MOBILE APPLICATION DEVELOPMENT

+ iOS

Lecture One

Course logistics

- **Lectures**
- Dr. Mohamed Elkholy

- **Main source**
  - **Lecture slides**
  - **Text books:**
    - **Android Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides) 3rd Edition** by Bill Phillips, Chris Stewart, Kristin Marsicano.
    - **Professional Android 4th Edition** by Reto Meier Ian Lake

Course objectives

- Developing Software applications for Mobile Devices
    - Android
    - iOS
- Look at these devices from a Computer Science point of view and see how it impacts software development.
- Understand main challenges and constrains for mobile application development.
- Understand new trends and technologies for mobile application development.

Course contents

- Mobile Device Architecture
- Android Architecture
- Android Studio
- Android Layout
- Android Activities
- Android Native Components
- Android Services
- Android Data Management
- iOS Architecture
- Flutter
- Dart language

Software application models

- **Desktop model - stand-alone application**
  - on a single-user machine.
  - this could be a console window or a GUI app.

  No need to Internet or network connections.

- **Client-server / web model**
  - An application broken into several pieces across multiple machines and is accessible to many users; GUI written in a markup language of some kind. “HTML”
  - Should be connected with a Network.

- **Mobile model**.

Mobile Devise Limitations

- Screen real estate - You no longer have a **huge screen** to show lots of information on.
- Interface - You no longer have a **mouse and keyboard**, You only **fingers**.
- Platform differences - Writing for **different browsers** is one thing. Writing for **different mobile platforms** is a whole other one.
- Platform guidelines - Along the same lines, we now have more rules to follow.
- App distribution - This is a completely different model than what we're used to with the other models. Larger size od data are sourced out side the user application.

Two Main Operating Systems “platforms”

- **iOS**
  - Developed by Apple, and it runs exclusively on Apple products.
  - Apple provides iOS developers with many native tools and libraries to develop iOS applications, and, though you do not have to be enforced to use Apple’s development tools to create your apps, you just need to have a mac running OS X to build your application.

- **Android**
  - Thousands of different devices of all shapes and sizes
  - Open Source, available to everyone.
  - Android is based on the Linux kernel, and Google releases the source code for Android as open source.

