**软件工程基础自评文档**

在本次项目中，我主要负责Android客户端的开发工作，也就是Bit\_List这个应用程序的UI设计与前端逻辑开发工作。总体而言，我认为我成功地构建了一个功能完善、技术现代且用户界面友好的应用程序。但在过程中也暴露了一些问题，值得反思和改进。

**一、 个人自评**

**工作亮点与成果**

本次Android客户端开发工作的总体框架如下，我具体采用了业界主流的现代化Android开发技术栈，包括 Kotlin 语言、Jetpack Compose 用于UI声明式构建、MVVM 架构模式、ViewModel 进行状态管理、Retrofit 和 OkHttp 处理网络请求，以及 Kotlin Coroutines 处理异步任务。以此确保了客户端应用具有良好的性能、可维护性和扩展性。

具体而言，本次我的开发工作按照需求以模块化的方式进行展开，主要分为用户模块（包括用户注册登录以及个人信息修改等）、任务模块（最关键的部分，包括任务类的设置、任务的前端显示逻辑以及任务与其他模块之间的交流互通）、日历模块（日历的显示、日周月的视图切换，拖放任务等更复杂的操作功能）、团队协作模块（设置团队数据类，构建了包括团队创建、加入、成员管理和任务分配等关键功能）以及数据统计模块（集成Vico图标库将后端的数据可视化为任务完成率趋势图、状态分布图和时间分配报告等）另外，为了使任务完成更加高效化，在主页的任务列表还加入了番茄钟专注功能，能够与具体任务进行交互，使得用户数据能够更加具体化，更容易为用户提供更多的专注建议。

在任务开展过程中，我参与了UI设计、前端逻辑开发以及前后端交互的部分。整个过程中，我认为我的开发推进还相对积极，没有拖团队的任务后腿，在后端开发完成之前就完成了前端开发任务并参与到前后端交互之中，从任务完成度和任务态度上讲我认为我已经达到了自己的及格标准。并且我在开发过程中，极大地锻炼了开发能力，并且熟悉了一个完整的项目开发流程与相关专业知识，我也从中学习到了很多。

**不足与反思**

本次的任务开发不足之处也有很多，这里碍于篇幅对不足中最严重问题进行详细展开反思。这里我要反思的问题就是前后端交互协同的偏差，这也是本次项目中暴露出来的最大的问题。在项目初期，我首先开发了一个大致框架，并在Apifox中设置了我所需要的接口，之后后端同学的开发就是依据Apifox中的接口信息对核心功能进行开发，但是之后在我依照项目需求文档补充项目细节时，又设定了更多的交互接口，并在Task、User等关键数据类中加入了包括权限、团队ID等细节，并将修改后的接口在Apifox中进行更新。但是在与后端进行交互沟通时我发现，很多我想要从后端获取的数据都没有达到预期，经过进一步与后端开发的同学交流我发现后端依然使用的是第一版的接口设计。因此只能从头再来，将后端的数据库和管理逻辑进行补充，并一起沟通交流修改前端接口和后端逻辑，这一部分花了很长的时间，只能一点点将前后端的功能进行对比修改，也因此使得项目最后的完成度做了部分功能删除。

这暴露了我在项目初期未能与后端同学充分沟通、未能共同制定并严格遵守统一的API规范的问题。我设计并实现了很多前端功能（如修改密码、数据统计、附件权限管理），但其依赖的API接口和实体字段在后端并未实现。我应该在开发前，通过API文档或共享文档的形式，与后端敲定每一个接口的路径、方法、请求和响应体以及所有数据类型。

**二、团队建议**

基于我对开发过程中的视角，为了未来项目更高效地进行，我向团队提出以下建议：

1、 API先行，文档驱动: 在编码前，前后端应共同制定一份详尽的API文档，在我们团队的开发过程中使用的是Apifox。这份文档应成为双方共同遵守的“法律”，明确所有接口细节，并对修改和设置进行积极的团队沟通，避免后期因不匹配而产生的大量返工。

2、 高频集成，尽早暴露问题: 我们应避免前后端长时间各自“闭门造车”。建议建立每日或每两日的构建集成机制，尽早将前后端连接起来进行测试，第一时间发现并解决接口不一致、逻辑错误等问题。

3、 推行代码审查机制: 建立团队内的代码审查机制。通过互相审查代码，不仅能发现类似“注册后忘记保存Token”这样的逻辑漏洞，还能统一编码风格，促进团队成员技术上的共同进步。

**三、课程建议**

结合本次项目实践，我认为该软件工程课程可以在以下方面加强，我对课程的建议如下：

1、 强化全栈项目实战: 增加更多要求学生同时负责前后端开发的项目。只有亲身经历过从数据库设计、到API编写、再到前端UI实现的全过程，才能深刻理解前后端协同的痛点，学会如何设计出合理、一致的系统。

2、 教授协同开发流程与工具: 课程不应只停留在“写代码”，更应教授“如何与团队一起写好代码”。应将 Git 的协同工作流（如 Git Flow）、API文档工具（Apifox）以及项目管理工具（Jira/Trello）的教学纳入核心课程，这些是现代软件开发的必备技能。

3、 深入软件工程规范: 除了基础功能实现，课程应更深入地讲解软件质量保障的手段，包括但不限于测试驱动开发（TDD）、持续集成/持续部署（CI/CD）的概念与实践、以及如何设计健壮的错误处理和日志系统。