

1. Create a new GitHub repository.

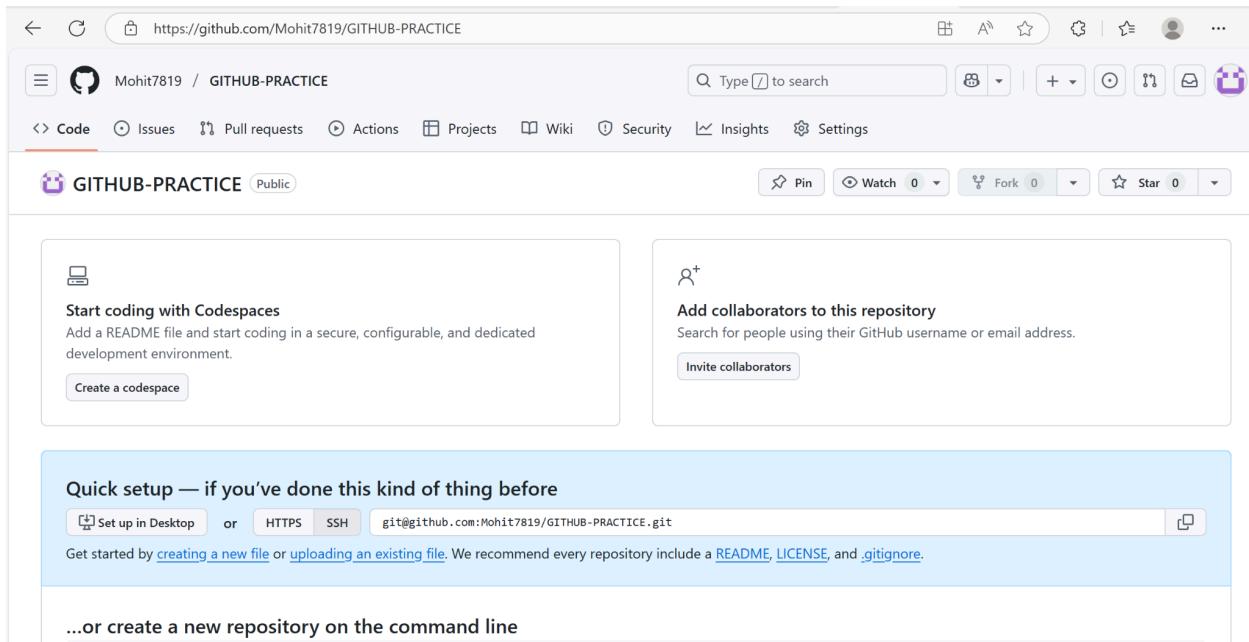
- Clone the repository to your local machine using SSH (generate an SSH key if needed, add the public key to your GitHub account).
- Create a new branch named after your username (e.g., `Tutedude`).
- Add your Flask project files to this branch.
- Commit the changes and merge the branch into the `main` branch.

◆ Step 1: Create a New GitHub Repository

- I created a new repository named GITHUB-PRACTICE.

URL (SSH):

`git@github.com:Mohit7819/GITHUB-PRACTICE.git`

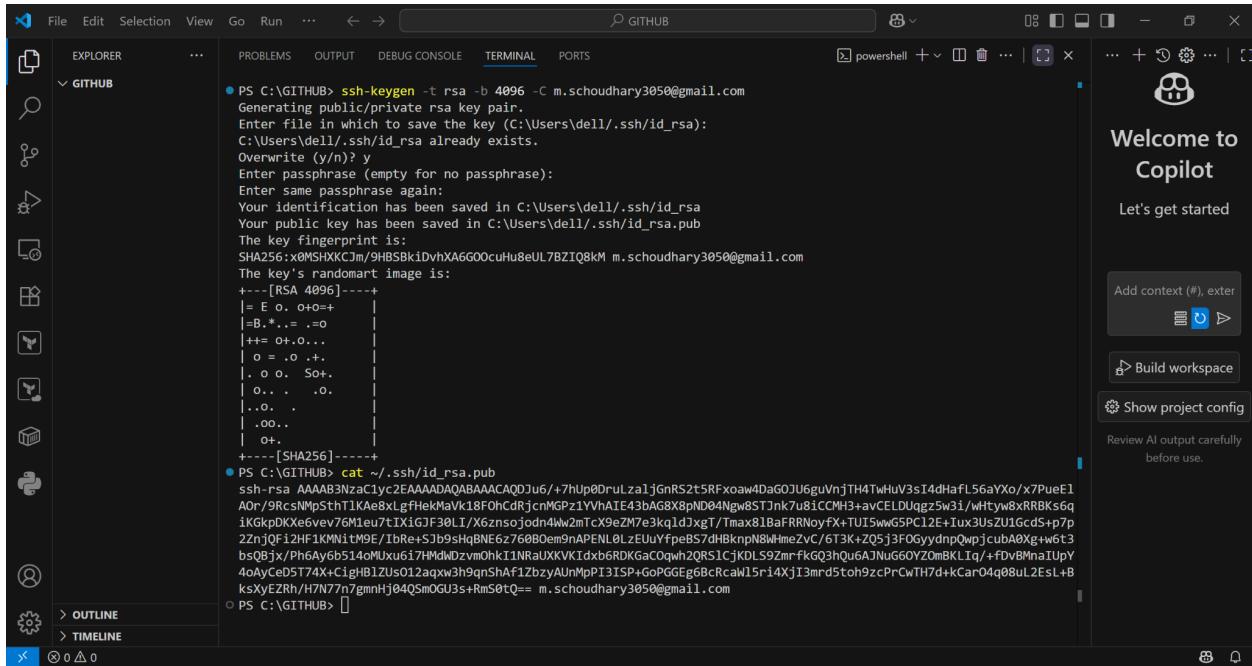


◆ Step 2: Generate SSH Key and Add to GitHub

`ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`

- Copied the public key:

`cat ssh.pub`

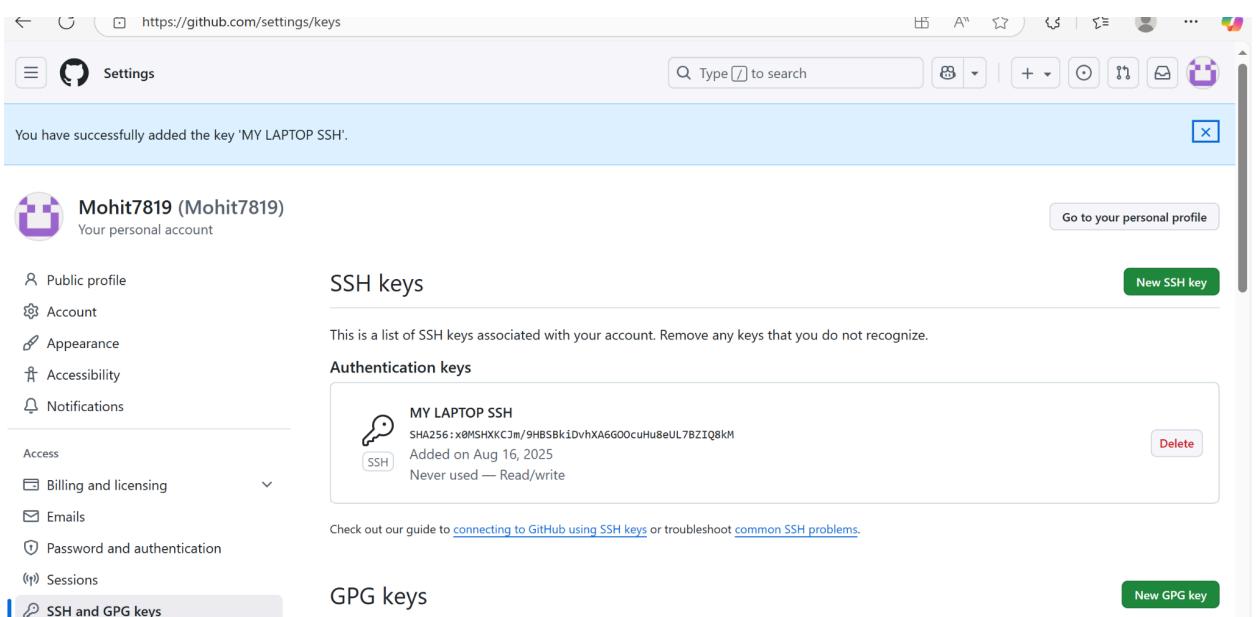


```

PS C:\GITHUB> ssh-keygen -t rsa -b 4096 -C m.schoudhary3050@gmail.com
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/dell/.ssh/id_rsa):
C:/Users/dell/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:/Users/dell/.ssh/id_rsa
Your public key has been saved in C:/Users/dell/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:x0MSHXKCJm/9HBSBk1dvhXA6GO0cuHu8eUL7BZI08kM m.schoudhary3050@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
| = E o. o+=+ |
| |=B.*..= .o= |
| ++= o+.o... |
| o = .o .+ |
| . o o. So+ |
| . o .. o. |
| . o.. . |
| . ooo.. |
| . o+.. |
+----[SHA256]----+
PS C:\GITHUB> cat ./ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQACQDJu6/+7hUp0DrulZalJGnRS2tRFxoaW4DaG0JU6guVnjTH4TwHuV3sI4dHaFL56aYXo/x7PueEl
AOr/9RcsNMpStHtKAe8xLgfhKmaV18F0hCdRjcnMGPz1VWhAIE43bAG8X8pN0d4Ngw8STJnk7u81CCMh3+avCELDUqgz531/wHtyw8xRRBKs6q
ikGpkDKx6vev76Meu7tX1Gf30lI/X6znsjodn4Ww2mTCX9ezMTe3kqlJxgT/Tmax81BaFRRNoyFx+TUI5wwG5PCl2E+IuxUsZU16cds+p7p
2ZnjQFj2HF1KMN1tM9E/IbRe+Sjb9sHqBNE6z760B0em9nAPENL0LzEuYFpeB57dhBknP8WhmeZvC/GT3K+ZQ5j3FOgydnPQwpjubaA6Xg+w6t3
bsQ8jx/Ph6Ay6b514oMuX617HMdWDzvmOhK1NRAUXKV1Idxb6RDKGaCqgnh20R51CjKDL92mrFkG3hQu6AJNu66OYZ0mBKL1q/+fdVbMnaTUpY
4oAyc0e5T7Ax-C1ghB1ZUs012aqwx3h9qrmShAf1BzbyAUmpPIISP+G0PGGeg6RCaw15r14xjI3mrd5t0h9zcPrCwTH7d+kCar04q08uL2EsL+B
ksXYeZRh/H7N7n7gmnHj04QSmOGU3s+RmS0tq== m.schoudhary3050@gmail.com
PS C:\GITHUB>

```

- Added it to GitHub under **Settings > SSH and GPG keys**



You have successfully added the key 'MY LAPTOP SSH'.

Mohit7819 (Mohit7819) Your personal account

[Go to your personal profile](#)

- Public profile
- Account
- Appearance
- Accessibility
- Notifications

- Access
- Billing and licensing
- Emails
- Password and authentication
- Sessions
- SSH and GPG keys**

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys

 MY LAPTOP SSH SHA256:x0MSHXKCJm/9HBSBk1dvhXA6GO0cuHu8eUL7BZI08kM Added on Aug 16, 2025 Never used — Read/write	Delete
--	------------------------

[Check out our guide to connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

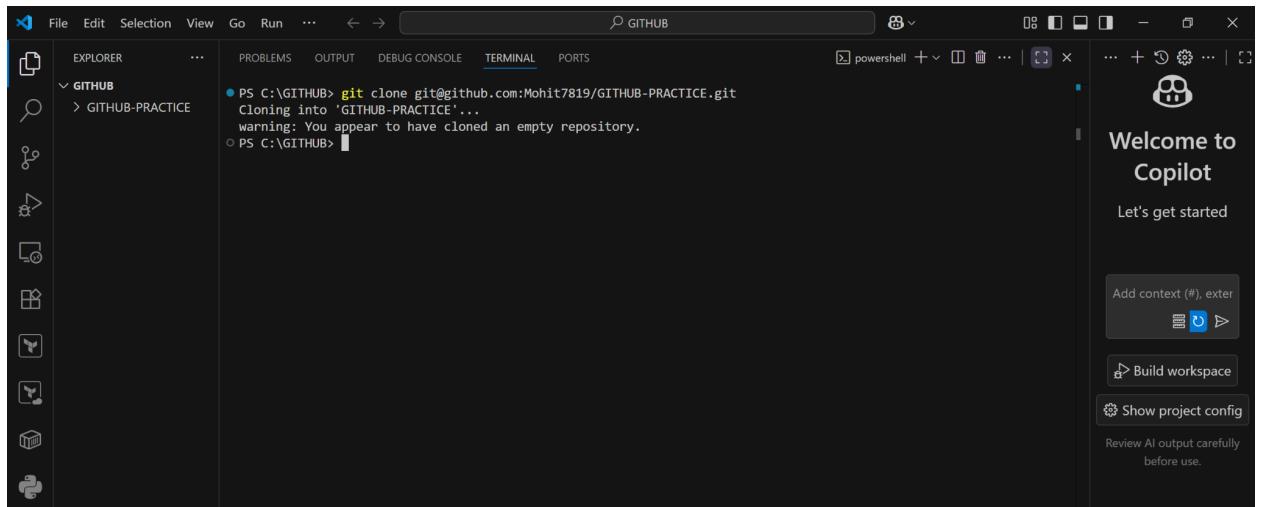
GPG keys

[New GPG key](#)

- ◆ **Step 3: Clone the Repository via SSH**

```
git clone git@github.com:Mohit7819/GITHUB-PRACTICE.git
```

