EduScale Engine: Code the Future of Systemic Change

Guidelines for Innovation & Responsibility

Welcome, Innovators!

Welcome to the EduScale Engine Hackathon! We are thrilled to see what you will build.

The Goal: Build an intelligent data infrastructure that transforms messy, diverse educational data into actionable insights - automatically adapting to any data format without manual configuration.

The Challenge: As new regions join the network, each brings different data formats, quality levels, and baseline conditions. Your mission is to build a system that automatically adapts to any data structure, enables fair cross-regional performance comparisons, and measures the impact of educational interventions over time - all while keeping the solution simple, secure, and deployable.

This is a real challenge facing Eduzměna Foundation as they scale from pilot programs to nationwide impact. Your solution could serve 5,000+ Czech schools.

These guidelines ensure your innovation is both brilliant AND deployable.

Let's build the future, responsibly.

1.Understanding the Challenge: What You're Building

THE CORE PROBLEM:

Our organization collects diverse educational data from multiple sources with NO standardized structure. Each source has different formats (alphanumeric, voice recording of interviews, diaries, inspection reports, etc.), field names, and quality issues.

As Eduzměna scales nationwide, new regions join continuously - each bringing their own data structures and baseline conditions. We need to track performance across regions AND measure the impact of interventions over time.

Your Mission: Build a strategic intelligence platform for scaling educational impact that can:

- Accept ANY data format (CSV, Excel, JSON, audio records) from any new region
- Automatically understand what the data represents (AI-powered)
- Clean and validate data (detect errors, inconsistencies)
- Track performance metrics across multiple regions comparatively
- Measure the impact of interventions and activities over time
- Make data queryable and useful for strategic decision-making

YOU WILL NOT RECEIVE REAL DATASETS.

We'll provide sample data type specifications and common error scenarios.

YOUR SOLUTION MUST BE DATA-AGNOSTIC AND SCALE-READY.

Think: "universal data processor with regional benchmarking," not "specific dataset analyzer." Also, bear in mind that reports must work continuously even though data structure may be changing over time.

Example Success Scenario:

INITIAL STATE:

Region A (pilot) uploads: "Student ID, Test Score, Date, Intervention Type"

→ System learns baseline performance for Region A

NEW REGION JOINS:

Region B uploads: "StudentID, ExamResult, TestDate, Activity" (different format)

→ System recognizes similar concepts despite different naming

→ Automatically creates comparable metrics

PERFORMANCE TRACKING:

Your system should enable queries like:

✓ "Which interventions had the highest impact in Region A? Apply to Region B?"

✓ "Show trend: How did Region A perform 6 months after joining vs today?"

✓ "Network-wide: What are the intervention types with the highest impact?"

WHAT THIS REQUIRES:

1. ADAPTIVE DATA INGESTION

- New region uploads data → System interprets structure automatically
- Maps to common concepts: "test scores," "interventions," "dates"
- Flags quality issues specific to that region's data

2. TEMPORAL TRACKING

- Captures baseline when region joins
- Tracks changes after interventions/activities
- Enables before/after comparisons

3. CROSS-REGIONAL COMPARISON

- Normalizes metrics across different regional data formats
- Identifies: "Region B improved 15% faster than Region A did at the same stage"
- Handles different regional contexts (urban vs rural, different starting points)

4. INTERVENTION IMPACT MEASUREMENT

- Links activities/interventions to outcome changes
- Shows: "After teacher training program, Region A math scores +12% in 3 months"
- Helps answer: "Should we replicate this intervention in Region C?"

5. NETWORK-WIDE INTELLIGENCE

- Aggregate performance across all regions
- Identifies best practices from high-performing regions
- Detects system-wide trends vs region-specific anomalies

CRITICAL FEATURES YOUR SOLUTION SHOULD INCLUDE:

- Proposal of a suitable Data Store type to collect data inputs (ideally allowing inputs from both PC and mobile devices)
- Regional Dashboard: overall clear outputs - dashboards, filtering, reports, data exports.
- Comparison View: Side-by-side regional performance
- Timeline View: Track changes over time (monthly/quarterly/yearly)
- Intervention Tracking: Tag data with activities, measure impact
- Benchmarking: "How does Region X compare to the network average?"

- Predictive Insights: "Based on Region A's trajectory, what can Region B expect?"

