

Chapter 14.



Testing & Publication

2023-2024

COMP7506 Smart Phone Apps Development

Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)

Department of Computer Science, The University of Hong Kong

Agenda

- Types of Testing
- Testing and Publication of Android Apps
- Testing and Publication of iOS Apps

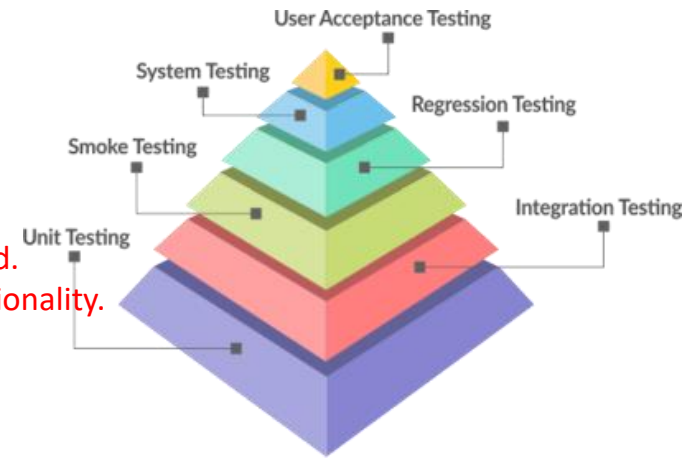
Performance Metrics

Front-end Performance Metrics	Back-end Performance Metrics
<ul style="list-style-type: none">• Application Loading Time• Application Response Time• Screen Rendering• Background Tasks• Interaction with Other Apps• Rate of App Crashes• Battery Usage• Memory Consumption• Compatibility with Various Hardware / Software	<ul style="list-style-type: none">• Server-side Load• API Latency (e.g., Cloud, AI)• Time to First Byte (TTFB)• HTTP Requests• DNS Lookups• Throughput

Client-side Testing

Smoke Testing: Check if a build of the software is stable enough for it to be tested.

Regression Testing: Verify if any recent changes have impacted the existing functionality.



- Functional Testing
 - Ensures that the functions and features of the application work properly
 - Focuses on the functionality of the application
- Input-Output Testing (IO Tests)
 - Part of functional testing
 - A test in which you specify an input and an expected output. If the application returns the same output as expected, the test passes.
- User Acceptance Testing (UAT)
 - Part of functional testing
 - Also known as beta testing or end-user testing
 - The application is tested in the “real world” by the intended audience or business representative.

Client-side Testing

- Non-functional Testing
 - Examines other aspects of how well the application works
 - Tests the performance of different functions in the application
- Compatibility Testing
 - Checks whether your application is capable of running on different hardware, operating systems, applications, network environments or mobile devices

Server-side Testing

- The following 4 types of testing are closely related.
- Stress Testing
 - Tests the performance under extreme conditions. For example, will the server program and the backend database be overloaded when handling a huge number of users at once?
- Capacity Testing
 - Tests how many users the system can handle before performance drops below the permissible levels.
- Volume Testing / Flood Testing
 - Evaluates an application's ability to handle and process large amounts of data without slowing or breaking down or losing any information.
- Spike Testing
 - Measures an application's ability to performance under sudden jumps in workload or volume increase.

Security Testing

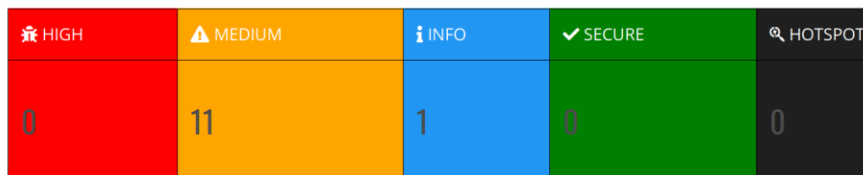
- Penetration testing
 - Common Vulnerabilities in Mobile Apps:
 - Many mobile applications contain vulnerable open-source components that open the door to cyberattacks.
 - Excessive device permissions and a failure to follow secure coding practices also create blind spots that allow malicious adversaries to inject applications with harmful malware and exfiltrate sensitive data.
 - Weak server-side controls, security misconfigurations, and inadequate logging also create vulnerabilities in mobile apps.
 - Penetration testing is also known as pen test
 - A simulated cyber attack against your application to check for exploitable vulnerabilities
 - May involve application protocol interfaces (APIs), frontend and backend servers

Security Testing

● Sample penetration test results by MobSF:



📊 FINDINGS SEVERITY



</> CODE ANALYSIS

NO	ISSUE	SEVERITY
1	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning

☰ APPLICATION PERMISSIONS

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.
android.permission.WAKE_LOCK	normal	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
com.google.android.c2dm.permission.RECEIVE	signature	C2DM permissions	Permission for cloud to device messaging.
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	unknown	Unknown permission	Unknown permission from android reference

☰ APPLICATION PERMISSIONS

PERMISSIONS	STATUS	INFO	REASON IN MANIFEST
NSLocationWhenInUseUsageDescription	dangerous	Access location information when app is in the foreground.	
NSPhotoLibraryUsageDescription	dangerous	Access the user's photo library.	此應用需要訪問圖庫



Testing and Publication of Android Apps

Input Testing for Android Apps – UI / Application Exerciser Monkey

- The Monkey is a program that runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events.
- You can use the Monkey to test applications that you are developing, in a random yet repeatable manner.
- The Monkey includes a number of options, but they break down into four primary categories:
 - Basic configuration options, such as setting the number of events to attempt.
 - Operational constraints, such as restricting the test to a single package.
 - Event types and frequencies.
 - Debugging options.

