Part 1 – GitHub Repo, Branching, and Flask Project

1. Create a New GitHub Repository Command/Action:

Go to GitHub.com

Click "New" to create a repository.

Choose repository name (e.g., flask-mongo-form). Select Public or Private as required.

Click "Create repository".

Explanation:

This step sets up a central place on GitHub where your project will be stored and version- controlled. By creating a repository, you prepare a space for your code, documentation, and any updates to be tracked and shared with others.

1. Clone Repository Using SSH Commands:

# Generate SSH Key

ssh-keygen -t rsa -b 4096 -C "[your\_email@example.com](mailto:your_email@example.com)"

# Start ssh-agent

eval "$(ssh-agent -s)" ssh-add ~/.ssh/id\_rsa

# Copy Public Key to GitHub cat ~/.ssh/id\_rsa.pub

Go to GitHub → Settings → SSH and GPG keys → New SSH Key → Paste the copied key →

Save.

# Clone Repository

git clone git@github.com:username/repo\_name.git cd repo\_name

Explanation:

SSH keys allow secure, password-free communication between your local machine and GitHub. First, we generate a key (ssh-keygen) and add it to the SSH agent.

We then upload the public key to GitHub so GitHub can authenticate our machine.

Finally, we clone the repository using SSH, allowing us to push and pull changes securely.

1. Create a Branch with Your Username Command:

git checkout -b sahal

Explanation:

A branch is like a separate workspace in Git. By creating a branch named after your username, you can work on your features or files without affecting the main branch. This helps in keeping your main branch clean and stable until changes are reviewed and merged.

1. Add Flask Project Files Example app.py:

from flask import Flask, jsonify app = Flask( name )

@app.route("/") def home():

return "Hello, Flask!"

@app.route("/api") def api():

return jsonify({"message": "API route works!"})

if name == " main ": app.run(debug=True)

Explanation:

This is a simple Flask application:

/ route → Returns a plain text message "Hello, Flask!".

/api route → Returns a JSON response {"message": "API route works!"}.

The debug=True flag allows automatic reloads and shows detailed error messages during development.

Adding these files to your branch prepares your project for deployment or further development.

1. Commit and Merge Commands:

git add .

git commit -m "Added Flask project" git checkout main

git merge sahal

git push origin main Explanation:

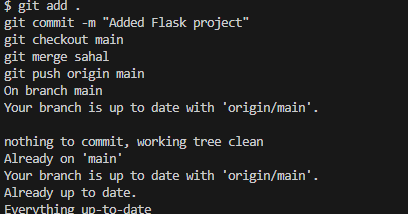
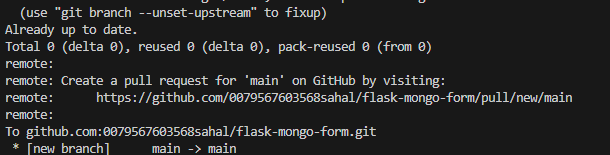
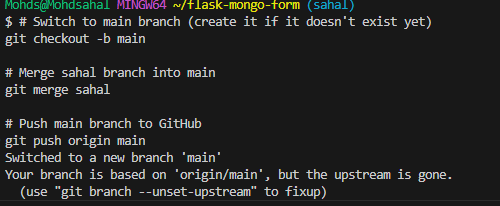
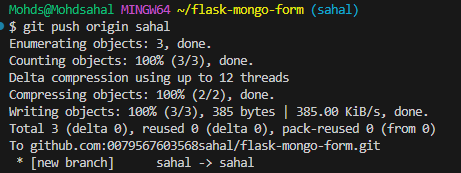
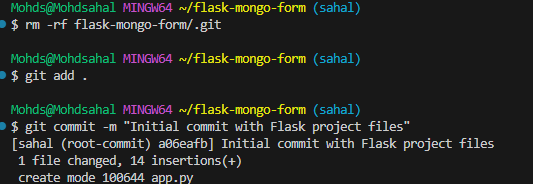
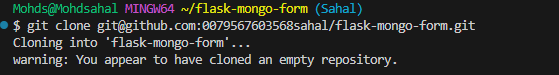
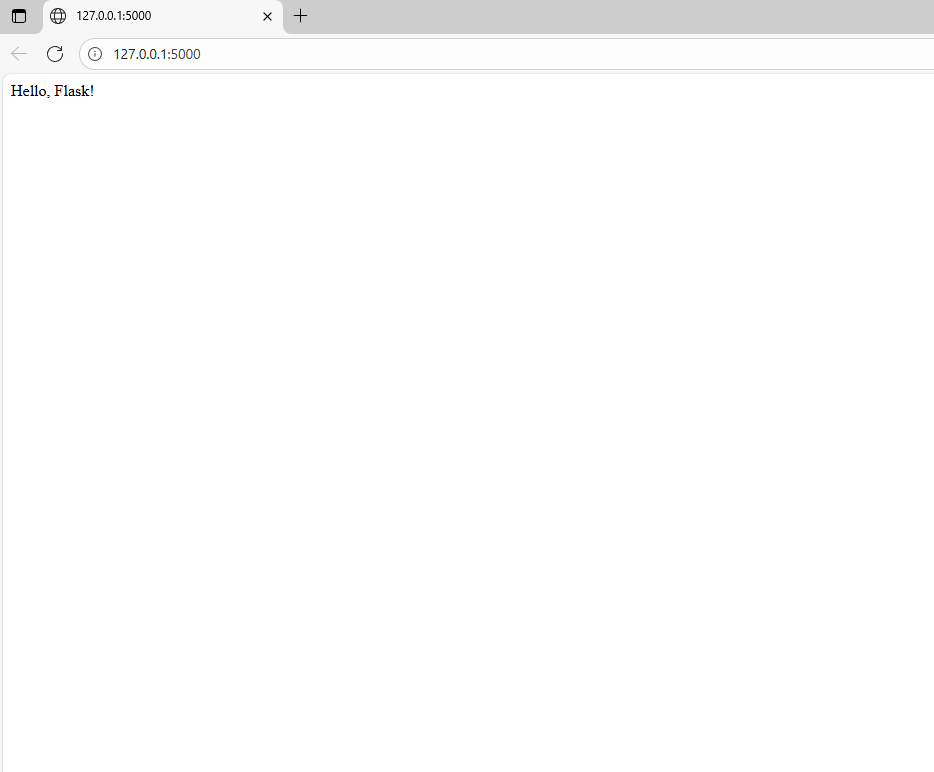
git add . → Stages all changes for commit.

