

Introduction

For this assignment, I chose to work with the GitHub REST API because it provides real-world authentication, supports full CRUD (Create, Read, Update, Delete) operations, and is widely used in software development. The API base URL is <https://api.github.com>.

Objectives:

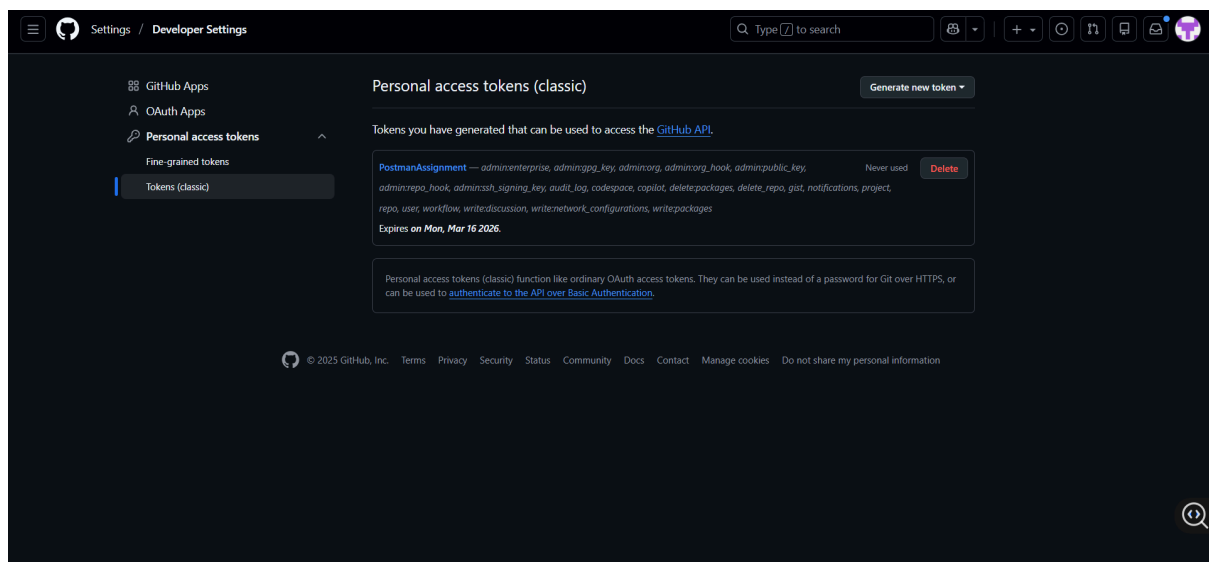
1. Understand API authentication using tokens
2. Implement CRUD operations (Create, Read, Update, Delete)
3. Use Postman environments and variables for dynamic testing
4. Automate testing with Postman Collection Runner
5. Document the complete workflow

Authentication Setup

I used Personal Access Token authentication with the Bearer Token method.

Steps Taken:

1. Generated a token from GitHub → Settings → Developer Settings
2. Selected repo and user scopes for necessary permissions
3. Stored the token securely in Postman environment variables as `auth_token`
4. Configured all requests to use: Authorization: Bearer `{{auth_token}}`



Screenshot 1: GitHub token generation page

API Endpoints Documentation

Endpoint	Method	Purpose	Headers Required	Body Format
/user	GET	Get authenticated user info	Authorization: Bearer {token}	None
/user/repos	POST	Create new repository	Authorization: Bearer {token}	JSON: name, description, private
/repos/{owner}/{repo}	PATCH	Update repository	Authorization: Bearer {token}	JSON: updated fields
/repos/{owner}/{repo}	DELETE	Delete repository	Authorization: Bearer {token}	None

