



# **A SPEECH TRANSCRIPTION AND EVALUATION TOOL WITH CANVAS LTI INTEGRATION**

## **Abstract**

This report documents the development of a web-based platform that transcribes, evaluates, and delivers feedback on spoken language responses using SYSTRAN's FasterWhisper model

**KWADWO AMPONSAH AMPOFO**

J00946242@students.jsums.edu

# Contents

<b>EXECUTIVE SUMMARY .....</b>	<b>3</b>
Background and Motivation .....	4
1. Introduction .....	5
2. Project Overview .....	5
3. Files and Structure .....	5
4. Dependencies and Setup.....	6
- Configuration .....	6
5. Code Overview and Explanation.....	7
5.1 app.py .....	7
5.2 whisper_eval.py.....	7
5.3 lti_config.py .....	7
5.4 LTI Configuration Files.....	7
5.5 Frontend Assets (static/ and templates/) .....	8
6. Usage Instructions .....	8
7. Testing Approach .....	9
Functional Testing.....	9
Model Validation .....	9
LTI Integration Testing.....	9
UI Testing .....	9
Results .....	9
8. Limitations and Future Work .....	10
9. Security Considerations .....	11
10. Appendices.....	12
Appendix A: app.py.....	12
Appendix B: whisper_eval.py.....	17
Appendix C: lti_config.py .....	18
Appendix D: tool_config.json .....	18
Appendix E: tool_config_backend.json .....	19
Appendix F: tool_config_template.json.....	19
Appendix G: requirements.txt .....	20
Appendix H: jwk_public.json .....	20
Appendix I: CSS (static/css/style.css) .....	20

Appendix J: JavaScript (static/js/main.js) .....21

Appendix K: HTML Template (templates/index.html) .....21

## EXECUTIVE SUMMARY

This report documents the development of a web-based platform that transcribes, evaluates, and delivers feedback on spoken language responses using SYSTRAN's FasterWhisper model. Designed to integrate seamlessly into educational platforms like Canvas LMS via the Learning Tools Interoperability (LTI) 1.3 standard, the system offers students a streamlined way to submit oral assessments and receive immediate, rubric-based evaluation.

The project addresses a key challenge in modern education: the need for scalable, automated, and intelligent tools that can assess spoken responses fairly and efficiently. By leveraging FasterWhisper for high-accuracy transcription and a customizable evaluation rubric, this system enables instructors to evaluate clarity, relevance, organization, and grammar in student submissions - quickly and consistently.

The backend is implemented in Python using Flask and is capable of handling multiple audio file formats and real-time recordings. Users can either upload pre-recorded audio files or capture responses directly via the web interface. The transcribed content is processed, evaluated against predefined rubric keywords, and presented to the user along with qualitative feedback.

From a systems integration perspective, the application supports secure LTI launches, allowing educational institutions to embed the tool directly within Canvas assignments. Authentication, session management, and public key handling follow industry best practices as defined by the IMS Global Learning Consortium (IMS Global Learning Consortium., 2025).

This report presents the entire project lifecycle - from design rationale to implementation details - along with source code, deployment instructions, and an in-depth walkthrough of the system's frontend and backend components. Future improvements will focus on expanding the evaluation model, bundling it into a mobile application, and enhancing the user interface for accessibility and performance.

## Background and Motivation

In the evolving landscape of digital education, the ability to assess spoken responses has become increasingly important. Oral assessments play a critical role in language learning, communication skills development, and reflective evaluations. However, traditional methods of evaluating oral submissions - such as manual listening and scoring - are time-consuming and difficult to scale, especially in large classrooms or online learning environments.

The rapid advancement of automatic speech recognition (ASR) and natural language processing (NLP) has opened new opportunities for improving how educational institutions handle oral submissions. Tools powered by ASR can now provide real-time transcription of spoken language with high accuracy. However, integrating these technologies into Learning Management Systems (LMS) like Canvas remains a challenge due to compatibility, authentication requirements, and lack of standardization.

This project, Oral Response Evaluator, was conceived to address these challenges by providing:

- A lightweight, secure, and accessible interface for students to record or upload speech.
- A high-accuracy transcription backend using Systran's FasterWhisper (SYSTRAN, 2025), an optimized and efficient implementation of the Whisper speech model (OpenAI, 2025).
- An automatic rubric-based evaluation of responses, providing immediate feedback on important criteria like clarity, organization, relevance, and grammar.
- Seamless integration with Canvas LMS using LTI 1.3, ensuring secure authentication, context awareness, and compatibility with academic workflows.

By combining open-source tools and standardized protocols, this system bridges the gap between modern AI-based evaluation and classroom implementation. It reduces the burden on educators while promoting a more interactive and responsive learning experience for students.

Ultimately, this work is motivated by the vision of creating a smart, scalable, and transparent tool for academic speech assessment - one that supports inclusive education and enhances feedback loops in both in-person and virtual classrooms.