git commit -m "..." → Saves the staged changes to Git history with a message.

git checkout main → Switches to the main branch.

git merge sahal → Merges the work from your branch into the main branch. git push origin main → Sends the updated main branch to GitHub.

# Screenshots



Part 2 – JSON Update and Conflict Resolution

1. Create a new branch named sahal\_new git checkout -b sahal\_new

Explanation:

This command creates a new branch called sahal\_new and switches to it immediately. Branches allow you to work on changes without affecting the main branch.

1. Edit data.json (used in /api route)

{

"message": "Updated content for Sahal"

}

Explanation:

Here, the data.json file is modified to update its message. This change will be committed to the new branch. The /api route in your Flask app likely reads this file to display updated content.

1. Stage the modified file for commit git add data.json

Explanation:

This stages the updated data.json file, telling Git to include it in the next commit.

1. Commit the changes

git commit -m "Updated JSON file in sahal\_new" Explanation:

This creates a commit in the sahal\_new branch with a message describing the change.

1. Switch back to the main branch git checkout main

Explanation:

Changes will now be merged into the main branch, so we switch back from sahal\_new to main.

1. Merge sahal\_new into main

git merge sahal\_new Explanation:

This merges the changes from sahal\_new into main. If there are no conflicts, the merge completes automatically.

1. If there is a merge conflict

# Open conflicted file → keep sahal\_new changes → save

git add data.json

git commit -m "Resolved merge conflict using sahal\_new version" Explanation:

If Git detects conflicting changes in data.json, it will ask you to resolve them manually. Open the file, choose the sahal\_new changes, and save it.

Stage and commit the resolved file.

1. Push changes to remote repository git push origin main

Explanation:

This uploads your updated main branch to the remote repository (e.g., GitHub). Part 3 – Feature Development

1. Ensure you are on the main branch git checkout main

Explanation:

Always start new features from the latest main branch to avoid outdated code.

1. Create a new branch master\_1 git checkout -b master\_1 Explanation:

Creates and switches to a new branch master\_1, where you can work on a specific feature independently.

1. Switch back to main git checkout main

Explanation:

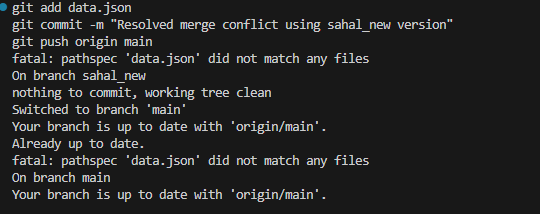
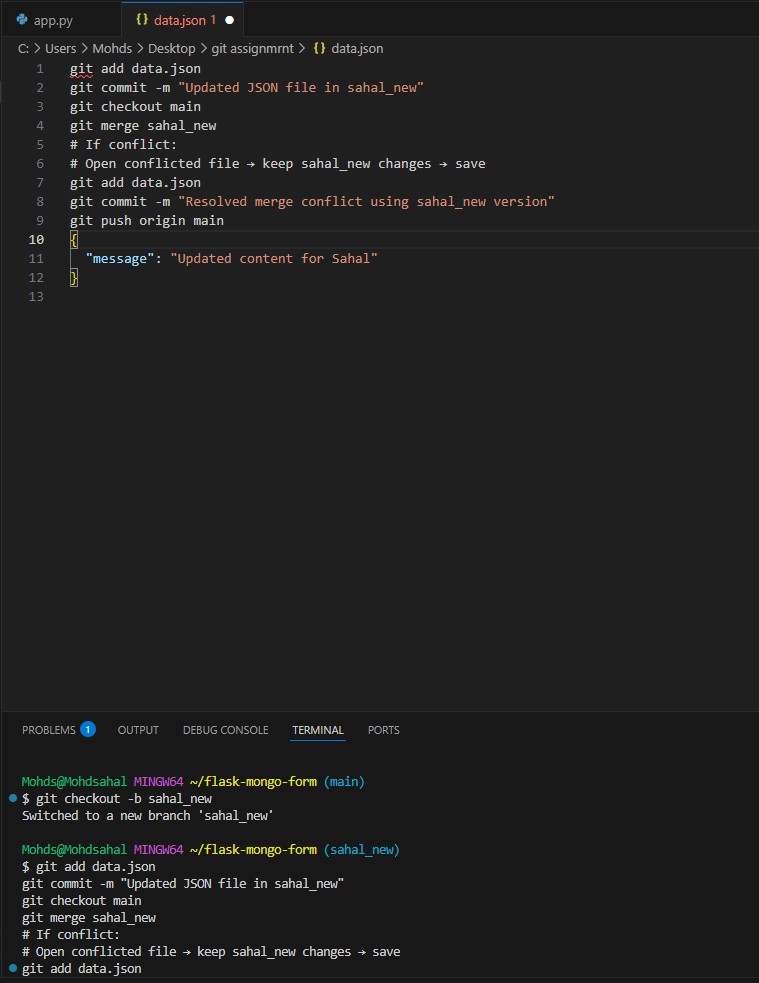
Return to the main branch to start another feature branch.

