**Software Requirements Specification (SRS)**

**English → Hindi Language Translator Web App**

---

## 1. Introduction

### 1.1 Purpose

- To develop a web-based application that translates English text into Hindi.
- The system uses the NLP model **Helsinki-NLP / opus-mt-en-hi**.
- Backend developed using Python Flask.
- Deployment on **Render** cloud platform.

### 1.2 Scope

- Users can enter English text.
- System processes input using NLP model.
- Displays translated Hindi text instantly.
- Accessible via web browser (desktop/mobile).

### 1.3 Technology Stack

- **Frontend:** HTML, CSS, Bootstrap
- **Backend:** Python + Flask
- **NLP Model:** Helsinki-NLP/opus-mt-en-hi
- **Libraries:** Transformers, Torch
- **Deployment:** Render

---

## 2. Overall Description

### 2.1 Product Perspective

- Standalone web application.
- Hosted online using cloud deployment.
- Uses pre-trained NLP transformer model.

### 2.2 Product Functions

- Accept English input text.
- Validate input.
- Process text through translation model.
- Display Hindi output.

- Handle errors (empty input, server issues).

**2.3 User Classes**

- Students
- Professionals
- General users
- Researchers

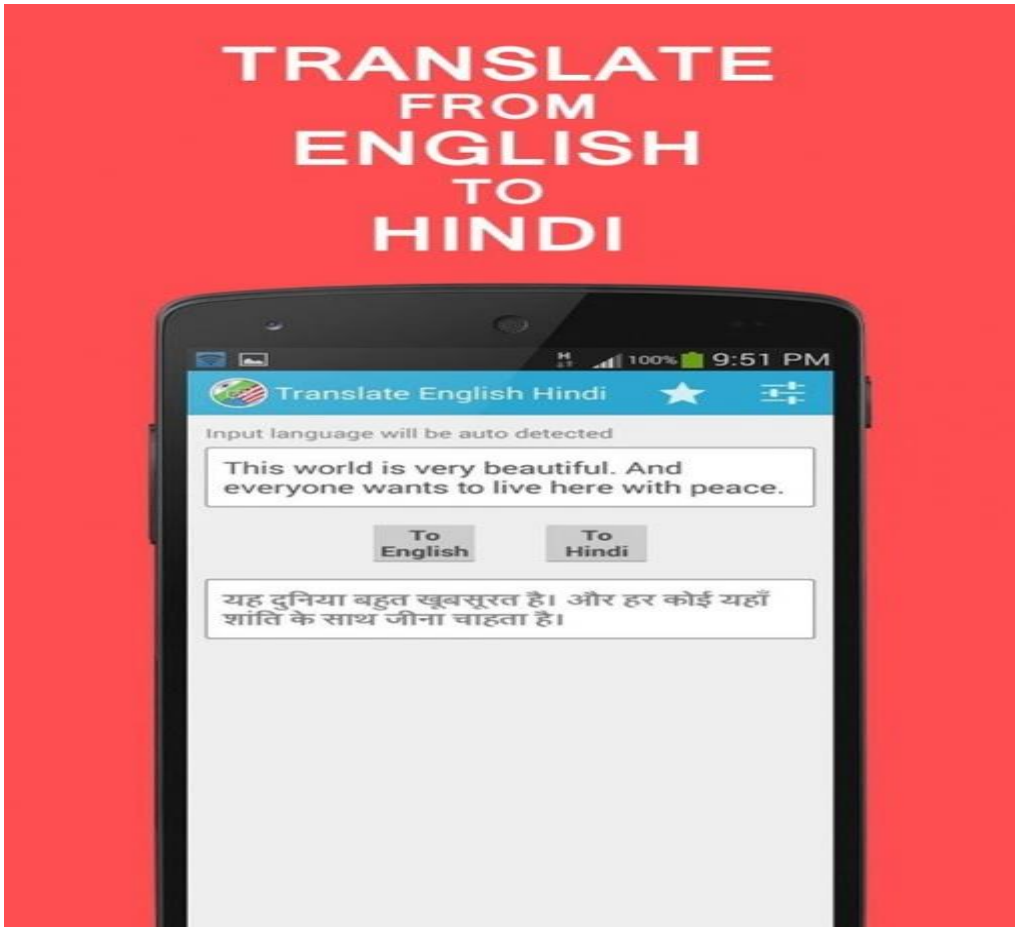
**2.4 Operating Environment**

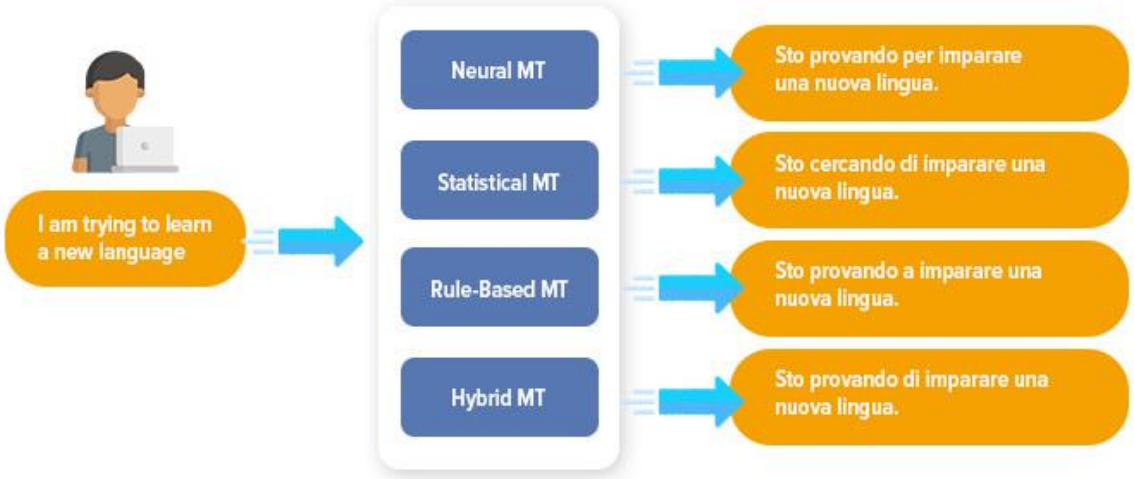
- Web browser (Chrome, Edge, Firefox).
- Python 3.x environment.
- Render cloud hosting.

---

**3. System Features**

**3.1 Text Translation Feature**




4

**Description**

- User enters English sentence in text area.
- Clicks "Translate" button.
- System displays translated Hindi text.

**Functional Requirements**

- FR1: System shall accept English text input.
- FR2: System shall load NLP model at server startup.
