Voice for how India actually talks

Rime · Inter IIT Tech Meet 15.0 problem statement

About Rime

Rime builds speech models that learn from how people actually talk, not from studio narration. Rime makes these models for voice agents and other products that speak. This problem statement is about Indian voice technology in general, not only about customer-service agents.

Dates. The organizers announce the dates of all milestones on Discord: mid-term, freeze, evaluation window, end-term and final presentation.

Choosing your track. Tell the organizers on Discord which track your team enters. Mentors for that track then join your team's group. Within one week after the release of this problem statement, also confirm these items on Discord:

- What you will build and how you will measure it, in two or three sentences. In Track 1, name your lead part.
- In Track 1, a baseline for each part, with its first scores on your own development data.
- In Track 1, your Part 1 language.

A judge approves your baselines on Discord. If several teams plan the same approach for their lead part, the judges tell them and can suggest alternatives. A team does not have to change its approach.

Open by design. This problem statement cannot settle every choice in advance. Where it is silent, use your judgment and explain your choice in your report. The judges score your process and what you learned, not only your numbers.

Your choices are binding after you confirm them.

- You cannot change your track.
- You can change your Part 1 language or your lead part only before the mid-term submission, and only with a judge's approval on Discord. After a language change, every language must still have at least two teams.
- You can replace an approved baseline only with a stronger eligible baseline, before the freeze, and only with a judge's approval.

Support. Discord is the main support channel. Ask questions at any time. Judges answer within two working days.

Two tracks

Each team enters one of two tracks and tells the organizers on Discord. The two tracks use the same rules, scoring rubric and deliverables. Only the tasks and the results measures are different. All teams from both tracks are ranked together, and there is one overall winner.

Track	What you build
1. Multilingual Indian Voice System	One voice system in four connected parts: data for a low-resource language, cross-lingual voice cloning, reading real agent text, and an automatic judge
2. Recursive Self-Improving Speech System	A speech captioner and a voice generator that improve each other on Hindi and Tamil speech

In the shared sections, a "part" is one of the four parts of Track 1. Part numbers, the Part 1 language and the lead part apply only to Track 1. Track 2 is scored only as its own section says. It has no lead part, so both novelty criteria in Scoring judge the whole Track 2 system.

Track 1: Multilingual Indian Voice System

The problem

Indian voice technology still fails in basic ways:

- Most Indian languages have little clean, transcribed speech for training.
- A cloned voice usually cannot speak the other languages of the person.
- Voice agents misread the text that Indian users actually send and see: Hinglish, amounts, IDs and names.
- There is no reliable way to measure the quality of Indian TTS without rooms full of human raters.

These failures are connected. A voice for India must speak the language of the listener and sound like the same person in every language. It must read its text correctly. It must also be possible to measure the voice without a room of raters. This problem statement asks each team to build that voice as one system, and to do real speech research during the work.

The challenge

Build one multilingual Indian voice system with four connected parts.

Part	What you build	Stage of the speech pipeline
1. Data from almost nothing	An automated pipeline that turns scattered, messy audio into training data for an underrepresented Indian language	Before training
2. My voice, my languages	One system that takes a 30-second clip in which a person speaks Hindi, Tamil or English. The system then speaks Hindi, Tamil, English and your Part 1 language in the voice of that person	The model
3. Say it right	The same system reads raw agent text correctly (Hinglish, Tamil–English, amounts, IDs and names). It changes language in the middle of a sentence and keeps the same voice	Text into speech
4. Judge without a jury	An automatic evaluator that scores naturalness, pronunciation and speaker similarity in Hindi and Tamil. Its scores agree with native listeners	After generation

The parts feed each other. The data from Part 1 teaches a new language to your Part 2 system. Part 3 makes that system read real agent text in a cloned voice. Part 4 measures what the system produces. We expect teams to divide into sub-teams, one for each part, and to integrate their work into one system.

Choose a lead part. Every part must work: your frozen system must produce an output for every evaluation item in each part. We report each part next to its own baseline. Within one week after the release of this problem statement, name one part on Discord as your lead research contribution.

The judges score novelty and depth mainly on your lead part. The other parts need a working system, an honest comparison with the baseline and measured results. We welcome novel work in the other parts and give credit for it. Scoring gives the exact points.

We score depth, not completion. We score results and research quality separately. A focused idea in your lead part can score well on research quality, even when its results are negative. For this, test the idea carefully and give an honest analysis of what worked and what did not work.

Room to go further. Each part gives optional directions that give your team its own angle. These directions count under novelty and depth, not under results points. They add no required work. Measure what you claim for them with a listening check or an automatic measure.

You build your own evaluation. The design of a fair test is part of the research. At the mid-term, every team contributes a test set for each part. We evaluate every part on sets that other teams contributed. In Part 1, these sets are held-out sentences that other teams wrote in your language. Part 1 also uses samples of the output of your pipeline and its baseline. Teams freeze their systems before the final input release and the random sampling. Then they submit raw evidence for verification (see How evaluation works).

Languages at a glance:

- Part 1: one of Bhojpuri, Chhattisgarhi, Magahi or Maithili.
- Part 2: Hindi, Tamil and English, plus your Part 1 language.
- Part 3: Hindi–English and Tamil–English.
- Part 4: Hindi and Tamil.

Part 1 · Data from almost nothing

Problem. Public Indian speech datasets mostly cover the major languages. Millions of people speak languages like Bhojpuri, Chhattisgarhi, Magahi and Maithili. But most recorded speech in these languages is noisy and spontaneous, and accurate text for it is rare.

Your task. Build an automated pipeline that turns the raw data pool of your language into clean, aligned text–audio pairs for TTS training. Use its output to teach the language to your Part 2 system. Show that the system speaks the language better than the same recipe trained on the existing transcripts of the pool.

Languages and data pool. Choose one of four languages: Bhojpuri, Chhattisgarhi, Magahi or Maithili. Your raw data pool is the subset of Vaani (IISc and ARTPARK) for that language. Use config `audio/<Language>` of the Hugging Face dataset `ARTPARK-IISc/Vaani` at revision `9e6ed4f94f7060e633cbc500480623ae7521522c`. The dataset is free under CC-BY-4.0. Request access on Hugging Face.

The pool is spontaneous speech from many speakers and districts. The speakers talk about images that they see. Only a small part of the pool has transcripts:

- Bhojpuri: about 490 hours of audio, of which 27 hours have transcripts.
- Chhattisgarhi: about 407 hours, of which 18 hours have transcripts.
- Magahi: about 111 hours, of which 6.5 hours have transcripts.
- Maithili: about 466 hours, of which 21 hours have transcripts.

The clips are 16 kHz mono WAV, and most clips are 2–11 seconds long. The config for each language is 12–56 GB. Many clips have background noise. Many clips with the label Magahi or Maithili are closer to Hindi. So the selection of usable speech in your language is part of the task.

In your Part 1 language, the pool is your only source of speech. Do not add other recordings or synthetic speech in that language. This includes speech from APIs. You can use public pretrained checkpoints released before this problem statement, even if they include the language. You can also use text from any source and speech in other languages.

Name your language on Discord within one week after the release of this problem statement. Each language needs at least two teams, because held-out sentences come from other teams. If only one team chooses a language, a judge asks that team on Discord to choose a different language.

Directions you might take:

- Data selection: language identification, and speaker and quality filtering, to find the clips that are worth transcribing.
- Cleaning and segmentation: denoising, music removal, diarization (who spoke when), and splitting into utterances.
- Transcripts: train ASR on the transcribed part of the pool, or start from the ASR of a related language, with confidence filtering.
- Automatic scoring of each pair, to filter out the worst data.
- Training-side work: transfer from a related language, or weakly supervised training.

The bar is intelligible speech, not studio quality. Something is better than nothing.

Room to go further (optional):

- Keep the spontaneous, conversational style of the pool in the trained voice, instead of filtering the pool down to speech that sounds read. Most Indian training data is narration.
- Run the same pipeline on a second underrepresented language, to show that it generalizes.

Held-out set you submit at the mid-term: 100 new sentences in your language. Write native words in Devanagari and any English words in Latin script. Each sentence has 5–20 words and is about an everyday topic. Put them in `inputs/sentences.csv` with the columns `sentence_id` and `text`. Do not take the sentences from Vaani or any public text, and do not share them with other teams. The organizers release them to the other teams in your language when they release the evaluation inputs. No team is evaluated on sentences that it wrote. Do not use your own sentences to train or tune your system.

Before teams generate outputs, the organizers' native listeners can reject a selected sentence. They can reject it only if it is not an understandable sentence in your language. The organizers record the reason and take the next sentence in hash order. If fewer than 30 sentences remain, use all of them.

