**UNIVERSITY OF THE PUNJAB**

**Assignment # 1**

Introduction to Data Science

**Topic :-**

Git and GitHub

**Submitted by :-**

Muhammad Umar Shafiq

**Submitted to :-**

Dr. Muhammad Nadeem Majeed

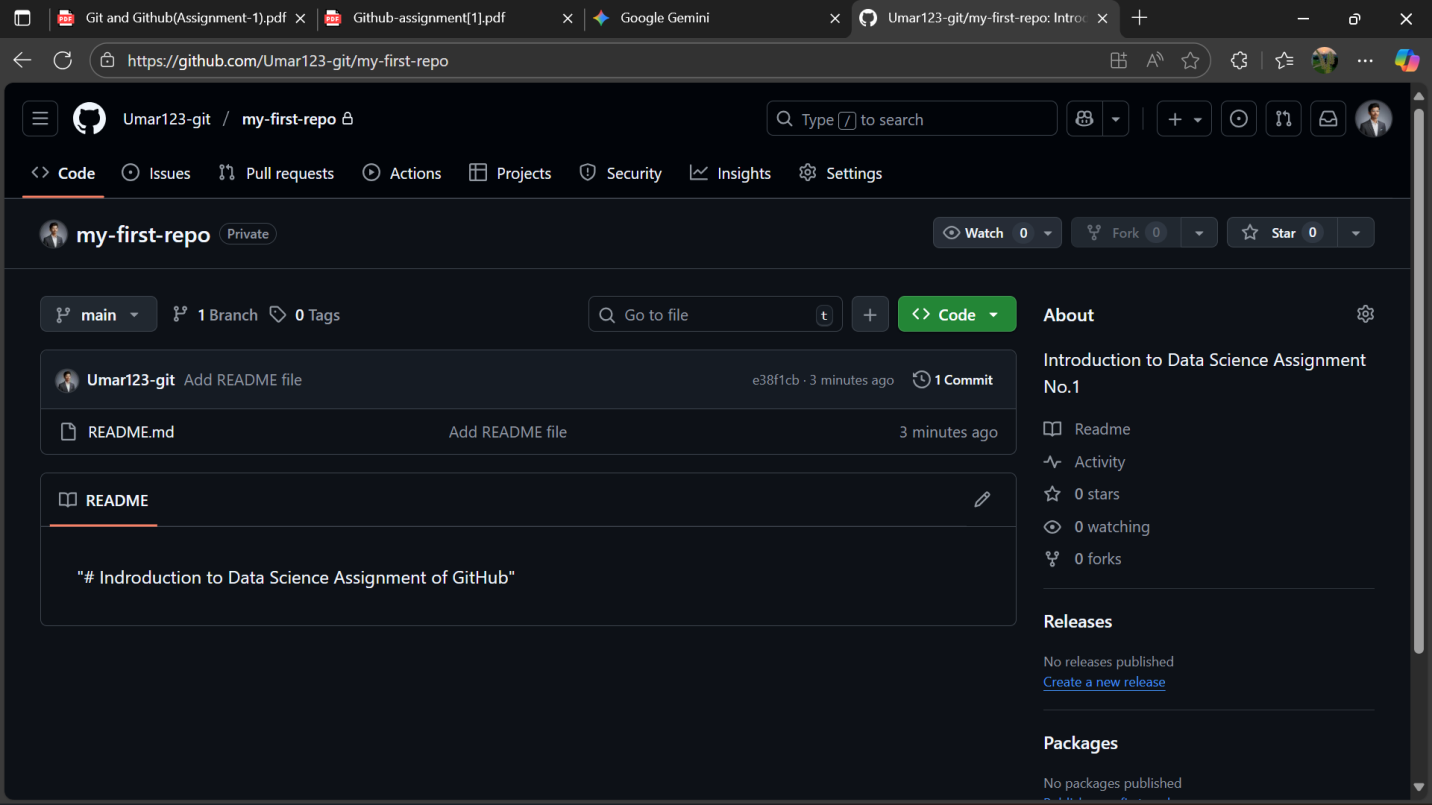
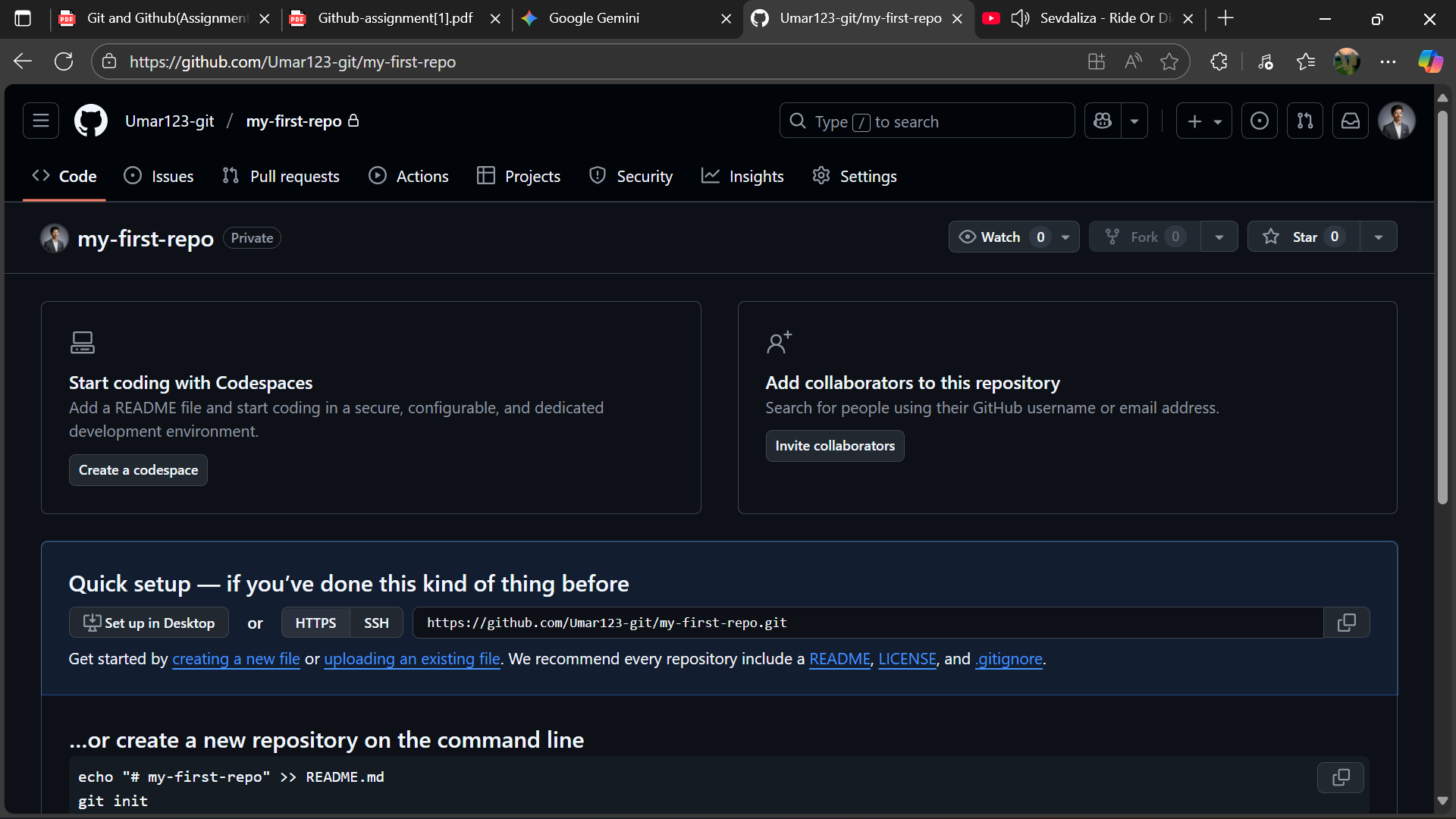
**Student Roll No. :-**

