

# Lab: GitHub Automation & AI Reviewer

---

**Objective:** Build a script that uses PyGithub and Gemini to automatically review Pull Requests.

## Prerequisites

1. **GitHub Account.**

2. **Personal Access Token (PAT):**

- Settings → Developer Settings → Personal access tokens → Fine-grained tokens.
- Permissions: Read/Write Access to "Pull Requests", "Issues", "Code".

3. **Gemini API Key.**

## Task 1: PyGithub Warmup

1. Install `PyGithub`: `pip install PyGithub`.
2. Write a script `list_repos.py`:
  - Authenticate.
  - Print the names and URL of your last 5 created repositories.
3. Create a **new repository** programmatically named `test-automation-lab`.
  - Add a `README.md` via the API.

## Task 2: The Setup

1. In your new `test-automation-lab` repo:
  - Create a file `calculator.py` on `main`.
  - Create a new branch `bad-code`.
  - Modify `calculator.py` to add a bug (e.g., `def add(a, b): return a - b`).
  - Create a **Pull Request** from `bad-code` to `main`.
2. Note the PR number (e.g., #1).

## Task 3: Fetching the Diff

Write a script `fetch_pr.py` :

1. Get the repo.
2. Get the PR object using the number.
3. Iterate through `pr.get_files()`.
4. Print the `filename` and the `patch` (the diff).

### Expected Output:

```
File: calculator.py
Patch:
@@ -1,2 +1,2 @@
  def add(a, b):
-     return a + b
+     return a - b
```

## Task 4: The AI Reviewer

Integrate Gemini:

### 1. Construct a prompt:

```
You are a senior code reviewer. Review the following patch for bugs and logic errors.  
Be concise. If the code is buggy, explain why.
```

```
Patch:
```

```
{file_patch}
```

### 2. Send to Gemini API.

### 3. Print the response.

## Task 5: Posting the Review

Close the loop:

1. Take the LLM response.
2. Use `pr.create_issue_comment(body=llm_response)` to post the review as a general comment on the PR.
  - *Advanced:* Use `create_review_comment` to post on the specific line number (requires calculating position from the diff).

**Run the script and check your GitHub PR! You should see your bot's comment.**

## Challenge: GraphQL "Stargazer" Analytics

Write a script using `requests` and GraphQL to:

1. Find your most starred repository.
2. Fetch the last 10 stargazers.
3. Print: "Your repo X was starred by: [User1, User2...]"

Use the [GitHub GraphQL Explorer](#) to build your query first.