**Requirements Document: Translation Evaluation Tool for Arabic–Urdu Broadcast Translation**

**📌 Objective**

The goal of this component is to **evaluate and refine the Arabic-to-Urdu text translation system** used in a real-time broadcast translation pipeline. The tool is intended to:

* Accept text input from a real-time stream or pre-recorded file.
* Translate it from Arabic to Urdu (text-to-text).
* Generate timestamped transcripts.
* Allow human evaluators to review, approve, or edit translations.
* Persist approved/edited translations as **ground truth data** for future model retraining.

**1. 📥 Input Specifications**

**1.1 Input Source**

* Accepts input in two modes:
  + **Streaming mode**: Live transcription or subtitle feed in Arabic.
  + **File mode**: Pre-recorded text files in Arabic, ideally in .srt, .vtt, or .json with timestamped segments.

**1.2 Input Format**

* Arabic text with associated time windows.
* Expected format (JSON example):

json

CopyEdit

[

{

"start\_time": "00:00:01.000",

"end\_time": "00:00:04.000",

"text": "مرحبا بكم في الأخبار"

},

...

]

**2. 🔁 Translation Processing**

**2.1 Translation Model**

* Text is passed through an Arabic-to-Urdu translation model (not part of this tool, assumed to be available).
* Output is returned in the same format, aligned to the input timestamps.

**2.2 Output Format**

* Urdu translation per segment with timestamps.

json

CopyEdit

[

{

"start\_time": "00:00:01.000",

"end\_time": "00:00:04.000",

"original\_text": "مرحبا بكم في الأخبار",

"translated\_text": "خبروں میں خوش آمدید"

},

...

]

**3. 🧪 Evaluation & Feedback Interface**

**3.1 User Interface**

* A simple web-based interface showing:
  + Original Arabic segment.
  + Automatically translated Urdu segment.
  + Editable Urdu field.
  + Timestamp.
  + Approve / Edit / Reject options.

**3.2 Features**

* Editable translation field.
* Navigation across time windows.
* Status indicators for each segment (Pending / Approved / Edited / Rejected).
* "Save All" or auto-save functionality.

**4. ✅ Feedback and Ground Truth Storage**

**4.1 Versioned Edits**

* Each human edit must be saved alongside:
  + Original translation (model output).
  + Final approved or edited translation.
  + Timestamp and editor ID (if applicable).

**4.2 Ground Truth Format**

json

CopyEdit

[

{

"start\_time": "00:00:01.000",

"end\_time": "00:00:04.000",

"original\_text": "مرحبا بكم في الأخبار",

"translated\_text": "خبروں میں خوش آمدید",

"approved\_translation": "خبروں میں خوش آمدید",

"status": "approved",

"edited\_by": "user123",

"edited\_at": "2025-07-17T10:35:00Z"

},

...

]

**4.3 Export Options**

* Ground truth can be exported in JSON or CSV.
* Option to push ground truth to centralized training dataset repository.

**5. 📊 Reporting & Metrics**

**5.1 Translation Quality Metrics (Optional)**

* BLEU / TER scores if reference translations are available.
* Word error rate and alignment mismatch for flagged segments.

**5.2 Reviewer Feedback Stats**

* Number of segments approved, edited, or rejected.
* Average edit time per segment.
* Reviewer-wise breakdown.

**6. ⚙️ System Requirements**

**6.1 Performance**

* Low-latency updates for human-in-the-loop review.
* Support for large file sizes (up to 1 hour of content).
* Real-time syncing in streaming mode.

**6.2 Technology Stack (Suggested)**

* Backend: Python (FastAPI or Flask)
* Frontend: React or simple HTML/JS for rapid development
* Storage: SQLite / PostgreSQL / flat JSON storage
* Optional: WebSocket support for live feedback during streaming

**7. 🔐 Security & Access Control**

* Basic authentication for reviewers.
* Role-based access if multiple users involved (admin, editor, viewer).
* Audit logs for all changes.