- FR3: System shall process text using opus-mt-en-hi model.
- FR4: System shall display Hindi translation output.
- FR5: System shall handle empty input with warning message.

---

**4. External Interface Requirements**

**4.1 User Interface**

- Simple text input box.
- Translate button.

- Output display section.
- Responsive design.

**4.2 Hardware Interface**

- No special hardware required.

**4.3 Software Interface**

- Python
- Flask Framework
- Transformers Library
- Torch

---

**5. Non-Functional Requirements**

**5.1 Performance**

- Response time < 5 seconds (depending on server load).

**5.2 Security**

- Input validation to prevent malicious scripts.
- HTTPS deployment on Render.

**5.3 Usability**

- Clean and simple interface.
- Easy navigation.

**5.4 Reliability**

- 99% uptime (based on hosting plan).

**5.5 Scalability**

- Can upgrade Render instance for higher traffic.

---

**6. System Architecture**

- User sends request via browser.
- Flask server receives request.
- NLP model processes translation.
- Result returned to frontend.

**Architecture Type:** Client-Server Architecture

---

**7. Deployment Details**

- Push code to GitHub repository.
- Connect repository to Render.
- Configure build command:

pip install -r requirements.txt

- Start command:

gunicorn app:app

- Deploy as Web Service.

---

**8. Future Enhancements**

- Add Hindi → English translation.
- Add voice input feature.
- Add multiple language support.
- Add translation history storage.

---

✅ **Conclusion**

This system provides an efficient English-to-Hindi translation solution using a transformer-based NLP model deployed as a scalable web application. It ensures ease of use, fast translation, and reliable cloud deployment.