Communication Scoring Platform – Full Documentation (PDF Source)

Version: 2.1.1 (lite mode if semantic disabled)
Hosted Frontend: https://communication-scoring-platform-nlp-rubric-fast-api-ot0ra8fqb.vercel.app
(Optional Backend Semantic Mode)

Table of Contents

1. Overview
2. Feature Summary
3. Architecture
4. Rubric & Scoring Formula (v2.1.1)
5. Data Extraction (Name / Age / Class)
6. API Specification
7. Local Installation & Run (Backend + Frontend)
8. Deployment Guide (Render + Vercel)
9. Environment Variables
10. Evaluation Walkthrough
11. Troubleshooting
12. Warm-Up Endpoint (Optional)
13. Testing Strategy
14. Extensibility & Roadmap
15. Video Demo Guide
16. Submission Checklist

17. License
18. Acknowledgments
19. Appendix: Example Full Evaluation JSON

1. Overview

The Communication Scoring Platform evaluates student (or speaker) self-introduction transcripts via a rubric-based, explainable NLP pipeline. It produces per-metric scores, structured feedback, and extracted personal details for transparency. Architecture: FastAPI backend + React/Vite/Tailwind frontend.

2. Feature Summary

- Modular metric pipeline (salutation, keywords, flow, speech rate, grammar, vocabulary, clarity, sentiment, conceptual coverage).
- Concept-based keyword detection (regex + extraction) rather than raw substring tokens.
- Grammar analysis via LanguageTool (with graceful fallback).
- Sentiment scoring via VADER.
- Filler rate calculation.
- Speech rate (WPM) scoring using transcript duration.
- Vocabulary diversity via Type–Token Ratio (TTR).
- Optional semantic conceptual coverage (Sentence-Transformers) – toggled by `ENABLE_SEMANTIC` .
- Extracted metadata: speaker name, age, class/school phrase.
- Performance timing in ms.
- Lite mode (semantic disabled) vs full mode (semantic enabled).

3. Architecture

```
frontend (React/Vite/Tailwind)
    ├─ Transcript input + duration
    ├─ Calls POST /api/v2/evaluate
    └─ Renders metric cards & details

backend (FastAPI)
    ├─ main.py (routes, CORS, health, evaluate)
    ├─ models.py (Pydantic response schemas)
    ├─ scoring/
    │    ├─ metrics.py (all rule-based metrics)
    │    ├─ extraction.py (regex-based detail extraction)
    │    ├─ semantic.py (heavy embedding model; optional)
    │    ├─ pipeline_v2.py (orchestrates evaluation flow)
    │    ├─ utils.py (tokenization, helpers)
    │    └─ constants.py (keyword lists)
    └─ tests/ (unit + integration tests)
```

4. Rubric & Scoring Formula (v2.1.1)

4.1 Notation

- `T` : transcript text
- `W` : word count
- `S` : sentence count
- `D` : duration seconds (optional)
- `WPM = W / (D / 60)` if `D > 0`

- `F` : count of filler words
- `R_filler = (F / W) * 100`
- `E` : grammar errors
- `E100 = (E / W) * 100`
- `grammar_raw = 1 - min(E100 / 10, 1)`
- `TTR = unique_word_count / W`
- `pos_prob` : VADER positive probability
- `avg_sim` : semantic similarity average (full mode only)

4.2 Metrics

Metric	Max	Logic (Band Summary)
Salutation	5	none=0, normal=2, good=4, excellent=5
Keyword Presence	30	Must-have (8 concepts ×4 pts), Good-to-have (each 2 pts), capped 30
Flow Order	5	Greeting → basics → additional → closing sequence required
Speech Rate	10	Ideal 111–140 WPM=10; fast/slow=6; extremes=2; no duration=0
Grammar	10	grammar_raw mapped to 10/8/6/4/2
Vocabulary (TTR)	10	TTR bands ≥0.9=10; 0.7–0.89=8; 0.5–0.69=6; 0.3–0.49=4; <0.3=2
Clarity (Filler Rate)	15	0–3%=15; 4–6%=12; 7–9%=9; 10–12%=6; ≥13%=3
Engagement (Sentiment)	15	pos_prob ≥0.9=15; 0.7–0.89=12; 0.5–0.69=9; 0.3–0.49=6; <0.3=3
Conceptual Coverage	10	avg_sim bands (≥0.80=10 → <0.50=2) (full mode only)