Input Testing for Android Apps – UI / Application Exerciser Monkey

- When the Monkey runs, it generates events and sends them to the system. It also watches the system under test and looks for three conditions, which it treats specially:
 - If you have constrained the Monkey to run in one or more specific packages, it watches for attempts to navigate to any other packages, and blocks them.
 - If your application crashes or receives any sort of unhandled exception, the Monkey will stop and report the error.
 - If your application generates an application not responding error, the Monkey will stop and report the error.

Input Testing for Android Apps – Basic use of the Monkey

- Step 1: Run your application in Android Studio
- Step 2: Open command prompt and move to Android SDK folder, “platform-tools” sub-folder which contains “adb” command tool (i.e. “adb.exe”) (e.g. the path is “C:\Users\twchim\AppData\Local\Android\Sdk\platform-tools” on my Windows computer and the path is /Users/twchim/Library/Android/sdk/platform-tools” on my Mac computer)
- Step 3: Execute the following command (with ./ for Mac):
 - `adb -e shell monkey --ignore-crashes -p <package name of your project> <number of events count> > <file name of log>`
 - E.g. `adb -e shell monkey --ignore-crashes -p hk.hkucs.hcfcaculator 1000 > test_logs.txt`
 - Parameters:
 - `-e`: to run on emulator / `-d`: to run on device
 - `--ignore-crashes`: If you specify this option, the Monkey will continue to send events to the system, until the specified count is completed.
 - `-p`: the package name is defined afterwards
 - `1000`: number of events count
 - `> test_logs.txt`: file name for storing logs

Input Testing for Android Apps – Sample output with the HCF Calculator example

The screenshot displays the Android Studio environment. On the left, the 'Command Prompt' window shows the following output:

```
arg: "hk.hkucs.hcfcalculator"
arg: "1000"
arg="--ignore-crashes" mCurArgData="null" mNextArg=1 argwas="--ignore-crashes" nextarg="-p"
data="hk.hkucs.hcfcalculator"
// CRASH: hk.hkucs.hcfcalculator (pid 8945)
// Short Msg: java.lang.NumberFormatException
// Long Msg: java.lang.NumberFormatException: For input string: "y2k+r-k, k "
// Build Label: google/sdk_gphone_x86_arm/generic_x86_arm:9/PSR1.180720.117/5875966:userdebug/dev-keys
// Build Changelist: 5875966
// Build Time: 1568436596000
// java.lang.NumberFormatException: For input string: "y2k+r-k, k "
//   at java.lang.Integer.parseInt(Integer.java:615)
//   at java.lang.Integer.parseInt(Integer.java:650)
//   at hk.hkucs.hcfcalculator.MainActivity$clicker.onClick(MainActivity.java:35)
//   at android.view.View.performClick(View.java:6597)
//   at android.view.View.performClickInternal(View.java:6574)
//   at android.view.View.access$3100(View.java:778)
//   at android.view.View$PerformClick.run(View.java:25885)
//   at android.os.Handler.handleCallback(Handler.java:873)
//   at android.os.Handler.dispatchMessage(Handler.java:99)
//   at android.os.Looper.loop(Looper.java:193)
//   at android.app.ActivityThread.main(ActivityThread.java:6669)
//   at java.lang.reflect.Method.invoke(Native Method)
//   at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:493)
//   at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:858)
// CRASH: hk.hkucs.hcfcalculator (pid 9276)
// Short Msg: java.lang.NumberFormatException
// Long Msg: java.lang.NumberFormatException: For input string: ""
// Build Label: google/sdk_gphone_x86_arm/generic_x86_arm:9/PSR1.180720.117/5875966:userdebug/dev-keys
// Build Changelist: 5875966
```

Below the crash log, a green status bar indicates: "Install successfully finished in 575 ms. App restart successful without requiring a re-install."

On the right, the mobile emulator shows the 'HCF Calculator' app. The input field contains 'K12inm', and a keyboard is visible. A red text overlay 'Random inputs' is positioned over the input field. Another red text overlay 'Exception caught - NumberFormatException' is positioned over the crash log.

Input Testing for Android Apps – Sample output with the HCF Calculator example

- The NumberFormatException is due to the following lines, where we did not catch possible exceptions:

```
x = Integer.parseInt(a);  
y = Integer.parseInt(b);
```

- In fact, we should handle improper inputs like the following (though we do nothing in the handling routine):

```
try {  
    x = Integer.parseInt(a);  
    y = Integer.parseInt(b);  
}  
catch (NumberFormatException e) {  
}
```

We use try-catch-finally in Java / Kotlin to handle risky operations. If the risky operation cannot be done, exceptions will be caught.

Input Testing for Android Apps – Sample output with the HCF Calculator example

The screenshot displays the Android Studio environment. The top toolbar includes icons for Run, Layout Captures, Build Variants, and Favorites. The left sidebar shows the Project, Resource Manager, and Structure tabs. The main editor area is split: the left pane shows the 'app' module with Gradle Scripts (build.gradle, gradle-wrapper.properties, proguard-rules.pro, gradle.properties, settings.gradle, local.properties), and the right pane shows the MainActivity.java file. The code in MainActivity.java defines a Clicker class implementing Button.OnClickListener, which parses two text inputs (a and b) into integers and calculates their Highest Common Factor (HCF). The code is as follows:

```
class clicker implements Button.OnClickListener {
    public void onClick(View v) {
        String a,b;
        int x = 0, y = 0, hcf = 0;
        a = textbox1.getText().toString();
        b = textbox2.getText().toString();
        try {
            x = Integer.parseInt(a);
            y = Integer.parseInt(b);
        }
        catch (NumberFormatException e) {
        }
    }
}
```

Below the editor, a Command Prompt window shows the execution of the app using the Monkey tool. The command is: `C:\Users\WIN10 17 8700\AppData\Local\Android\Sdk\platform-tools>adb -e shell monkey --ignore-crashes -p hk.hkucs.hcfcalculator 1000 > test_logs.txt`. The output shows the arguments passed to the app, including the package name and the number of events (1000). The text "No exception caught" is overlaid in red on the Command Prompt output.

