Director of Engineering Case Study

Time: Max 3 hours

The Challenge

Build an AI-powered dashboard that identifies at-risk students in Varsity Tutors and recommends interventions.

Requirements:

- Student list with risk indicators (high/medium/low)
- AI-generated explanations of WHY each student is at-risk
- AI-generated action recommendations per student (see section below for ideas)
- At least one data visualization (engagement trends, etc.)

Generate Sample Data

Run this script to create your test data:

```python
# generate_data.py
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import random

np.random.seed(42)
random.seed(42)

data = []
start_date = datetime(2025, 1, 6)
```

```
subjects = ["Math - Algebra", "Math - Geometry", "English - Writing",
          "Science - Biology", "History - US History"]
names = ["Alex", "Jordan", "Taylor", "Morgan", "Casey", "Riley", "Avery",
         "Quinn", "Parker", "Reese"] * 8

for i in range(75):
    archetype = random.choice(["thriving", "stable", "declining", "struggling"])
    base_engagement = {"thriving": 9, "stable": 7, "declining": 8, "struggling":
4}[archetype]
    sessions_per_week = {"thriving": 3, "stable": 2, "declining": 2, "struggling":
1}[archetype]

    for week in range(12):
        engagement = base_engagement
        if archetype == "declining":
            engagement = max(2, base_engagement - week * 0.6)

        for _ in range(sessions_per_week if random.random() > 0.2 else
sessions_per_week - 1):
            data.append({
                "student_id": f"STU-{i+1:03d}",
                "student_name": names[i],
                "grade_level": random.randint(6, 10),
                "session_date": (start_date + timedelta(weeks=week,
days=random.randint(0, 6))).strftime("%Y-%m-%d"),
                "session_duration_minutes": random.choice([30, 45, 60]),
                "subject": random.choice(subjects),
                "tutor_id": f"TUT-{random.randint(101, 125)}",
                "completed": random.random() > 0.1,
                "engagement_score": int(engagement) + random.randint(-1, 1),
                "tutor_notes": f"Session notes here. Student engagement:
{int(engagement)}/10",
                "homework_completed": random.random() > 0.3,
                "sessions_this_week": sessions_per_week,
                "total_sessions": week * sessions_per_week + 1
            })

pd.DataFrame(data).to_csv("tutoring_data.csv", index=False)
print(f"✅ Generated tutoring_data.csv with {len(data)} sessions for 75 students")
```

Run: python generate_data.py
Output: tutoring_data.csv (75 students, 12 weeks, ~1500 sessions)

AI-Generated Recommendation Examples (VT-Specific) ideas for you

Note you're only limited by VT Specific services we offer or something compelling and new that you think of!

Tutoring Adjustments:

- "Increase session frequency from 1x to 2x per week"
- "Schedule back-to-back sessions to maintain momentum"
- "Switch from 30-minute to 60-minute session format"
- "Match with different tutor who specializes in [subject]"
- "Add second subject (struggling in multiple areas)"

Scheduling & Availability:

- "Move sessions to earlier time slot (better engagement pattern)"
- "Book consistent weekly schedule (currently irregular)"
- "Add weekend session to catch up on missed material"

Parent/Admin Communication:

- "Send progress alert to administrator"
- "Schedule parent conference to discuss attendance pattern"
- "Share tutor notes and engagement trends with guardian"

Academic Support:

- "Assign practice problems between sessions"
- "Provide supplemental learning materials for [topic]"
- "Focus next 3 sessions on [specific struggling area]"
- "Review session recordings to identify knowledge gaps"

Intervention Escalation:

- "Flag for academic intervention team review"
- "Recommend assessment for additional support needs"
- "Place on monitoring list for weekly check-ins"

Technical Requirements

- Must use an LLM (Claude, OpenAI, Gemini, etc) for risk explanations and recommendations
- Deploy to public URL (Vercel, Netlify, etc.)
- Load time <3 seconds
- Tech stack: Your choice (React, Vue, Python, Node.js, etc.)
- AI coding tools: Use any (Cursor, Claude Code, Copilot, Windsurf, etc)

Submit

Email: [RECRUITER EMAIL]
Subject: "Director of Engineering Case Study - [Your Name]"

Include:

1. Deployed dashboard URL (ideally a public link that we can access)
2. GitHub repository
3. 3-5 min demo video link (Loom/YouTube/etc)
4. AI development evidence:
    - Development log with prompts & timestamps
    - Git history with AI attribution

Required Documentation (in your repo README)

1. Setup - How to run locally
2. AI Development Process ⭐ - Which tools, % AI-generated, effective prompts, time breakdown
3. Risk Logic - How you define "at-risk" and why
4. AI Implementation - How you're using LLM for insights
5. Production Considerations - Scalability, cost, security
6. Trade-offs - What you cut, what you'd add next

Evaluation (35-min live interview if selected)

- 10 min: Demo walkthrough

- 15 min: AI usage deep dive
- 10 min: Production discussion

Scoring:

- AI-Native Development (40%) - Tool proficiency, prompts, iteration
- Technical Execution (30%) - Features work, clean code, deployed
- Problem-Solving (15%) - Risk logic, scalability, security, trade-offs
- Documentation (15%) - Clear README, AI evidence

Tips

- Use AI for: Scaffolding, components, data processing, LLM integration, debugging
- You decide: Architecture, risk logic, UX, prompt engineering
- Ship working is better than perfect - Core features matter most

Questions? Email us!

Good luck!