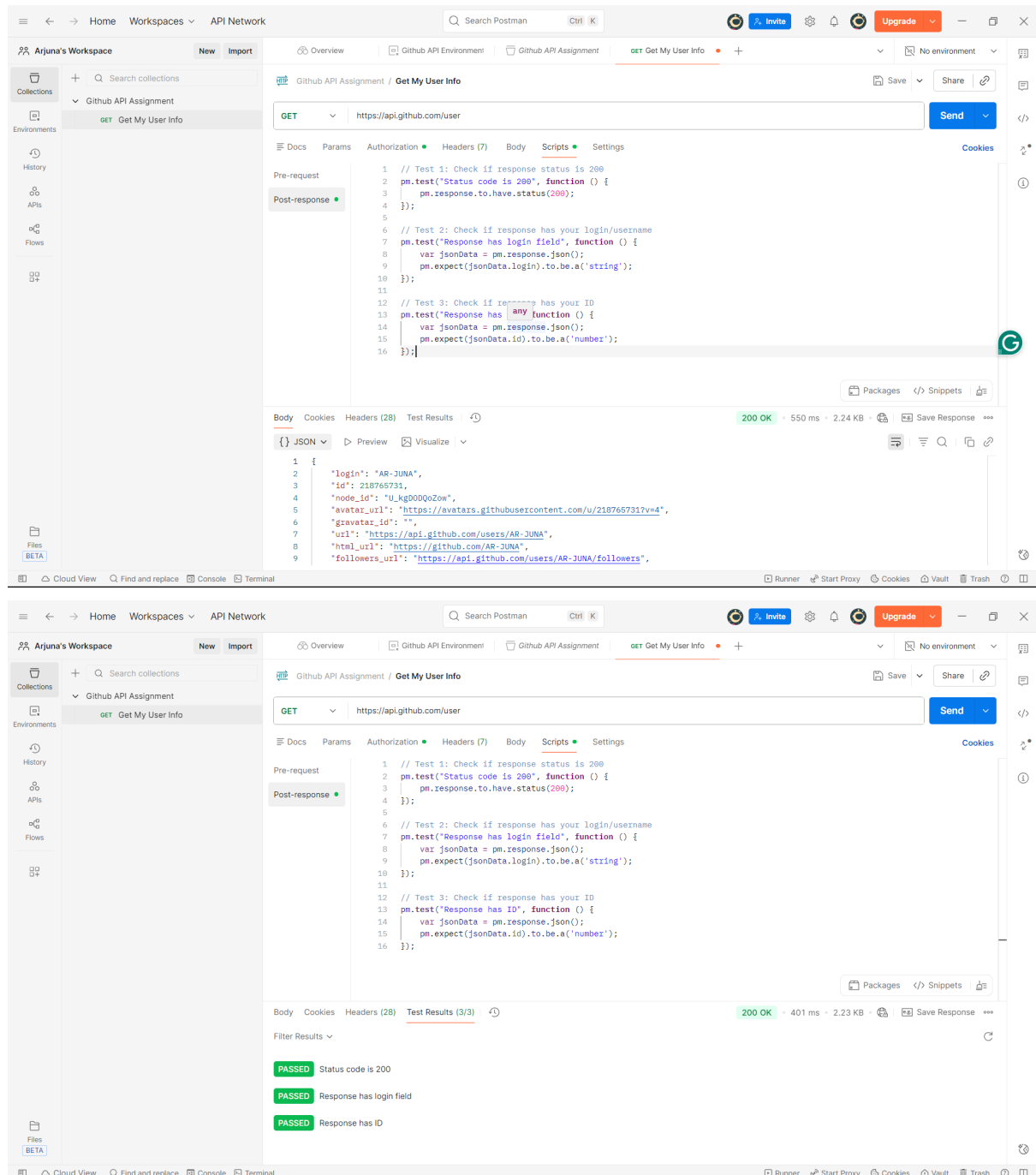
CRUD Operations Implementation

READ (GET)

