Android App Anatomy（应用结构）

Manifest /src/main/AndroidManifest.xml 元信息、权限、组件	
Source files .java / .kt	
Layout files res/layout/*.xml UI 布局	
Images res/drawable/ 图标/图像资源	
String constants res/values/strings.xml 多语言字符串	
Build script build.gradle 项目构建配置 minSDK / targetSDK、dependencies 版本、插件等	

/app/build.gradle	模块级（App依赖文件
/build.gradle	顶层项目构建文件
类型	📌 dependencies 部分说明：
implementation	主项目运行时依赖（UI库、网络库等）
testImplementation	单元测试相关依赖
androidTestImplementation	仪器测试依赖（如 Espresso）
compileOnly / api	特定场景下使用的依赖控制选项

AndroidManifest.xml	
组件声明（Activities、Services、BroadcastReceivers）	<activity android:name=".MainActivity" />
应用权限（如网络、定位等）	<uses-permission android:name="android.permission.INTERNET"/>
应用入口点定义（含 MAIN 和 LAUNCHER intent-filter）	<intent-filter> <action android:name="android.intent.action.MAIN"/>
设置应用级别配置（图标、主题、最低 API 等）	<application android:icon="@mipmap/ic_launcher" />

res/layout/*.xml
存放于 res/layout/ 文件夹中
描述界面中有哪些 UI 元素（Widgets），它们的 **属性、位置、行为**
元素类型（如 Button, TextView）
布局尺寸、边距、排列方式（如 layout_width, layout_height, margin, gravity）
内容（text, src, hint 等）

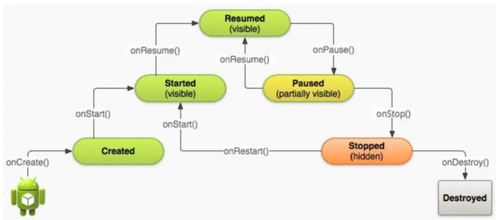
> Android 应用由 结构化文件系统 + 框架驱动的生命周期 + MVVM 架构 组成；Android 不只是操作系统，它还是：
> - ✅ 一个 **框架**（Framework）
> - ✅ 一个 **插件架构**（Pluggable Architecture）
> - ✅ 一个 **软件生态系统**（Ecosystem）

Android applications	architecture	：MVVM
层级	功能	常见技术
Model	数据层	Room, Retrofit
ViewModel	状态管理	LiveData, Lifecycle
View	UI 显示层	Activity, Fragment, Layout

技术	解释	Design Tools & Techniques
Abstraction	Hide details to focus on essentials	抽象：忽略细节，聚焦核心
Modularity	Break system into manageable parts	模块化：分解系统，便于管理
Encapsulation	Limit access to internal state	封装：保护内部结构
Hierarchy	Organize components in layers	分层：分级组织组件结构

分类	描述
插件（Plugins）	增强用户体验的扩展模块，可选安装。类似 microkernel 架构。
框架（Framework）	提供开发"骨架"，只需扩展。好处是**控制反转 Inversion of Control，IoC**（如 onClickListener）
生态系统（Ecosystem）	由多个开发者共享的模块集合，如 Jetpack、NPM、Maven。

Android Activity Lifecycle



分层结构（Layered Architecture）	Android Framework 架构
Presentation	UI 展示层（Activity、View）
Application	控制与服务层（App logic）
Domain Logic	业务逻辑层（模型/数据）
Data Access	数据访问层（数据库、网络）
特点：层与层之间只能"上下相连"，不跨层通信	有利于维护与模块复用

Fitts's Law	（费茨法则）
Time to tap = Distance / Size	点击所需时间 ∝ 距离 ÷ 目标大小
→ Put common buttons where the thumb reaches	把常用按钮放在容易触达的地方
→ Make buttons big enough to be tapped	按钮要足够大，避免误触

