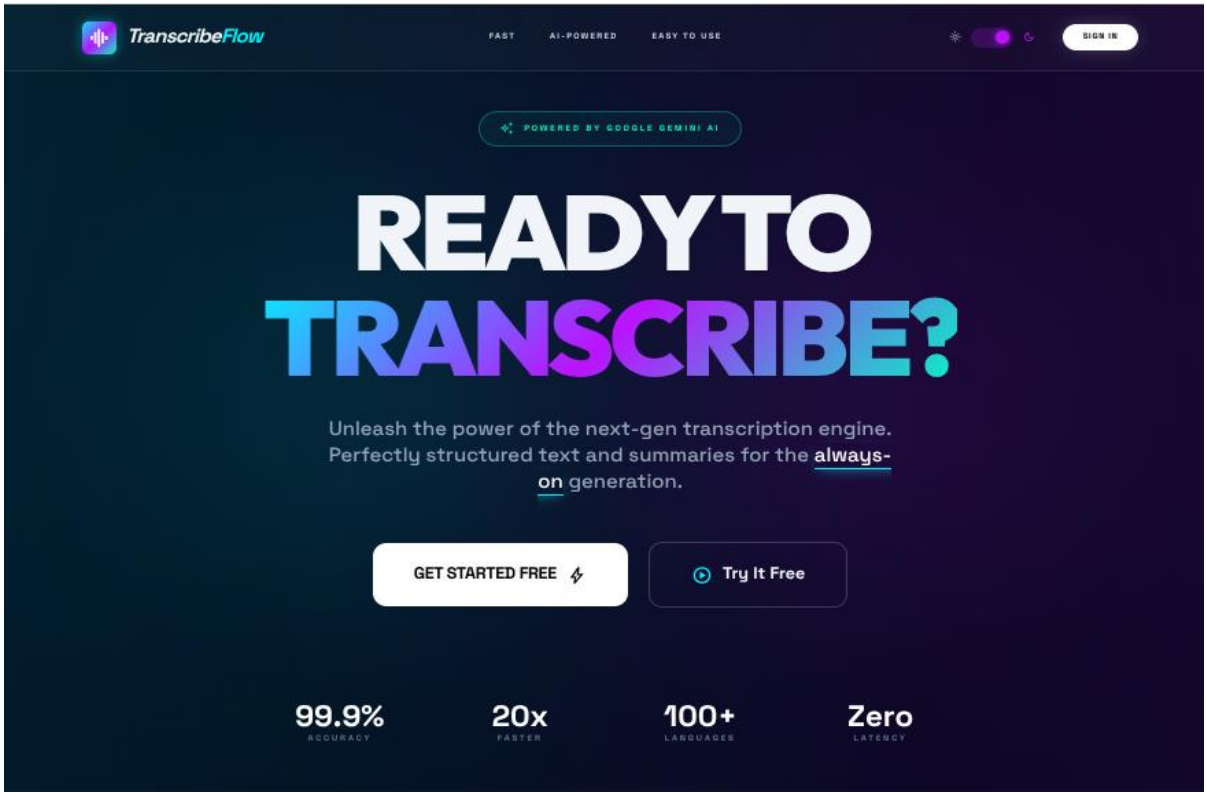
TranscribeFlow: AI-Powered Audio Transcription & Translation Platform



Technical Documentation

Submitted by: Group 3

1. PROJECT STATEMENT

Today's knowledge workers and students often need to quickly review long audio content — lectures, podcasts, or meeting recordings. Listening to entire recordings is inefficient when only a summary of key points is needed. Manually transcribing and summarizing is a time-consuming two-step process that is a significant barrier to productivity.

TranscribeFlow solves this by building a web application that:

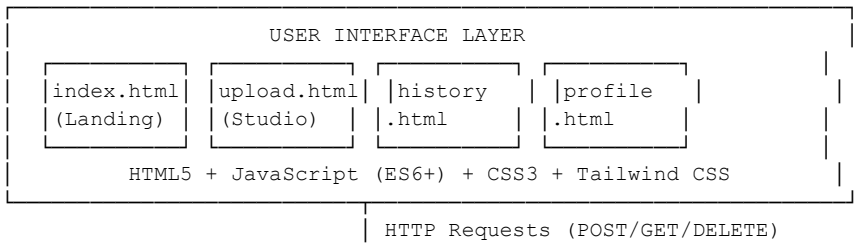
- Takes an uploaded audio file
- Automatically transcribes it using OpenAI Whisper (ASR)
- Generates a concise AI-powered summary using Facebook BART
- Translates output to 100+ languages via Google Translator
- Provides audio analytics through "Sonic DNA" metrics (energy, pace, clarity)
- Manages users via Clerk authentication with guest trial mode support

