

北京理工大学计算机学院

《Android 技术开发基础》课程设计模板

班级 07112005 学号 1120202944 姓名 董若扬

1 App 的运行与开发环境


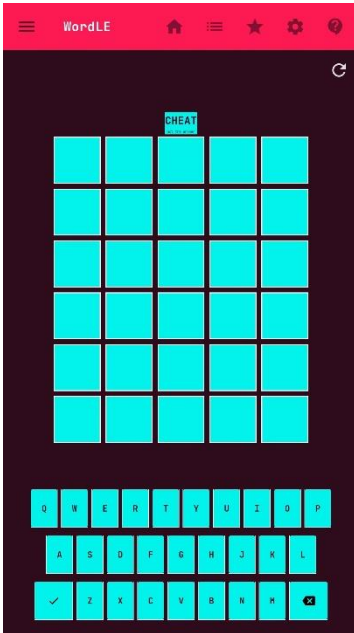
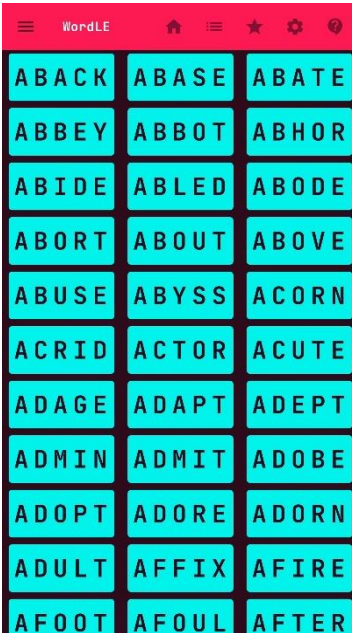
- (1) 运行环境: Android 10.0 min SDK 29
部署方法: 直接安装 app-release.apk 即可
- (2) 开发环境: Android Studio 2021.1.1 Patch 3, 创建数据集,生成列表和最后生成 UML 类图有用到 IntelliJ IDEA Ultimate 2022.1
- (3) 手写代码行数: 手机端约 3100 行 (不包含数据) (Kotlin 100%)

2 App 功能说明:

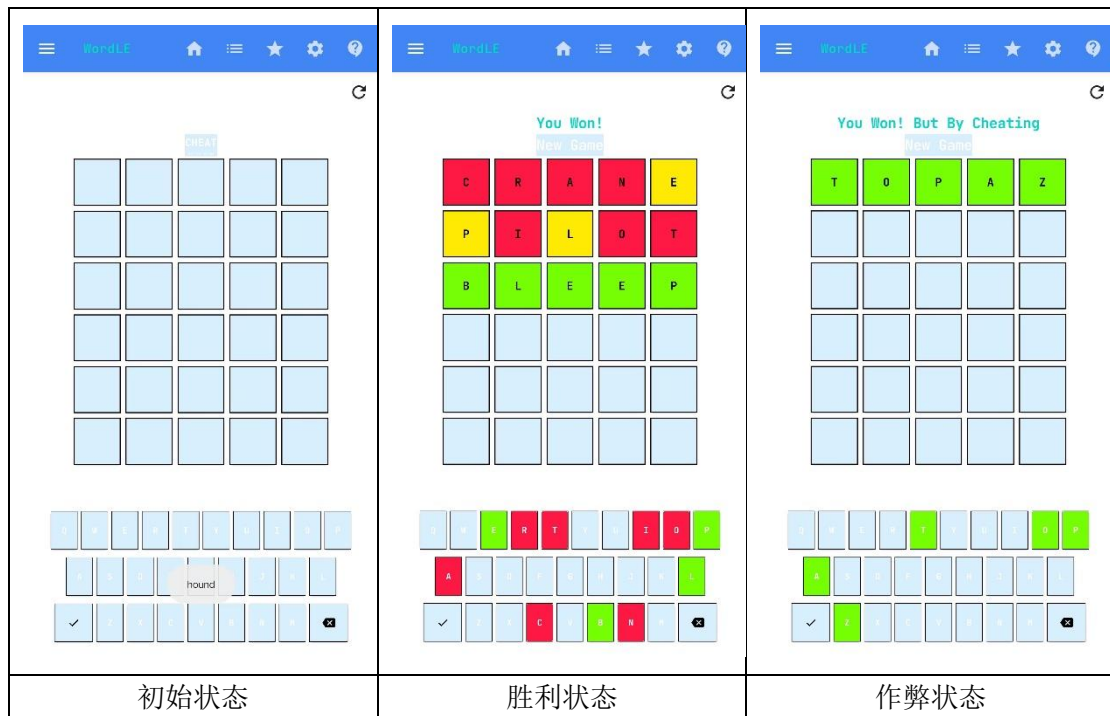
视频介绍:

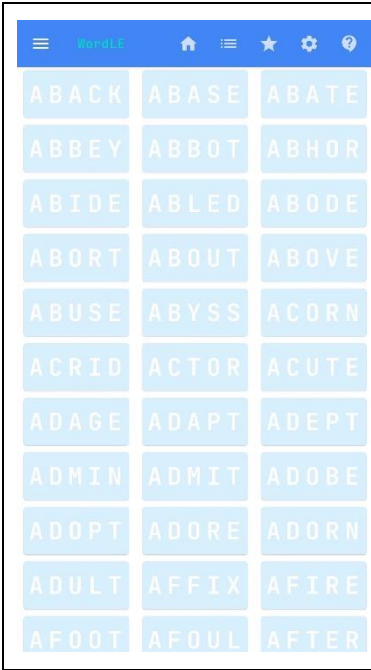
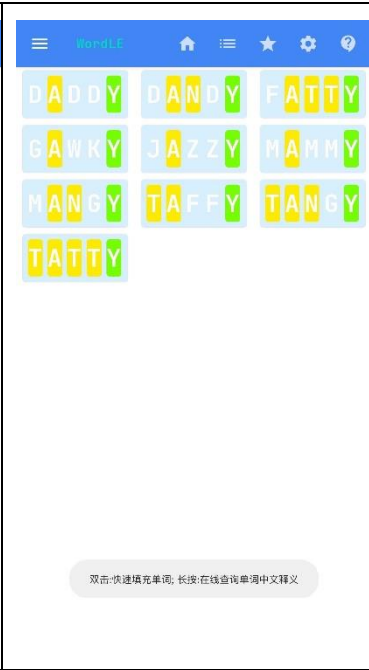

[App 操作 2 分钟简介.mp4](#)