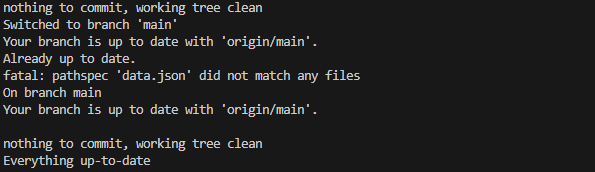
1. Create another branch master\_2 git checkout -b master\_2

Explanation:

Creates and switches to another feature branch named master\_2

# Screenshots





Part 3 – Feature Development App.py

# Import required libraries

from flask import Flask, render\_template, request, redirect, url\_for from pymongo import MongoClient

import certifi import os

# Load MongoDB URI from environment variable MONGO\_URI = os.getenv(

"MONGO\_URI",

"[mongodb+srv://sahalusr1:securepass@cluster0.bdnyrmo.mongodb.net/?retryWrites=true&w=m](mailto:securepass@cluster0.bdnyrmo.mongodb.net) ajority&appName=Cluster0"

)

# Create the Flask app app = Flask( name )

# Route to handle form submission @app.route("/submittodoitem", methods=["POST"]) def submit\_todo\_item():

"""

Handles form submission from your HTML form.

Inserts the submitted itemName and itemDescription into MongoDB.

"""

# Get the submitted data from the form item\_name = request.form.get("itemName")

item\_description = request.form.get("itemDescription")

if item\_name and item\_description: # Connect to MongoDB

client = MongoClient(MONGO\_URI, tlsCAFile=certifi.where()) db = client["todo\_database"]

collection = db["todos"]

# Insert the submitted item into MongoDB collection.insert\_one({

"name": item\_name, "description": item\_description

})

# Redirect back to the home page showing all tasks return redirect(url\_for("show\_todo\_list"))

# Route to display the todo list @app.route("/")

def show\_todo\_list():

# Connect to MongoDB

client = MongoClient(MONGO\_URI, tlsCAFile=certifi.where()) db = client["todo\_database"]

collection = db["todos"]

# Fetch all tasks

tasks = collection.find()

# Render tasks to HTML template

return render\_template("todo.html", tasks=tasks) # Run the Flask app

if name == " main ": app.run(debug=True)

Todo.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Todo List</title>

<link href="[https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.3.0/dist/css/bootstrap.min.css)" rel="stylesheet">

</head>

<body class="container mt-5">

<h1 class="mb-4">Todo List</h1>

<!-- Form to add new todo -->

<form action="{{ url\_for('submit\_todo\_item') }}" method="post" class="mb-4">

<div class="mb-3">

<input type="text" name="itemName" class="form-control" placeholder="Item Name" required>

</div>

<div class="mb-3">

<textarea name="itemDescription" class="form-control" placeholder="Item Description" required></textarea>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

<!-- Display todo list -->

<ul class="list-group">

{% for task in tasks %}

<li class="list-group-item">

<strong>{{ task.name }}</strong>: {{ task.description }}

</li>

{% endfor %}

</ul>

</body>

</html>

Explanation

we create two separate branches for different features of the To-Do application:

master\_1 → Frontend (HTML Form) master\_2 → Backend (MongoDB Integration)

This approach allows parallel development without interfering with each other’s work.

Step 1 – Create Feature Branches git checkout main

git checkout -b master\_1 git checkout main

git checkout -b master\_2 Explanation:

Switch to the main branch to ensure you start from the latest code. Create a new branch master\_1 for the frontend work.

Switch back to main again.

Create another branch master\_2 for the backend work. Step 2 – Develop master\_1 (To-Do Frontend)

File: templates/todo.html

<form action="/submittodoitem" method="post">

<input type="text" name="itemName" placeholder="Item Name"><br>

<textarea name="itemDescription" placeholder="Item Description"></textarea><br>

<button type="submit">Submit</button>

</form> Explanation:

This HTML form collects item name and description from the user.

