Wrap Up

CSE 5236: Mobile Application Development
Instructor: Adam C. Champion
Course Coordinator: Dr. Rajiv Ramnath
Essence of a Mobile Device

• The dominant end-user device
• Available to serve everywhere, any time.
• Interwoven into daily life: work, play, study
• Represents and intimately “knows” the user
• Brings in the outside world: sensing, location, communication
Varied Shapes, Sizes, Capabilities

Sources: Apple, Google, Nintendo, Amazon
Mobile App. Development Challenges

- Competitive, fluid vendor landscape (Apple, Android consortium incl. Amazon, RIM, HP) means apps need to be multi-platform for wide adoption
- No “standard” device (iOS, Windows Phone devices?)
- Low bandwidth input (in most cases; what about tablets?)
- Limited screen size (tablets?)
- Unreliability in connectivity and device (network access, power, ambient light, noise, at least for now)
- Integration tradeoffs with cloud and enterprise services
Application Development Support

• Object-Oriented Languages
  – iOS: Swift
  – Android: Java/Kotlin

• Cross-platform frameworks:
  – Flutter, Xamarin, Titanium, PhoneGap, …
  – Scripting languages (JavaScript, Ruby)

• C and C++ (native code)

• Integrated into mobile app dev. frameworks
Android Framework Support

Applications
- Home Screen
- Contacts
- Dialer
- SMS, MMS
- Camera
- Media Player
- Web Browser
- ...

Applications use framework via API calls

Application Framework
- Activity Manager
- WindowManager
- Content Providers
- View System
- Notification Manager
- Package Manager
- Telephony
- Resource Manager
- Location Manager
- ...

Application framework is based on runtime

Android Runtime (ART)
- Core Libraries (Java)
  - ART/Dalvik VM

Core libraries provide Java data structures, etc.
All Android apps run via ART/Dalvik VM

Native Libraries (C/C++)
- Bionic
- WebKit
- SQLite
- Media Framework
- Surface Flinger
- Audio Flinger
- ...

Runs atop native (C/C++) libraries

Exposes API to Android from hardware
Interacts with kernel

Hardware Abstraction Layer
- Graphics
- Audio
- Camera
- Bluetooth
- GPS
- Cellular Radio
- Wi-Fi
- ...

"Talks to" underlying hardware via kernel

Linux Kernel
- USB Driver
- Display Driver
- WiFi Driver
- Bluetooth Driver
- Audio Drivers
- Camera Driver
- Keypad Driver
- Shared Mem. Driver
- More drivers, functions

Google's Kernel Enhancements
- Binder
- IPC
- Power Management
Framework Capabilities, Add-Ons

• Built-in Services:
  – GUI
  – OS services (file I/O, threads, device mgmt.)
  – Graphics
  – Device access (GPS, camera, media players, sensors)
  – Networking
  – Standard language libraries (Java, Kotlin)

• Add-ons:
  – Google Play services (e.g. Google Maps, etc.)
  – Database support (SQLite)
  – Chromium WebView
# Tooling Support

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>Android Support</th>
<th>iOS Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source Integrated Dev. Environments (IDEs)</td>
<td>Android Studio</td>
<td>–</td>
</tr>
<tr>
<td>Proprietary IDEs</td>
<td>IntelliJ IDEA</td>
<td>Xcode</td>
</tr>
<tr>
<td>Testing tools</td>
<td>JUnit, Espresso</td>
<td>XCUnit</td>
</tr>
<tr>
<td>Profiling tools</td>
<td>Android Profiler</td>
<td>Xcode Instruments</td>
</tr>
<tr>
<td>Source code management</td>
<td>Git, Subversion, CVS (for both Android, iOS)</td>
<td></td>
</tr>
<tr>
<td>Software emulators</td>
<td>Android Emulator, Intel HAXM, Genymotion</td>
<td>iPhone Simulator</td>
</tr>
<tr>
<td>Sensor injection tools</td>
<td>Built into emulators (for both Android, iOS)</td>
<td></td>
</tr>
</tbody>
</table>
IDE Support

• Open IDEs – Android Studio (Android)
• Proprietary (Xcode for iOS)
• Testing tools (unit tests, UI tests, test harnesses)
• Performance profiling tools
• SCM integration (Git, SVN, CVS)
• Software emulators
• Sensor injection (GPS, accelerometer)
Thank You

Questions and comments?