给出手机屏幕截图, 底下给出文字说明。

		
抽屉导航栏	游戏界面	词库界面

统计界面(没有游戏记录时)	统计界面(有游戏记录时)	设置界面
关于界面	退出界面	



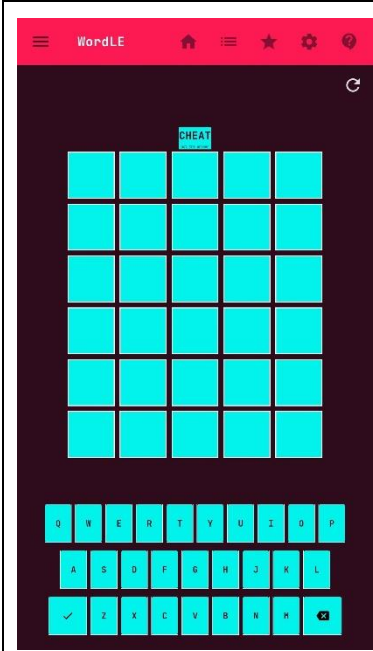
 <p>The screenshot shows the Wordle app's initial word bank. It consists of a 10x3 grid of word tiles. The words are: ABACK, ABASE, ABATE, ABBEY, ABBOT, ABHOR, ABIDE, ABLED, ABODE, ABORT, ABOUT, ABOVE, ABUSE, ABYSS, ACORN, ACRID, ACTOR, ACUTE, ADAGE, ADAPT, ADEPT, ADMIN, ADMIT, ADOBE, ADOPT, ADORE, ADORN, ADULT, AFFIX, AFIRE, AFOOT, AFOUL, AFTER.</p>	 <p>The screenshot shows the Wordle app during a game. The word bank is partially filled with words that have been selected: DADDY, DANDY, FATTY, GAWKY, JAZZY, MAMMY, MANGY, TAFFY, TANGY, and TATTY. A button at the bottom reads: 双击:快速填充单词; 长按:在线查询单词中文释义.</p>	 <p>The screenshot shows the Baidu Translate app interface. The word 'TAFFY' has been entered, and the app displays its Chinese translation '太妃糖' (Tāifēitáng), which means 'candy' or 'caramel'. It also provides the English pronunciation [ˈtæfi] and a button to '原读' (original reading). Below the translation, there are sections for '详细释义' (detailed meaning), '双语例句' (bilingual examples), and '英英释义' (English-English meaning). The '双语例句' section shows two examples: 'I'm going to get some saltwater taffy.' and 'By my fifth piece of peppermint taffy, I realized... maybe I wasn't banking on'.</p>
词库初始状态	游戏过程中筛选了候选词	长按直接翻译

查看历史记录	历史记录详情	分享到社交平台
分享内容	删除一条记录	

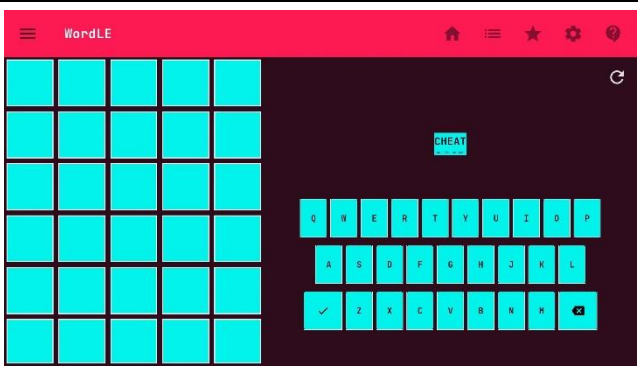
设置个性化主题	深色模式(跟随系统)	浅色模式(跟随系统)
舒适护眼模式	BBS 论坛风格	微信配色模式

		
抖音配色模式	随机模式(闪烁几次)	设置难度

	
全部收起	部分展开



游戏界面竖屏



游戏界面横屏



词库界面竖屏



词库界面横屏

	
详情界面竖屏	详情界面横屏
	
关于界面竖屏(展开状态能保留)	关于界面横屏(展开状态能保留)



3 App 架构设计及技术实现方案

(1) 对用到的算法的说明，简要介绍其算法步骤、时间空间复杂度等（可选）

按从复杂（不是指复杂度）到无需贅言排序：

- 1) 更新输入方格的状态，更新数据结构 `Map<Int,String>`，`Set<String>`，`List<List<String>>`，若输入字符数组对应位等于单词对应位，将 `Map` 索引处值更新为该字符，并返回绿色反馈。否则：若输入字符数组对应位在单词

中，添加到 **Set** 中，添加到 **List<List<String>>** 对应索引处的 **List<String>** 中，并返回黄色反馈，否则返回红色反馈。为什么要 **Map**，**Set**，**List** 各种各样数据结构都上一遍呢，因为 **Wordle** 中单词形式太多了，要避免答案中只有一个某字母，但输入中多个该字母标黄；答案和输入中都有多个某字母，但它们不一一对应；答案中只有一个某字母，但输入中存在多个该字母，一个标绿的同时其他标黄了；答案和输入中都有多个某字母，但输入中标绿的和标黄的字母对应的是答案中同一个字母。大致情况可以引用某数学博主 **3Blue1Brown** 的视频中的图。这个算法应该不是最优的，因为用到太多数据结构，谁让 **Wordle** 判定规则这么复杂。但是目前来看表现良好。排在第一是因为是一步步摸索出来的，前面的多多少少都有小问题。复杂度 $O(1)$ ，没有循环。

- 2) 判断是否在候选词列表中剔除某单词：对于候选词表中的某一个元素的每一位，即双重 **forEach** 循环，若索引位上反馈是正确且单词对应位上不等于 **Map** 中对应索引的值，则应该剔除。若索引位上反馈是不正确且单词中存在该错误字母，则应该剔除。若索引位上反馈是部分正确且单词对应位上的字母存在输入位上，则应该剔除。算法本身不难，但由于状态判断错综复杂，但单词存在很多特殊情况，如包含重复字符，包含重复字母但给出的反馈不同，给出部分正确反馈但是不能把所有含该字母的都保留下来。特殊情况大致有四种，且需要从用户输入中调取输入信息。复杂度：若词库中剩余单词为 n ，则复杂度为 $O(5n)$ ， n 的初始规模为 3000，每一轮都指数级减小，两三轮后 n 的数量级一般不超过 100，最多 6 轮，所以时间复杂度是 $O(n)$
- 3) 词库界面给候选词填色的算法，传入 **ViewModel** 中维护的 **Map**，**Set**，**List**，若候选词对应位等于字典索引处值，填充绿色。若候选词对应位在 **Set** 中且不在 **List<List<String>>** 中，填充黄色。否则不填色。算法难度源于 **Wordle** 的判定规则。复杂度： $O(1)$ ，没有循环。
- 4) 在历史记录中由字符串复现棋盘的算法，其实现逻辑和算法（1）一致，除了不需要更新 **Map**，**Set**，**List**，但是需要由长度为 30 的字符串复现回棋盘和生成     纯文本信息用于在社交网站上分享，也需要对颜色状态进行判断。复杂度 $O(1)$ ，没有循环。
- 5) 让键盘颜色与与方格颜色同步（即状态同步）：对键盘，即 **List<List<T>>** 按索引映射 **mapIndexed**，对每个 **List** 再对每个元素映射 **map** 让键盘对应字母与格子对应字母状态相等。返回映射操作后的键盘。复杂度： $O(mnp)$ ， $m=3$ ， $n=8\sim 9$ ， $p=26$
- 6) 对位置的控制，输入位置指的是下一次输入的位置，即开始时是 0，最后是 5。若未输入到最后一位，每次输入，位置加一。若位置小于 5，无法提交尝试。若提交成功位置下移一行。若按退格，位置减一再清空格子状态。本来位置是很好控制的，但是由于添加了退位操作，必须让出现一个虚位置 5（即第六个，大于长度 5），但此虚位要正确处理，不然会引发越位异常。同时由于输入到最后一位完成时，再按任何字母输入位置都不该变动了，不然退位键也得跟着多按几次，不合逻辑。另外就是最后一位和其他位之间特殊性是不一样的，所以最后一位的位置操作需要特殊考虑，不然就会要么出现普通位删除但字母不消失，要么出现特殊位删除字母消失但光标看似多移动了一位的不健壮的情况。由于此算法分步在各种方法中，