When submitted, it sends data to the /submittodoitem route using the POST method. The form will later be processed by the backend in master\_2.

Git Commands to Commit and Push:

git add templates/todo.html

git commit -m "Added To-Do page frontend" git push origin master\_1

Explanation:

git add stages the new todo.html file.

git commit saves the changes locally with a meaningful message. git push uploads the branch to GitHub.

Step 3 – Develop master\_2 (Backend MongoDB) Code Update in app.py:

from flask import request

from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/") db = client.todoDB

@app.route("/submittodoitem", methods=["POST"]) def submit\_todo():

name = request.form["itemName"]

desc = request.form["itemDescription"] db.todos.insert\_one({"name": name, "description": desc}) return "Item Added!"

Explanation:

Connects to a local MongoDB database (todoDB).

Defines a /submittodoitem route that accepts POST requests. Reads itemName and itemDescription from the form.

Inserts them into the todos collection in MongoDB. Returns a confirmation message "Item Added!".

Git Commands to Commit and Push:

git add app.py

git commit -m "Added backend route to store To-Do item in MongoDB" git push origin master\_2

Explanation:

Stages the updated app.py.

Commits with a descriptive message. Pushes master\_2 branch to GitHub. Step 4 – Merge Both Features into Main

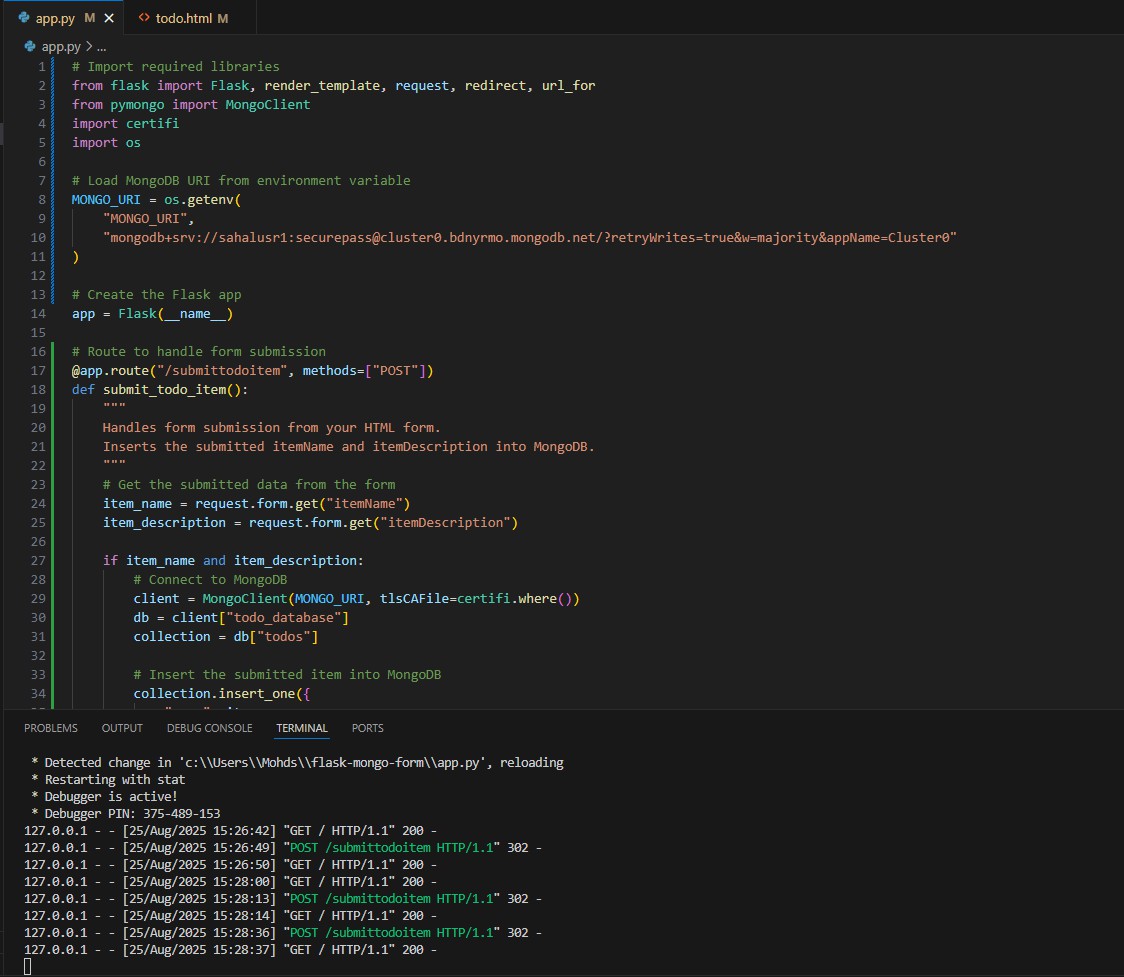
git checkout main git merge master\_1 git merge master\_2 git push origin main

