8月20号 星期二

1.今天做了什么

今天我学习了产品经理的工作内容和技能要求，了解了产品经理的基础知识

2.今天最有收获的点

对产品经理所需的多样化技能有了清晰的认识，并掌握了如何通过有效的需求分析推动产品开发。

3.今天解决了什么问题，解决方案

问题：在了解产品经理的技能要求过程中，对如何进行有效的需求分析存在困惑。

解决方案：通过查阅相关资料和案例，深入了解了需求分析的步骤，包括需求收集、需求整理、需求优先级排序和需求文档编写。具体步骤包括：

需求收集：通过用户调研、市场分析和竞品分析获取需求信息。

需求整理：将收集到的需求进行分类和整理，确定每项需求的背景和业务价值。

需求优先级排序：根据需求的重要性和紧急性进行优先级排序，使用工具如MoSCoW方法来确定哪些需求是必须的、应该的、可以的和不会做的。

需求文档编写：编写清晰详细的需求文档，包括需求背景、功能描述、用户故事和验收标准，确保开发团队理解并能准确实施需求。

4.今天还有什么问题未解决，接下来大致计划

仍需深入了解如何有效与开发团队沟通和协作，以及如何处理需求变更和冲突。接下来计划通过参加相关培训和阅读案例研究，提升在这些领域的能力。

8月21日 星期三

1. 今天做了什么

今天我了解了互联网项目管理的基本要素、常用工具以及流行的开发模式。通过学习，我掌握了 GitHub 的基本使用方法，特别是在合作开发中的应用。重点学习了如何创建和管理仓库、分支的操作以及拉取请求的处理。

2. 今天最有收获的点

掌握了 GitHub 的基本操作流程，尤其是分支管理和拉取请求的创建，这对未来的团队合作开发至关重要。

3. 今天解决了什么问题，解决方案

问题：在使用 GitHub 进行分支管理时遇到了一些困惑，特别是如何有效地管理多个分支以及处理分支合并时的冲突。

解决方案：

为了提高对 GitHub 分支管理的理解，我通过以下步骤解决了相关问题：

创建和管理分支：

创建分支：使用命令 git checkout -b <分支名> 创建新的分支，并立即切换到该分支。这样可以在不影响主分支的情况下进行开发。

切换分支：使用 git checkout <分支名> 切换到已有的分支进行工作。

删除分支：在不再需要的分支上使用 git branch -d <分支名> 删除本地分支，用 git push origin --delete <分支名> 删除远程分支。

处理分支合并：

合并分支：在主分支上使用 git merge <分支名> 合并指定的分支。合并时可能会遇到冲突，需要手动解决冲突并提交。

解决冲突：Git 会标记冲突文件，并在文件中显示冲突标记。需要手动编辑文件，删除冲突标记，保存更改，然后使用 git add <文件名> 和 git commit 提交解决后的文件。

创建拉取请求：

创建：在 GitHub 页面上选择要合并的分支，点击 “Pull request” 按钮，填写标题和描述，选择目标分支，然后提交拉取请求。

审查和合并：团队成员可以在拉取请求中进行代码审查，讨论更改，最终由有权限的人员合并拉取请求。

通过这些操作，我有效掌握了如何在团队开发中使用分支进行并行开发，并解决了分支合并过程中的冲突问题。

4. 今天还有什么问题未解决，接下来大致计划

今天主要集中在学习 GitHub 的基本操作和解决分支管理的问题。接下来，我计划进一步了解 GitHub Actions 的自动化功能，以提高开发流程的效率。同时，探索如何更好地管理大型项目中的任务和问题，特别是如何使用 GitHub Issues 跟踪和管理开发任务。