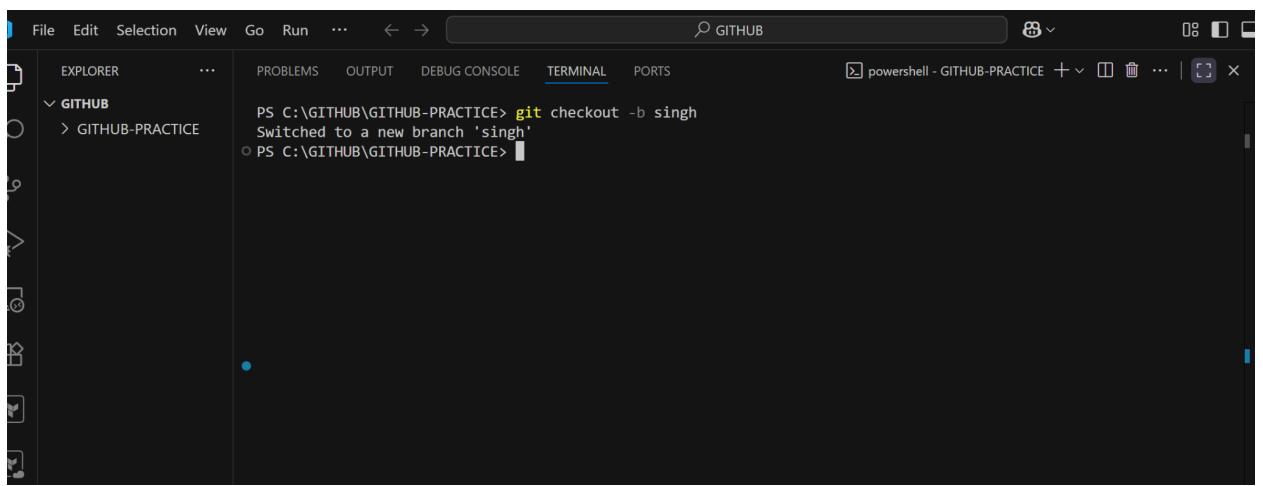
```
cd GITHUB-PRACTICE
```



A screenshot of the Visual Studio Code interface. The title bar says "GITHUB". The left sidebar shows an "EXPLORER" view with a folder named "GITHUB" expanded, containing "GITHUB-PRACTICE". The main area is the "TERMINAL" tab, which displays the command "git clone git@github.com:Mohit7819/GITHUB-PRACTICE.git" being run in a PowerShell window. The output shows the repository is being cloned into "GITHUB-PRACTICE" and includes a warning message: "warning: You appear to have cloned an empty repository.". The status bar at the bottom indicates "powershell". On the right side, there's a "Welcome to Copilot" panel with options like "Add context (#), exter", "Build workspace", "Show project config", and a note to "Review AI output carefully before use".

- ◆ **Step 4: Create a New Branch (Named After Your Username)**

```
git checkout -b your-username
```



A screenshot of the Visual Studio Code interface, similar to the previous one but with a different terminal history. The title bar says "GITHUB". The left sidebar shows an "EXPLORER" view with a folder named "GITHUB" expanded, containing "GITHUB-PRACTICE". The main area is the "TERMINAL" tab, which displays the command "git checkout -b singh" being run in a PowerShell window. The output shows the branch "singh" has been switched to. The status bar at the bottom indicates "powershell - GITHUB-PRACTICE".

- ◆ **Step 5: Add Your Flask Project Files**
- **Copy your Flask project files into the GITHUB_PRACTICE folder.**

- ◆ **Step 6: Add, Commit,Push and Merge Changes**

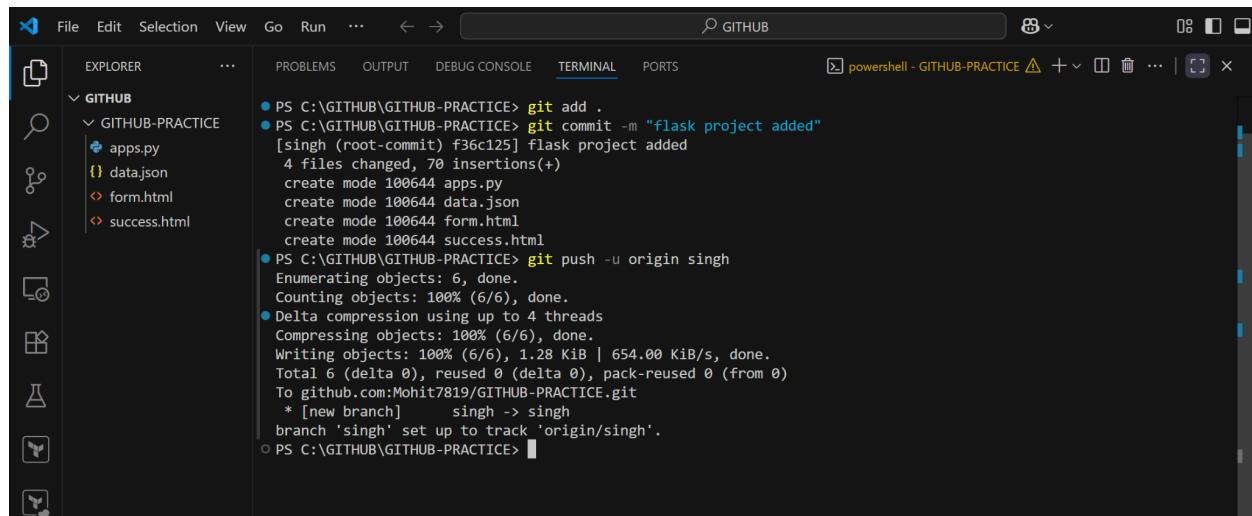
```
git add .
```

```
git commit -m "Flask project added"
```

```
git push -u origin singh
```

```
git checkout main
```

```
git merge singh
```

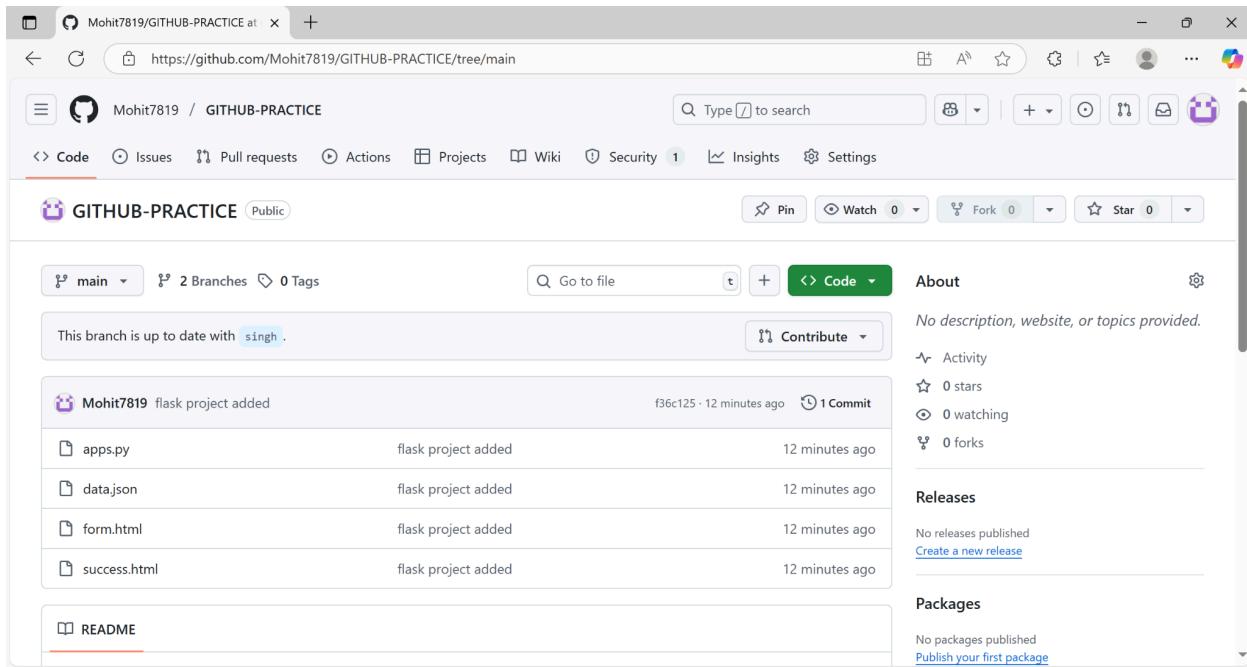


```

File Edit Selection View Go Run ...
GITHUB
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell - GITHUB-PRACTICE + ... x

PS C:\GITHUB\GITHUB-PRACTICE> git add .
PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "flask project added"
[singh (root-commit) f36c125] flask project added
 4 files changed, 70 insertions(+)
  create mode 100644 apps.py
  create mode 100644 data.json
  create mode 100644 form.html
  create mode 100644 success.html
PS C:\GITHUB\GITHUB-PRACTICE> git push -u origin singh
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.28 KiB | 654.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Mohit7819/GITHUB-PRACTICE.git
 * [new branch]      singh -> singh
branch 'singh' set up to track 'origin/singh'.
PS C:\GITHUB\GITHUB-PRACTICE>

