南開大學
Nankai University

计算机学院

软件工程实验报告

项目管理

姓名：杨馨仪

学号：2011440

专业：计算机科学与技术

2024 年 6 月 24 日

目录

1 实验要求

- 熟练掌握 git 的基本指令和分支管理指令
- 掌握 git 支持软件配置管理的核心机理
- 在实践项目中使用 git/github 管理自己的项目源代码
- 本次实验由个人单独完成（使用团队作业的文件，但仓库需要个人新建）

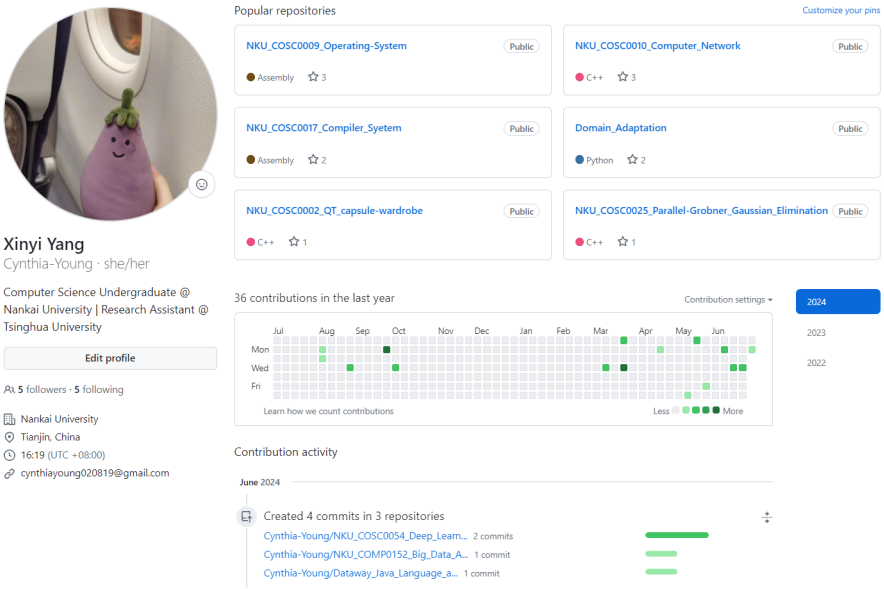
2 安装 Git

2.1 本地机器上安装 Git

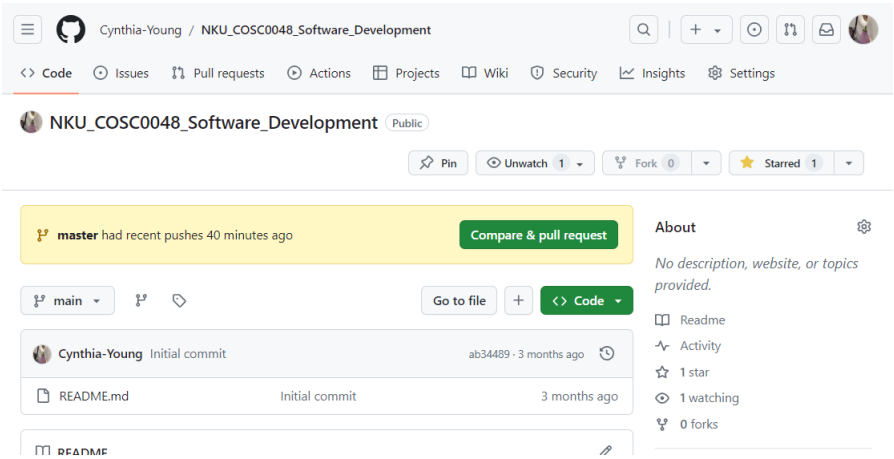
git 版本号： git version 2.35.1.windows.2

2.2 申请 github 账号

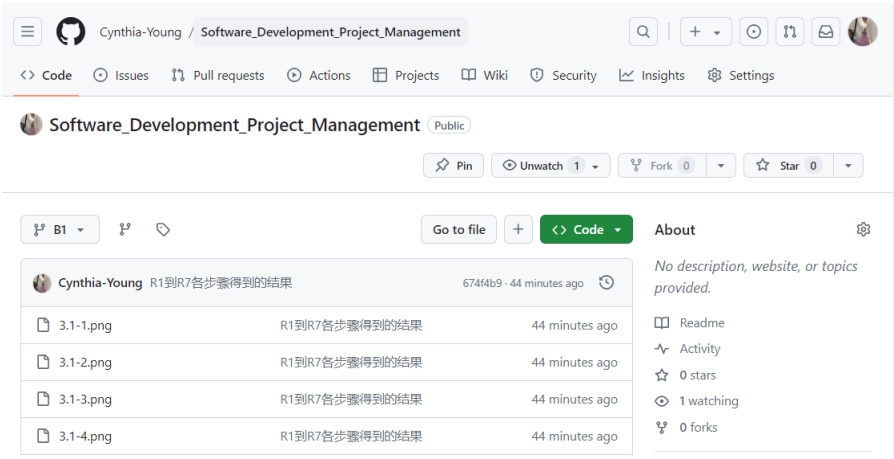
github 账号： Cynthia-Young