Full mode total max: 110

Lite mode total max: 100 (conceptual coverage disabled, score=0, band=disabled)

4.3 Keyword Concepts

- Must-have: name, age, class, school, family, hobby, interest, like
- Good-to-have: origin, parents are from, ambition, goal, dream, achievement, strength, fun fact, unique, aspire, interesting Score formula: `S_kw = min(4*M_found + 2*G_found, 30)`

4.4 Grammar

`grammar_raw = 1 - min(E100 / 10, 1)` → map to band and score. Fallback: if LanguageTool fails → score 10.

4.5 Speech Rate Bands

WPM Range	Score
≤80 / >160	2
81–110 / 141–160	6
111–140	10

4.6 Total Score

`total_score = Σ(metric_scores)`

`max_total = 110 (full)` or `100 (lite)`

5. Data Extraction

Regex/extraction functions identify:

- Name: “my name is X”, “myself X”, “I am X”
- Age: “I am NN years old”
- Class: “class NN” or “studying in class NN”
- School phrase: “ School/Academy/College”, optionally combined with class.

Extraction fields appear under `extracted` in response; they do not affect points.

6. API Specification

6.1 Endpoints

Method	Path	Description
GET	/api/v2/health	Status/version JSON
POST	/api/v2/evaluate	Transcript evaluation
GET	/api/v2/ping	Simple timestamp (optional)
GET	/api/v2/warmup	Preload heavy resources (optional)

6.2 Request

`POST /api/v2/evaluate`

```
{
  "transcript": "Hello everyone, my name is Arjun. ... Thank you.",
  "duration_seconds": 55
}
```

6.3 Response (Lite Example)

```
{
  "total_score": 87.0,
  "max_total": 100,
  "word_count": 44,
  "metrics": [...],
  "extracted": {"name":"Arjun","age":13,"school_class":"Riverdale School, Class 8"},
  "version": "2.1.1-lite",
  "performance_ms": 412
}
```

6.4 Error Responses

Status	Reason
400	Empty or <10 word transcript
422	Malformed JSON
500	Internal evaluation error (rare; check logs)

7. Local Installation & Run

7.1 Backend

```
cd backend
python -m venv .venv
# Windows: .venv\Scripts\Activate.ps1
source .venv/bin/activate
pip install -r requirements.txt
python -c "import nltk; nltk.download('vader_lexicon')"
uvicorn app.main:app --reload
```

Health: http://127.0.0.1:8000/api/v2/health

Enable semantic (optional):

```
export ENABLE_SEMANTIC=true
uvicorn app.main:app --reload
```

7.2 Frontend

```
cd frontend
npm install
echo "VITE_API_BASE_URL=http://127.0.0.1:8000" > .env
npm run dev
```

Visit http://localhost:5173

8. Deployment Guide

8.1 Backend (Render)

- Root directory: `backend`
- Build command:

  ```
  pip install -r requirements.txt && python -c "import nltk; nltk.download('vader_lexicon')"
  ```

- Start command:

  ```
  uvicorn app.main:app --host 0.0.0.0 --port $PORT
  ```

- Environment variable: `ENABLE_SEMANTIC=false` (recommended on free tier)

8.2 Frontend (Vercel)

- Root: `frontend`
- Build: `npm run build`
- Output: `dist`
- Env Vars: `VITE_API_BASE_URL=https://<backend-domain>`

8.3 CORS

In `main.py` adjust origins:

```
allow_origins=["https://<your-frontend>.vercel.app"]
```

8.4 Warm-Up (Optional)

GET `/api/v2/warmup` once post-deploy to reduce first-user latency.

9. Environment Variables

Variable	Purpose	Default
ENABLE_SEMANTIC	Load sentence-transformers semantic metric	false
VITE_API_BASE_URL (frontend)	Backend base URL	(must set)

10. Evaluation Walkthrough

1. User enters transcript & duration.
2. Frontend POSTs JSON → backend.
3. Pipeline assembles metrics:
    - Tokenization → counts
    - Regex/extraction for concepts
    - Grammar check → errors → score
    - Sentiment via VADER
    - Filler rate calculation
    - Speech rate band
    - Semantic coverage (if enabled)
4. Summation → total score.
5. Response returned ~0.3–1.0 sec (lite mode faster).

