Software Requirement Specification (SRS) Document

Team 16

Project: Indian Language Sentence Completion (Telugu)

Company: Perceptive Analytics Pvt. Ltd.

Team Members:

1. Shreyansh (2022111002)
2. Siddharth Agarwal (2022101062)
3. Priet Ukani (2022111039)
4. Garvit Gupta (2022101113)

Brief problem statement

This project addresses the challenge of typing in non-English languages, focusing on Telugu. Traditional keyboards lack support for Telugu, making the typing process cumbersome.

To alleviate this issue, the project introduces a machine learning model that facilitates sentence autocomplete functionality, significantly improving the ease and efficiency of typing in Telugu script. So our task is to deploy this ML-based NLP model as a Chrome extension.

The ML model is approximately 600 Mb in size, so downloading the model locally as an extension is infeasible. Hence our task is to deploy the ML model on a cloud platform.

System Requirements

- Server to deploy backend ML model and backend services.
- A chrome extension connected to the backend to fetch user input from live websites and to give prediction outputs.

Users Profile

The target audience for the Chrome extension is the general public and Telugu writing who are seeking a convenient and accessible solution for writing faster with improved efficiency. Users will utilize the app to receive helpful autocomplete writing recommendations.

Example of users:

- **Casual Conversational**: Streamlining digital conversations in Telugu with a more convenient and faster typing experience.
- **Telugu Authors**: Accelerating the writing process for Telugu authors through efficient sentence autocomplete functionality.
- **Students**: Enhancing students' academic endeavors in Telugu by facilitating seamless typing for assignments and note-taking.

It is reasonable to assume that different members of the general public are at different levels of comfort with technology, including with regard to using Chrome extensions. While some people may have little experience with these kinds of technologies, others may be quite skilled at using Chrome web. However, users are expected to be familiar with basic usage and installation of Chrome extensions on computers/laptops and also be able to understand and write Telugu script up to a decent extent.

Project Modules:

Module 1 Deployment of ML Model on server

Module 2 Chrome Extension Module

Below are the specific details for each of the modules, with exact requirements and features list

MODULE 1: Deployment (Backend)

Given the size of the ML Model (>300 MB), it must be deployed on a server. Therefore, in order to deploy the ML Model on the server, we would first need to look for an ideal and practical service offered by the cloud servers (taking into account Logical and Financial aspects). The Module 1 (Prediction Management Module) will communicate with the server through API requests in order to obtain predictions.

No.	Features	Description	Release
1	Loading ML Model to Local Server	ML model is loaded in a python notebook running on a local system	R1
2	Handling API requests from Frontend	Frontend makes a POST request to the server.	R1
3	Inference on model to get predictions out for the requested input text	Model runs on the input got by Post request.	R1

4	Send back predictions via API from Local server to Frontend	Server returns the model output to the frontend.	R1
5	Loading ML Model to Server	The ML Model will be set up in a storage bucket on a server.	R2 (optional)
6	Loading Inference code to Server	Additionally, a server will host the inference code.	R2 (optional)
7	Handling API requests (Invocation)	The prediction input text will be carried by the API request, which will also start the server and all of its operations.	R2 (optional)
8	Loading ML Model to processing site	The ML Model will be loaded to the processing site for prediction inference whenever an API request invokes the server.	R2 (optional)
9	Inference on model to get predictions out for the requested input text	To create predictions, the server would run its inference function after receiving input from the body via the API.	R2 (optional)
10	Send back predictions via API Gateway	The generated prediction would be returned by the server via API Gateway.	R2 (optional)

NOTE: Things are optional in R2 based on the Client giving access to the AWS cloud. (AWS related things were moved from R1 to R2 as Client was unable to give cloud service by this time.)

MODULE 2:Chrome Extension Module (Frontend & Backend)

This module is designed to provide a seamless and user-friendly experience to all users who want to type in telugu script efficiently and accurately. The module is built in the form of a chrome extension, which will serve as the primary interface for users to interact with the ML Model for predictions. The extension will monitor the live web page of users and will provide/display predictions for possible next words for completion of sentence. Overall, the Chrome extension Module is critical in providing users with an accessible means of getting predictions for possible next words.