Request: GET {{base_url}}/user

Response: 200 OK with user profile information

Verified: Authentication working correctly



Screenshot 3: GET request showing user data

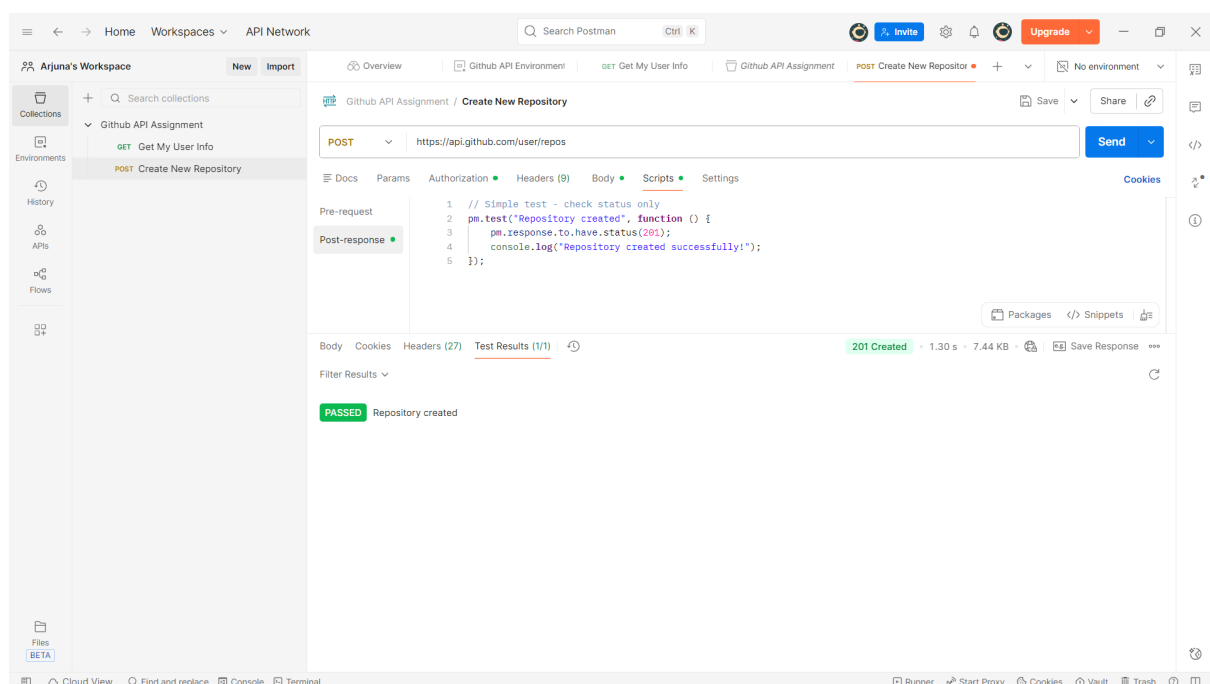
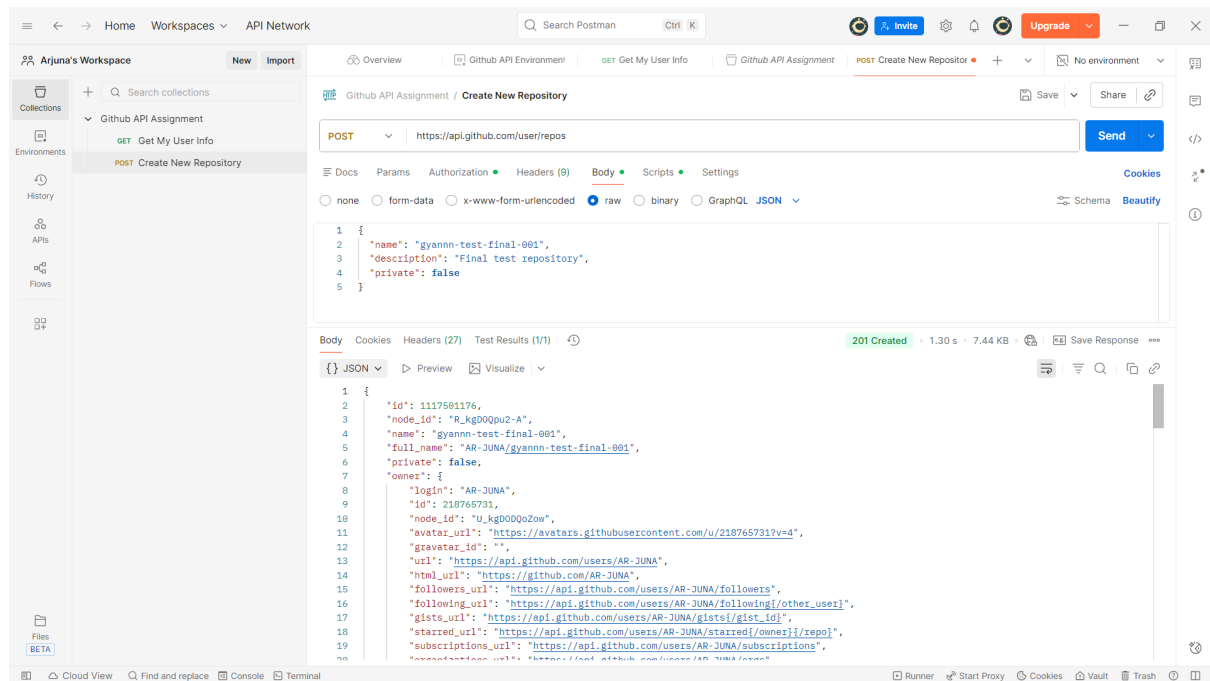
CREATE (POST)

Request: POST `{{base_url}}/user/repos`

Body:

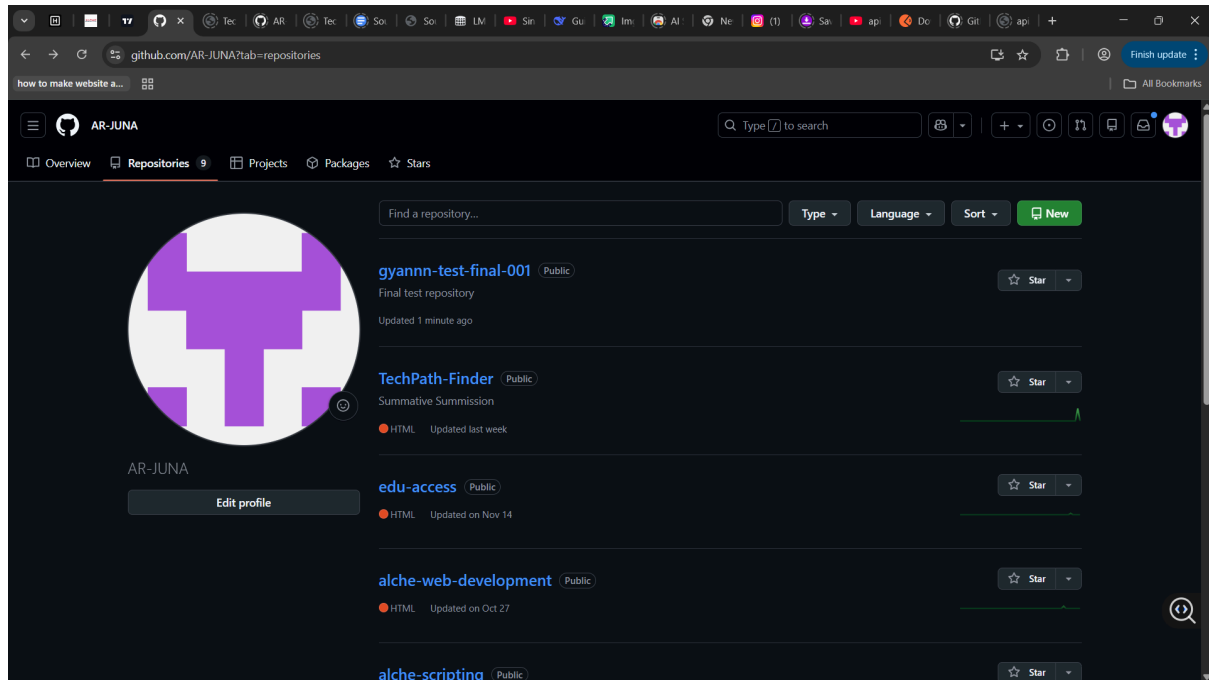
json

```
{
  "name": "test-repo-postman",
  "description": "Repository created via Postman",
  "private": false
}
```



Response: 201 Created with repository details

Screenshot 4: POST request and successful response



Screenshot 5: After the POST request, the repository was created on GitHub

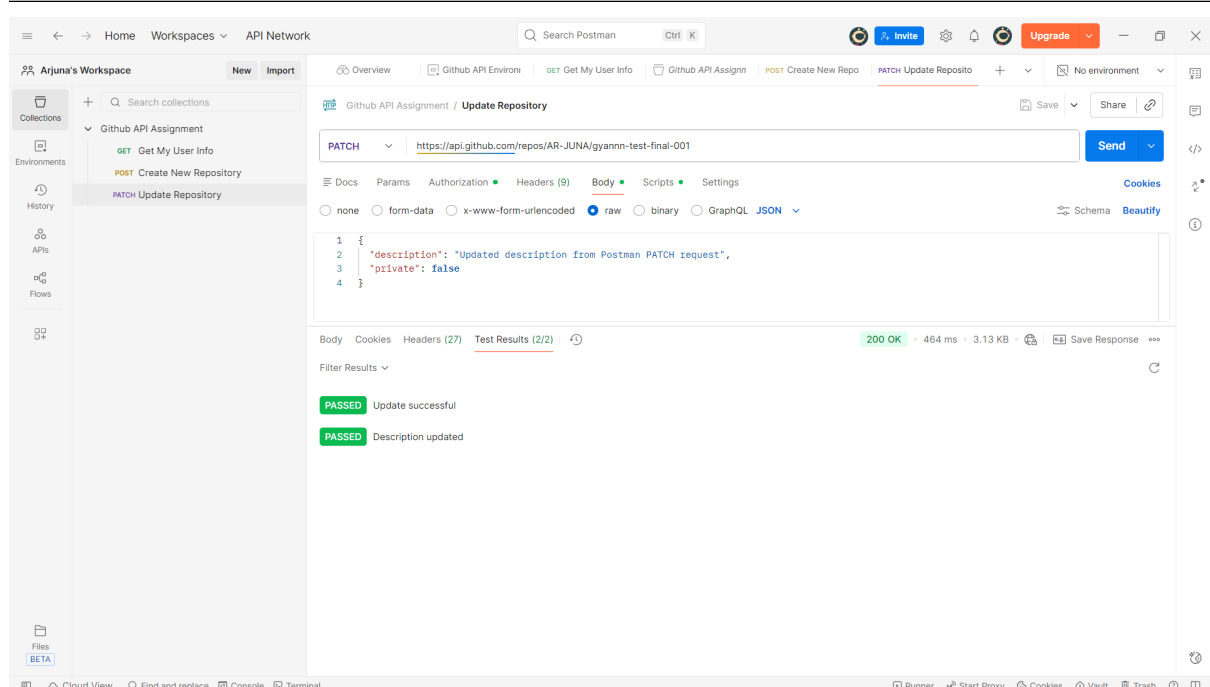
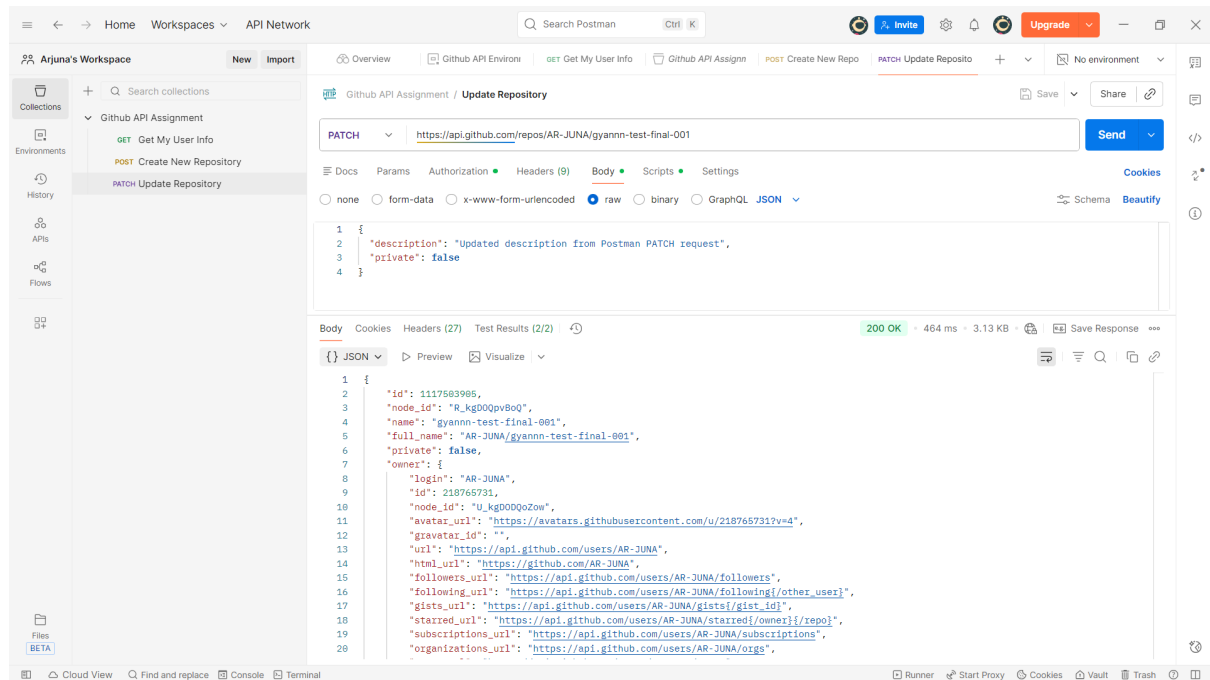
UPDATE (PATCH)