# 1. Introduction

This project implements an Oral Response Evaluator that transcribes and evaluates speech responses using Systran's Faster Whisper model. It integrates with Canvas LMS through LTI 1.3 to provide seamless authentication and course context, allowing instructors and students to access oral transcription and evaluation directly within Canvas.

## 2. Project Overview

The project consists of:

- A Flask backend server that hosts the speech transcription and evaluation model.
- LTI integration endpoints to support secure launches from Canvas.
- A frontend interface for recording or uploading audio.
- Evaluation logic that scores responses based on keyword rubrics.

## 3. Files and Structure

File	Description
app.py	Main Flask application with routes for LTI, transcription, upload.
whisper_eval.py	Transcription and evaluation logic using Faster Whisper.
lti_config.py	Alternative Flask app handling LTI login and launch endpoints.
tool_config.json	Public LTI tool configuration JSON for Canvas integration.
tool_config_backend.json	Backend LTI configuration with client secrets (sample).
tool_config_template.json	Template config for easy deployment setup.
requirements.txt	Python dependencies required to run the app.
jwk_public.json	Public key JSON Web Key Set (JWKS) file for OAuth security.
static/	Static frontend assets (CSS, JS, images).
templates/	HTML templates for frontend UI.
keys/	RSA keys for JWT signing and verification.

## 4. Dependencies and Setup

### - Required Python Packages

- Flask
- pylti1p3 (for LTI 1.3 support)
- python-dotenv (for environment variable support)
- requests
- openai-whisper (Faster Whisper transcription model)

You can install dependencies using:

*bash*

```
pip install -r requirements.txt
```

### - Configuration

- Provide your Canvas Client ID, Deployment ID, and RSA keys in `tool_config_backend.json` and the `keys/` directory.
- Update URLs in `tool_config.json` to match your deployment endpoint (e.g., your public ngrok or server URL).

## 5. Code Overview and Explanation

### 5.1 app.py

- **Purpose:** Flask app implementing routes for LTI login, launch, transcription API, and upload UI.
- Uses pylti1p3 (Viskov, 2025) for LTI 1.3 OIDC login flow and launch validation.
- Limits transcription requests with Flask-Limiter.
- Saves uploaded audio files and passes them to the transcription and evaluation module.
- Provides an HTML interface allowing users to upload or record audio.

### 5.2 whisper\_eval.py

- Contains WhisperTranscriber class to load and run Faster Whisper transcription on audio.
- SimpleEvaluator class checks for rubric keywords and provides a score and feedback.
- transcribe\_and\_evaluate() combines both to return a transcript and evaluation metrics.

### 5.3 lti\_config.py

- A simpler Flask app (Pallets, 2025) focusing on LTI login and launch endpoints.
- Stores session launch info.
- Useful for testing LTI flows separately.

### 5.4 LTI Configuration Files

- tool\_config.json and tool\_config\_backend.json define the tool's integration points with Canvas, including OAuth URLs and keys.
- tool\_config\_template.json serves as a deployment starter template.



## 5.5 Frontend Assets (static/ and templates/)

- CSS file styles the upload and recording interface.
- JavaScript handles audio recording and upload asynchronously.
- HTML template renders transcription results or upload UI depending on context.

## 6. Usage Instructions

- Clone the project and install Python dependencies.
- Configure LTI keys and Canvas client IDs.
- Run the Flask app (python app.py).
- Expose your local server via a public URL (ngrok or cloud).
- Configure Canvas to recognize your LTI tool with the public URLs.
- Use the Canvas assignment selection to launch the oral response evaluator.
- Upload or record audio responses.
- Receive transcript and rubric-based evaluation feedback instantly.

## 7. Testing Approach

### Functional Testing

- Verified upload and recording functionality across browsers.
- Ensured compatibility with multiple audio formats: .wav, .mp3, .webm, .m4a, .flac, and .mp4.
- Validated real-time transcription and scoring with sample responses.

### Model Validation

- Used sample speech inputs with known expected transcripts.
- Compared Whisper outputs against manual transcripts for accuracy.

### LTI Integration Testing

- OIDC login tests and LTI 1.3 launches via Canvas LMS Developer Keys yet to be conducted.
- Verified secure launch payloads, deployment IDs, and role parsing yet to be conducted.

### UI Testing

- Ensured form validation, real-time status updates, and responsive rendering across devices.

### Results

- The system achieves highly reliable transcription on clean audio.
- Evaluation scores correctly reflect rubric term presence.
- Real-time recording works in Chromium-based browsers.

## 8. Limitations and Future Work

Despite successful implementation, the project has several limitations and opportunities for expansion:

### 8.1 Current Limitations

- **Model Limitations:** The base FasterWhisper model performs well but may misrecognize speech in low-quality recordings or with heavy accents.
- **Deployment Fragility:** Hosting via Ngrok is not suited for production. A more robust deployment (e.g., on GCP, AWS, or Azure) is required.
- **No Authentication for Frontend Uploads:** Currently, uploads outside Canvas LTI are unauthenticated.

