EduAI

Ishwar Babu

June 5, 2025

1 Introduction

This document provides a technical overview of the Student Performance Analysis System, a Python application , made from google colab, that processes student test data, generates visual analytics, and produces AI-powered feedback reports.

2 System Architecture

The system consists of four main components:

- **Data Processing Layer**: Handles JSON input validation and transformation
- **Analytics Layer**: Generates visualizations and performance metrics
- **AI Feedback Layer**: Uses LLM to generate personalized feedback
- **Reporting Layer**: Creates professional PDF reports
- **API Used**: DeepSeek V3 from OpenRouter, Gradio

3 Core Functions

3.1 Data Processing

The system begins by validating and processing the input JSON data:

```
def load_and_validate_data(file):
    if file is None:
        raise ValueError("No file uploaded!")

    try:
        with open(file.name, 'r') as f:
            submission = json.load(f)

        required = ['totalTimeTaken', 'totalMarkScored',
                    'totalAttempted', 'totalCorrect',
                    'accuracy', 'subjects', 'sections']

        for field in required:
            if field not in submission:
                raise ValueError(f"Missing required field: {field}")

        return submission
    except json.JSONDecodeError:
        raise ValueError("Invalid JSON file format")
```

Key features:

- Validates required fields exist in JSON
- Handles both single objects and arrays
- Provides descriptive error messages

3.2 Data Transformation

The system processes subject and chapter data into structured DataFrames:

```
def process_subject_data(subjects):
    return pd.DataFrame([{
        "Subject": s['subjectId']['$oid'],
        "Accuracy (%)": round(s['accuracy'], 2),
        "Attempts": s.get('attempted', 0),
        "Correct": s.get('correct', 0)
    } for s in subjects])
```

3.3 Visualization Generation

Creates a comprehensive dashboard with four subplots:

```
def create_visualizations(subjects_df, chapter_df):
    plt.style.use('ggplot')
    fig, axs = plt.subplots(2, 2, figsize=(16, 12))
    fig.suptitle('Student Performance Analysis Dashboard',
                 fontsize=18, y=0.98, fontweight='bold')

    # Subject-wise accuracy plot
    subjects_sorted = subjects_df.sort_values('Accuracy (%)', ascending=False)
    bars1 = axs[0,0].bar(subjects_sorted['Subject'],
                         subjects_sorted['Accuracy (%)'],
                         color='#4C72B0', alpha=0.8)
    axs[0,0].set_ylim(0, 100)
    axs[0,0].set_title("Subject-wise Accuracy",
                       fontsize=14, fontweight='bold')
```

Visualizations include:

- Subject-wise accuracy bar chart
- Top chapters by accuracy
- Time vs accuracy scatter plot
- Marks distribution

3.4 AI Feedback Generation

The system composes a detailed prompt for the LLM:

```
def compose_feedback_prompt(student_data, subjects_df, chapter_df):
    return f"""
You are an experienced educational mentor providing personalized feedback...

STUDENT'S TEST PERFORMANCE SUMMARY:
- Total Time Taken: {time_formatted}
- Total Marks Scored: {student_data['totalMarkScored']}
- Questions Attempted: {student_data['totalAttempted']}
- Overall Accuracy: {student_data['accuracy']:.1f}%

Please provide a comprehensive, personalized feedback report with:
1. MOTIVATIONAL OPENING
2. DETAILED PERFORMANCE ANALYSIS
3. KEY INSIGHTS & OBSERVATIONS
4. ACTIONABLE IMPROVEMENT STRATEGIES
5. PERSONALIZED STUDY PLAN
6. ENCOURAGING CONCLUSION
"""
```

3.5 PDF Report Generation

Creates a professional PDF with multiple sections:

```
class generatePDF(FPDF):
    def header(self):
        self.set_fill_color(41, 128, 185)
        self.rect(0, 0, 210, 25, 'F')
        self.set_text_color(255, 255, 255)
        self.set_font('Arial', 'B', 20)
        self.cell(0, 15, 'Student Performance Analysis Report',
                  ln=1, align='C')
```

PDF features:

- Custom header/footer
- Section styling
- Data tables
- Embedded visualizations
- Formatted text

4 Gradio Interface

The web interface provides user-friendly interaction:

```
def create_gradio_interface():
    with gr.Blocks(title="Student Performance Analyzer") as interface:
        gr.Markdown("# Student Performance Analysis Dashboard")

        with gr.Row():
            file_input = gr.File(label="Upload JSON File")
            analyze_btn = gr.Button("Analyze Performance")

        with gr.Row():
            summary_output = gr.Markdown()
            chart_output = gr.Image()
            pdf_output = gr.File()
            feedback_output = gr.Textbox()
```

Interface components:

- File upload
- Analysis button displays
- PDF download

5 Error Handling

The system implements robust error handling:

```
def generate_llm_feedback(prompt):
    try:
        # API call
    except Exception as e:
        fallback_feedback = f"""Dear Student,
I apologize, but I'm unable to generate personalized AI feedback...
"""
        return fallback_feedback
```

6 Conclusion

This system provides a comprehensive solution for analyzing student performance data with:

- Robust data processing
- Insightful visualizations

- AI-powered feedback
- Professional reporting
- User-friendly interface

Potential improvements could include:

- Database integration
- Batch processing
- Additional analytics metrics

Project Repository

The source code and related files for this system are available on GitHub:
`https://github.com/ib105/EduAI`