4. Testing Flow

Here's what happens when you open a PR:

Step	Expected Behavior	What to Check
1. PR Created	GitHub sends a webhook payload to backend.	Backend logs: Webhook received: pull_request opened.
2. Backend Fetches Files	Backend calls GitHub API to get diff of changed files.	Backend logs: Fetched 3 files from PR #12.
3. AI Service Triggered	Backend sends code diff to Python AI service.	AI service logs request → returns JSON with score, comments.
4. Results Saved to MongoDB	Backend stores PR metadata + AI results in Atlas.	Check Atlas → reviews collection updated.
5. Feedback Posted to GitHub	Inline comments appear directly on the PR.	GitHub PR shows AI comments.

1. Testing Goals

We need to verify that:

Goal	What It Proves
Webhook → Backend works	PR events are received correctly and files are fetched.
AI Service works	Code diffs are analyzed and meaningful results returned.
MongoDB Atlas storage works	PR metadata and AI review JSON are saved properly.
Multiple files are handled	Backend can process more than one file in a PR.
GitHub API integration works	Inline AI comments are posted back to the PR.
Edge cases are handled	No crashes for empty diffs, large changes, or bad code.

---

2. PR Test Plan

PR #	Type of Test	Description of Change	Expected AI Feedback
PR 1	**Basic PR (Happy Path)**	Add a simple log line in a small file like app.js or index.js.	Should receive a clean review, maybe a small style comment.
PR 2	**Buggy Code**	Introduce bad patterns like: - eval()	AI should flag lint and security issues.

PR #	Type of Test	Description of Change	Expected AI Feedback
		- Unused variables - Console logs left in production code.	
PR 3	**Multiple File Changes**	Edit app.js, routes/exams.js, and models/Exam.js together.	AI should return comments for multiple files in one response.
PR 4	**Performance Edge Case**	Add inefficient code like nested loops or unoptimized DB queries.	AI should recommend optimizations.
PR 5	**Empty / No Code Change**	Open a PR but don't change any code.	System should **not crash** and store metadata correctly.
PR 6	**Large Code Addition**	Add a completely new file with ~50 lines of code.	Tests AI's ability to handle big diffs without timeouts.

---

3. Detailed Examples for Each PR

PR 1 – Basic Test

File: app.js

```
// Add this at the top

console.log("Webhook and AI test run successful!");
```

Expected:

- Backend logs webhook reception.
- MongoDB stores PR metadata + AI response.
- GitHub shows AI comment like:

"Remove console logs before production deployment."

PR 2 – Buggy Code

File: controllers/examController.js

```
function evaluateExam(input) {

  eval(input); // Security risk

  var temp = 5; // Unused variable

  console.log("Debugging output...");

}
```

Expected AI Comments:

- Flag eval() as unsafe.
- Flag unused variable temp.
- Recommend removing console.log().

PR 3 – Multiple Files

Edit three files in one PR:

- app.js
- routes/exams.js
- models/Exam.js

Example changes:

- Add a route in routes/exams.js.
- Update schema in models/Exam.js.
- Modify a middleware in app.js.

Expected:

- MongoDB stores **all three file paths** in one PR entry.
- AI generates comments **grouped by file**.

PR 4 – Performance Issue

File: controllers/examController.js

```
for (let i = 0; i < exams.length; i++) {
  for (let j = 0; j < exams.length; j++) {
    console.log(exams[i], exams[j]);
  }
}
```

Expected AI Feedback:

- Suggest optimizing nested loop.
- Recommend using Map or database-side filtering.

PR 5 – Empty PR

- Create a branch but **do not change any files**.
- Open a PR.

Expected:

- Backend logs webhook received.
- MongoDB saves PR metadata with empty files_changed array.
- No crash on AI or GitHub posting step.

PR 6 – Large Diff

- Create a new file: utils/bulkImport.js.
- Add ~50 lines of code with functions and classes.

Expected:

- Backend and AI handle the large payload.
- MongoDB stores full diff.
- AI might flag general best practices or unused code.

---

4. MongoDB Expected Output

For **PR 3 (multiple files)**, MongoDB document should look like this:

```
{
  "_id": "66d97e123456",
  "pr_number": 3,
  "repo": "Mahfooz/exam-app",
  "branch": "feature-multiple-file-test",
  "files_changed": [
    "app.js",
    "routes/exams.js",
    "models/Exam.js"
  ],
  "ai_feedback": {
    "score": 78,
    "categories": {
      "lint": 90,
      "bugs": 80,
      "security": 85,
      "performance": 60
    },
    "summary": "Schema updated, but new route lacks error handling.",
    "comments": [
      {
        "path": "routes/exams.js",
        "line": 22,
        "body": "Add validation for incoming exam data."
      },
      {
        "path": "models/Exam.js",
        "line": 5,
        "body": "Consider adding indexes for better performance."
      }
    ]
  },
  "created_at": "2025-09-03T18:30:00Z"
}
```

---

5. Suggested Order of PR Creation

Run tests in this sequence:

1. **PR 1 (Basic) →** confirm entire flow works.
2. **PR 2 (Buggy) →** verify AI detection.
3. **PR 3 (Multi-file) →** verify aggregation of files.
4. **PR 5 (Empty) →** confirm no crashes on edge case.
5. **PR 4 (Performance) →** test AI logic depth.
6. **PR 6 (Large) →** stress-test payload handling.