### 8.2 Future Enhancements

- Upgrade to **larger Whisper models** for more accurate transcription.
- Support **CSV/JSON exports** for instructor-gradebooks.
- Secure and scale the app using **Docker, HTTPS, and production-grade databases**.
- Extend LTI scoring logic to **push grades directly to Canvas** using the LTI AGS (Assignment and Grade Services) spec.

## 9. Security Considerations

Given the system interacts with student responses and educational platforms, it adheres to several security principles:

- **File Upload Restrictions:** Only whitelisted audio formats are accepted. Filenames are sanitized using `secure_filename`.
- **Size Limitations:** Flask is configured with a 10MB file size cap to prevent Denial of Service (DoS) via large uploads.
- **Session Protection:** HTTP-only and SameSite cookie settings are enforced to mitigate Cross-Site Request Forgery and Cross-Site Scripting (CSRF/XSS).
- **LTI Security:**
  - The tool authenticates launches via OpenID Connect (OIDC) with JSON Web Key Sets (JWKS).
  - Deployment ID and client validation ensure only known LMS instances can launch the tool.
- **Temporary File Handling:** Uploaded files are saved using tempfile and removed after evaluation.
- **No Persistent Storage of Transcripts or Audio** (currently): All processing is ephemeral unless extended in future builds.

# 10. Appendices

## Appendix A: app.py

Python

```
1 from flask import Flask, request, jsonify, send_file, redirect, session, render_template_string, url_for
2 from pylti3.tool_config import ToolConfJsonFile
3 from pylti3.contrib.flask import FlaskRequest, FlaskCacheDataStorage, FlaskMessageLaunch, FlaskSessionService, FlaskCookieService
4 from pylti3.exception import LtiException
5 from flask_caching import Cache
6 from flask_limiter import Limiter
7 from flask_limiter.util import get_remote_address
8 from functools import wraps
9 from werkzeug.utils import secure_filename
10 from pylti3.oidc_login import OIDCLogin
11 import os
12 import json
13 import tempfile
14
15 from whisper_eval import transcribe_and_evaluate
16
17 print("Working directory:", os.getcwd())
18
19 app = Flask(__name__)
20 app.secret_key = 'your_super_secret_key'
21
22 flask_cache = Cache(app, config={'CACHE_TYPE': 'SimpleCache'})
23 cache = FlaskCacheDataStorage(flask_cache)
24
25 limiter = Limiter(get_remote_address, app=app, default_limits=["10 per minute"])
26
27 app.config.update(
28     SESSION_COOKIE_HTTPONLY=True,
29     SESSION_COOKIE_SAMESITE='Lax',
30     SESSION_COOKIE_SECURE=False,
31     MAX_CONTENT_LENGTH=10 * 1024 * 1024
32 )
33
34 ALLOWED_EXTENSIONS = {'wav', 'mp3', 'm4a', 'mp4', 'flac', 'webm'}
35 UPLOAD_FOLDER = "uploads"
36 os.makedirs(UPLOAD_FOLDER, exist_ok=True)
37
38 LTI_CONFIG_PATH = os.path.join(os.path.dirname(__file__), 'tool_config.json')
39 LTI_KEYS_PATH = os.path.join(os.path.dirname(__file__), 'keys')
40 LTI_ISSUER = "https://canvas.instructure.com"
41
42 tool_config = ToolConfJsonFile("tool_config_backend.json")
43
44 def allowed_file(filename):
45     return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
46
47 def lti_required(f):
48     @wraps(f)
49     def decorated_function(*args, **kwargs):
50         if 'launch_id' not in session:
51             return redirect(url_for('lti_login'))
52         return f(*args, **kwargs)
53     return decorated_function
54
55 @app.route("/lti/login", methods=["GET"])
56 def lti_login():
57     try:
58         flask_request = FlaskRequest(request)
59         session_service = FlaskSessionService(flask_request)
60         cookie_service = FlaskCookieService(flask_request)
61         oidc_login = OIDCLogin(flask_request, tool_config, session_service, cookie_service)
62         return oidc_login.redirect(flask_request)
63     except Exception as e:
64         return f"OIDC login error: {e}", 400
65
66 @app.route("/lti/launch", methods=["POST"])
67 def lti_launch():
68     try:
69         flask_request = FlaskRequest(request)
70         message_launch = FlaskMessageLaunch(flask_request, tool_config, cache).verify()
71         session['launch_id'] = message_launch.get_launch_id()
72         user_info = message_launch.get_launch_data().get("https://purl.imsglobal.org/spec/lti/claim/custom", {})
73         return f"✔ LTI Launch Success! Hello {user_info.get('user_id', 'student')}."
74     except LtiException as e:
75         return f"LTI Launch error: {e}", 400
76
77 @app.route("/keys", methods=["GET"])
78 def jwks():
79     jwk_file = os.path.join(LTI_KEYS_PATH, "jwks.json")
80     if os.path.exists(jwk_file):
81         with open(jwk_file, "r") as f:
82             jwks_data = json.load(f)
```

```

81         with open(jwk_file, "r") as f:
82             jwks_data = json.load(f)
83             return jsonify(jwks_data)
84         else:
85             return jsonify({"error": "JWKS not found"}), 404
86
87 @app.route("/tool_config.json", methods=["GET"])
88 def serve_tool_config():
89     return send_file("tool_config.json", mimetype="application/json")
90
91 @app.route("/")
92 def index():
93     return "## Oral Response Evaluator is running!"
94
95 @limiter.limit("3 per minute")
96 @app.route("/transcribe", methods=["POST"])
97 def transcribe():
98     if "file" not in request.files:
99         return jsonify({"error": "No file uploaded"}), 400
100
101     file = request.files["file"]
102     if file.filename == "":
103         return jsonify({"error": "Empty filename"}), 400
104
105     if not allowed_file(file.filename):
106         return jsonify({"error": "Invalid file type"}), 400
107
108     filename = secure_filename(file.filename)
109     file_path = os.path.join(UPLOAD_FOLDER, filename)
110     file.save(file_path)
111
112     try:
113         result = transcribe_and_evaluate(file_path)
114         return jsonify(result)
115     except Exception as e:
116         return jsonify({"error": str(e)}), 500
117
118 UPLOAD_FORM = """
119 <!doctype html>
120 <html>
121 <head>
122 <title>Upload or Record Audio</title>