Dataset freeze. By the freeze, lock two complete manifests. `pipeline_manifest.csv` lists the final output of your pipeline. This output is exactly the Part 1 data that your final system trained on. `baseline_pipeline_manifest.csv` is the manifest for the baseline. Do not repeat audio under different IDs in either manifest. Make the baseline manifest with these steps:

1. Start with every pool clip that has a non-empty `transcript`.
2. Remove the clips whose transcript contains `[unintelligible]`, `[inaudible]` or `<UNKNOWN_SEGMENT>`, in any letter case.
3. In the other transcripts, delete every other tag in angle or square brackets, such as `<pause>`, `<static_noise>`, `</static_noise>` or `[breathing]`. Keep the words between paired tags.

4. Delete all text in curly braces. This text is Vaani's standard or English spelling of the word just said.
5. Delete `--`.
6. Collapse whitespace, and drop the transcripts that are then empty. Make no other change.
7. Keep each remaining clip once, with its original audio.

Each manifest includes stable `pair_id` values, transcripts, audio paths and the SHA-256 hashes of the audio. In your pipeline's transcripts, write native words in Devanagari and English words in Latin script. Sort the rows by `pair_id`. Record the SHA-256 hashes of both manifests in `freeze.txt`, and include the unchanged manifests in `code.zip`.

After the release of the random seed, use the selection procedure in Appendix A to sample 200 pairs from each manifest. If a manifest has fewer than 200 pairs, use all of them, and count each missing pair as incorrect. Put both samples in `outputs.zip`, labeled by pipeline for scoring. Also put in it the held-out sentences that your system and the voice-gain baseline read. Because you lock the manifests at the freeze, both sampling populations are fixed before anyone knows the seed.

Evaluation (5 results points, see Scoring):

- **Pipeline precision (2.5 points).** The organizers check the 200 sampled pairs from your pipeline and the 200 from the baseline manifest (400 pairs in total). They mix the samples and hide which pipeline produced each pair. At least two of their native listeners check each pair once. A listener marks a pair correct only if the text is exactly what is said, the audio is clean, and the speech is in your Part 1 language, not Hindi. Hindi or English words mixed into a sentence are acceptable. Listeners judge spoken words only, and ignore tags in brackets. Any accepted spelling of a word counts. Where your language has no standard spelling, use a published orthography or your own, and document it in your report. The score of a pair is the share of its listeners who marked it correct. Precision is the average pair score. Your points are 2.5 × the precision of your pipeline. The organizers report the precision of the baseline manifest next to yours, but it does not change your points. The checks can start when the seed is released, because the pairs were frozen at the freeze.
- **Voice (2.5 points).** The seed picks 30 held-out sentences in your language. It also picks one pooled Part 2 speaker from a team that does not work in your language. Every team in your language reads all 30 sentences in the voice of that speaker. The judges consider how far that speaker's enrollment language is from your Part 1 language, and you can discuss this in your report. At least 10 of the organizers' native listeners first hear the enrollment clip. Then a listener hears your version and another team's version of the same sentence, blind. The listener rates the pair on a CMOS scale from −3 to +3. A positive rating means that your version is better. A version loses by 3 if the listener cannot understand it, or if it does not sound like the enrolled speaker. If both versions fail, the rating is 0. The seed puts the teams of your language in a ring (Appendix A). Your opponents are your two neighbours in the ring, or the one other team if your language has two teams. Each pair is judged once and counts for both teams. Your score uses only the sentences that your team did not write. Your points are 2.5 × (mean CMOS + 3) ÷ 6, with the mean over listeners, sentences and opponents.
- **Voice-gain baseline (reported, not scored).** Also run the voice-gain baseline on the same sentences, and report it next to your system. It is different from your system only in its Part 1 training data: it trains on the baseline manifest, not on the output of your pipeline. All other things stay the same: starting checkpoint, other data, seeds, hyperparameters, training steps, checkpoint selection, inference settings and any later training.

Part 2 · My voice, my languages

Problem. Many Indians speak three or four languages. If you clone the voice of a person from a short clip today, the clone has one of two problems. It speaks only the recorded language, or it loses the voice of the person when it changes language. Large models now start to do this well. But to do it well for Indian languages, with a model size that a team can train, is still mostly an open problem.

Your task. You get a clip of at most 30 seconds, in which a person speaks one language (Hindi, Tamil or English). Synthesize that person speaking Hindi, Tamil, Indian English and your Part 1 language. More languages are optional. In each language, the output must sound like the same person and be clearly intelligible. The pronunciation should be like that of a native speaker of each language. You can adapt the model to the enrollment clip, for example with LoRA. The adaptation procedure must be frozen and run automatically for any new speaker.

We judge pronunciation against how a native speaker of the target language says the words, not against the accent of the enrolled speaker. Keeping the person's own accent counts under novelty, if you show it against real reference speech. Part 1's voice test measures your Part 1 language, and that test uses a cloned voice.

Two routes, both welcome:

- **Architecture:** change the model itself. For example, separate "who is speaking" from "which language's sounds" inside the model, or add new speaker or language conditioning.
- **Post-processing:** generate with a strong native-speaker voice, then convert it to the voice of the target speaker (voice conversion). Or refine the output after generation.

You can combine the two routes. Explain why your route works.

Room to go further (optional):

- **Controllable delivery:** speak the same words at a requested pace or tone (calm, warm, urgent). Keep the words intelligible and keep the identity of the speaker. Test if listeners recognize the intended delivery.
- **Expressive prosody:** questions marked only by pitch (*aap kal aa rahe ho?*), particles like *na* and *ji*, and natural fillers like *haan ji* and *accha*.
- **Conversational from narration:** make the voice sound conversational when most of its training data is read narration.

Test set you contribute at the mid-term: three consenting multilingual speakers who are not in any public dataset. Use Hindi for the enrollment clip of one speaker, Tamil for the second speaker and English for the third. For each speaker, provide:

- An enrollment clip of at most 30 seconds in one language.
- 10 reference sentences that the speaker recorded in each of the other two languages.
- Signed consent on the template in Appendix D. Your team keeps the signed forms and shows them to the judges on request. Do not put the forms in `test_set.zip`.

Recording specification. Each speaker is at least 18 years old. Speakers do not have to be team members, and an accent in a non-native language is fine. Record in a quiet room with no music or other voices, and with one speaker in each clip. Record mono WAV at 16 kHz or higher, with samples of at

least 16 bits. The enrollment clip is 10–30 seconds of natural speech. Write new reference sentences of 5–20 words about everyday topics. The three speakers can read the same sentence texts. Do not take them from public datasets, and do not include personal information.

Put the enrollment clips and the sentence text in `inputs/`. Put the reference recordings in `answers/`.

Evaluation (5 results points). You synthesize every reference sentence for every pooled speaker except your own. An item is one reference sentence for one pooled speaker. Its target language is Hindi, Tamil or English. Compute each measure separately for each target language, then average the three languages with equal weight. The measures are:

- **Speaker similarity (2 points):** the cosine similarity between the enrollment clip and your output, with `microsoft/wavlm-base-plus-sv`, averaged over items. The reference value is the same average for the real reference recordings of the speaker. In each language, divide your similarity by the reference value, and keep the ratio between 0 and 1. Your points are 2 × the average ratio over the three languages.
- **Intelligibility (1.5 points):** the character error rate from `openai/whisper-large-v3`, averaged over items. In each language, divide 1 − your error rate by 1 − the error rate of the real reference recordings. Keep the ratio between 0 and 1. Your points are 1.5 × the average ratio over the three languages.
- **Naturalness (0.75 points) and pronunciation (0.75 points):** the seed picks 25 items for each language, and up to three other Track 1 teams as your opponents. At least five of the organizers' native listeners hear your output and an opponent's output of the same item, blind. A comparison uses only an item whose speaker neither team contributed. The listener rates the pair twice on a CMOS scale from −3 to +3, once for naturalness and once for pronunciation. A positive rating means that your output is better. Listeners rate naturalness by how much the clip sounds like a real person speaking. They rate pronunciation by how close the clip is to a native speaker of that language. For each measure, average the ratings over listeners, items and opponents. Your points are 0.75 × (mean CMOS + 3) ÷ 6 for each measure.

Consent. Use public datasets whose licences permit research use. All new recordings need the written consent of the speaker, including consent to clone their voice. Do not publish a cloned voice of a person without that consent.

Part 3 · Say it right

Problem. Voice agents in India read text that LLMs write, and that text is messy:

- Hinglish typed in English letters, where some words look English but are Hindi: in "*woh log kal yahan the*", *the* is थे ("were"), not English "the".
- English words with Tamil endings, like *meeting-ல்* ("in the meeting").
- Amounts, dates and IDs: ₹2.5 cr, 4:30, a PNR, a UPI ID, a vehicle number.
- Indian names, often romanized in ways that do not show how people say them.

TTS systems misread these items all the time, and one misread amount or name can break a call. A sentence that changes language is also a small cross-lingual cloning problem: the voice must stay the same person across the change.