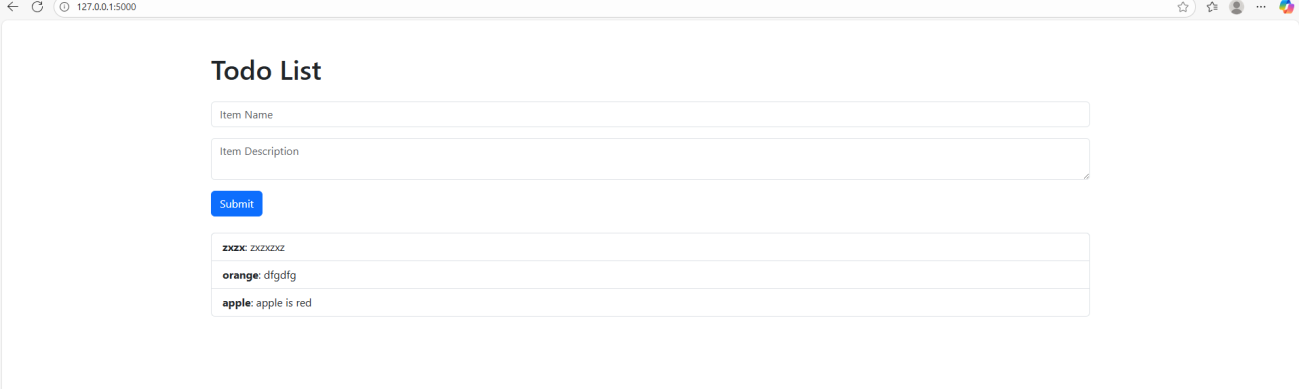
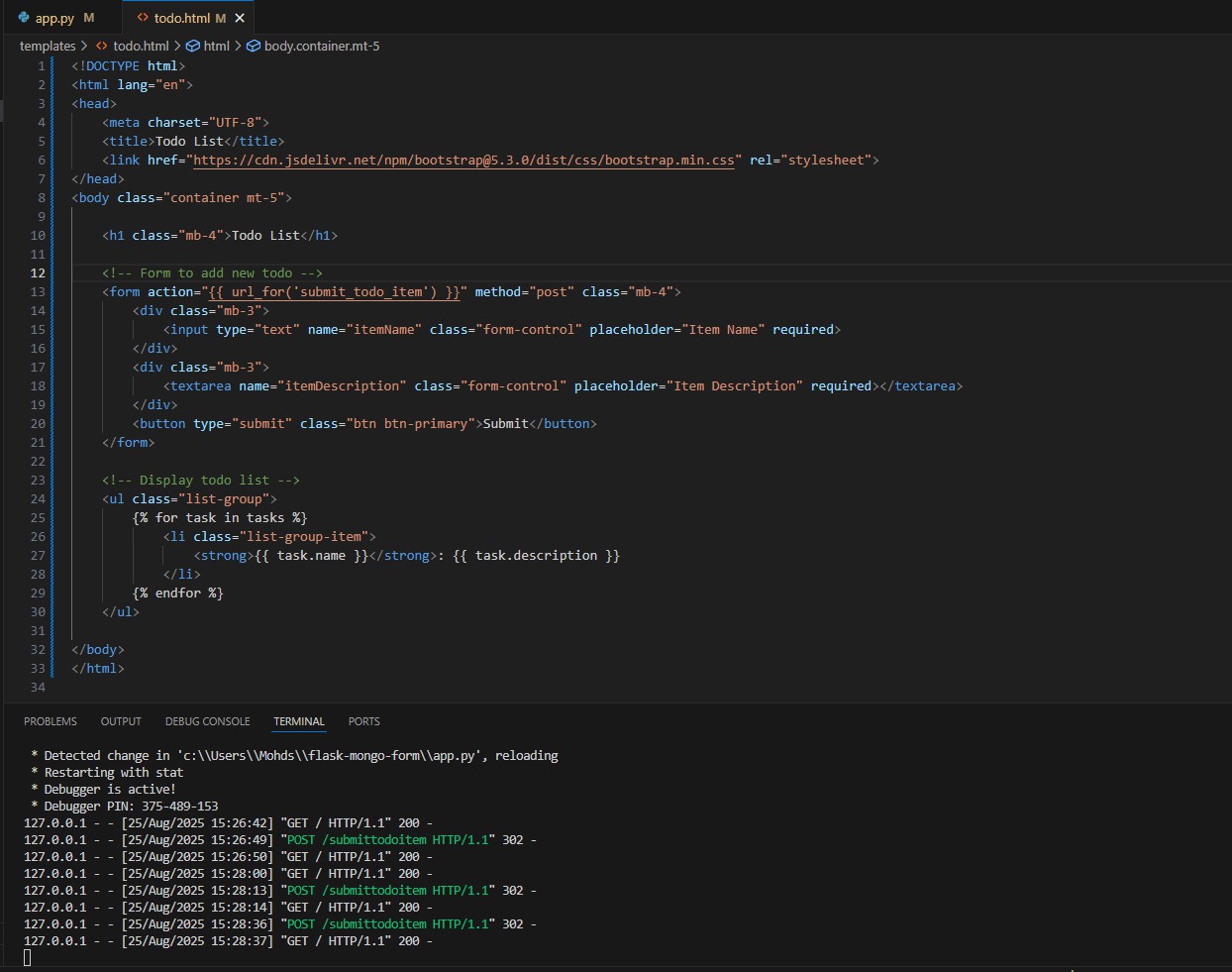
Explanation:

Switch back to main.

