Self-Introduction Speech Scoring Tool

An AI-powered tool that analyzes and scores student self-introductions based on comprehensive communication rubrics. The tool evaluates content structure, speech rate, grammar, vocabulary richness, clarity, and engagement.

🎯 Features

- **Comprehensive Scoring**: 100-point rubric-based evaluation across 6 criteria
- **Real-time Analysis**: Instant feedback on speech transcripts
- **Detailed Breakdown**: Per-criterion scores with actionable feedback
- **Modern UI**: Clean, responsive interface with visual score representations
- **Dual Backend Options**:
  - Python Flask API with NLP libraries
  - Claude AI-powered analysis (used in React frontend)

📊 Scoring Criteria

Criterion	Max Points	Key Factors
Content & Structure	40	Salutation, keywords (name, age, school, family, hobbies), flow
Speech Rate	10	Words per minute (ideal: 111-140 WPM)
Language & Grammar	15	Grammar errors using LanguageTool
Vocabulary Richness	10	Type-Token Ratio (TTR)
Clarity	10	Filler word rate
Engagement	15	Sentiment analysis (positive/neutral/negative)

🚀 Quick Start

Prerequisites

- Python 3.8+
- Node.js 16+ (for React frontend)
- pip (Python package manager)

Installation

1. **Clone the repository**

```
bash
```

```
git clone https://github.com/yourusername/speech-scoring-tool.git
cd speech-scoring-tool
```

2. **Backend Setup (Python)**

```bash
# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Download required NLTK data
python -c "import nltk; nltk.download('punkt'); nltk.download('brown')"
```

3. **Run Backend Server**

```bash
python app.py
```

The API will be available at `http://localhost:5000`

API Endpoints

POST /api/analyze

```json
{
  "transcript": "Your self-introduction text here...",
  "duration_seconds": 52
}
```

Response:

```json
```

```
{
  "overall_score": 86,
  "word_count": 131,
  "sentence_count": 11,
  "duration_seconds": 52,
  "wpm": 151,
  "criteria": [
    {
      "name": "Content & Structure",
      "score": 35,
      "max_score": 40,
      "details": {...},
      "feedback": "Strong opening. All essential information included..."
    }
    // ... other criteria
  ],
  "summary": "Excellent self-introduction with strong communication skills..."
}
```

GET /api/health

- Returns server health status

🎨 Frontend (React)

The React frontend uses Claude AI API for analysis and provides:

- Interactive text input
- Real-time scoring visualization
- Detailed criterion breakdown
- Sample text loading
- Responsive design

Test with Sample Data

The application includes sample text from the case study:

```
Hello everyone, myself Muskan, studying in class 8th B section from Christ Public School...
```

Click "Load Sample" button to test the analysis.

📁 Project Structure

```
speech-scoring-tool/
```

```
├── app.py              # Flask backend API
├── requirements.txt      # Python dependencies
├── README.md           # Documentation
├── DEPLOYMENT.md        # Deployment guide
├── sample_data.json      # Sample test data
└── tests/
    └── test_api.py       # API tests
```

🔬 Scoring Methodology

1. Content & Structure (40 points)

Salutation (5 points)

- No salutation: 0 points
- Normal (Hi, Hello): 2 points
- Good (Good Morning, Hello everyone): 4 points
- Excellent (excited to introduce): 5 points

Keywords (20 points)

- Must-have (4 points each): Name, Age, School/Class, Family, Hobbies
- Good-to-have (2 points each): Family details, Location, Ambition, Unique fact, Strengths

Flow (15 points)

- Proper order: Salutation → Basic details → Additional details → Closing

2. Speech Rate (10 points)

Based on WPM calculation: `(word_count / duration_seconds) * 60`

- Ideal: 111-140 WPM (10 points)
- Fast/Slow: 81-110 or 141-160 WPM (6 points)
- Too Fast/Slow: <80 or >161 WPM (2 points)

3. Language & Grammar (15 points)

Grammar Score = `1 - min(errors_per_100_words / 10, 1)`

- Uses LanguageTool Python for error detection

4. Vocabulary Richness (10 points)

Type-Token Ratio: `distinct_words / total_words`

- > 0.9: 10 points

- 0.7-0.89: 8 points
- 0.5-0.69: 6 points

5. Clarity (10 points)

Filler Word Rate: `(filler_count / total_words) * 100`

- Detects: um, uh, like, you know, so, actually, basically, etc.

6. Engagement (15 points)

Sentiment analysis using TextBlob

- Positive probability (0-1 scale)
- Higher positivity = higher score

🧪 Testing

Manual Testing

```bash
# Test with curl
curl -X POST http://localhost:5000/api/analyze \
  -H "Content-Type: application/json" \
  -d @sample_data.json
```

Python Tests

```bash
python tests/test_api.py
```

🚀 Deployment

Local Deployment

1. **Start Backend**

```bash
python app.py
```

2. **Access API**

- Local: `http://localhost:5000`
- Network: `http://YOUR_IP:5000`

Cloud Deployment Options

AWS Free Tier

1. Launch EC2 instance (t2.micro)
2. Install dependencies
3. Run with Gunicorn:

```bash
gunicorn -w 4 -b 0.0.0.0:5000 app:app
```

Heroku

```bash
# Create Procfile
echo "web: gunicorn app:app" > Procfile

# Deploy
heroku create your-app-name
git push heroku main
```

Railway/Render

- Connect GitHub repository
- Set Python environment
- Auto-deploy on push

📝 Sample Input/Output

Input:

```json
{
  "transcript": "Hello everyone, myself Muskan, studying in class 8th B section from Christ Public School. I am 13 years old.
  "duration_seconds": 52
}
```

Expected Output:

```json
```

```
{
  "overall_score": 86,
  "word_count": 131,
  "sentence_count": 11,
  "duration_seconds": 52,
  "wpm": 151,
  "criteria": [...]
}
```

🛠️ Technology Stack

Backend:

- Flask (Web framework)
- LanguageTool (Grammar checking)
- TextBlob (Sentiment analysis)
- NLTK (Natural language processing)

Frontend:

- React 18
- Tailwind CSS
- shadcn/ui components
- Lucide icons

AI Integration:

- Claude AI API (Anthropic)

📚 Rubric Details

The scoring rubric is based on the Nirmaan AI Communication Program requirements with precise calculations for each criterion. See case study documentation for complete rubric specifications.

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit changes (`git commit -m 'Add AmazingFeature'`)
4. Push to branch (`git push origin feature/AmazingFeature`)
5. Open Pull Request

📄 License

This project is created for the Nirmaan AI Intern Case Study.

👥 Author

Created as part of the Nirmaan AI Internship Application

🙏 Acknowledgments

- Nirmaan AI for the case study opportunity
- Case study rubric based on communication assessment standards
- Sample transcript provided in case study materials

📧 Contact

For questions or feedback, please open an issue in the GitHub repository.

Note: This tool is designed for educational assessment purposes and provides automated feedback on communication skills. Results should be used as guidance alongside human evaluation.