```



2. Create a new branch named <your_name>_new (e.g., Tutedude_new).

- Update the content of the JSON file used for the /api route in this branch.
- Merge the <your_name>_new branch into the main branch.
- If there are conflicts during the merge, resolve them by accepting the changes from the <your_name>_new branch.
- Add the resolved changes to the staging area, commit them, and push the updates to the remote repository.

◆ Step 1: Create a New Branch Named <your_name>_new

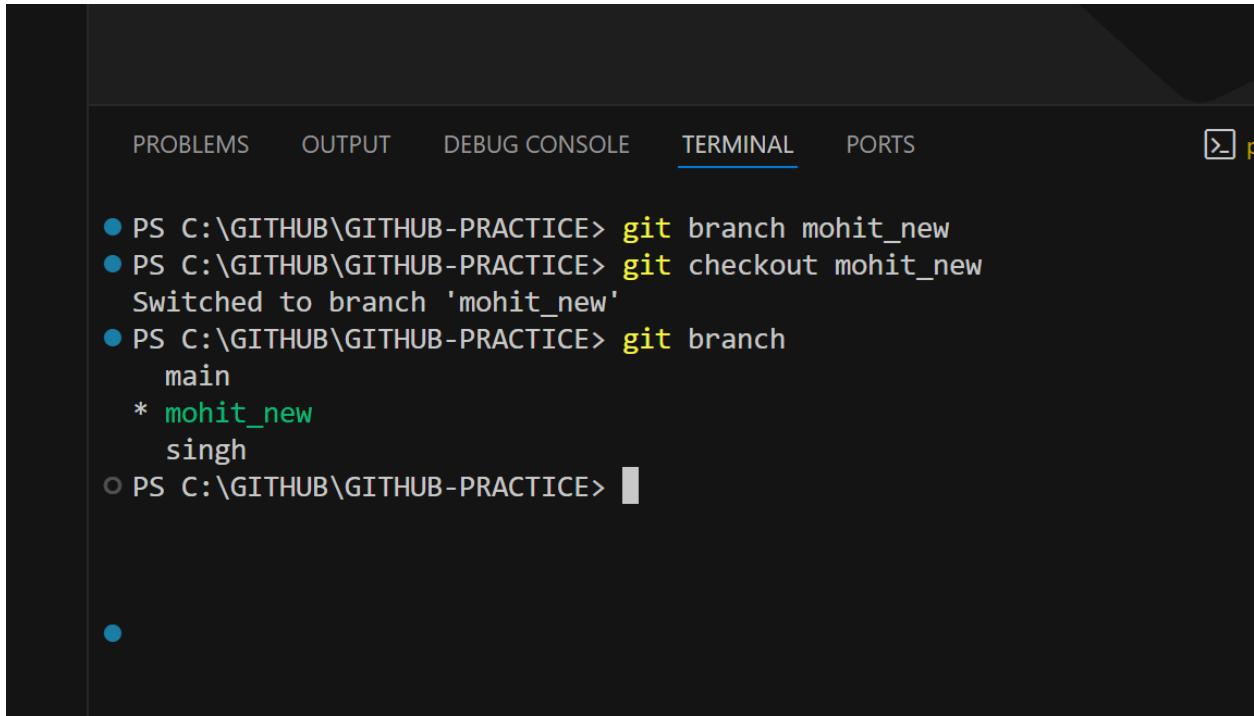
```
git checkout -b yourname_new
```

OR

```
git branch mohit_new
```

```
git checkout mohit_new
```

```
git branch
```



A screenshot of a terminal window titled "TERMINAL". The window shows a command-line interface with the following text:

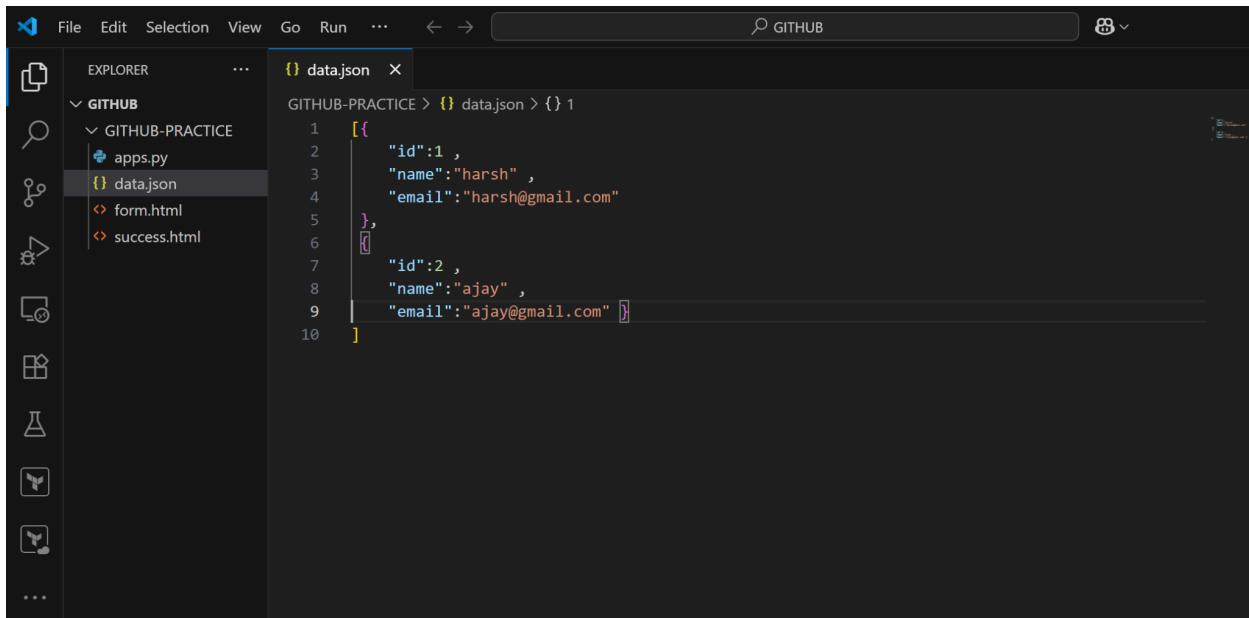
- PS C:\GITHUB\GITHUB-PRACTICE> `git branch` mohit_new
- PS C:\GITHUB\GITHUB-PRACTICE> `git checkout` mohit_new
Switched to branch 'mohit_new'
- PS C:\GITHUB\GITHUB-PRACTICE> `git branch`
main
* mohit_new
singh
- PS C:\GITHUB\GITHUB-PRACTICE> █

◆ **Step 2: Update the JSON File Used in /api Route**

Let's say your Flask app has a `data.json` file used in the `/api` route.
Open it and update the content.