EXAMPLE QUERIES YOUR SYSTEM SHOULD HANDLE:

- "What's the average time for a new region to reach network baseline performance?"
- "Compare Region C's current performance to Region A's performance at 6 months?"
- "Which interventions correlate with the fastest improvement across all regions?"
- "Show network-wide trend: Are we closing the quality gap over time?"

THE SCALABILITY CHALLENGE:

TODAY: 1 pilot region with 3 years of data
FUTURE: 10+ regions joining over the next 5 years

Your system must handle:

- Different regional data maturity (Region A: 3 years of data, Region X: 2 weeks of data)
- Varying data quality across regions
- Different data types, including audio recordings
- New intervention types are being introduced
- Evolving metrics as the Foundations learns what matters most
- Data analytics across different layers to be used by different teams for various purposes

WHAT WE DON'T NEED:

❌ Solutions requiring manual configuration for each new region
❌ Systems that can't compare regions with different data structures
❌ Tools that lose historical context when data formats change
❌ Dashboards that show only the current state without temporal tracking

FOCUS ON: Building the "intelligence layer" that:

1. Makes messy, diverse data usable automatically
2. Enables fair cross-regional performance comparisons
3. Tracks the impact of interventions over time
4. Scales as the network grows from 5 to 500 regions

1.1. Core Principle: Data Security & Confidentiality

This is our most important rule. You will be working with a dataset containing anonymized information. Protecting this data is your top priority. **The Golden Rule:** The provided dataset and any derivatives of it **must not leave the designated hackathon environment.**

1.2. Using External AI and "AI as a Service" Tools

We encourage you to use the vast ecosystem of AI tools available today (e.g., public ChatGPT, GitHub Copilot, Claude, Midjourney, etc.), but with one critical restriction: **You must NEVER send, paste, or upload any provided sensitive data to any external, public, or third-party service.**

Doing so is a direct violation of our data security policy and **will result in immediate disqualification** of your team.

How to use external AI tools safely:

The key is **abstraction and anonymization**. You can rephrase your problem or use the tools for general-purpose tasks.

BAD EXAMPLE (Forbidden):

Pasting into ChatGPT: *"Here is a list of 50 job descriptions from our company. Cluster them by seniority and required cloud certifications."*

GOOD EXAMPLE (Allowed):

Asking ChatGPT: *"Write a Python script using scikit-learn that takes a list of text documents and clusters them using TF-IDF and K-Means. The script should output the top 5 keywords for each cluster."*

You can use external tools to generate code, brainstorm approaches, debug algorithms, or create presentation assets, but

GOLDEN RULE: Never send real educational data to public AI services.

FOR PRODUCTION SOLUTIONS - YOU MUST USE:

Option 1: Self-Hosted Models (Recommended - Cheapest & Safest)

- Ollama with Llama 3 / Mistral (runs on your laptop)
- sentence-transformers for embeddings
- Data never leaves your infrastructure

Option 2: EU Cloud AI (Requires legal agreements)

- Google Vertex AI (EU region)
- Azure OpenAI (EU region)
- Must document which region and data protection agreement is needed

Option 3: No External AI

- Use traditional algorithms (regex, fuzzy matching, statistics)
- Fast, free, zero privacy risk

RED FLAGS - Will Not Be Accepted:

❌ "Uses OpenAI API" without specifying EU deployment
❌ Architecture diagram showing data going to US services
❌ Missing data location documentation
❌ Cloud services without region specifications

REQUIRED: Your README must state:

Data Privacy Statement:
- Where does data get processed? (Local server / EU cloud / specific region)
- Which AI services were used? (Self-hosted Llama 3 / None / Vertex AI EU)

- Does data leave the EU? (No / Only aggregated statistics)
- Monthly cost estimate: €X

AUTOMATIC DISQUALIFICATION IF:
- You send real educational data to public AI APIs
- You cannot document where the data is processed
- Architecture shows data leaving the EU

That's it. Keep it simple, keep it safe.

2. Technology & Architecture Guidelines

Our goal is to create solutions that could one day be integrated into our organization's ecosystem. Following these technical guidelines will increase the long-term viability of your project.

2.1. Cloud Infrastructure & Deployment Philosophy

Our Goal: Solutions that can go live within 48 hours with minimal IT overhead.

Since our organization uses Google Workspace services and Podio, we strongly encourage solutions that leverage this existing infrastructure. Minimal implementation effort into Production is our top priority non-functional requirement for the solution.

However, we recognize that not all use cases fit the above platforms.