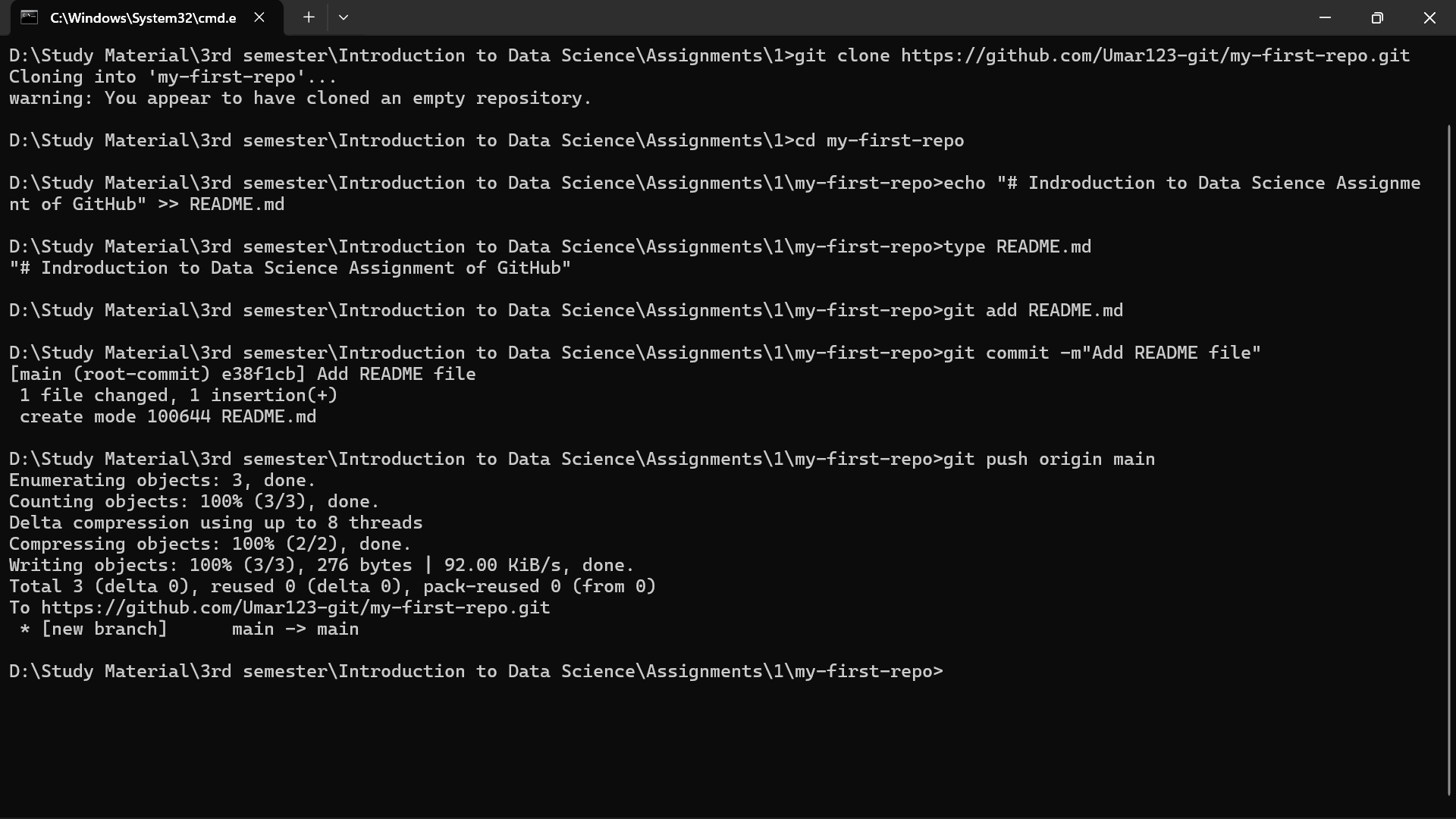
BSDSF24A050

**Department :-**

BS(DS) 3rd Semester 2024

**TASK 1**

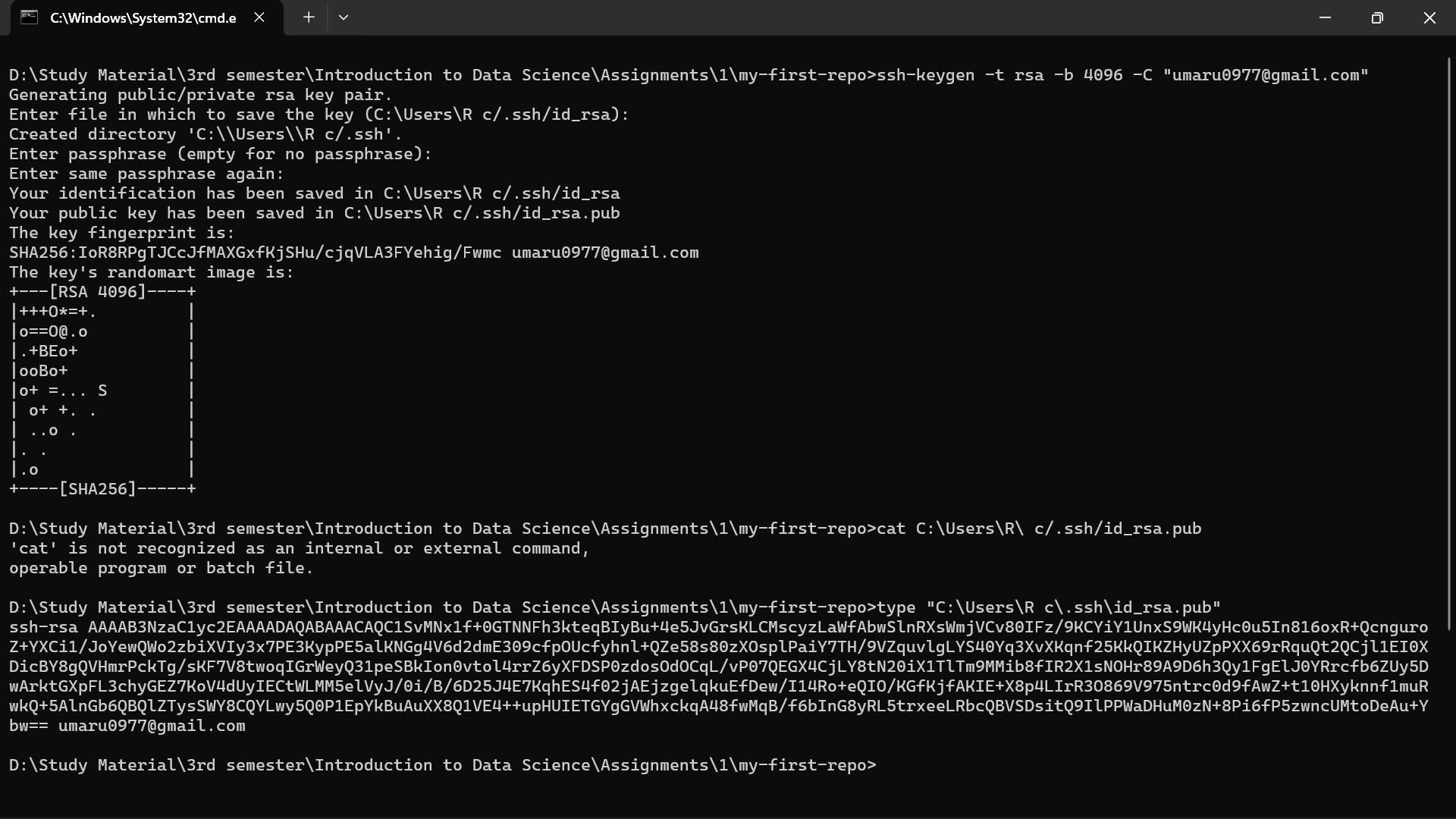
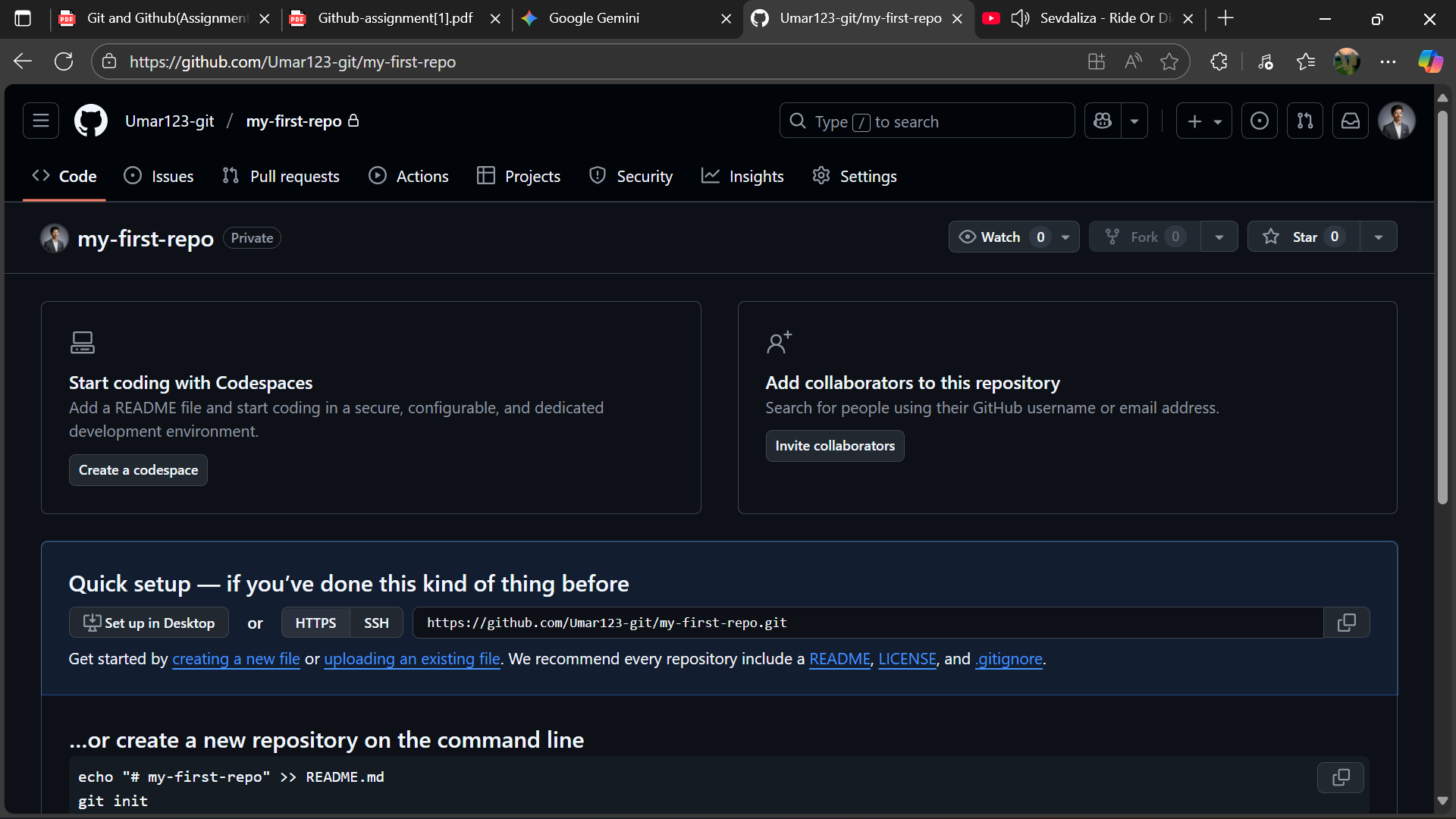
****