Request: PATCH `{{base_url}}/repos/{username}/test-repo-postman`

Body:

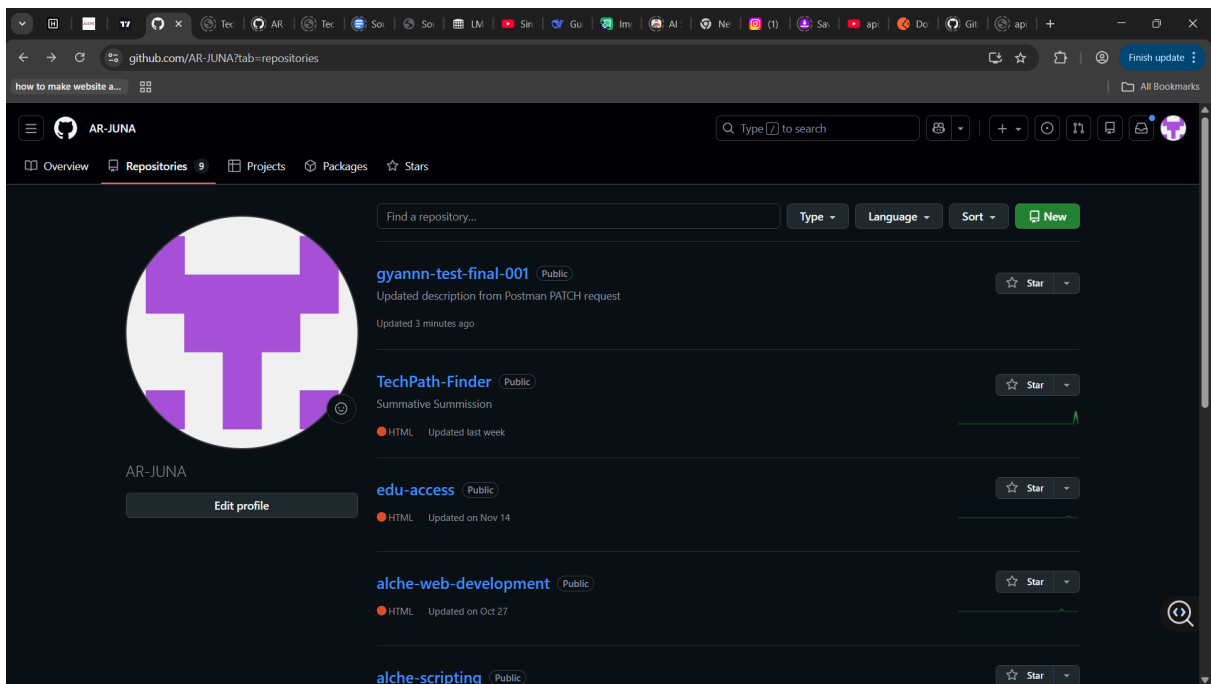
json

```
{  
  "description": "Updated description via PATCH request"  
}
```