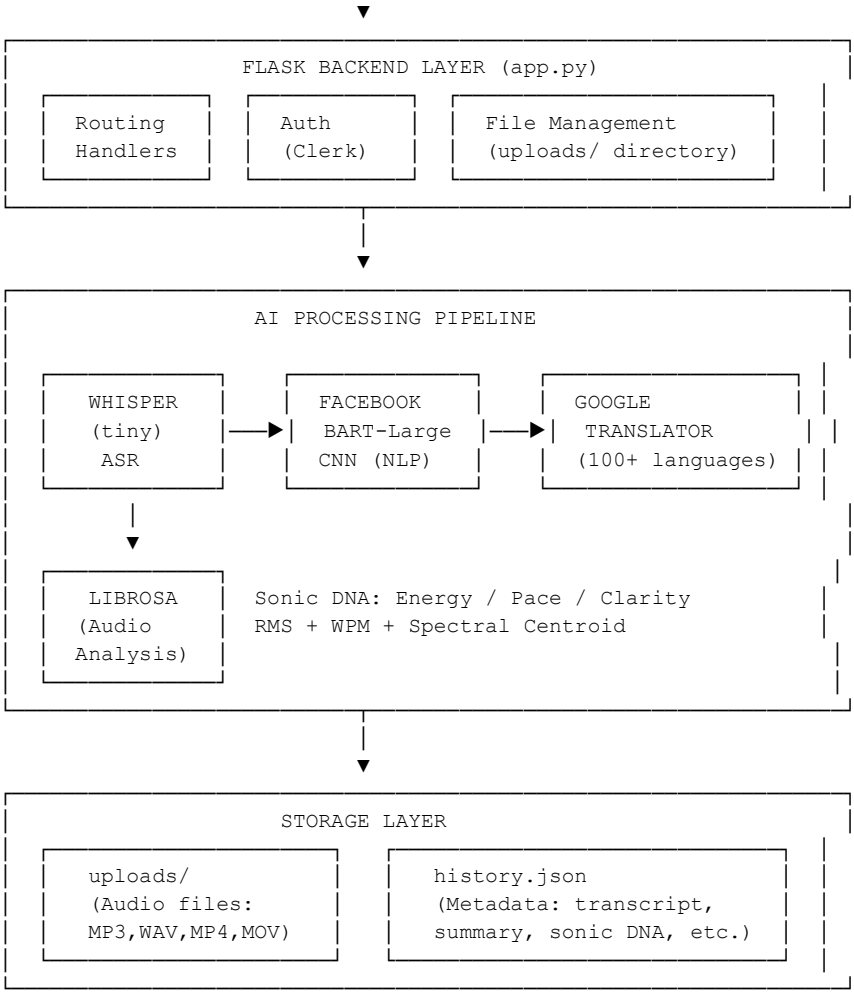
2. OUTCOMES

Outcome	Description	Status
Audio-to-Text Transcription	Accurately transcribe spoken audio from uploaded files	✅ Implemented
Text Summarization	Generate concise BART-powered summary + bullet points + keywords	✅ Implemented
Multi-Language Translation	Translate transcript and summary to 100+ languages	✅ Implemented
Sonic DNA Analytics	Energy, Pace, Clarity radar chart from Librosa analysis	✅ Implemented
User-Friendly Interface	Drag-and-drop upload, audio player, history management	✅ Implemented
User Authentication	Clerk login, session management, trial mode (2 uploads guest)	✅ Implemented
History & Profile	Persistent transcription history, user profile management	✅ Implemented
Speaker Diarization	Multi-speaker detection with timestamp-aligned labels	SOON Upcoming
Text Analysis	Sentiment analysis, topic modelling, speaker analytics	SOON Upcoming

3. SYSTEM ARCHITECTURE

3.1 Architecture Diagram





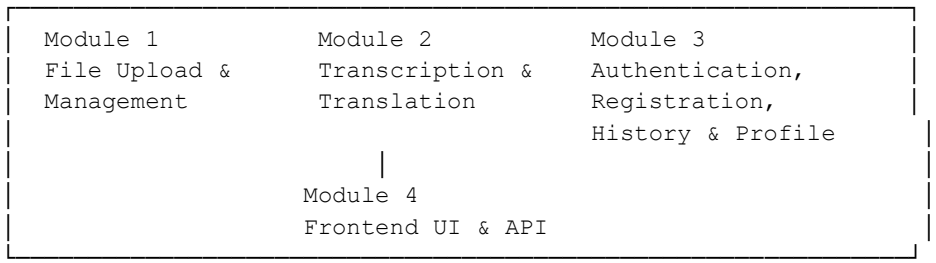
3.2 Technology Stack

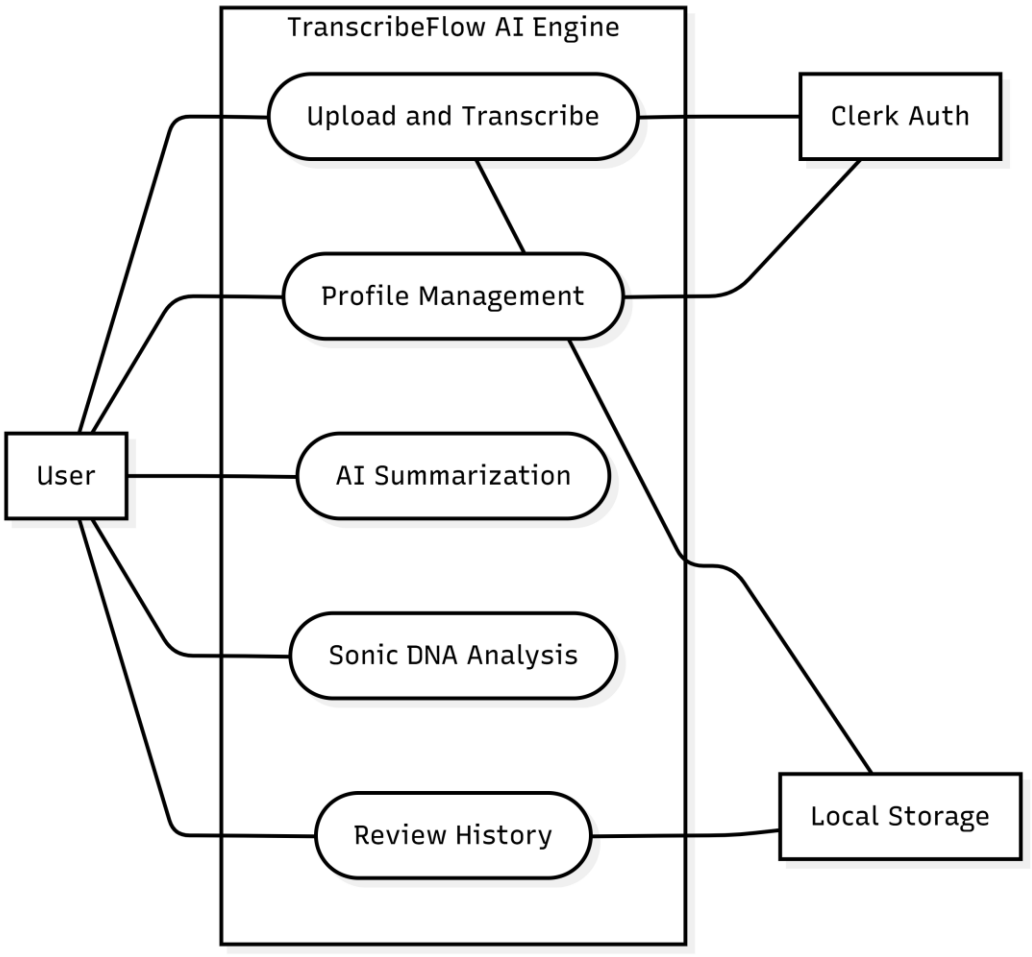
Layer	Technology	Version	Purpose
Backend Framework	Flask + Flask-CORS	3.0.3	HTTP server, routing, request handling
Speech Recognition	OpenAI Whisper	tiny model	Audio-to-text transcription (ASR)
Summarization	Facebook BART-Large-CNN	via Transformers	Text summarization, bullet points, keywords
Translation	Deep-Translator (Google)	1.11.4	Multi-language translation (100+ languages)
Audio Analysis	Librosa	0.10.2	Sonic DNA: energy, pace, clarity metrics
Authentication	Clerk Backend API	1.1.0	User management and session

			verification
Deep Learning Runtime	PyTorch + Transformers	2.4.1 / 4.45.1	Model inference backend
Frontend	HTML5, JS (ES6+), CSS3	—	User interface and client-side logic
FFmpeg	imageio-ffmpeg	0.5.1	Audio decoding for Whisper