On the right, a mobile emulator is shown displaying the HCF Calculator app. The app has a green header with the title "HCF Calculator". Below the header, there are two input fields. The first field contains the text "82 i&ml 'ven". The second field is empty. A red text overlay "Random inputs" is placed over the input fields. Below the input fields is a button labeled "FIND HCF". The output of the calculation is displayed below the button as "HCF = 0".

At the bottom of the screen, a status bar shows the time as 37:32, the network as CRLF, and the encoding as UTF-8.

Testing of Android Apps on Real Device

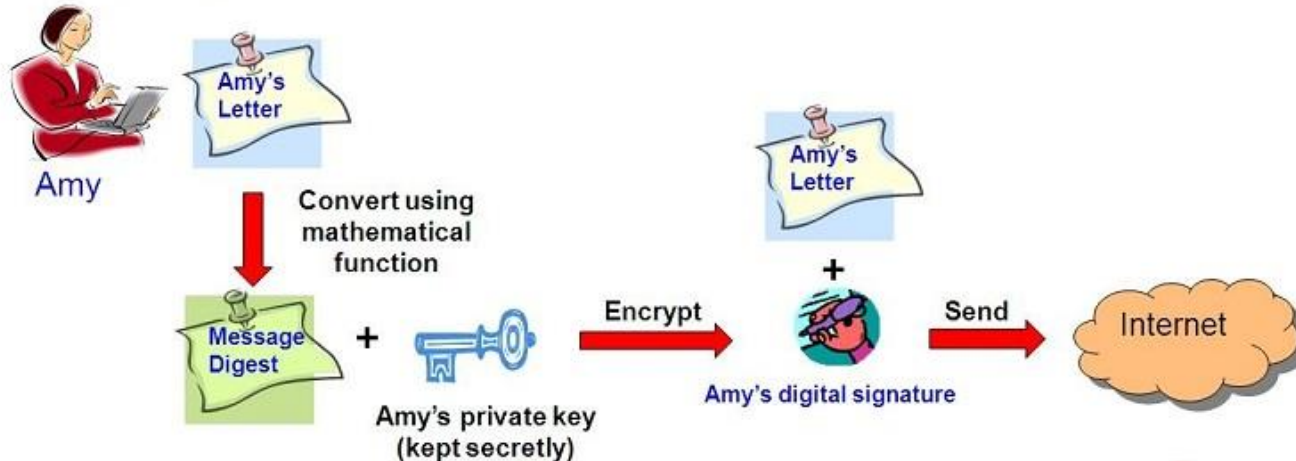
- To test your Android app on a real Android device, you can connect your computer and Android device using a USB cable.
- For some brands of Android devices, you need to first open the "Developer Option". For example in Samsung devices, you need to follow procedures here:
<https://www.samsung.com/tw/support/mobile-devices/how-to-open-close-developer-options/> .
- After that, you should see your real Android device in addition of the created AVDs in Android Studio.
- When you run your app, the APK file will be transferred to your real Android device for installation.

Digital Signature & Certificate

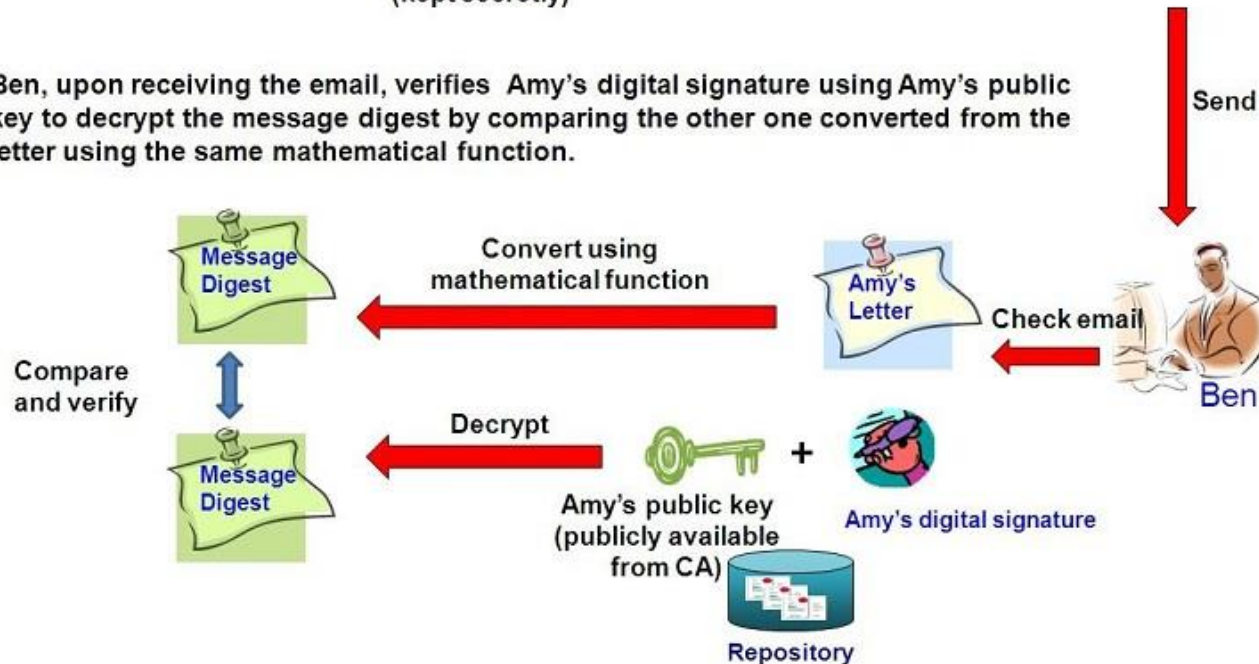
- In public key cryptography, each user has a pair of keys:
 - *public key* – widely distributed and used by other users
 - *private key* – kept secret
- The keys are related mathematically, but the user's private key cannot be easily derived from the widely used public key.
- Incoming messages are encrypted with the *recipient's public key* and can only be decrypted with the *recipient's private key*.
- In digital signature scheme, the digest (e.g., hash) of a message will be signed by the *sender's private key*. The signature can then be verified by anyone with the *sender's public key*.
 - ➔ Authenticity: no one can alter or pretend to be the sender
- Certification authority (CA): a trusted entity that creates, signs and revokes digital certificates that bind *public keys* to user identities.