复杂度不好计算。

- 7) 检查输入合法性，即检查单词是否在列表中
- 8) 复位各种状态，即让状态恢复初始值，根据列表大小 n ，复杂度位 $O(n)$

(2) 程序架构设计及技术实现方案

[框架介绍详细信息 PDF](#)

[框架介绍详细信息图片](#)

[框架介绍详细信息 Markdown](#)

程序架构设计：

根目录：

- 1) `@AndroidEntryPoint` `MainActivity`，接收依赖注入。实例化数据库，重写 `OnCreate` 方法，实例化 `CoroutineScop` 用于协程，`NavController` 用于导航。创建可组合元素 `Scaffold`，为界面搭建主要的导航框架，包括 `TopBar` 顶部导航栏，`Drawer` 抽屉导航栏，为 `Drawer` 中每个元素定义点击导航事件，在 `NavHost` 中根据 `route` 值构建导航关系
- 2) `sealed class`: `NavDrawerItem`，封装不同页面的 `route`，`icon` 资源索引，页面 `title`，用于 `MainActivity` 中的导航

按字母顺序分为这些包：

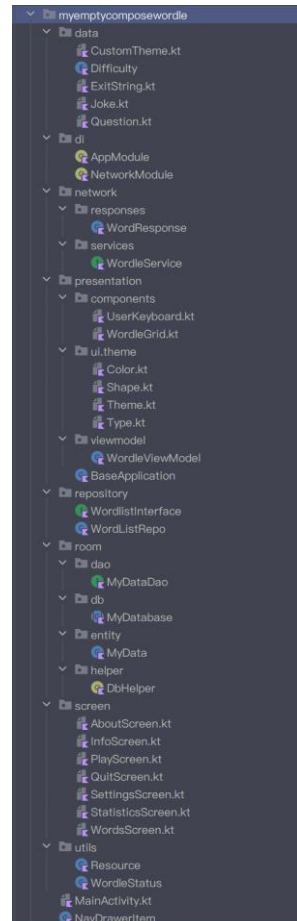
- 1) `data` 包：存放数据类，密封类，枚举类
 - a) `data class`: `CustomTheme`，存放主题的设置代码，描述文本，预览图片的资源索引值
 - b) `enum class`: `Difficulty`，存放难度值选择
 - c) `data class`: `exitString`，存放退出界面的标题
 - d) `data class`: `Joke`，存放退出界面的程序员冷笑话
 - e) `data class`: `Question`，存放关于界面的文本信息
- 2) `di` 包
 - a) `object`: `AppModule`，用于依赖注入，将 `Activity` 上下文返回 `BaseApplication` 基类
 - b) `object`: `NetworkModule`，用于依赖注入，提供网络服务，获取 API 返回的信息
- 3) `network` 包
 - a) `response` 包
 - i. `data class`: `WordResponse`，接受储存返回信息的数据类
 - b) `services` 包
 - i. `interface`: `WordService`，提供挂起函数 `getWord`，注解 `@GET("getword")` 从 API 上得到单词
- 4) `presentation` 包
 - a) `components` 包
 - i. `@Composable`: `UserKeyboard`，手写实现的键盘组件，用于显示键盘和提供对应的点击事件

- ii. **@Composable: WordleGrid**, 手写实现的格子组件, 用于展示输入的结果, 根据状态显示不同颜色
- b) **theme 包**
 - i. **kt: Color**, 定义颜色常量
 - ii. **kt: Shape**, 定义一些圆角形状
 - iii. **kt: Theme**, 定义不同自定义主题下选用的颜色搭配 (**primary**, **secondary**, **background**, **onBackground**, **surface** 等)
 - iv. **kt: Type**, 定义字体, 主要是增加 **JetBrains Mono** 等宽编程字体改善单词显示效果
- c) **viewmodel 包**
 - i. **class: @HiltViewModel WordleViewModel**, 接受注入 **WordListRepo** (其实现了 **Word listInterface** 接口, 包含一个获取单词的方法); 里面包含全局都在使用的状态, 数据, 改变这些数据触发页面重组的方法。还充当互不依赖两界面之间的桥梁。组件的初始数据, 进入游戏的初始化方法等解决很多开发难题的中间媒介。
- d) **class: @HiltAndroidApp BaseApplication**, 依赖注入的基类
- 5) **repository 包**
 - a) **interface: WordlistInterface**, 声明挂起函数用于获取单词
 - b) **class: WordListRepo**, 接收注入 **WordleService** 接口其含有从 API 调去单词的指令, 实现了 **WordlistInterface** 接口
- 6) **room 包**
 - a) **dao 包**
 - i. **interface: @Dao MyDataDao**, 数据存取对象, 将 **SQL** 命令与 **Kotlin** 接口方法关联起来, 提供增删改查方法
 - b) **db 包**
 - i. **abstract class : MyDatabas**, 从 **RoomDatabase** 中派生出子类, 用于存取本 app 用到的数据库。声明实现 **MyDataDao** 接口的抽象函数
 - c) **entity 包**
 - i. **data class: Mydata**: 自定义的数据库中的数据类, 使用 **@Parcelize** 注解实现 **Parcelable** 完成序列化
 - d) **helper 包**
 - i. **object: DbHelper**, 定义了拓展 **Date** 类的返回年月日时分秒的方法, 定义一个创建一个数据实例的方法
- 7) **screen 包**
 - a) **@Composable AboutScreen**, 关于界面, 提供游戏的相关信息
 - b) **@Composable InfoScreen**, 详情界面, 提供历史记录详情和分享功能
 - c) **@Composable PlayScreen**, 游戏界面, 游戏的主界面
 - d) **@Composable QuitScreen**, 退出界面, 展现随机冷笑话的地方
 - e) **@Composable SettingsScreen**, 设置界面, 提供简单的设置选项, 如切换主题, 设置游戏难度
 - f) **@Composable StatisticsScreen**, 统计界面, 存放了用户的游戏数据, 用户可逐条查看和删除, 还能分享到社交平台
 - g) **@Composable WordsScreen**, 词库界面, 根据游戏进度的不同, 展现当前阶段所有实际上有可能成为答案的候选词, 并制作手势操作, 提升输入体验, 还能

查询中文释义，一个迷你的背单词功能

8) utils 包

- a) **seal class: Resource<T>**，封装了获取单词可能出现的所有状态，成功，失败，加载中，初始状态（空），并附带得到的单词或异常信息，为游戏界面的操作服务
- b) **seal class: WordleStatus**：封装了每个格子元素，键盘元素的状态，正确，部分正确，错误，输入中，空，并根据这些状态为这些元素装填相应提示颜色



项目框架 1

程序架构与功能介绍：

1) Main Activity

a) Scaffold

- i. **TopBar 顶部导航栏**
 - 1. 提供与侧边导航栏功能一致的快捷按钮
 - 2. 应用名称
 - 3. 打开侧边导航栏的按钮
- ii. **Drawer 侧边导航栏**
 - 1. 可通过点击和滑动进入
 - 2. 提供导航不同页面的详细信息
- iii. **Navigation 导航**
 - 1. 提供不同页面的 **route**，图标，标题