Your task. Make your system read this type of text correctly and naturally, in Hindi–English and Tamil–English, in a voice cloned with Part 2. Your contribution must be learned. We compare it with a prompted LLM, not with hand-written rules. Learned means that your contribution includes model weights that your team trains or fine-tunes on data. Prompting, hand-written rules or lookup tables alone do not count as the contribution. You can use them next to a trained component, if your controlled comparison shows what the trained component adds.

Baseline. The baseline is your Part 2 voice system without your reading contribution. It reads text that an open LLM rewrote with the prompt and settings in Appendix C, unchanged. The voice system of the baseline is the checkpoint of your final system from before your Part 3 training, with its own hash in `freeze.txt`. If your Part 3 contribution does not change the weights of the voice system, the baseline uses the same checkpoint as your final system. The LLM is `sarvamai/sarvam-m` (24B parameters, Apache-2.0) at revision `01534a53c46f2788e392dbb3d994e0fa8f04d3fd`. Run it with 4-bit or higher precision.

Freeze the checkpoint, prompt and inference settings of every LLM that generates your Part 3 training data, and report its size. All model-generated training data for this part must come from open models. You can report your own normalization rules as an additional comparison.

Two routes, both welcome:

- **Learned frontend:** train a model that rewrites raw agent text into a form that your voice system reads correctly. This means the correct language and script for each word, numbers and IDs expanded the way a bilingual speaker says them, and names respelled or converted to phonemes. A design that works on streaming text with only a few words of lookahead counts under novelty.
- **End to end:** fine-tune the speech model in your voice system to read raw mixed text directly, with no separate frontend.

Efficiency can count under novelty and depth. Compare your complete system with the complete baseline (prompted LLM plus voice system) on the same hardware and inputs. Report model sizes, inference settings and time to first audio. Speed counts only if your results are not clearly worse than the results of the baseline.

Room to go further (optional):

- **Phoneme-level pronunciation control:** a shared sound inventory for Indian languages, so that a user can correct a mispronounced name by its sounds.
- **Fix one word:** repair a mispronounced name or brand term in an existing clean recording, without re-recording it. Regenerate only that word with the correct pronunciation, so that it fits the voice and recording around it. Measure if the word is now correct, and if blind listeners can find the edit.
- **Written to spoken:** change formal written language into natural spoken language, for example literary Tamil into spoken Tamil. Keep the meaning, names, numbers and the intended politeness. Compare speech from the changed text with speech from the original text, in the same voice.

Starting data: transliteration sets (Dakshina, Aksharantar), code-mixing sets (LinCE, GLUECoS), and synthetic agent replies that you generate. Record generated data in your data manifest.

Test set you contribute at the mid-term: 50 realistic agent replies in each of Hindi–English and Tamil–English. Each reply has at most 40 words and comes from a domain such as banking, healthcare, travel or government services. Write Hindi–English replies in Latin script, like the first example above. Write

Tamil–English replies in Tamil script, with English words in Latin script, like the second example. Every reply contains at least one amount, date, time, ID or name.

For every amount, date, time and ID, list all accepted readings under the policy in Appendix B. For example, *dhai crore* and "two point five crore" are both correct for ₹2.5 cr. For every name, include a reference recording of the accepted pronunciation, with the speaker's signed consent on the Appendix D template. Put the replies in `inputs/`. Put the accepted readings and name recordings in `answers/`.

Evaluation (5 results points). The seed picks 25 pooled replies for each language pair, none of them your own, and one pooled Part 2 speaker. You synthesize those 50 replies in the voice of that speaker, and the baseline does the same. At least five of the organizers' native listeners hear each reply in two versions: your output and the output of the baseline. They hear each version once, blind and mixed together. The scores of the baseline are reported next to yours, but they do not change your points.

- **Facts and names (3 points):** for each amount, date, time and ID, the listener writes down the words that they heard, in any script. The listener does not change the words to digits or symbols. A written answer is correct when it has the same words in the same order as an accepted reading. Script, spelling, spacing and punctuation do not matter. A reading that the contributor did not list is also correct if it follows Appendix B; the organizers mark it as unlisted. For each name, the listener plays the reference recording of the name and answers if the reply said the name the same way. The organizers record each match decision. The score of a listener for a reply is the share of its facts and names that the listener got correct. The score of a reply is the average over its listeners, and the measure is the average over replies. Your points are 3 × the measure.
- **Mixed-language reading (2 points):** for each reply, a listener answers three yes-or-no questions. The listener passes the reply only if all three answers are yes. The score of a reply is the share of its listeners who passed it, and the measure is the average over replies. Your points are 2 × the measure.
  - **Right language:** was every word said with the sounds of its own language? Answer no if a word is read in the wrong language, such as the Hindi *the* read as English "the". Also answer no if an English word does not sound the way an Indian English speaker says it.
  - **Same person:** did the voice stay the same person across every change of language? Answer no if, at any change, the voice sounds like a different speaker or its quality jumps.
  - **Natural:** does it sound like a fluent bilingual speaker? Answer no for unnatural pauses at changes of language, stress in the wrong place, or a flat, robotic reading.

Part 4 · Judge without a jury

Problem. The standard check of TTS quality is a MOS test, where people listen and rate. It is slow and expensive, so models are trained without a quality check between rare listening tests. Automatic predictors exist, but they were built mostly on non-Indian speech. Speech-recognition round trips also hide the errors that are important in Indian languages. The language model of the recognizer "fixes" a word that was mispronounced. For example, पल (*pal*, "moment") and फल (*phal*, "fruit") differ only by a puff of air.

Your task. Build an automatic evaluator for Indian TTS that agrees with native listeners on all three criteria, in Hindi and Tamil:

- **Naturalness:** does it sound human?

- **Pronunciation accuracy:** are the sounds correct? This includes contrasts that change meaning, like aspiration (*pal*/*phal*), retroflex vs dental (ट/त), double consonants (*patā*/*pattā*), vowel length and nasal vowels.
- **Speaker similarity:** does it sound like the intended person?

Your metric can be language-specific, speaker-specific or language-agnostic. One natural direction is a minimal-pair test: a set of word pairs that differ by one sound, with a classifier that checks which word the TTS actually produced. Use your evaluator on your own Parts 2 and 3 during development. We do not score that use, but it is a natural first test of the evaluator.

Room to go further (optional):

- **Delivery:** judge if speech has a requested pace or tone, or the prosody that a sentence needs, such as a question marked only by pitch.
- **Conversational naturalness:** judge if speech sounds conversational, not read.
- **Explained scores:** show the word or sound that was mispronounced, not only a number.

Input contract. Your evaluator is a command-line tool. It reads a manifest CSV with these columns:

- `clip_id`, `audio_path` and `language`.
- `text`: what the clip was meant to say.
- `reference_audio_path`: for speaker similarity.
- `target_word` and `contrast`: the two possible words for minimal-pair items.

Columns that a criterion does not need can be empty. All required audio, including speaker-reference audio, goes in `inputs/`.

Output contract. Write a CSV with `clip_id`, `criterion` and `score`, with one row for each clip and criterion. Use `naturalness`, `pronunciation` or `speaker_similarity` as criterion names. For each minimal-pair item, also write `predicted_word`: the word in `target_word`, the word in `contrast`, or `unclear`. The baseline must return the same fields.

Test set you contribute at the mid-term: 100 to 150 clips of at most 20 seconds in each of Hindi and Tamil, in the input format. The clips come from at least three TTS systems plus human recordings. Your own Part 2 system or a commercial TTS API can be one of the three. Name every system in your report. Include at least 20 minimal-pair items for each language. Each clip counts as one item, whether or not items share a sentence. Submit the clips with `answers/clip_metadata.csv`. Then the organizers' native listeners annotate every clip on all three criteria, with two listeners for each clip who work independently. The organizers add their answers to your set:

- `answers/ratings.csv`: `clip_id`, `criterion`, `rater_id`, `rating`. Each clip has two 1–5 ratings for each criterion. A naturalness rating cannot replace a pronunciation or speaker-similarity rating.
- `answers/heard_words.csv`: `clip_id`, `rater_id`, `heard_word`. For every minimal-pair item, each listener records the word that they heard: `target_word`, `contrast` or `unclear`, in the same way as the output contract. Store the actual word, not the column name. Keep the original answers of both listeners, including disagreements.

- `answers/clip_metadata.csv`: `clip_id`, `system_id`, `text_id`. Identify the source system or human recording. Give the same `text_id` to clips with the same intended text in the same language. Do not put this metadata in the inputs of the evaluator. The set must include the same texts rendered by different systems. Clips that are compared for speaker similarity must share the same target-speaker reference.

Evaluation (5 results points). We score your frozen evaluator on the sets of other teams.