```

---

```

122 <title>Upload or Record Audio</title>
123 <style>
124     body {
125         font-family: Arial, sans-serif;
126         background: #f5f5f5;
127         display: flex;
128         justify-content: center;
129         align-items: center;
130         height: 100vh;
131         margin: 0;
132     }
133     .upload-container {
134         background: white;
135         padding: 30px;
136         border-radius: 12px;
137         box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
138         text-align: center;
139         max-width: 400px;
140         width: 100%;
141     }
142     input[type="file"] {
143         margin-bottom: 15px;
144     }
145     button, input[type="submit"] {
146         padding: 10px 20px;
147         margin: 10px;
148         font-size: 16px;
149         border: none;
150         border-radius: 6px;
151         background-color: #3498db;
152         color: white;
153         cursor: pointer;
154     }
155     button:disabled {
156         background-color: #ccc;
157     }
158     #status {
159         margin-top: 10px;
160         color: #555;
161     }
162 </style>

```

```

162 </style>
163 </head>
164 <body>
165 <div class="upload-container">
166 <h2>Upload or Record Audio</h2>
167 <form method="post" enctype="multipart/form-data">
168 <input type="file" name="audio_file" required><br>
169 <input type="submit" value="Upload">
170 </form>
171 <hr>
172 <h3>Or Record Now</h3>
173 <button id="recordBtn">Start Recording</button>
174 <button id="stopBtn" disabled>Stop Recording</button>
175 <p id="status"></p>
176 <form id="recordForm" method="post" enctype="multipart/form-data" style="display:none;">
177 <input type="file" name="audio_file" id="recordedAudio">
178 <input type="submit" value="Submit Recording">
179 </form>
180 </div>
181 <script>
182 let mediaRecorder;
183 let chunks = [];
184
185 recordBtn.onclick = async () => {
186   chunks = [];
187   const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
188   mediaRecorder = new MediaRecorder(stream);
189   mediaRecorder.ondataavailable = e => chunks.push(e.data);
190   mediaRecorder.onstop = () => {
191     const blob = new Blob(chunks, { type: 'audio/webm' });
192     const file = new File([blob], 'recording.webm', { type: 'audio/webm' });
193     const dataTransfer = new DataTransfer();
194     dataTransfer.items.add(file);
195     recordedAudio.files = dataTransfer.files;
196     recordForm.style.display = "block";
197     status.innerText = "Recording ready. Click submit.";
198   };
199   mediaRecorder.start();
200   recordBtn.disabled = true;
201   stopBtn.disabled = false;
202   status.innerText = "Recording...";

```

Normal text file length: 10,340 lines: 340 Ln: 102 Col: 28 Pos: 3,734 Windows (CR LF) UTF-8 INS

```

202   status.innerText = "Recording...";
203 };
204
205 stopBtn.onclick = () => {
206   mediaRecorder.stop();
207   recordBtn.disabled = false;
208   stopBtn.disabled = true;
209 };
210 </script>
211 </body>
212 </html>
213 """"
214
215 RESULTS_PAGE = """"
216 <!doctype html>
217 <html>
218 <head>
219 <title>Transcription Results</title>
220 <style>
221   body {
222     font-family: Arial, sans-serif;
223     background: #f3f4f6;
224     margin: 0;
225     padding: 40px 20px;
226     display: flex;
227     justify-content: center;
228     align-items: flex-start;
229     min-height: 100vh;
230   }
231   .container {
232     background: white;
233     border-radius: 12px;
234     padding: 30px;
235     max-width: 700px;
236     width: 100%;
237     box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
238   }
239   h1, h2 {
240     color: #2c3e50;
241     margin-bottom: 15px;
242