2. 实现根据 route 值导航页面

2) Compose

a) 游戏界面

i. 图形化输入方格

1. 状态, 颜色
2. Empty 白色
3. Input 白色
4. Correct 绿色
5. Partial 黄色
6. Incorrect 红色

ii. 图形化键盘

1. 状态, 颜色
2. 退格键
 - a) 删除当前位置输入
3. 确认键
 - a) 检查输入条件后提交结果
 - i. 更新方格颜色
 - ii. 更新键盘颜色
 - iii. 更新词库候选词
 - iv. 判断游戏是否结束
4. 字母键
 - a) 输入字母

iii. 作弊键

1. 弹出非强制性消息, 透露此次答案
2. 取消此次游戏记录
3. 让玩家产生胜之不武的愧疚感

iv. 提供了横屏适配, 横屏状态下能不丢失数据的情况下调整页面元素布局, 使其更方便玩家操作, 以及使页面图形比例更协调

b) 词库界面

i. 根据用户的尝试输入结果更新可行候选词

1. 包含用户输入中得到“红色”反馈的字母的单词不会出现在词库中
2. 用户输入中得到“绿色”反馈的字母不在起位置上的单词不会出现在词库中
3. 用户输入中得到“黄色”反馈的单词在报错位置上的单词不会出现在词库中

ii. 提供手势快捷操作

1. 单击: 为了防止误触, 单击为提醒玩家手势操作
2. 双击: 快速将此单词填入方格中
3. 长按: 访问网络, 得到此单词的中文翻译

iii. 提供横屏适配, 横屏状态下每行能显示更多的单词, 充分利用屏幕空间

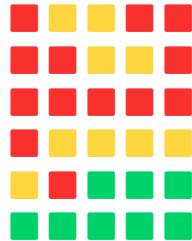
1. 使用 LazyVerticalGrid, 虽然是实验性 API, 但性能也挺不错

c) 统计界面

i. LazyColumn

1. 展示玩家成功通关游戏的记录

- a) 时间
 - b) 游戏模式
 - c) 尝试次数
 - d) 答案
2. 详情按钮：可查看当时的每一步操作
 3. 分享按钮，可以把结果分享给社交平台的好友



4. 垃圾桶按钮：删除此条记录
 - a) 删除时弹出警告框，防止用户手滑删除心爱的 in 2 tries 记录

d) 设置界面

- i. 个性化主题设置
 1. ”跟随系统“：会跟随系统深浅色颜色调整
 2. ”舒适护眼“：适合用户在夜间使用
 3. ”阴间论坛“：灵感来源上世纪 BBS 论坛
 4. ”葳兴薇恰“：配色来自微信公开的配色方案
 5. “百特丹斯”：配色来自抖音图标
 6. “随机选择”：屏幕快速闪动，最后随机选择
- ii. 难度设置
 1. 随机输入 (Easy)：玩家可以输入一个不是单词的字母串，可以更快缩小选择范围
 2. 单词输入 (Medium)：玩家只能输入合法的单词（合法还有另一层含义）
 3. 随机词串 (Hard)：答案本身将不一定是一个合法的单词，玩家输入也可以任意输入。

e) 关于界面

- i. 一些关于游戏的文本内容
- ii. 制作了动画过程，增强窗口扩展，缩小时的视觉效果
- iii. 使用 LazyColumn 提高性能
- iv. 制作了横屏适配，且用户在横竖屏切换的过程中，窗口之间的打开与否相互独立且不会丢失状态。而且横屏状态下每行多显示一个元素，充分利用屏幕空间

f) 退出界面

- i. 随机展示程序员冷笑话（目前量不多）
- ii. 致敬“一个测试工程师走进一家酒吧点了乱七八糟的东西”，所以进入这个界面，程序会假“闪退”

3) 主题设置

a) 不同主题搭配的颜色

- i. 跟随系统模式
- ii. 黑黄撞色潮流模式
- iii. PTT 论坛模式
- iv. 仿微信模式

- v. 仿字节跳动模式
- b) 自定义几何元素
 - i. 小, 中, 大三种圆角
- c) 字体
 - i. 普通阅读字体
 - ii. 适于显示英语单词的等宽编程字体 **JetBrains Mono**
- 4) Room 数据库
 - a) Data 数据类
 - i. **@PrimaryKey** 时间
 - ii. 游戏模式
 - iii. 尝试次数
 - iv. 单词
 - b) DbUtil 工具类
 - i. 拓展 **Date** 类, 增加一个获取当前时间并转为“年-月-日 时: 分: 秒”的函数
- 5) 从 **WordleViewModel** 中得到字段值, 创建一个数据实例并返回
 - a) Database 数据库
 - i. 声明数据表
 - ii. 声明数据库版本号
 - iii. 得到 **Dao** 接口的方法
 - b) Dao 接口
 - i. **@Insert** 提供添加数据操作
 - ii. **@Delete** 提供删除数据操作
 - iii. **@Update** 提供更新数据操作
 - iv. **@Query**(“select * from mydata”) 提供查询返回所有数据操作
 - v. **@Query**(“select * from mydata where time=:time”) 提供查询相应主键的数据
(编写但暂未使用)
- 6) DI 依赖注入
 - a) AppModule: BaseApplication: Application
 - i. AppModule 提供 Context as BaseApplication
 - ii. BaseApplication 程序入口, 并注入到 Main Activity 中
 - b) NetworkModule
 - i. 提供网络组件支持
 - ii. 解析 Url, 获取 API
- 7) 网络
 - a) WordResponse 数据类
 - i. **val word: String**, 用于保存得多的单词
 - b) WordleService 接口
 - i. 提供挂起函数 **@GET**(“getword”)得到包装成 **WordResponse** 的单词(同时包含单词和获取状态)
- 8) Repository 仓库
 - a) WordlistInterface 接口提供挂起函数获取资源加载状态
 - b) WordListRepo 接收注入的 **WordleService**, 由接口的方法获取一个单词并返回成功状态, 若捕获异常, 打包错误信息返回错误状态
- 9) ViewModel

a) 接收注入 WordListRepo，该类实现了 WordleService 接口的 getWord()函数

10) 状态

a) MutableState<>类型

- i. gameModel: Difficulty; 默认值: Free; 游戏模式状态
- ii. cheatOpen: Boolean; 默认值: false; 是否点了作弊按钮
- iii. won: Boolean; 默认值: false; 是否赢得了胜利
- iv. gameOver: Boolean; 默认值: false; 游戏是否结束，此与 won 并非相对关系
- v. word: Resource; 默认值: Resource.Empty(); 当前游戏的谜底。
- vi. answer: String; 默认值: “ ”; 即 word 的字面值，工具人，word 的替身，用于简化一些操作
- vii. _userGrid; 输入方格的状态，默认全为 Empty()
- viii. _userKeyboard; 键盘的状态，默认全为 Input(), 字面量为键对应的大写字母
- ix. 一个 Map<Int,String>, 用于筛选候选词，筛选对应位置不是“绿色”的单词。若用户在尝试中得到了“绿色”反馈，就把字母填入到 Map 键所指的值得中
- x. 一个 List<List>, 用于筛选候选词，筛选对应位置是“黄色”的单词。若用户在尝试中得到了“黄色”反馈，就把字母填入到与得到“黄色”反馈相同位置索引的 List 中
- xi. 一个 Set, 用于提示候选词，提示不是“黄色”位置上出现“黄色”字母的单词。若用户在尝试中得到了“黄色”反馈，就把字母填入 Set 中