- **Agreement with listeners (3 points):** the Spearman correlation between your scores and the average rating of the listeners. Compute it in each qualifying contributed set for each criterion and language, because each set has its own listeners. Average over sets for each criterion and language. Then average over the six criterion–language combinations. Use clips where the two ratings for that criterion differ by one point or less. The reference value is the correlation between those two listeners on the same clips, averaged the same way. Your points are 3 × your correlation ÷ the reference value, kept between 0 and 3.
- **Word identification (1 point):** the accuracy of `predicted_word` against the agreed `heard_word` on minimal-pair items. Include an item only when both listeners give the same word label, including `unclear`. Your points are 1 × the accuracy.
- **Pairwise ranking (1 point):** for naturalness and speaker similarity, the accuracy of the order that you give to pairs of clips of the same text from different systems. Average over the two criteria. In each contributed set, use every pair that has the same `text_id` and language, different `system_id` values, and a strict preference in the same direction from both listeners for that criterion. Speaker-similarity pairs must share the target-speaker reference. Count each pair once. A predicted tie is incorrect. Your points are 1 × the accuracy.

Usable coverage. A contributed set counts toward agreement for a criterion and language only if it meets all these conditions after the filters above:

- It has at least 30 usable clips.
- The ratings of each listener vary across those clips.
- The average ratings of the listeners also vary.

These conditions make it possible to calculate the agreement score and its human reference value. A set counts toward word identification if it has at least 10 agreed minimal-pair items in that language. It counts toward pairwise ranking if it has at least 10 usable pairs for that criterion and language. The organizers report the coverage of each set with the mid-term feedback. If your set falls short, fix it after the mid-term feedback. If you do not fix it, it is left out of the pool where it falls short.

Undefined scores. If an evaluator gives every clip in a set the same score for a criterion, its correlation on that set is 0. A missing score or `predicted_word` is a missing output (see Scoring). An invalid `predicted_word` counts as incorrect. These rules also apply to the baseline. In a bootstrap replicate, use a correlation of 0 when either correlated vector is constant. Use the same sampled clips for the system and the baseline, and keep the qualifying sets and averaging weights the same in all replicates.

Unavailable measures. If a criterion and language has no qualifying set, it is left out of the average of that measure. If word identification or pairwise ranking has no qualifying data at all, its points move to Agreement with listeners. If agreement has no qualifying data, divide the full 5 points equally among the other available measures. If none of the three measures has qualifying data, award 2.5 of the 5 results points as a neutral score, and report that Part 4 could not be measured. These fallbacks apply only to missing qualifying evaluation data. Missing system outputs still follow the scoring rules.

Track 2: Recursive Self-Improving Speech System

Track 2 is “Recursive Self-Improving Speech System”. Each team picks Track 1 (our multilingual voice system) or Track 2. Both tracks use the same rubric, and one ranking covers all teams.

The task

Teams build a captioner and a generator for Hindi and Tamil, and run the loop for at least one round after round 0; more rounds are allowed, not required. In each round:

- The captioner describes the speech pool. This expands the training set.
- The generator trains on that set to speak from text plus a description.
- The generator makes speech that targets the weak areas of the captioner. That speech trains the next captioner.

People, other models and datasets can help at every step of training. The submitted captioner and generator must then run on their own: open models only, no closed API inside them, and no person correcting or choosing their outputs. The captioner takes one audio file of at most 30 s and returns text; the generator takes text plus a description and returns audio at 24 kHz or higher. The final captioner and generator are what we score. The loop is how you get there, and the fewer rounds it takes, the better. Teams may also build one model that both describes and generates speech, as UALM does for audio.

What they start with

- **Data:** Vaani Hindi and Tamil audio (about 14,900 and 850 hours), with its transcripts and metadata, plus any other public dataset.
- **Round 0 (reported, not scored):** Any open model, at any precision, as long as the recipe is reproducible. We suggest *Qwen3-Omni-30B-A3B-Captioner* for understanding and *Qwen3-TTS-12Hz-1.7B-CustomVoice* for generation. Teams submit round 0 at the mid-term and report it next to every later round. Scores are absolute.
- **Descriptions:** free text in English that says what a careful listener hears, with times and numbers, not a list of tags. Three axes, all required; the item types under each are the ones in Appendix C.
- **The voice:** Speaker, Source, Morphology, Pitch, Timing, Event. Who speaks; how the voice is produced (clear, breathy, creaky, strained, falsetto); how each phrase starts and ends; pitch in semitones and timing in milliseconds; what happens, and when.
- **The recording:** Channel, Space, Other sounds, Coupling. The line or microphone, the room, what else is audible, and how the voice changes in response to it.
- **The intent:** Addressee, Subtext, Change. Who is being spoken to, what is meant beyond the words, and how that moves across the clip.
- There are no fixed label sets. Each description must also work as an instruction to the generator, so several attributes can be requested at once, as with Fugatto’s free-form instructions.
- Descriptions can also be fantastical, beyond anything recorded, as in Fugatto: for example, a hip-hop verse rapped over a beat made of construction-site machinery.
- At evaluation, a clip is at most 30 s, and a description is cut at 300 words before grading, so that listeners can check every rubric item by ear.

Two descriptions at the level we expect are in the Appendix tab. They are the upper bound; most clips are plain, and a plain clip needs the same exactness about the few things that happen.

The main rules

- Teams can use any label source: the loop models, other models, public datasets, tools and people. For example, people can correct captions or pick between outputs.
- One manifest records the source and the labeler of every training example.
- For each model, a Pareto frontier of its score on your own held-out audio against model size and compute.
- **Report the scores and the compute of every round.**
- Teams report every round on held-out real audio. Add harder clips when your models stop failing on it.

What teams contribute

Each team is evaluated on the clips of the other teams, 30 × (n – 1) per language for n teams, and Rime writes the rubric for all 30 × n. At the mid-term, each team records, for each language, 30 clips for the captioner to describe, from as many native speakers as it can, with no speaker in more than five clips, every clip at most 30s, and writes 30 descriptions for the generator to render, each with the text to say. A generation item has no recording and no speaker sample: one description has many valid renderings. Rime tags every clip with the axes and item types it exercises (Appendix C), so every score can be reported by axis.

Rime writes every rubric; teams never see one. A rubric is a set of items, each on one axis and of one item type; the axes and item types are published (Appendix C), the items are not. Rime gives each item a weight by how hard it is. A clip's score is the weight of the true items the description states, minus the weight of the items it contradicts or falsely states, divided by the total weight of the true items, floored at 0; so an easier rubric earns no more than a hard one. An item counts as stated only if the description commits to it; a guess between alternatives does not. A description rubric has true items, global ones about the whole clip and local ones with a start and end time, for example "The speaker laughs at 3.2–3.9 s", and a few statements that are false for the clip: a description that asserts one of them loses points, which is the penalty for inventing detail. A timed item counts when the description places it inside the item's span widened by 250 ms on each side. Two listeners check every item.

A generation rubric is the same rubric in the other direction: its items are the statements the description makes, and listeners check each one against the audio. What is in the audio but not in the description is not an error. Every output also gets the same fixed checks for natural speech (no artefacts, one consistent voice, natural timing, events that sound real), a blind CMOS against the other teams' outputs for the same description, and the diversity count in the scoring table.

How we score it

Measured results are worth 20 of 100 points. Judges score the other 80 on problem framing, novelty and depth, rigour, reproducibility and presentation, against the anchors in the competition rules. Every measure is absolute: a higher score is better, whatever round 0 was. Each measure is reported with a bootstrap confidence interval over items. Ties break by less compute, as the Pareto frontier shows.

Part	Measure	Points	How it works
Description	Rubric score	10	A fixed grader model checks each description against the rubric and computes the weighted score above; points = 10 × the mean over clips.
Generation	Instruction relevance	5	Three listeners mark each check as met or not; the majority decides. Points = 5 × the share of checks met.
Generation	Naturalness (CMOS)	2	Blind CMOS (–3 to +3) against each other team's output for the same description, averaged over opponents. Points = 2 × (mean + 3) ÷ 6.
Generation	Speaker diversity	2	Generate 10 samples per description. Count the distinct speakers among the samples that pass the relevance checks, using a fixed speaker-verification model at its standard same-speaker threshold. Points = 2 × the mean count ÷ 10. One voice for every description scores 0.
Generation	Intelligibility	1	Character error rate with AI4Bharat IndicConformer against the native-script text, averaged over descriptions. Points = 1 − CER, floored at 0.

Ideas we suggest (optional)

- Target the weak areas of the captioner that matter most for speech.
- Use the captioner as an automatic judge and reward for the generator, possibly in combination with a strong text LLM.
- Make labels for free by transforming clean speech in ways you control: convolve it with a chosen room response, add a chosen noise at a chosen SNR, shift its pitch by a known amount, or splice a laugh in at a known time. The description of what changed is then exact, and the pair (before, after, description) trains both models.
- Consider building instruction data from templates, LLM rewrites and known transformations of the audio, as Fugatto does. Be careful to only write that which can be confirmed in the audio. For example, calm does not imply low pitched voice, shouting does not imply anger.