项目 URL 地址：实验场景（1）的仓库使用的是 https://github.com/Cynthia-Young/NKU_COSC0048_Software_Development/tree/master



实验场景(2)的仓库使用的是https://github.com/Cynthia-Young/Software_Development_Project_Management/tree/B1



3 Git 操作过程

3.1 实验场景（1）：仓库创建与提交

R0：查看状态

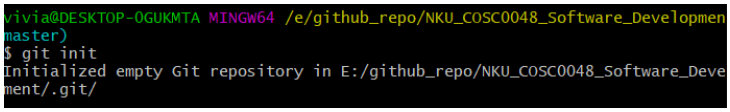
针对 R1 和 R7，在进行每次 git 操作之前，随时查看工作区、暂存区、git 仓库的状态，确认项目里的各文件当前处于什么状态。

```
1 git status
```

R1：初始化 Git 仓库

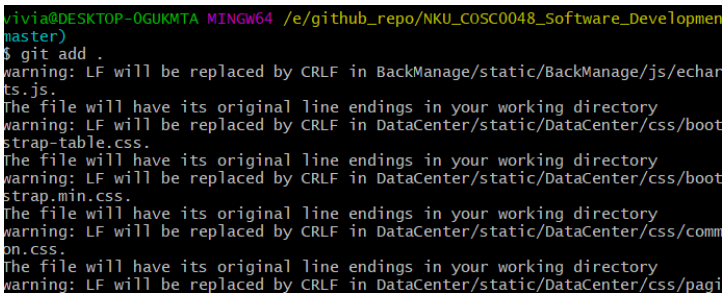
1. **创建本地 Git 仓库：**进入项目文件夹，执行以下命令初始化 Git 仓库：

```
git init
```