```

Normal text file length: 10,340 lines: 340 Ln: 102 Col: 28 Pos: 3,734 Windows (CR LF) UTF-8 INS

```

241     margin-bottom: 15px;
242 }
243 p {
244     background: #f0f0f0;
245     padding: 15px;
246     border-radius: 8px;
247     font-family: 'Courier New', monospace;
248     color: #333;
249 }
250 ul {
251     list-style: none;
252     padding-left: 0;
253 }
254 li {
255     background: #eaf3ff;
256     margin-bottom: 8px;
257     padding: 10px;
258     border-left: 5px solid #3498db;
259     border-radius: 6px;
260 }
261 .score {
262     font-weight: bold;
263     color: #27ae60;
264 }
265 .button-link {
266     display: inline-block;
267     margin-top: 20px;
268     padding: 10px 20px;
269     background-color: #4CAF50;
270     color: white;
271     text-decoration: none;
272     border-radius: 6px;
273     transition: background-color 0.3s ease;
274 }
275 .button-link:hover {
276     background-color: #45a049;
277 }
278 @media (max-width: 600px) {
279     .container {
280         padding: 20px;
281     }
282 }

```

Normal text file length: 10,340 lines: 340 Ln: 102 Col: 28 Pos: 3,734 Windows (CR LF) UTF-8 INS

```

282 }
283 </style>
284 </head>
285 <body>
286 <div class="container">
287 <h1> 🎙️ Transcription</h1>
288 <p>{{ transcript }}</p>
289 <h2> 📊 Evaluation</h2>
290 <ul>
291 <li><strong>Word Count:</strong> {{ evaluation.word_count }}</li>
292 <li><strong>Contains "important":</strong> {{ evaluation.contains_keyword }}</li>
293 {% if evaluation.rubric_score is defined %}
294 <li class="score">Rubric Score: {{ evaluation.rubric_score }} / {{ total_rubric }}</li>
295 <li><strong>Feedback:</strong></li>
296 <ul>
297     {% for fb in evaluation.rubric_feedback %}
298     <li>{{ fb }}</li>
299     {% endfor %}
300 </ul>
301 {% endif %}
302 </ul>
303 <a class="button-link" href="{{ url_for('upload_audio') }}">🔍 Upload Another</a>
304 </div>
305 </body>
306 </html>
307 """
308
309 @app.route("/upload", methods=["GET", "POST"])
310 def upload_audio():
311     if request.method == "POST":
312         if 'audio_file' not in request.files:
313             return "No audio file part", 400
314         file = request.files['audio_file']
315         filename = secure_filename(file.filename) if file.filename else 'recording.webm'
316         if not allowed_file(filename):
317             return "Invalid file type", 400
318         with tempfile.NamedTemporaryFile(delete=False, suffix=os.path.splitext(filename)[1]) as tmp:
319             file.save(tmp.name)
320             temp_path = tmp.name
321         try:
322             result = transcribe_and_evaluate(temp_path)

```

Normal text file length: 10,340 lines: 340 Ln: 102 Col: 28 Pos: 3,734 Windows (CR LF) UTF-8 INS



```
322         result = transcribe_and_evaluate(temp_path)
323     finally:
324         os.remove(temp_path)
325     evaluation = result.get("evaluation", {})
326     if "score" in evaluation and "feedback" in evaluation:
327         evaluation["rubric_score"] = evaluation["score"]
328         evaluation["rubric_feedback"] = evaluation["feedback"]
329     return render_template_string(
330         RESULTS_PAGE,
331         transcript=result.get("transcript", ""),
332         evaluation=evaluation,
333         total_rubric=4
334     )
335 else:
336     return UPLOAD_FORM
337
338 if __name__ == "__main__":
339     app.run(debug=False, port=5000)
340
```

Normal text file

length: 10,340 lines: 340

Ln: 102 Col: 28 Pos: 3,734

Windows (CR LF)