b) 普通类型

- i. tries: Int; 默认值: 0; 表示玩家尝试次数，其实默认值是多少都没有关系，但 kotlin 不写这个 0 要多写延迟初始化和类型判断
- ii. 单词 List, 存放三千个候选词。一个折中的方法，避免使用 asset 相关读取。不太漂亮的实现方式，但仅仅为了这点数据实在不至于。

c) 函数

- i. 复位输入方格
- ii. 复位输入键盘
- iii. 获取单词，在 viewModelScope 协程中运行
 - 1. 获取 word 值和状态
 - 2. 若当前模式为随机词串模式，将 word 替换为一个随机生成的乱七八糟的字符串
 - 3. 如果 word 状态非异常，将值传给 answer
 - 4. 调用复位方格和复位键盘的函数
 - 5. 复位输入位置的行和列为 0; 复位词库候选词; 复位 Map, Set, List; 复位作弊开关为 false 和尝试次数为 0.
- iv. 更新方格
 - 1. 响应输入行为，将字母显示在方格中
 - 2. 更新输入位置
- v. 更新键盘
 - 1. 使键盘格子颜色与方格集一致，辅助玩家进行输入判断
- vi. 更新方格集
 - 1. 判断输入是否合法（不同游戏模式下法则不一）
 - 2. 对玩家提交的答案做出反馈，并给方格和键盘涂上辅助色
 - 3. 更新 Map, Set, List

- 4. 判断游戏是否结束
 - 5. 更新输入位置
 - vii. 退格
 - 1. 复位方格
 - 2. 更新输入位置
 - viii. 更新候选词库
 - ix. 判断输入合法性
- 11) 一些数据类
 - a) 自定义主题
 - i. 主题代号
 - ii. 主题描述
 - iii. 主题预览图
 - b) 难度（枚举类）
 - i. **FREE**（随意输入）
 - ii. **RESTRICTED**（单词输入）
 - iii. **RANDOM**（随机词串）
 - c) 测试工程师进酒吧
 - i. 存放各种各样测试工程师以各种各样方式进酒吧点各种各样东西的描述
 - d) 冷笑话
 - i. 用于在退出界面展示
 - e) 问题
 - i. 在关于界面给用户提供帮助
- 12) 资源文件
 - a) 原创 App logo
 - i. 运用游戏的三种主色调与方格构成，红色的方格微微浮起试图传达一种键盘的感觉，组成的 **W** 为 **Wordle** 的首字母
 - b) 原创 **WordLE** 图标
 - i. 根据不同主题选取不同颜色组成，能和页面和谐融合，设计理念大胆，创新。效果美观，大方
 - c) 原创版权图标
 - i. 内容是我 **GitHub** 昵称，至于为什么是这个昵称，我也不知道，也许 **GitHub** 用户不要这么会抢名字，也许我就叫其他昵称了。其设计理念与 **WordLE** 图标一致，共同组成主题的一部分
 - d) 原创主题预览图
 - i. 用于在主题设置界面给用户提提供视觉参考
 - e) 来自阿里适量图标库的图标
 - i. 用于辅助标识导航栏中的导向
 - f) **Jetbrains** 的开源等宽字体
 - i. 其显示候选词和历史记录时视觉效果整齐统一
 - g) 从网页 **HTML** 获取的，根据网上公开资料提取的，根据 **Android Material** 规范组合的颜色
 - i. 用于主题搭配
 - h) 深色模式
 - i. 使应用显示适应用户的手机的颜色深浅

- i) 字符串
 - i. 各种数据类中的字符串都引用至此，为之后提供多语言支持做准备，保证良好的可维护性
- 13) 需要的权限
 - a) 网络权限
 - b) 获取网络状态
 - c) 读写外部存储
- 14) Gradle 配置
 - a) kotlin 相关
 - b) 依赖注入相关
 - c) Compose 相关
 - d) Room 相关
 - e) 协程相关
 - f) jetpack 相关

技术实现方案：

[实现思路详细信息 PDF](#)

[实现思路详细信息图片](#)

[实现思路详细信息 Markdown](#)

- 1) 使用 Room
 - a) 移植后构建失败
 - i. 注解上添加 `exportSchema`，数据库版本手动加一
 - b) 运行时闪退
 - i. 给 Dao 中的函数声明为挂起函数，把操作放在协程中运行
 - c) 操作后数据不更新
 - i. 数据接收容器设为状态，获取数据事件放在 `OnClick` 中触发
 - d) 应用切至后台时意外退出
 - i. 序列化自定义数据类
 - e) 数据的更新不及时
 - i. 由于数据的更新在协程中完成，而协程要放在点击事件中触发，用户直接进入统计界面，不会触发任何数据更新。一开始我让用户每次查看数据都要手动展开列表，这样展开列表的动作同时还包括了数据更新。但这样即便用户没有添加新数据进来还得忍受每次查看一个数据就要更新一次列表的情况，于是引入一个图标用于在列表展开时供用户刷新，同时原先是否状态的状态用 `rememberSavable` 委托，即便屏幕方向变化，列表也不会莫名其妙收回了。
- 2) 设置不同主题颜色搭配
 - a) 如何在程序运行中切换颜色
 - i. 在程序的比较顶层的位置引入一个表示状态模式的状态，并在 `WordleTheme` 中根据不同状态选择不同 `primary`，`background` 等颜色。然后向下传递改变此状态的动作，直到传入到“设置”中，当设置中点击切换主题时就触发了在顶

层改变主题状态的动作。即预先把“设置”中的状态提升了。

- b) 如何制作界面快速闪烁的视觉效果
 - i. 如果在设置中使用循环重复执行触发事件，似乎这样的重组行为会被“优化”掉。于是我选择在程序进入 `WordleTheme` 前设置一个状态用于检测是否需要这样的闪烁，每次用户点击产生该闪烁的按钮时，这个状态就会加 10，并在每次闪烁后-1。

3) 横屏适配

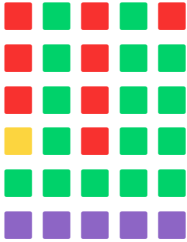
- a) 因为屏幕比例的关系，手机屏幕方向发生变化时，界面元素轻则发生畸变，重则不能在屏幕上显示。
 - i. 很显然，解决方法就是锁定屏幕方向。
 - ii. 但这种做不到就干脆禁止用户进行这种操作的行为是消极的开发行为，于是我在 YouTube 上学到了使用 `LocalConfiguration.current` 获取设备当前的状态，再判断是 `Configuration` 类下的 `ORIENTATION_PORTRAIT` 还是 `ORIENTATION_LANDSCAPE`，再把界面元素封装一下，为两种方向分别设计布局。