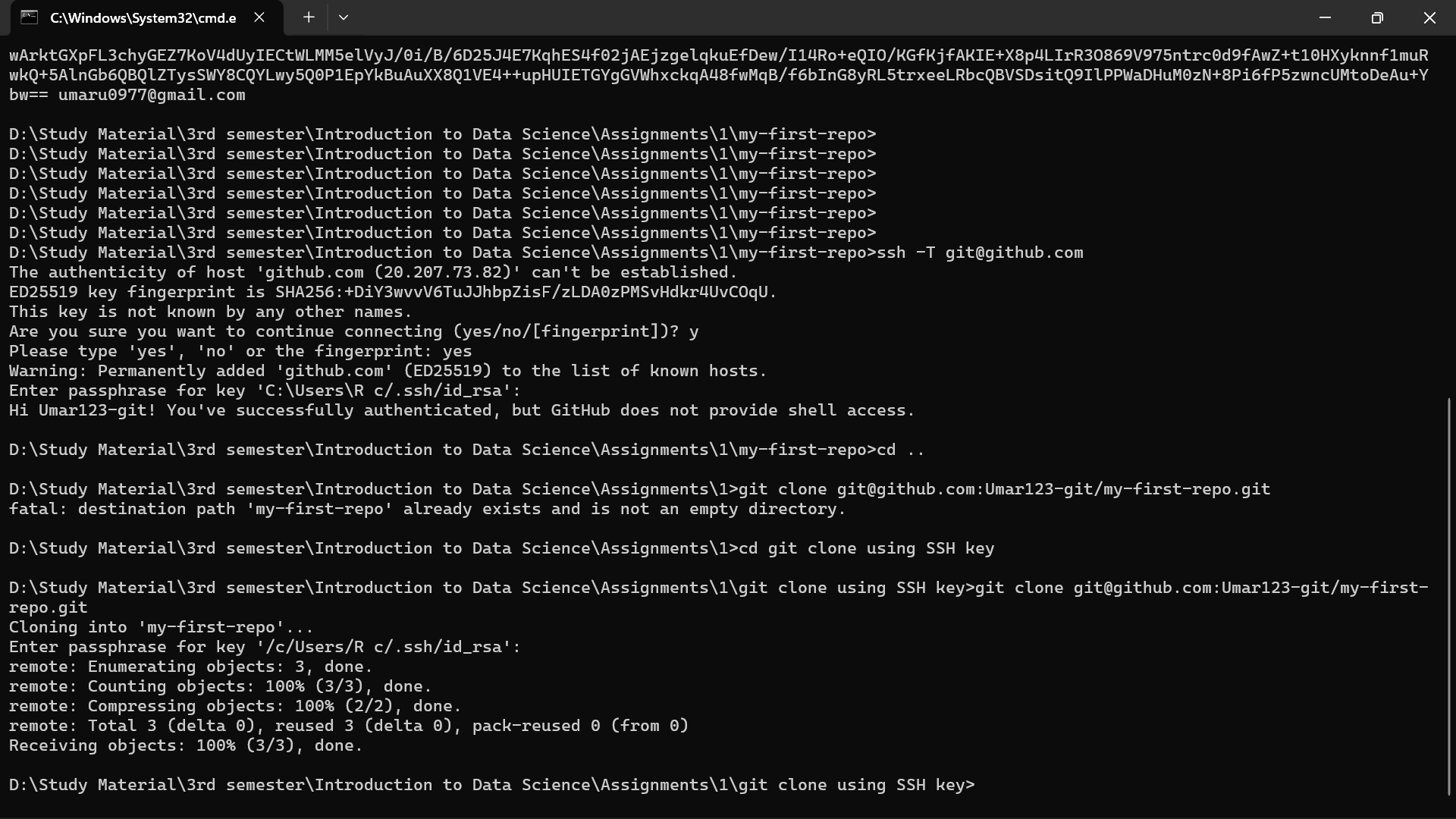


1: Create a GitHub Repository and Clone It Locally

I first created the my-first-repo on GitHub. I then used the git clone command with the HTTPS URL to download the entire repository history locally. After navigating into the directory, I created the README.md file, staged it using git add README.md, and recorded the changes with git commit -m "Add README file". Finally, I uploaded the local commit to the remote repository on GitHub using git push origin main.

**TASK 2**

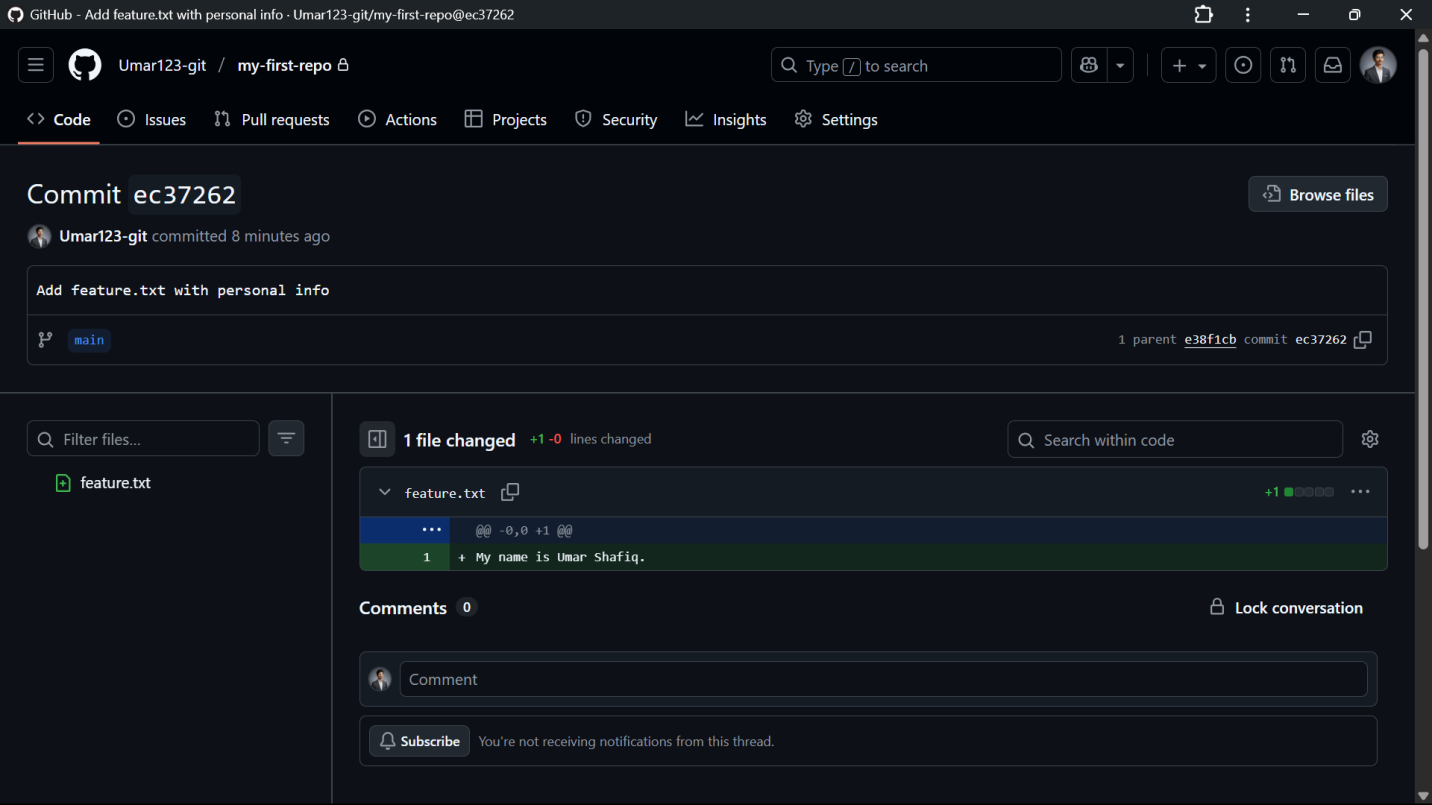
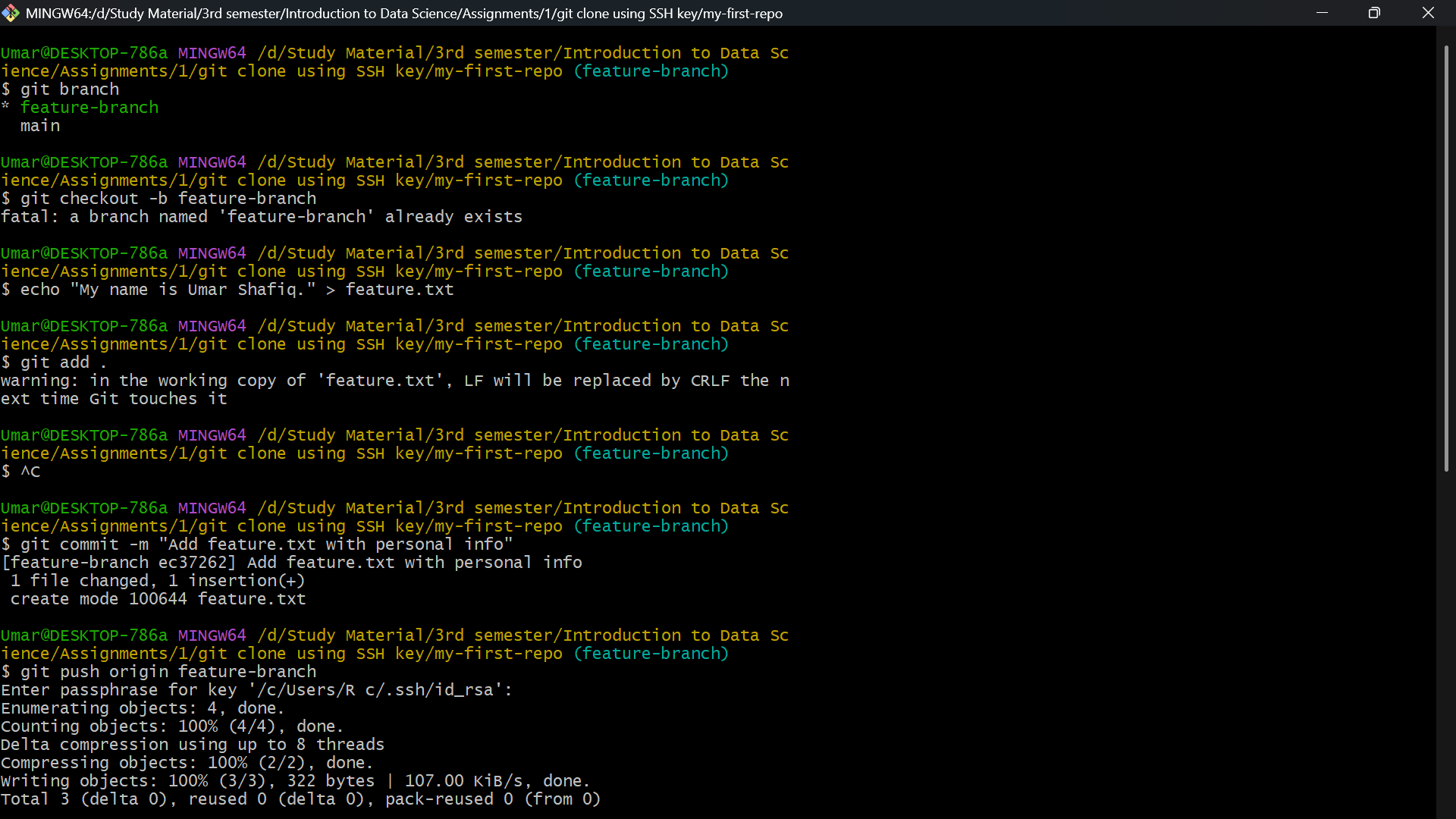