RECOMMENDED STACK (Privacy-Safe):

Best Choice - Local Processing:

- Google Cloud Run (europe-west1 region)
- Azure Container Apps (westeurope region)

Good Choice - EU Serverless:

- Docker container with self-hosted models
- Runs on any EU server

Acceptable - Platform Services:

- Explicitly set EU regions in all configs
- Document required data protection agreements

⚠ **WILL NOT BE CONSIDERED FOR PRODUCTION:**

- Solutions requiring manual server provisioning
- Applications needing 24/7 DevOps monitoring
- Architectures with >5 separate services
- Solutions with monthly costs >€200

2.2. Design Philosophy: Smart – Simple – Privacy Safe

Build intelligent systems that don't compromise privacy.

THE SIMPLICITY SCORECARD:

Ask yourself these questions. The more "yes" answers, the better:

Architecture:

- ☐ Could a junior developer understand this in 30 minutes?
- ☐ Can it run on a safe EU cloud or a single server/container?
- ☐ Does it have fewer than 3 external dependencies?

Data Flow:

- ☐ Can you draw the data flow in 5 boxes or fewer?
- ☐ Does data pass through fewer than 3 transformation steps?
- ☐ Could you explain the logic to Eduzměna's non-technical staff?

Maintenance:

- ☐ If the original developer disappears, can someone else maintain it?
- ☐ Does it use standard tools (Python/pandas vs. custom frameworks)?
- ☐ Are there fewer than 100 lines of "clever" code?

Production Operations:

- ☐ Does it have built-in error logging?
- ☐ Can it restart automatically if it crashes?
- ☐ Does it fail gracefully with clear error messages?

ANTI-PATTERNS TO AVOID:

❌ Microservices architecture (this is a 24-hour hackathon, not Netflix)
❌Training custom ML models from scratch when self-hosted models work (use pre-trained Llama 3, Mistral, sentence-transformers)
❌ Building your own database when Google Sheets/SQLite suffices
❌ Real-time streaming when batch processing is adequate
❌ GraphQL APIs when REST or CSV export is enough

ENCOURAGED PATTERNS:

✅ Monolithic applications (easier to deploy and debug)
✅ Serverless functions (no server management)
✅ Managed services (let cloud providers handle complexity)
✅ Static site generators (for dashboards/reports)
✅ Scheduled jobs (cron/Cloud Scheduler) vs. always-on services

AI INTEGRATION PATTERNS (Privacy-Safe):

For this challenge, use AI to:

✅ **Schema Understanding**

- LLM interprets column names: "What does 'Pocet_zaku' likely mean?"
- Generates field type suggestions automatically
- Decide carefully where AI/LLMs are to be used within the process. Better think of a good process flow with the right AI intervention touch points than too complex AI coverage.

✅ **Data Quality Detection**

- Identifies patterns: "90% of IDs are 4-digit numeric, but 10% contain letters"
- Suggests corrections: "'O' should be '0' based on context"

✅ **Entity Matching**

- Uses embeddings to find: "Skola" = "Škola" = "School name"
- Cross-references data from different sources

✅ **Natural Language Queries**

- Users ask: "Which schools improved in math?"
- AI translates to appropriate data operations

2.3. Open Source Software (OSS) Licensing: Keep it Simple

We love Open Source! However, for a project to be viable, it must be built with appropriately licensed software.

THE SIMPLE RULE:

If you're unsure about a license, stick to these proven safe choices:

- Python: MIT-licensed libraries (pandas, flask, streamlit)
- JavaScript: MIT/Apache (React, Vue, Express)
- Databases: PostgreSQL, SQLite (permissive licenses)

RED FLAGS - ASK AN ORGANIZER BEFORE USING:

- Any library with "GPL" or "AGPL" in the license

- Commercial software requiring paid licenses in production
- Libraries with "non-commercial use only" restrictions

QUICK LICENSE CHECK:

Visit: https://choosealicense.com/appendix/
Or run: pip show <package-name> (look for "License:" field)

When in doubt, ask! We'd rather help you find an alternative than disqualify a great solution for a licensing issue.

2.4. Version Control & Code Hygiene

You are encouraged to use Git and GitHub to collaborate. However, you must follow secure coding practices to protect our data and your credentials.

NEVER Commit Credentials: Never hardcode API keys, passwords, or other secrets directly in your source code. Once committed, they can be exposed forever in the repository's history.

Solution: Use environment variables. A common pattern is to use a .env file for local development.