Response: 200 OK with updated repository data

Screenshot 6: PATCH request and updated response

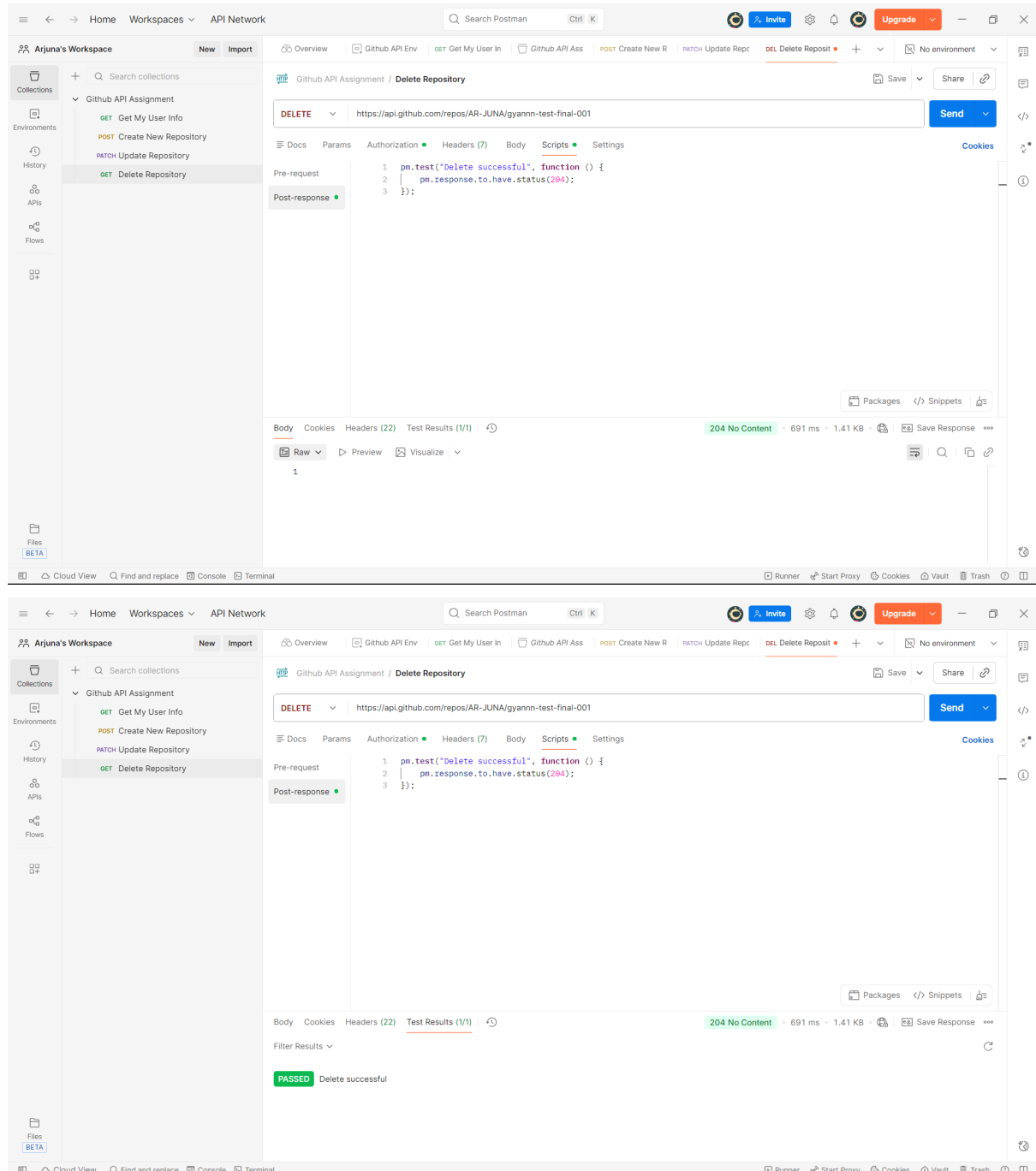


Screenshot 7: After the PATCH request, the README for the repository got updated

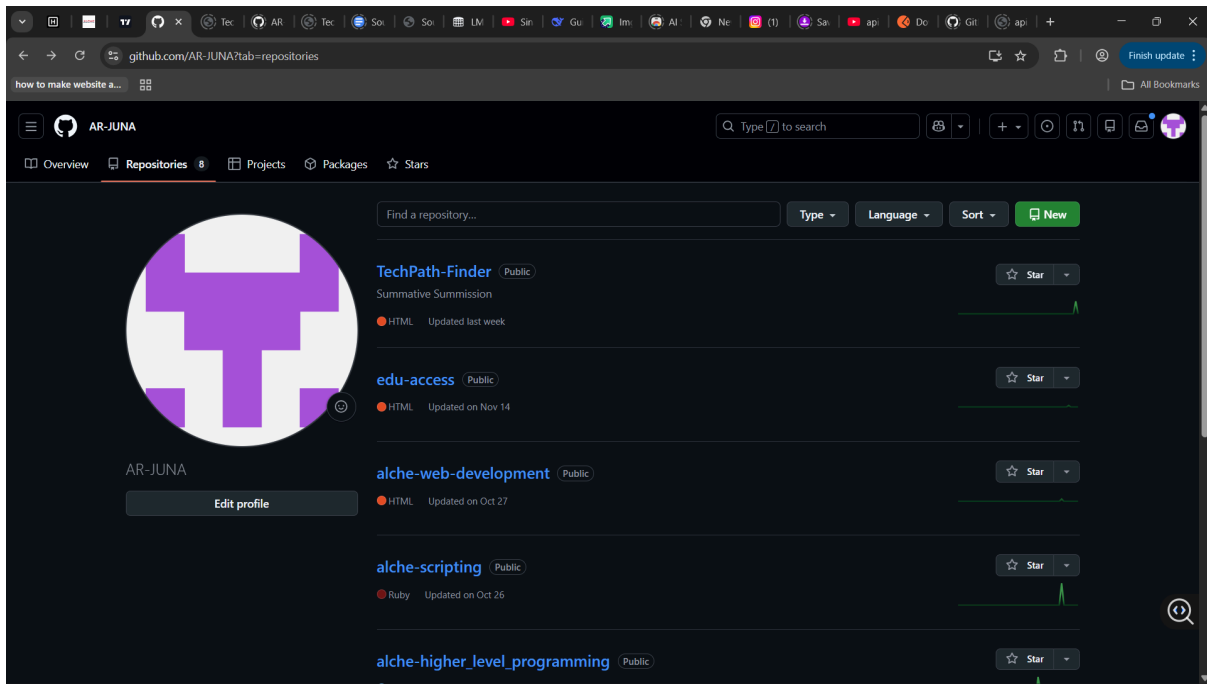
DELETE (DELETE)

Request: DELETE {{base_url}}/repos/{{username}}/test-repo-postman