4) 展示列表，`LazyColumn`，`LazyRow`，`LazyVerticalGrid`。

- a) 列表中每个元素都有相应的标题和内容，在不需要的时候，为了能看更多条消息，需要每条内容能够单独控制扩展和收起
 - i. 在 `LazyColumn` 的每个元素中添加一个按钮，其图标是简易箭头，当点击时吧元素的扩展状态取反，并根据其值是否为真，决定是否显示内容，当其值为真时，显示详细内容。
- b) 伸展与收缩过程非常突兀，就像手机卡住了一样
 - i. 为 `Column` 的变化添加动画过程 `animateContentSize(...)`
- c) 列表横屏时被拉伸，影响阅读体验
 - i. 容器由 `LazyColumn` 改成 `LazyVerticalGrid`，并让横屏状态下显示的格子数大致是竖屏下的 1.8 倍（16:9）
- d) `LazyVerticalGrid` 中的元素张开时，同一行的其他独立元素大小也会发生变化，显得很奇怪
 - i. 再换成多个 `LazyColumn`，并把要展示的元素切片到多个列表中分别传到 `LazyColumn` 中
- e) 这样操作的结果是，屏幕方向变化，每个元素是否展开的状态“不会被留下来”，实际上这样操作横竖屏使用的是两套状态
 - i. 状态提升到判断屏幕方向的可组合函数处
- f) 这样操作的结果状态可以保留了，但是每个元素均不能单独控制了
 - i. 把单个状态改成“状态字典”，分别映射了每个元素和是否展开
- g) 字典作为状态时，点击事件只改变字典的键值对，字典真值没有改变，页面不重组
 - i. 由于没找到可以替代字典的数据类型，所以最初的笨办法是新增一个状态，其值是字典的 `toString()` 值，但显然这个 `String` 值没有任何含义，为了提升性能，将这个“附属状态”改成一个 `Boolean` 值，每次键值对发生变化时，这个“附属状态”就取反
 - ii. 最终还要把这个“附属状态赋值给一个垃圾变量，不然运行时忽略这个“附属状态的变化”，页面不发生重组。”

5) 导航

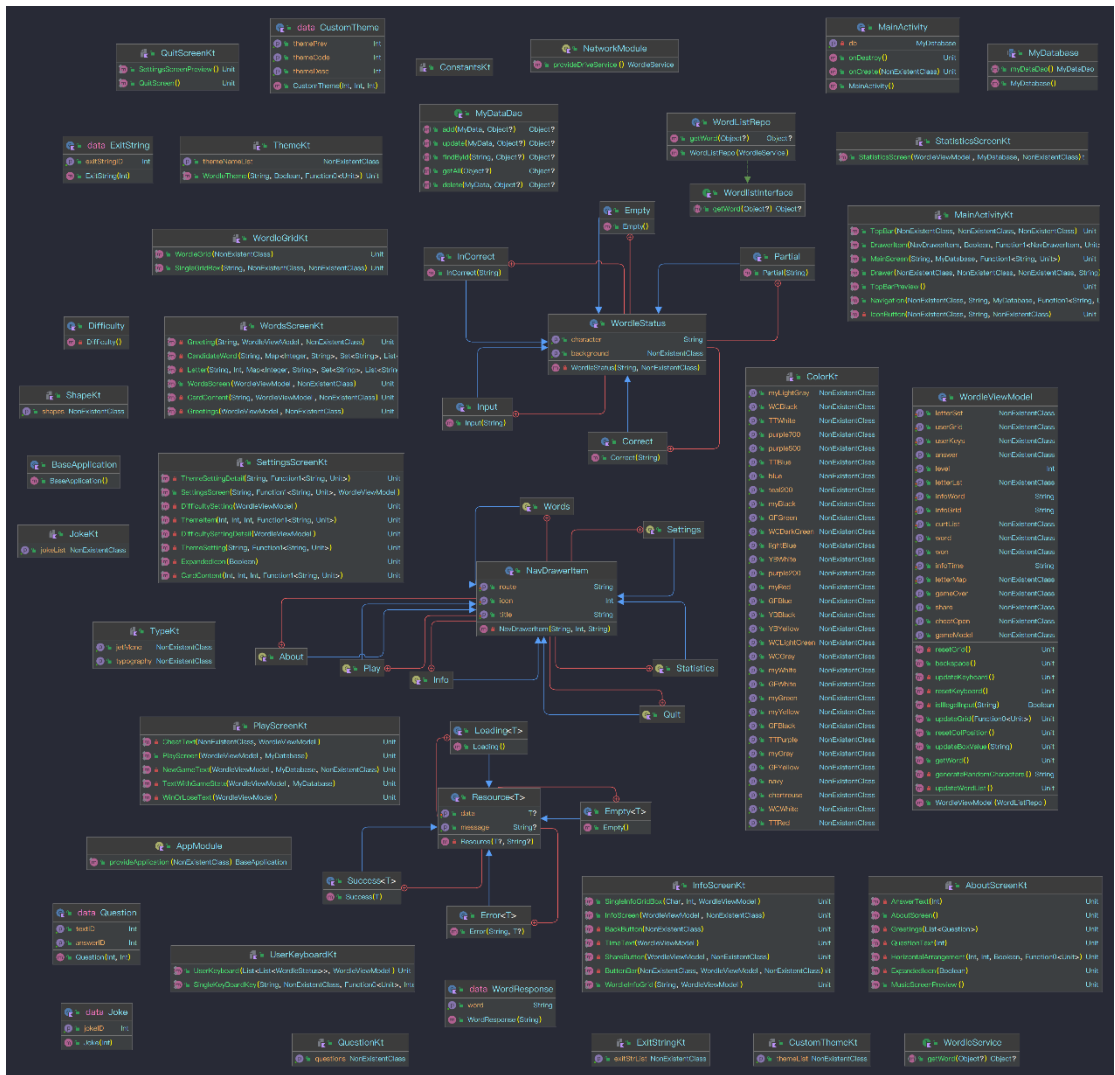
- a) 导航造成了堆栈错综复杂，用户回退可能出现不同寻常的结果
 - i. 强制回退时退出到主界面，防止产生大量堆栈

- ii. 导航中令 `launchSingleTop` 和 `restoreState` 值为 `true`
 - iii. 将 `MainActivity` 以外的导航事件由传入一个点击事件改为传入一个 `NavController`，使其堆栈发生在当前界面而非主界面。
- 6) Compose
 - a) 缺少上下文 `Context`
 - i. 传统做法从 `MainActivity` 千里迢迢传一个 `this` 到很深的地方，改进为在 `Composable` 中使用 `LocalContext.current` 得到 `Context`，此方法可解决 `Toast` 缺失上下文
 - ii. 对 `Toast` 函数来说，把唤起 `Toast` 事件放在 `Activity` 再传入，避免在 `Composable` 中使用 `Toast`。
 - b) 退出应用
 - i. 尝试了好几种方法都不能在 `Composable` 中直接使用，最后还是使用了 `Java` 中被废弃的 `Handler`（）来操作一个延时任务（退出）
- 7) 分享
 - a) 以何种方式分享？调用权限截屏分享？制作图片？一串文字？要么技术太过棘手，要么效果很差
 - i. 
- 8) ViewModel
 - a) 储存“全局状态”
 - i. 在 `viewModel` 中储存：是否胜利，是否游戏结束，是否作弊，候选词列表，本次答案，输入位置，分享信息等其他部分需要了解的状态
 - b) 提供方法
 - i. 包括开始一局新游戏，更新格子，更新键盘，复位格子，复位键盘，对候选词列表的更新，控制输入位置的移动等对这些状态的操作
 - c) 担任信息沟通中间人
 - i. 在统计界面数据库获取信息后，需要把相关信息传递给分享界面，但是操作数据库要求比较高，我希望最后是传递数据而不是传递动作，但分享和统计两个功能本身不存在依赖关系，只能在统计界面把数据存入 `viewModel`，在分享界面从 `viewModel` 把数据读出来
- 9) 算法
 - a) 更新候选词列表
 - i. 使用 `Map<Int,String>`, `Set`, `List<List>` 三种数据结构
 - ii. 当得到绿色反馈时：把 `Map` 对于索引的值改为该字母
 - iii. 当得到黄色反馈时，把该字母加到 `Set` 中，并加到 `List<List>` 对应索引的 `List` 中
 - b) 筛选掉不可能的词汇
 - i. 若得到绿色反馈：如果对应位置不是这个字母，则筛选掉这个词汇
 - ii. 红色反馈：如果词汇中含有这个字母，则筛选掉
 - iii. 黄色反馈：如果词汇对应位置含有 `List<List>` 中记录的字母，则筛选掉
 - c) 显示颜色提示