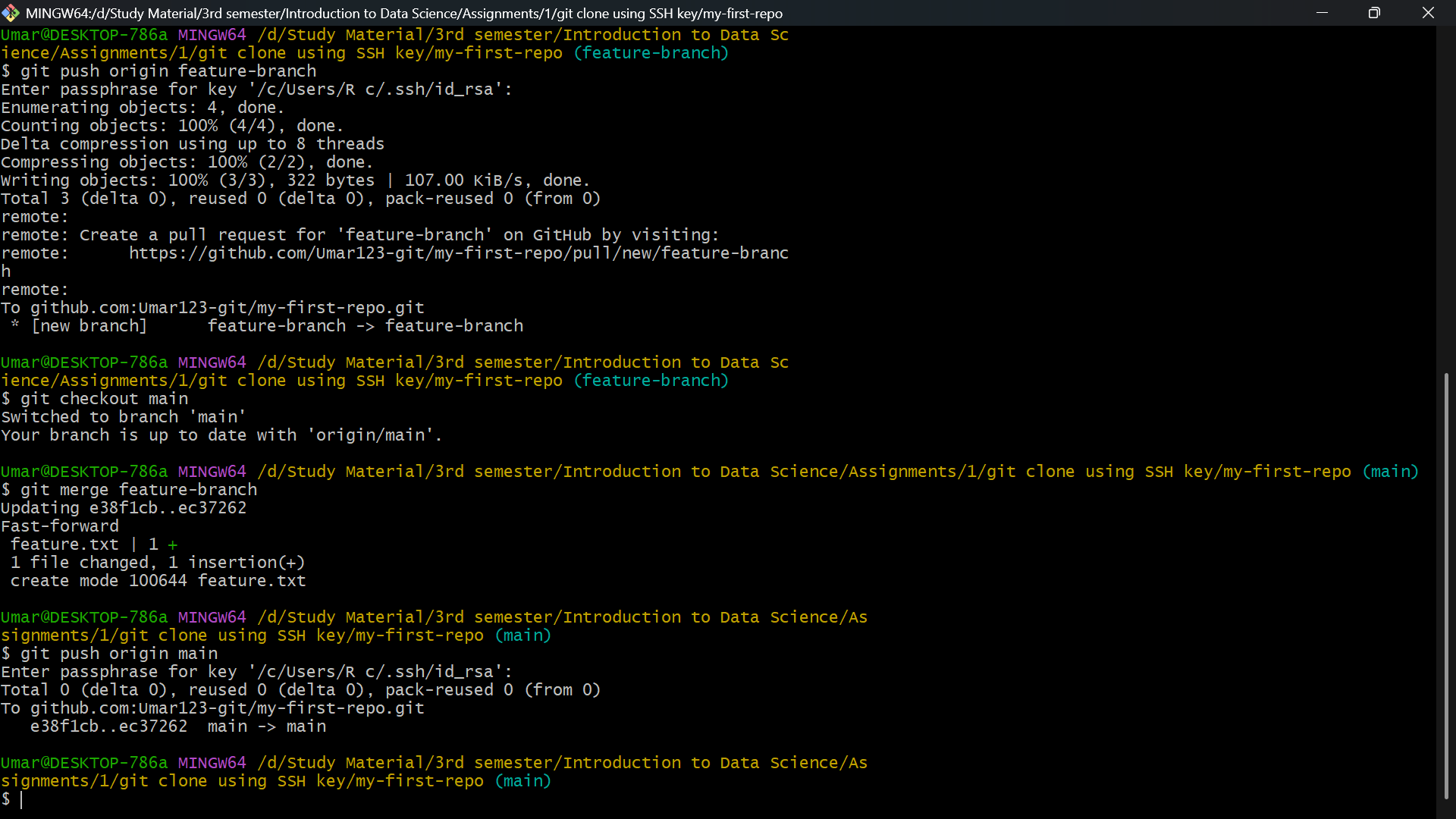


2: Add an SSH Key to GitHub

I generated a public/private SSH key pair on my local machine using the ssh-keygen command. I copied the contents of the public key (id\_rsa.pub) and added it to my GitHub account settings. I verified the connection with ssh -T git@github.com. To confirm the setup was successful, I performed a final git clone using the **SSH URL** instead of the HTTPS URL, successfully authenticating with the key.