8月22日 星期四

1.今天做了什么

今天深入学习了高质量软件开发的两大方面：软件工程方法和个人代码质量提升。具体包括研究了软件工程中的质量保障流程，以及编写和审查高质量代码的最佳实践。

2.今天最有收获的点

理解到高质量软件开发不仅依赖于工程方法，也需要持续改进个人的编码能力和习惯。

3.今天解决了什么问题，解决方案

问题：在软件开发过程中，发现部分代码质量不高，导致维护困难。

解决方案：实现了一套代码审查流程以提升代码质量，包括建立代码规范、引入自动化代码检查工具和定期进行代码审查会议。通过代码规范化，自动化工具（如静态分析器）可以提前发现潜在问题，而代码审查会议则促进了团队的知识共享和最佳实践的推广。

详细说明：

代码规范：制定了一套详细的编码标准，涵盖命名规则、注释规范、函数和类的设计原则等。

自动化工具：集成了静态代码分析工具（如 SonarQube），在每次提交代码时自动进行检查，及时反馈代码问题。

代码审查：每周组织一次团队代码审查会议，讨论高质量代码的最佳实践和当前项目中的改进点。

4.今天还有什么问题未解决，接下来大致计划

目前尚未完全解决的主要问题是如何将新引入的流程有效融入现有的开发周期中。接下来的计划是调整团队工作流程，逐步实施新的审查机制，并收集反馈进行优化。

8月23日 星期五

1. 今天做了什么：

今天，我深入了解了不同测试岗位的工作职责，包括功能测试、性能测试和自动化测试等。同时，我掌握了测试用例设计的基本方法，学习了如何编写有效的测试用例以确保软件的质量。

2. 今天最有收获的点：

我学会了如何设计详细且高效的测试用例，这对提高测试覆盖率和发现潜在问题非常重要。

3. 今天解决了什么问题，解决方案：

今天，我面临的主要问题是如何将测试用例设计理论应用到实际的测试工作中。经过调研和实践，我采用以下方法来解决这个问题：

确定测试目标： 首先明确测试的目标是什么，确保测试用例能够覆盖所有重要的功能点和需求。

设计测试用例： 根据功能需求文档，设计详细的测试用例，涵盖正向测试和负向测试场景。测试用例包括输入数据、预期结果以及测试步骤。

测试用例优化： 通过将相似的测试用例合并，避免冗余，并确保测试用例的可维护性。

编写测试用例文档： 将测试用例文档化，确保团队成员能够清晰理解每个测试用例的目的和操作步骤。

4. 今天还有什么问题未解决，接下来大致计划：

虽然我已经掌握了基本的测试用例设计方法，但在实际应用中还需要进一步提升效率。接下来的计划包括：

进行更多的实际操作： 在实际项目中应用测试用例设计方法，积累更多经验。

学习自动化测试工具： 探索和学习自动化测试工具，提升测试效率。

参与团队讨论： 向经验丰富的同事请教，获取关于测试用例设计和实施的宝贵建议。

8月24日 星期六

1. 今天做了什么：

今天，我发布了校园预约系统小程序项目，采用PPT进行了展示，并认真聆听了其他组成员的项目介绍，从中获取了不少有用的信息。

2. 今天最有收获的点：

通过聆听其他项目展示，我学到了如何更有效地进行项目汇报和展示。

3. 今天解决了什么问题，解决方案：

在发布过程中，主要解决了以下问题：

问题： 确保PPT展示内容准确无误且具有吸引力，以便有效传达项目的核心功能和价值。

解决方案：

内容整理： 在发布前，对项目的核心功能、用户需求和实现方式进行了深入整理，确保PPT内容逻辑清晰。

视觉设计： 采用简洁明了的视觉设计风格，使用高质量的图表和截图来展示系统的界面和功能，提高观众的理解度。

演练： 多次演练演讲内容，调整节奏和重点，确保在实际发布时流畅自如。

收集反馈： 在发布结束后，收集了观众的反馈意见，针对提出的问题做出详细解答，并记录下来以便后续改进。

4. 今天还有什么问题未解决，接下来大致计划：

尽管发布过程顺利，但在实际演示中仍有一些细节可以优化。接下来的计划是：

进一步优化PPT内容： 根据反馈调整PPT的结构和设计。

改进项目功能： 根据观众反馈对校园预约系统进行功能改进。

安排团队讨论： 与团队成员讨论发布过程中的收获和问题，确保在未来的项目发布中进一步提升效果。

8月25日 星期日

今天放假休息

8月26日 星期一

1.今天做了什么

今天我对微信小程序的开发环境进行了初步了解，并完成了基础的小程序创建和配置。我阅读了相关文档，安装了开发工具，并创建了一个简单的“Hello World”小程序以熟悉开发流程。