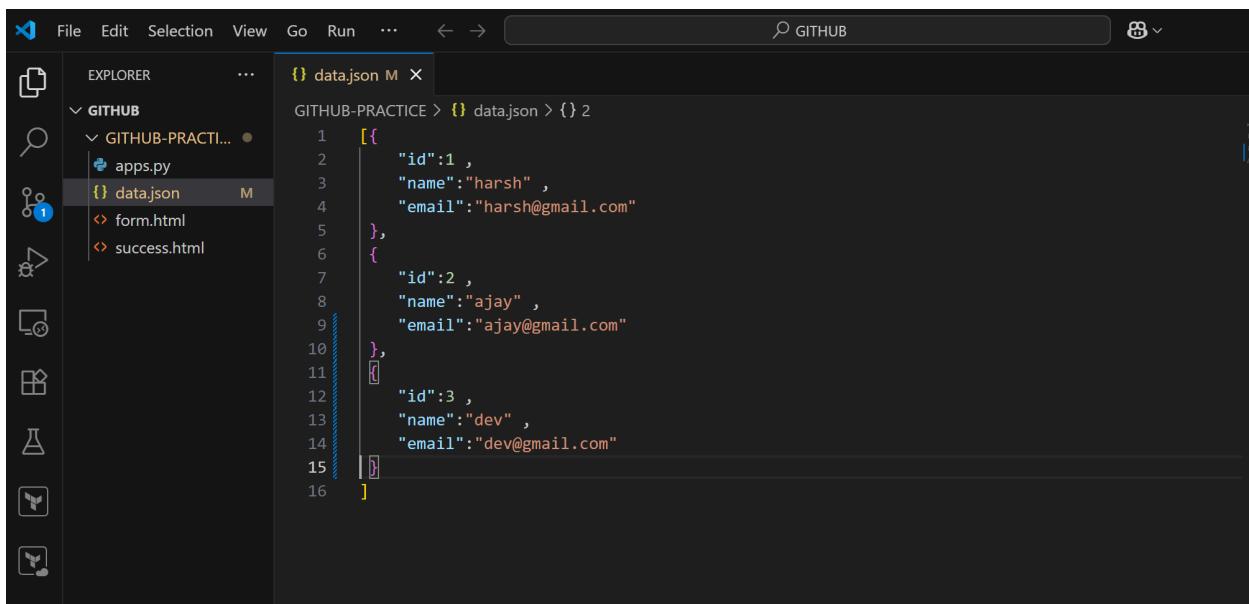
For example:

Before (`data.json`):



```
{} data.json x
GITHUB-PRACTICE > {} data.json > {} 1
1 [ {
2   "id":1 ,
3   "name":"harsh" ,
4   "email":"harsh@gmail.com"
5 },
6 [
7   "id":2 ,
8   "name":"ajay" ,
9   "email":"ajay@gmail.com" ]
10 ]
```

After:



```
{} data.json M x
GITHUB-PRACTICE > {} data.json > {} 2
1 [ {
2   "id":1 ,
3   "name":"harsh" ,
4   "email":"harsh@gmail.com"
5 },
6 [
7   "id":2 ,
8   "name":"ajay" ,
9   "email":"ajay@gmail.com"
10 },
11 [
12   "id":3 ,
13   "name":"dev" ,
14   "email":"dev@gmail.com"
15 ]
16 ]
```

- ◆ **Step 3: Add, Commit, and Push the Changes**

```
git add data.json
```

```
git commit -m "Updated JSON content in yourname_new branch"
```

```
git push origin mohit_new
```

```
PS C:\GITHUB\GITHUB-PRACTICE> git branch
  main
* mohit_new
  singh

PS C:\GITHUB\GITHUB-PRACTICE> git add data.json
>> git commit -m "Updated JSON content in yourname_new branch"
>> git push origin mohit_new
[mohit_new 4e0dd8f] Updated JSON content in yourname_new branch
  1 file changed, 7 insertions(+), 1 deletion(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 371 bytes | 371.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'mohit_new' on GitHub by visiting:
remote:   https://github.com/Mohit7819/GITHUB-PRACTICE/pull/new/mohit_new
remote:
To github.com:Mohit7819/GITHUB-PRACTICE.git
 * [new branch]      mohit_new -> mohit_new
PS C:\GITHUB\GITHUB-PRACTICE>
```

◆ Step 4: Merge `mohit_new` Branch into `main`

First, switch back to the `main` branch:

```
git checkout main
```

Then, merge:

```
git merge mohit_new
```

```
PS C:\GITHUB\GITHUB-PRACTICE> git checkout main
Switched to branch 'main'
PS C:\GITHUB\GITHUB-PRACTICE> git merge mohit_new
Updating f36c125..4e0dd8f
Fast-forward
  data.json | 8 ++++++-
  1 file changed, 7 insertions(+), 1 deletion(-)
PS C:\GITHUB\GITHUB-PRACTICE>
```

◆ Step 5: Push Final Changes to Remote

```
git push origin main
```

```
● PS C:\GITHUB\GITHUB-PRACTICE> git push origin main
  Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
  To github.com:Mohit7819/GITHUB-PRACTICE.git
    f36c125..4e0dd8f  main -> main
○ PS C:\GITHUB\GITHUB-PRACTICE>
```

3. Branch Creation:

- Create two branches: `master_1` and `master_2` from the `main` branch.
- **Feature Development in `master_1`:**
- In the `master_1` branch, create a **To-Do Page** in the frontend.
 - The page should contain a form with the following fields:
 - **Item Name**
 - **Item Description**
- **Backend API in `master_2`:**
- In the `master_2` branch, create a backend route named `/submittodoitem`.
- This route will:
 - Accept `itemName` and `itemDescription` via a POST request.
 - Store these details in a MongoDB database.
- **Merging Changes:**
- Merge the changes from both `master_1` and `master_2` into the `main` branch.

◆ Step 1: Create Branches `master_1` and `master_2` from `main`

```
# Make sure you're on main and up to date
```

```
git checkout main
```

```
git pull origin main
```

```
# Create and switch to master_1
```

```
git checkout -b master_1
```

```
# Switch back to main
git checkout main

# Create and switch to master_2
git checkout -b master_2
```

The screenshot shows a terminal window with the following history:

- PS C:\GITHUB\GITHUB-PRACTICE> **git checkout main**
Already on 'main'
- Your branch is up to date with 'origin/main'.
- PS C:\GITHUB\GITHUB-PRACTICE> **git pull origin main**
From github.com:Mohit7819/GITHUB-PRACTICE
* branch main → FETCH_HEAD
Already up to date.
- PS C:\GITHUB\GITHUB-PRACTICE> **git checkout -b master_1**
Switched to a new branch 'master_1'
- PS C:\GITHUB\GITHUB-PRACTICE> **git checkout main**
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
- PS C:\GITHUB\GITHUB-PRACTICE> **git checkout -b master_2**
Switched to a new branch 'master_2'
- PS C:\GITHUB\GITHUB-PRACTICE> **git checkout main**
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
- PS C:\GITHUB\GITHUB-PRACTICE> **git branch**

