**PR 1 – Basic Test**

**File:** app.js  
**Change:**

js

// Add this at the top

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

**Expected Console Output:**

text

🚀 Server running on port 5000

MongoDB connected successfully

👉 Webhook received: pull\_request

📥 Processing PR #1 from repo: Mahfooz/exam-app

🔹 Fetching PR #1 details for repo: Mahfooz/exam-app

✅ PR metadata fetched: Basic Test

🔹 Diff URL from API: https://github.com/Mahfooz/exam-app/pull/1.diff

✅ Direct diff fetched, length: 120

🤖 Calling AI service for PR #1...

✅ AI analysis completed for PR #1

💾 Saved PR #1 analysis to DB

📤 Posting review to GitHub PR #1...

✅ Mapped comment for app.js line 2 → position 2

✅ Review posted to PR #1 with event: COMMENT

**Expected MongoDB Document:**

json

{

"\_id": "66d97e1a1c2d3e4f5a6b7c8d",

"pr\_number": 1,

"repo": "Mahfooz/exam-app",

"branch": "feature-console-log-test",

"files\_changed": ["app.js"],

"ai\_feedback": {

"score": 90,

"categories": {

"lint": 85,

"bugs": 95,

"security": 95,

"performance": 95

},

"summary": "Minor linting issue found. Code is otherwise clean and secure.",

"comments": [

{

"path": "app.js",

"line": 2,

"body": "Remove console logs before production deployment to avoid leaking sensitive information."

}

],

"fix\_suggestions": []

},

"created\_at": "2025-09-03T10:15:00Z"

}

**Expected GitHub Comment:**

**AI Code Review** [Score: 90/100]  
**Summary:** Minor linting issue found. Code is otherwise clean and secure.

**app.js:2**  
console.log("Webhook and AI test run successful!");  
Remove console logs before production deployment to avoid leaking sensitive information.

**PR 2 – Buggy Code**

**File:** controllers/examController.js  
**Change:**

js

function evaluateExam(input) {

eval(input); // Security risk

var temp = 5; // Unused variable

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

}

**Expected Console Output:**

text

👉 Webhook received: pull\_request

📥 Processing PR #2 from repo: Mahfooz/exam-app

✅ PR metadata fetched: Buggy Code Test

✅ Direct diff fetched, length: 250

🤖 Calling AI service for PR #2...

✅ AI analysis completed for PR #2

💾 Saved PR #2 analysis to DB

📤 Posting review to GitHub PR #2...

✅ Mapped comment for controllers/examController.js line 2 → position 2

✅ Mapped comment for controllers/examController.js line 3 → position 3

✅ Mapped comment for controllers/examController.js line 4 → position 4

✅ Review posted to PR #2 with event: REQUEST\_CHANGES

**Expected MongoDB Document:**

json

{

"\_id": "66d97e2b4c5d6e7f8a9b0c1d",

"pr\_number": 2,

"repo": "Mahfooz/exam-app",

"branch": "feature-buggy-code",

"files\_changed": ["controllers/examController.js"],

"ai\_feedback": {

"score": 40,

"categories": {

"lint": 50,

"bugs": 30,

"security": 20,

"performance": 90

},

"summary": "Critical security vulnerability and multiple code quality issues detected.",

"comments": [

{

"path": "controllers/examController.js",

"line": 2,

"body": "Critical Security Risk: The use of `eval()` is highly dangerous as it can execute arbitrary code, leading to injection attacks. Use a safe alternative like `JSON.parse()` or a validation library."

},

{

"path": "controllers/examController.js",

"line": 3,

"body": "Bug: Unused variable 'temp'. Remove it to clean up the code."

},

{

"path": "controllers/examController.js",

"line": 4,

"body": "Lint: Remove `console.log()` before deploying to production."

}

],

"fix\_suggestions": [

{

"path": "controllers/examController.js",

"patch": "- eval(input); // Security risk\n+ // eval(input); // REMOVED: Security risk\n+ const parsedInput = JSON.parse(input); // Example safe alternative"

}

]

},

"created\_at": "2025-09-03T12:45:00Z"

}

**PR 3 – Multiple Files**

**Changes in:** app.js, routes/exams.js, models/Exam.js

**Expected Console Output:**

text

👉 Webhook received: pull\_request

📥 Processing PR #3 from repo: Mahfooz/exam-app

✅ PR metadata fetched: Multiple Files Test

✅ Direct diff fetched, length: 1500

🤖 Calling AI service for PR #3...

✅ AI analysis completed for PR #3

💾 Saved PR #3 analysis to DB

📤 Posting review to GitHub PR #3...

✅ Mapped comment for routes/exams.js line 22 → position 45

✅ Mapped comment for models/Exam.js line 5 → position 12

✅ Review posted to PR #3 with event: COMMENT

**Expected MongoDB Document:**  
*(This matches the example you provided exactly)*

json

{

"\_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."