UTF-8

INS

## Appendix B: whisper\_eval.py

### Python

```
1 from faster_whisper import WhisperModel
2 import os
3
4 class WhisperTranscriber:
5     def __init__(self, model_size="base", device="cpu", compute_type="int8"):
6         print("Loading Whisper model...")
7         self.model = WhisperModel(model_size, device=device, compute_type=compute_type)
8         print("Model loaded.")
9
10    def transcribe(self, audio_path):
11        if not os.path.exists(audio_path):
12            raise FileNotFoundError(f"Audio file not found: {audio_path}")
13
14        segments, _ = self.model.transcribe(audio_path)
15        full_text = " ".join([segment.text.strip() for segment in segments])
16        return full_text.strip(), segments
17
18
19 class SimpleEvaluator:
20     def __init__(self, rubric_keywords=None):
21         self.rubric_keywords = rubric_keywords or ["clarity", "organization", "relevance", "grammar"]
22
23     def evaluate(self, transcript):
24         score = 0
25         feedback = []
26
27         for keyword in self.rubric_keywords:
28             if keyword.lower() in transcript.lower():
29                 score += 1
30                 feedback.append(f"✓ Mentioned: {keyword}")
31             else:
32                 feedback.append(f"✗ Missing: {keyword}")
33
34         return {
35             "score": score,
36             "total": len(self.rubric_keywords),
37             "feedback": feedback
38         }
39
40
41 def transcribe_and_evaluate(audio_path):
42     """
43     Combines transcription and evaluation into a single call.
44
45     Returns:
46     {
47         "transcript": <str>,
48         "evaluation": {
49             "word_count": <int>,
50             "contains_keyword": <bool>,
51             "rubric_score": <int>,
52             "rubric_feedback": <list of str>
53         }
54     }
55     """
56     transcriber = WhisperTranscriber()
57     transcript, _ = transcriber.transcribe(audio_path)
58
59     evaluator = SimpleEvaluator()
60     rubric_result = evaluator.evaluate(transcript)
61
62     result = {
63         "transcript": transcript,
64         "evaluation": {
65             "word_count": len(transcript.split()),
66             "contains_keyword": "important" in transcript.lower(),
67             "rubric_score": rubric_result["score"],
68             "rubric_feedback": rubric_result["feedback"]
69         }
70     }
71
72     return result
73
```

Normal text file | length: 2,297 | lines: 73 | Ln: 73 | Col: 1 | Pos: 2,298 | Windows (CR LF) | UTF-8 | INS

```
42     """
43     Combines transcription and evaluation into a single call.
44
45     Returns:
46     {
47         "transcript": <str>,
48         "evaluation": {
49             "word_count": <int>,
50             "contains_keyword": <bool>,
51             "rubric_score": <int>,
52             "rubric_feedback": <list of str>
53         }
54     }
55     """
56     transcriber = WhisperTranscriber()
57     transcript, _ = transcriber.transcribe(audio_path)
58
59     evaluator = SimpleEvaluator()
60     rubric_result = evaluator.evaluate(transcript)
61
62     result = {
63         "transcript": transcript,
64         "evaluation": {
65             "word_count": len(transcript.split()),
66             "contains_keyword": "important" in transcript.lower(),
67             "rubric_score": rubric_result["score"],
68             "rubric_feedback": rubric_result["feedback"]
69         }
70     }
71
72     return result
73
```

Normal text file | length: 2,297 | lines: 73 | Ln: 73 | Col: 1 | Pos: 2,298 | Windows (CR LF) | UTF-8 | INS

## Appendix C: lti\_config.py

python

```
1 from flask import Flask, request, redirect, session, jsonify
2 from pylti3.contrib.flask import FlaskRequest
3 from pylti3.tool_config import ToolConfJsonFile
4 from pylti3.session import SessionDataStorage
5 from pylti3.message_launch import MessageLaunch
6
7 app = Flask(__name__)
8 app.secret_key = 'your_secret_key' # Replace with a strong secret key
9
10 # Load tool configuration
11 tool_config = ToolConfJsonFile('tool_config_backend.json')
12 launch_data_storage = SessionDataStorage(session)
13
14 @app.route('/lti/login', methods=['GET'])
15 def login():
16     flask_request = FlaskRequest(request)
17     oidc_login = flask_request.get_oidc_login(tool_config)
18     return oidc_login.enable_check_cookies() \
19         .redirect(flask_request.get_param('target_link_uri'))
20
21 @app.route('/lti/launch', methods=['POST'])
22 def launch():
23     flask_request = FlaskRequest(request)
24     message_launch = MessageLaunch(flask_request, tool_config, launch_data_storage)
25     launch_data = message_launch.get_launch_data()
26     session['launch_id'] = message_launch.get_launch_id()
27     return jsonify({
28         "message": "LTI Launch Successful!",
29         "user": launch_data.get("name"),
30         "roles": launch_data.get("https://purl.imsglobal.org/spec/lti/claim/roles"),
31         "context": launch_data.get("https://purl.imsglobal.org/spec/lti/claim/context")
32     })
33
34 @app.route('/lti/session', methods=['GET'])
35 def get_session():
36     if 'launch_id' not in session:
37         return jsonify({"error": "No active launch session"}), 401
38     return jsonify({"launch_id": session['launch_id']})
39
40 if __name__ == '__main__':
41     app.run(debug=True)
42
```

Normal text file length: 1,649 lines: 42 Ln: 42 Col: 1 Pos: 1,650 Windows (CR LF) UTF-8 INS