- i. 如果字母位置的字母是 **Map** 索引处的值，则显示绿色
 - ii. 如果字母在 **Set** 中，但不在 **List<List>** 中，显示黄色
 - iii. 其他显示红色
- 10) 词库元素交互
 - a) 让候选词列表的功能不仅仅是展示给用户单词
 - i. 使用 **detectTapGestures** ()
 - ii. 单击：提示用户使用方法
 - iii. 双击，快速填充候选词
 - iv. 长按：查看单词中文释义
- 11) 获取单词
 - a) 如何从网络拉取一个随机的单词
 - i. **GitHub** 上公开了一个 **Wordle** 的 **API**，可使用 **http.get(“getword”)** 随机得到一个单词
- 12) 一些手写组件
 - a) 键盘
 - i. 传入 **List<List>**，使用 **forEach** 循环打印每个格子，并根据格子值传入点击事件：输入，删除或是提交。
 - b) 格子
 - i. 传入 **SnapshotStateList<List>**，使用 **forEach** 循环打印每个格子，并根据格子自身的状态控制显示不同的颜色
- 13) 内容的封装
 - a) 一些需要搭配使用的内容，例如主题设置有相应的代码，描述，预览图。关于界面的文本有标题和正文
 - i. 使用数据类封装，用 **@StringRes** 注解标记字段是字符串索引，并将文本内容存在 **String** 文件中，为后期升级多语言做准备
 - b) 一些需要随机展示的字符串，如测试师进酒吧，退出时的冷笑话
 - i. 封装在数据类中，操作起来更方便
- 14) 导航栏
 - a) 需要顶部导航栏，抽屉导航栏
 - i. 使用 **Scaffold**，定义 **TopBar**，**Drawer**，在 **Drawer** 中为每个 **Item** 定义点击导航事件，在密封类中封装每个页面的 **title**，**route**，**icon**，最后在 **NacHost** 中实现导航功能
 - **App** 中设计了哪几个类，类之间有何关系：继承、组合、实现某接口……
 - 你将这些类划分成了多少个包？
 - 组件化开发：**App** 引用了哪些第三方组件？你自己开发了哪些组件？组件之间的依赖关系是怎么样的？
 -

推荐使用 **UML** 类图、包图和组件图描述你的程序架构。

如果是移动互联应用，需要同时展示 **Server** 端的架构及技术实现方案



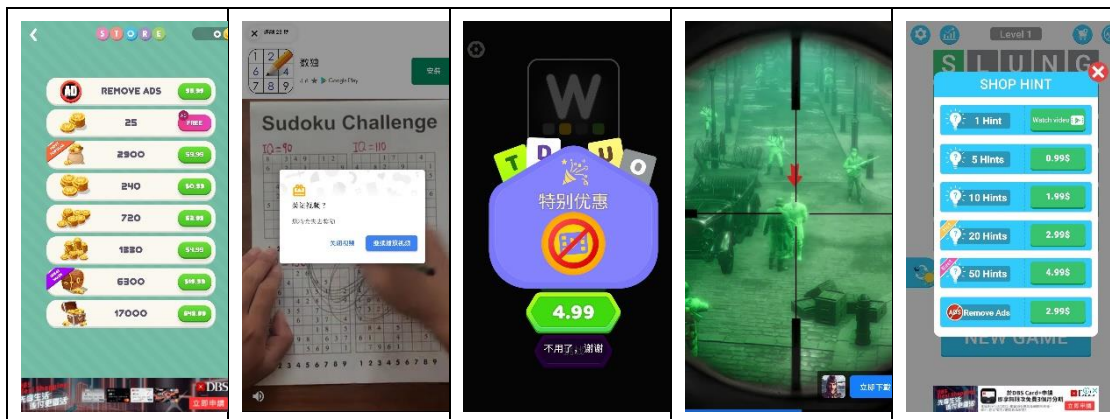
UML 类图(由 IDEA 生成, 部分类型无法识别)

4 技术亮点、技术难点及其解决方案

技术亮点和技术难点和解决方案在此前“实现思路“中已经陈述过了；
不过在技术之外，还有一个亮点，就是 App 的图标是我自己设计的！

- 与其他类似的 App 相比，你这个 App 有哪些与众不同之处？

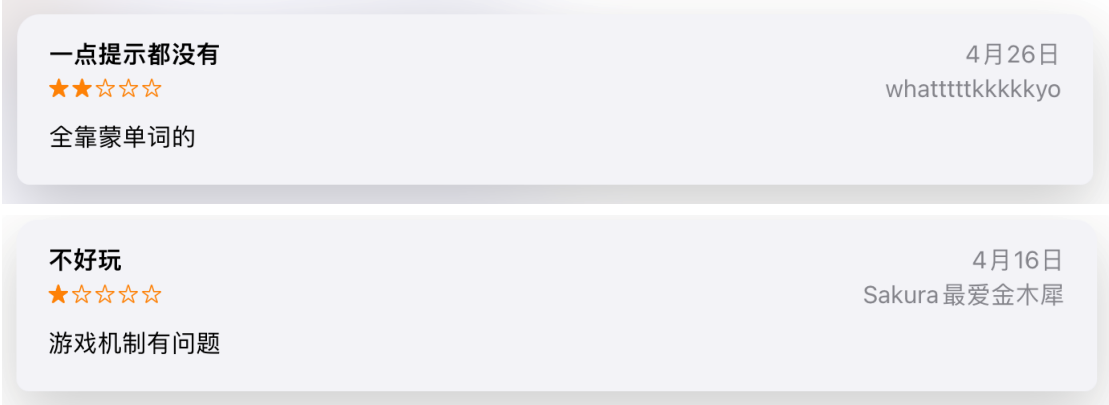
我下载了 Google Play 上排名前五的应用并使用了一段时间，发现与市场上成熟产品相比，我开发的应用无广告，无内购，完全免费



而如果在例如华为应用商店，完全找不到类似的产品（因为游戏最初是网页版的），说明在国内应用市场上，在一定程度上，成熟的 wordle 游戏还是一片空白。



让我们稍微离题一点，在苹果应用商店搜索 Wordle，评分最高的 wordle 游戏也只有少得可怜的 3 条评论，其中两条差评：



从中我们可以发现 App Store 上的 Wordle 游戏至少存在两个问题，一，没有辅助玩家游戏的功能，对于非母语人士来说，完全丧失了游戏本身的乐趣，退化为类似猜大小这样的无聊游戏。二，游戏判定有问题，就如我在文档中说的那样，wordle 对玩家来说规则简单，但这种简单是建立在它对单词复杂的判断逻辑中的，玩家能够通过十分简单的反馈信息调整自己的尝试，是因为玩家相信程序能够给出完全正确的反馈，这就像两个人做游戏，游戏的一方完全信任裁判一方。而我目前对我写的单词反馈逻辑很满意。

当然在功能性上我的 APP 还远远比不上市面上用 Unity 开发的 wordle，后者甚至像愤怒的小鸟一样有闯关还能买道具，但我想 Android 最原生的应用能带给用户最自然的体验。