}

],

"fix\_suggestions": []

},

"created\_at": "2025-09-03T18:30:00Z"

}

**PR 4 – Performance Issue**

**File:** controllers/examController.js  
**Change:**

js

for (let i = 0; i < exams.length; i++) {

for (let j = 0; j < exams.length; j++) {

console.log(exams[i], exams[j]);

}

}

**Expected MongoDB Document:**

json

{

"\_id": "66d97e3d4e5f6a7b8c9d0e1f",

"pr\_number": 4,

"repo": "Mahfooz/exam-app",

"branch": "feature-nested-loop",

"files\_changed": ["controllers/examController.js"],

"ai\_feedback": {

"score": 65,

"categories": {

"lint": 70,

"bugs": 60,

"security": 90,

"performance": 40

},

"summary": "Critical performance issue identified with an O(n²) nested loop.",

"comments": [

{

"path": "controllers/examController.js",

"line": 2,

"body": "Performance: Inefficient O(n²) nested loop detected. For large datasets, this will cause significant slowdown. Consider using a `Map` for O(1) lookups or filtering on the database side."

},

{

"path": "controllers/examController.js",

"line": 3,

"body": "Lint: Remove `console.log()` in performance-critical code."

}

],

"fix\_suggestions": [

{

"path": "controllers/examController.js",

"patch": "-for (let i = 0; i < exams.length; i++) {\n- for (let j = 0; j < exams.length; j++) {\n- console.log(exams[i], exams[j]);\n- }\n-}\n+// Example optimization: Using a Map for comparisons\n+const examMap = new Map(exams.map(exam => [exam.id, exam]));\n+for (const exam of exams) {\n+ // Perform operations using examMap.get(someId)\n+}"

}

]

},

"created\_at": "2025-09-03T14:20:00Z"

}

**PR 5 – Empty PR**

**Expected Console Output:**

text

👉 Webhook received: pull\_request

📥 Processing PR #5 from repo: Mahfooz/exam-app

✅ PR metadata fetched: Empty PR Test

🔹 Fetching PR #5 files...

✅ PR files fetched: []

⚠️ No files changed in PR #5. Skipping AI analysis and GitHub review.

💾 Saved PR #5 metadata to DB. (ai\_feedback: null)

**Expected MongoDB Document:**

json

{

"\_id": "66d97e4f5a6b7c8d9e0f1a2b",

"pr\_number": 5,

"repo": "Mahfooz/exam-app",

"branch": "empty-test-branch",

"files\_changed": [],

"ai\_feedback": null,

"created\_at": "2025-09-03T16:00:00Z"

}

**PR 6 – Large Diff**

**File:** utils/bulkImport.js (New file, ~50 lines)

**Expected Console Output:**

text

👉 Webhook received: pull\_request

📥 Processing PR #6 from repo: Mahfooz/exam-app

✅ PR metadata fetched: Large Diff Test

✅ Direct diff fetched, length: 3850

🤖 Calling AI service for PR #6... (This may take a moment)

✅ AI analysis completed for PR #6

💾 Saved PR #6 analysis to DB

📤 Posting review to GitHub PR #6...

✅ Mapped 8 comments to their respective positions.

✅ Review posted to PR #6 with event: COMMENT

**Expected MongoDB Document:**

json

{

"\_id": "66d97e5a6b7c8d9e0f1a2b3c",

"pr\_number": 6,

"repo": "Mahfooz/exam-app",

"branch": "feature-bulk-import",

"files\_changed": ["utils/bulkImport.js"],

"ai\_feedback": {

"score": 72,

"categories": {

"lint": 75,

"bugs": 70,

"security": 80,

"performance": 65

},

"summary": "New utility added with generally good structure. Several linting and best practice suggestions below.",

"comments": [

{

"path": "utils/bulkImport.js",

"line": 10,

"body": "Consider using async/await for better readability over promise chaining."

},

{

"path": "utils/bulkImport.js",

"line": 25,

"body": "This function is defined but never used within the module. Consider removing it if it's not needed."

},

{

"path": "utils/bulkImport.js",

"line": 35,

"body": "Avoid hardcoding the API URL. Use an environment variable instead."

}

// ... more comments

],

"fix\_suggestions": []

},

"created\_at": "2025-09-03T20:10:00Z"

}

Of course. Here are **10 additional test cases** to ensure your AI-powered MERN Code Reviewer is robust, covering edge cases, security, and real-world scenarios:

**1. PR 7 – Dependency Change Only**

**Change:** Only package.json or package-lock.json is modified.

* **File:** package.json
* **Change:** Adds a new dependency: "axios": "^1.6.0"

**Expected Behavior:**

* AI should recognize it's a dependency file.
* **Expected AI Comment:** "Consider reviewing the security and maintenance status of the new 'axios' dependency. Avoid using wildcard (^) in production for better version locking."
* Should NOT crash or throw errors.