## Appendix D: tool\_config.json

json

```
1 {
2     "title": "Oral Response Evaluator",
3     "scopes": [
4         "https://purl.imsglobal.org/spec/lti-ags/scope/lineitem",
5         "https://purl.imsglobal.org/spec/lti-ags/scope/result.readonly",
6         "https://purl.imsglobal.org/spec/lti-ags/scope/score"
7     ],
8     "extensions": [
9         {
10             "platform": "canvas.instructure.com",
11             "settings": {
12                 "platform": "canvas.instructure.com",
13                 "placements": [
14                     {
15                         "placement": "assignment_selection",
16                         "message_type": "LtiResourceLinkRequest",
17                         "target_link_uri": "https://f2a7-143-132-2-2.ngrok-free.app/lti/launch"
18                     }
19                 ]
20             },
21             "privacy_level": "LtiPrivacyLevel.Public"
22         }
23     ],
24     "public_jwk_url": "https://f2a7-143-132-2-2.ngrok-free.app/keys",
25     "target_link_uri": "https://f2a7-143-132-2-2.ngrok-free.app/lti/launch",
26     "oidc_initiation_url": "https://f2a7-143-132-2-2.ngrok-free.app/lti/login"
27 }
```

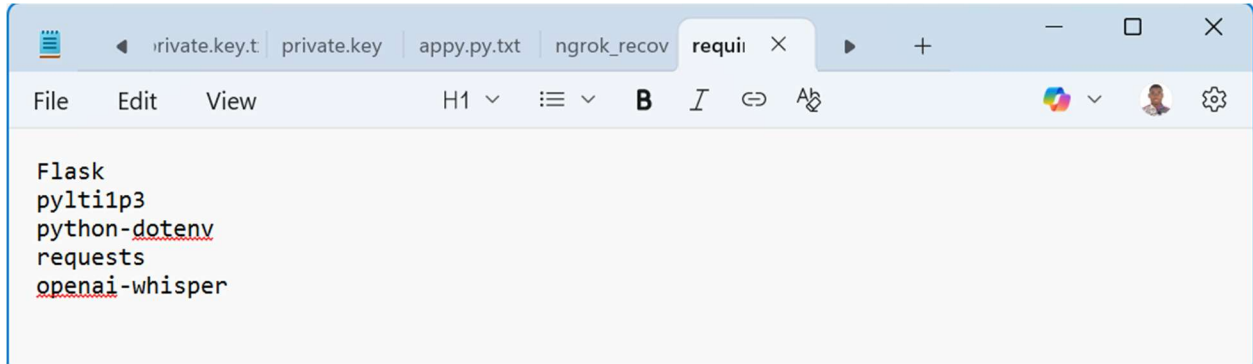
json

## Appendix F: tool\_config\_template.json

json

Normal text file	length: 944 lines: 28	Ln: 28 Col: 1 Pos: 945	Windows (CR LF)	UTF-8	INS
------------------	-----------------------	------------------------	-----------------	-------	-----

## Appendix G: requirements.txt



A screenshot of a web browser window with multiple tabs. The active tab is titled 'requi' and shows a text editor with the following content:

```
Flask
pylti1p3
python-dotenv
requests
openai-whisper
```

The browser's address bar shows several tabs: 'private.key.t', 'private.key', 'appy.py.txt', 'ngrok\_recov', and 'requi'. The browser interface includes a menu bar with 'File', 'Edit', and 'View', and a toolbar with various icons for font settings and editing.

