MCQDoctor: Multiple Choice Questions Validator

Documentation & User Guide

Release Date: December 2025

Platform: Universal (HTML5/JavaScript) - Client-Side Only

1. Introduction

MCQDoctor is an AI-powered clinical audit tool designed for academicians and medical educators. It automates the validation of Multiple Choice Questions (MCQs) against international best practices (such as NBME and Haladyna guidelines).

The application diagnoses issues in question stems, distractors, and answer keys, providing a detailed report with actionable suggestions and scientifically improved versions of the questions.

2. User Manual

2.1. Getting Started

When you first launch the application, you will be greeted by a **Configuration Screen**.

1. **API Key Requirement:** The app requires a Google Gemini API Key to function.
2. **Obtaining a Key:**
   - Click the link provided in the app: aistudio.google.com/app/apikey.
   - Log in with your PERSONAL Google account. Sometimes corporate Gmail accounts do not offer a free tokens tier
   - Click "**Create API Key**".
3. **Setup:** Copy the key string (starts with `AIzaSy...`) and paste it into the input field. Click "**Initialize App**".
   - Note: The key is saved in your browser's temporary session storage. You will need to re-enter it if you close the browser tab.

2.2. Uploading Questions

- For best results and to respect the free API tokens tier, better use a maimum of 10 MCQs per run.
- There are two ways to input data into MCQDoctor:

Option A: File Upload (Recommended)

You can upload existing question files. The app uses AI to extract questions from raw documents automatically.

- **Supported Formats:** `.pdf`, `.docx` (Word), `.txt`, `.rtf`, `.json`.

- **How to use:** Click the "**Import File**" button and select your document. The extracted questions will appear in the text area as JSON code.

Option B: Manual Input / Paste

If you have questions in a raw text format or pre-formatted JSON, you can paste them directly into the large text area.

Format: The app expects a JSON array:

```
<>JSON
[
  {
    "question": "Patient presents with...",
    "options": ["Diagnosis A", "Diagnosis B"],
    "correctAnswer": "Diagnosis A"
  }
]
```

Option C: Load Sample

Click "**Load Sample Case**" to populate the area with dummy data to test the application's capabilities.

2.3. Running the Diagnosis

Once the file is loaded or text is entered:

1. Click the "**Diagnose Questions**" button (Stethoscope icon).
2. The app will show a "Analyzing Vital Signs" loading screen.
3. Depending on the number of questions, analysis takes 5–15 seconds.

2.4. Interpreting the Report

The results screen is divided into three main sections:

1. **Executive Summary:** A high-level overview of the quality of the entire batch of questions.

2. **Question Item Cards:** Each question is displayed as a card.
   - Visual Indicators:
     - ✓ **Green (Pass):** No issues found.
     - ✓ **Amber (Warning):** Minor issues (e.g., negative phrasing, slight ambiguity).
     - ✓ **Red (Fail):** Critical flaws (e.g., cueing, multiple correct answers).

3. **Detailed Breakdown (Click to Expand):** Click on any question card to see:
    - **Diagnostic Report:** Specific comments on the Stem, Distractors, and Key.
    - **Optimized Version:** An AI-rewritten version of the question that fixes the flaws while keeping the clinical context.
    - **Rationale:** An explanation of why the correct answer is correct.

2.5. Exporting Results

- Click the "**Download PDF Report**" button at the top right of the results section.
- The app will generate a professional, A4-formatted PDF containing the summary and detailed analysis for every question, suitable for printing or emailing to faculty.

3. Technical Documentation (For Developers)

3.1. Architecture

The application is a **Client-Side Single Page Application (SPA)** built with **React 18**. It relies on modern browser capabilities and does not require a dedicated backend server for logic, as it communicates directly with the Google Gemini API.

3.2. Tech Stack

- Core: React 18, TypeScript (TSX).
- Styling: Tailwind CSS (Utility-first CSS framework).
- Icons: Lucide React.
- AI Integration: `@google/genai` (Google Gemini SDK).
- File Processing:
    - `mammoth.js`: For extracting text from `.docx` files.
    - `html2pdf.js`: For generating PDF reports from the DOM.
- Build/Runtime: Can be run as a standalone HTML file (via Babel standalone) or compiled via Vite/Webpack.
- **Data Persistence:** None. Refreshing the page clears all data (Security Feature).
- **Concept & Logic:** Dr. Muhammad AlShorbagy, Dean, College of Pharmacy, GMU.
- **Technical Implementation:** AI-Assisted Development (Code generation).

  Methodology: "This single-file HTML application demonstrates a 'No-Code/Low-Code' development approach. The domain expertise, algorithm logic, and user experience design were provided by Dr. Muhammad AlShorbagy, while the source code was generated via prompt engineering using Large Language Models (LLMs)."

3.3. API Model Configuration

The application is currently configured to use:

- **Model:** `gemini-2.5-flash` (Optimized for speed and efficiency).
- **Parameters:** JSON Mode enabled (`responseMimeType: "application/json"`) with a strict schema to ensure consistent data parsing.

3.4. Security Considerations

- **API Key:** The API key is stored in `sessionStorage`. It is never sent to any server other than Google's API endpoints. It is cleared when the browser tab is closed.
- **Data Privacy:** Question data is processed in browser memory and sent only to the AI model for analysis. No data is stored in a persistent database by the app itself.

4. Troubleshooting & FAQ

Q: I get a "Failed to call Gemini API" error.

- **Check:** Is your API key valid?
- **Check:** Does your API key have the "Generative Language API" enabled in the Google Cloud Console?
- **Check:** Are you on a restricted network (hospital/university VPN) that might block Google API calls?

Q: The PDF report is blank or cut off.

- **Fix:** Ensure you wait for the "Generating PDF..." overlay to disappear. Do not switch tabs while the PDF is generating. The app uses a virtual overlay to render the PDF; ensuring the browser window is active helps this process.

Q: My .doc file isn't uploading correctly.

- **Fix:** .doc is an older binary format. The app supports it via basic text reading, but formatting is often lost. Please save your file as .docx or .pdf for best results.

Q: The AI didn't detect the correct answer in my uploaded file.

- **Fix**: If the file doesn't explicitly mark the answer (e.g., with an asterisk * or bold text), the AI attempts to infer it. Always review the "Input JSON" before clicking Diagnose to ensure the data looks correct.