📋 QuickForms AI

Smart Form & Workflow Builder

Course: CSC 318 – Software Engineering Principles
Assignment: Final Project – QuickForms AI
Deliverable: *Readme.md*
Project Title: QuickForms AI – Smart Form & Workflow Builder

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

![TypeScript](https://img.shields.io/badge/TypeScript-5.x-blue)
![React](https://img.shields.io/badge/React-18.x-61DAFB)
![Express](https://img.shields.io/badge/Express-4.x-black)
![License: MIT](https://img.shields.io/badge/License-MIT-green.svg)

TypeScript
React
Express
License: MIT

📌 Project Overview

QuickForms AI is a lightweight, AI-powered form generator designed for **small businesses, clubs, events, student teams, and individuals**.

The system provides an **AI-assisted way** to create and manage forms, integrate responses with tools like Google Sheets, and streamline workflows without requiring programming knowledge.

- Build forms in minutes
- Share via a simple link
- Collect and store responses
- Export results to CSV for reporting

🎥 Demo & Screenshots

- YouTube Demo: *(Insert Link)*
- Example Screenshots:
  - **Form Builder UI**
  - **Dashboard with responses**
  - **CSV export view**

🚀 Features

- **Form Builder:** Add text, numbers, checkboxes, dropdowns, radio buttons, dates
- **AI Assistance:** Generate form fields from natural language prompts
- **Form Sharing:** Public shareable links for distribution
- **Response Collection:** Secure storage via SQLite (default) or Postgres
- **Data Integration:** Export submissions to CSV / Google Sheets
- **Responsive Frontend:** React + Vite + TypeScript
- **Backend API:** Express + Prisma ORM
- **Testing:** Unit, integration, and system tests

🏗️ Tech Stack

Layer	Technology	Purpose
Frontend	React + Vite + TypeScript	Dynamic, responsive UI
Backend	Express.js + TypeScript	API endpoints & AI integration
Database	Prisma ORM + SQLite/Postgres	Form & response storage
Testing	Vitest + Supertest	Unit, integration, system tests
Build Tools	pnpm, PowerShell automation	Dependency management & builds

---

📁 Project Structure

```
quickforms-ai/
│
├── apps/
│   ├── web/          # React (Vite) frontend
│   └── api/          # Express backend (TypeScript + Prisma)
│
├── docs/             # Reports, UML diagrams, user & dev manuals
│   ├── diagrams/
│   ├── reports/
│   └── user-manual/
│
├── prisma/           # Database schema & migrations
├── tests/            # Unit & integration tests
│
├── README.md         # This file
├── package.json
└── .gitignore
```

⚡ Getting Started

1. Clone the Repository

```
git clone https://github.com/YOUR-USERNAME/quickforms-ai.git
cd quickforms-ai
```

2. Install Dependencies

Frontend:

```
cd apps/web
npm install
```

Backend:

```
cd ../api
npm install
```

3. Configure Environment

Create `.env` inside `apps/api/`:

```
DATABASE_URL="file:./dev.db"
PORT=4000
OPENAI_API_KEY="your_api_key_here"
```

4. Run Locally

Frontend → http://localhost:5173

```
cd apps/web
npm run dev
```

Backend → http://localhost:4000

```
cd apps/api
npm run dev
```

🧪 Testing

Run automated tests:

```
npm run test
```

Test Coverage Includes:

- ✅ Unit tests (functions & components)
- ✅ Integration tests (API endpoints)
- ✅ System tests (form creation → submission → CSV export)
- ✅ Manual peer code review

📖 Documentation

- 📘 **Final Report** → `/docs/final-report/report.pdf`
- 🧑‍💻 **Developer Guide** → `/docs/developer-guide/developer.pdf`
- 🧑 **User Manual** → `/docs/user-manual/manual.pdf`
- 🧪 **Testing Report** → `/docs/reports/testing.pdf`
- 📝 **Code Review Summary** → `/docs/reports/code-review.pdf`
- 📊 **UML Diagrams** → `/docs/diagrams/uml.pdf`
- 📑 **SRS (Requirements Spec)** → `/docs/reports/SRS.pdf`

📦 Packaging

To build a final submission bundle:

```
.\all-in-finals-build.ps1
```

This creates `quickforms-ai-final.zip` containing all deliverables.

🔮 Roadmap

- Add **user authentication & roles**
- Introduce **analytics dashboard**
- Expand **AI-driven smart form generation**
- Deploy to **Vercel (frontend)** + **Render/Heroku (backend)** with Postgres

---

👥 Development Process

- Agile methodology with **2-week sprints**
- Unified Process (UP) for requirement → design → implementation → test cycle
- Git-based collaboration with feature branches & pull requests
- Peer review before merging

🎓 Course Context

This project was developed for **CSC 318 – Software Engineering Principles (UAT)**.
It demonstrates **end-to-end SDLC coverage**: requirements engineering, UML modeling, system design, Agile implementation, testing, documentation, and presentation.

🏆 Lessons Learned

- Applied **Agile & Unified Process** effectively
- Designed & implemented **layered MVC architecture**
- Practiced **object-oriented design principles**
- Conducted **unit, integration, and system testing**
- Strengthened **team workflow using Git & GitHub**

📜 License