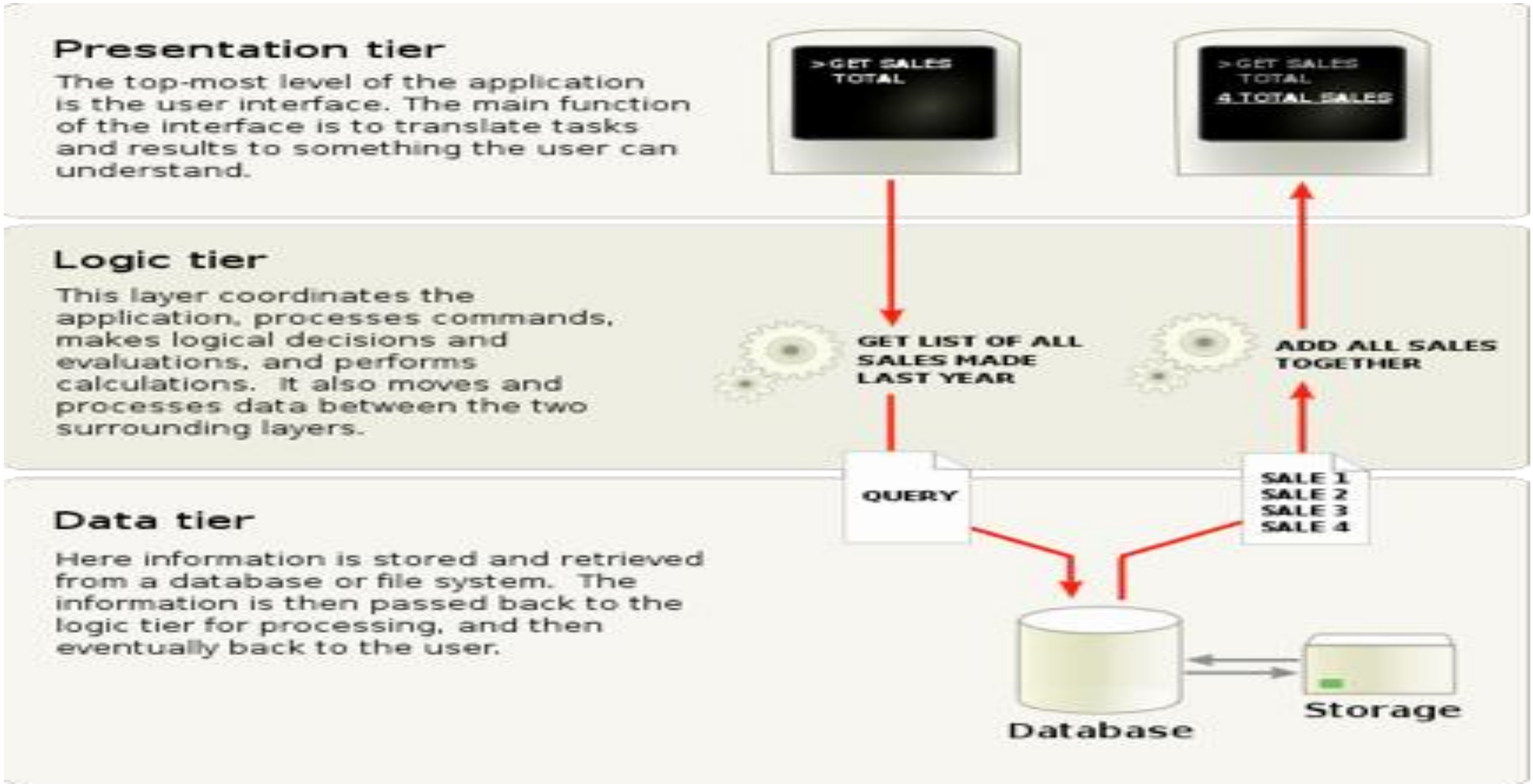
Programming languages and frameworks for mobile application development

- **iOS** apps are developed using the Xcode environment integrated into Mac OS X “operating system for apple machines” and iOS, in Objective C, Swift, C and C ++.

- **Android** apps are developed using **Android Studio** using (**Java and Kotlin)** languages.

- **Cross-platform frameworks** allow to reduce labor costs and accelerate the release of software in case you need your software to support multiple platforms. simultaneously. You should take into account the features of the selected framework.
- **React Native**
- **Flutter**
- **Xamarin**
- **Adobe PhoneGap**