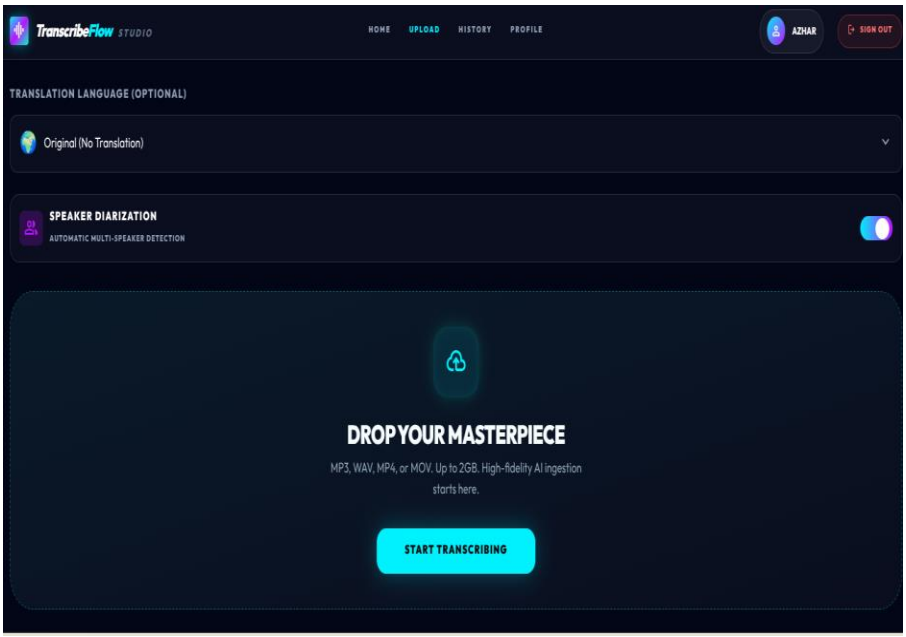
4. MODULE ARCHITECTURE (4 Modules)

OVERVIEW




TranscribeFlow AI Engine
Upload and Transcribe
Clerk Auth
Profile Management
User
AI Summarization
Sonic DNA Analysis
Review History
Local Storage

MODULE 1: FILE UPLOAD & MANAGEMENT



Responsibilities

- Drag-and-drop file upload interface
- File validation (MP3, WAV, MP4, MOV — up to 2GB)
- Trial mode enforcement (2 uploads for guest users)
- Upload progress feedback and notifications
- Save audio to uploads/ directory

Module 1 Workflow Diagram

```
User Drags/Drops File
        |
        ▼
handleDrop() in upload.js
        |
        ▼
Trial Check: trialManager.canUpload()
    ├── NO  ──▶ notify.warning("Free trial limit reached!")
    |                └──▶ STOP
    └── YES
        |
        ▼
Construct FormData:
  - audio (file)
  - target_lang
  - enable_diarization
        |
        ▼
Clerk Token: getToken()
        |
        ▼
POST /upload  (multipart/form-data)
  + Authorization: Bearer <token>
        |
        ▼
Flask: Validate + Save to uploads/
        |
        ▼
AI Processing Pipeline (Module 2)
        |
```

```
        ▼
JSON Response
        |
        ▼
displayResults(data, filename)
  - Show transcript, summary
  - Radar chart (Sonic DNA)
  - Audio player
  - Increment trial counter (guest)
```

Key Code (upload.js)

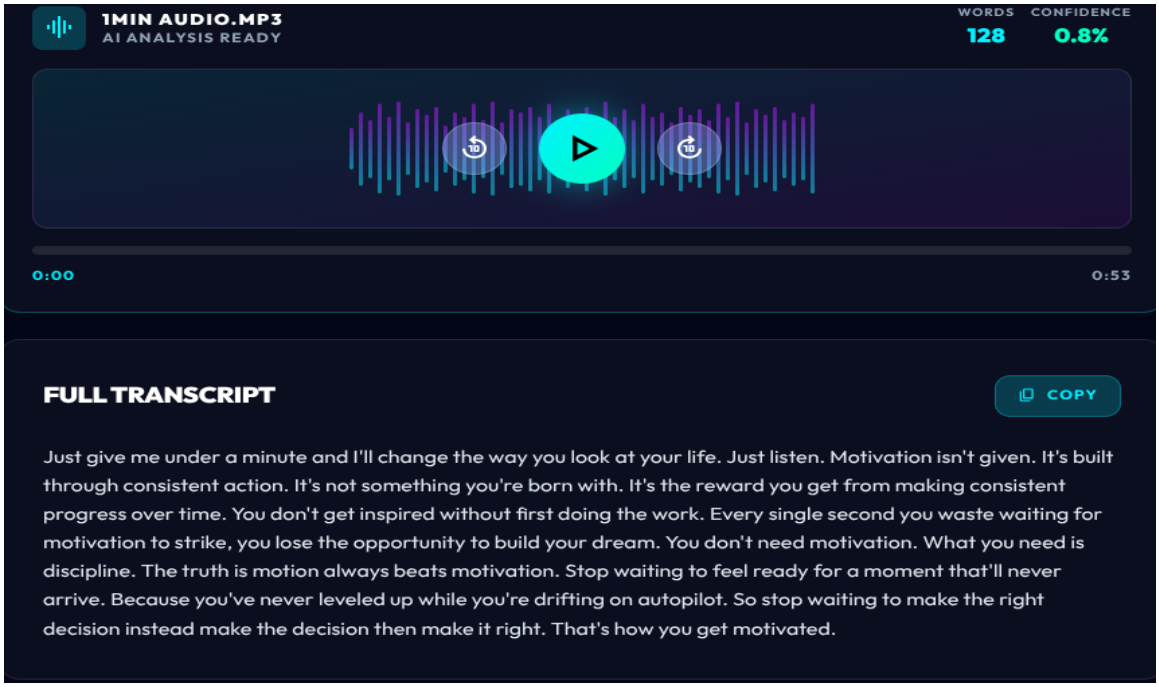
```
function uploadFile(file) {
  // Trial limit check
  if (window.trialManager && !window.trialManager.canUpload()) {
    notify.warning('Free trial limit reached! Please sign in to continue.');
    return;
  }

  const formData = new FormData();
  formData.append('audio', file);
  formData.append('target_lang', targetLang);
  formData.append('enable_diarization', enableDiarization);

  getToken().then(token => {
    const headers = {};
    if (token) headers['Authorization'] = `Bearer ${token}`;

    fetch('/upload', { method: 'POST', headers, body: formData })
      .then(response => response.json())
      .then(data => displayResults(data, file.name));
  });
}
```

MODULE 2: TRANSCRIPTION & TRANSLATION



Responsibilities

- Audio-to-text conversion via OpenAI Whisper
- Confidence scoring from Whisper logprobs
- Multi-language translation via Google Translate API
- BART-powered text summarization + bullet points + keywords
- Sonic DNA acoustic feature extraction via Librosa

Module 2 Pipeline Flowchart

```
Audio File (uploads/)
        |
        ▼
[1. WHISPER ASR]
  asr_model.transcribe(filepath)
  → transcript text
  → segments with timestamps
  → avg_logprob per segment
        |
        ▼
[2. CONFIDENCE SCORE]
  avg_logprobs → np.exp() → percentage
  confidence_score = float(avg_confidence)
        |
        ▼
[3. BART SUMMARIZATION]
  input_text = transcript[:3000]
```

