Here's a more detailed version of the GitHub repository structure and content based on the provided resources for \*\*Version Control\*\* and \*\*Issue Tracking\*\* labs. I'll elaborate on each part, adding more detail and making it easier to follow for a class or lab session.

### 1. \*\*Create the GitHub Repository\*\*

1. Go to [GitHub](https://github.com/).

2. Log in with your credentials.

3. Click on \*\*New\*\* to create a new repository.

4. Name it: \*\*CMPS310-VersionControl-IssueTracking-Lab\*\*.

5. Check the option to initialize the repository with a \*\*README.md\*\* file.

6. Choose \*\*MIT License\*\* (or any other open-source license, if needed).

7. Create the repository.

---

### 2. \*\*Folder Structure\*\*

The following folder structure will help you organize the labs:

├── README.md

├── VersionControl

│ ├── Lab Instructions.md

│ ├── Tasks

│ │ ├── Task1.md

│ │ ├── Task2.md

│ │ ├── Task3.md

│ └── Solutions

│ ├── Solution1.md

│ ├── Solution2.md

│ ├── Solution3.md

├── IssueTracking

│ ├── Lab Instructions.md

│ ├── Tasks

│ │ ├── Task1.md

│ │ ├── Task2.md

│ └── Solutions

│ ├── Solution1.md

│ ├── Solution2.md

└── Resources

└── references.md

### 3. \*\*Detailed Content for Each Section\*\*

#### a. \*\*Version Control (`02-Version Control`)\*\*

\*\*Folder\*\*: `VersionControl/Lab Instructions.md`

markdown

# Lab 2: Version Control

## Objectives:

- Learn to use Git for version control in collaborative software projects.

- Perform common Git operations: clone, commit, push, pull, merge, and branch.

- Understand how to resolve conflicts and keep the repository up-to-date.

## Overview:

This lab covers the essentials of Git and GitHub to manage project versions. Each task will guide you through key Git operations. Follow the instructions in the respective task files to complete the exercises.

### Lab Tasks:

1. \*\*Task 1: Cloning a Repository\*\*

2. \*\*Task 2: Making Changes and Committing\*\*

3. \*\*Task 3: Branching, Merging, and Resolving Conflicts\*\*

\*\*Folder\*\*: `VersionControl/Tasks/Task1.md`

markdown

# Task 1: Cloning a Repository

## Goal:

Clone an existing GitHub repository and set it up locally for development.

## Instructions:

1. Open your terminal (or Git Bash on Windows).

2. Navigate to the directory where you want to clone the repository:

bash

cd path/to/your/folder

3. Run the following command to clone the repository:

bash

git clone https://github.com/yourusername/CMPS310-VersionControl-IssueTracking-Lab.git

4. Check that the repository is cloned successfully:

bash

ls

5. Navigate into the cloned repository:

bash

cd CMPS310-VersionControl-IssueTracking-Lab

6. List the files to confirm:

bash

ls

## Expected Outcome:

You should see the files and folders of the repository locally on your machine.

\*\*Folder\*\*: `VersionControl/Tasks/Task2.md`

markdown

# Task 2: Making Changes and Committing

## Goal:

Make changes to a file, commit the changes, and push them to the remote repository.

## Instructions:

1. Open any file in the cloned repository, or create a new file called `test.txt`:

bash

echo "This is a test file." > test.txt

2. Check the status of your repository to see the new file:

bash

git status

3. Add the new file to the staging area:

bash

git add test.txt

4. Commit the changes with a descriptive message:

bash

git commit -m "Added test.txt file for testing purposes"

5. Push the changes to the remote GitHub repository:

bash

git push origin main

6. Verify the file is pushed by checking your GitHub repository online.

## Expected Outcome:

You should see the newly created `test.txt` file in the GitHub repository.

\*\*Folder\*\*: `VersionControl/Tasks/Task3.md`

markdown

# Task 3: Branching, Merging, and Resolving Conflicts

## Goal:

Learn to create branches, make changes, merge them into the main branch, and resolve conflicts.