Digital Signature

1. Amy converts her letter into a message digest by using a mathematical function. She then creates her digital signature by encrypting the message digest using her private key. Her letter, together with her digital signature are sent to Ben via email.

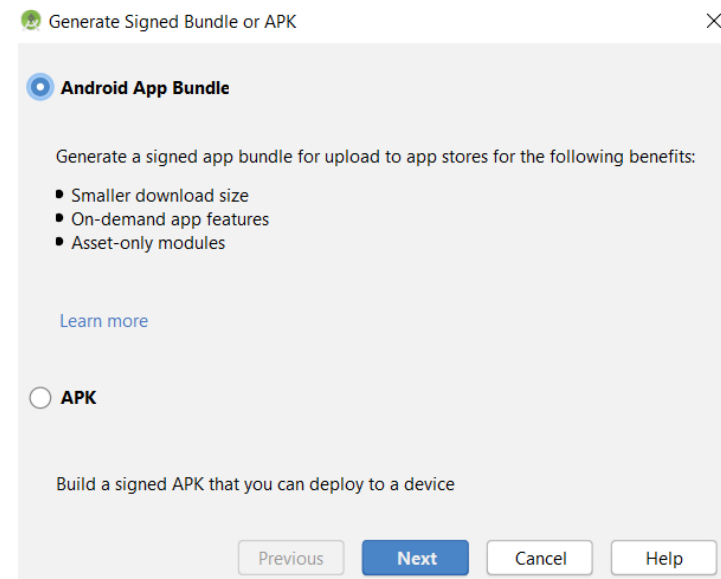


2. Ben, upon receiving the email, verifies Amy's digital signature using Amy's public key to decrypt the message digest by comparing the other one converted from the letter using the same mathematical function.



Generation of Signed APK

- Note that the APK of your project can be found in the "app/build/outputs/apk/debug" folder of the project.
- However, this APK cannot be installed on other Android devices because it is not signed.
- To generate a signed app, you should follow the procedures at this link:
<https://developer.android.com/studio/publish/app-signing>.
- In brief, you can invoke "Build > Generate Signed Bundle/APK" in Android Studio. You will then be asked to pick the following. You need them to sign new versions of the app in the future.
 - a key alias
 - a key password
 - a store password
- A key file will also be generated. You should keep it properly.



Generation of Signed APK

Generate Signed Bundle or APK

Module: app

Key store path:

☒ Create new... ☐ Choose existing...

Key store password:

Key alias:

Key password:

☐ Remember passwords

☒ Export encrypted key for enrolling published apps in [Google Play App Signing](#)

Encrypted key export path: C:/Users/twchim/Desktop

Previous Next Cancel Help

Java Keystores (*.jks, binary files serving as repositories of certificates and private keys) will be generated at the specified location.

New Key Store

Key store path: C:\Users\twchim\Desktop\test.jks

Password: Confirm:

Key

Alias: key0

Password: Confirm:

Validity (years): 25

Certificate

First and Last Name: T.W. Chim

Organizational Unit: CS

Organization: HKU

City or Locality: HK

State or Province: HK

Country Code (XX): 852

OK Cancel

Generate Signed Bundle or APK

Destination Folder:

debug
release

Build Variants:

Previous Finish Cancel Help

Generate Signed APK

APK(s) generated successfully for module 'HCF_Calculator.app' with 1 build variant:
Build variant 'debug': [locate](#) or [analyze](#) the APK.

Publication to Google Play

- You need to provide the following information:
 - Minimum SDK version: Users using devices older than this version cannot install your app.
 - Target SDK version: The SDK version that you think under which your app can perform best.
 - Maximum SDK version: Users using devices newer than this version cannot install your app.
- All these information will be stored in “build.gradle”:

```
defaultConfig {  
    applicationId "comp7506.app1"  
    minSdkVersion 25  
    targetSdkVersion 30  
    ...  
}
```

Publication to Google Play

- The publication site is: <https://developer.android.com/distribute/console/>
- For the first time, you need to sign up and pay \$25 USD (life membership).
- One thing to pay attention is that every time you upload to Google Play, you have to update the `versionCode` and `versionName` in “`build.gradle`”.
 - `versionCode` must be an integer and you should increase it by 1 for each new version.
 - `versionName` can be a double value. The initial value should be 1.0. For minor updates, increase it by 0.1 (e.g., 1.1, 1.2). For major updates, increase it by 1 (e.g., 2.0).

```
defaultConfig {  
    ...  
    versionCode 6  
    versionName "1.5"  
    ...  
}
```

Publication to Google Play

- After login, sign in “Google Play Console” again. you should see the following panel. Then go to “All apps” and then click “Create app”.

The screenshot displays the Google Play Console interface. On the left is a sidebar with navigation links: 'All apps' (selected), 'Inbox' (29), 'Policy status', 'Users and permissions', 'Order management', 'Download reports' (expanded), 'Reviews', 'Statistics', 'Financial', 'Account details', 'Developer page', 'Associated developer accounts', 'Activity log', 'Setup' (expanded), 'Email lists', and 'Pricing templates'. The main content area is titled 'All apps' and includes a 'Create app' button. Below this, there's a section for 'Pinned apps' with a description. A table lists 4 apps, with a 'Filter by' dropdown set to 'All' and a search bar. The table columns are 'App', 'Installed audience', 'App status', 'Update status', and 'Last updated'. The apps listed are MemoMe, Moodle Helper, Omni Steerer, and 來電好幫手.