UX antipattern

🎯 名称	❌ 问题描述	✅ 正确做法
Misdirection	用户被引导点击非他们本意的选项，例如：消极操作按钮比积极操作更突出，或表达方式故意绕弯	Use clear, neutral wording. Avoid double negatives and misleading design.
Roach Motel	用户很容易"进入"某个状态（如订阅、注册），却很难"退出"或取消（如隐藏的取消按钮）	Provide clear and accessible exit options like "Unsubscribe", "Logout", or "Delete Account".
Bait and Switch	用户以为点击是某个功能（如"播放歌曲"），结果弹出广告或付款界面	Keep UI elements consistent with their intended function and avoid changing behavior unexpectedly.
Trick Question	通过文字游戏混淆用户选择，如勾选项表述为否定句	Use straightforward, intuitive language aligned with user expectations.
Inconsistent UI Elements	同样的功能按钮使用了不同格式或样式，容易误导用户	Ensure consistent styling and semantics across similar elements.
No Hierarchy of Actions	所有按钮颜色/大小相同，用户不清楚哪个操作是主次	Use contrast, color, and placement to distinguish primary, secondary, and dangerous actions.
Bad Feedback / Error Messaging	表单错误提示太模糊，或登录错误提示暴露过多信息（安全风险）	Provide clear, helpful, and secure feedback without overexposing system behavior.
Contrast & Color Issues	图标、导航栏颜色对比不足，影响可读性和可访问性	Ensure strong visual contrast; follow WCAG accessibility guidelines.
No KISS	KISS = Keep It Simple, Stupid；界面信息过多、逻辑混乱	Follow KISS: Keep It Simple, Stupid. Reduce unnecessary clutter.
No Layout Testing	不同屏幕或设备下布局错乱	Use responsive design tools and test on multiple devices/resolutions.
No Undo	No way to reverse an action	Always provide a way to undo destructive actions or confirm them in advance.

- ✅ Trick Question：用户以为勾选是"订阅"，实际上是"拒绝订阅"
- ✅ Roach Motel：注册很容易，但找不到退订按钮
- ✅ Bait and Switch：商品标明"免费"，点击却变成"付费"

View	组件功能	使用方法
ListView	单列列表，可绑定简单数据	使用 ArrayAdapter
GridView	网格列表	使用自定义 Adapter
RecyclerView	可自定义布局、复用视图、高性能滚动	需要实现 ViewHolder 和 Adapter，支持 GridLayoutManager、LinearLayoutManager 等
ScrollView	滚动的容器，适合内容较长页面	包裹一个子布局（如 LinearLayout），用于滚动文本、表单等

Adapter 机制 概念	（AdapterView 或 RecyclerView） 说明
Adapter	数据与视图的桥梁，如 ArrayAdapter
ViewHolder	RecyclerView 中的元素容器，减少 findViewById 频率
Binding	将数据绑定到视图
LayoutManager	控制 RecyclerView 中元素的布局方式
XML：	推荐方式，结构清晰，可视化强
Java：	适用于动态生成、代码控制布局

Widget 控件类型	功能	场景
TextView / EditText	展示或输入文本	表单、昵称、搜索框（News Profile 页面输入）
Button / ImageButton	操作触发器	提交、登录、拍照（Demo_Buttons）
CheckBox	多选	"我同意多个条款"
RadioButton	单选	✅"选一项使用 RadioButton 或 Spinner"
Switch / ToggleButton	状态切换（开/关）	启用通知、切换功能
SeekBar / Slider	选定范围值	✅"赛车游戏中调节灵敏度"
Toast / Dialog / Popup	反馈、信息提示	Toast 自动消失，Dialog 需要确认
Spinner	下拉选择器（只选一项）	✅模拟题："Spinner 和 RadioButton 可用于单选"

android.view.View

ViewGroup	布局容器特点	适用场景	示例题
LinearLayout	元素按垂直或水平顺序依次排列，可设置 layout_weight 控制空间分配	表单、设置项、一列输入框等	✅"在此布局中，元素逐一排列。"
RelativeLayout	元素相对父容器或其他元素位置排列	复杂对齐需求较多时	button 位置基于 editText
GridLayout / GridView	元素按网格排列，支持横向或纵向滚动	图像墙、键盘、九宫格菜单	✅"宠物图库切换大图小图视图：用 RecyclerView + Grid/List layout"
ConstraintLayout	高级布局，支持任意方向约束，适用于复杂 UI	更强控制、更少嵌套	Lab 示例使用
FrameLayout	所有子视图层叠，显示在彼此之上	用于堆叠组件、浮动按钮	Dialog 内嵌内容常见

