2025-08-20 14:55 (America/Chicago)

Scope: Gamified Quiz Generator Tool

Project Type: Next.js web application
~~Deployment: Netlify (with background functions for AI calls)~~
Design Requirement (non-negotiable): Match Varsity Tutors production tools UI/UX

Project Overview

Build a gamified, interactive quiz generator usable by students, teachers, and parents. Users can create quizzes via (1) manual inputs or (2) lesson-plan upload. Quizzes must be multiple-choice, playable in the browser with scoring and a completion screen, and exportable to PDF. Implementation details for AI prompt/response and internal data structures are flexible; only the UI/UX alignment with production tools is fixed. The provided example AI prompts are examples only. They are not intended to be used necessarily, only as a potential guide

We will need your gmail to provide access to the shared folder and your git in order to give access to git repository

---

Required Features

Generation Options

1. Manual Input

- Subject (dropdown, required)
- Grade (dropdown, optional)
- Number of problems (numeric, required)
- Area of focus (text, required)
- Difficulty (easy / moderate / challenging, required)

2. Lesson Plan Upload

- File upload: PDF, PNG, JPG/JPEG (required)
- Number of problems (numeric, required)
- Difficulty (easy / moderate / challenging, required)

Quiz Build and Play

- Generate: on submit, show a “Building your quiz…” screen; simulate ~5 seconds for the prototype.
- ~~Netlify detection: use Netlify Background Functions for long AI calls in production; use standard async flow locally.~~
- Render interactive multiple-choice items with four options, a single correct answer, and per-question point values (default 1).
- Progress indicator (e.g., “Question 3 of 10”).
- Scoring: compute total points and percentage.
- Completion screen with score summary and brief feedback message.

Export

- Download PDF containing questions, options, points, and a separate answer key section.

Technical Requirements

- Framework: Next.js
- ~~Hosting: Netlify~~
    - ~~Use Background Functions for long-running AI calls.~~

    - ~~Detect environment (Netlify vs local) and branch logic accordingly.~~

- AI: OpenAI (model, prompt, and response handling are developer's choice).
- Styling: Align with Varsity Tutors production tools.
- PDF Export: Any reliable approach (e.g., jsPDF, pdfkit, react-pdf).

Deliverables

1. ~~Netlify deployment URL demonstrating full functionality.~~
2. GitHub repository with clear commit history and documentation.
3. README with setup instructions for local dev ~~and Netlify deployment~~, including environment configuration.

Evaluation Criteria

- Visual and interaction parity with Varsity Tutors production tools.
- Both generation paths function (manual inputs and lesson-plan upload).
- Interactive play experience with progress and scoring.
- Correct, well-formatted PDF export with answer key.
- Maintainable, documented code and sensible project structure.

~~Security and Operations Notes~~

- ~~Do not hardcode or commit API keys. Use environment variables (e.g., `OPENAI_API_KEY`) and a `.env.local` ignored by git.~~

- ~~The key shared in chat should be rotated immediately and replaced with a secure environment variable before any public commits or deployments.~~
- ~~For Netlify, set environment variables in site settings; for local, use `.env.local`.~~

Setup Outline (suggested)

- Next.js app with pages or app router.
- UI shell and components aligned to production tools (buttons, forms, cards, loaders).
- API route(s):
    - Local: standard async call to OpenAI.
    - ~~Production: Netlify Background Function wrapper that forwards to OpenAI and returns a normalized payload.~~
- Quiz state machine: idle → generating → active → completed.
- PDF export utility that renders content and answer key.

Resources, Links, and Examples

Project Assets and Repos

- Primary repository: https://github.com/informedecommerce/vt_quiz_generator
- Drive assets folder: https://drive.google.com/drive/u/0/folders/1-WSK1LkGCJPeQ2mrvvq18dEFK0mO4Uls

Design and Production References

- Varsity Tutors production tools (design baseline): http://ai.varsitytutors.com/tools/

- Practice Problem Generator (structure reference): https://ai.varsitytutors.com/tools/practice-problem-generator

- Client demo example (logic reference only): https://quiz-generator-tool-3njz.bolt.host/

Platform and APIs

- ~~Netlify Background Functions: https://docs.netlify.com/build/functions/background-functions/~~

- Next.js documentation: https://nextjs.org/docs

- OpenAI Responses API (reference): https://platform.openai.com/docs/guides/responses

- File uploads in Next.js (API routes / middleware): https://nextjs.org/docs/app/building-your-application/routing/route-handlers

PDF and Utilities (examples)

- jsPDF: https://github.com/parallax/jsPDF

- react-pdf: https://react-pdf.org/

- pdfkit: https://pdfkit.org/

Accessibility and UI Consistency

- WAI-ARIA Authoring Practices: https://www.w3.org/WAI/ARIA/apg/

- Color contrast checker: https://webaim.org/resources/contrastchecker/

Notes on Keys

- Replace any inline key with `process.env.OPENAI_API_KEY`.

- Do not include secrets in commits, issues, or READMEs.

Time Tracking and Reporting

We require accurate, lightweight time tracking for this project.

Tools

- Recommended: Toggl Track — https://track.toggl.com/

Expectations

- Track all project time from first setup to final delivery.