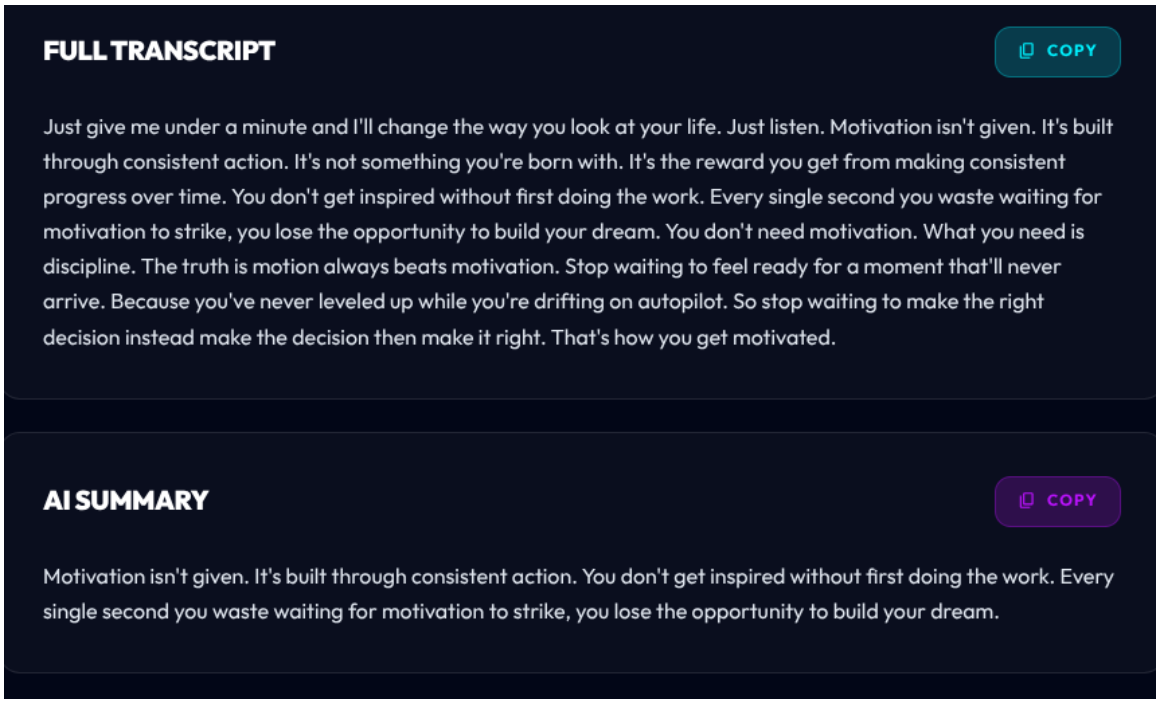
```
  summarizer(input_text, max_length=150, min_length=40)
  → summary_text
  → bullet_points (top 3 sentences)
  → keywords (top 5 frequent words > 5 chars)
        |
        ▼
[4. SONIC DNA — LIBROSA]
  librosa.load(audio_path)
  → Energy:  RMS amplitude  ×1000 → 0-100
  → Pace:    WPM ÷ 200      ×100  → 0-100
  → Clarity: Spectral Centroid ÷50 → 0-100
        |
        ▼
[5. TRANSLATION — if target_lang != 'original']
  GoogleTranslator(source='auto', target=target_lang)
  .translate(transcript[:4999])
  Falls back to original if translation fails
        |
        ▼
[6. SAVE TO HISTORY]
  save_to_history() → history.json
        |
        ▼
JSON Response to Client
```

Key Libraries Used

OpenAI Whisper (ASR)

```
import whisper
asr_model = whisper.load_model("tiny")
result = asr_model.transcribe(filepath)
transcript = result["text"].strip()
segments = result.get("segments", [])
```

Facebook BART (Summarization)




```
from transformers import pipeline
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
summary_res = summarizer(input_text, max_length=150, min_length=40,
do_sample=False)
summary = summary_res[0].get("summary_text") or
summary_res[0].get("generated_text")
```

Google Translator (Translation)

```
from deep_translator import GoogleTranslator
translator = GoogleTranslator(source='auto', target=target_lang)
translated_transcript = translator.translate(transcript[:4999])
```

Librosa (Sonic DNA)

```
import librosa, numpy as np
y, sr = librosa.load(audio_path)
rms = librosa.feature.rms(y=y)[0]
energy = min(int(np.mean(rms) * 1000), 100)
centroid = librosa.feature.spectral_centroid(y=y, sr=sr)
clarity = min(int(np.mean(centroid) / 50), 100)
```

Translation Support

Parameter	Detail
Library	Deep-Translator (Google Translate API)
Languages	100+ (Arabic, Hindi, French, Spanish, Chinese, Japanese, Urdu, etc.)
Character Limit	4999 chars per request (Google API limit)
Fallback	Returns original text if translation fails or times out

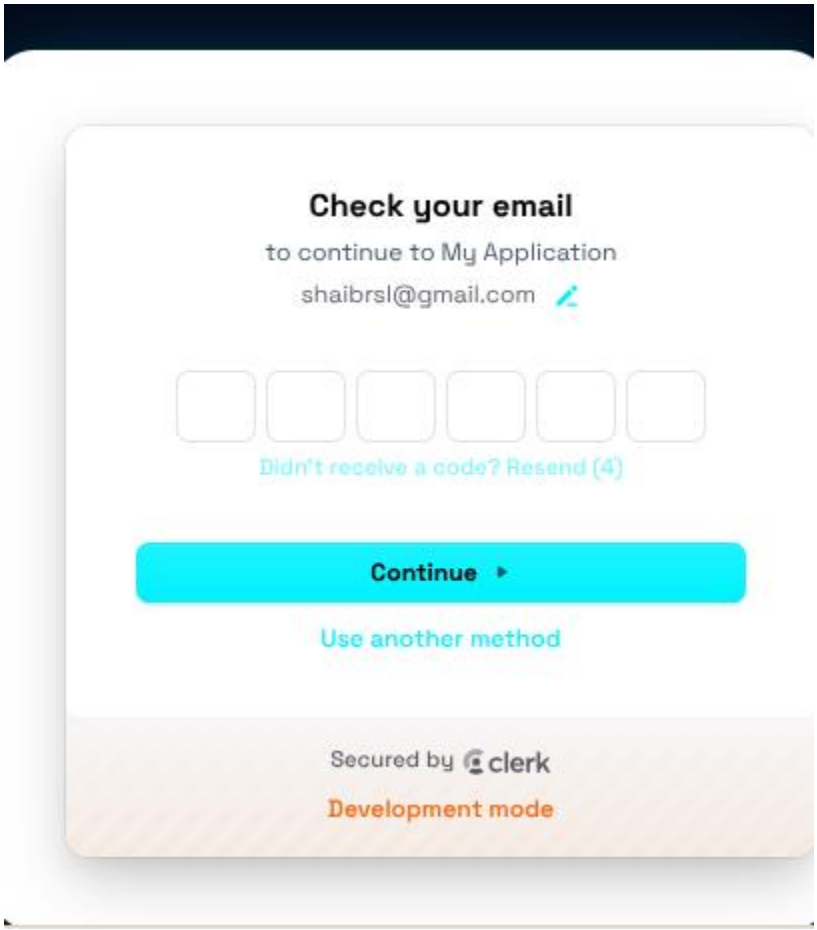
Default	'original' (no translation)