Appendix

A. Two descriptions at the level we expect

The first two are invented to show the upper bound; they are not real recordings. The third is a plain clip, where the job is to say exactly that little happens.

Hindi, 9.2 s. Channel and space: a narrowband mobile call, band-limited to about 300–3400 Hz, with a codec that turns sibilants into a diffuse hiss and clips the loudest syllables softly, adding a buzz of odd harmonics on each stressed vowel. Behind the voice, a kitchen: a steady low hum, and from 7.0 s a

pressure-cooker whistle. The voice: a woman around sixty, eastern Hindi with Bhojpuri vowel quality (centralised schwa, lengthened final vowels), median pitch near 185 Hz, firm modal phonation with a pressed edge: strong second and third harmonics, almost no breath noise, a dark, chesty resonance with low formants for a female voice.

0.0–3.9 s, enumeration. She reads a grocery list as a repeated cell: clean plosive onset, short continuant, termination with a fall of about two semitones, five cells at a nearly isochronous period of 720 ms. Pitch stays within three semitones and loudness is flat; the voice is smooth and the rhythm repetitive. The intent is purely functional: she is reading to someone who is writing it down.

3.9–4.6 s, rupture. She stops mid-word on an open vowel, which does not terminate but is cut by a glottal closure, followed by a sharp nasal exhalation, a 180 ms noise band centred near 1.5 kHz. The register drops a fourth and the source changes to creaky voice on “arre”: a pulse rate near 40 Hz, rough and with no clear pitch. “woh toh bhool hi gayi” is spoken slowly, each syllable about 60% longer than in the list, with a gradual return from creak to modal voice over the phrase; the subtext is self-reproach, aimed at herself rather than at the listener.

5.6–6.4 s, laughter. Three closed-mouth pulses of about 180 ms, nasal and breathy, high (around 320 Hz) and descending across the three, joined by a voiced, breathy hum between them: a self-mocking laugh that releases the tension of the previous gesture.

7.0–9.2 s, speaking up over the whistle. The whistle enters as a quasi-periodic tone rising from about 1.8 to 2.4 kHz over two seconds with a noisy halo; through the phone band it sits among her upper formants. She answers it by raising her level by roughly 6 dB, tensing her vowels (the first formant rises, the voice brightens) and widening her range to about an octave. The last word is lengthened and carries a fall-rise, ending warmer, breathier and slower: a cadence that closes the clip.

Tamil, 8.4 s. Channel and space: a laptop microphone at roughly 40 cm in a small room with tiled walls; early reflections inside 10 ms colour the voice with a boxy band around 200–400 Hz, and a ceiling fan adds a steady rumble near 50 Hz with a faint periodic whoosh every 0.4 s. The voice: a man in his early twenties, Chennai Tamil with English technical vocabulary, median pitch about 150 Hz but a wide range, modal phonation under high subglottal pressure: bright, forward resonance with strong energy from 2 to 4 kHz and slightly nasalised vowels.

0.0–2.8 s, acceleration. A narrative gesture that speeds up: the syllable rate climbs from about 5 to 8 per second, the rhythm is clipped, with abrupt aspirated releases on the stops. The English words (“full toss”, “no-ball”) sit on higher, longer stressed syllables and stick out of the Tamil as rhythmic accents. Pitch rises in steps of roughly a semitone per phrase, loudness grows, and the energy concentrates in the upper harmonics, so the spectrum brightens as the excitement builds.

2.8 s, climax. On the stressed syllable of “no-ball” the voice flips into falsetto for about 200 ms: the pitch jumps an octave to around 310 Hz, the quality turns thin, breathy and almost sinusoidal, and the register break leaves a brief aperiodic transient. The body loses control of the gesture, which is the point of the story.

3.5–5.9 s, reversal. A dry alveolar click, a broadband transient of about 15 ms, cuts the flow. “enna da idhu” arrives in a new register: low, slow (syllable durations doubled), vowels stretched under a flattened falling contour; the voice darkens, the upper harmonics thin out, and a touch of creak appears on the final vowel. Mock disbelief, performed for the friend.

6.2–8.4 s, interruption and intimacy. A door slams off-mic, a low-mid thud with a reverberant tail of about 300 ms that rings the room's modes. He stops for about 400 ms, as if listening, then resumes in a near-whisper: unvoiced, breath-driven, the spectrum dominated by turbulence above 1 kHz with a few voiced fragments; at this level the lip smacks and tongue noises become part of the texture. Short bursts separated by pauses, conspiratorial: the intimacy comes from the change of register, not from the words.

Read sentence, Hindi, 5.8 s. A woman in her thirties reading a prompt, median pitch about 210 Hz, range five semitones, clear modal voice, relaxed. Two breath groups, 0.0–2.7 s and 3.05–5.8 s, with a 350 ms pause at the comma and one 180 ms inhale in it. Pitch falls gently across each group, 4.5 syllables a second, soft onsets, no emphasis anywhere. Close microphone, about 15 cm, a treated room, no reverberation, noise floor well below the voice. Reading for a corpus, addressed to nobody. Nothing else happens, and the description says so.

B. Background reading

- Fugatto (ICLR 2025): generating instruction data for audio generation, and combining instructions at inference.
- UALM (ICLR 2026): one model for audio understanding, generation and reasoning.
- Audio Flamingo 3 (NeurIPS 2025): an open audio-language model with staged training and on-demand thinking.
- MMAU (ICLR 2025): about 10,000 clips with expert-written questions across 27 skills in speech, sound and music.
- MMAU-Pro (AAAI 2026): 5,305 items across 49 skills, with multi-audio, long-form, multiple-choice and open-ended questions.

C. Rubric axes and item types

Every rubric item belongs to one axis and one item type. Teams see this list and never the items written for a clip.

Item type	Axis	Example
Speaker	the sound	Woman, roughly 50 to 70, eastern Hindi vowels
Source	the sound	Creaky voice on the second phrase
Morphology	the sound	Abrupt glottal cut mid-word at about 3.9 s
Pitch	the sound	Range within about three semitones in the first 4 s
Timing	the sound	Five list items at a near-constant 0.7 s period
Event	the sound	Three closed-mouth laugh pulses at 5.6 to 6.4 s
Channel	the ecology	Narrowband phone call, light clipping on loud syllables
Space	the ecology	Small tiled room, short bright reflections
Other sounds	the ecology	Pressure-cooker whistle rising from about 7 s
Coupling	the ecology	Louder and brighter once the whistle starts
Addressee	the intent	Reading a list to someone writing it down
Subtext	the intent	The laugh is at herself, not at the listener
Change	the intent	Functional at the start, self-mocking by the end

D. Ten terms

- **Modal voice**: the ordinary, clear way the vocal folds vibrate.
- **Breathy**: air leaks through the folds; soft and airy.
- **Creaky**: very slow, irregular fold pulses; a low rattle, also called vocal fry.
- **Strained**: folds and throat tight; loud, harsh, effortful.
- **Falsetto**: the high, thin register above the normal voice.
- **Semitone**: the smallest step on a piano; 12 make an octave.
- **Formant**: a resonance of the mouth and throat that shapes vowel colour.
- **Onset**: how a sound starts: abrupt, soft, or with a breath.
- **Register**: here, both the pitch range in use and the social tone (formal, intimate).
- **Lombard effect**: speaking louder and brighter when there is noise.

Both tracks

How evaluation works

Evaluation sets. At the mid-term, every Track 1 team submits a test set for each part. Every Track 2 team submits its clips and descriptions. We score the quality of the test sets at the mid-term (see Scoring). In Track 2, we evaluate each team on the clips and descriptions of the other teams. In Track 1, the source of the final evaluation items depends on the part:

Part	Final evaluation items
1	The held-out sentences of every team in your Part 1 language, plus the frozen manifests of your pipeline and its baseline. The seed selects 30 sentences, one pooled Part 2 speaker to read them, up to two opponents, and 200 pairs from each manifest (400 pairs in total).
2	The speaker sets of other teams, with Hindi, Tamil and English enrollment.
3	25 replies of other teams for each language pair, chosen by the seed. Your system and the baseline read them in the voice of one pooled Part 2 speaker that the seed picks.
4	The labeled clips of other teams in Hindi and Tamil.

Different items for each team. Pooled items never include your own contributions. So each team gets a slightly different selection. The seed ranks the same candidates in the same order for every team. So in Parts 2–4, most selected items are shared between teams. Your system and its baseline always use exactly the same items.

Inputs and answers. Each test set has two folders. `inputs/` holds what a system gets. `answers/` holds the scoring references and labels. A system must not need anything in `answers/` to produce an output. In Part 2, target-sentence recordings are answers. In Part 3, the accepted readings and name recordings are answers. In Part 4, speaker-reference audio is an input, but criterion ratings, heard-word labels and clip-grouping metadata are answers.

Order of the final window.

