AI-Powered MERN Code Reviewer — Project Specification

1. Project Overview

You will build a full-stack application that automatically reviews GitHub Pull Requests (PRs) in MERN projects using AI agents.
The system will:

1. Receive PR events via GitHub Webhooks.

2. Fetch changed files using the GitHub API.

3. Send code to a Python AI service (LangChain + LangGraph) for multi-agent analysis.

4. Generate review feedback with lint, bug, security, and performance insights.

5. Post AI feedback directly back to the GitHub PR.

6. Display review history in a React dashboard.

---

2. Core Requirements

Backend (Node.js + Express)

JWT authentication (jsonwebtoken) for dashboard login.

Password hashing (bcrypt or bcryptjs).

MongoDB (Mongoose) for storing:

Users

PR metadata

AI review results

Webhook endpoint:

POST /webhooks/github — receives pull_request events.

Verifies HMAC SHA-256 signature using crypto.

GitHub API Integration:

Fetch PR file changes: GET /repos/{owner}/{repo}/pulls/{pull_number}/files

Post review comments: POST /repos/{owner}/{repo}/pulls/{pull_number}/reviews

Node → Python Communication:

Send PR diffs via axios POST to Python AI service.

Receive structured JSON review results.

AI Service (Python + FastAPI)

FastAPI app with /analyze endpoint.

LangChain + LangGraph orchestration:

Agent 1: Lint & Style Checker — code formatting, best practices.

Agent 2: Bug Detector — logical/functional issues.

Agent 3: Security Scanner — secrets, injections, risky patterns.

Agent 4: Performance Reviewer — optimization suggestions.

Coordinator Agent — merges all outputs into final structured JSON.

Output format:

```
{
  "score": 85,
  "categories": {
    "lint": 90,
    "bugs": 80,
    "security": 85,
    "performance": 80
  },
  "summary": "Overall good PR, but needs variable naming improvements.",
  "comments": [
    {
      "path": "src/App.jsx",
      "line": 42,
      "body": "Consider memoizing this function to avoid re-renders."
    }
  ],
  "fix_suggestions": [
    {
      "path": "src/utils/helper.js",
      "patch": "diff/udiff..."
    }
  ]
}
```

Frontend (React.js)

Login & Registration:

Form submission → JWT auth.

Store token in localStorage or httpOnly cookie.

Dashboard:

List all reviews for the logged-in user.

Show PR title, repo name, date, overall score.

Review Detail Page:

Show AI feedback by category.

Display inline comments and fix suggestions.

Score breakdown chart (optional: Chart.js or Recharts).

Protected Routes:

Redirect to login if no JWT token.

---

3. Technology Stack

Backend (Node.js):

express, mongoose, jsonwebtoken, bcrypt, axios, express-validator, dotenv, crypto

AI Service (Python):

fastapi, uvicorn, langchain, langgraph, openai (or other LLM API), pydantic

Frontend (React.js):

react, react-router-dom, axios, material-ui or tailwindcss, optional: chart.js or recharts

Database:

MongoDB (local, Docker, or MongoDB Atlas)

---

4. Key Features

1. Secure JWT-based authentication for dashboard.

2. Webhook-based PR review automation.

3. Multi-agent AI analysis for different code quality dimensions.

4. Direct GitHub PR commenting via API.

5. Frontend dashboard for viewing review history and details.

---

5. Implementation Steps (Recommended Order)

1. Setup Backend (Node.js)

Express server, MongoDB connection, JWT auth, bcrypt hashing.

2. GitHub Webhook Integration

Secure webhook endpoint, signature verification, PR file fetching.

3. Setup AI Service (Python)

FastAPI, LangChain, LangGraph agents for lint, bug, security, performance.

4. Service Communication

Node sends PR files → Python → returns structured JSON.

5. Post AI Review to GitHub

Use API to create review and inline comments.

6. Frontend Development

Login, dashboard, review detail, API integration.

7. Testing & Deployment

Test with real GitHub repo PRs, deploy both services, record demo.

---

6. Deliverables

Public GitHub repo with:

Backend (Node.js) code

AI service (Python) code

React frontend code

.env.example with required environment variables.

Postman collection or curl commands for API testing.

README with:

Setup instructions

Architecture diagram

API endpoints

Webhook configuration guide

---

7. Exclusions

No admin panel.

No team/project management UI.

No analytics dashboard beyond basic score display.

No repository indexing beyond PR scope.

---

8. Suggested Timeline

Moderate MERN skill: 3 weeks

Full-time focus: 7–10 days

Phases:

1. Backend auth + MongoDB

2. GitHub integration

3. AI service build

4. Node ↔ Python connection

5. Frontend dashboard

6. Testing & polish