UI Design Tips

🧭 指南（Guide）	✅ 建议（Design Tip）	
元素需清晰可见	使用颜色对比、留白、字号大小使元素脱颖而出	Make elements clearly visible using contrast, whitespace, and size
减少用户输入	利用自动完成、语音输入、选择器、传感器等方式降低输入负担	Minimize user input using autocomplete, voice input, pickers, sensors
使用一目了然的图标或文字	图标 + 文本组合最易理解，也更易传达意义	Use intuitive icons or icon+text combinations for clear communication
避免多个按钮混杂	每个界面尽量只提供 1~2 个主要按钮（KISS 原则）	Avoid cluttering with too many buttons (Follow KISS: Keep It Simple, Stupid)
控制元素位置	把主要操作按钮放在右下角等用户拇指易触区域，符合交互设计	Place key action buttons in reachable thumb zones (e.g., bottom-right corner)

Design

Design is achieving goals within constraints.	设计是在限制条件下实现目标的过程。
Goals: Who is it for? Why do they need it?	目标：为谁设计？为什么？
Constraints: time, cost, platform, materials	限制：时间、成本、平台、材料
Design is about making trade-offs.	设计就是不断权衡取舍。

维度	解释	E³ Framework
Effective	Gets the job done—for the **user**!	有效：完成用户的任务！
Efficient	Minimal steps, clear, no distractions	高效：最少的操作，无干扰
Exciting	Fun, rewarding, emotionally engaging	有趣：愉悦、有吸引力的用户体验

类型	Design vs Engineering vs Research	
Research	Understands why/how things work	研究：探索问题本质与机制
Design	Creates solutions to meet goals	设计：提出解决方案
Engineering	Builds working implementations	工程：将方案落地实现

Trade-off	
Performance vs. Reusability	性能 vs 可重用性
Cost vs. Robustness	成本 vs 稳定性
Rapid Dev vs. Feature Complete	快速开发 vs 功能完备
Backward Compatibility vs. Readability	向后兼容 vs 代码可读性

原则		Mobile UX Best Practices
✅ KISS	Keep it simple, stupid!	保持简单，一屏一事
✅ Cross-device	Seamless experience	跨设备一致性
✅ Clear Navigation	Intuitive and consistent	导航直观一致
✅ Less Input	Use autocomplete, sensors	减少输入，调用自动填充或传感器
✅ Thumb Zone	Design for one-hand use	设计考虑拇指区域

Benchmarking 类型	性能基准测试描述
Specification-based	有统一标准，开发者实现逻辑（关注"业务问题"）
Kit-based	提供整套实现和数据包（关注"统一环境"）
Hybrid	两者结合，允许少量自定义
Synthetic	使用脚本模拟用户行为
Microbenchmark	针对特定函数或模块进行性能测量
Kernel benchmark	对核心功能的子系统进行测量
Application benchmark	用实际应用来评估性能

类型	Profiling	特点
Instrumentation	插桩	精确度高，适合**短操作**，但会引入性能开销
Sampling	采样	精度较低，但**性能影响小**，适合长期采样

类型	阶段	监控与性能工程（SPE）
Profiling	开发期	开发阶段用于分析和优化性
Monitoring	生产期	上线后用于追踪运行状态

Espresso Feature	
精确匹配	可根据 ID、Text、State 查找 UI 元素
自动断言	验证页面状态或组件值是否符合预期
支持手势操作	支持滑动、点击、长按等用户操作
意图校验（Espresso-intents）	支持 Activity 切换与意图传递验证
列表控件测试（Espresso-lists）	支持 ListView / RecyclerView 等动态列表测试
录制与回放测试（Record and Replay）	自动记录交互生成测试代码，可重复执行
测试脚本可编程扩展	可以修改自动生成的测试代码
支持断言与状态检查	使用 assert 检查视图文本、是否可见等
支持多层 UI 层级定位	能深入 Activity → Layout → View 层级
适配异步任务（IdlingResource）	等待异步操作完成再继续测试