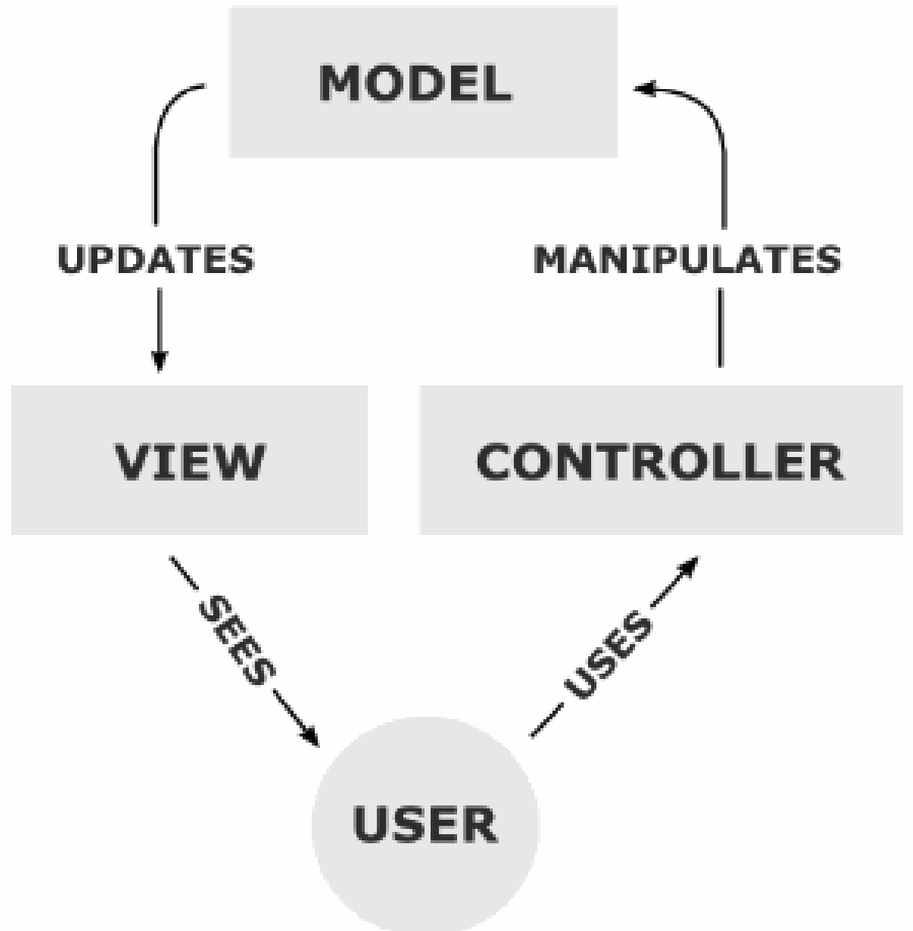
- Both Operating systems follow a three-tier design philosophy
- More specifically: MVC = Model / View / Controller



- The concepts of the three-tiered architecture apply to many design scenarios
  - Keep the presentation separate so it's lightweight, easier to maintain, and can be tested separately
  - Keep the logic separate so you can change the logic as needed without having to change the presentation too much
  - Keep the data separate because you should NEVER build a system based on the current data values
- How does it apply to: websites? Java apps?

MVC pattern map

- Identifies what the user is asking for
- Loads a particular resource
- Displays the pertinent info about that resource back to the user



Controllers

- The role of the controller is basically traffic cop
- It takes the request from the user and (with the assistance of the server and routing rules) turns it into a method call of sorts
- It finds the appropriate model to load
- It finds the appropriate view to load
- It returns the whole thing back to the user

Models

- The model is the representation of the data
- This may or may not be directly linked to a database (but often is in larger apps)
- A model is often translated directly into a DB table, with the columns as its attributes
- Often contains relationship rules (a Student has many Classes, for instance)

View

- The closest thing to what you've been dealing with so far is the view
- View may be a HTML template that will be populated with the appropriate data from the loaded model.
- All UI components go here may also uses PHP.