**2. PR 8 – Secret Leak**

**Change:** Accidentally hardcodes a secret.

* **File:** config/database.js
* **Change:**

js

const password = "mySecretPassword123"; // Hardcoded secret

**Expected Behavior:**

* Security Agent must flag this critically.
* **Expected AI Comment:** "CRITICAL: Hardcoded secret detected. Immediately revoke this password and use environment variables or a secret management service."
* GitHub review should be posted with event: REQUEST\_CHANGES.

**3. PR 9 – SQL Injection Vulnerability**

**Change:** Uses user input directly in a SQL query.

* **File:** controllers/userController.js
* **Change:**

js

const query = `SELECT \* FROM users WHERE name = '${req.body.name}'`; // SQL injection risk

**Expected Behavior:**

* Security Agent should detect the pattern.
* **Expected AI Comment:** "SQL Injection vulnerability: Do not interpolate user input directly into SQL queries. Use parameterized queries or an ORM."

**4. PR 10 – Large Binary File**

**Change:** Attempts to add a large non-text file (e.g., image, PDF).

* **File:** assets/large-image.png (new file, 5MB)

**Expected Behavior:**

* Backend should skip sending binary files to the AI service.
* **Console Log:** ⚠️ Skipping binary file: assets/large-image.png
* AI review should only process text-based files.
* No crash should occur.

**5. PR 11 – Syntax Error**

**Change:** Introduces a JavaScript syntax error.

* **File:** utils/helpers.js
* **Change:**

js

function brokenFunction( {

return "missing parenthesis";

}

**Expected Behavior:**

* The AI service might fail to parse the code.
* **Expected Console Log:** ❌ AI service returned an error: Could not parse code in utils/helpers.js.
* System should handle the error gracefully, perhaps posting a generic comment: "Syntax error detected in this file. Please check the code for typos or missing symbols."

**6. PR 12 – Rename/Move File**

**Change:** Renames or moves a file without changing its content.

* **Action:** Rename server.js to index.js

**Expected Behavior:**

* GitHub API returns this as a file change.
* AI should recognize it as a rename and not analyze the content for bugs (since it didn't change).
* **Expected AI Comment:** "File renamed. Consider updating any references to the old file name."

**7. PR 13 – Merge Conflict Markers**

**Change:** Accidentally commits merge conflict markers.

* **File:** app.js
* **Change:**

js

function someFunction() {

<<<<<<< HEAD

console.log("New change");

=======

console.log("Old change");

>>>>>>> feature-branch

}

**Expected Behavior:**

* Lint Agent should flag this immediately.
* **Expected AI Comment:** "Merge conflict markers detected. Resolve the conflicts before merging."

**8. PR 14 – AI Service Down**

**Scenario:** The Python AI service is offline or returns a 500 error.

**Expected Behavior:**

* Node.js backend should handle the HTTP error gracefully.
* **Console Log:** ❌ AI service unavailable. Could not review PR #14.
* Should NOT crash the webhook handler.
* Should still save PR metadata to DB with ai\_feedback: null.

**9. PR 15 – Cross-Site Scripting (XSS)**

**Change:** Outputs user input directly into HTML (common in SSR or frontend).

* **File:** views/profile.hbs (Handlebars template)
* **Change:**

hbs

<h1>Welcome, {{unescapedUserName}}!</h1> // XSS risk if unescaped

**Expected Behavior:**

* Security Agent should detect unescaped output in a template.
* **Expected AI Comment:** "Potential XSS vulnerability: Avoid using unescaped output with user data. Use {{{escapedUserName}}} or the appropriate safe output method for your templating language."

**10. PR 16 – Rate Limit Test**

**Scenario:** Rapid-fire PRs are opened (e.g., 5 PRs in 2 minutes).

**Expected Behavior:**

* System should handle GitHub API rate limits.
* **Console Log for later PRs:** ⚠️ GitHub API rate limit approaching. Retrying after delay...
* Implementation should include retry logic with exponential backoff for GitHub API calls.
* All PRs should eventually be processed without data loss.

**Summary Table of Additional Test Cases**

| PR # | Test Case | Key Challenge | Expected AI Action |
| --- | --- | --- | --- |
| 7 | Dependency Change | Recognize dependency files | Comment on versioning & security |
| 8 | Secret Leak | Detect hardcoded secrets | CRITICAL security flag |
| 9 | SQL Injection | Detect string interpolation in SQL | Flag vulnerability & suggest parameterized queries |
| 10 | Large Binary File | Skip non-text files | Skip analysis, log warning |
| 11 | Syntax Error | Handle unparseable code | Graceful error handling |
| 12 | Rename/Move File | Detect file operations | Comment on rename, skip content analysis |
| 13 | Merge Conflicts | Detect conflict markers | Flag as linting error |
| 14 | AI Service Down | Handle external service failure | Graceful degradation, save metadata |
| 15 | XSS Vulnerability | Detect unescaped output in templates | Flag security risk |
| 16 | Rate Limiting | Handle GitHub API rate limits Implement retry logic with backoff |  |

These test cases will help you ensure your system is **production-ready** and handles real-world scenarios effectively. 🧪✅