**TASK 3**

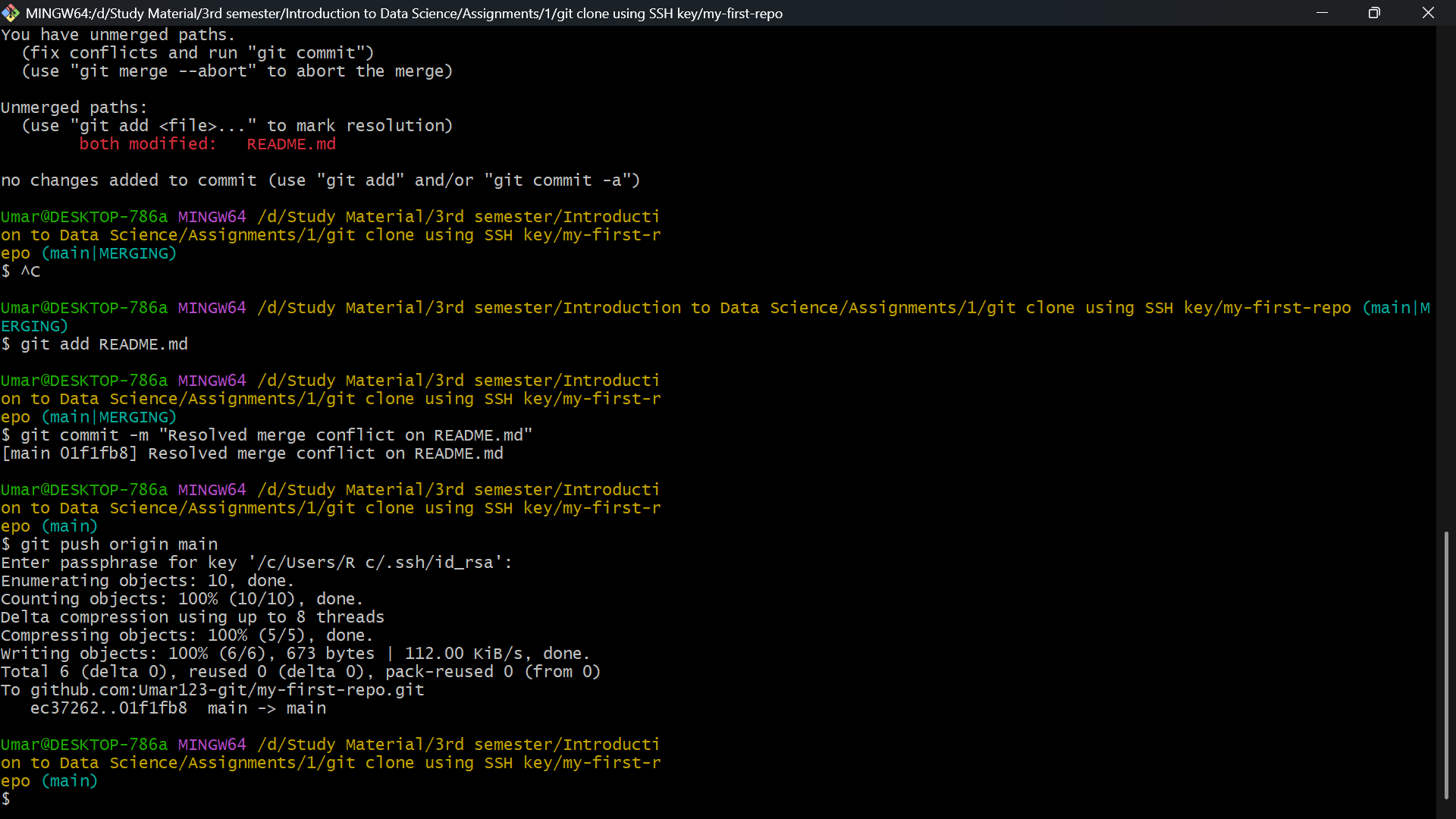
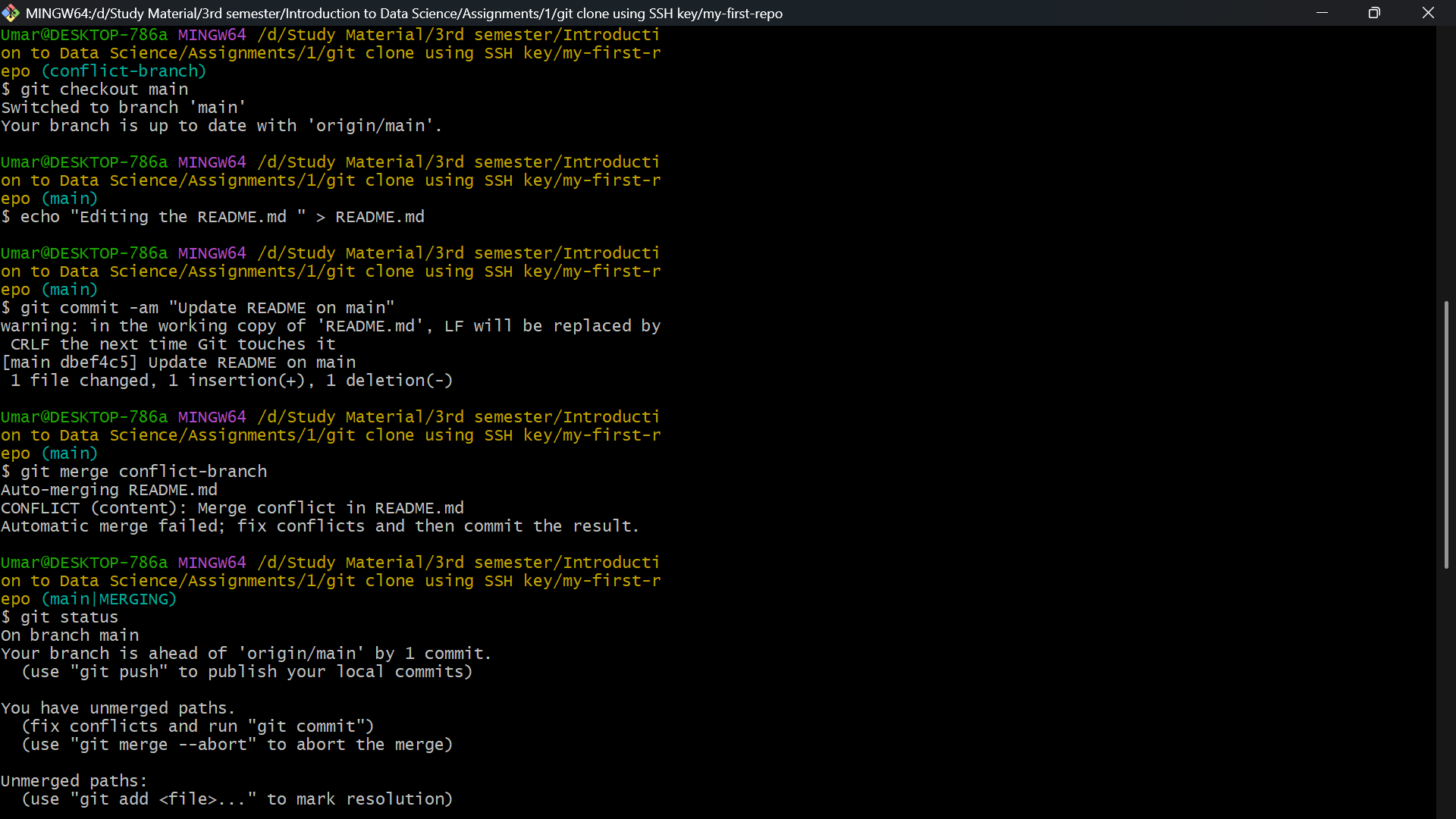
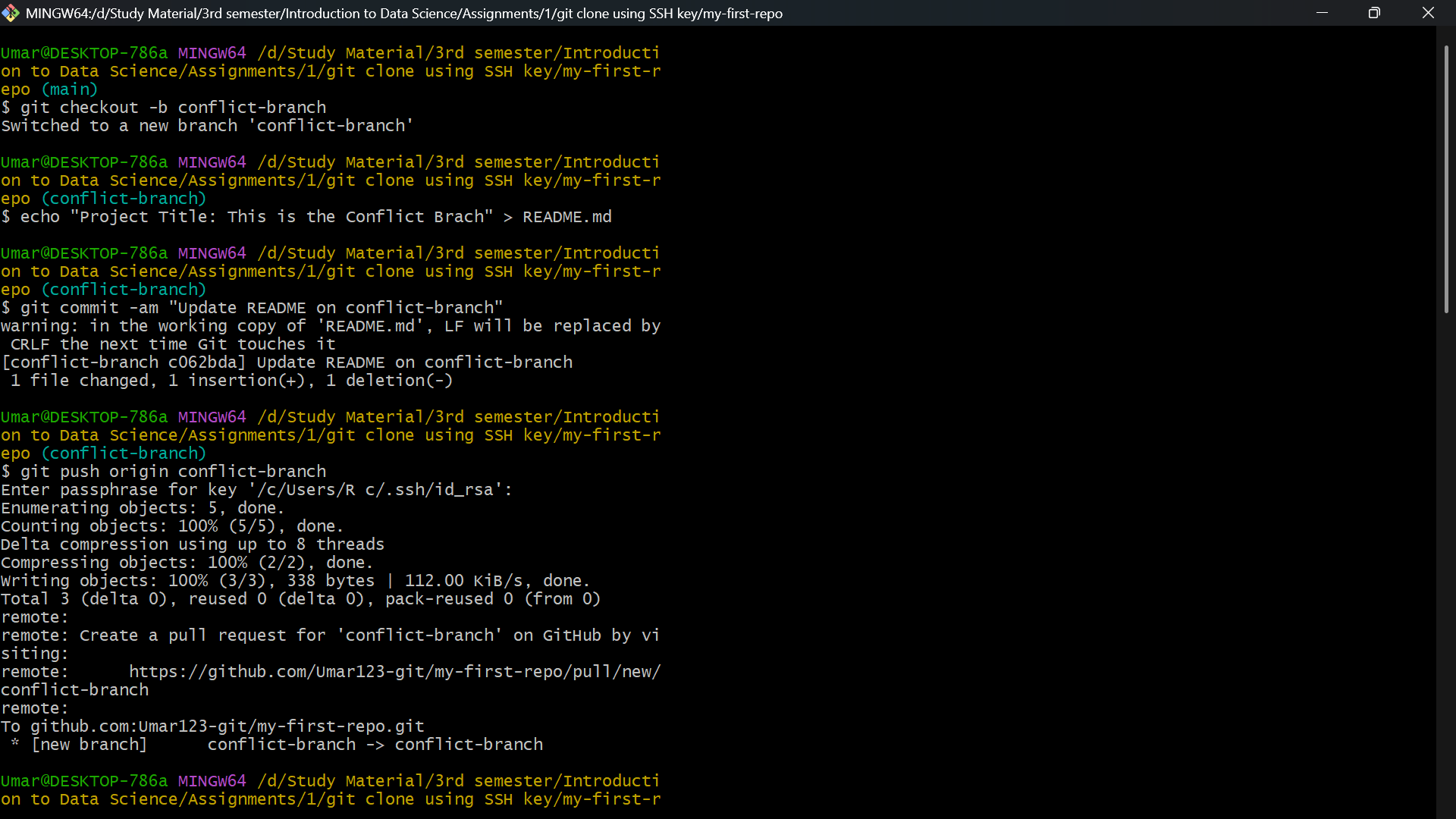
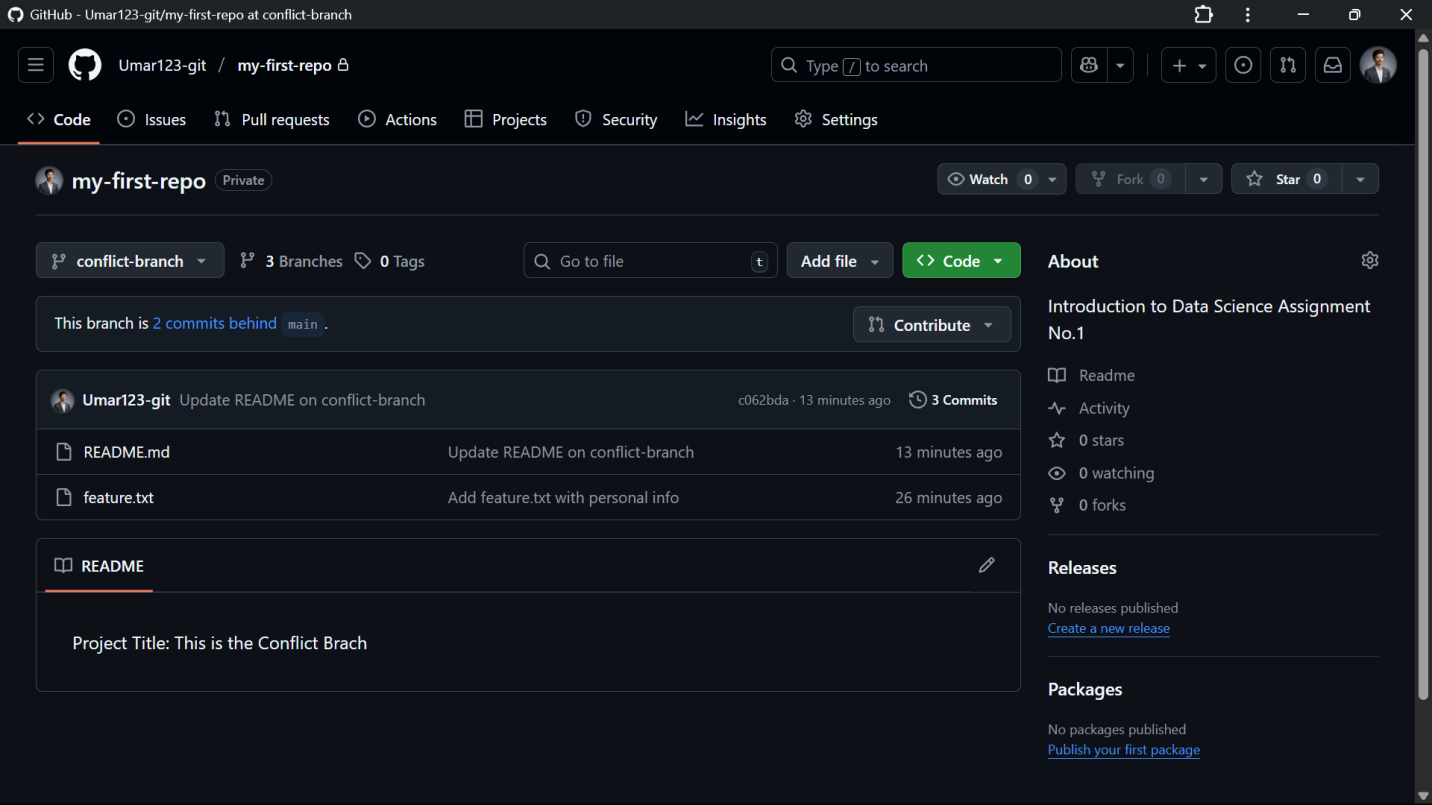
****