2. **将文件加入 Git 管理：**将项目中的所有源文件加入 Git 管理：

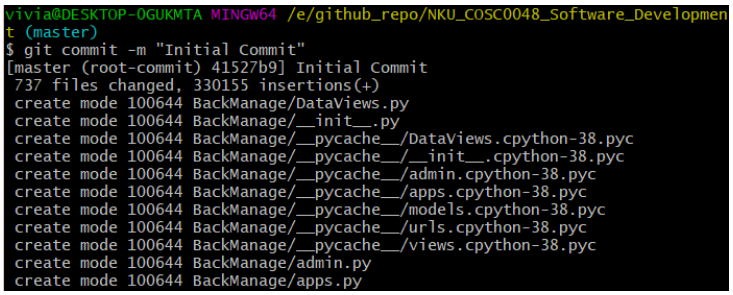
```
git add .
```



R2：提交

提交所做的更改：

```
git commit -m "Initial commit"
```



R3：查看修改内容

手动对团队作业中的三个文件进行修改，然后使用以下命令查看自上次提交以来的修改内容：

```
git diff
```

```
warning: LF will be replaced by CRLF in UnderWater/templates/UnderWater/under
water.html.
The file will have its original line endings in your working directory
diff --git a/BackManage/views.py b/BackManage/views.py
index b61de04..b62fb85 100644
--- a/BackManage/views.py
+++ b/BackManage/views.py
@@ -18,8 +18,8 @@ def BackManage(request):
             device.status = '正常'
         elif device.state == 1:
             device.status = '故障'
-        else:
-            device.status = '未知'
+        # else:
+        #     device.status = '未知'
     context = {'users': users, 'devices': devices}  # 数据字典
     return render(request, 'BackManage/chronic.html', context)

diff --git a/MainInformation/apps.py b/MainInformation/apps.py
index 8d54617..c0ef419 100644
--- a/MainInformation/apps.py
+++ b/MainInformation/apps.py
@@ -3,4 +3,4 @@ from django.apps import AppConfig

 class MaininformationConfig(AppConfig):
     default_auto_field = "django.db.models.BigAutoField"
-    name = "MainInformation"
+    name = "MainInformation-change!!!"
diff --git a/UnderWater/templates/UnderWater/underwater.html b/UnderWater/tem
plates/UnderWater/underwater.html
index c52993a..d3f1b7e 100644
--- a/UnderWater/templates/UnderWater/underwater.html
+++ b/UnderWater/templates/UnderWater/underwater.html
@@ -3,7 +3,7 @@
 <html lang="en">
 <head>
     <meta charset="UTF-8">
-    <title>海洋牧场可视化系统</title>
+    <title>海洋牧场可视化监控系统</title>
     <script src="{% static 'UnderWater/js/Plugin/jquery-3.3.1.min.js' %}"></
script>
    <script src="{% static 'UnderWater/js/Plugin/echarts.min.js' %}"></scrip
t>
    <script src="{% static 'UnderWater/js/common.js' %}"></script>
```

R4：重新提交

将修改过的文件重新提交：

```
git add .
git commit -m "modified"
```

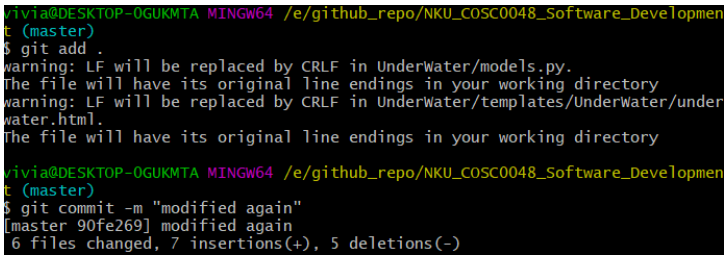
```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/NKU_COSC0048_Software_Developmen
t (master)
$ git add .
warning: LF will be replaced by CRLF in UnderWater/templates/UnderWater/under
water.html.
The file will have its original line endings in your working directory

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/NKU_COSC0048_Software_Developmen
t (master)
$ git commit -m "modified"
[master c1a389a] modified
 3 files changed, 4 insertions(+), 4 deletions(-)
```

R5：再次修改和提交

再次手动对这三个文件进行修改并提交：

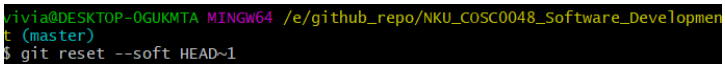
```
git add .
git commit -m "modified again"
```