App	Installed audience	App status	Update status	Last updated
MemoMe cs.hku.hk.memome	0	Production		Dec 16, 2019
Moodle Helper cs.hku.hk.moodlehelper	4	Production		Feb 22, 2020
Omni Steerer com.makerlab.onmi.steer...	5	Production		Aug 21, 2020
來電好幫手 comp7506.gpassignment	24	Unpublished		Sep 26, 2018

Publication to Google Play

- For the initial publication, you will be asked to complete the form below:

Create app

App details

App name

This is how your app will appear on Google Play

Default language

English (United States) – en-US ▼

App or game

You can change this later in Store settings

- ☐ App
- ☐ Game

Free or paid

You can edit this later on the Paid app page

- ☐ Free
- ☐ Paid

Create app

Declarations

Developer Program Policies

- ☐ Confirm app meets the Developer Program Policies

The application meets [Developer Program Policies](#). Please check out [these tips on how to create policy compliant app descriptions](#) to avoid some common reasons for app suspension. If your app or store listing is [eligible for advance notice](#) to the Google Play App Review team, [contact us](#) prior to publishing.

Play App Signing

- ☐ Accept the Play App Signing Terms of Service

To publish [Android App Bundles](#) on Google Play you need to accept the [Play App Signing Terms of Service](#). You will be able to choose your app signing key when creating a release. [Learn more](#)

US export laws

- ☐ Accept US export laws

I acknowledge that my software application may be subject to United States export laws, regardless of my location or nationality. I agree that I have complied with all such laws, including any requirements for software with encryption functions. I hereby certify that my application is authorized for export from the United States under these laws. [Learn more](#)

© 2022 Google · [Mobile app](#) · [Terms of Service](#) · [Privacy](#) · [Developer Distribution Agreement](#)

Cancel

Create app

Publication to Google Play

- You will then be asked to provide further information.

The screenshot shows the Google Play Console interface. On the left is a sidebar with navigation options: Dashboard (selected), Inbox (9), Statistics, Publishing overview, Release (Releases overview, Production, Testing, Open testing, Closed testing, Internal testing, Pre-registration, Pre-launch report), Reach and devices, App bundle explorer, and Setup. The main content area is titled 'Set up your app' and includes a progress bar showing '1 of 11 complete'. Below the progress bar, there are two sections: 'LET US KNOW ABOUT THE CONTENT OF YOUR APP' and 'MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED'. The first section lists steps like App access, Ads, Content rating, Target audience, News apps, COVID-19 contact tracing and status apps, and Data safety. The second section lists 'Select an app category and provide contact details' and 'Set up your store listing'. A blue text box on the right side of the page states: 'You will also be asked to provide the following: - The cover banner image in Google Play - At least 3 screen captures of your app - A short description of your app'.

You will also be asked to provide the following:

- The cover banner image in Google Play
- At least 3 screen captures of your app
- A short description of your app

Publication to Google Play

- Note that you can publish testing versions using this site:
 - Open testing: Create and manage open testing releases to make your app available to testers. Anyone can join your tests on Google Play.
 - Closed testing: Test pre-release versions of your app with your own groups of testers.
 - Internal testing: Create and manage internal testing releases to make your app available to up to 100 internal testers.


Publication to Google Play


🕒 You can then make formal releases.

🔍 Search Play Console

🔗

?

 Omni Steerer



Production

Create and manage production releases to make your app available to all users in your chosen countries. [Learn more](#)

Create new release

Track summary

Active • Latest release: 1.1 • 177 countries / regions • 9 installs

Release dashboard

Releases

Countries / regions

Releases

1.1

View release details

✔ Available on Google Play • 1 version code • Last updated Aug 21, 2020 11:12 AM

Hide summary ^

Version codes

4

Countries / regions

177

Supported Android devices

17,438

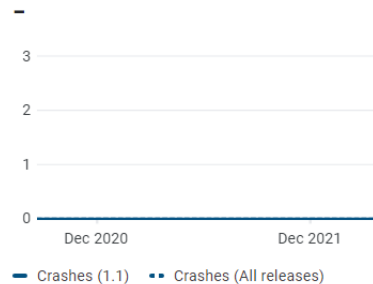
[Go to device catalog](#)

Publication to Google Play

- The site also shows information like release stability, installs and uninstalls, etc.

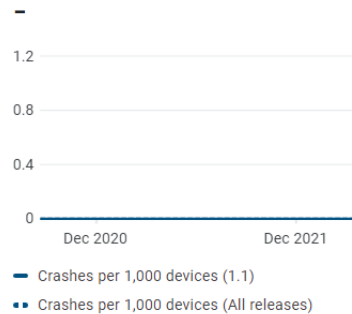
Release stability

Crashes ?

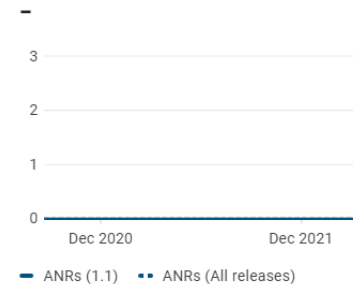


[Explore →](#)

Crashes per 1,000 devices ?

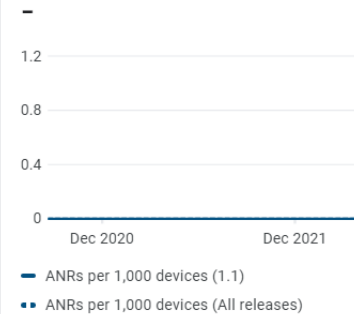


ANRs ?