****

3: Branching and Merging

I created and switched to the feature-branch using git checkout -b feature-branch. I added a new file (feature.txt), committed the changes, and pushed the new branch to GitHub. I then switched back to the stable main branch using git checkout main. I integrated the new feature into main using git merge feature-branch, which completed cleanly via a fast-forward merge. I finalized the process by using git push origin main to update the central repository.

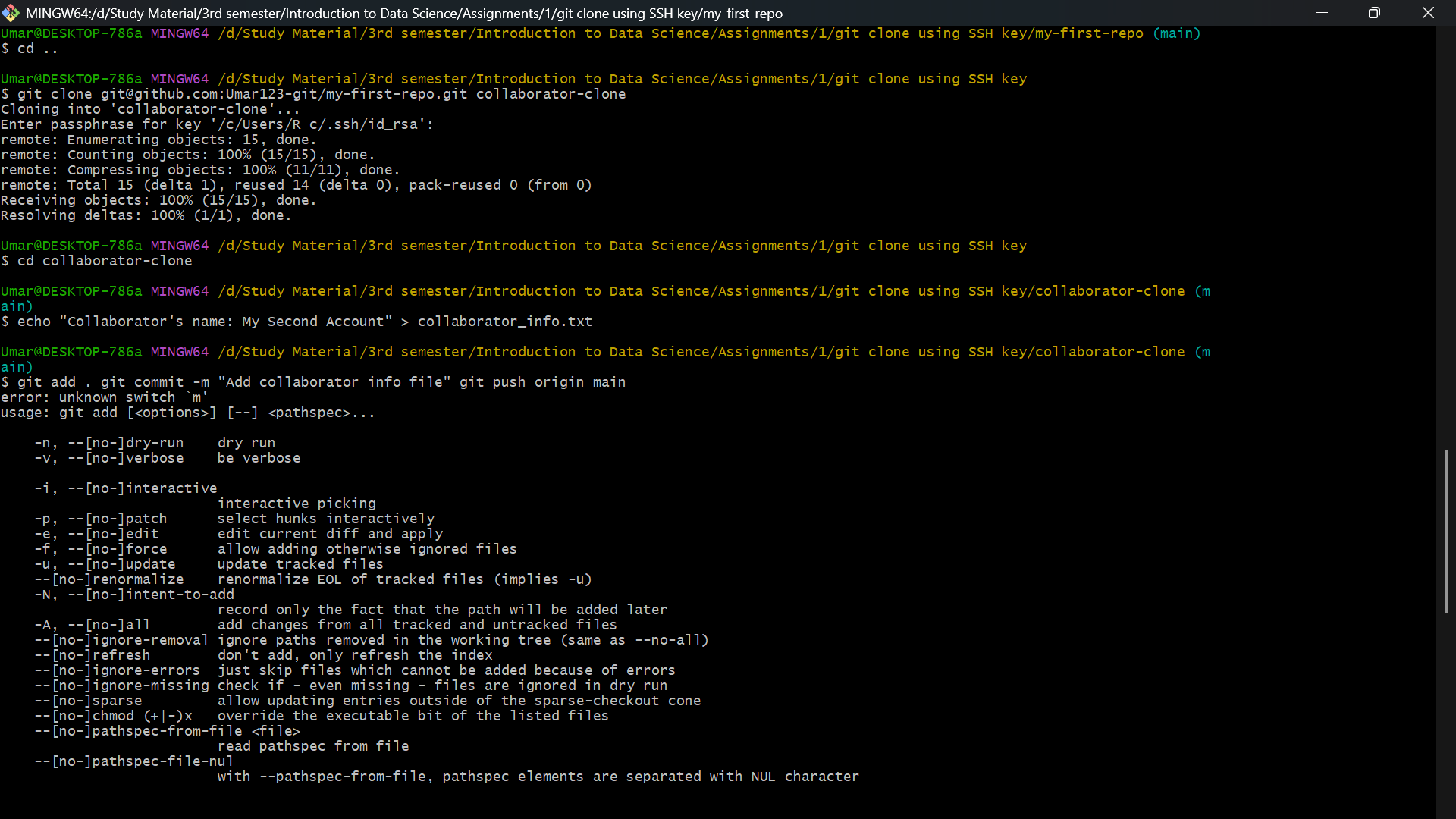
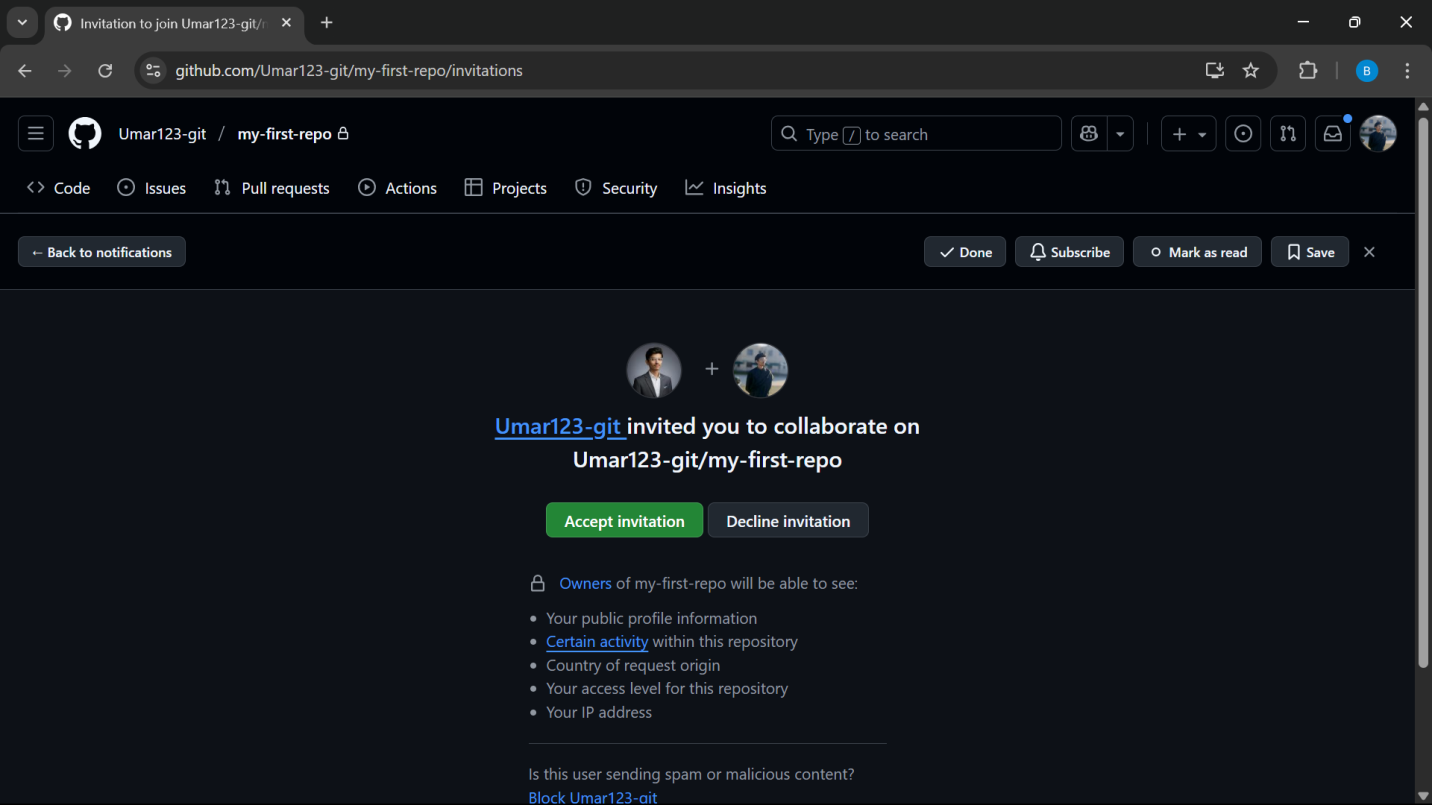
**TASK 4**