R6：撤销最后一次提交

撤销最后一次提交：

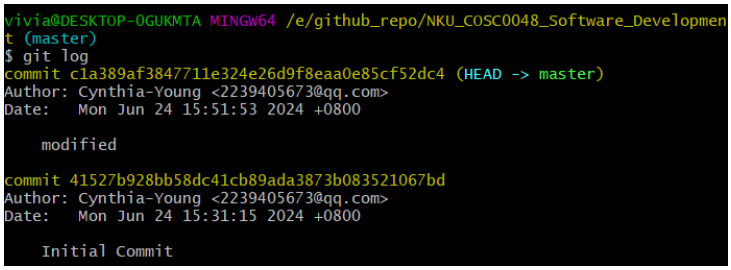
```
git reset --soft HEAD~1
```



R7：查询提交记录

查询提交记录：

```
git log
```



3.2 实验场景（2）：分支管理

在 Github 上建立项目

1. **创建远程仓库：**在 Github 上创建一个新的仓库。
2. **添加文件：**通过 Web 界面向仓库中添加不少于 10 个文件（程序代码、文档等），形成初始分支 B1。
3. **创建分支：**基于 B1 创建两个并行分支 B2 和 B3，并在各自分支上对文件进行不同程度的修改并提交。

R8：克隆已有仓库

从 Github 上克隆一个已有的 Git 仓库到本地：

```
git clone <repository_url>
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo (master)
$ git clone https://github.com/Cynthia-Young/Software_Development_Project_Manage
ment.git
Cloning into 'Software_Development_Project_Management'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 36 (delta 14), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (36/36), 15.31 KiB | 402.00 KiB/s, done.
Resolving deltas: 100% (14/14), done.
```

R9：获得全部分支

获得该仓库的所有分支：

```
cd <repository_name>
git branch -r
git fetch --all
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo (master)
$ cd Software_Development_Project_Management

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git branch -r
  origin/B1
  origin/B2
  origin/B3
  origin/HEAD -> origin/B1

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git fetch --all
Fetching origin
```

R10：创建新分支 C4

在 B2 分支基础上创建一个新分支 C4：

```
git checkout B2
git checkout -b C4
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git checkout B2
Switched to a new branch 'B2'
branch 'B2' set up to track 'origin/B2'.

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B2)
$ git checkout -b C4
Switched to a new branch 'C4'
```

R11：修改 C4 文件并提交

在 C4 分支上，对四个文件进行修改并提交：

```
git add .
git commit -m "Modified four files on C4"
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git add .

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git commit -m "Modified four files on C4"
[C4 078abc6] Modified four files on C4
 4 files changed, 8 insertions(+)
```

R12：在 B3 分支上修改并提交

在 B3 分支上对同样的四个文件做不同修改并提交：

```
git checkout B3
git add .
git commit -m "Modified four files on B3"
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git checkout B3
Switched to branch 'B3'
Your branch is up to date with 'origin/B3'.

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3)
$ git add .

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3)
$ git commit -m "Modified four files on B3"
[B3 2320527] Modified four files on B3
 4 files changed, 6 insertions(+), 1 deletion(-)
```

R13：合并分支并解决冲突

将 C4 和 B3 分支合并，若有冲突，手工解决：

```
git checkout B3
git merge C4
# 若有冲突，手动解决后
git add .
git commit -m "Resolved conflicts and merged C4 into B3"
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3)
$ git merge C4
Auto-merging DataViews.py
CONFLICT (content): Merge conflict in DataViews.py
Auto-merging admin.py
CONFLICT (content): Merge conflict in admin.py
Auto-merging apps.py
CONFLICT (content): Merge conflict in apps.py
Auto-merging models.py
Automatic merge failed; fix conflicts and then commit the result.
```

出现冲突，手工解决后，如下图：

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3|MERGING)
$ git add .