1. **Freeze.** Submit `freeze.txt`. It contains your code commit hash, the SHA-256 hash of every model checkpoint, and the exact generation command and its seed, for your system and every baseline. This includes the precision-baseline pipeline and the voice-gain baseline of Part 1, and the hashes of both complete Part 1 pipeline manifests.
2. **Inputs released.** The organizers release the pooled inputs for every Track 1 part and for Track 2, including the held-out sentences of Part 1, and a random seed. The organizers' Part 1 precision checks use the seed with your two frozen manifests, and the checks can start.
3. **Outputs locked.** Run your frozen system and every baseline on the evaluation items for each part. Put all outputs and the sampled pairs of Part 1 in `outputs.zip`. Then submit `outputs.zip`, and `outputs_hash.txt` with the SHA-256 hash of the ZIP. Generate one output for each input, or 10 for each Track 2 description. Do not edit outputs, and do not choose between several generations.
4. **Answers released.** The organizers release the scoring answers for Parts 2–4 of Track 1. Track 2 rubrics stay private, and Rime scores Track 2. Score your locked outputs on the automatic measures. The organizers run all listening tests, including the Part 1 checks of the sampled pairs and synthesized sentences.

5. **End-term submission.** Your `outputs.zip` must match the hash that you submitted.

Sampling and confidence intervals. The random seed, released with the evaluation inputs, sets the evaluation samples, the speakers and the listening order. Follow the selection and bootstrap procedures in Appendix A. The appendix is required and is part of the frozen evaluation rules. The organizers compute every listener-based measure and its confidence interval with these procedures. In Track 1, teams compute the automatic measures.

Listening tests.

- The organizers recruit and manage all listeners for scored tests. Teams do not recruit listeners or run listening tests for scoring. Listeners are native speakers of the language, and they are not members of any team.
- Listening is blind: listeners do not know which system produced a clip, and the order is random.
- Each listener rates each selected clip or pair once.
- The organizers run the Part 1 precision checks after the release of the random seed. They run all other listening after the outputs are locked, in Hindi, Tamil, English and the Part 1 languages. They share the listening results of each team after the evaluation window.

Fixed measurement tools. In Track 1, use the named tools: `microsoft/wavlm-base-plus-sv` for speaker similarity, and `openai/whisper-large-v3` for character error rate. Run Whisper with the language set to the language of the sentence. Before you compute the character error rate, apply Unicode NFC normalization to both texts. Then lowercase them, remove punctuation and collapse whitespace. You can also use your own extra metrics next to these tools.

Verification.

- **Automatic measures.** Judges run the scoring of the top-ranked teams again from their frozen code. A re-run within 2% (relative) of the reported value counts as a match.
- **Review.** If a check is flagged, you get 24 hours to explain. If the difference is an honest error, the values of the judges replace yours, with no penalty.
- **Violations.** Fabricated data, edited outputs or outputs that do not match the hash cost all results points, and the case is reported to the organizing committee.
- **Close calls.** In Track 1, if teams near a top-three boundary are within 2 total points of each other, the organizers collect more listening ratings for those teams.

Baselines. Each part has its own baseline. Baselines are reported next to your results, but they do not change your points. Use reproducible baselines with public code and model weights that you can run and freeze. For each part, choose the strongest eligible baseline that fits the specified comparison. Get your baselines approved by a judge on Discord within one week after the release of this problem statement:

- **Part 1:** fixed, as defined in Part 1: the baseline manifest for precision, and the same TTS training recipe trained on that manifest for the voice comparison.
- **Part 2:** a public cross-lingual voice-cloning system. To be eligible, a system must meet three conditions. It has public code and weights released before this problem statement. It clones a voice from 30 seconds of audio or less. It supports Hindi, Tamil and English. If no eligible system supports all three languages, the judges approve the closest system.
- **Part 3:** the prompted LLM defined in Part 3.

- **Part 4:** an existing public evaluator for each criterion.
- **Track 2:** round 0, as defined in Track 2. It needs no approval.

The required controlled comparison can be an additional experiment, if it is different from the baseline.

Additional comparisons. You can report the results of commercial APIs separately. Name the provider, model and version, request settings and run date. These results do not count toward results points, and the checkpoint-freeze requirement does not apply to them. Keep their returned outputs with the comparison evidence in `evaluation.zip`.

Rules

- **Models.** Use any open model whose licence permits research, and name the exact checkpoint. There is no limit on model size. You can use closed commercial APIs for additional comparisons. You can also use them to generate training data, except Part 3 training data and speech in your Part 1 language. You can also use them as a TTS source for the Part 4 test set. Never use closed commercial APIs inside the submitted system or its baselines.
- **Compute and data.** There is no limit, except the data rules of Part 1. Report the GPU type, the total GPU-hours and the hours of audio that you used. GPU-hours include failed runs and baselines. For each model, also show a Pareto frontier of quality against model size and compute. We give no credit for more compute or data, and you do not have to use all the compute you have.
- **Controlled comparisons (required).** Show the effect of your contribution in each part alone. Use the same starting model and conditions, and remove your contribution or replace it with a simple alternative. In Track 1, without a controlled comparison for your lead part, Rigour is capped at a judge score of 1. This is at most 2.5 of its 10 points. Examples follow this list.
- **Data provenance.** Submit a data manifest that lists each source, its licence and the hours used. Do not redistribute audio or text unless its licence permits it. Do not record anyone covertly, and remove personal information.
- **Anonymity.** Submissions must not show your institute. Use your team ID everywhere. Publish public repositories only after the results are announced.
- **No Rime product is required.**

Controlled comparison examples (Track 1):

Part	Example comparison
1	The same training recipe on data before and after your filtering method
2	The same base system with and without your conditioning or conversion method
3	The same voice system with the prompted-LLM baseline, and with your learned approach
4	The same evaluator with your contribution removed or replaced by a simple alternative, on the same held-out audio

Scoring

Every team gets a score with the same rubric, and all teams from both tracks are ranked together. The results measures are different for each track. The measures for Track 1 are below. The measures for Track 2 are in its How we score it section, and nothing in this section changes them.

Stage	Criterion	Points
Mid-term (30)	Problem framing and prior work	10
	Baselines and evaluation setup	5
	Contributed test sets	5
	Early results and revised plan	10
Final (60)	Results (computed)	20
	Novelty and depth: lead part	14
	Novelty and depth: rest of the system	6
	Rigour	10
	Reproducibility	10
Presentation (10)	Final presentation	8
	`summary.md`	2

Results come from your verified measurements. Every measure is absolute: its points are its weight × a share from 0 to 1. A higher score is better, whatever the baseline scored. Baselines run and are reported next to your results, but they do not change your points. Each measure is reported with a 95% bootstrap confidence interval over items (Appendix A). In Track 1, each part is worth 5 results points:

Part	Measure	Points	Formula
1	Pipeline precision	2.5	2.5 × precision
1	Voice	2.5	2.5 × (mean CMOS + 3) ÷ 6, against up to two other teams in your language
2	Speaker similarity	2	2 × your similarity ÷ that of the real recordings, at most 2
2	Intelligibility	1.5	1.5 × (1 – your error rate) ÷ (1 – the error rate of the real recordings), from 0 to 1.5
2	Naturalness	0.75	0.75 × (mean CMOS + 3) ÷ 6, against up to three other teams
2	Pronunciation	0.75	0.75 × (mean CMOS + 3) ÷ 6, against up to three other teams
3	Facts and names	3	3 × the share of facts and names heard correctly
3	Mixed-language reading	2	2 × the share of listeners who pass a reply
4	Agreement with listeners	3	3 × your correlation ÷ the correlation between listeners, from 0 to 3
4	Word identification	1	1 × accuracy
4	Pairwise ranking	1	1 × accuracy

Each part defines exactly how to compute its measures. In Track 1, where a measure divides by a reference value of 0 or less, it uses your own value instead, kept between 0 and 1. If a Track 1 CMOS measure has no opponent with shared items, it earns half its points. Appendix E lists the item types that each Track 1 part tests. Your results score is the sum of the results measures of your track, out of 20.

Every part must work. In Track 1, a part works when your frozen system produces an output for every evaluation item in that part. If your system is missing outputs, every measure in that part scores 0.

Worked example (illustrative numbers). Part 2 intelligibility is worth 1.5 points. In Hindi, your character error rate is 10.5%, and the real recordings have 5.0%. The Hindi ratio is (1 – 0.105) ÷ (1 – 0.050) = 0.895 ÷ 0.950 = 0.94. With ratios of 0.94, 0.90 and 0.98 in Hindi, Tamil and English, the average ratio is 0.94. The measure earns 1.5 × 0.94 = 1.41 points.

Comparing the two tracks. Both tracks use the same 100-point rubric, and one ranking covers all teams. The results measures differ because the tasks differ, so their scales can differ. A safety net keeps one track from being held down. It only ever scales Track 1 scores, so Track 2 is scored exactly as its section says:

- At the mid-term, the organizers compare the early results of both tracks. They tell teams on Discord if the scales look far apart.

