Math Trainer (AI): Project Workflow

1. Ideation & Motivation

- I noticed that traditional math practice apps are often repetitive and disengaging.
- To make it fun, I thought of combining **math problems with hobbies (sports)**.
- The idea: generate math questions as a combination of:
  Math Topic × Hobby (sport) × Student Grade × Difficulty Level
- Bonus: Let students choose how many questions they want (up to 5) or practice in "endless" single-question mode.

---

2. Defining Requirements

I broke the project into **functional** and **technical** requirements:

Functional Requirements:

- Student selects: *Math Topic, Hobby, Grade, Difficulty*.
- App generates unique questions (MCQs).
- Provide explanations + auto-evaluate MCQs.
- Track progress across sessions.

Technical Requirements:

- Pure frontend (HTML, CSS, Vanilla JS).
- Optional **LLM integration** to make questions more natural.
- Use **Cloudflare Workers** as a bridge to hide API keys and handle requests.
- Deploy as a **static site on GitHub Pages**.
- Store session data in **localStorage** (no external DB).

---

3. Initial Design

- Drew a simple **flow diagram**:
  User Input → Question Generator → Render Question → Answer Validation → Feedback → Next Question.

- Split the logic into **modules**:
    - **UI Controller** (rendering, buttons, feedback).
    - **Question Engine** (random numbers, sports templates).
    - **MCQ Generator** (create distractors, shuffle).
    - **Validator & Feedback** (check answers, highlight correct).
    - **Storage** (localStorage for settings/progress).
    - **API Client** (calls Cloudflare Worker for Gemini).

---

4. Prompting the LLMs for Code

- At first, I used **Groq's LLaMA-3.3 model** to generate some base question logic, but later switched to **Google Gemini 1.5** for richer, more human-like question text.
- My prompts were structured, e.g.:

  Generate 5 unique math problems for Grade 3, difficulty = easy, topic = addition, hobby = cricket. Format the output as JSON with {question, correctAnswer, explanation, options []}.
- I refined prompts iteratively:
    - Added constraints like "ensure MCQ options are unique."
    - Specified answer formats so my JS parser could handle responses easily.
    - Asked LLM to produce **code snippets** in Vanilla JS for rendering/validation.
- Used **ChatGPT + Gemini** in parallel: one for **ideation**, the other for **code scaffolding**.

---

5. Implementation (Frontend)

- **HTML/CSS**:
    - Built a clean, responsive UI.
    - Added keyboard shortcuts (1–4 to pick MCQs).
    - Included accessibility (focus indicators, high-contrast colors).
- **JavaScript**:
    - Random number generator for deterministic fallback.
    - Sports-based templates (Virat scored 12 runs...).
    - Auto-evaluation for MCQs with instant feedback.

- Progress tracking stored in localStorage (e.g., streaks, last settings).

---

6. Cloudflare Worker Setup

- Created a Worker as a secure **middleware**:
  - Accepts POST requests from frontend (/api/next).
  - Adds API key & forwards request to **Gemini API**.
  - Handles **CORS** and **preflight checks**.
  - Returns clean JSON to the frontend.
- This way, **API keys never touch the browser**.

---

7. Testing & Debugging

- Unit tests for question generation (edge cases, MCQ uniqueness).
- Manual QA on:
  - Mobile responsiveness.
  - Error handling (rate limits, network issues).
  - Friendly error messages instead of raw codes.
- Found and fixed 3 major security issues in the Worker (API key exposure, open CORS policy, unvalidated inputs).

---

8. Deployment

- Hosted the **frontend on GitHub Pages** → public link for demos.
- Configured **Cloudflare Worker environment variables** (API_KEY, MODEL).
- Verified CORS and rate limits were working correctly.

---

9. Tools I Used

- **Frontend Dev:** VS Code, GitHub, Sublime Text (for quick edits).
- **Serverless:** Cloudflare Workers (proxy).
- **Version Control:** Git/GitHub.
- **LLMs:** Google Gemini 1.5 (main), ChatGPT (for scaffolding).

- **Debugging:** Browser DevTools (network tab, console logs).

---

10. Outcome

- Delivered a **fun, AI-powered Math Trainer** that:
    - Generates endless unique math questions.
    - Supports **LLM-based and deterministic generation**.
    - Securely integrates Gemini via Cloudflare Workers.
    - Is live on GitHub Pages: Math Trainer App