Response: 204 No Content (successful deletion)



Screenshot 8: DELETE request with 204 response



Screenshot 9: Verified: Repository removed from GitHub account

Postman Environment & Variables

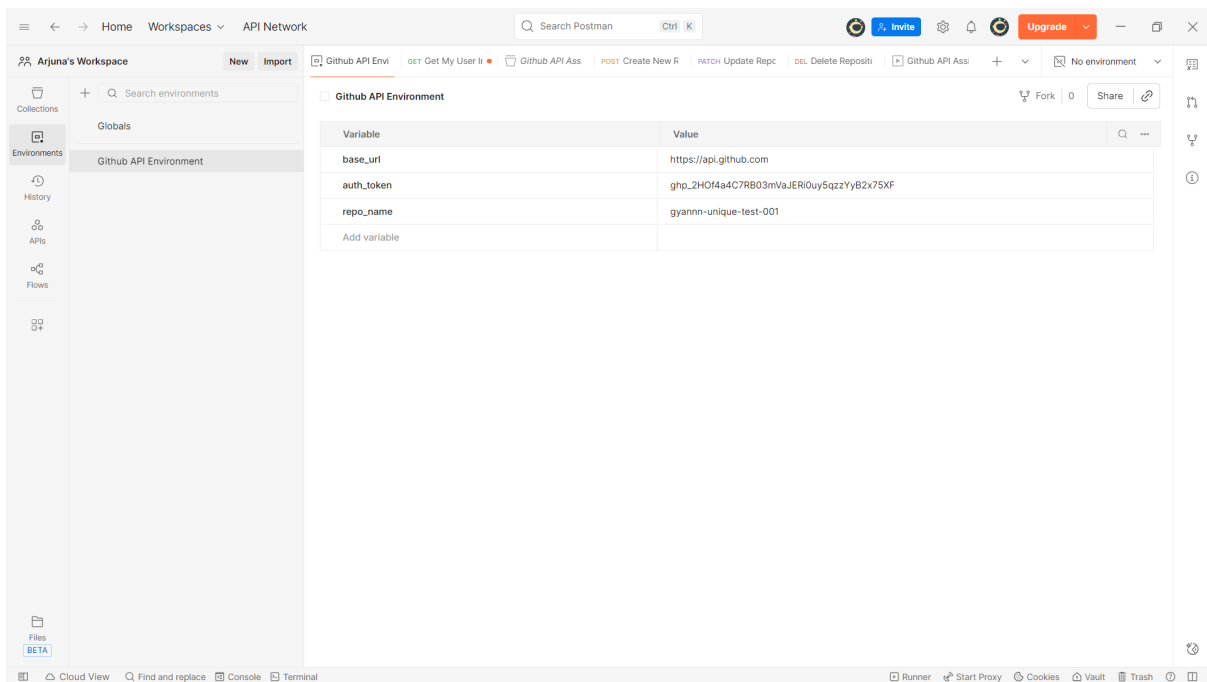
I created a Postman environment called GitHub API Environment with:

Variables Used:

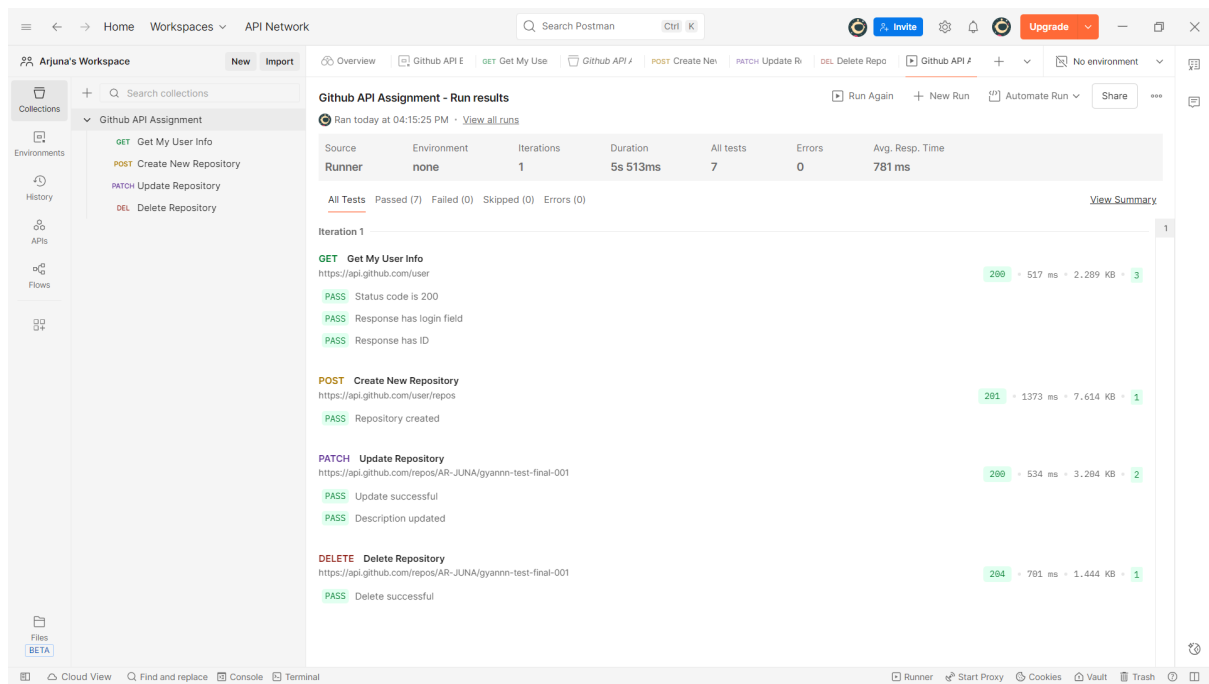
base_url: <https://api.github.com>

auth_token: [My GitHub personal access token]

repo_name: test-repo-postman



Screenshot 10: Environment variables panel showing all variables



Screenshot 11: After running all the requests, all the results passed.

Automation & Testing

Test Scripts

Added validation tests to each request:

javascript

// Example test for GET request

```
pm.test("Status code is 200", function() {
  pm.response.to.have.status(200);
});
```

```
pm.test("Response has user login", function() {
  var jsonData = pm.response.json();
  pm.expect(jsonData.login).to.be.a('string');
});
```

Challenges & Solutions

Challenge 1: 422 Validation Error

Problem: Repository name already existed

Solution: Used unique naming with a timestamp, deleted conflicting repos

Challenge 2: Test Placement

Problem: Tests in Pre-request Script instead of Tests tab

Solution: Moved validation scripts to the correct location

Challenge 3: Token Permissions

Problem: Insufficient scopes initially

Solution: Regenerated token with repo and user scopes