[Explore →](#)

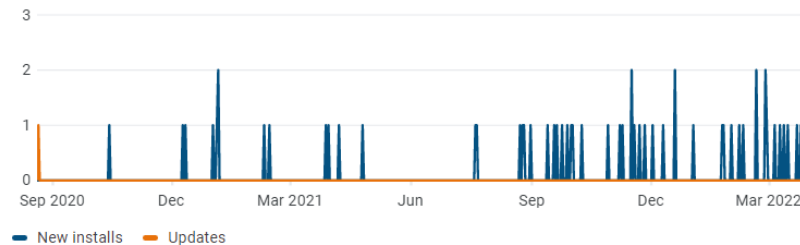
ANRs per 1,000 devices ?



Installs and uninstalls

Install and update events ?

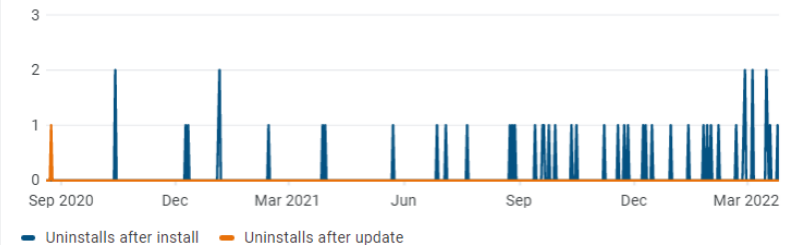
57



[Explore →](#)

Uninstall events ?

48



[Explore →](#)



Testing and Publication of iOS Apps

Testing of iOS Apps on Real Device

- Unlike Android and Windows Phone, Apple has strict security regarding who can distribute apps
- Each app that is compiled to run on a device must be “code-signed”
 - Your app is built and signed by you or a trusted team member.
 - Apps signed by you or your team run only on designated development devices.
 - Apps run only on the test devices you specify.
 - Your app isn’t using app services you didn’t add to your app.
 - Only you can upload builds of your app to iTunes Connect.
 - If you choose to distribute outside of the store (Mac only), the app can’t be modified and distributed by someone else.
- So you need to do two things:
 - Obtain Developer Certificate to “sign” the app
 - Obtain Provisioning Profile which identifies your certificate, device, and app on that device
- Reference:
 - <https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingCertificates/MaintainingCertificates.html>
 - <http://developer.xojo.com/userguide/device-deployment>

Certificate Types and Names Revisited

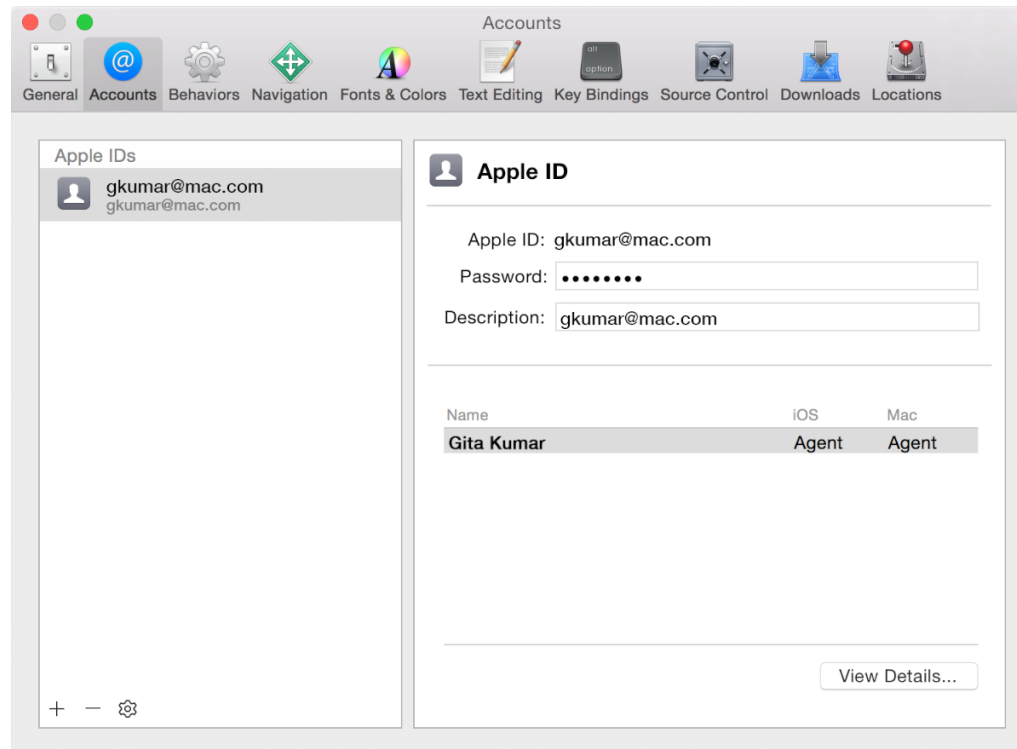
Certificate type	Certificate name	Description
iOS Development	iPhone Developer: Team Member Name	Used to run an iOS, tvOS, or watchOS app on devices and use certain app services during development.
iOS Distribution	iPhone Distribution: Team Name	Used to distribute your iOS, tvOS, or watchOS app on designated devices for testing or to submit it to the store.
Mac Development	Mac Developer: Team Member Name	Used to enable certain app services during development and testing.
Mac App Distribution	3rd Party Mac Developer Application: Team Name	Used to sign a Mac app before submitting it to the Mac App Store.
Mac Installer Distribution	3rd Party Mac Developer Installer: Team Name	Used to sign and submit a Mac Installer Package, containing your signed app, to the Mac App Store.
Developer ID Application	Developer ID Application: Team Name	Used to sign a Mac app before distributing it outside the Mac App Store.
Developer ID Installer	Developer ID Installer: Team Name	Used to sign and distribute a Mac Installer Package, containing your signed app, outside the Mac App Store.