- 你这个 App 中，你认为最得意的地方在哪里？
最得意的是最后选择的使用 jetpack Compose 完成结课设计，其他得意之处都源于此。
最得意的部分是原先我打算留到暑假有时间再开发的“词库”，“统计”和当初根本没想到的“分享”功能。一开始我只做了一个游戏界面，其他界面都是用“开发中”占位。随着一边学习一边实践，在谷歌开发者网站上学习了 LazyColumn 和 rememberSavable 的用法，变着花样制作了关于，设置和退出界面，说得不好听，这只是在给应用换皮，其功能性没有任何进展，虽然有一两天我乐此不疲地在 ps 里做图，调色，但我意识到这些都是小儿科的技术，而我打算以此次提交结课设计为应用版本 1.0，既然 1.0 就不应该留有未完成的功能，于是我急转向，本来决定实现的多语言也就不再考虑了。首先我完成的是词库功能，至少完成这个功能后我

自己玩自己做的游戏已经没有问题了，在这个过程中我用的还是 `LazyColumn` 和 `rememberSavable` 这套技术，但我感觉我的应用功能性有了巨大进展。然后接着完成统计功能，我觉得这个是所有游戏必不可少的功能，为了实现这个功能又另外学了 `Room`（虽然理论上来说 `room` 的知识本身在教学范围内，但我当时没怎么学明白），`Room` 是这个应用中继 `Compose` 和依赖注入后收获最大的知识点。搞清楚细枝末节的“语法”后，发现是真好用，不愧是谷歌亲自动手的东西。最后统计功能完成后我又再次基础上开发了分享功能，可以把自己的胜利记录分享到社交媒体。至此最令我得意的是一个完整且应该成熟了的 App 由我从零开始实现了。

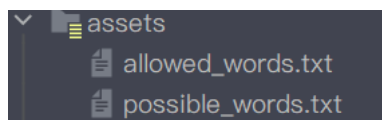
- 你开发这个程序中遇到了哪些具体困难，又是如何解决的？

如果把开发程序中难到我的都算作困难，那篇幅将太大了，所以在此我将困难定义为我向老师请教的技术要点：

1. 网络问题：最后查遍国内外各种搜索引擎和中英文网站，最后在 CSDN 中查到把 `C:\Users\<Username>\.gradle\gradle.properties` 文件中代理记录删了就行（也得是 CSDN，英文网站可能找不到这样的问题吧）

```
20:06 Gradle sync failed: Plugin [id: 'com.android.application', version: '7.1.3', apply: false] was not found in any of the following sources:
- Gradle Core Plugins (plugin is not in 'org.gradle' namespace)
- Plugin Repositories (could not resolve plugin artifact 'com.android.application:com.android.application.gradle.plugin:7.1.3')
Searched in the following repositories:
Gradle Central Plugin Repository
Google
MavenRepo (3 m 18 s 883 ms)
```

2. 以字典作为状态，键值对改变，页面不重组：得到的建议是实现 `Parcelable` 接口，并附加 `@Parcelize` 注解，实现 `State<T>` 或 `MutableState<T>` 接口或者调整数据结构。我最后的解决方法是使用一个附属状态 `Boolean`，让键值对变化的时候让附属状态取反，比较取巧，不过性能还不错？不过我现在写文档的时候想道另一个方法就是更改 `Map` 后把新 `Map` 赋值给状态 `Map` 的 `value` 属性，应该也能达到目的，不过既然一切运行正常，就不改了。
3. 在 `Android/data` 中找不到放进 `asset` 中的资源文件，网上说是需要 `root` 权限，而我考虑到我使用华为手机调试的，整了一晚上没整出来，疑虑有我不了解的东西限制了我使用，最后放弃了这条路线改用 `Room`



4. `Room` 一开始进行的非常不顺利，最后该改成挂起函数的改了，该放进协程的放了，把数据集定义成状态，把事件放在点击中。数据库版本号手动增加（最后由我从 1 加到了 7）这些都是看了课堂视频回放后一点点解决的。

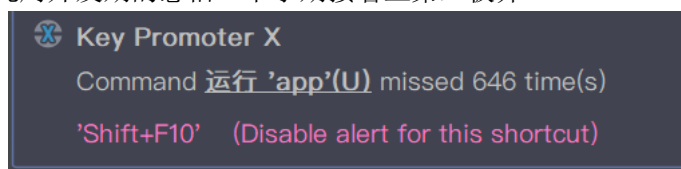
5月22日 给程序写注释

既然标题是简要开发过程，这里就不详细讲了。详细 7000 字开发过程和项目技术实现演变过程和心路历程请看附件！

6 学习感悟及对本课程的建议（可选）

经过一个学期的理论与技术学习和亲自动手开发，你有何感悟？对本课程的教学有何建议？

因为老师的教学理念——教最新的知识，用最新的技术，所以最终我选择使用了 Compose，我从来没有这么强烈的感受——新东西比老东西好用。这一点对我学习计算机技术有很大的启发。包括最初老师选择用 Kotlin 教授 Android，虽然我上学期跟着老师学的 Java，在最后做结课设计时，看到 Java 的示例代码就头大。然后就是老师在 Java 课程中说过在 Android 课程中也许没说过的一句话：没有通宵开发的经历的大学生活是不完整的，虽然我不确定我开发到看到北京四点初亮的天空在老师的标准里算不算通宵，不过沉浸在开发中的感觉真的很好，看着自己的作品一点点完善，特别是我开始开发的时候选题咨询了留德的朋友，因为时差，我凌晨把最新版发给他，起床就能收到意见，关于还可以添加什么功能，发现了什么 Bug 等等。最重要的是在分享的过程中开发，就不像是自娱自乐，不像是交一个结课设计，就像老师说的，做一个代表自己最高水平的“里程碑式”的作品。以上就是我这学期，尤其是最后几周开发期的感悟。下学期接着上第三板斧。



总结开发过程的一张图

课程建议，从结课后的第一个星期开始我倒回去看课件时我发现老师把安卓前期介绍得事无巨细，比如请求码，结果码，这可能在安卓发展的历程中是一个很重要的节点，在将来更具体可能 Server 端还有其他应用，但是对新手来说当时要记住这些 Activity 到 Activity，Fragment 到 Fragment，Activity 到 Fragment 跟背解题模板一样，结果到 jetpack 的 viewmodel，发现新技术比原生技术好用太多了。老师可能有其他考虑，但我觉得 jetpack 的内容可以稍微提前。

以及到了协程，难度曲线一下变陡了许多，我不了解这是不是我个人的感受，不过如果老师接收到很多人都提到这点的话可以考虑调整一下节奏，因为下课的时候经常感觉学的都快装不下了老师还吐槽今天讲得慢了。

还有 Compose 真是越看越喜欢，希望老师以后上课能多分给 Compose 一些课时，以及 Compose 上课时的上课形式，诚然 Compose 比 View 简单太多了，但是突然之间的转换，以我自己的体验就是：语法看不明白。所以我建议老师上课可以像安卓开发者频道的这个教程 <https://youtu.be/k3jvNqj4m08> 一样从最简单的 Text 开始搭建，一个小时的内容就能完成包含 LazyColumn, rememberSavable, 状态提升，和最基础的语法知识。或者将这个视频提前推荐给学生，因为我自己从这个视频中学到了很多。

最后感谢老师教授的最新的安卓技术和课后详尽的回复。