Testing

Software Testing	Strategies	特点
Big Bang Testing	大爆炸测试	一次性测试整个系统，适合小项目；缺点是难以定位错误。
Incremental Testing	增量测试	按模块逐步集成和测试，易于定位错误，开发中常用。
Top-down	自顶向下测试	使用 stub 替代底层模块，早发现架构问题
Bottom-up	自底向上测试	使用 driver 替代高层模块，适合已有底层实现

项目	Drivers（驱动模块）	Stubs（桩模块）
用于	测试"底层"模块是否能正确工作	测试"顶层"模块调用是否正常
作用	模拟上层模块，向下传递数据并触发调用	模拟下层模块，返回固定响应或数据
数据流	"向下"传递，执行真正底层代码	"向上"模拟，跳过底层实现
举例	测试数据库模块：用 driver 模拟 UI 输入调用	测试 UI 模块：用 stub 模拟数据库响应

Performance Testing	Types	特点 / 目标
Load Testing	负载测试	测试在"预期负载"下的系统表现（正常使用压力）
Stress Testing	压力测试	测试系统超出最大承载时的表现（宕机等）
Endurance Testing	耐久性测试	长时间高负载下的稳定性测试
Spike Testing	峰值测试	模拟瞬间涌入或减少用户，检测系统弹性
Capacity Testing	容量测试	确定系统能承载最大用户或任务量

Step	测试流程总结	Testing Flow Summary:
1	明确测试目标（功能 / 性能 / 用户行为）	Define test goals (functional, performance, user interaction)
2	编写测试脚本或设计测试场景	Write test scripts or design test cases/scenarios
3	执行测试（手动或自动）	Execute tests (manual or automated)
4	收集结果并分析	Collect results and analyze
5	生成报告并进行回归测试	Generate reports and perform regression testing

TestCases	概念	Example
Equivalence Class	等价类	每个变量的有效/无效分类
Marginal Value	边界值	0, 3, 3.1, 6 小时数边界测试
Valid Test Cases	有效用例	TC1~TC3，逻辑正常应通过
Invalid Test Cases	无效用例	TC7~TC9，应抛出提示
Coverage Strategy	覆盖策略	每类至少测试一次；覆盖所有输入组合类型

UI Testing	测试类型	特点
Manual UI Testing	手动 UI 测试	人工操作并比对结果
Automated UI Testing (Espresso)	自动 UI 测试	使用代码模拟用户行为并验证结果
Record and Replay	录制回放测试	自动记录操作生成测试脚本
Scripted Testing	脚本测试	按脚本预定义路径测试，稳定可靠
Exploratory Testing	探索式测试	测试员主观探索界面，发现潜在问题
UX Testing	用户体验测试	根据用户主观体验反馈界面好坏

测试类型	特点 / 区别
Alpha Testing	✅ 在开发方或公司内部进行✅ 测试人员是 QA 或开发人员✅ 可访问源代码✅ 发现大多数 bug、崩溃等问题
Beta Testing	✅ 在真实用户环境中进行（非开发者）✅ 收集用户反馈和使用行为✅ 通常是"封测"或"公开测试"✅ 评估真实可用性、接受度和用户体验
White Box Testing ✅	✅ 测试人员**了解代码逻辑**✅ 测试语句覆盖、分支覆盖、条件覆盖等✅ 开发人员常用，配合单元测试
Black Box Testing ✅	✅ 测试人员**只关注输入和输出**，不关心代码内部✅ 用于 UI 测试、功能测试✅ 真实模拟用户操作
Grey Box Testing	✅ 测试人员**了解部分内部结构**✅ 综合功能 + 部分结构性测试（如接口层）