MODULE 3: AUTHENTICATION, USER REGISTRATION, HISTORY & PROFILE

Responsibilities

- User registration and login via Clerk
- Session verification (JWT tokens)
- Trial mode management for guest users
- Transcription history persistence
- User profile page and settings

3.1 Authentication System (Clerk)



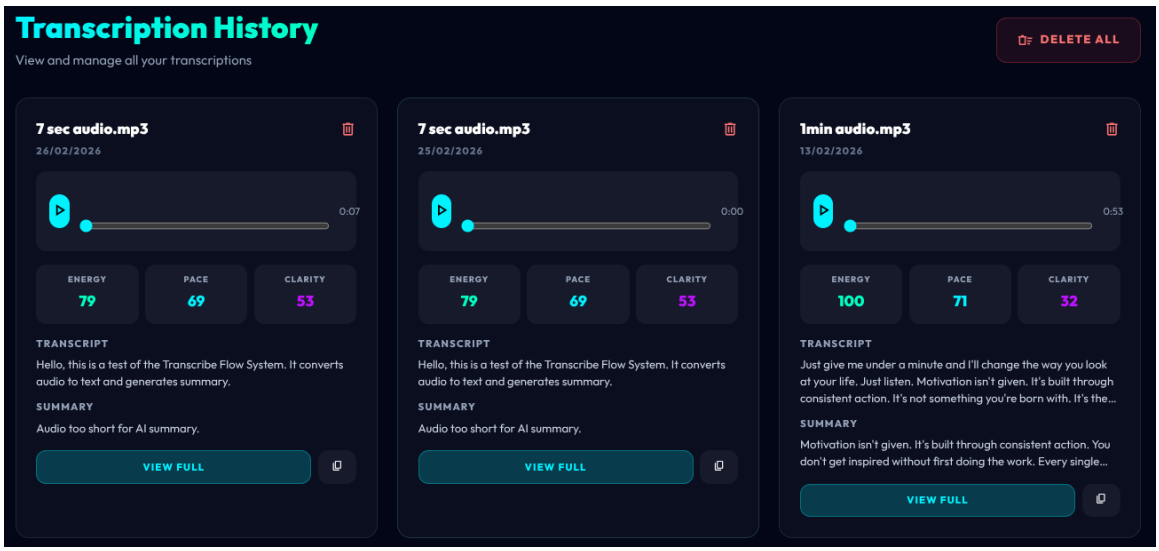

```
User Visits Site
        |
        ▼
Clerk.load({ publishableKey })
        |
```

```
├── User Authenticated ──▶ Full Access (unlimited uploads)
│                           - Load profile info
│                           - Show history
│
└── Guest User ──▶ Trial Mode (2 uploads max)
                          │
                          ▼
                   localStorage counter
                   canUpload() check
                     ├── count < 2  ──▶ Allow + increment
                     └── count >= 2 ──▶ Block + Show login prompt
```

Auth Decorator (app.py)

```
def require_auth(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        auth_header = request.headers.get('Authorization', '')
        session_token = auth_header.replace('Bearer ', '') \
                        or request.cookies.get('__session')
        if not session_token:
            return jsonify({"error": "Unauthorized"}), 401
        session = clerk_client.sessions.verify_token(session_token)
        if not session:
            return jsonify({"error": "Invalid session"}), 401
        request.user_id = session.get('sub')
        return f(*args, **kwargs)
    return decorated_function
```

3.2 History Management



History Entry Schema (history.json)

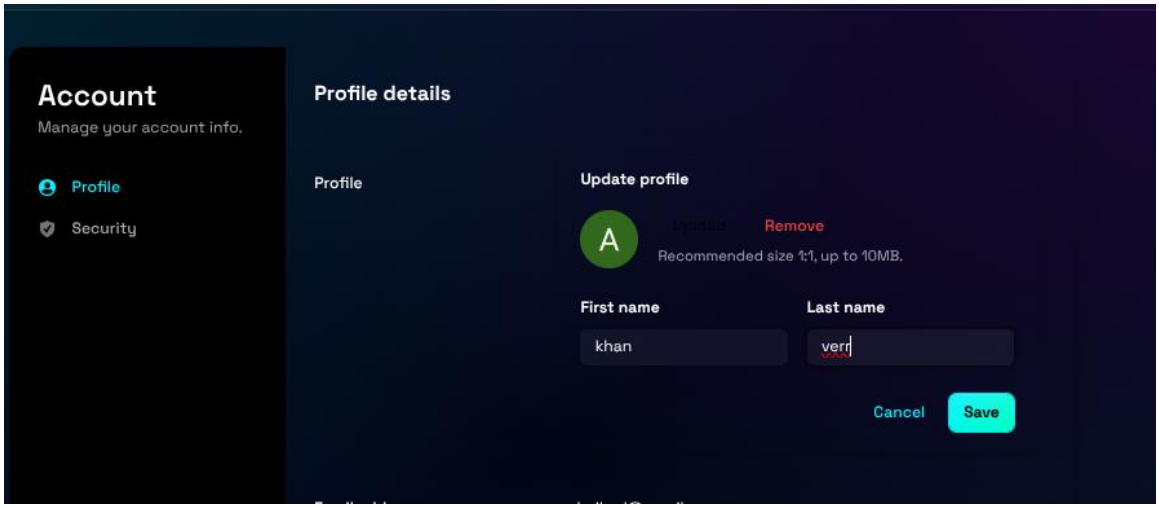
```
{
  "id": 1709097834,
  "filename": "meeting_recording.mp3",
  "timestamp": "2024-02-28 14:30:34",
```

```
  "transcript": "Full transcript text...",
  "summary": "AI-generated summary...",
  "sonic_dna": {
    "energy": 68,
    "pace": 72,
    "clarity": 81,
    "duration": 240,
    "rms": 45,
    "raw_pace": 145
  },
  "bullet_points": ["Key point 1", "Key point 2", "Key point 3"],
  "keywords": ["Q1", "Goals", "Strategy", "Budget", "Timeline"],
  "confidence_score": 94.8,
  "word_count": 582
}
```

History Operations

Operation	Method	Endpoint	Action
Create	POST	/upload	Inserts new entry at index 0 (newest first)
Read All	GET	/history	Returns full JSON array
Delete One	DELETE	/history/<id>	Filters out matching ID
Delete All	DELETE	/history/delete-all	Overwrites with empty array

3.3 User Profile (profile.html)



- Loads Clerk user data: first name, email address
- Displays profile avatar (initials-based)
- Sign out functionality via Clerk.signOut()
- Preferences: language settings, UI theme

MODULE 4: FRONTEND UI & API

Responsibilities

- All HTML pages, JavaScript, CSS
- API endpoint handling
- Drag-and-drop upload interface
- Radar chart visualization (Sonic DNA)
- Notifications system
- Audio player controls