Recall Our iOS Development Team

- Our CS Department has joined the iOS Developer University Program. All our staff and students can apply for an iOS Development certificate for free. This certificate will expire when you leave our Department.
- You can fill in the online application form at:
<https://intranet.cs.hku.hk/iosdev/> and our technical support will send invitation to you to join the Apple Development Team.
- Details of iOS Developer University Program:
<https://developer.apple.com/programs/ios/university/>
- If you want to publish your app to Apple App Store, you need to subscribe the iOS Distribution Certificate by paying US\$99 per year or publish under the name of HKU for free (but the contents of your app will be checked by our computer centre).

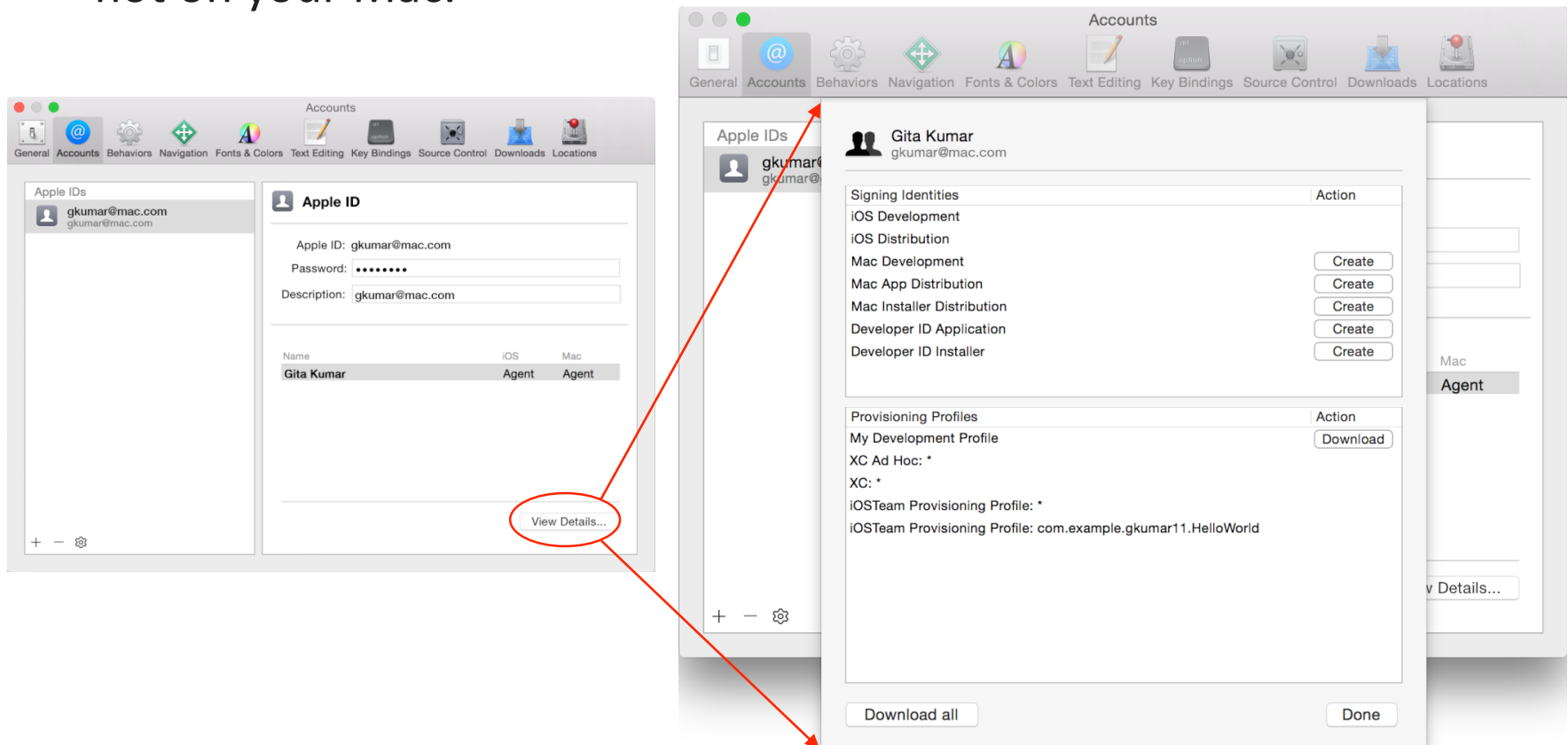
View Certificates in Xcode

- To view account details
 - Choose Xcode > Settings.
 - Click Accounts at the top of the window.
 - Click “+” to add your accounts. After that, you can select the account you want to view, and click View Details.



Create and Download Certificates

- If a **Create** button appears next to a certificate, it hasn't been created yet. If a **Download** button appears next to a provisioning profile, it's not on your Mac.