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3|MERGING)
$ git commit -m "Resolved conflicts and merge C4 into B3"
[B3 16a4d9f] Resolved conflicts and merge C4 into B3
```

R14：查看分支合并情况

查看目前哪些分支已经合并、哪些分支尚未合并：

```
git branch --merged
git branch --no-merged
```

首先查看 B3 分支上，是哪些分支合并的结果，以及 B3 还没有和哪些分支进行合并。如下图可知，B3 是全部分支合并后的结果，没有分支还未与其合并。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3)
$ git branch --merged
  B1
  B2
* B3
  C4

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B3)
$ git branch --no-merged
```

然后查看 B1 分支，如下图可知，B1 还没有和任何分支合并。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git branch --merged
* B1

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git branch --no-merged
  B2
  B3
  C4
```

接着查看 B2 分支，如下图可知，B2 还未与 B3 和 C4 分支进行合并。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B2)
$ git branch --merged
  B1
* B2

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B2)
$ git branch --no-merged
  B3
  C4
```

最后查看 C4 分支，其未与 B3 进行合并。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git branch --merged
  B1
  B2
* C4

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git branch --no-merged
  B3
```

R15：删除合并分支并创建新分支

删除 C4 和 B3 合并后的分支，将尚未合并的分支合并到一个新分支，分支名字为学号：

```
1 git branch -d B3
2 git checkout -b <your_student_id>
3 git merge B2
```

由于 C4 和 B3 的合并结果发生在 B3 上，因此对 B3 分支进行了删除处理。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (C4)
$ git branch -D B3
Deleted branch B3 (was 16a4d9f).
```

没有尝试过合并的分支上 B1 和 B2 分支，因此将它们合并到一个新分支。

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (B1)
$ git checkout -b 2011440
Switched to a new branch '2011440'

vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (2011440)
$ git merge B2
Updating 7202563..67e1d03
Fast-forward
 DataViews.py |  4 ++--
 apps.py      |  2 +-
 quota.html   | 10 ++++++++++
 trend.html   | 24 +----------------------
 4 files changed, 14 insertions(+), 26 deletions(-)
```

3.3 实验场景（3）：远程分支管理

R16：推送分支到 Github

将本地以学号命名的分支推送到 Github 上：