No.	**Features**	**Description**	**Release**
1	Monitoring	The extension backend must monitor all possible input text fields in the DOM of the web page the user is on and would start fetching input texts being typed, if the user types into any input text field.	R1
2	Detect Telugu	The extension backend must be able to detect when telugu text is	R1

		entered by the user on a webpage.	
3	API Request	Extension backend should send user input to Server using API and get predictions for output through API.	R1
4	Display Suggestions	The extension frontend needs to have the ability to work with DOM and show the user the recommended outputs from the backend.	R1
5	Autocomplete in thin text	The most accurate prediction ought to be displayed in light text that Tab can accept, much like Copilot does in code.	R2
6	Enable user to choose one option out of all predictions being displayed	After the next word predictions get displayed on frontend, the user would be able to choose one of them to proceed further.	R2

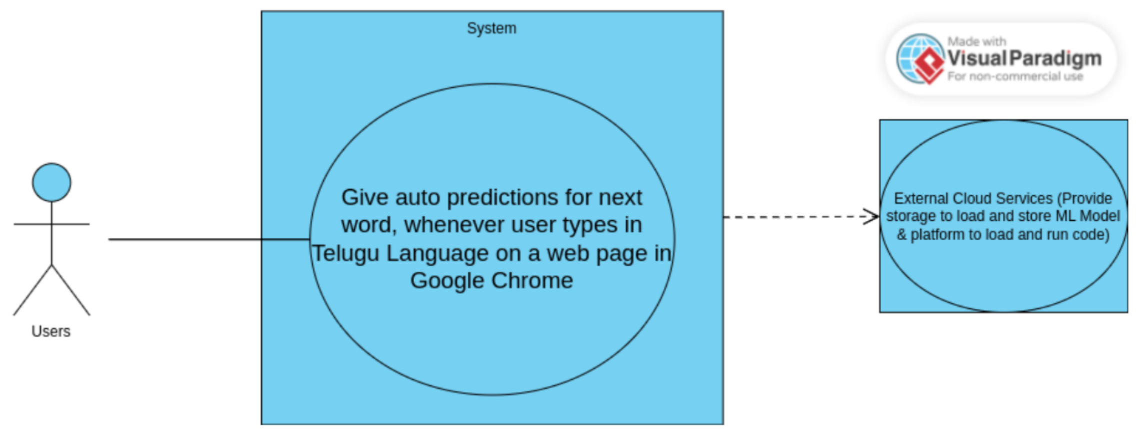
NOTE: Here, things were moved from R2 to R1 as most of the cloud things were moved to R2 and for the need to develop some working part of the project by R1.