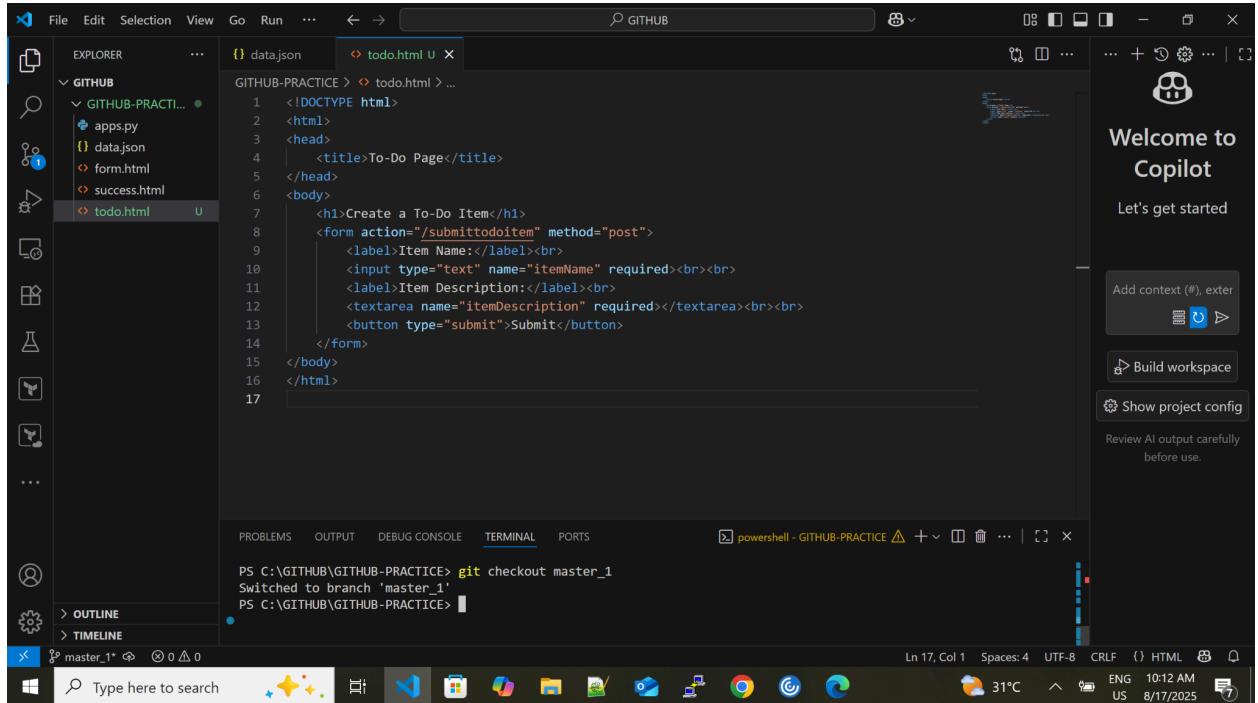
The output shows the current branches: * **main**, **master_1**, **master_2**, **mohit_new**, and **singh**.

◆ Part A: Feature Development in **master_1** (Frontend To-Do Page)

1. Switch to **master_1** branch if not already:

```
git checkout master_1
```

2. Create a file `todo.html`:



```
<!DOCTYPE html>
<html>
<head>
<title>To-Do Page</title>
</head>
<body>
<h1>Create a To-Do Item</h1>
<form action="/submittodoitem" method="post">
<label>Item Name:</label><br>
<input type="text" name="itemName" required><br><br>
<label>Item Description:</label><br>
<textarea name="itemDescription" required></textarea><br><br>
<button type="submit">Submit</button>
</form>
</body>
</html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\GITHUB\GITHUB-PRACTICE> git checkout master_1
Switched to branch 'master_1'
PS C:\GITHUB\GITHUB-PRACTICE>
```

Type here to search

3. In your Flask app route (e.g., `apps.py`):

```
@app.route('/todo', methods=['GET'])

def todo():

    return render_template('todo.html')
```

```
1  from flask import Flask, render_template, request, redirect, url_for
2  from pymongo import MongoClient
3  import os
4  app = Flask(__name__)
5  MONGODB_URI = "mongodb+srv://vikramsingh91870:GctbJt1H2Kp0IVv@cluster0.2psj6fk.mongodb.net/"
6  client = MongoClient(MONGODB_URI)
7  db = client.formData
8  collection = db.submissions
9  @app.route('/', methods=['GET'])
10 def index():
11     return render_template('form.html', error=None)
12 @app.route('/todo', methods=['GET'])
13 def todo():
14     return render_template('todo.html')
15 @app.route('/submit', methods=['POST'])
16 def submit():
17     name = request.form.get('name')
18     email = request.form.get('email')
19     if not name or not email:
20         return render_template('form.html', error="Please fill all fields")
21     try:
22         collection.insert_one({'name': name, 'email': email})
23         return redirect(url_for('success'))
24     except Exception as e:
25         return render_template('form.html', error=str(e))
26 @app.route('/success')
27 def success():
28     return render_template('success.html')
29 if __name__ == '__main__':
30     app.run(debug=True)
```

4. Add, commit, and push:

```
git add .
```

```
git commit -m "Added frontend To-Do page in master_1"
```

```
git push origin master_1
```

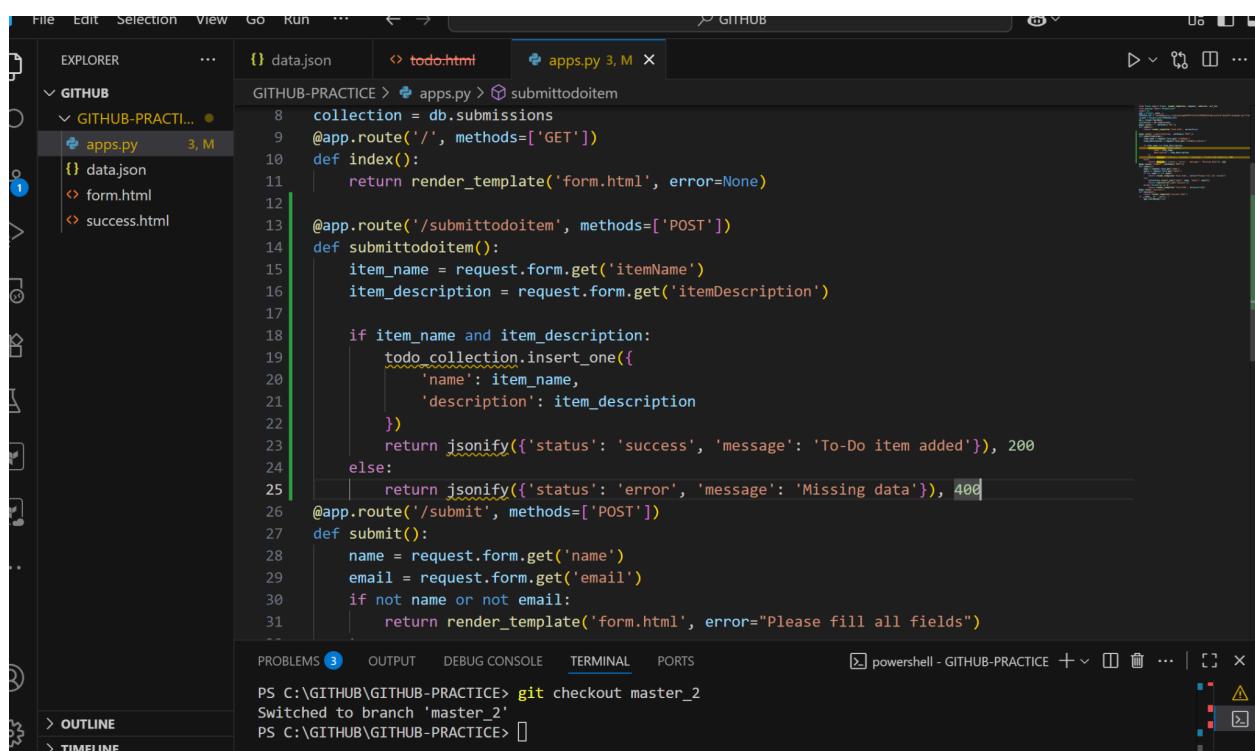
```
PS C:\GITHUB\GITHUB-PRACTICE> git add .
>> git commit -m "Added frontend To-Do page in master_1"
>> git push origin master_1
[master_1 e0266e4] Added frontend To-Do page in master_1
 2 files changed, 19 insertions(+)
  create mode 100644 todo.html
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 690 bytes | 345.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'master_1' on GitHub by visiting:
remote:   https://github.com/Mohit7819/GITHUB-PRACTICE/pull/new/master_1
remote:
To github.com:Mohit7819/GITHUB-PRACTICE.git
 * [new branch]      master_1 -> master_1
PS C:\GITHUB\GITHUB-PRACTICE>
```