- After the final results, the organizers compare the best results score of each track. If Track 1's best is below 75% of Track 2's best, every Track 1 results score is scaled up by one factor. Track 1's best then equals 75% of Track 2's best.
- If Track 2's best is below 75% of Track 1's best, every Track 1 results score is scaled down by one factor. Track 2's best then equals 75% of Track 1's best.
- Otherwise, or if the best score of a track is 0, nothing is scaled.

Ties. If two teams have the same total, rounded to two decimals, the team that reported fewer total GPU-hours ranks higher.

Everything else is judged. Each judge scores each criterion from 0 to 4 against the anchors below. Scores 1 and 3 are between the anchors. The points are the average judge score ÷ 4 × the weight of the criterion. The total is rounded to two decimals. Five Rime judges score every team in both tracks, against the same anchors. If two teams still tie after the compute rule, the lead judge of their track decides, or all five judges if the teams are in different tracks. All five judges decide the overall ranking together.

Criterion	0	2	4
Problem framing and prior work	Restates the problem statement	Relevant prior work, and a stated contribution	A precise gap in prior work, a testable hypothesis, and a reason why it should work
Baselines and evaluation setup	No running baseline	Baselines run for every part	Also controlled comparisons designed, and errors divided by category
Contributed test sets	A test set is missing	Every required test set meets the format	Also test sets that cover hard cases, and, in Track 1, a Part 4 set that meets usable coverage
Early results and revised plan	No results	First results against the baselines	Results with failure analysis, and a revised plan that follows from it
Novelty and depth: lead part	Existing models used unchanged	A meaningful adaptation of known methods	A new method or insight explored in depth. A clearly explained negative result can reach 4
Novelty and depth: rest of the system	Existing models used unchanged everywhere	A meaningful adaptation in at least one other part or model	Meaningful adaptations across the rest, all working in one system
Rigour	No controlled comparison (in Track 1, none for the lead part caps Rigour at 1)	Controlled comparisons and some ablation	Controlled comparisons for every part, ablations, errors by category, confidence intervals, honest limits
Reproducibility	Does not run	Runs after fixes	Runs as documented, and the frozen system reproduces the submitted outputs
Final presentation	Unclear	Clear account of the work	Clear, shows deep understanding, and answers questions well
`summary.md`	Missing, or written for specialists	Clear account of the problem and the result	A non-specialist could explain what you built and found, with one concrete example

Deliverables

Mid-term:

The mid-term checks that the direction, baselines and evaluation of each team are sound while there is still time to change course. It also collects the test sets that every team is evaluated on.

- `report.pdf`: at most 6 pages, plus references. **Why:** it carries the 30 mid-term points (problem framing, baselines and evaluation setup, early results and revised plan). Judges and mentors give feedback on it.
- `results.csv`: baseline and current numbers for each part on your development data. In Track 2, give the numbers for each round so far. **Why:** it shows in one comparable format that every baseline runs, and where your first results are against it.
- `test_set.zip`: one folder for each part, each split into `inputs/` and `answers/` where applicable: Part 1's 100 held-out sentences, Part 2's speaker set, Part 3's agent replies with their name recordings, and Part 4's clips, which the organizers label. In Track 2, your clips and descriptions for each language, for which Rime writes the rubrics. **Why:** every team is evaluated on the sets of other teams. So the sets must arrive early enough for the organizers to label, check and pool them. Their quality is scored (5 points).
- `round0.zip` (Track 2): the code, settings and checkpoint hashes of round 0. **Why:** it fixes the starting point that every later round is reported against.

Freeze:

- `freeze.txt`: code commit hash, checkpoint SHA-256 hashes, generation commands and seeds, for your system and every baseline. Also the SHA-256 hashes of both complete Part 1 pipeline manifests. In Track 2, the checkpoint hashes of every round, including round 0. **Why:** it proves that the evaluated system and baselines are the ones that were fixed before the release of the evaluation inputs and seed.

Outputs lock:

- `outputs_hash.txt`: the SHA-256 hash of your `outputs.zip`. **Why:** it fixes your outputs before the release of the scoring answers.
- `outputs.zip`: the file described under End-term, uploaded at the outputs lock. **Why:** the organizers need your outputs immediately to run the listening tests.

End-term:

The end-term collects everything that is necessary to score, verify and reproduce your final system.

- `report.pdf`: at most 8 pages in double-column conference format, excluding references, plus an appendix of at most 4 pages. Put the lead contribution, the main results for every part (in Track 2, both models and the loop) and the essential controlled comparisons in the main report. Use the report appendix for additional experiments and implementation details. **Why:** it is the main evidence for novelty and depth and for rigour (30 of the final-stage points).
- `code.zip`: source that matches the frozen commit, including your selection, bootstrap and scoring code. Include a `README.md` with setup steps, one command for each part that reproduces its headline result, and pinned dependencies (`requirements.txt` or equivalent). Also include every prompt, random seed, batch size and data order that is needed to reproduce your results. Put model weights behind a download link in the README, matching the frozen hashes. Include Part 1's complete, unchanged `pipeline_manifest.csv` and `baseline_pipeline_manifest.csv`, and Part 3's frozen baseline prompt. **Why:** judges run the scoring again and reproduce your outputs from it. It is scored under reproducibility (10 points).

- `outputs.zip`: the outputs of your system and every baseline on the evaluation items of each part, matching `outputs_hash.txt`. Include the 200 sampled audio–transcript pairs from each Part 1 pipeline (400 pairs in total), with their pipeline identities kept for scoring. Include Part 4's score CSVs with the predicted-word fields. In Track 2, include the descriptions from your captioner for every evaluation clip, and 10 audio samples from your generator for every evaluation description, from your final round. **Why:** these are the outputs that get scored, and they must match the hash fixed at the outputs lock.
- `evaluation.zip`: your measurements, the evidence for your automatic measures, and any additional comparisons. **Why:** judges can check your automatic measurements and additional comparisons.
- `data_manifest.csv`: every data source, its licence and the hours used. **Why:** it shows that every source is licensed and that you followed the data rules of Part 1.
- `demo.mp4`: at most 4 minutes. In Track 2, show one clip that round 0 and your final captioner describe, one description that both generators render, and one failure that remains. In Track 1, show one cloned voice of a speaker who agreed to the demo. The voice speaks your Part 1 language and reads a mixed-language agent reply, each next to the baseline. Also show one failure that remains. **Why:** judges can hear what the system actually sounds like next to its baseline.
- `summary.md`: at most one page, in plain language for a person who is not a speech researcher: the problem, what you built, what you found, and one example. It is scored as its own presentation criterion. **Why:** it shows that you can explain the work to a person outside the field.

No other files are accepted. Extra files get the standard Inter IIT penalty. The technical constraints, evaluation scheme and deliverables above are frozen at release.

Glossary

Term	Meaning
TTS	Text-to-speech: software that reads text aloud
ASR	Automatic speech recognition: speech to text
MOS	Mean opinion score: listeners rate quality from 1 to 5
CMOS	Comparative mean opinion score: listeners hear two clips and rate which is better, from –3 to +3
Diarization	Finding who spoke when in a recording
Alignment	Matching each piece of text to the part of the audio where it is spoken
Voice conversion	Changing who a recording sounds like, while keeping what was said
Enrollment clip	The short recording of a person from which a cloning system copies the voice
Speaker embedding	A vector that summarizes how a voice sounds. Similar voices give similar vectors
Code-mixing	Mixing two languages in one sentence, like Hinglish or Tanglish
Transliteration	Writing a word in another script: *mera* ↔ मेरा
Normalization	Changing symbols and digits into words, like ₹2.5 cr into "dhai crore"
Minimal pair	Two words that differ by one sound and mean different things (पल / फल)
Spearman correlation	How well two rankings agree, from -1 to 1
Ablation	Removing one part of a system to measure what that part contributes
Confidence interval	The range where a measured score would probably be if you did the test again
Random seed	A published number that fixes a random selection, so anyone can reproduce it

Appendix A: Required sampling and bootstrap protocol

This appendix is required and is part of the frozen evaluation rules. It applies to Track 1.

Random selection. The random seed fixes every random choice. Every team computes it the same way, with no random-number library:

1. For each candidate, compute the SHA-256 hash of the text `seed|selection|item_id`. Here, `seed` is the published nonnegative integer, written in decimal without leading zeros. `selection` is the name in the table below, and `item_id` is the ID of the candidate. The ID of a pooled item is `team_id/item_id`. Hash UTF-8 bytes with literal `|` separators, and with no added spaces, byte-order mark or trailing newline. These encoding rules also apply to the bootstrap.
2. Sort the candidates by hash, read as an unsigned hexadecimal integer, from lowest to highest. If two hashes are equal, sort them by ID in ascending UTF-8 byte order. IDs are case-sensitive. Team IDs and local item IDs use only ASCII letters, digits, periods, underscores and hyphens. Local IDs must be unique in their submitted collection.