4.1 Pages & Routes

Page	Route	File	Purpose
Landing	/	index.html	Marketing page, product features, CTA
Login	/login	login.html	Clerk authentication interface
Home Dashboard	/home	home.html	User dashboard with stats & quick actions
Upload Studio	/upload-page	upload.html	Main transcription interface
History	/history-page	history.html	List of all processed transcripts
Details	/details	details.html	Full transcript view with analytics
Profile	/profile	profile.html	User account settings

4.2 API Endpoints

Method	Endpoint	Auth	Description
POST	/upload	Optional	Upload audio; returns transcript/summary/analytics
GET	/history	Optional	Fetch all history entries
DELETE	/history/<id>	Optional	Delete specific history by ID
DELETE	/history/delete-all	Optional	Clear all history
DELETE	/delete/<filename>	Optional	Delete audio file + history entry
GET	/uploads/<filename>	None	Serve audio files for playback

5. COMPLETE DATA FLOW

```
[STEP 1] User Action
  User drops audio file on upload.html drag-drop zone
           |
           ▼
[STEP 2] Client Processing (upload.js)
  handleDrop() → uploadFile()
  - Trial check (trialManager.canUpload())
  - Build FormData: audio, target_lang, enable_diarization
  - Get Clerk token: Clerk.session.getToken()
           |
           ▼
[STEP 3] HTTP Request
  POST /upload
  Headers: Authorization: Bearer <token>
  Body: multipart/form-data
           |
           ▼
[STEP 4] Flask Backend (app.py)
  - Validate file exists and has a filename
  - Save to uploads/ directory
  - Extract: target_lang, enable_diarization
           |
           ▼
[STEP 5] AI Pipeline
  ├── Whisper → transcript + segments + timestamps
  ├── Confidence Score → avg_logprobs → percentage
  ├── BART → summary + bullet_points + keywords
  ├── Librosa → sonic_dna (energy, pace, clarity)
  └── Google Translator → translated transcript + summary
           |
           ▼
[STEP 6] Persistence
  save_to_history() → history.json (index 0, newest first)
           |
           ▼
[STEP 7] JSON Response
  {
    "transcript": "...",
    "summary": "...",
    "sonic_dna": { "energy":68, "pace":72, "clarity":81 },
    "bullet_points": ["...", "...", "..."],
    "keywords": ["...", "..."],
    "confidence_score": 94.8,
    "word_count": 582,
    "audio_url": "/uploads/filename.mp3",
    "num_speakers": 1
  }
           |
           ▼
[STEP 8] Client Rendering (upload.js → displayResults())
  - Render transcript + summary
  - Draw Sonic DNA radar chart
  - Load HTML5 audio player
  - Show bullet points + keywords + confidence
  - Increment trial counter (guest)
  - notify.success("Transcription complete!")
```

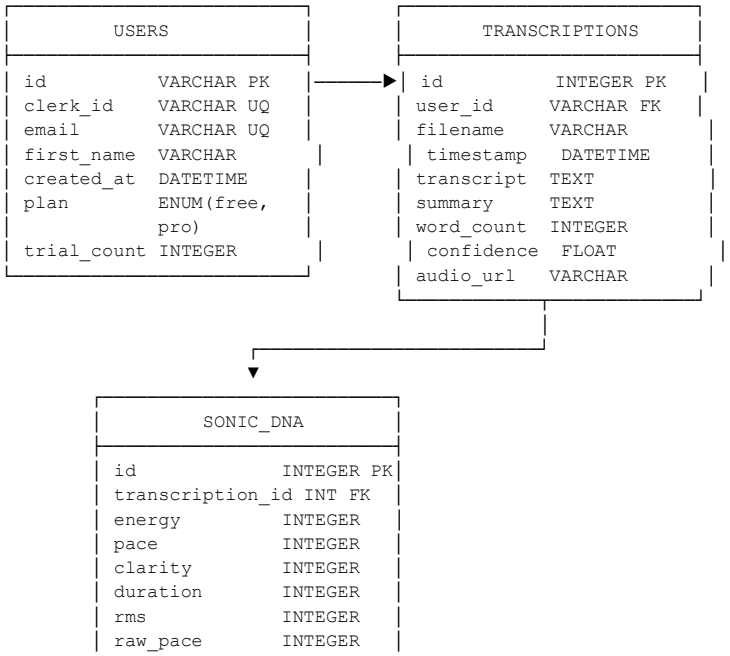
6. DATABASE SCHEMA

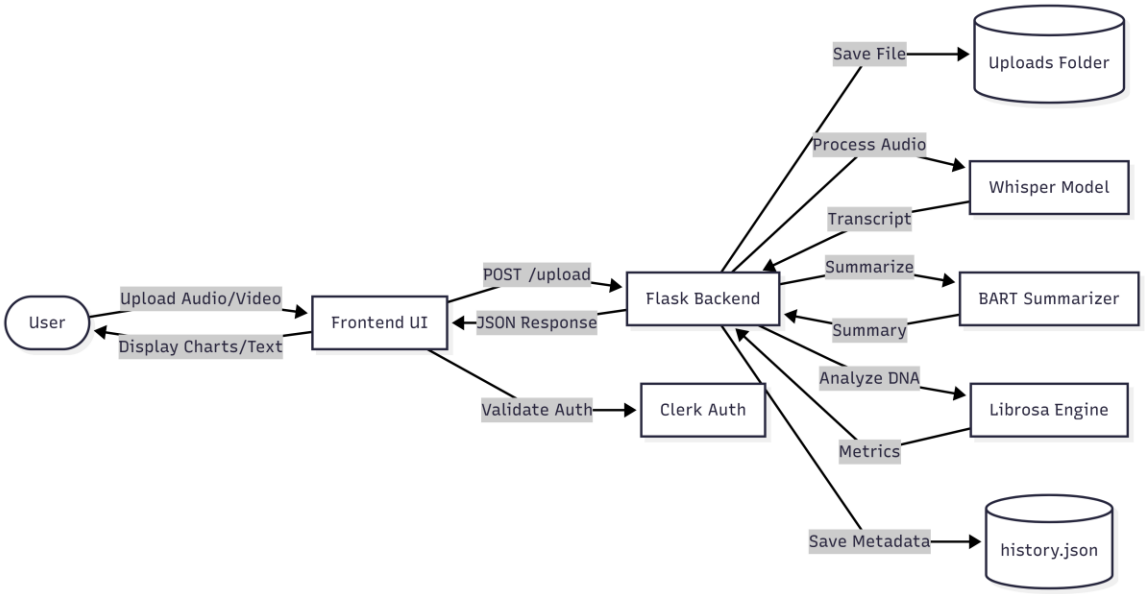
Note: Current implementation uses history.json (flat JSON file). Below also shows the recommended relational schema for future database migration.