## Instructions:

1. Create a new branch:

bash

git checkout -b feature-branch

2. Make changes to the `README.md` file (for example, add a new section).

3. Add and commit the changes:

bash

git add README.md

git commit -m "Updated README on feature branch"

4. Switch back to the main branch:

bash

git checkout main

5. Merge the feature branch into the main branch:

bash

git merge feature-branch

6. If there are merge conflicts, resolve them manually by editing the files, then add and commit the resolved changes:

bash

git add .

git commit -m "Resolved merge conflict in README.md"

## Expected Outcome:

You should see the changes from the feature branch merged into the main branch, with any conflicts resolved.

#### b. \*\*Issue Tracking (`03-Issue Tracking`)\*\*

\*\*Folder\*\*: `IssueTracking/Lab Instructions.md`

markdown

# Lab 3: Issue Tracking

## Objectives:

- Understand how to manage issues in GitHub.

- Learn how to create, assign, and close issues.

- Use labels and milestones to organize issues effectively.

## Overview:

This lab introduces issue tracking using GitHub. You will create and manage issues, assign them to team members, and link them with code changes. The tasks will guide you step-by-step through issue creation and management.

### Lab Tasks:

1. \*\*Task 1: Creating an Issue\*\*

2. \*\*Task 2: Assigning and Closing Issues\*\*

\*\*Folder\*\*: `IssueTracking/Tasks/Task1.md`

markdown

# Task 1: Creating an Issue

## Goal:

Create a new issue in GitHub for tracking a feature or bug.

## Instructions:

1. Go to your GitHub repository.

2. Click on the \*\*Issues\*\* tab.

3. Click \*\*New Issue\*\*.

4. Enter a title and description for the issue (e.g., "Add user login feature").

5. Add any relevant labels (e.g., `enhancement`, `bug`).

6. Assign the issue to yourself or a team member.

7. Submit the issue.

## Expected Outcome:

You should see the new issue listed under the \*\*Issues\*\* tab with the assigned labels and assignees.

\*\*Folder\*\*: `IssueTracking/Tasks/Task2.md`

markdown

# Task 2: Assigning and Closing Issues

## Goal:

Assign an issue to a collaborator and close it when done.

## Instructions:

1. Navigate to an open issue in your repository.

2. On the right-hand side, click \*\*Assignees\*\* and assign the issue to a team member.

3. Once the issue is resolved, commit the related code and reference the issue number in the commit message:

bash

git commit -m "Fixed login bug, resolves #1"

4. Push the changes to the remote repository:

bash

git push origin main

5. Verify that the issue is automatically closed in GitHub.

## Expected Outcome:

The issue should be marked as \*\*Closed\*\* in the repository, and the related commit should be linked to the issue.

### 4. \*\*Adding Resources\*\*

\*\*Folder\*\*: `Resources/references.md`

markdown

# References

1. [Git Documentation](https://git-scm.com/doc)

2. [GitHub Issue Tracking Guide](https://guides.github.com/features/issues/)

3. [Version Control with Git - Atlassian](https://www.atlassian.com/git/tutorials)

4. [Git Branching and Merging](https://www.git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging)

---

### 5. \*\*Uploading and Maintaining the Repository\*\*

After creating this structure on your local system, push the entire directory to GitHub:

bash

git init

git add .

git commit -m "Initial lab setup with detailed tasks"

git remote add origin https://github.com/yourusername/CMPS310-VersionControl-IssueTracking-Lab.git

git push -u origin main

**Exercise 1**

1. Clone your repository using GitHub Desktop.
2. Create a 01-initiation directory under your repository.
3. Move all the files that you have created previously under 01-initiation.
4. Use GitHub Desktop to
   1. stage your changes,
   2. commit them using a descriptive message, and
   3. push your work.

**Structure**