Example (Python): A great library for this is python-dotenv .

1. Install it: pip install python-dotenv
2. Create a file named .env in your project root with your secret:

HACKATHON_API_KEY="your_secret_key_here"

3. Load it in your code:

python

```
import os
from dotenv import load_dotenv

# Load variables from .env file into environment
load_dotenv()

api_key = os.getenv("HACKATHON_API_KEY")
```

Crucially, you must add .env **to your** .gitignore **file** to prevent your secrets from ever being committed.

2.4.1. Project Structure Best Practices

Your project must be understandable in 5 minutes. Clear structure and documentation are critical.

MANDATORY PROJECT STRUCTURE:

```
/your-project
├── README.md            # CRITICAL - see requirements below
├── requirements.txt     # All dependencies with versions
├── .env.example         # Template for API keys (NO SECRETS!)
├── docker-compose.yml   # If using containers (recommended)
├── /src                 # Your source code
├── /docs
│   ├── ARCHITECTURE.md  # How your system works
│   └── DATA_FLOW.md     # Visual diagram: upload → processing → output
└── /tests               # Even basic tests help
```

MANDATORY: Every project must have a README.md with:

1. One-sentence description
2. Technology stack (languages, key libraries, AI models used)
3. Data Privacy Statement:

a. Where data is processed (Local / EU cloud / Specific region)
b. AI services used (Self-hosted Llama 3 / Vertex AI EU / None)
c. Does data leave the EU? (No / Only aggregated statistics)
d. Monthly cost estimate: €X

4. Prerequisites (Python 3.11, Docker, etc.)
5. Setup instructions (copy-paste commands)
6. How to run locally
7. How to deploy to production
8. Known limitations

3. Evaluation

3.1. Criteria

Criterion	Strategic Priority
Feasibility	Production-ready & simplicity ⭐⭐⭐
Quality	Well-designed solution ⭐⭐
Relevance	Technical fit & scalability ⭐⭐
Originality	Innovative approach ⭐
Transferability	Broader applicability ⭐

3.2. Pitching

At the end of the hackathon, you'll present a short demo of your work. Focus on what you've built, what you've learned, and the potential value of your idea.

We will host a short pitching workshop on the 2nd day. You will have **5 minutes to present** your solution, followed by **2 minutes of questions** from the jury or audience.

3.3 Submission

All solutions must be submitted by 11:59 AM on the 2nd day of the hackathon on the designated platform: Git Hub This will be the documentation available to the jury to evaluate & your pitch.

3.3.1. DigiEduHack

All team members must register on the DigiEduHack platform before the official start of the hackathon to be considered as an official participant.

After successful submission of your solutions, you will have 1 additional hour (until 1 PM) to submit the DigiEduHack Solution Canvas (available on Git Hub) to the DigiEduHack submission platform. Only 1 team member is submitting the solution.

4. Code of Conduct & Hackathon Etiquette

Collaborate and Respect: This is a team event. Support your teammates, listen to their ideas, and work together. Extend that respect to all other teams and the event organizers.

Innovate and Have Fun: Think outside the box! This is your chance to experiment with new technologies and bold ideas. Don't be afraid to fail and learn. The primary goal is to learn and have fun.

Ask for Help: Mentors and organizers are here to help you. If you're stuck on a technical problem, have a question about the guidelines, or need any other assistance, please don't hesitate to reach out.

We are incredibly excited to see what you create. Let's make this a fantastic event for everyone involved.

Happy Hacking!

BONUS: Free Access to Premium AI Models

Ready to build amazing AI applications? Thanks to our partner AI Tinkerers, you get unlimited access to thousands of open-source AI models with Featherless.ai - a serverless AI inference platform that handles all the infrastructure while you focus on building!

#Special Hackathon Offer

✅ Use coupon code: **DigiEduHack**

✅ Get Feather Premium access ($25/month value)

✅ Access to DeepSeek V3, Llama 3.3 70B, Kimi-K2, GLM-4.6 & more!

#Get Started in 3 Steps:

1️⃣Sign up

2️⃣Go toPricing → Select Feather Premium (**coupon pre-applied!**)

3️⃣Get your API key from Account → API Keys

4️⃣Choose your model from the catalog and start building!

Resources:

• Model Catalog - Browse 12K+ models

• Documentation - Full API docs & guides

• GitHub - Code samples & cookbook

• Application guides - Cursor, Aider, n8n, Dify & more