

# Product Design

Team 46  
Adyansh Kakran  
Rohan Girish  
Roja Sahoo  
Sarthak Chittawar

## 1. Introduction

This document provides a level-by-level description of our software product. We have made a website with a text summarization engine, part of the Intelligent Caller Response pipeline. This document aims to understand the design of our website and its functionality. It explains why we structured our website in the way we did.

## 2. System Overview

The current effort is to develop a module to summarize the responses from the document search engine of Intelligent Caller Response 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.

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 self-explanative 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. Design Overview

### a. Architectural design

The code consists of a React Frontend, Flask Backend, and Python Model code. A user enters input text on the front website, and this input text is collected from the website using the backend(GET) and given as input to the text summarization model. Then the model which has been imported using a python library generates the summarized output text, which is returned to the backend code and published on the website(POST).

The front end deals with UI and making it intuitive for the user. It consists of two simple text boxes and a submit button(Input->Submit->Output). The back-end code is the API

wrapper that connects the website and the text summarizer. The model is the actual text summarizer ML model, distilBART that performs the text summarization.



## b. System interfaces

### i. User Interface

The User Interface is implemented using react JS. It consists of a textbox where users can input their text, a submit button which when clicked sends a request to the backend which returns the summarized text to the frontend. While the frontend waits for the backend, the submit button is disabled and turned into a loader to ensure that multiple requests can't be made from the frontend simultaneously and to ensure the user that their request was received and undergoing processing. Once the output arrives, the input is cleared and the output is displayed below the text box.

#### Text Summarizer

##### Input Text

Text to Summarize

SUBMIT

##### Output Summary

## Text Summarizer

### Input Text

Text to Summarize

Adolf Hitler (German: [ˈadɔlf ˈhɪtlɐ] (listen); 20 April 1889 – 30 April 1945) was an Austrian-born German politician who was the dictator of Germany from 1933 until his suicide in 1945. He rose to power as the leader of the Nazi Party,[a] becoming the chancellor in 1933 and then taking the title of Führer und Reichskanzler in 1934.[b] During his dictatorship, he initiated World War II in Europe by invading Poland on 1 September 1939. He was closely involved in military operations throughout the war and was central to the perpetration of the Holocaust: the genocide of about six million Jews and millions of other victims.

Hitler was born in Braunau am Inn in Austria-Hungary and was raised near Linz. He lived in Vienna later in the first decade of the 1900s and moved to Germany in 1913. He was decorated during his service in the German Army in World War I. In 1919, he joined the German Workers'

SUBMIT

### Output Summary

## Text Summarizer


### Input Text

Text to Summarize

Adolf Hitler (German: [ˈadɔlf ˈhɪtlɐ] (listen); 20 April 1889 – 30 April 1945) was an Austrian-born German politician who was the dictator of Germany from 1933 until his suicide in 1945. He rose to power as the leader of the Nazi Party,[a] becoming the chancellor in 1933 and then taking the title of Führer und Reichskanzler in 1934.[b] During his dictatorship, he initiated World War II in Europe by invading Poland on 1 September 1939. He was closely involved in military operations throughout the war and was central to the perpetration of the Holocaust: the genocide of about six million Jews and millions of other victims.

Hitler was born in Braunau am Inn in Austria-Hungary and was raised near Linz. He lived in Vienna later in the first decade of the 1900s and moved to Germany in 1913. He was decorated during his service in the German Army in World War I. In 1919, he joined the German Workers'

SUBMIT

  
Loading

## Text Summarizer

### Input Text

Text to Summarize

SUBMIT

### Output Summary

Adolf Hitler was an Austrian-born German politician who was the dictator of Germany from 1933 until his suicide in 1945. During his dictatorship, he initiated World War II in Europe by invading Poland on 1 September 1939. He was central to the perpetration of the Holocaust: the genocide of about six million Jews and millions of other victims. In 1923, he attempted to seize power in a failed coup in Munich and was imprisoned with a sentence of five years.

## ii. APIs

We have 1 API POST request which will be exposed, with which the user can send a request. The 'Content-Type' header must be set to 'application/json', and the body set to a json object with 1 attribute, text, whose value is set to the text whose summary the user needs. The API returns the summarized text in the response.

### iii. Model

Our project does not consist of any classes in the traditional sense but consists of Javascript/Python modules which can be run on the server and might be running on the frontend as well.

#### 1. Backend

State -

- This module consists of information regarding the model and downloads the model from the hugging face transformers package on start up of the server.

Functions -

- API endpoint summarize - This is a POST API request that takes in text as the body of the request and returns the summarized text after passing it to the summarize function.
- Summarize function - takes in input text as a parameter and returns the summarized text after passing to the distilBART model downloaded from hugging face.