Merge master\_1 (frontend) changes into main. Merge master\_2 (backend) changes into main.

Push the updated main branch to GitHub so it contains both frontend and backend functionality. Screenshots





Part 4 – Enhancements + Git Reset + Rebase

from flask import Flask, render\_template, request, redirect, url\_for from pymongo import MongoClient import certifi import os

Load MongoDB URI from environment variable or use default MONGO\_URI = os.getenv( "MONGO\_URI",

"[mongodb+srv://sahalusr1:securepass@cluster0.bdnyrmo.mongodb.net/?retryWrites=true&w=m](mailto:securepass@cluster0.bdnyrmo.mongodb.net) ajority&appName=Cluster0" )

Initialize Flask app app = Flask(name)

Initialize MongoDB client and collection

client = MongoClient(MONGO\_URI, tlsCAFile=certifi.where()) db = client["todo\_database"] collection = db["todos"]

Route to display the todo list

@app.route("/") def index(): tasks = list(collection.find()) return render\_template("to\_do.html", tasks=tasks)

Route to handle form submission

@app.route("/submittodoitem", methods=["POST"]) def submit\_todo\_item(): # Collect all fields from form itemID = request.form.get("itemID") itemUUID = request.form.get("itemUUID") itemHash = request.form.get("itemHash") itemName = request.form.get("itemName") itemDescription = request.form.get("itemDescription")

if itemID and itemUUID and itemHash and itemName and itemDescription: # Insert into MongoDB

collection.insert\_one({ "itemID": itemID, "itemUUID": itemUUID, "itemHash": itemHash, "name": itemName, "description": itemDescription

})

# Redirect back to the todo list page return redirect(url\_for("index"))

Run the Flask app

if name == "main": app.run(debug=True)

Todo.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Todo List</title>

<link href="[https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.3.0/dist/css/bootstrap.min.css)" rel="stylesheet">

</head>

<body class="container mt-5">

<h1 class="mb-4 text-center">Todo List</h1>

<!-- Form to add new todo -->

<form action="{{ url\_for('submit\_todo\_item') }}" method="post" class="mb-5">

<div class="row g-3">

<div class="col-md-4">

<input type="text" name="itemID" class="form-control" placeholder="Item ID"

required>

</div>

<div class="col-md-4">

<input type="text" name="itemUUID" class="form-control" placeholder="Item UUID" required>

</div>

<div class="col-md-4">

<input type="text" name="itemHash" class="form-control" placeholder="Item Hash"

required>

</div>

</div>

<div class="mt-3">

<input type="text" name="itemName" class="form-control mb-3" placeholder="Item Name" required>

<textarea name="itemDescription" class="form-control" placeholder="Item Description" rows="3" required></textarea>

</div>

<button type="submit" class="btn btn-primary mt-3">Submit</button>

</form>

<!-- Display todo list -->

<ul class="list-group">

{% for task in tasks %}

<li class="list-group-item">

<div class="mb-2">

<span class="badge bg-secondary">ID: {{ task.itemID }}</span>

<span class="badge bg-info text-dark">UUID: {{ task.itemUUID }}</span>

<span class="badge bg-warning text-dark">Hash: {{ task.itemHash }}</span>

</div>

<strong>{{ task.name }}</strong>: {{ task.description }}

</li>

{% endfor %}

</ul>

</body>

</html>

Enhancements in master\_1

git checkout master\_1

# 1st commit - Add Item ID field # (Edit todo.html)

<input type="text" name="itemID" placeholder="Item ID"> git add templates/todo.html

git commit -m "Added Item ID field"

# 2nd commit - Add Item UUID field

<input type="text" name="itemUUID" placeholder="Item UUID"> git add templates/todo.html

git commit -m "Added Item UUID field"

# 3rd commit - Add Item Hash field

<input type="text" name="itemHash" placeholder="Item Hash"> git add templates/todo.html

git commit -m "Added Item Hash field" git push origin master\_1

Merge into main

git checkout main git merge master\_1

Git Reset to first commit (Item ID only)

git log # find commit hash for "Added Item ID field" git reset --soft <commit\_hash>

git commit -m "Revert to only Item ID field" git push origin main --force

Rebase changes

git checkout master\_1 git rebase main

git push origin master\_1 --force

Explanation

1. Enhancements in master\_1 git checkout master\_1

Switches to the master\_1 branch where we will make frontend enhancements. First commit – Add Item ID field

<input type="text" name="itemID" placeholder="Item ID">

git add templates/todo.html

git commit -m "Added Item ID field"

A new input field for Item ID is added to the To-Do page.

git add stages the change, and git commit saves it as a snapshot in the branch history. Second commit – Add Item UUID field

<input type="text" name="itemUUID" placeholder="Item UUID">

git add templates/todo.html

git commit -m "Added Item UUID field"

Another input field Item UUID is added to the same HTML file.

