📝 QuickForms AI – Code Review Summary

Course: CSC 318 – Software Engineering Principles
Assignment: Final Project – QuickForms AI
Deliverable: *Code Review Summary*
Project Title: QuickForms AI – Smart Form & Workflow Builder

Student: Eric Amoh Adjei
Instructor: Professor Rawad Habib
Date: August 2025

---

1. Introduction

As part of the Final Project, a structured code review was conducted on the QuickForms AI codebase. The review focused on readability, maintainability, adherence to object-oriented design principles, and compliance with software engineering best practices. The goal was to ensure that the system is not only functional but also scalable, testable, and easy for future developers to extend.

---

2. Review Process

Tooling & Methods:

- **Manual review** via GitHub pull requests.
- **Automated linting & formatting**: ESLint and Prettier.
- **Testing coverage checks**: Vitest, Jest, and Supertest.
- **Checklist-based evaluation** of code style, modularity, and documentation.

Scope of Review:

- **Frontend** – React + Vite + TypeScript (components, hooks, state management).
- **Backend** – Express + TypeScript (controllers, services, error handling).
- **Database** – Prisma ORM with SQLite/Postgres (schema design and migrations).

---

3. Strengths Identified

- ✅ **Clear Project Structure** – Organized into `frontend/`, `backend/`, `docs/`, `uml/`.
- ✅ **Consistent Naming** – camelCase for functions/variables; PascalCase for React components.
- ✅ **Layered MVC Architecture** – Proper separation of concerns between models, controllers, and views.
- ✅ **Reusable Components** – Modular React components and form generation functions.
- ✅ **Documentation** – README files, UML diagrams, and inline comments support onboarding.
- ✅ **Version Control Discipline** – Regular commits with descriptive messages improved traceability.

---

4. Issues Identified & Fixes

ID	Area	Issue	Recommendation
C1	Backend API	Generic error messages	Provide detailed, user-friendly erro
C2	Frontend Form Builder	Missing input validation	Added validation hooks for empty f
C3	Database Schema	The initial schema lacked normalization	Updated with `forms` ↔ `responses`
C4	Code Style	Inconsistent indentation	Applied Prettier auto-format
C5	Unit Tests	Limited edge case coverage	Added tests for invalid email & em
C6	API Security	No rate limiting on endpoints	Added basic middleware for throttl

---

5. Lessons Learned

- **Early linting prevents technical debt** – applying Prettier and ESLint from the start would have reduced style inconsistencies.
- **Error handling improves UX** – detailed error messages reduce user frustration and debugging time.
- **Modularization is key** – splitting logic into small, reusable components makes scaling and debugging easier.
- **Peer review catches what tools miss** – manual reviews flagged issues that automated checks overlooked.
- **Edge case testing matters** – validating unusual inputs (e.g., empty responses) improved system robustness.

6. Conclusion

The QuickForms AI codebase demonstrates **strong software engineering discipline**:

- ✅ Follows object-oriented design principles.
- ✅ Organized in a layered MVC architecture.
- ✅ Code is linted, formatted, and maintainable.
- ✅ Functional and non-functional requirements are met.
- ✅ Ready for future extension, scaling, and deployment.

Overall finding: QuickForms AI is a clean, maintainable MVP that reflects industry-standard practices.