# Real-Time SMS Translator (Hindi ↔ Bhojpuri)

## Problem Statement

In regions such as Bihar, Uttar Pradesh, and Jharkhand, Bhojpuri is the primary language of communication for millions of people. However, most official, business, and even educational communications are issued in Hindi. This discrepancy creates a language barrier that makes it difficult for Bhojpuri speakers to fully understand important messages, particularly SMS-based notifications, government alerts, or business-related information. Elderly individuals, rural populations, and less literate communities face challenges in interpreting these messages, often leading to missed opportunities or misunderstandings.   
  
The lack of real-time translation tools specifically targeting Hindi ↔ Bhojpuri communication worsens this issue. While general translation services exist, they often lack contextual accuracy for regional dialects and idiomatic expressions. Thus, there is a pressing need for a dedicated, intelligent, and real-time translation system tailored to Bhojpuri speakers.

## Team Details

Team Name: 17

Team Lead: Durga Prasad (23B21A4591)

Team Members:

• 23B21A4534  
• 23B21A4546  
• 23B21A4563  
• 23B21A4592

## Introduction

The Real-Time SMS Translator (Hindi ↔ Bhojpuri) is a web-based application designed to reduce linguistic barriers and enhance communication in multilingual regions of India. Hindi and Bhojpuri, though linguistically related, have notable differences in vocabulary, syntax, and tone. Many Bhojpuri speakers, particularly in rural and semi-urban areas, find it difficult to interpret official or professional communications written in Hindi.   
  
This project addresses the need for an accessible, user-friendly platform that allows individuals to input, translate, and view SMS or text-based messages in both Hindi and Bhojpuri instantly. By leveraging Artificial Intelligence (AI) and Natural Language Processing (NLP) models, the system ensures high accuracy while maintaining readability, tone, and context.

## Objectives

The objectives of the Real-Time SMS Translator project are as follows:

• Develop a web-based platform for seamless Hindi ↔ Bhojpuri text translation.

• Enable real-time translation for fast and accurate message interpretation.

• Support offline usage by integrating browser caching and on-device models.

• Maintain strong data privacy and security standards.

• Improve digital accessibility for rural and semi-urban communities.

## Proposed Solution

The proposed solution is a web application that allows users to input SMS text, official communication, or free-form text and receive instant Hindi ↔ Bhojpuri translations. The system automatically detects the source language and processes it through an AI-powered translation engine based on IndicTrans2 and HuggingFace NLP models. The translated text is then displayed alongside the original for easy comparison.  
  
The application will also provide a history log, favorites for saving important translations, and a clean, minimal UI that caters to both tech-savvy and non-technical users. Offline support is achieved using service workers and local caching, ensuring that the platform remains usable in low-connectivity environments common in rural areas.

## Features

• Bi-directional Hindi ↔ Bhojpuri translation for text messages and SMS.

• Real-time text processing and translation.

• Copy, save, or share translated content.

• Offline translation support through browser storage and caching.

• History and favorites for frequently accessed messages.

• Simple, intuitive user interface designed for all age groups.

## Technology Stack

• Frontend: React.js / Next.js with Tailwind CSS for responsive UI.

• Backend: Node.js + Express / Python Flask or FastAPI.

• Database: MongoDB / Firebase for storage and sync.

• AI Models: IndicTrans2, HuggingFace Transformers, Google ML Kit.

• Web Integration: REST APIs, WebSockets for live translation.

## System Design

Flow of operations:  
Users → Web UI (React) → Language Detector → Translation Engine → Output Display  
  
The system listens for text input from the user, applies a language detector to identify Hindi or Bhojpuri, routes the content to the translation model, and updates the user interface in real time with the translated result. Additionally, the backend logs translations for history and allows users to save important outputs.

## Implementation Steps

1. Build web-based input/output interface using React.js.

2. Develop backend API to handle translation requests.

3. Integrate IndicTrans2/HuggingFace model for translations.

4. Implement automatic language detection service.

5. Add features such as history, favorites, and copy/share options.

6. Ensure offline usability with service workers and local storage.

7. Test system with real Hindi ↔ Bhojpuri communication data.

## Weekly Implementation Plan

Week 1: Research & Setup  
- Finalize project requirements, datasets, and technology stack.  
- Set up GitHub repository and configure React + Node.js/Flask environment.  
- Conduct background research on IndicTrans2 and other relevant models.  
  
Week 2: Web UI Development  
- Create the text input and output interface with React.js.  
- Implement Tailwind CSS for responsive design.  
- Add initial backend connection for test data.  
  
Week 3: Language Detection Module  
- Integrate a simple NLP-based language detector.  
- Train/test it on Hindi vs Bhojpuri datasets.  
- Connect detection with the input pipeline.  
  
Week 4: Translation Engine Integration  
- Set up IndicTrans2 or HuggingFace translation model.  
- Deploy backend API to handle translation requests.  
- Test end-to-end pipeline (Input → Detection → Translation → Output).  
  
Week 5: Feature Expansion  
- Implement history and favorites feature with database integration.  
- Add options to copy, save, or share translated text.  
- Provide toggle view for original vs translated content.  
  
Week 6: Offline Mode & Security  
- Implement browser caching and service workers for offline usage.  
- Add encryption for data storage and secure API communication.  
- Optimize performance for low-bandwidth conditions.  
  
Week 7: Testing & Debugging  
- Conduct testing with real SMS/text datasets.  
- Debug errors in translation accuracy, idioms, and usability issues.  
- Collect feedback from users for refinement.  
  
Week 8: Deployment & Presentation  
- Deploy the application on Firebase Hosting or Vercel.  
- Document all modules and prepare project report.  
- Final presentation with demo.

## Challenges

• Achieving accurate translation for idioms and local expressions.

• Managing tone, readability, and context during translations.

• Providing smooth offline functionality in web browsers.

• Ensuring end-to-end data privacy and security.

## Results

• A working prototype of a web-based translation platform.

• Real-time Hindi ↔ Bhojpuri text translation demonstrated successfully.

• Offline support and caching available for low-connectivity users.

• History and favorites features implemented with functional UI.

## Future Scope

• Expand support to additional regional languages such as Maithili, Magahi, and Awadhi.

• Build browser extensions for instant translation on any webpage.

• Introduce speech-to-text and voice-based translation features.

• Incorporate contextual AI translation models for tone preservation.

## Conclusion

The Real-Time SMS Translator (Hindi ↔ Bhojpuri) web application highlights the potential of AI to reduce linguistic barriers and empower millions of regional language speakers. By providing an intuitive, accessible, and offline-capable solution, the project enhances inclusivity and digital literacy in rural areas. With real-time translation, history, and user-friendly features, this system proves that AI can effectively bridge the gap between Hindi and Bhojpuri speakers, ensuring no critical communication is lost due to language differences.

## References

• IEEE & Springer research papers on NLP and regional translation.

• HuggingFace Transformers and IndicTrans2 documentation.

• Google ML Kit references for language models.

• Firebase and React.js documentation.