3. Take the first *m*.

Selection	Candidates	*m*
`part1_sentences`	Every pooled held-out sentence in your Part 1 language	30
`part1_ring`	Every team in your Part 1 language, sorted by hash into a ring, so that the last team is next to the first	All
`part1_pipeline_pairs`	Every pair in `pipeline_manifest.csv`	200
`part1_baseline_pairs`	Every pair in `baseline_pipeline_manifest.csv`	200
`part1_speaker`	Every pooled Part 2 speaker from teams that do not work in your Part 1 language, or every pooled speaker if there is none	1
`part2_items_hi`, `part2_items_ta`, `part2_items_en`	Every pooled Part 2 item in that language except your own	25 each
`part2_opponents`	Every other Track 1 team	3
`part3_speaker`	Every pooled Part 2 speaker except your own	1
`part3_replies_hi-en`, `part3_replies_ta-en`	Every pooled reply in that language pair except your own	25 each

Bootstrap. Confidence intervals resample evaluation items, not listeners. The items are sentences for voice, pairs for precision, items in Part 2, replies in Part 3, and clips, minimal-pair items or ranking pairs in Part 4. Use this procedure:

1. Sort the eligible items by ID in ascending UTF-8 byte order. Use `team_id/item_id` for pooled items. The ID of a ranking pair is its two pooled clip IDs, sorted in that order and joined with `~`. Number the item positions from 0 to $n - 1$, where n is the number of eligible items.
2. Run replicates $r = 1$ through 1,000. In each replicate, make n draws with replacement, numbered $j = 0$ through $n - 1$. Hash the exact text `seed|bootstrap|measure|r|j`. Write r and j in decimal without leading zeros. Read the full SHA-256 digest as an unsigned hexadecimal integer h. Select the item at zero-based position h mod n.
3. Use the exact `measure` keys below. Angle brackets mark substitutions, and they are not part of the hash input. `language` is `hi`, `ta` or `en` in Part 2, and `hi` or `ta` in Part 4. In Part 4, `criterion` is `naturalness`, `pronunciation` or `speaker_similarity`. Paired measures use identical draws for the system and baseline. Part 1 precision uses separate keys for its two pipelines. Part 2 resamples each language separately, then averages the three languages. Part 4 agreement resamples each contributed set, criterion and language separately, then repeats the published averaging procedure. Pairwise ranking resamples each criterion separately, then averages the two accuracies.

Measure	Exact bootstrap key
Part 1 precision, system / baseline	`precision_pipeline` / `precision_baseline`
Part 1 voice	`part1_voice`
Part 2 speaker similarity	`part2_speaker_similarity/<language>`
Part 2 intelligibility	`part2_intelligibility/<language>`
Part 2 naturalness / pronunciation	`part2_naturalness/<language>` / `part2_pronunciation/<language>`
Part 3 facts and names	`facts_and_names`
Part 3 mixed-language reading	`mixed_language_reading`
Part 4 agreement	`part4_agreement/<team_id>/<language>/<criterion>`
Part 4 word identification	`part4_word_identification`
Part 4 pairwise ranking	`part4_pairwise_ranking/<criterion>`

In each replicate, compute the measure again from the resampled items, and sort the 1,000 values from lowest to highest. The endpoints of the confidence interval are the 25th and 975th values, at zero-based positions 24 and 974, with no interpolation. Apply Part 4's undefined-correlation rule in each replicate. Do not drop a replicate, and do not draw it again.

Worked example. With seed `123`, measure `facts_and_names`, replicate $r = 1$ and $n = 10$ sorted items, the first three draws are:

Draw *j*	Exact UTF-8 hash input	Zero-based item position
0	`123|bootstrap|facts_and_names|1|0`	0
1	`123|bootstrap|facts_and_names|1|1`	8
2	`123|bootstrap|facts_and_names|1|2`	1

Appendix B: Accepted readings policy

This appendix is part of the frozen evaluation rules. List every reading that a fluent bilingual speaker naturally says on a voice-agent call. A reading can use either language of the pair, or mix them the way a speaker does.

- **Amounts:** Indian number words (*lakh*, *crore*; Tamil *latcham*, *kodi*) in either language. Fractions such as *dhai*, *saade* or Tamil *irandarai* ("two and a half"), where a speaker uses them. Decimal readings such as "two point five crore". The currency word before or after the number. Not accepted: reading a symbol literally ("rupee sign"), or a change to a different scale ("twenty-five million") when the text uses lakh or crore.
- **Dates and times:** day and month in either language, with the day as an ordinal or a cardinal number. Clock times as a speaker says them, such as *saade chaar baje*, Tamil *naalarai mani* or "four thirty". Not accepted: reading the punctuation, as in "four colon thirty".

- **IDs** (PNR, UPI ID, vehicle number, phone number): letters as English letter names. If a part of a UPI ID or email is a word or name, such as rahul or sharma, it can be said as that word. Digits one at a time or in natural groups of two, in either language. Repeated digits can use "double" or "triple", as in "double nine" for 99. "@" as "at" or "at the rate", and "." as "dot". Every character must be read, in order.
- **Names:** the pronunciation in the reference recording of the name. Spelling variants do not matter, because listeners compare the sound with that recording.

Appendix C: Part 3 baseline prompt

This appendix is part of the frozen evaluation rules. Freeze the model and tokenizer revisions and the chat template. Send the prompt below as the only user message. Use the chat template of the model unchanged, add no system message, and turn off thinking mode. Replace `{reply}` with the raw reply text. Decode greedily (temperature 0, one beam, no repetition penalty), with at most 1,024 new tokens. Remove special tokens, and give only the newly generated text to the baseline voice system, unchanged. The prompt is the same for both language pairs.

```
Rewrite the agent reply below so that a text-to-speech system can read it aloud correctly.

Rules:
1. Keep the meaning and the word order. Don't add, remove or translate anything.
2. Write Hindi words in Devanagari and Tamil words in Tamil script. English words may stay in
Latin letters or be written as they sound in the reply's Indian script. A word may keep an
English stem with a Tamil suffix, as in meeting-ல.
3. Write every amount, date, time and other number as the words a fluent bilingual speaker
would say on a phone call, using Indian conventions such as lakh and crore.
4. Read IDs character by character: letters as English letter names, digits as number words,
"@" as "at the rate" and "." as "dot".
5. Keep names as they are said, written in the script of the language they come from.
6. Output only the rewritten reply.

Reply:
{reply}
```

Appendix D: Speaker consent template

Use this text, filled in, for every speaker that your team records for a test set in Parts 2, 3 and 4 of Track 1 or in Track 2. Each speaker signs before the recording.

Recording consent: Inter IIT Tech Meet 15.0, Rime problem statement

- **Who is recording:** team [team ID], contact [name and email].
- **What is recorded:** [for example, a 30-second clip of me speaking Hindi, and 10 sentences in each of Tamil and English].
- **How it is used:** teams in this event, its judges and the organizers' listeners can use my recordings, and generate speech in my voice, only to evaluate systems for this event.
- **Who receives it:** the organizers share my recordings with the other participating teams and the judges, and the organizers play them to their listeners. Nobody will publish my recordings. Nobody will publish speech generated in my voice or use it outside this event.
- **Deletion:** all copies of my recordings, all speech generated in my voice, and voice data made from them, such as speaker embeddings, are deleted by [deletion date announced by the organizers].
- **Withdrawal:** I can withdraw if I tell the contact above before the mid-term submission opens. Then my recordings are not submitted.
- **Age:** I am 18 or older.

Name, signature and date.

Appendix E: Track 1 item types

This appendix lists the kinds of items that each Track 1 part tests. Teams see this list. They do not see the items of other teams until the evaluation inputs are released.

Part	Item type	Example
1	Exact transcript	Every word that is said, and no other word
1	Clean audio	One voice, with no music and no clipping
1	Right language	Speech in your Part 1 language, not Hindi
1	Held-out sentence	An everyday sentence of 5–20 words, with English words in Latin script
2	Speaker identity	A Tamil output that sounds like the Hindi enrollment speaker
2	Pronunciation	Tamil words said the way a native Tamil speaker says them
2	Naturalness	A clip that sounds like a real person speaking
3	Amount	₹2.5 cr read as *dhai crore* or "two point five crore"
3	Date and time	4:30 read as *saade chaar baje* or "four thirty"
3	ID	99 in a phone number read as "double nine"
3	Name	A name said the way its reference recording says it
3	Right language	The Hindi *the* not read as English "the"
3	Same person	The same voice at every change of language
3	Natural reading	No unnatural pause at a change of language
4	Criterion rating	A 1–5 rating for naturalness, pronunciation or speaker similarity
4	Minimal pair	Which of two words that differ by one sound was heard
4	Pairwise order	Which of two systems is better for the same text