**综合实验：Git & GitHub 协同开发与版本控制实践**

**背景**：

你和你的同学正在合作开发一个名为“CollaborativeProject”的Web应用程序。为了管理项目的版本和协作流程，你们决定使用Git作为版本控制工具，并将代码托管在GitHub上。在这个项目中，你们将实践分支开发、合并代码、两地协同工作、两人协作以及使用标签标记重要版本。

**任务**：

1. **初始化与设置**：
   1. 在你的本地计算机上初始化一个新的Git仓库，并创建一个main分支作为主分支。
   2. 在GitHub上创建一个新的公开仓库，命名为“CollaborativeProject”，并将本地仓库与远程仓库关联。
2. **分支开发与合并**：
   1. 创建一个名为branchA的分支，用于开发用户登录功能。
   2. 在branchA分支上完成功能开发后，将其合并到main分支，并解决可能出现的冲突。
   3. 将main分支的最新代码推送到远程仓库。
3. **两人协作**：
   1. 邀请你的同学（假设GitHub用户名为Colleague）作为协作者加入你的GitHub仓库。
   2. 你的同学需要克隆远程仓库，并在本地创建一个名为branchB的分支，用于开发用户资料功能。
   3. 同学在branchB分支上完成功能开发后，将其推送到远程仓库。
   4. 你将同学的branchB分支合并到你的本地main分支，并处理任何冲突。
4. **两地协同**：
   1. 假设你需要在另一台计算机上继续开发。在这台新计算机上，克隆远程仓库，并拉取最新的main分支代码。
   2. 在新计算机上，基于main分支创建一个新的branchC分支，用于开发仪表盘功能。
   3. 完成branchC分支的开发后，将其推送到远程仓库，并合并到main分支。
5. **使用标签**：
   1. 在项目的某个重要里程碑（如完成用户登录功能）时，为main分支创建一个标签（tag），命名为v1.0。
   2. 将v1.0标签推送到远程仓库。
6. **总结与提交**：
   1. 提交你的GitHub仓库链接作为答案的一部分。
   2. 描述在分支开发、合并、两人协作、两地协同以及使用标签过程中遇到的挑战和解决方案。
   3. 总结这次实践对你理解Git和GitHub协同开发与版本控制流程的帮助。

**要求**：

* 每次分支开发完成后，都需要将代码合并到main分支，并确保代码能够正常工作。
* 在合并分支时，需要妥善处理冲突，并保留所有有效的更改。
* 两人协作时，确保代码同步，避免重复工作，并及时沟通。
* 在两地协同工作时，确保能够无缝切换工作环境，并保持代码的一致性。
* 使用标签来标记项目的重要版本，方便后续的版本管理和回滚。

**注意**：为了完成这个编程题目，你需要熟练掌握Git的基本操作，包括初始化仓库、创建分支、提交更改、推送和拉取代码等。同时，你还需要了解GitHub的使用，包括创建仓库、添加协作者、管理分支和标签等。