1. **Educational Apps:**Build something that helps others learn. Examples: Interactive tutorials, onboarding tools, or AI-enhanced learning platforms, etc.

Submissions to the Hackathon must meet the following requirements:

* Include a Project built with the required developer tools and meets the above Project Requirements.
* Include a text description that should explain the features and functionality of your Project.
* Include a demonstration video of your Project. The video portion of the Submission should describe for the judging panel how Kiro was used to create the project. Examples can include:
  + For building and vibe coding from scratch: How did you structure your conversations with Kiro to build your project? What was the most impressive code generation Kiro helped you with?
  + For agent hooks: What specific workflows did you automate with Kiro hooks? How did these hooks improve your development process?
  + For spec-to-code: How did you structure your spec for Kiro to implement? How did the spec-driven approach improve your development process?
  + should be less than three (3) minutes. Judges are not required to watch beyond three minutes
  + should include footage that shows the Project functioning on the device for which it was built
  + must be uploaded to and made publicly visible on YouTube, Vimeo, Facebook Video, or Youku, and a link to the video must be provided on the submission form on the Hackathon Website; and
  + must not include third party trademarks, or copyrighted music or other material unless the Entrant has permission to use such material.
* Provide a URL to your open source code repository for judging and testing.
  + The code repository must be public with an approved OSI Open Source License
* Identify which category you are submitting to.
* Provide a write up on how Kiro was used.

**1. Key Features of AI Math Tutor**

1. **Step-by-Step Problem Solver**
   * User submits a math problem.
   * Tutor parses and explains each step, not just the answer.
2. **Interactive Quizzes & Feedback**
   * Auto-generate quizzes, explain incorrect answers.
3. **Visualizations**
   * Plot vectors, matrices, system solutions, derivatives, integrals.
4. **Personalized Learning Path**
   * Track student progress, adapt topics/difficulty.
5. **AI-Powered Hints & Explanations**
   * Offer hints or “why” explanations using LLM or custom rules.
6. **Voice/Handwriting Input** *(optional advanced)*
   * Allow input via voice or handwriting (use OCR/Speech-to-Text APIs).
7. **Math for AI/ML**
   * Special modules: Eigenvalues/vectors, gradients, loss surfaces, optimization, etc.

**2. Technology Stack**

* **Python:** SymPy, NumPy, Matplotlib, scikit-learn, PyTorch/TensorFlow for deeper AI/ML math, OpenAI API for LLM-based explanations.
* **Go:** Backend API for scalability, real-time quiz/leaderboard, or session management.
* **Frontend:** React (web), Flutter (mobile), or simple HTML/JS for MVP.
* **Extras:** FastAPI (Python) for REST API, gRPC (Go/Python) for fast service communication.

**3. Architecture Sketch**

csharp

CopyEdit

[Frontend (Web/Mobile)]

|

REST/gRPC

|

[Go API Layer]

|

[Python Logic/AI Engine]

| |

SymPy/NumPy LLM API (OpenAI)

| |

Visuals (Matplotlib)

**4. Implementation Steps**

**A. Math Problem Parsing & Solving**

* **Linear Algebra/System of Equations:**
  + Use [SymPy](https://www.sympy.org/) in Python for symbolic math.
  + Example:

python

CopyEdit

from sympy import symbols, Eq, solve

x, y = symbols('x y')

eq1 = Eq(2\*x + y, 1)

eq2 = Eq(x - y, 3)

result = solve((eq1, eq2), (x, y))

# Output: {x: 4/3, y: -5/3}

* + For step-by-step, use sympy.solver or explain by breaking into row operations.
* **Calculus (Derivatives, Integrals):**

python

CopyEdit

from sympy import diff, integrate, sin

x = symbols('x')

derivative = diff(sin(x)\*x, x)

integral = integrate(sin(x)\*x, x)

* + Visualize with Matplotlib.
* **Math for AI/ML:**
  + Eigenvalues/vectors: numpy.linalg.eig, sympy.Matrix.eigenvals()
  + Gradients: sympy.diff, autograd via PyTorch for ML.

**B. Step-by-Step Explanation**

* Either:
  + Use SymPy's step APIs (where available).
  + OR integrate with **OpenAI GPT** (text-davinci-003, GPT-4, etc.) to explain the steps:
    - “Explain how to solve Ax = b for x, where A=..., b=...”
    - “Show step-by-step how to differentiate x^2 sin(x).”
* Present each step as a chat/message/card.

**C. Interactive Quizzes**

* Pre-seeded problem bank (JSON, CSV, or DB).
* Random generation with parameter ranges.
* Validate student answers using the Python solver.
* Give contextual hints (from LLM or predefined).

**D. Visualization**

* Use Matplotlib (Python) to plot:
  + Vector addition, dot/cross product (Linear Algebra).
  + Solution lines/planes for systems.
  + Graphs of functions, tangents, gradients, loss curves.

**E. Learning Path & Personalization**

* User login/session management (Go or Python).
* Store progress, quiz history.
* Use analytics to suggest next topics.
* Optionally, a recommender engine for learning path.

**F. Backend/Frontend Integration**

* **Go**: Efficient API layer, can handle many concurrent users, schedule tasks.
* **Python**: Heavy lifting for math and AI.
* **Frontend**: UI for entering problems, displaying steps, showing plots, quizzes, progress dashboard.

**5. Sample User Story Flow**

1. **User:** Enters “Solve for x: 2x + 3 = 7”
2. **App:** Parses, solves using SymPy, breaks into steps, sends each step with a short explanation.
3. **User:** Asks “What does this step mean?”
4. **App:** Uses OpenAI API to generate a natural language explanation.
5. **User:** Takes a quiz on similar problems.
6. **App:** Visualizes results, tracks streak, and adapts next quiz for user’s weaknesses.

**6. MVP Launch Checklist**

* **Frontend**: Simple web UI (React or even Streamlit for rapid prototyping)
* **Backend**: Python API (FastAPI/Flask), or Go API (with Python subprocess for math)
* **Core Features**:
  + Problem input, step-by-step output
  + Visualization
  + Quizzes
* **AI Explanation Integration** (optional): LLM API for explanations
* **User Progress Tracking**
* **Deployment:** Dockerize both services, deploy on cloud/free tier (Render, Heroku, AWS, etc.)

**Bonus: Open-Source References**

* SymPy Gamma — Symbolic calculator with steps.
* [Stepik Adaptive Learning](https://github.com/StepicOrg/adaptive-python) — Adaptive quizzes.
* [Manim](https://www.manim.community/) — Beautiful math animations (Python, advanced).
* [Mathigon](https://mathigon.org/) — For UI/UX inspiration.