## Appendix H: jwk\_public.json

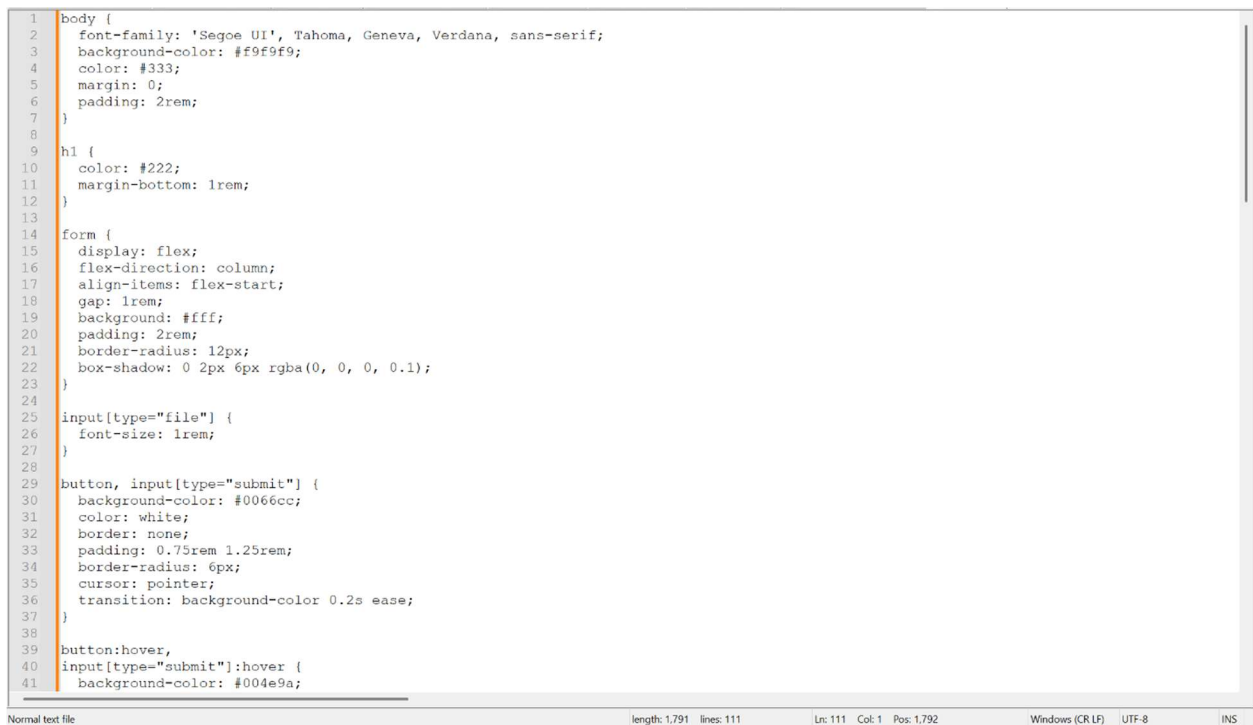
json



A screenshot showing a single line of JSON text: `"lnV"`. The text is highlighted in blue, and the cursor is positioned at the end of the string.

## Appendix I: CSS (static/css/style.css)

CSS



A screenshot of a code editor showing CSS code for a file named 'style.css'. The code is as follows:

```
1 body {
2   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
3   background-color: #f9f9f9;
4   color: #333;
5   margin: 0;
6   padding: 2rem;
7 }
8
9 h1 {
10  color: #222;
11  margin-bottom: 1rem;
12 }
13
14 form {
15  display: flex;
16  flex-direction: column;
17  align-items: flex-start;
18  gap: 1rem;
19  background: #fff;
20  padding: 2rem;
21  border-radius: 12px;
22  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
23 }
24
25 input[type="file"] {
26  font-size: 1rem;
27 }
28
29 button, input[type="submit"] {
30  background-color: #0066cc;
31  color: white;
32  border: none;
33  padding: 0.75rem 1.25rem;
34  border-radius: 6px;
35  cursor: pointer;
36  transition: background-color 0.2s ease;
37 }
38
39 button:hover,
40 input[type="submit"]:hover {
41  background-color: #004e9a;
```

The code editor has a status bar at the bottom showing 'Normal text file', 'length: 1,791 lines: 111', 'Ln: 111 Col: 1 Pos: 1,792', 'Windows (CR LF)', 'UTF-8', and 'INS'.

## Appendix J: JavaScript (static/js/main.js)

js

```
1 document.addEventListener('DOMContentLoaded', function () {
2   const form = document.querySelector('form');
3   const fileInput = document.querySelector('input[type="file"]');
4   const submitBtn = form.querySelector('button[type="submit"]');
5
6   const recordBtn = document.getElementById('recordBtn');
7   const stopBtn = document.getElementById('stopBtn');
8   const status = document.getElementById('recordingStatus');
9
10  let mediaRecorder;
11  let audioChunks = [];
12
13  // Spinner setup
14  let spinnerContainer = document.getElementById('loading-spinner');
15  if (!spinnerContainer) {
16    spinnerContainer = document.createElement('div');
17    spinnerContainer.id = 'loading-spinner';
18    spinnerContainer.style.display = 'none';
19    spinnerContainer.style.textAlign = 'center';
20    spinnerContainer.style.marginTop = '15px';
21    spinnerContainer.innerHTML = `
22      
23      <p>Transcribing... Please wait.</p>
24    `;
25    form.appendChild(spinnerContainer);
26  }
27
28  // Form file upload handler
29  if (form && fileInput && submitBtn) {
30    form.addEventListener('submit', function (e) {
31      if (!fileInput.files.length) {
32        e.preventDefault();
33        alert('Please select a file before submitting.');
```

## Appendix K: HTML Template (templates/index.html)

html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Oral Response Evaluator</title>
7   <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
8 </head>
9 <body>
10  <div class="container">
11    <h1> Oral Response Evaluator</h1>
12
13    {% if transcript %}
14    <section class="results">
15      <h2>Transcription</h2>
16      <p class="transcript">{{ transcript }}</p>
17
18      <h2>Evaluation</h2>
19      <ul class="evaluation">
20        <li><strong>Word Count:</strong> {{ evaluation.word_count }}</li>
21        <li><strong>Contains "important":</strong> {{ evaluation.contains_keyword }}</li>
22
23        {% if evaluation.rubric score is defined %}
24        <li><strong>Rubric Score:</strong> {{ evaluation.rubric_score }} / {{ total_rubric }}</li>
25        <li><strong>Feedback:</strong>
26          <ul class="feedback">
27            {% for fb in evaluation.rubric_feedback %}
28              <li>{{ fb }}</li>
29            {% endfor %}
30          </ul>
31        </li>
32        {% endif %}
33      </ul>
34
35      <a class="button" href="{{ url_for('upload_audio') }}">Upload another file</a>
36    </section>
37    {% else %}
38    <section class="upload-form">
```