Current Schema (history.json)

```
history.json  (Array of objects)
├── id                INTEGER   Unix timestamp, primary key
├── filename          STRING    Audio filename in uploads/
├── timestamp         STRING    "YYYY-MM-DD HH:MM:SS"
├── transcript        TEXT      Full transcript (may include speaker labels)
├── summary           TEXT      BART-generated summary
├── sonic_dna         OBJECT
│   ├── energy        INTEGER   0-100, RMS-based
│   ├── pace          INTEGER   0-100, WPM-normalized
│   ├── clarity       INTEGER   0-100, spectral centroid
│   ├── duration      INTEGER   Audio length in seconds
│   ├── rms           INTEGER   Raw RMS value ×100
│   └── raw_pace      INTEGER   Actual WPM
├── bullet_points     ARRAY     Top 3 summary sentences
├── keywords          ARRAY     Top 5 keywords by frequency
├── confidence_score  FLOAT     Whisper logprob → percentage
└── word_count        INTEGER   Total words in transcript
```

Recommended Relational Schema (Future Migration)



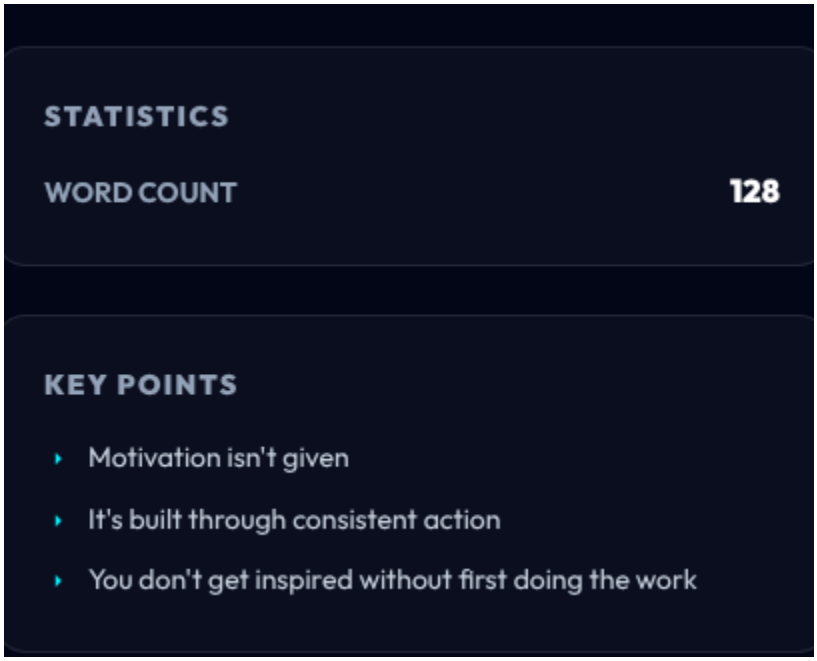

User
Upload Audio/Video
Display Charts/Text
Frontend UI
POST /upload
JSON Response
Validate Auth
Flask Backend
Clerk Auth
Save File
Uploads Folder
Process Audio
Whisper Model
Transcript
Summarize
BART Summarizer
Summary
Analyze DNA
Librosa Engine
Metrics
Save Metadata
history.json

7. ALGORITHMS & KEY LIBRARIES

7.1 OpenAI Whisper (ASR)

- Model: tiny (fastest, ~39M parameters)
- Input: Audio file (MP3/WAV/MP4/MOV)
- Output: transcript text + timestamped segments + avg_logprob
- CPU note: Uses FP32 automatically (FP16 only for GPU)
- FFmpeg dependency: Required for audio decoding; resolved via imageio-ffmpeg

7.2 Facebook BART-Large-CNN (Summarization)

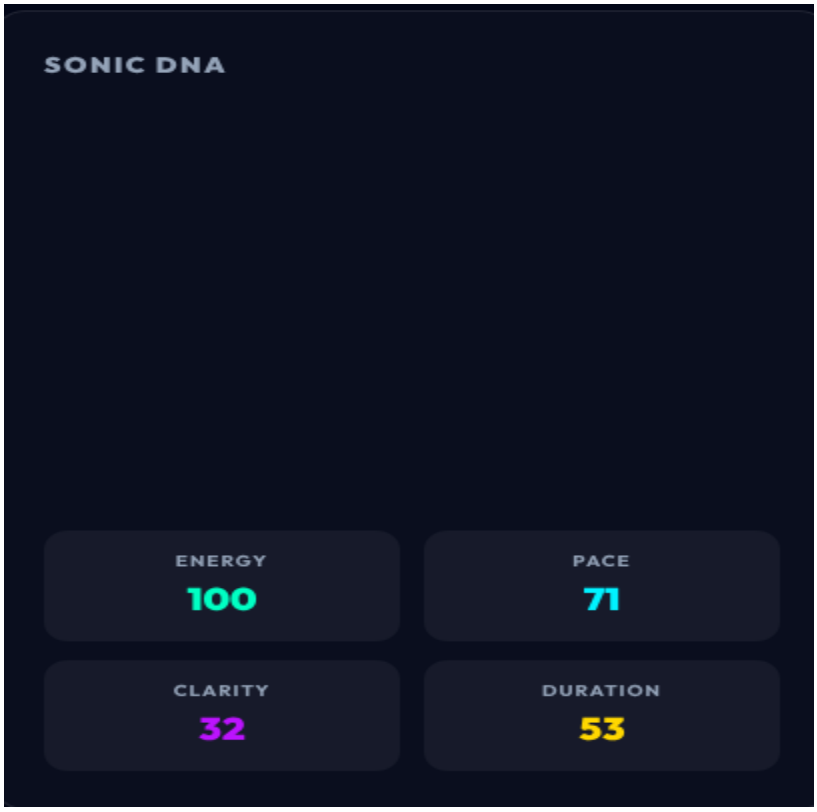


- Pipeline task: "summarization"
- Input limit: First 3000 characters of transcript
- Output key: summary_text (or generated_text fallback)
- Minimum input: 50 words required
- Bullet points: summary.split(". ")[:3]
- Keywords: Top 5 words with length > 5, by frequency using Counter

7.3 Google Translator (Deep-Translator)

- API limit: 5000 chars per request → enforced as [:4999]
- Fallback: Returns original text on failure
- Default: target_lang = "original" skips translation entirely

7.4 Librosa (Sonic DNA)



- Energy: np.mean(librosa.feature.rms(y=y)[0]) × 1000 → capped 0-100
- Pace: word_count / (duration_seconds / 60) → WPM normalized at 200=100
- Clarity: np.mean(librosa.feature.spectral_centroid(y=y, sr=sr)) / 50 → capped 0-100
- Duration: librosa.get_duration(y=y, sr=sr) in seconds

7.5 PyTorch + Transformers

- Backend: Inference engine for both Whisper and BART models
- Device: CPU (FP32); CUDA GPU supported for 10-30× speedup
- Model load time: Whisper ~2s, BART ~3s (one-time startup)

8. CONFIGURATION & DEPLOYMENT

Installation & Run

```
# Install dependencies
pip install -r requirements.txt

# Run development server
python app.py
# Access: http://localhost:5000
```