```
1  git push origin <your_student_id>
```

```
vivia@DESKTOP-OGUKMTA MINGW64 /e/github_repo/Software_Development_Project_Manage
ment (2011440)
$ git push origin 2011440
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for '2011440' on GitHub by visiting:
remote:      https://github.com/Cynthia-Young/Software_Development_Project_Manag
ement/pull/new/2011440
remote:
To https://github.com/Cynthia-Young/Software_Development_Project_Management.git
 * [new branch]      2011440 -> 2011440
```

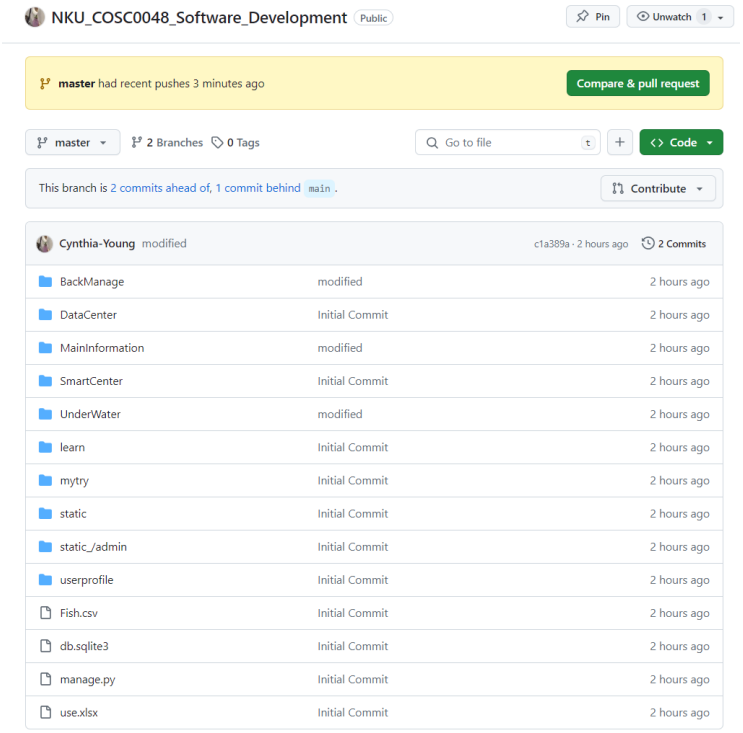
R17：推送初始结果到 Github

将 R1 到 R7 各步骤得到的结果推送到 Github 上：

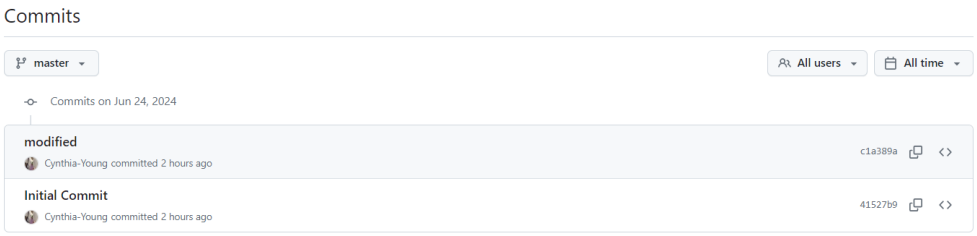
```
1  git push origin master
```

R18：通过 Web 界面查看仓库状态

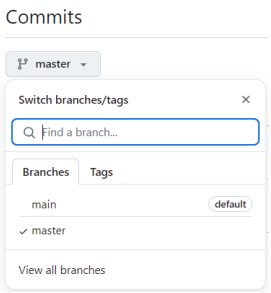
实验场景（1）的仓库使用的是 https://github.com/Cynthia-Young/NKU_COSC0048_Software_Development/tree/master，本地文件上传到了 master 分支。其文件内容显示如下：



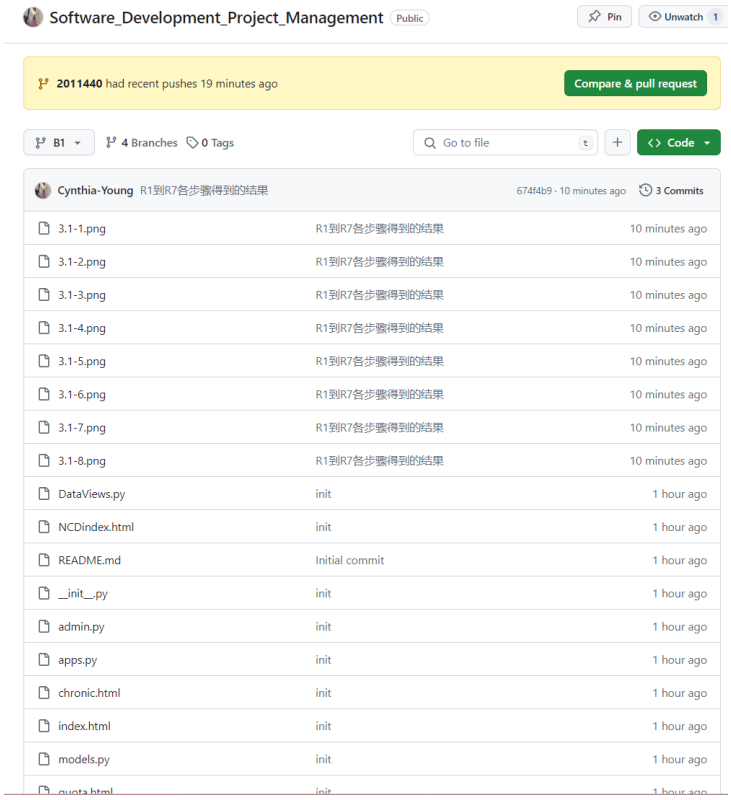
下图显示的是 master 分支下的 commit 记录：



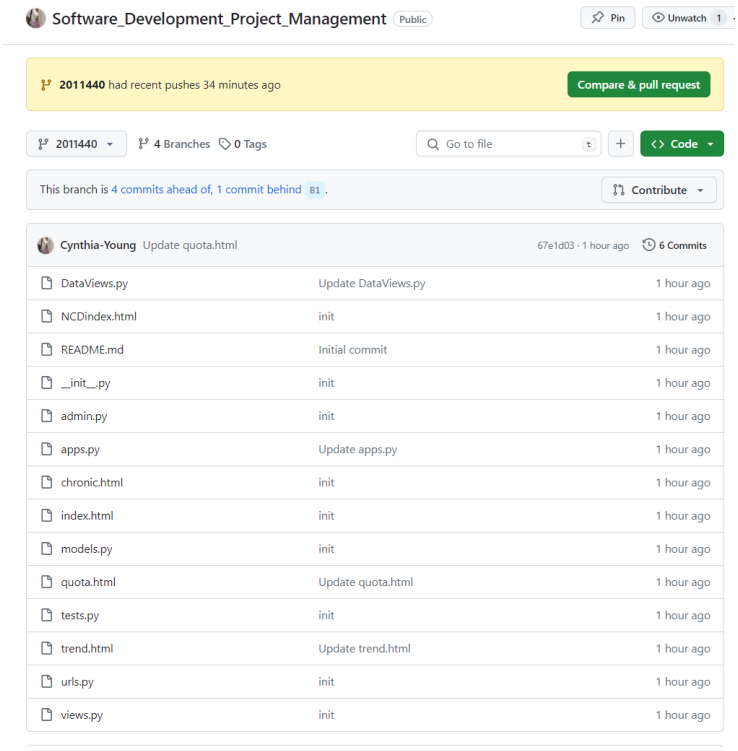
下图展示了该仓库的分支情况：