Front end and back end in mobile apps

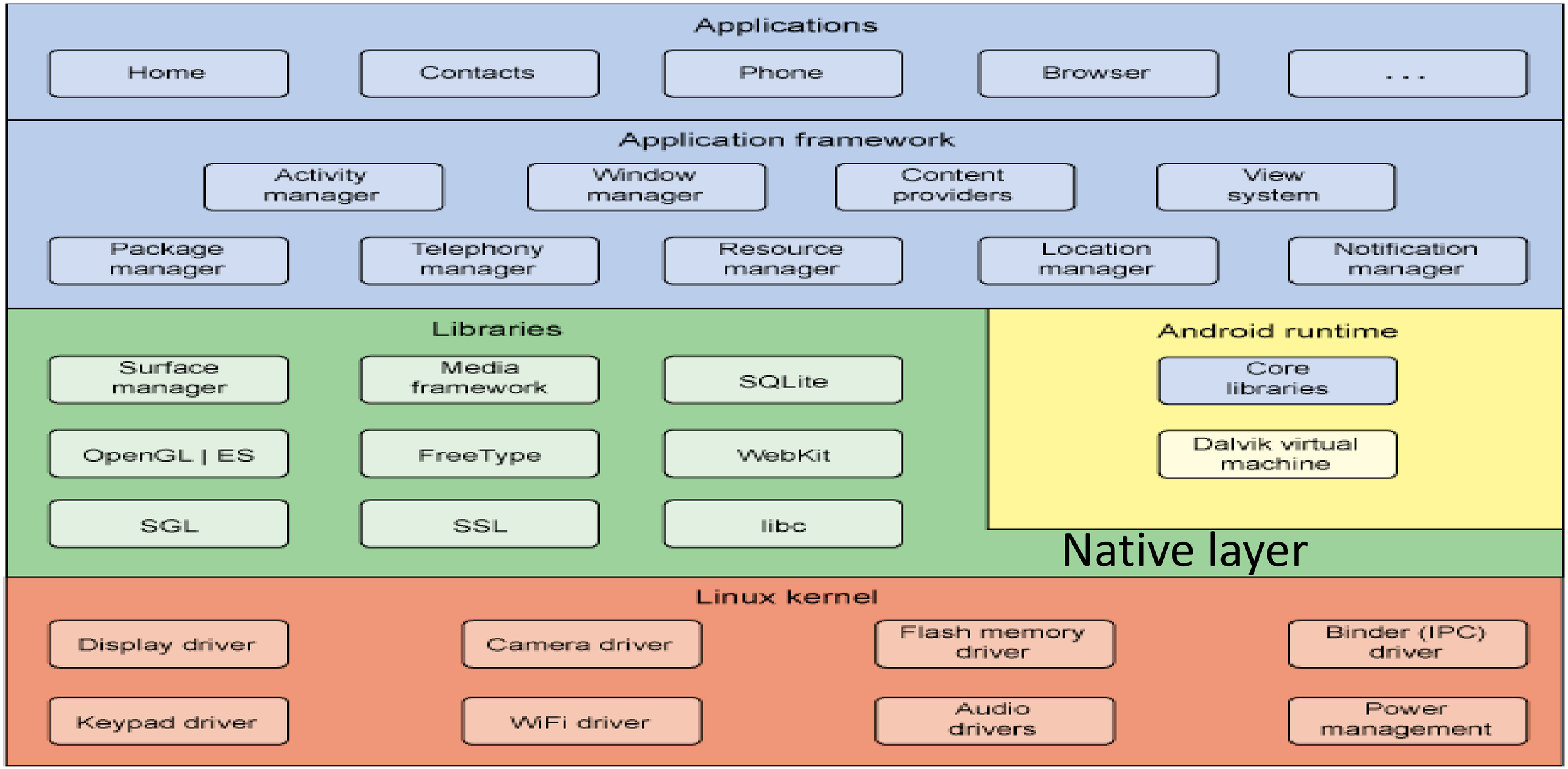
Back end

- It is a piece of software that runs on remote machines called servers.
- It can be accessed through the Internet.
- The backend is not meant to be used by humans directly but rather by other applications (frontend apps).

front end

- It is the piece of the application that the user can see and interact with.
- All the apps which you can download from the App Store (if you are an iOS user) or Google Play (if you use the Android system) are frontend apps.

