Take-Home Assignment: Training Performance Dashboard

Fullstack Software Engineer (Data & AI) - Attensi

Time Estimate: 4-6 hours

Project Overview

Build a simplified dashboard that visualizes training performance data from gamified simulations. We want to see your ability to create clean, intuitive interfaces that make data actionable, while demonstrating effective use of modern development practices and AI tools.

Requirements

1. Frontend - Data Visualization Interface

Create an interactive dashboard with the following features:

Visualizations (implement at least 2-3):

- **Performance Trend**: Line chart showing score trends over time
- **Skills Comparison**: Bar or radar chart comparing skills across departments
- **Your Choice**: One additional meaningful visualization that provides value

Interactive Features:

- Filter by date range (e.g., last 7 days, last month, custom range)
- Filter by department
- Hover interactions showing detailed information
- Responsive design (must work well on desktop and tablet)

UI/UX Considerations:

- Clean, professional design
- Clear data labels and legends
- Loading states while data is being fetched
- Empty states if no data matches filters

2. Backend API

Create a simple Node.js server with one endpoint:

```
GET /api/insights
```

This endpoint should:

- Read training data from a JSON file on disk (provided below)
- Process and return the data with basic statistics such as:
  - Average scores by department
  - Top performing skills
  - Recent performance trends
  - Total sessions and pass rates
- Support query parameters for filtering (e.g., `?department=Sales&startDate=2024-01-01`)

3. AI-Powered Feature

Implement **ONE** of the following AI-enhanced features:

Option A: Natural Language Insights

- Use an AI service (OpenAI, Claude API, or similar) to generate human-readable insights
- Example: "Sales department shows 15% improvement in communication skills this month"
- Can be mocked if you don't have API access - just demonstrate the integration pattern

Option B: Anomaly Detection

- Identify unusual patterns in the data (sudden drops, outliers, etc.)
- Highlight these in the UI with explanations
- Can use simple statistical methods or AI services

Option C: Predictive Performance Alerts

- Analyze historical patterns to predict potential issues
- Example: "Based on trends, the Sales team may need additional Product Knowledge training"
- Generate 3-5 automated alerts/recommendations based on data patterns
- Display these prominently in the dashboard with visual indicators (warning icons, color coding)

4. Technical Requirements

Required Stack:

- **Frontend**: Vue 3 + TypeScript (strongly preferred) OR React + TypeScript
- **Backend**: Node.js (Express or similar)
- **Styling**: Your choice (Tailwind CSS, styled-components, CSS modules, etc.)
- **Charts**: Any library (Chart.js, D3.js, Recharts, etc.)

Code Quality:

- TypeScript for type safety
- Clean, readable code with meaningful variable names
- Component-based architecture
- Proper error handling
- Basic performance considerations (avoid unnecessary re-renders, efficient data processing)

Data Structure

Your application should work with this data structure. Create a file called `training-data.json`:

```
{
  "metadata": {
    "generatedAt": "2024-11-01T10:00:00Z",
    "version": "1.0"
  },
  "sessions": [
    {
      "sessionId": "550e8400-e29b-41d4-a716-446655440001",
      "userId": "user001",
      "userName": "John Doe",
      "department": "Sales",
      "date": "2024-10-28",
      "overallScore": 85,
      "skills": {
        "communication": 88,
        "problemSolving": 82,
        "productKnowledge": 85,
        "customerService": 90
      },
      "completionTime": 25,
      "passed": true
    },
    {
      "sessionId": "550e8400-e29b-41d4-a716-446655440002",
      "userId": "user002",
      "userName": "Jane Smith",
      "department": "Support",
      "date": "2024-10-28",
      "overallScore": 92,
      "skills": {
        "communication": 95,
        "problemSolving": 90,
        "productKnowledge": 88,
        "customerService": 94
      },
      "completionTime": 22,
      "passed": true
    },
    {
      "sessionId": "550e8400-e29b-41d4-a716-446655440003",
      "userId": "user003",
      "userName": "Bob Johnson",
      "department": "Sales",
      "date": "2024-10-29",
      "overallScore": 68,
      "skills": {
        "communication": 70,
        "problemSolving": 65,
        "productKnowledge": 72,
        "customerService": 66
      },
      "completionTime": 35,
      "passed": false
    }
  ]
}
```


Note: Feel free to generate more sample data (20-50 sessions) across different dates and departments to better demonstrate your dashboard's capabilities. You can use AI tools to help generate realistic test data.

Deliverables

1. Code Repository

Submit a GitHub repository (public or private) containing:

- Complete source code
- `README.md` with:
  - Project overview
  - Setup instructions (step-by-step)
  - Available scripts (start, build, test, etc.)
  - Brief explanation of your design decisions (max 500 words)
  - Screenshots of the running application

2. AI Tools Documentation

Create a file called `AI_TOOLS_USAGE.md` documenting:

- Which AI tools you used (GitHub Copilot, ChatGPT, Claude, etc.)
- Specific tasks where AI tools helped
- Example prompts that worked well
- Time saved estimates
- Any AI-generated code that required manual fixes

Example format:

```
## AI Tools Usage

### Code Generation
- **Tool**: GitHub Copilot
- **Task**: Generated Vue 3 composables for data filtering
- **Time Saved**: ~15 minutes
- **Notes**: Had to manually adjust TypeScript types

### Problem Solving
- **Tool**: Claude
- **Task**: Debugged chart rendering issue
- **Prompt**: "My Chart.js line chart in Vue 3 is not updating when..."
- **Time Saved**: ~30 minutes
```

3. Future Improvements

List 2-3 features or improvements you would add with more time, such as:

- Real-time data updates
- Advanced analytics
- Export functionality
- Mobile responsive design
- Performance optimizations

Evaluation Criteria

We will evaluate your submission based on:

1. **Functionality** (30%)
   - All requirements are met
   - Application works without errors
   - Filters and interactions work correctly
2. **Code Quality** (25%)
   - Clean, maintainable code
   - Proper TypeScript usage
   - Good component structure
   - Error handling
3. **UI/UX Design** (20%)
   - Intuitive interface
   - Good data visualization choices
   - Responsive design
   - Professional appearance
4. **AI Integration** (15%)
   - Effective use of AI tools in development
   - Quality of AI-powered feature
   - Documentation of AI usage

5. **Documentation** (10%)
    - Clear setup instructions
    - Code comments where necessary
    - Good commit messages
    - Design decisions explained

Tips for Success

- **Start Simple**: Get a basic version working first, then enhance
- **Use AI Wisely**: Leverage AI tools for boilerplate and problem-solving, but ensure you understand the code
- **Focus on User Experience**: Think about how real users would interact with your dashboard
- **Show Your Process**: Good commit history and documentation help us understand your thinking
- **Ask Questions**: If something is unclear, reach out to your recruiter for clarification

---

Getting Started

1. Fork or create a new repository
2. Set up your Vue 3 + TypeScript (or React) project
3. Create a simple Node.js server
4. Start with reading and displaying the data
5. Add visualizations incrementally
6. Implement filters
7. Add your chosen AI feature
8. Polish UI and documentation

Submission

Please submit your completed assignment by sending:

1. Link to your GitHub repository
2. Any additional notes or clarifications
3. Confirmation that the project runs with the provided instructions
4. Email your submission to the recruiter.

Questions?

If you have any questions about the assignment, please don't hesitate to reach out. We want you to succeed and show us your best work!

Good luck, and we look forward to reviewing your submission!

Note: This assignment is designed to be completed in 4-6 hours. We respect your time - please don't spend significantly more time than this. We're more interested in seeing your problem-solving approach and code quality than a perfect, production-ready application.