This is committed as a separate commit to keep each enhancement logically independent. Third commit – Add Item Hash field

<input type="text" name="itemHash" placeholder="Item Hash">

git add templates/todo.html

git commit -m "Added Item Hash field"

A third enhancement: Item Hash input field.

Separate commits help in tracing changes and make future debugging easier. git push origin master\_1

Pushes all three commits from master\_1 to the remote repository.

1. Merge into main

git checkout main git merge master\_1

Switches to the main branch and merges changes from master\_1. Now main has all enhancements: Item ID, UUID, and Hash.

1. Git Reset to first commit (Item ID only)

git log # find commit hash for "Added Item ID field" git reset --soft <commit\_hash>

git commit -m "Revert to only Item ID field" git push origin main --force

Purpose: Undo all commits after the first one while keeping changes staged (--soft). Creates a new commit representing only the first enhancement (Item ID).

--force is used because the branch history is rewritten; it overwrites the remote history with this new state.

1. Rebase changes in master\_1 git checkout master\_1

git rebase main

git push origin master\_1 --force

Rebases master\_1 onto the current main.

Ensures that master\_1 incorporates the latest state of main (after reset) cleanly.

--force pushes the rebased branch since its history has been rewritten.

After this, master\_1 is synchronized with main but still retains its original commits, now applied on top of main.

Commands

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_2) $ cd C:\Users\Mohds\flask- mongo-form git checkout master\_1 bash: cd: C:UsersMohdsflask-mongo-form: No such file or directory error: pathspec 'master\_1' did not match any file(s) known to git

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_2) $ cd /c/Users/Mohds/flask- mongo-form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_2) $ git branch

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_2) $ git checkout main git pull origin main # make sure main is up to date git checkout -b master\_1 error: pathspec 'main' did not match any file(s) known to git fatal: 'origin' does not appear to be a git repository fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists. Switched to a new branch 'master\_1'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ templates/to\_do.html bash: templates/to\_do.html: No such file or directory

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ ls to\_do.html todo.html.txt Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ ls templates Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ code templates/to\_do.html

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ # First, stage only Item ID changes git add templates/to\_do.html git commit -m "Add Item ID field to To-Do form"

Then, stage Item UUID changes

git add templates/to\_do.html git commit -m "Add Item UUID field to To-Do form"

Then, stage Item Hash changes

git add templates/to\_do.html git commit -m "Add Item Hash field to To-Do form" [master\_1 (root-commit) ea4cec8] Add Item ID field to To-Do form 1 file changed, 51 insertions(+) create mode 100644 flask-mongo-form/templates/to\_do.html warning: could not open directory 'AppData/Local/Application Data/': Permission denied warning: could not open directory 'Application Data/': Permission denied warning: could not open directory 'Cookies/': Permission denied warning: could not open directory 'Local Settings/': Permission denied warning: could not open directory 'My Documents/': Permission denied warning: could not open directory 'NetHood/': Permission denied warning: could not open directory 'PrintHood/': Permission denied warning: could not open directory 'Recent/': Permission denied warning: could not open directory 'SendTo/': Permission denied warning: could not open directory 'Start Menu/': Permission denied warning: could not open directory 'Templates/': Permission denied On branch master\_1 Untracked files: (use "git add ..." to include in what will be

committed) ../.anaconda/ ../.conda/ ../.condarc ../.continuum/ ../.gitconfig ../.ipython/ ../.jupyter/ ..

/.keras/ ../.lesshst ../.matplotlib/ ../.spyder-

py3/ ../.ssh/ ../.vscode/ ../AppData/ ../Contacts/ ../Desktop/ ../Documents/ ../Downloads/ ../Favorit es/ ../Intel/ ../Links/ ../Music/ ../NTUSER.DAT ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c- 3887d58b62e8}.TM.blf ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c-

3887d58b62e8}.TMContainer00000000000000000001.regtrans- ms ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c-

3887d58b62e8}.TMContainer00000000000000000002.regtrans-

ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.0.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.1.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.2.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-

3887d58b62e4}.TxR.blf ../NTUSER.DAT{e35b560f-4c28-11f0-99cd-

3887d58b62e4}.TM.blf ../NTUSER.DAT{e35b560f-4c28-11f0-99cd-

3887d58b62e4}.TMContainer00000000000000000001.regtrans-ms ../NTUSER.DAT{e35b560f- 4c28-11f0-99cd-3887d58b62e4}.TMContainer00000000000000000002.regtrans-

ms ../OneDrive/ ../Pictures/ ../Saved

Games/ ../Searches/ ../Videos/ ../anaconda3/ ../day4\_youtube\_clone/ to\_do.html

todo.html.txt ../form.html ../miniconda3/ ../ntuser.dat.LOG1 ../ntuser.dat.LOG2 ../ntuser.ini ../ph p-docs-hello-world/

nothing added to commit but untracked files present (use "git add" to track) warning: could not open directory 'AppData/Local/Application Data/': Permission denied warning: could not open directory 'Application Data/': Permission denied warning: could not open directory 'Cookies/': Permission denied warning: could not open directory 'Local Settings/': Permission denied warning: could not open directory 'My Documents/': Permission denied warning: could not open directory 'NetHood/': Permission denied warning: could not open directory 'PrintHood/': Permission denied warning: could not open directory 'Recent/': Permission denied warning: could not open directory 'SendTo/': Permission denied warning: could not open directory 'Start Menu/': Permission denied warning: could not open directory 'Templates/': Permission denied On branch master\_1 Untracked files: (use "git add ..." to include in what will be

committed) ../.anaconda/ ../.conda/ ../.condarc ../.continuum/ ../.gitconfig ../.ipython/ ../.jupyter/ ..