Android Architecture

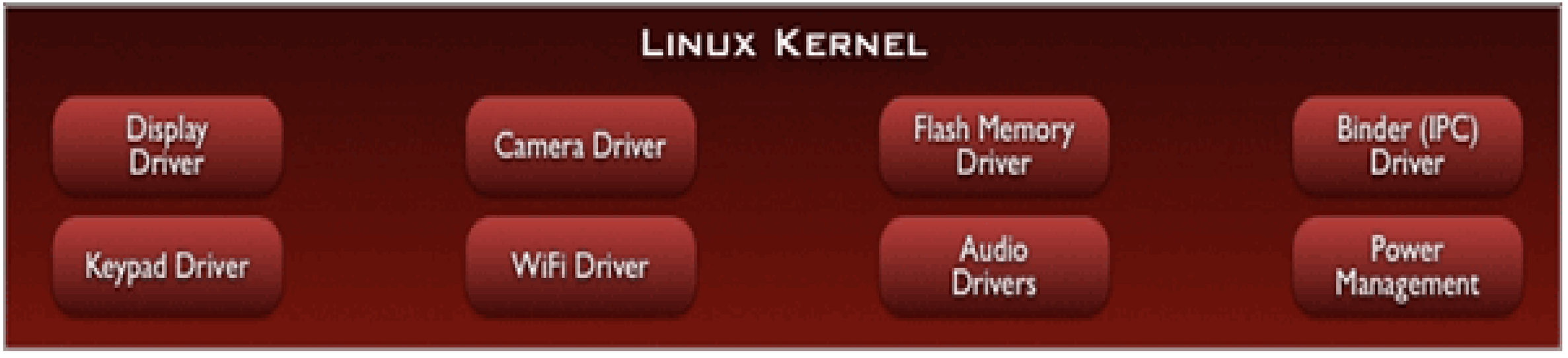


Layers in Android Stack

- Linux Kernel Layer
- Native Layer
- Application Framework Layer
- Applications layer

1st Linux Kernel Layer

Bottom layer of android operating system is Linux kernel. Linux Kernel provides the basic system functionality such as process management, memory management and device management like camera, keypad, display etc.



2nd Native layer

The second layer in the Android architecture includes.

Libraries and android runtime

- Libraries
  - Set of instructions that guide the device in handling different types of data.
  - For instance, the playback and recording of various audio and video formats is guided by the Media Framework Library.
- The Android runtime
  - provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.
  - Includes Dalvik Virtual Machine.

Libraries

SQLite	This library is used to access data published by content providers and includes SQLite database management classes
SSL	This is used to provide internet security
OpenGL	OpenGL is used to provide Java interface to the OpenGL/ES 3D graphics rendering API.
Media framework	It is used to provides different media codecs which allow the recording and playback of different media formats
WebKit	It is the browser engine used to display internet content or HTML content
Web browser	Based on the open-source WebKit layout engine, coupled with Chrome’s V8 JavaScript engine supporting HTML5 and CSS3.

Android runtime layer

- Includes a set of core Java libraries that allows Android application programmers build their apps using the Java programming language.

- Includes the Dalvik Virtual Machine.

- It is optimized for low memory requirements. It has been designed to allow multiple VM instances to run at once. Relies on the underlying OS for process isolation, memory management and threading support. Operates on DEX files.

3rd Application Framework Layer

Application framework manages the basic functions of android device such as resource management, voice call management etc. Application Framework layer provides many higher-level services to applications in the form of Java classes .

Activity Manager: Manage application written in Android studio

Content Providers: Manage the data sharing between applications.

Telephony Manager: Manages all voice calls. We use telephony manager if we want to access voice calls in our application.

Location Manager: Location management, using GPS or cell tower

Resource Manager: Manage the various types of resources we use in our Application

4th Application layer

The applications are at the topmost layer of the Android stack.

User of Android devices interact with this layer (for **basic functions, such as making phone calls, accessing the Web browser, accessing applications** etc.).

The layers further down are accessed mostly by developers, programmers.

- Examples
- SMS client app
- Dialer
- Web browser
- Contact manager

Types of mobile Apps

- Generally, mobile app developers can build an app in one of these three categories:

- Native apps — Coded in a language that's supported natively by a specific device's operating system. (Example: native iOS app vs. native Android app).

- Hybrid apps — Cross-platform development. Apps are coded in one language that can run on multiple platforms.

- Progressive web applications (PWA) — A lightweight app that runs in the URL of a device's web browser. It looks and feels like a mobile app, but it's not delivered natively on the device.