USE CASES

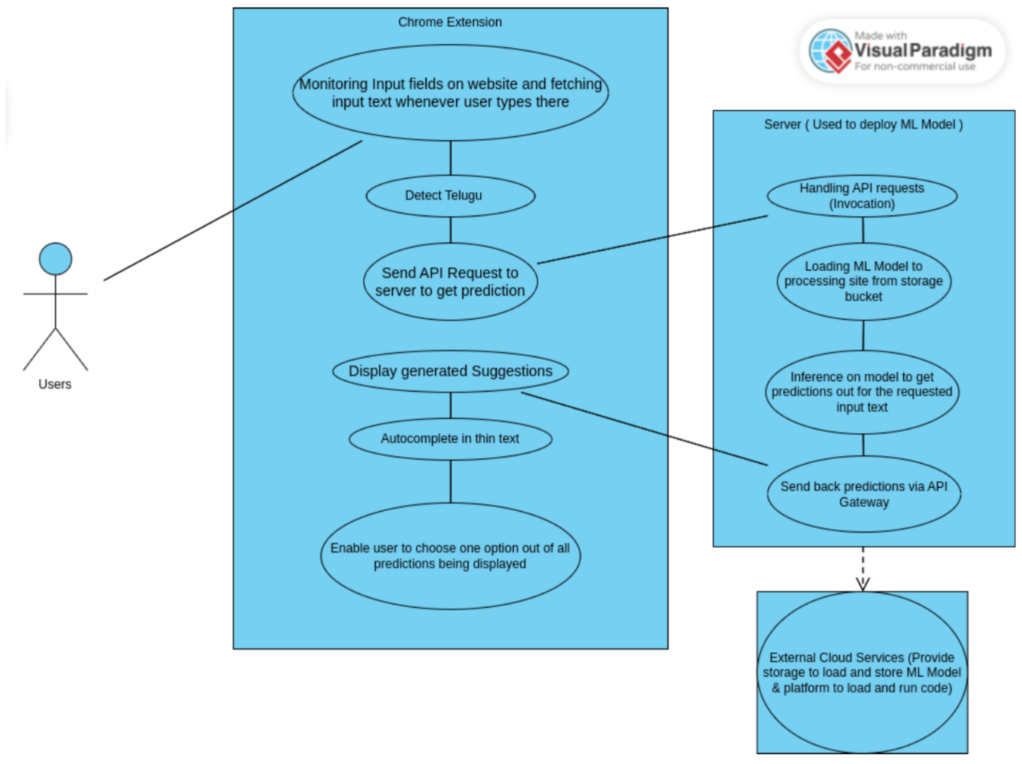
No.	Use Case Name	Description	Release
1	Give auto predictions for next word, whenever user types in Telugu Language on a web page in Google Chrome	To Monitor all input fields on the live website of the user, to fetch input text whenever user types, to identify telugu script, to send API request to server to get predictions for next word for the given input text, to receive predictions for next word and to display them on frontend.	R1

NOTE: This was moved from R2 to R1 for the need to develop some working part of the project by R1.

Use Case Diagram


System
Give auto predictions for next word, whenever user types in Telugu Language on a web page in Google Chrome
Users
Made with Visual Paradigm For non-commercial use
External Cloud Services (Provide storage to load and store ML Model & platform to load and run code)

Flow of the above mentioned Use Case Diagram


Chrome Extension
Monitoring Input fields on website and fetching input text whenever user types there
Detect Telugu
Send API Request to server to get prediction
Display generated Suggestions
Autocomplete in thin text
Enable user to choose one option out of all predictions being displayed
Users
Made with Visual Paradigm For non-commercial use
Server (Used to deploy ML Model)
Handling API requests (Invocation)
Loading ML Model to processing site from storage bucket
Inference on model to get predictions out for the requested input text
Send back predictions via API Gateway
External Cloud Services (Provide storage to load and store ML Model & platform to load and run code)

NOTE: Here the external server will be deployed locally for the time of R1. If client provides AWS services, it will be changed to AWS Cloud Server by the time of R2.

Use Case Description

Use Case Number:	1
Use Case Name:	Suggest Next word when user type in Telugu on a webpage.
Overview:	The overview entails overseeing input fields on the user's live website, capturing typed input text, detecting Telugu script, sending API requests to the server for next-word predictions based on the input text, receiving and displaying these predictions on the frontend.
Actors:	Extension Users
Pre Condition:	Chrome Extension must have been downloaded

Flow:	1. Monitoring input fields of live website & fetching input texts whenever user types in input field. 2. Detecting Telugu Language 2.1) If the fetched text is not Telugu, then start from step 1 2.2) If the fetched text is Telugu, then proceed to step 3 3. Send API Request to server to get prediction for fetched input 4. Server will run inference code on sent input and return generated prediction via API Gateway. Display the received prediction. 4.1) If didn’t get predictions from server before new fetched input, then jump back to step 2 4.2) If did get predictions then proceed to step 5 5. Display the received predictions on frontend. 6. Display auto-complete in thin text 7. Enable user to select one of the displayed predictions by selecting one of them or by pressing TAB.
Post Condition:	Display of predictions of possible next words, for the purpose of sentence completion