Touch	触摸事件	补充说明
Touch Mode	触摸模式（Android 中的输入模式）	用户使用手指输入时，不需要焦点选中（Unlike D-Pad）
MotionEvent	触摸事件类	包含 ACTION_DOWN、ACTION_MOVE、ACTION_UP 等
Pointer	指针（每根手指）	每个 pointer 有一个 **固定不变的 ID** 和可能变化的 **索引（index）**
事件顺序	指的手指按下/移动/离开顺序	示例顺序：DOWN → POINTER_DOWN → MOVE → POINTER_UP → UP
ACTION_INDEX & ID	多点触控下的手指识别	getActionIndex() 获取变动手指索引，getPointerId() 获取其 ID
标识符（ID）	Pointer ID	✓ 按下时分配，**整个手指生命周期不变**，用于唯一标识
索引（Index）	Pointer Index	❗ 在当前事件中的位置，**每次事件中可能不同**，用于访问坐标

Test ID	ID4
Title	Use Case4: Order food
Setup	The food alarm has gone off The user is in the Order Activity for a specific Pet
Data	Pet is "Lucy". Past order is "2023-02-15, 7kg dry food". Order was fulfilled by PetSmart (URL)
Procedure	The user clicks on renew order. The user clicks confirm new order. The user clicks on pay. The user confirms use of saved credit card.
Expected Result	Confirmation Toast appears that the order was placed. Data is updated in the pet's profile. Order status is set to "pending".

Gestures	类型	类 / 工具	说明
Basic Touch Gestures	基础手势：点击、滑动、长按等	GestureDetector	单点触控手势
Pinch/Zoom	双指捏合、缩放	ScaleGestureDetector	多点触控手势（如缩放地图）
Unistrokes / Graffiti	单笔迹输入 / 类似 Palm Graffiti 的笔势输入	Unistrokes.java，自定义手势库	自定义符号识别
Shape Writing	连续手写输入（如 Swype）	Shape Writing Algorithm	通过形状与轨迹识别词语
GestureListener	手势监听接口	SimpleOnGestureListener	重写如 onFling、onLongPress

手势类型	手势识别方法
单指点击	`onSingleTapConfirmed()`
滑动（Swipe）	`onFling()`
长按（Long Press）	`onLongPress()`
拖动（Drag）	`onScroll()`
双指缩放（Pinch）	`onScale()` in `ScaleGestureDetector`

动作常量	解释	触发条件	多点触控特性
`ACTION_DOWN`	第一根手指按下	第一次触摸屏幕时触发	表示"开始触摸序列"
`ACTION_POINTER_DOWN`	其他手指按下	第二根或更多手指按下时	多点触控的额外入口点
`ACTION_MOVE`	手指在屏幕上滑动	任意手指发生位移时都会触发	会持续回调
`ACTION_POINTER_UP`	其他手指抬起	非主手指抬起	主手指未抬起时
`ACTION_UP`	最后一根手指抬起	结束整个触摸事件流	与 `DOWN` 配对
`ACTION_CANCEL`	事件被系统取消	系统中断，如来电	非人为主动抬起

实用 API/类汇总	用途
`MotionEvent`	获取触摸信息（位置、压力、多指等）
`GestureDetector`	单指手势识别（滑动、点击）
`ScaleGestureDetector`	多指手势识别（缩放）
`SensorManager`	管理各种传感器
`SensorEventListener`	监听传感器数值变化
`Sensor`	表示单个传感器

类型		示例
Motion Sensors	运动传感器	加速度计（Accelerometer）、陀螺仪（Gyroscope）
Position Sensors	位置传感器	磁力计（Magnetometer）、重力传感器
Environment Sensors	环境传感器	温度、湿度、光照、压力（Barometer）