11. Troubleshooting

Symptom	Cause	Fix
400 "Transcript too short"	<10 words	Provide more text
Generic "Error" toast	Backend 500 / CORS	Check logs & ensure correct base URL
Very slow first request	LanguageTool JAR + model download	Warm-Up endpoint
Grammar always 10	Java missing (LanguageTool fails)	Install JRE
Semantic crash on free host	Memory limits	Disable semantic (ENABLE_SEMANTIC=false)

12. Warm-Up Endpoint

```
@app.get("/api/v2/warmup")
def warmup(bg: BackgroundTasks):
    def _load():
        if os.getenv("ENABLE_SEMANTIC","false").lower()=="true":
            from app.scoring.semantic import get_semantic_model
            get_semantic_model()
        from language_tool_python import LanguageTool
        LanguageTool('en-US')
    bg.add_task(_load)
    return {"status":"warming"}
```

13. Testing Strategy

Run:

```
cd backend
pytest -q
```

Test categories:

- Unit: grammar_metric, sentiment_metric, filler_words_metric, vocabulary_metric, extraction functions.
- Integration: pipeline total; semantic concept (when enabled).
- Regression: ensure total_score ≤ max_total.

14. Extensibility & Roadmap

Phases:

1. Transcript scoring (complete).
2. Lightweight semantic (TF-IDF alternative).
3. Audio pause & prosody analysis.
4. Instructor dashboard + historical comparisons.
5. Multi-language support (switch LanguageTool language code).
6. PDF report export / LMS integration.

Potential Additions:

- NER-based completeness scoring (spaCy).
- Adaptive feedback generation via LLM.
- Persistence (SQLite + user auth).

15. Video Demo Guide (Short Script)

1. Intro (what it does).
2. Paste transcript, set duration, click Score.
3. Show metric cards & JSON details.
4. Brief formulas (grammar Raw → band; keyword concepts).
5. Show pipeline_v2.py & metrics.py.
6. Mention optional semantic metric toggle.
7. Future roadmap summary.
8. Closing.

Target length: 2–3 minutes.

16. Submission Checklist

- Public repo with source, requirements, README.
- Docs: scoring formula, deployment steps (this file covers both).
- Deployed frontend & optional backend.
- Local run instructions (section 7).
- Video demo recorded.
- Optional tests & LICENSE.

17. License

MIT License (include `LICENSE` file in repo):

18. Acknowledgments

- LanguageTool (grammar)
- NLTK VADER (sentiment)
- Sentence-Transformers (optional conceptual coverage)
- FastAPI & React ecosystem

19. Appendix: Example Evaluation JSON (Full Mode)

```
{
  "total_score": 95.0,
  "max_total": 110,
  "word_count": 120,
  "sentence_count": 8,
  "duration_seconds": 65.0,
  "wpm": 110.77,
  "metrics": [
    {
      "id": "salutation",
      "name": "Salutation Level",
      "raw_score": 4,
      "max_score": 5,
      "details": {"level":"good","matched":["hello everyone"],"score":4,"max":5},
      "feedback": "Greeting level: Good."
      },
      {
        "id": "keywords",
        "name": "Keyword Presence",
        "raw_score": 30,
        "max_score": 30,
        "details": {
          "must_found": ["name","age","class","school","family","hobby","interest","like"],
          "must_missing": [],
          "good_found": ["fun fact","goal","dream","ambition"],
          "good_missing": ["origin","parents are from","achievement","strength","unique","aspire","interesting"],
          "score": 30,
          "max": 30
        },
        "feedback": "All key elements present."
      },
      {
        "id": "concept",
        "name": "Conceptual Coverage",
        "raw_score": 8,
        "max_score": 10,
        "details": {
          "average_similarity": 0.74,
          "band":"0.70-0.79",
          "score":8,
          "max":10
        },
        "feedback":"Conceptual coverage 0.74 (0.70-0.79)."
      }
    ],
    "extracted": {
      "name": "Arjun",
      "age": 13,
      "school_class": "Riverdale School, Class 8"
    },
    "transcript_preview": "Hello everyone, my name is Arjun...",
    "version": "2.1.1",
    "performance_ms": 690,
    "notes": "Full metric set"
  }
```

End of Document.