Software Requirements Specification (SRS) Document v2

Team 46
Adyansh Kakran
Rohan Girish
Roja Sahoo
Sarthak Chittawar

1. Brief problem statement

The intelligent caller response (ICR) is a multi-organization effort, to create a voice based smart response system capable of answering questions in Indian vernacular languages. The existing system architecture consists of an automated speech recognition (ASR) module, machine translation (ML) module, a semantic document search engine, and a text-to-speech (TTS) module. The larger project is currently under the data collection and model adaptation phase. The current effort is to develop a module to summarize the responses from the document search engine, to produce more “human friendly” responses. This will be limited to the use of existing summarization engines and exploring input content requirements to output quality. The major development is on building the API wrappers to the chosen ML engines and a simple UI for the purposes of demonstration.

2. Users profile

The UI developed is meant only for demonstrative purposes only, hence has one end-user with a specific use case - the dev team to demo to an audience. As such the user must be capable of using the system by providing input text and receiving output summarised text, through a clean selfexplanative UI. The backend API for the summarization module is however meant to be compatible for future incorporation to the larger ICR system, as such must be very well documented in terms of expected input & output, deployment instructions, and finally the choice and performance of ML model (to allow for future ML model adaptations).

3. Project Modules

Summarization Module:

This is the module our team must work on. It receives the documents from the subtl.ai module, summarizes them to derive the answer to the question and relays it to the Machine Translation module to convert it to Telugu (from English, which our module replies in) after which it is relayed to the Text To Speech module which gives the answer. It consists of the following sub-modules:

a. The API sub-module:

Handles the receiving of documents from the subtl.ai module and the relaying of the summarized text back to the MT module.

Features:

1. Receive input text and send it to the summarizer model.
2. Receive output text from the model and send it to the MT model.

b. The Summarizer Model:

The ML model that actually summarizes the text (to be found online not created from scratch).
Features:
1. Generate summarized text from given documents.

c. The Interface:

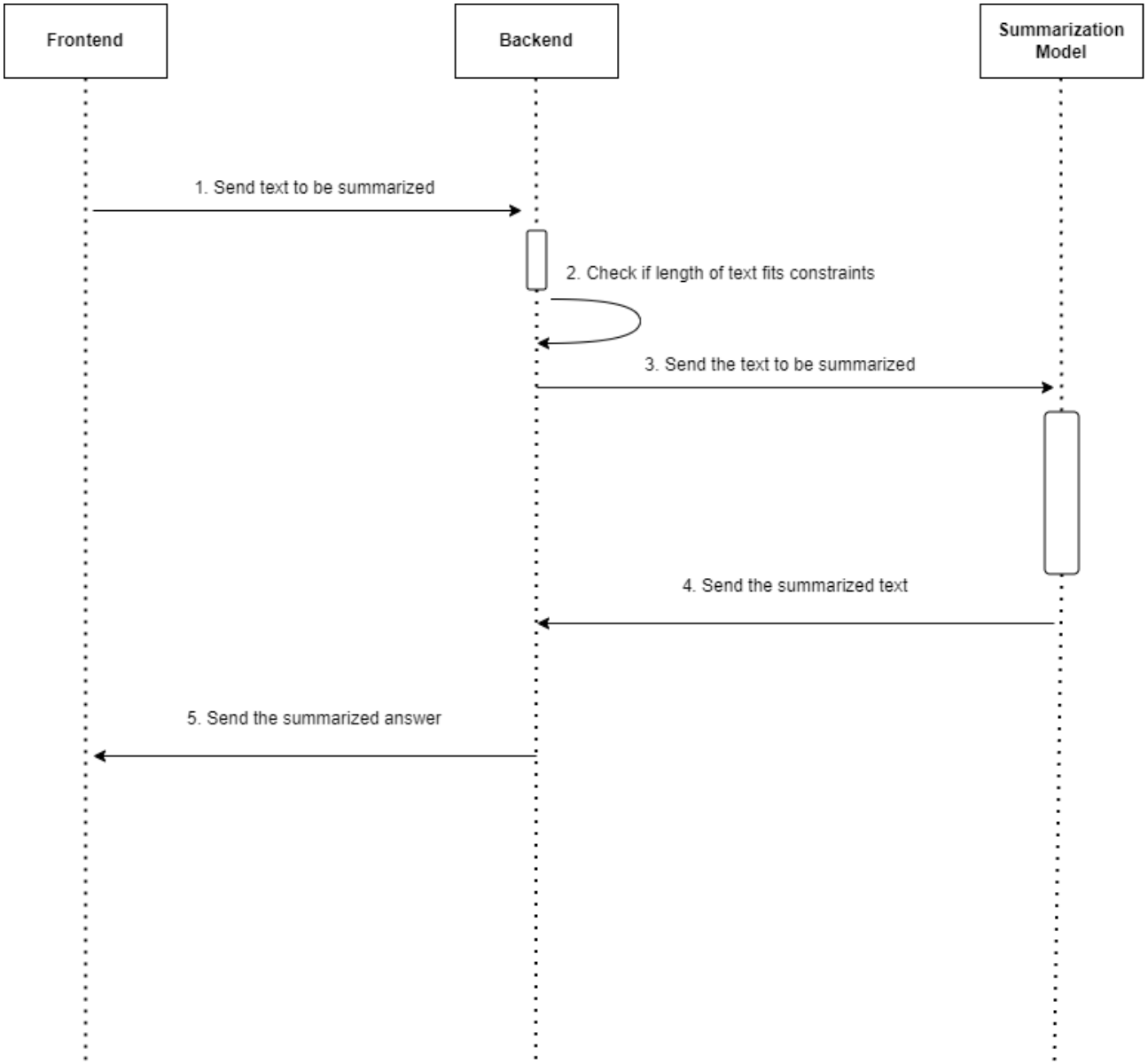
A small, 1-page interface for the model, mostly for testing purposes.

Features:
1. Enter input text.
2. Perform API calls.
3. Display output text.

4. Feature requirements (described using use cases)

1. Client testing
    a. R2
    b. Setup the frontend and the backend of the system. The frontend should consist of a text box for input, a submit button and some form of validation for the user (a loading screen) that the model is processing the input. The backend should allow for a POST request and pass the user input unfiltered to the model for summary generation, which is then returned to the user. It must also validate the size of the data and determine whether summarization is required or not. If required, pass it to the model. Else, return it unadulterated to the user. An appropriate model must also be selected which gives good generalized summaries so that even school students can understand it.

5. Use Case Diagram


Frontend
Backend
Summarization Model
1. Send text to be summarized
2. Check if length of text fits constraints
3. Send the text to be summarized
4. Send the summarized text
5. Send the summarized answer