****

4: Resolving Merge Conflicts

I created the conflict-branch and modified a line in README.md. I then switched back to main and modified the *same line* differently, creating two parallel, incompatible commits. When I ran git merge conflict-branch, Git stopped with a conflict. I manually opened README.md, resolved the conflict markers to choose the final desired content, and saved the file. I then staged the resolved file with git add README.md and completed the merge using a merge commit via git commit. The final step was to push the resolved merge to GitHub with git push origin main.

**TASK 5**

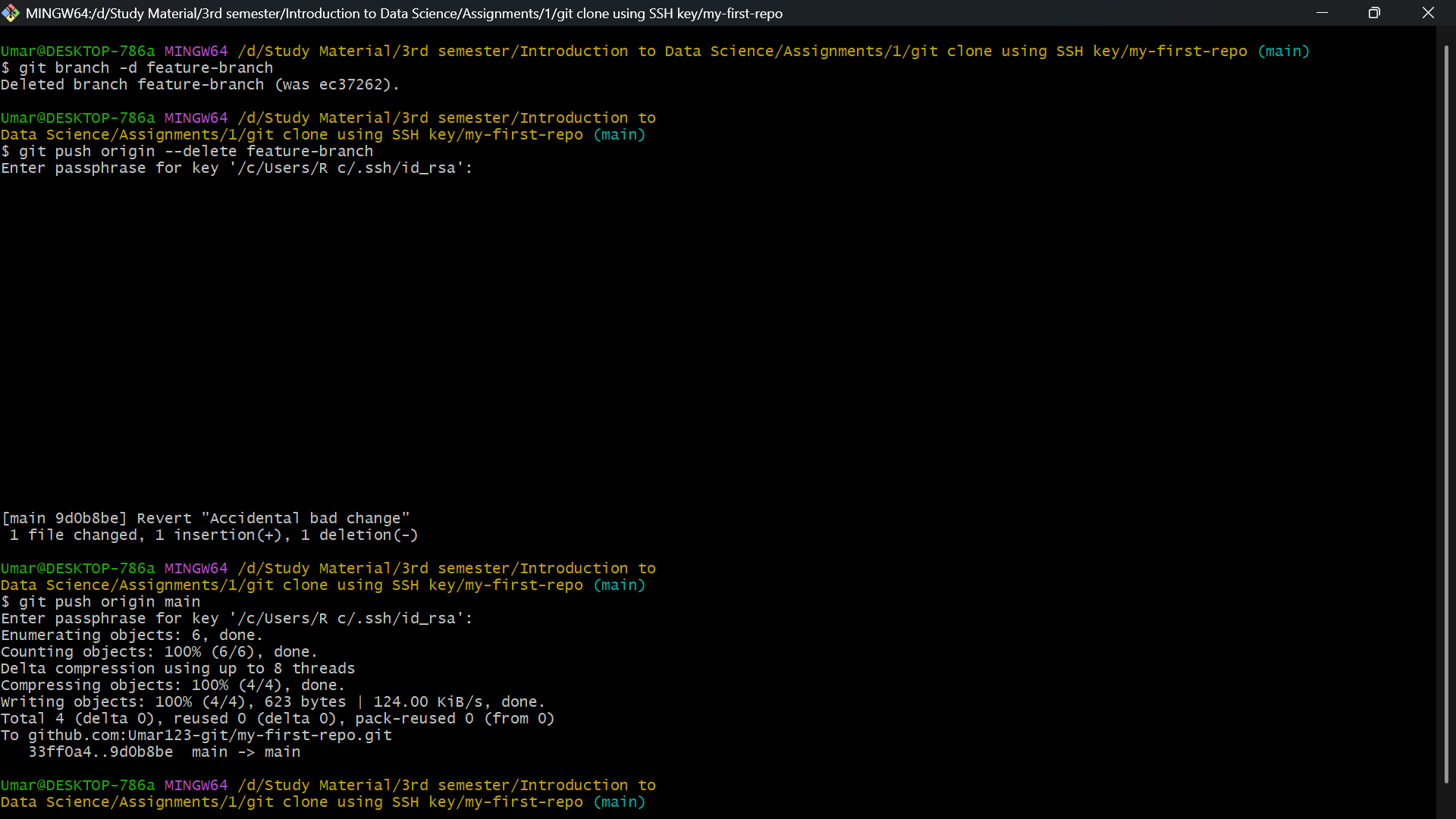
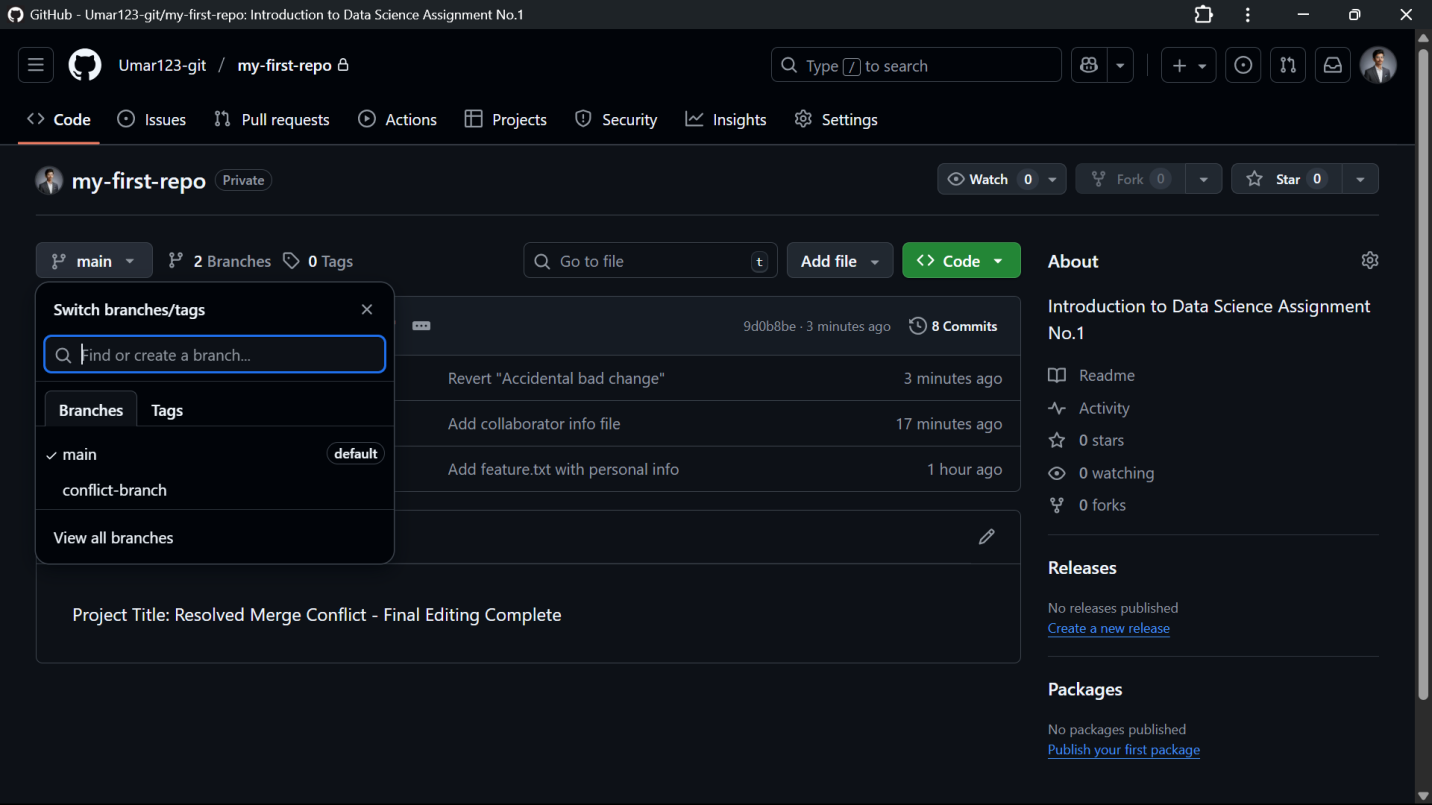
****

****

5: Collaborating on GitHub

I added my secondary GitHub account as a collaborator on the repository via the GitHub website. I then simulated the collaborator's actions by cloning the repository into a separate local folder, creating a new file (collaborator\_info.txt), committing it, and pushing it to the remote main branch. I then switched back to my main working directory and used the command git pull origin main. This command successfully fetched the collaborator's commit and merged it into my local branch, verifying the collaboration setup.

