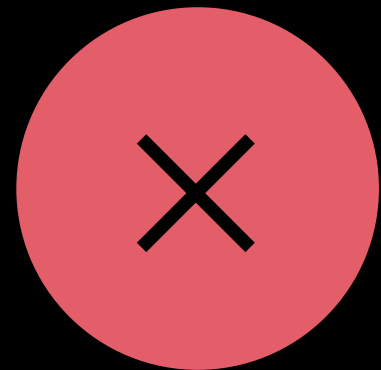
Ratings, Reviews, and Responses

Deep link to write a review in App Store



Use from an embedded button in your app

Such as a settings screen



Do not use from an alert

Use `SKStoreReviewController` instead

Deep Link to Write Review

URL is formed using regular product URL with an anchor tag

```
https://itunes.apple.com/us/app/itunes-u/id490217893?action=write-review
```

For creating product URLs visit
<https://linkmaker.itunes.apple.com/>

Ratings, Reviews, and Responses

Asking users for reviews

More information

<https://developer.apple.com/app-store/ratings-and-reviews/>

Summary

Summary

How to implement in-app purchases

Summary

How to implement in-app purchases

Promote in-app purchases in the App Store

Summary

How to implement in-app purchases

Promote in-app purchases in the App Store

New App Store design

Summary

How to implement in-app purchases

Promote in-app purchases in the App Store

New App Store design

New opportunities to improve your ratings and reviews

More Information

<https://developer.apple.com/wwdc17/303>

Related Sessions

What's New in iTunes Connect

WWDC 2017

Introducing the New App Store

WWDC 2017

[Advanced StoreKit](#)

Grand Ballroom A

Thursday 1:50PM

Labs

App Store and iTunes Connect Lab

Technology Lab H

Thu 12:00PM-1:50PM

StoreKit Lab

Technology Lab E

Thu 3:10PM-6:00PM

StoreKit Lab

Technology Lab E

Fri 1:50PM-4:00PM

