AI-Powered Virtual Science Labs: Project Plan

# 🎯 Goal

To build a system where an AI can generate realistic, interactive science lab simulations from user prompts. The system will:  
- Accept a topic (e.g., “Ohm’s Law”, “Acid-Base Titration”)  
- Use an LLM to generate the full experiment description and simulation code  
- Render the simulation in-browser using TypeScript + Canvas or a rendering library  
- Help students perform virtual labs, even without physical equipment

# 🧰 Current Assets (What’s Already Done)

- index.ts: Likely the main entry point for simulations  
- ml.ts: Module for machine learning/data labs  
- science.ts: Core science lab logic (chemistry/physics)

# ✅ The Plan (Phase by Phase)

## 🧩 Phase 1: MVP Architecture

Inputs:  
- A topic or lab goal  
- Optional grade level or lab type  
  
Process:  
1. Send the input to an LLM (e.g., GPT-4)  
2. LLM responds with:  
 - Lab metadata (title, objectives, materials, steps)  
 - TypeScript/JS code for a simple simulation  
3. Render the simulation in-browser using a <canvas> or SVG renderer  
  
Outputs:  
- A working virtual lab with:  
 - Visual apparatus  
 - User interaction (sliders, buttons)  
 - Measurable outputs (e.g., graphs, values)

## 🛠️ Phase 2: Simulation Framework

Build a framework that allows dynamically injecting and running the generated code.  
  
Tech Stack:  
- Frontend: React + TailwindCSS  
- Editor: Monaco Editor  
- Renderer: HTML5 Canvas / p5.js / D3.js / Three.js  
- Sandboxing: Run generated code in a secure iframe or Web Worker

## 🧠 Phase 3: Prompt & Code Template

Prompt Template Example:  
You are a virtual lab assistant. Based on the topic "{{topic}}", generate:  
1. A clear experiment title  
2. Objective and procedure  
3. TypeScript code to simulate the lab in-browser using a canvas  
  
Expected LLM Output Format:  
export const simulation = {  
 title: "Ohm’s Law",  
 objective: "Demonstrate the relationship between voltage and current",  
 code: `function runSimulation(canvas) { ... }`  
}

## 🧪 Phase 4: Validation & Security Layer

Before running any generated code:  
- Parse and scan for disallowed functions (e.g., eval, fetch, document)  
- Run it in an isolated context (iframe with sandbox or lightweight JS interpreter)  
- Catch and handle errors before user sees them

## 🖼️ Phase 5: Visual Lab Interactions

Add standard visual elements:  
- Apparatus images/icons  
- Sliders for variables (e.g., voltage, time, mass)  
- Real-time graph plotting (e.g., using Chart.js or D3)

# 🧑‍💻 What Help Is Needed

- ⚙️ TypeScript Dev: Build secure simulation renderer  
- 🎨 Frontend Dev: Build UI to render lab metadata + code + simulation  
- 🧠 Prompt Engineer: Optimize prompts and templates for lab generation  
- 🔐 Security Engineer: Set up sandbox/code-safety evaluation  
- 🧪 Science Expert: Help refine realistic lab procedures and accuracy

# 🧭 Long-Term Vision

- Support Physics, Chemistry, Biology, and Machine Learning labs  
- Curriculum-aware generation (e.g., GCSE, CBSE, AP)  
- Teacher-facing tools to edit and customize labs  
- Interactive grading and assessments  
- Mobile and tablet support for low-tech environments

# 🌍 Why This Matters

Science learning is inaccessible for many due to lack of labs. This project builds an AI-powered gateway to practical lab education — affordable, customizable, and scalable across regions and languages.