

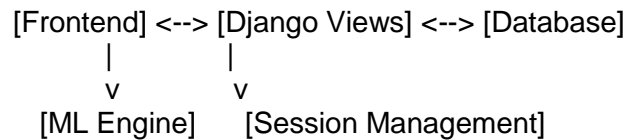
# Technical Document: Hindi Idioms to English Translation

## Table of Contents

1. **Overview**
  - Project Summary
  - Key Components and Workflow
2. **Project Directory Structure**
  - Breakdown of Project Folders and Files
3. **Main Workflow**
  - Data Preparation
  - Model Fine-Tuning
  - Translation Inference
  - Web Application Interface
4. **Detailed Workflow and Flowcharts**
  - Data Preparation Flowchart
  - Model Fine-Tuning Flowchart
  - Translation Inference Flowchart
  - Web Application Workflow Flowchart
5. **Component Details**
  - Dataset Preparation (dataset\_maker.py)
  - Translation Model (inference.py)
  - Django Web Application Components
    - Forms (forms.py)
    - Views (views.py)
    - URLs (translation\_project/urls.py)
    - Templates (templates/translate.html)
6. **Security Considerations**
  - CSRF Protection
  - ALLOWED\_HOSTS Configuration
  - Input Validation
  - HTTPS
7. **Logging and Debugging**
  - Logging Setup
  - Debugging Techniques
8. **Future Improvements**
  - Advanced NLP Models
  - Persistent Sessions
  - Admin Dashboard
  - Multilingual Support
9. **Conclusion**
  - Project Summary and Future Scalability

# Overview

This document outlines the architecture, components, and workflow for a project designed to translate Hindi idioms into English using a fine-tuned MarianMT model. The system integrates with a Django-based web application for end-user interaction.



## Project Directory Structure

HINDI\_IDIOMS\_MODEL\_TRANSLATION/

```
|
|-- dataset/          # Contains datasets used for training and evaluation
| |-- raw_data.csv
| |-- cleaned_data.csv
|
|-- fine_tuned_model/  # Stores the fine-tuned MarianMT model
| |-- opus-mt-hi-en/
|   |-- config.json
|   |-- pytorch_model.bin
|   |-- tokenizer_config.json
|
|-- translation_project/ # Django project folder
|   |-- translation_project/ # Django project configuration files
|   |-- __init__.py
|   |-- asgi.py
|   |-- settings.py
|   |-- urls.py
|   |-- wsgi.py
|
|-- translator/        # Django app for translation
|   |-- __init__.py
|   |-- admin.py
```

```
|-- apps.py
|-- forms.py
|-- models.py
|-- urls.py
|-- views.py
|-- migrations/
|-- templates/
    |-- translate.html
|-- translation/
    |-- inference.py
|-- dataset_maker.py      # Script for preparing the dataset
|-- evaluation_model.py   # Script for evaluating the trained model
```

## Main Workflow

The project converts Hindi idioms to English idioms through the following steps:

1. **Data Preparation:**
  - Collect raw data (Hindi idioms and their English translations).
  - Clean and preprocess the data using `dataset_maker.py`.
2. **Model Fine-Tuning:**
  - Fine-tune the MarianMT model on the Hindi idioms dataset.
  - Store the fine-tuned model in the `fine_tuned_model` directory.
3. **Translation Inference:**
  - Use the fine-tuned MarianMT model to translate text.
4. **Web Application Interface:**
  - A Django-based web application allows users to input Hindi idioms and get translations in English.

# Detailed Workflow and Flowcharts

## 1. Data Preparation

The data preparation involves cleaning raw data, tokenizing it, and splitting it into training and evaluation sets.

Flowchart: Data Preparation

**[Start] --> [Load Raw Data] --> [Clean Data] --> [Tokenize] --> [Save Preprocessed Dataset]**

## 2. Model Fine-Tuning

Fine-tune the MarianMT model using the cleaned dataset.

Flowchart: Model Fine-Tuning

**[Start] --> [Load Pre-trained MarianMT Model] --> [Load Preprocessed Dataset] --> [Fine-Tune Model] --> [Save Fine-Tuned Model]**

## 3. Translation Inference

Use the fine-tuned model to perform translations in real time.

Flowchart: Translation Inference

**[Start] --> [Load Fine-Tuned Model] --> [Input Text] --> [Tokenize Input] --> [Generate Translation] --> [Decode Output] --> [Return Translation]**

## 4. Web Application Workflow

The Django application serves as the interface for users to input Hindi idioms and view translations.

Flowchart: Web Application Workflow

**[User Input] --> [Django Translator App] --> [Form Validation] --> [Call Translation Inference Function] --> [Render Result on Template]**

# Component Details

## 1. Dataset Preparation

**Script:** dataset\_maker.py

This script performs:

- Loading and cleaning raw Hindi idioms dataset.
- Tokenizing and saving the processed data.

## 2. Translation Model

**Directory:** fine\_tuned\_model/

**Script:** inference.py

The inference.py script:

- Loads the fine-tuned MarianMT model and tokenizer.
- Defines the translate\_text() function to handle text translation.

## 3. Django Web Application

The web application is implemented using Django, with the following key components:

### a. Forms

**File:** forms.py

Defines the form for user input.

### b. Views

**File:** views.py

Handles user input, calls the translation function, and renders results.

### c. URLs

Translation\_project/urls.py

### d. Templates

**File:** templates/translate.html

HTML file for rendering the form and translation result.

## Security Considerations

1. **CSRF Protection:** Enabled for form submissions.
2. **ALLOWED\_HOSTS:** Ensure the `ALLOWED_HOSTS` setting is configured properly.
3. **Input Validation:** Validate user inputs to avoid SQL injection and XSS attacks.
4. **HTTPS:** Use HTTPS in production to encrypt communications.

## Logging and Debugging

**Logging:** Configured using Python's `logging` module.

```
logger = logging.getLogger(__name__)
```

- 
- **Debugging:**
  - Use `print` statements for quick debugging.
  - Check `django.log` for detailed logs.

## Future Improvements

1. **Advanced NLP:** Integrate more advanced NLP models like BERT.
2. **Persistent Sessions:** Store chat sessions in the database for persistence.
3. **Admin Dashboard:** Allow admins to manage Q/A pairs dynamically.
4. **Multilingual Support:** Enable chatbot responses in multiple languages.

## Conclusion

This project integrates machine translation capabilities with a web-based interface to provide accurate and user-friendly translations of Hindi idioms into English. The modular design ensures easy maintenance and scalability for future enhancements.