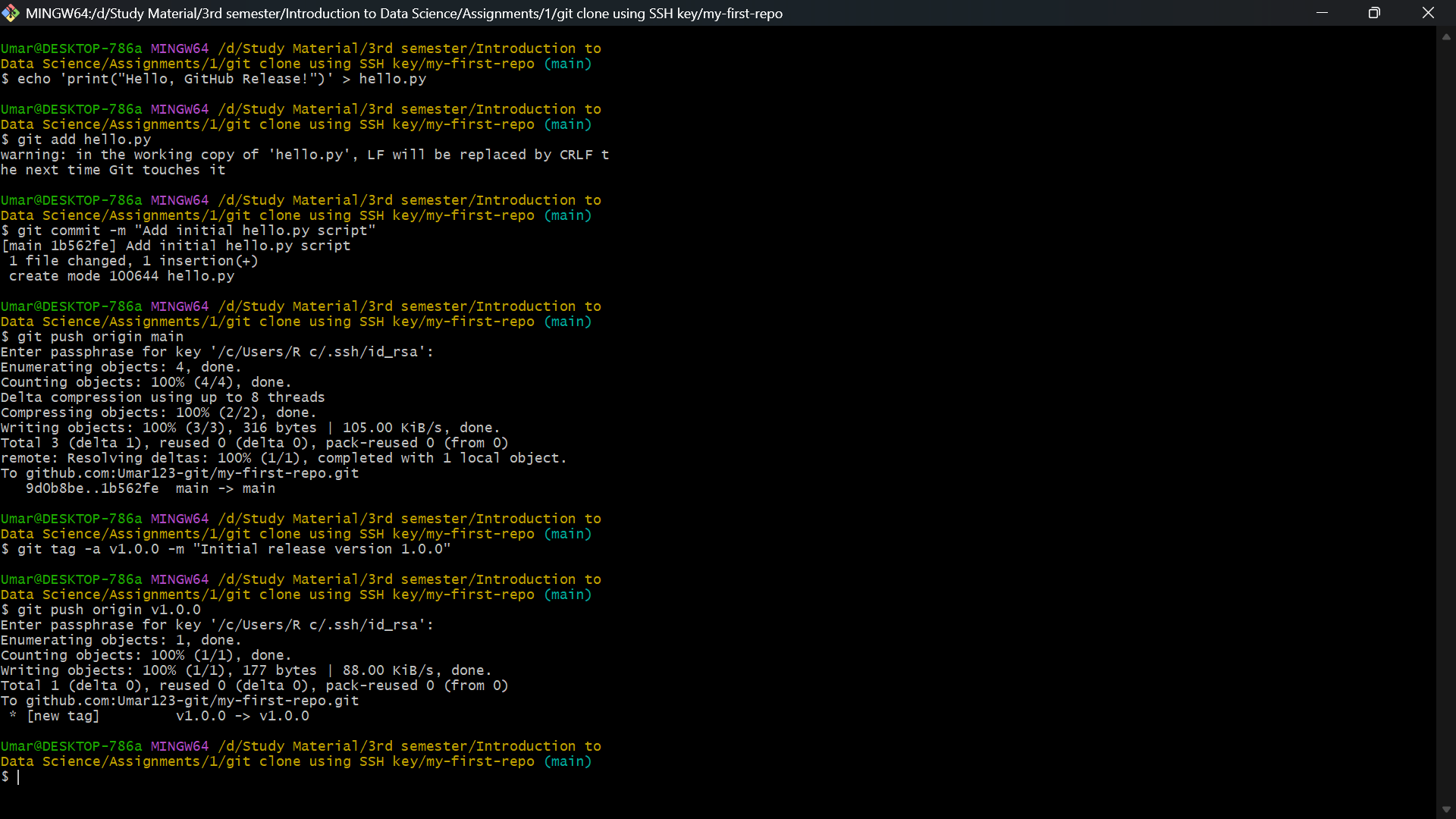
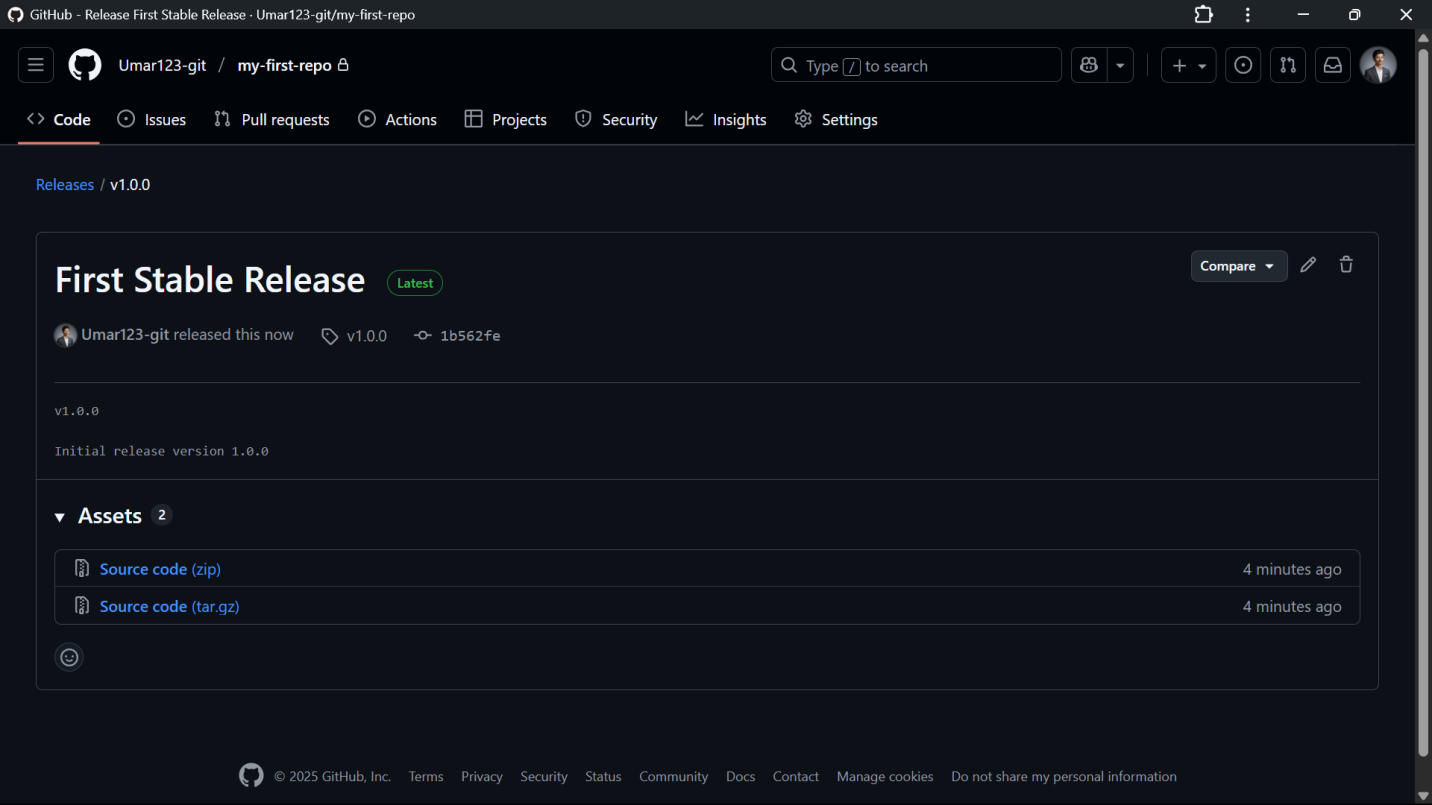
**TASK 6**

****

6: Deleting a Branch and Reverting Changes

I first cleaned up the repository by deleting the successfully merged feature-branch both locally (git branch -d feature-branch) and remotely (git push origin --delete feature-branch). To practice undoing mistakes, I intentionally made a "bad" change to README.md and committed it. I then used the git revert HEAD command, which is the recommended method for undoing commits on shared branches. This command created a new commit that precisely reversed the effect of the bad commit. I finalized the cleanup by pushing the entire history to GitHub with git push origin main.

**TASK 7**

****

7: Creating a GitHub Release with a Python Program

I wrote a simple Python script (hello.py), committed it, and pushed the new file to the main branch. To designate this point as a stable version, I used the command git tag -a v1.0.0 -m "Initial release version 1.0.0" to create an annotated tag. Because tags are not pushed by default, I explicitly pushed the tag with git push origin v1.0.0. Finally, I went to the GitHub website, navigated to the Releases section, and created an official Release associated with the v1.0.0 tag.