✅ 常用 Manifest 权限（Manifest Permissions）

权限	中文说明	用途举例
`INTERNET`	网络权限	网络请求（如下单、API 调用）
`ACCESS_FINE_LOCATION`	精确定位权限	GPS 追踪宠物位置
`ACCESS_COARSE_LOCATION`	粗略定位权限	使用 Wi-Fi/基站定位
`ACCESS_BACKGROUND_LOCATION`	后台定位权限（Android 10+）	App 在后台运行时继续定位
`WRITE_EXTERNAL_STORAGE`	写入外部存储	保存宠物照片到相册
`READ_EXTERNAL_STORAGE`	读取外部存储	加载本地照片
`FOREGROUND_SERVICE`	前台服务权限	持续定位（通知栏常驻）
`USE_BIOMETRIC`	生物识别权限	指纹/面部识别支付验证
`CAMERA`	相机权限	拍宠物照片或二维码
`BLUETOOTH` / `BLUETOOTH_ADMIN`	蓝牙权限	连接智能项圈或喂食器设备
`VIBRATE`	震动	通知提醒、交互反馈
`SCHEDULE_EXACT_ALARM`	精准定时	闹钟、定时通知（Android 12+）
`READ_CONTACTS`	读取联系人	社交分享、邀请好友等
`RECORD_AUDIO`	麦克风权限	语音输入、语音识别

🎯 常用监听器 & 辅助类（Listeners and Helper Classes）

类名	中文说明	用途
`OnClickListener`	点击事件监听器	按钮点击（下单、添加宠物等）
`OnTouchListener`	触摸事件监听器	处理地图触摸，宠物图标点击
`ScaleGestureDetector`	缩放手势检测器	地图缩放、图片放大缩小
`GestureDetector`	手势识别器	单击、滑动、长按识别等
`SensorEventListener`	传感器监听器	连接硬件项圈、检测距离
`Adapter`	数据适配器	绑定宠物列表、订单数据等
`RecyclerView.Adapter`	高级适配器	高性能列表展示
`ViewHolder`	列表项缓存器	优化性能，复用列表项 View
`Handler/Runnable`	消息处理器 异步任务处理	异步任务或 UI 线程更新，延迟执行、消息机制
`Intent`	意图类	页面跳转，传递参数（如宠物ID）
`AlarmManager`	定时任务调度器	定时提醒、自动任务等
`MediaPlayer` / `ExoPlayer`	媒体播放辅助类	音视频播放、预览功能
`BroadcastReceiver`	广播接收器	接收系统广播或事件通知
`OnLongClickListener`	长按监听器	长按选择、删除等
`LocationListener`	位置监听器	GPS 实时定位追踪
`BroadcastReceiver`	广播接收器	接收系统广播或事件通知
`TextWatcher`	输入框监听	实时监听 EditText 内容变化

🧩 常用 UI 元素（UI Elements）

元素	中文说明	用途
`TextView`	文本视图	显示宠物信息、标签等
`EditText`	输入框	输入宠物昵称、地址、数量等
`ImageView`	图像视图	显示宠物照片、地图图标等
`Button`	按钮	提交订单、切换界面
`FloatingActionButton`	悬浮按钮	快速添加宠物或跳转地图
`ListView` / `RecyclerView`	列表视图	展示宠物、订单、商品等信息
`Switch` / `ToggleButton`	开关	控制功能（如开启提醒）
`CardView`	卡片视图	美化布局（宠物信息卡）
`ConstraintLayout`	约束布局	灵活布局控制（如头像+文字）
`LinearLayout`	线性布局	垂直或水平排列元素
`FrameLayout`	框架布局	用于 Fragment 容器或覆盖图层
`ProgressBar`	进度条	显示加载、下单进度
`Toast / Snackbar`	吐司提示	下单成功、错误提示等
`AlertDialog`	弹窗对话框	确认下单、取消提醒
`TabLayout`	标签页	多个功能页切换（如宠物/订单/设置）
`ScrollView`	滚动容器	可滚内容页
`TabLayout` + `ViewPager2`	标签页 + 页面滑动	页面导航（宠物、订单、设置）
`SeekBar` / `Slider`	滑动条	调节音量、亮度、等级等
`CheckBox`	多选框	勾选偏好
`RadioButton` + `RadioGroup`	单选框	选择性别/类型等