实验场景（2）的仓库使用的是 https://github.com/Cynthia-Young/Software_Development_Project_Management/tree/B1，在场景（2）下的操作皆为该仓库对应的本地文件上的操作，并未推送到 Github 上。但在实验场景（3）下，把实验结果推送到了该仓库的 B1 分支，把学号命名的分支推送到了 github 上。因此可看到 B1 分支下的 github 远程仓库文件如下图所示：



2011440 分支下的 github 远程仓库文件如下图所示：



4 小结

对本次实验过程和结果的思考：

- **Q:** 比较之前的开发经验，使用 git 的优点?

  A: 在没有使用 git 之前，开发过程中常常面临代码备份困难、历史版本管理混乱等问题。引入 Git 后，能够轻松管理项目的版本历史，追踪每次修改，便于回溯和错误修复。Git 的分支管理还能使得同时开发多个功能或修复不同版本的问题更为便捷，而且可以在分布式环境下高效工作，保证数据的安全性和完整性。

- **Q:** 在个人开发和团队开发中，git 起到的作用有何主要差异?

  A: 在个人开发中，Git 主要用于自我管理和版本控制，帮助开发者记录和追溯个人项目的变化。而在团队开发中，Git 的分支管理和协作功能尤为重要，团队成员可以独立地开发和测试功能，通过 Pull Request 进行代码审查和集成，保证代码的质量和一致性，同时提升整体开发效率。

- **Q:** 之前是否用过其他的版本控制软件？如果有，同 git 相比有哪些优缺点？如果没有查阅资料对比一下不同版本控制系统的差别。

  A: 之前没有使用过其他的版本控制软件。

  SVN 是另一个集中式版本控制系统。SVN 在简单权限管理和中央化管理上有优势，适合需要严格控制访问权限和统一管理的企业项目。然而，SVN 在分支管理和离线工作支持上不如 Git 灵活，操作复杂且易产生冲突。Git 通过分布式架构和高效的分支管理解决了这些问题，但使用起来有一定的学习曲线。

- **Q:** 在什么情况下适合使用 git、什么情况下没必要使用 git?

  A: Git 适合于需要频繁版本控制、多人协作开发和分布式工作环境的项目。特别是开源项目和需要持续集成的团队开发中，Git 的优势尤为明显。然而，对于简单的个人项目、主要是文档或临时性文件的管理，Git 的复杂性可能显得不必要，可以考虑使用简单的备份工具或文档管理系统来替代。