2.今天最有收获的点

通过实际操作，我学会了如何使用微信开发者工具进行项目的创建和调试，并掌握了基本的开发流程和常用的开发组件。

3.今天解决了什么问题，解决方案

今天遇到的主要问题是微信开发者工具中出现的“未找到小程序代码”的错误。这一问题导致我无法进行本地调试和预览。解决方案如下

问题分析：错误提示“未找到小程序代码”表明开发工具无法定位到项目文件夹中的小程序代码。这可能是由于项目文件夹结构不正确或配置文件缺失导致的。

解决步骤：

检查项目结构：确保项目文件夹中存在 app.json、app.js 和 app.wxss 等基本文件。这些文件是小程序的核心配置文件，必须存在。

重新配置项目：

打开微信开发者工具，选择“项目”菜单中的“新建项目”。

在弹出的窗口中，确认选择了正确的项目目录，并检查是否勾选了“生成 npm”或其他必要选项。

清理缓存：在开发者工具中，选择“工具”菜单中的“清理缓存”选项，有时缓存问题会导致项目无法正确加载。

重新编译：点击“编译”按钮，重新编译项目代码。如果问题依然存在，尝试关闭并重启开发者工具。

4.今天还有什么问题未解决，接下来大致计划

当前还存在的问题是对小程序的API调用不够熟悉，具体的接口和参数使用需要进一步学习。接下来的计划是：

学习API文档：深入阅读微信小程序官方文档，特别是有关API的部分，以便更好地理解如何在项目中实现复杂功能。

实践开发：尝试实现更多功能模块，例如用户登录、数据存储等，并不断调试和优化代码。

问题解决：遇到具体问题时，通过文档查找解决方案，或向开发者社区寻求帮助。

通过今天的工作，我对微信小程序的开发流程有了更清晰的认识，接下来将继续提升开发技能，解决现有问题并扩展功能。

8月27日 星期二

1.今天做了什么

今天我在昨天创建的小程序基础上，进一步实现了数据绑定和页面跳转功能。我还探索了小程序的状态管理，完成了用户登录功能的集成，并测试了这些功能的基本效果。

2.今天最有收获的点

学会了如何利用微信小程序的状态管理实现用户数据的存储和传递，从而提高了应用的互动性和用户体验。

3.今天解决了什么问题，解决方案

问题描述：在实现用户登录功能时，遇到了一些关于用户会话管理和数据存储的问题。特别是，登录后的用户数据未能在页面间正确传递，导致用户每次页面跳转后需要重新登录。

解决方案：

问题分析：这个问题的根源在于用户登录状态和数据未被正确存储和传递。微信小程序提供了全局状态管理功能，但在实现过程中可能遗漏了相关的配置或方法调用。

解决步骤：

全局状态管理：

确保在 app.js 中正确设置了全局数据对象 App()，用于存储和管理用户信息。

在用户登录成功后，使用 wx.setStorageSync() 将用户信息存储到本地缓存中。

数据传递：

在需要获取用户信息的页面，使用 wx.getStorageSync() 从本地缓存中读取用户数据。

确保在页面的 onLoad 方法中调用 wx.getStorageSync()，并在页面数据渲染前完成数据加载。

页面跳转：

在页面跳转时，使用 wx.navigateTo() 或 wx.redirectTo() 方法，并确保跳转前检查和设置必要的参数。

确保在目标页面加载时，正确处理数据传递和状态恢复。

4.今天还有什么问题未解决，接下来大致计划

目前还未完全掌握微信小程序的云开发能力，特别是在数据存储和查询方面。接下来的计划是：

学习云开发文档：深入阅读微信小程序的云开发文档，了解如何利用云函数和数据库进行更复杂的数据操作。

实现云功能：尝试在小程序中实现云数据存储和查询功能，提升应用的扩展性。

优化功能：基于当前进展，优化现有功能的代码，提升用户体验和应用性能。