#### 2. Frontend

State -

- The main state that the frontend stores is the text that is being displayed and has to be given to the backend to be summarized.
- There is also an isLoading state that takes care of the loading phase of the backend and shows it in the frontend.

Functions -

- API call to backend - This is an API call to the backend that gives the displayed text in the body of POST request and gets the data showing it in the output portion of the frontend.

### c. Sequence Diagram(s)

The UML of the whole product's pipeline (Our project is only the summarization part).

### Usage Model and Diagrams (if any)

Fig. 1 shows the overall system architecture for ICR and how the current work is expected to fit in, and Fig. 2 shows a simple architecture of the current work.

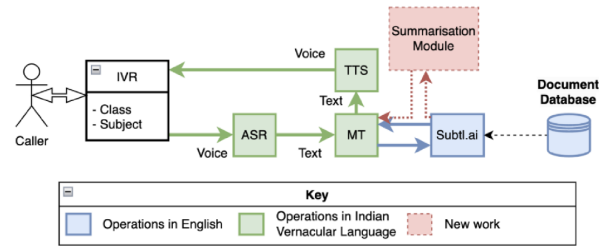


Figure 1: Overall ICR system diagram.

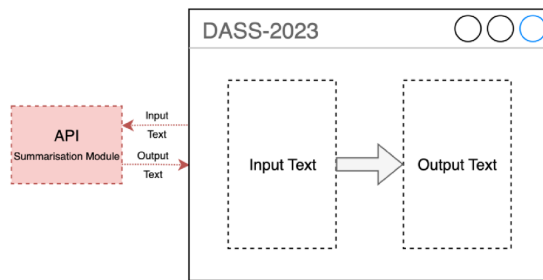
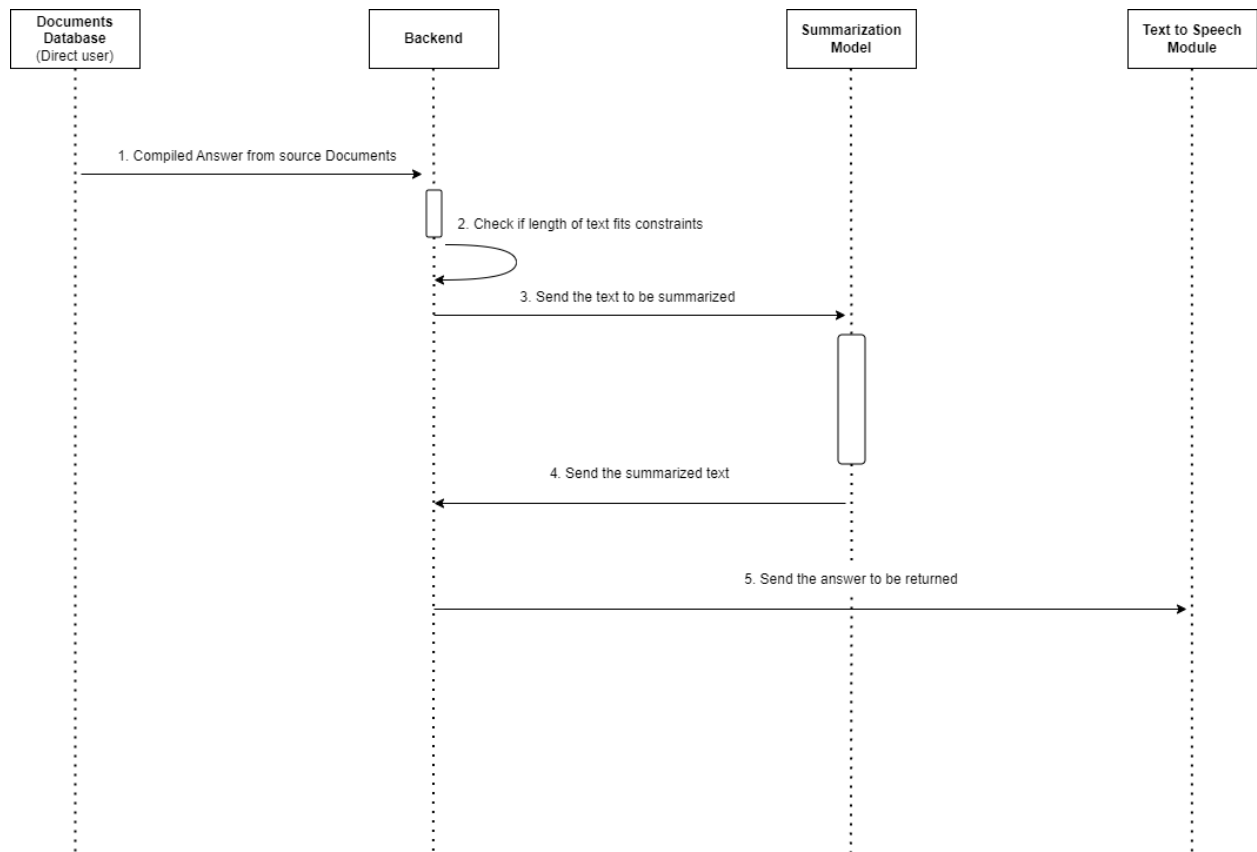
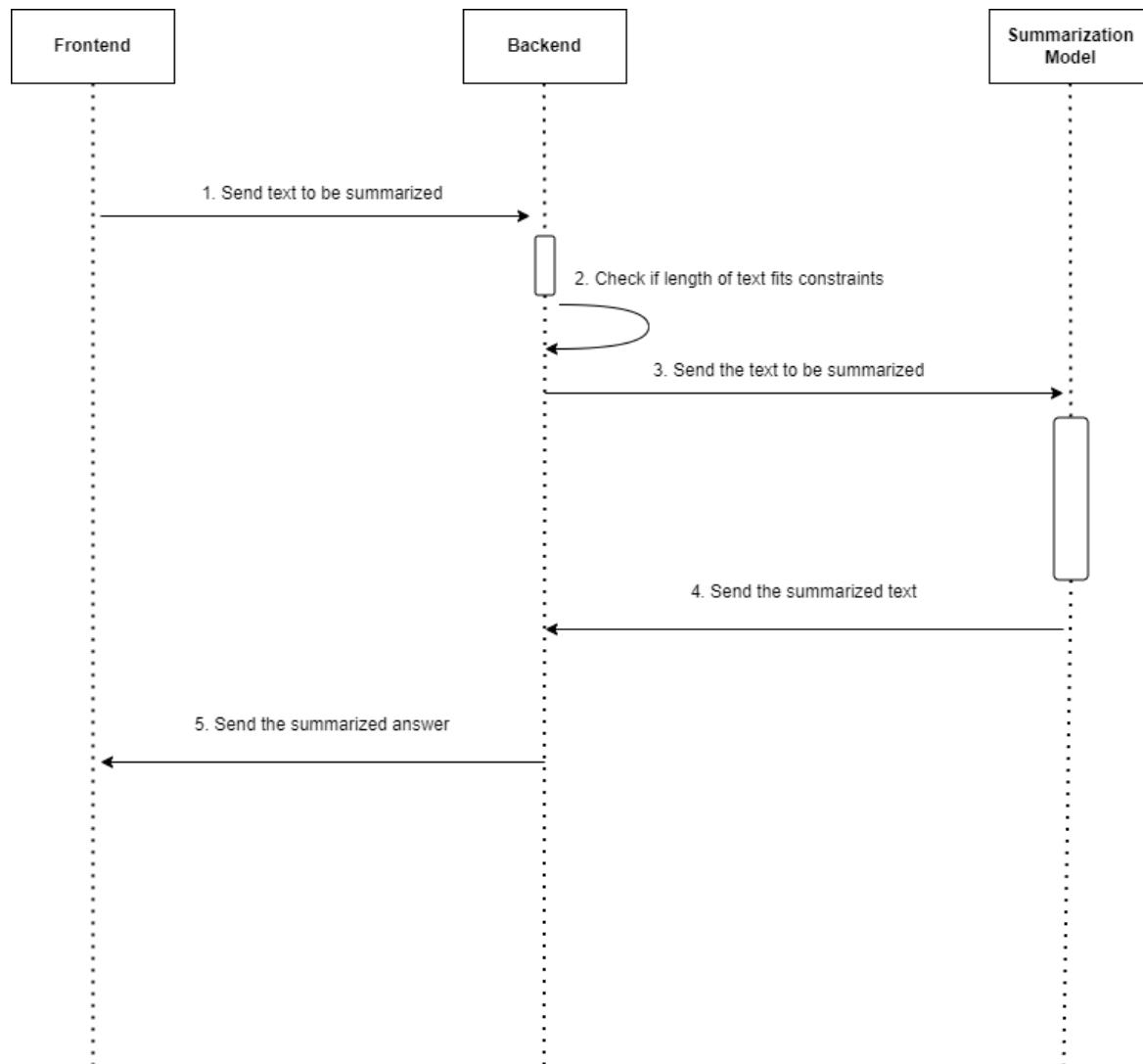


Figure 2: System architecture of current work.

How our module will behave while being implemented in the pipeline:



How we were asked to implement the module by the client (For easier testing):



#### d. Design Rationale

Our design always comprised a simple UI and a backend that connects the summarization model to the website. Initially, if a certain model took too long to load and summarize, we would discard it as it did not make the website efficient. We also had to modify our model code according to the availability of CUDA GPUs to ensure a reasonable running time. We finally found a model for our requirements and integrated it into our pipeline. We also chose to build our backend in Flask as our model was loaded from a Python library.