🥞 Layouts 常用布局类型

布局名	中文说明	特点/适用场景
`LinearLayout`	线性布局	元素纵向或横向顺序排列
`RelativeLayout`	相对布局	元素相对其他元素对齐
`ConstraintLayout`	约束布局 ✅（推荐）	扁平高效，适用于复杂 UI
`FrameLayout`	框架布局	层叠视图，常用于 Fragment 容器
`GridLayout`	网格布局	表格式控件排列，如计算器键盘
`ScrollView`	滚动视图	内容超出屏幕时可滑动（单子元素）
`NestedScrollView`	嵌套滚动容器	支持更复杂的嵌套滚动场景
`DrawerLayout`	侧滑菜单布局	主流 App 菜单结构
`CoordinatorLayout`	协调布局	与 FloatingActionButton、AppBar 等联动动画

1. Small fonts in the title of the app; Increase font
2. Important action (Add event) included in the Action Bar menu; Add a button in the main Activity.
3. No information about what the colors in the calendar mean; Add a legend or a menu item.
4. The color of the Action Bar menu is not appropriate to distinguish it in the Action Bar; Change color
5. "Sign out" is an action that should not be as prominent; Put in the Action Bar menu.
6. Add meeting: icon is not intuitive to the action it represents; Change icon

a. Activities:
1. User Profile
2. Pet Profile
3. Map
4. Order food
5. Create profile

We expect at least the 4 first activities. -1 point for every activity missing (max -3 points).

Intent-filters for the following transitions:
User -> Pet
Pet -> Order
Create -> User
Pet -> Map

-1 point for every transition missing (max -3 points).

✦ 输入变量及等价类 Equivalence Classes

变量	有效值 (Valid Classes)	无效值 (Invalid)	边界值 (Marginal Values)
家具类型	Beds, Cabinets, Drawers (+10) Desks, Dining sets (+5) Bookcases, Chairs (basic)	其他（如 Sofa, Mirror）	无
安装时间	Morning（基本） Afternoon (+20)	其他时间（如 "Evening"）	无
工时	0–3h（基本） 3.1–6h (+10/h)	>6h, <0h	0, 3, 3.1, 6

✦ 测试用例设计 Test Cases

✓ 有效测试用例（Valid）：
1. Bookcase, Morning, 2 hours
2. Desk, Afternoon, 5 hours
3. Bed, Morning, 3 hours

✓ 边界测试用例（Marginal）：
4. Drawer, Morning, 0 hours
5. Dining Set, Morning, 3 hours
6. Cabinet, Afternoon, 3.1 hours
7. Chair, Afternoon, 6 hours

✓ 无效测试用例（Invalid）：
8. Sofa, Morning, 2 hours (invalid furniture)
9. Desk, "Evening", 2 hours (invalid time)
10. Bookcase, Morning, 6.5 hours (invalid duration)

Permissions
1. Internet (for online order)
2. Google Services (for Google Maps)
3. Local storage (for Pet photos)
4. GPS (for location)

-1 point for every transition missing (max -2 points).

Unless empty answer, award 2 points.

We expect to see at least the following mentioned:
OnTouchListener (for simple touch interactions with the lists)
ScaleGestureDetector (for zoom in and out of the map)
OnClickListener (for regular buttons)
Adapter (to handle list items)

We expect to see at least these 4 mentioned. For every class missing, -1 point (max -3 points)

Unless empty answer, award 2 points.

We need to have a list in the User profile for the pets, a button to add a pet and a menu to edit profile, sign out, settings and so on.
We can have an Activity with two fragments for the Pet profile: one with photo(s) of the pet at the top and another with the pet's information at the bottom. We also need buttons to locate the pet (that would open the Map Activity) and to order food (that would open the Order Activity).

We have 7 UI elements in total. For every element missing, -1 point (max -5 points). For every other element that is well justified, award 1 point (max +5 points).
Unless empty answer, award 2 points.