8月28日 星期三

1.今天做了什么

今天我在小程序中实现了用户登录功能，并集成了数据存储API。通过实际编码，我测试了用户授权和数据存取的流程，同时优化了之前写的代码。

2.今天最有收获的点

通过实现用户登录和数据存储，我深入理解了微信小程序的用户认证和数据管理机制。

3.今天解决了什么问题，解决方案

今天遇到的主要问题是微信小程序的用户登录功能无法正确获取用户信息。具体问题如下：

问题描述：在用户点击登录按钮后，获取用户信息的接口总是返回空数据，导致无法正确显示用户的基本信息。

解决步骤：

检查接口调用：

确认 wx.login() 和 wx.getUserInfo() 方法调用是否正确。

确保在调用 wx.getUserInfo() 之前已经成功调用了 wx.login()。

授权设置：

检查小程序的授权设置，确保在微信公众平台中配置了正确的用户权限。

在小程序中，使用 wx.getSetting() 方法检查用户是否已授权获取用户信息。

错误处理：

添加错误处理逻辑，检查接口返回的错误信息，并对可能的错误进行处理。例如，捕获 wx.getUserInfo() 返回的错误码，确定具体原因。

测试和调试：

在微信开发者工具中调试代码，确保接口调用和数据处理逻辑没有遗漏。

使用 console.log() 输出调试信息，检查接口调用的返回结果。

4.今天还有什么问题未解决，接下来大致计划

目前尚未解决的问题是数据存储的同步机制，特别是在高并发情况下的数据一致性问题。接下来的计划是：

研究数据存储机制：深入学习微信小程序的云开发功能和数据同步策略。

优化代码：提升代码的性能和稳定性，确保在不同设备和网络条件下都能顺利运行。

实施功能测试：对已实现的功能进行全面测试，确保没有潜在的bug。

8月29日 星期四

1.今天做了什么

今天我在小程序中实现了用户数据的动态更新功能，并优化了界面响应速度。同时，完成了与微信云开发的集成测试。

2.今天最有收获的点

深入理解了微信云开发中数据动态更新的实现细节和最佳实践。

3.今天解决了什么问题，解决方案

今天遇到的问题是用户数据在实时更新时显示不一致。问题具体描述如下：

问题描述：在小程序中，当用户修改个人信息时，更新后的数据在某些情况下未能及时反映在界面上。用户需要刷新页面才能看到最新信息。

解决步骤：

确认数据更新：

使用 wx.cloud.database() 查询数据并确认云端数据是否成功更新。

在数据修改后，调用 db.collection('users').doc(userId).update() 方法更新用户信息。

界面刷新机制：

确认在数据更新后，界面是否有相应的刷新机制。例如，使用 this.setData() 更新页面数据。

确保使用 wx.cloud.callFunction() 后，页面能及时反映数据更改。

使用实时数据监听：

实现数据的实时监听功能，以确保用户数据变化时页面自动更新。

使用 db.collection('users').watch() 方法监控数据变化，确保数据在客户端的即时同步。

调试和测试：

在不同网络条件和设备上测试数据更新的及时性，确保功能稳定。

使用微信开发者工具的调试功能，检查数据流和界面更新的执行情况。

4.今天还有什么问题未解决，接下来大致计划

目前的问题是处理高并发情况下的数据一致性问题。接下来的计划是：

研究高并发处理方案：深入学习微信云开发中的数据一致性和冲突处理机制。

优化数据刷新逻辑：进一步优化数据实时更新的逻辑，确保在所有设备上表现一致。

扩展功能测试：增加用户操作场景的测试，以确保在各种情况下数据都能正确更新。