◆ Part B: Backend API in master_2 (MongoDB Integration)

1. Switch to master_2 branch:

```
git checkout master_2
```

2. Add the /submittodoitem route to apps.py



```
File Edit Selection View Go Run ... GITHUB
EXPLORER ... {} data.json todo.html apps.py 3, M
GITHUB-PRACTICE > apps.py > submittodoitem
8     collection = db.submissions
9     @app.route('/', methods=['GET'])
10    def index():
11        return render_template('form.html', error=None)
12
13    @app.route('/submittodoitem', methods=['POST'])
14    def submittodoitem():
15        item_name = request.form.get('itemName')
16        item_description = request.form.get('itemDescription')
17
18        if item_name and item_description:
19            todo_collection.insert_one({
20                'name': item_name,
21                'description': item_description
22            })
23            return jsonify({'status': 'success', 'message': 'To-Do item added'})
24        else:
25            return jsonify({'status': 'error', 'message': 'Missing data'}), 400
26    @app.route('/submit', methods=['POST'])
27    def submit():
28        name = request.form.get('name')
29        email = request.form.get('email')
30        if not name or not email:
31            return render_template('form.html', error="Please fill all fields")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - GITHUB-PRACTICE + ...

```
PS C:\GITHUB\GITHUB-PRACTICE> git checkout master_2
Switched to branch 'master_2'
PS C:\GITHUB\GITHUB-PRACTICE>
```

Add, commit, and push:

```
git add .
```

```
git commit -m "Added /submittodoitem backend API in master_2"
```

```
git push origin master_2
```

The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** GITHUB
- Explorer:** Shows a folder structure under GITHUB-PRACTICE containing files: data.json, todo.html, and apps.py (3 changes).
- Terminal:** Displays a PowerShell session with the following command history:

```
PS C:\GITHUB\GITHUB-PRACTICE> git add .
>> git commit -m "Added /submittodoitem backend API in master_2"
>> git push origin master_2
>>
[master_2 0fd9139] Added /submittodoitem backend API in master_2
  1 file changed, 14 insertions(+)
  Enumerating objects: 5, done.
● Counting objects: 100% (5/5), done.
  Delta compression using up to 4 threads
  Compressing objects: 100% (3/3), done.
  Writing objects: 100% (3/3), 656 bytes | 656.00 KiB/s, done.
  Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
  remote: Resolving deltas: 100% (1/1), completed with 1 local object.
  remote:
  remote: Create a pull request for 'master_2' on GitHub by visiting:
  remote:     https://github.com/Mohit7819/GITHUB-PRACTICE/pull/new/master_2
  remote:
  To github.com:Mohit7819/GITHUB-PRACTICE.git
   * [new branch]      master_2 -> master_2
PS C:\GITHUB\GITHUB-PRACTICE>
```
- Bottom Status Bar:** powershell - GITHUB-PRACTICE +

◆ Step 4: Merge `master_1` and `master_2` into `main`

1. First, switch to `main`:

```
git checkout main
```

2. Merge `master_1`:

```
git merge master_1
```

✓ Resolve any conflicts (if any), commit the resolved version:

```
git add .
```

```
git commit -m "Merged master_1 into main"
```

3. Then merge master_2:

```
git merge master_2
```

✓ Resolve conflicts (if any), commit:

```
git add .
```

```
git commit -m "conflict was fix"
```

4. Push final merged code to GitHub:

```
git push origin main
```

The screenshot shows the VS Code interface with the terminal tab active, displaying the following command history:

```
PS C:\GITHUB\GITHUB-PRACTICE> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● PS C:\GITHUB\GITHUB-PRACTICE> git merge master_1
Updating 4e0dd8f..e0266e4
Fast-forward
 apps.py | 3 +++
 todo.html | 16 ++++++
 2 files changed, 19 insertions(+)
 create mode 100644 todo.html
● PS C:\GITHUB\GITHUB-PRACTICE> git merge master_2
Auto-merging apps.py
CONFLICT (content): Merge conflict in apps.py
Automatic merge failed; fix conflicts and then commit the result.
● PS C:\GITHUB\GITHUB-PRACTICE> git add .
● PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "conflict was fix"
[main 051dd2d] conflict was fix
● PS C:\GITHUB\GITHUB-PRACTICE> git push origin main
Already up to date.
● PS C:\GITHUB\GITHUB-PRACTICE> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 409 bytes | 409.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
● remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:Mohit7819/GITHUB-PRACTICE.git
 4e0dd8f..051dd2d main -> main
PS C:\GITHUB\GITHUB-PRACTICE>
```

The Explorer sidebar shows files like data.json, todo.html, and apps.py. The Copilot sidebar is visible on the right.

4. Enhancing the To-Do Form in `master_1`:

- In the `master_1` branch, add the following fields to the To-Do form:
 - **Item ID**
 - **Item UUID**
 - **Item Hash**
- **Committing in Sequence:**
- Add and commit each field separately in the following order:
 - **First commit:** Add **Item ID** field.
 - **Second commit:** Add **Item UUID** field.
 - **Third commit:** Add **Item Hash** field.
- **Merging to `main`:**
- Merge the `master_1` branch into the `main` branch.
- **Git Reset and Commit Deletion:**
- In the `main` branch, use **Git Reset** to roll back to the commit where only the **Item ID** field was added.
- Use `git reset --soft` to ensure changes remain staged.
- Re-commit this state to the `main` branch.
- Merge this updated state to the `main` branch.
- **Rebasing Changes:**
- Rebase the updated changes in the `main` branch to the `master_1` branch.

Clarification:

- During rebasing, **preserve individual commits** to maintain the commit history for each change (i.e., do not squash commits).
- Use `git rebase main master_1` to integrate changes from the `main` branch back into the `master_1` branch.

◆ Step 1: Add Fields to To-Do Form in `master_1`

Make sure you're in the correct branch:

```
git checkout master_1
```

◆ First Commit: Add Item ID Field

Update `todo.html`:

```
<label>Item ID:</label><br>
<input type="text" name="itemId" required><br><br>
```

Then commit:

```
git add todo.html
```

```
git commit -m "Add Item ID field to To-Do form"
```

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure under 'GITHUB-PRACTICE' with files: apps.py, data.json, form.html, success.html, and todo.html. The todo.html file is open in the main editor area, showing its HTML code. The terminal at the bottom shows the command-line history for committing changes:

```
Switched to branch 'master_1'
PS C:\GITHUB\GITHUB-PRACTICE> git checkout master_1
PS C:\GITHUB\GITHUB-PRACTICE> git add todo.html
PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Add Item ID field to To-Do form"
[master_1 0266622] Add Item ID field to To-Do form
 1 file changed, 2 insertions(+)
PS C:\GITHUB\GITHUB-PRACTICE>
```

The right side of the interface features the 'Copilot' AI panel, which includes a 'Welcome to Copilot' message, a 'Let's get started' button, and options to 'Add context (#), external', 'Build workspace', and 'Show project config'. A note at the bottom of the panel says 'Review AI output carefully before use.'

◆ Second Commit: Add Item UUID Field

Add to `todo.html`:

```
<label>Item UUID:</label><br>
<input type="text" name="itemUUID" required><br><br>
```

Then commit:

```
git add todo.html
```

```
git commit -m "Add Item UUID field to To-Do form"
```

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows files in the 'GITHUB-PRACTICE' folder: apps.py, data.json, form.html, success.html, and todo.html.
- Editor:** The 'todo.html' file is open, displaying the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>To-Do Page</title>
</head>
<body>
    <h1>Create a To-Do Item</h1>
    <form action="/submittodoitem" method="post">
        <label>Item ID:</label><br>
        <input type="text" name="itemId" required><br><br>
        <label>Item UUID:</label><br>
        <input type="text" name="itemUUID" required><br><br>
        <label>Item Name:</label><br>
        <input type="text" name="itemName" required><br><br>
        <label>Item Description:</label><br>
        <textarea name="itemDescription" required></textarea><br><br>
        <button type="submit">Submit</button>
    </form>
```
- Terminal:** Shows the command history for committing changes to 'todo.html':
 - PS C:\GITHUB\GITHUB-PRACTICE> git add todo.html
 - PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Add Item ID field to To-Do form"
 - [master_1 0266622] Add Item ID field to To-Do form
1 file changed, 2 insertions(+)
 - PS C:\GITHUB\GITHUB-PRACTICE> git add todo.html
 - PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Add Item UUID field to To-Do form"
 - [master_1 aa92bfd] Add Item UUID field to To-Do form
1 file changed, 2 insertions(+)
 - PS C:\GITHUB\GITHUB-PRACTICE>
- Status Bar:** Shows the current branch is 'master_1', the commit hash is 'aa92bfd', and the file has 0 changes.

◆ Third Commit: Add Item Hash Field

Add to `todo.html`:

```
<label>Item Hash:</label><br>

<input type="text" name="itemHash" required><br><br>
```

Then commit:

```
git add todo.html
```

```
git commit -m "Add Item Hash field to To-Do form"
```

The screenshot shows the VS Code interface with the GitHub extension installed. The Explorer sidebar has a 'GITHUB' folder expanded, containing files such as 'apps.py', 'data.json', 'form.html', 'success.html', and 'todo.html'. The 'todo.html' file is currently selected and open in the main editor area, showing its HTML code. The bottom right corner of the editor shows a small preview of the rendered HTML page. The bottom of the screen features the standard VS Code navigation bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing a PowerShell session with several git commands run:

```

1 file changed, 2 insertions(+)
PS C:\GITHUB\GITHUB-PRACTICE> git add todo.html
● PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Add Item UUID field to To-Do form"
● [master_1 aa92bfdf] Add Item UUID field to To-Do form
1 file changed, 2 insertions(+)
PS C:\GITHUB\GITHUB-PRACTICE> git add todo.html
● PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Add Item Hash field to To-Do form"
● [master_1 705ad55] Add Item Hash field to To-Do form
1 file changed, 2 insertions(+)
○ PS C:\GITHUB\GITHUB-PRACTICE>

```

◆ Step 2: Merge master_1 to main

```
git checkout main
```

```
git merge master_1
```

```
git push origin main
```

◆ Step 3: Git Reset in main to Keep Only First Commit

1. View commit history:

```
git log --oneline
```

The screenshot shows the VS Code interface with the GitHub extension installed. The bottom right corner features a PowerShell terminal window. The command 'git log --oneline' was run, displaying the commit history. The first commit, which added the Item UUID field, is highlighted in yellow.

```

● PS C:\GITHUB\GITHUB-PRACTICE> git log --oneline
● e4c66a0 (HEAD -> main) conflict was fix
3494727 (origin/main) todo.html is updated
051d2df conflict was fix
0fd9139 (origin/master_2, master_2) Added /submittodoitem backend API in master_2
e0266e4 (origin/master_1) Added frontend To-Do page in master_1
4e0dd8f (origin/mohit_new, mohit_new) Updated JSON content in yourname_new branch
f36c125 (origin/singh, singh) flask project added
○ PS C:\GITHUB\GITHUB-PRACTICE>

```

2. Reset to that commit (keep changes staged):

```
git reset --soft 3494727
```

3. Recommit only the Item ID state:

```
git commit -m "Keep only Item ID field after reset"
```

```
git push origin main --force
```

```
● PS C:\GITHUB\GITHUB-PRACTICE> git reset --soft 3494727
PS C:\GITHUB\GITHUB-PRACTICE> git commit -m "Keep only Item ID field after reset"
● >> git push origin main --force
[main b16a05b] Keep only Item ID field after reset
  1 file changed, 1 insertion(+), 1 deletion(-)
  Enumerating objects: 5, done.
  Counting objects: 100% (5/5), done.
  Delta compression using up to 4 threads
  Compressing objects: 100% (3/3), done.
  Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
  Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
  remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
  To github.com:Mohit7819/GITHUB-PRACTICE.git
    3494727..b16a05b  main -> main
○ PS C:\GITHUB\GITHUB-PRACTICE> █
```

◆ Step 4: Rebase `main` into `master_1` (Preserve Commit History)

Switch to `master_1`:

```
git checkout master_1
```

Then rebase:

```
git rebase main
```

 During rebase:

- You may see interactive options in the editor.
- Just **keep all commits as-is** to preserve the history.
- If conflicts arise, resolve them, then continue:

```
+ file changes, + insertion(s), + deletion(s)
PS C:\GITHUB\GITHUB-PRACTICE> git checkout master_1
Switched to branch 'master_1'
● PS C:\GITHUB\GITHUB-PRACTICE> git branch
  main
* master_1
  master_2
  mohit_new
  singh
② PS C:\GITHUB\GITHUB-PRACTICE> git rebase main
● CONFLICT (content): Merge conflict in todo.html
  error: could not apply 0266622... Add Item ID field to To-Do form
  hint: Resolve all conflicts manually, mark them as resolved with
  hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
  hint: You can instead skip this commit: run "git rebase --skip".
  hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
  hint: Disable this message with "git config set advice.mergeConflict false"
  Could not apply 0266622... # Add Item ID field to To-Do form
PS C:\GITHUB\GITHUB-PRACTICE> git rebase main
● Current branch master_1 is up to date.
○ PS C:\GITHUB\GITHUB-PRACTICE> █
```

git add .

```
git rebase --continue
```

```
PS C:\GITHUB\GITHUB-PRACTICE> git add .
>> git rebase --continue
>>
fatal: no rebase in progress
○ PS C:\GITHUB\GITHUB-PRACTICE>
```