/.keras/ ../.lesshst ../.matplotlib/ ../.spyder-

py3/ ../.ssh/ ../.vscode/ ../AppData/ ../Contacts/ ../Desktop/ ../Documents/ ../Downloads/ ../Favorit es/ ../Intel/ ../Links/ ../Music/ ../NTUSER.DAT ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c- 3887d58b62e8}.TM.blf ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c-

3887d58b62e8}.TMContainer00000000000000000001.regtrans- ms ../NTUSER.DAT{3e3a1365-569d-11f0-9a0c-

3887d58b62e8}.TMContainer00000000000000000002.regtrans-

ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.0.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.1.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-3887d58b62e4}.TxR.2.regtrans- ms ../NTUSER.DAT{e35b560e-4c28-11f0-99cd-

3887d58b62e4}.TxR.blf ../NTUSER.DAT{e35b560f-4c28-11f0-99cd-

3887d58b62e4}.TM.blf ../NTUSER.DAT{e35b560f-4c28-11f0-99cd-

3887d58b62e4}.TMContainer00000000000000000001.regtrans-ms ../NTUSER.DAT{e35b560f- 4c28-11f0-99cd-3887d58b62e4}.TMContainer00000000000000000002.regtrans-

ms ../OneDrive/ ../Pictures/ ../Saved

Games/ ../Searches/ ../Videos/ ../anaconda3/ ../day4\_youtube\_clone/ to\_do.html

todo.html.txt ../form.html ../miniconda3/ ../ntuser.dat.LOG1 ../ntuser.dat.LOG2 ../ntuser.ini ../ph p-docs-hello-world/

nothing added to commit but untracked files present (use "git add" to track)

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ cd /c/Users/Mohds/flask- mongo-form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master\_1) $ rm -rf .git # careful: deletes the current Git repo git init Initialized empty Git repository in C:/Users/Mohds/flask-mongo- form/.git/

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html git commit -m "Add Item ID field to To-Do form" [master (root-commit) 86b7ba4] Add Item ID field to To-Do form 1 file changed, 51 insertions(+) create mode 100644 templates/to\_do.html

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html git commit -m "Add Item Hash field to To-Do form" On branch master Untracked files: (use "git add ..." to include in what will be committed) todo.html.txt

nothing added to commit but untracked files present (use "git add" to track)

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git checkout -b main Switched to a new branch 'main'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git merge master Already up to date.

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git log --oneline 86b7ba4 (HEAD

-> main, master) Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git checkout main git reset --soft 86b7ba4 Already on 'main'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git commit -m "Recommit To-Do form with only Item ID field" On branch main Untracked files: (use "git add ..." to include in what will be committed) todo.html.txt

nothing added to commit but untracked files present (use "git add" to track)

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git checkout master Switched to branch 'master'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git rebase main Current branch master is up to date.

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html git rebase --continue fatal: no rebase in progress

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git log --oneline 86b7ba4 (HEAD -> master, main) Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html git commit -m "Add Item UUID field to To-Do form" On branch master Untracked files: (use "git add ..." to include in what will be committed) todo.html.txt

nothing added to commit but untracked files present (use "git add" to track) Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git commit -m "Add Item UUID field to To-Do form" On branch master Untracked files: (use "git add ..." to include in what will be committed) todo.html.txt

nothing added to commit but untracked files present (use "git add" to track) Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git commit -m "Add Item UUID field to To-Do form" [master d8b74f0] Add Item UUID field to To-Do form 1 file changed, 9 insertions(+), 3 deletions(-)

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git add templates/to\_do.html git commit -m "Add Item Hash field to To-Do form" [master 9302e85] Add Item Hash field to To- Do form 1 file changed, 3 insertions(+)

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git checkout -b main fatal: a branch named 'main' already exists

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git checkout main Switched to branch 'main'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git log --oneline 86b7ba4 (HEAD

-> main) Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git reset --hard 86b7ba4 HEAD is now at 86b7ba4 Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git log --oneline 86b7ba4 (HEAD

-> main) Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (main) $ git checkout master Switched to branch 'master'

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git rebase main Current branch master is up to date.

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $ git log --oneline 9302e85 (HEAD -> master) Add Item Hash field to To-Do form d8b74f0 Add Item UUID field to To-Do form 86b7ba4 (main) Add Item ID field to To-Do form

Mohds@Mohdsahal MINGW64 ~/flask-mongo-form (master) $

Screenshots