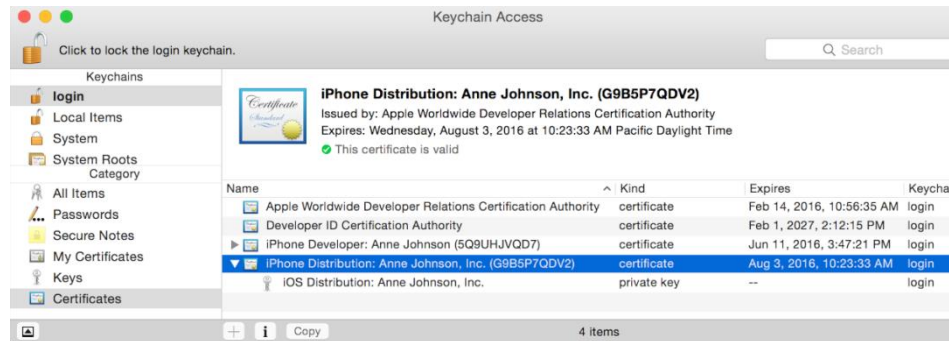


Verifying Your Certificates

- Via **Member Center**
- <http://developer.apple.com/membercenter>

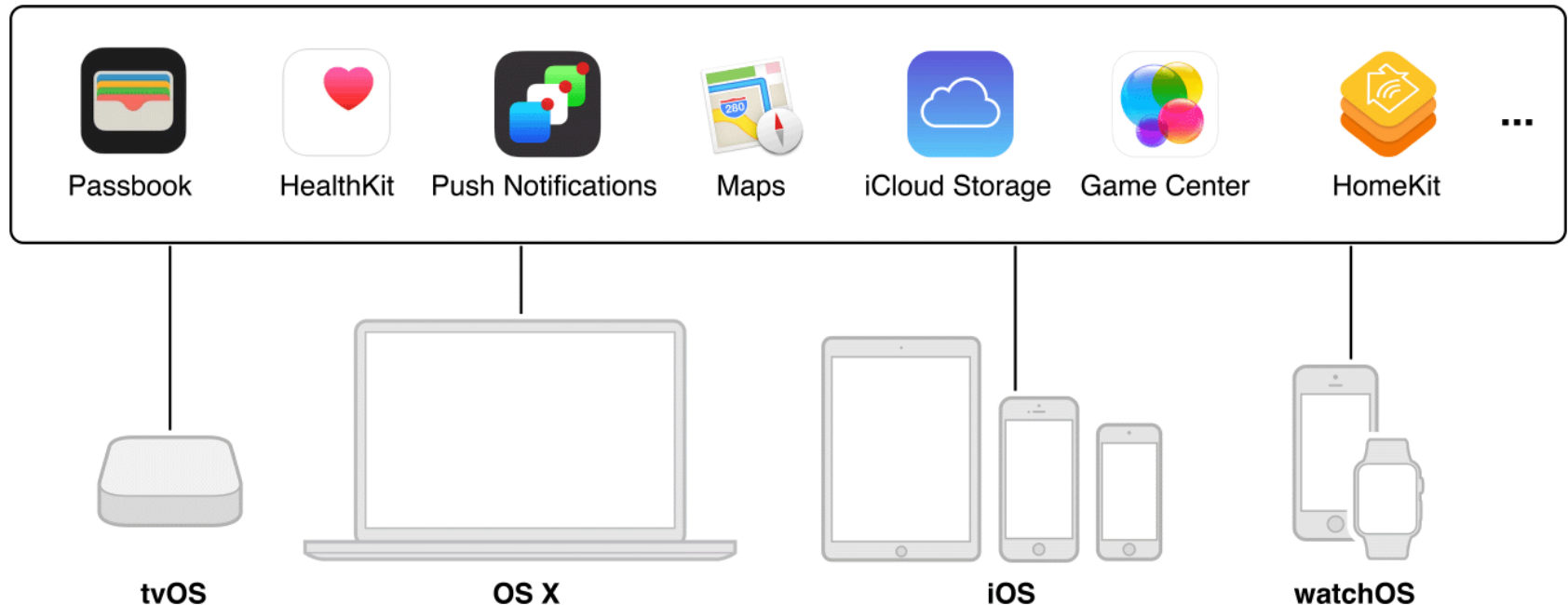


- Or via **Keychain Access**
- **Keychain Access** located in /Applications/Utilities.



Adding Capabilities

- Certain app services require additional configuration in your Xcode project, Member Center, and sometimes iTunes Connect



Adding Capabilities (Cont'd)

- The settings for capabilities are located in the Capabilities pane in the project editor for your target.

Choose your project.

Choose a target.

Click Signing & Capabilities.

Enter text to filter the capabilities.

Double-click or drag capability from the library.

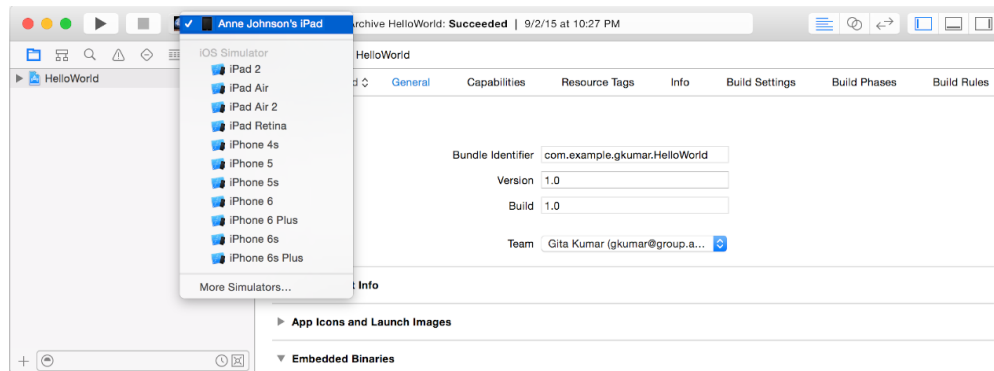
Read a description of the selected capability.

Details:

https://developer.apple.com/documentation/xcode/adding_capabilities_to_your_app

Launching App on Your Devices

- Make sure that you have created your code signing identity and team provisioning profile
 1. Connect your device to your Mac
 2. In the project navigator, choose your device from the Scheme toolbar menu.
- Xcode assumes you intend to use the selected device for development and automatically registers it for you.



3. Click the Run button. Xcode installs the app on the device before launching the app.
4. If a prompt appears asking whether codesign can sign the app using a key in your keychain, click Always Allow.

iOS Review Guidelines

- iOS app submission has to go through a comparatively strict review process
 - iOS App Submission Review Guidelines
 - <https://developer.apple.com/app-store/review/guidelines/>

Chapter 14.



End

2023-2024

COMP7506 Smart Phone Apps Development

Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)

Department of Computer Science, The University of Hong Kong