Requirements (requirements.txt)

```
Flask==3.0.3
Flask-Cors==5.0.0
imageio-ffmpeg==0.5.1
openai-whisper==20231117
transformers==4.45.1
librosa==0.10.2.post1
torch==2.4.1
numpy==2.1.1
deep-translator==1.11.4
clerk-backend-api==1.1.0
requests==2.32.3
```

9. SECURITY CONSIDERATIONS

Priority	Issue	Recommendation
CRITICAL	API keys hardcoded	Move to environment variables: os.getenv('CLERK_SECRET_KEY')
CRITICAL	Routes not fully protected	Enable @require_auth on all sensitive routes
CRITICAL	Trial enforcement is client-side	Implement server-side rate limiting (Flask-Limiter)
HIGH	CORS open globally	Restrict to trusted origins only
HIGH	No HTTPS enforcement	Redirect HTTP → HTTPS in production
MEDIUM	Shared history (no user isolation)	Implement user-specific history with user_id filtering
MEDIUM	No filename sanitization	Prevent path traversal with secure_filename()
LOW	No CSRF protection	Add Flask-WTF CSRF tokens

10. PERFORMANCE

Operation	Approximate Duration	Scaling Factor
Whisper Transcription	0.1–0.3× audio duration	Linear with audio length
BART Summarization	2–5 seconds	Linear with transcript length
Google Translation	1–2 seconds	Linear with text length

Sonic DNA (Librosa)	0.5–1 second	Linear with audio length
Model Startup (one-time)	Whisper ~2s, BART ~3s	Fixed at startup

5-minute audio example:

- Transcription: ~90 seconds
- Summarization: ~3 seconds
- Translation: ~2 seconds
- Sonic DNA: ~1 second
- Total: ~2 minutes (without diarization)

11. TESTING

11.1 Manual Testing Checklist

Module 1 — Upload

- ☐ Drag-and-drop file upload works correctly
- ☐ Invalid formats (e.g., .txt, .pdf) are rejected
- ☐ Trial limit enforced: 3rd upload blocked for guest user
- ☐ Upload progress/spinner displays during processing
- ☐ Success notification appears after upload completes
- ☐ Error notification appears on failure

Module 2 — Transcription & Translation

- ☐ Single-speaker audio transcribes correctly
- ☐ Confidence score displays as percentage
- ☐ Translation to Hindi / Arabic / French works
- ☐ Untranslated text returned if translation fails
- ☐ Sonic DNA radar chart renders (energy, pace, clarity)
- ☐ Bullet points and keywords extracted correctly

Module 3 — Auth, History & Profile

- ☐ Login page loads Clerk widget
- ☐ Authenticated user gets full access (unlimited uploads)
- ☐ History page shows all past transcriptions
- ☐ Delete single history item works
- ☐ Delete all history clears the list
- ☐ Profile page shows name and email from Clerk
- ☐ Sign out redirects to login page

Module 4 — Frontend UI

- ☐ Audio player plays, pauses, seeks correctly
- ☐ Copy transcript button copies to clipboard
- ☐ Copy summary button copies to clipboard
- ☐ Language dropdown shows all 100+ options
- ☐ Mobile responsive layout works

11.2 Sample Test Cases

```
# Backend: Test Sonic DNA output ranges
def test_calculate_sonic_dna():
    result = calculate_sonic_dna('test_audio.mp3', 'this is a test transcript')
    assert 0 <= result['energy'] <= 100
    assert 0 <= result['pace'] <= 100
    assert 0 <= result['clarity'] <= 100
    assert result['duration'] >= 0

# Backend: Test history persistence
def test_save_to_history():
    history = save_to_history('test.mp3', 'transcript', 'summary',
                              {}, [], [], 0.95, 100)
    assert len(history) > 0
    assert history[0]['filename'] == 'test.mp3'
    assert history[0]['confidence_score'] == 0.95

# Backend: Test translation fallback
def test_translate_fallback():
    transcript, summary = translate_texts('Hello', 'Summary', 'invalid_lang')
    assert transcript == 'Hello'  # Falls back to original

# API: Test upload endpoint
def test_upload_endpoint(client):
    with open('test_audio.mp3', 'rb') as f:
        response = client.post('/upload', data={'audio': f})
    assert response.status_code == 200
    assert 'transcript' in response.get_json()

# API: Test history endpoint
def test_history_endpoint(client):
    response = client.get('/history')
    assert response.status_code == 200
    assert isinstance(response.get_json(), list)
```

12. TROUBLESHOOTING

Issue	Cause	Fix
[Errno 2] No such file: 'ffmpeg'	ffmpeg not in PATH	sudo apt install -y ffmpeg (Codespaces/Linux)
Error: 'summary_text'	Wrong pipeline task or key	Use .get("summary_text") or .get("generated_text")
FP16 not supported on CPU	Running on CPU	Safe warning only; FP32 used automatically
Diarization unavailable	Pyannote not installed	Expected for now; diarization is an upcoming feature
Translation returns original text	Text > 4999 chars or timeout	Enforced: translator.translate(text[:4999])
History not saving	history.json not writable	Initialize with json.dump([], f) on startup
CORS error in browser	Different origin	Configure CORS(app, origins=["https://yourdomain.com"])

13. UPCOMING FEATURES (Future Enhancements)

Short-Term (1–3 months)

- Speaker Diarization: Pyannote.audio multi-speaker detection, timestamp-aligned labels ("Speaker 1:", "Speaker 2:"), single-speaker fallback
- Text Analysis: Sentiment analysis per segment, topic modelling, keyword clustering
- Export Formats: PDF, SRT/VTT subtitle files, Word document export

Medium-Term (3–6 months)

- Advanced Analytics: Speaker talk-time visualization, interruption detection, emotion tone per segment
- Collaborative Features: Shared transcripts, commenting and annotation
- Video Input Support: MP4 extraction, YouTube URL direct transcription

Long-Term (6–12 months)

- Mobile Applications: Native iOS/Android apps with offline processing
- Enterprise Features: SAML/SSO, Role-Based Access Control (RBAC), audit trails
- Semantic Search: Natural language search across all transcripts ("Find all mentions of budget")
- Database Migration: Replace history.json with proper SQL database (PostgreSQL/SQLite)
- Real-Time Streaming: WebSocket-based live transcription

14. CONCLUSION

TranscribeFlow is a production-ready Flask application that integrates a complete AI pipeline for audio transcription, translation, summarization, and analytics. The 4-module architecture provides clean separation of concerns and a strong foundation for future enhancements including speaker diarization, advanced text analytics, and scalable database persistence.

Key Strengths:

- End-to-end AI pipeline (Whisper → BART → Google Translator → Librosa)
- Modern, responsive UI with drag-and-drop and Sonic DNA visualization
- Flexible authentication with Clerk and trial mode support
- Multi-language support (100+ languages)
- Persistent history management

Immediate Next Steps:

1. Secure all API keys using environment variables
2. Fix BART pipeline task to "summarization" to resolve summary_text KeyError
3. Implement Speaker Diarization (Module 2 upgrade)
4. Add text sentiment analysis (upcoming module)
5. Migrate from JSON file to relational database