

# **Project Documentation Template**

**Project Title: Text Summarizer Using Asyncflows and Gradio**

---

## **Table of Contents**

- 1. Introduction**
  - 2. Problem Statement**
  - 3. Solution Overview**
  - 4. Target Audience**
  - 5. Project Setup**
    - **Prerequisites**
    - **Installation**
    - **Running the Project**
  - 6. Flow Design**
    - **Flow Diagram**
    - **Flow Description**
  - 7. Custom Action**
    - **Description**
    - **Code**
    - **Integration**
    - **Reusability**
  - 8. User Interface**
    - **UI Design**
    - **Gradio Implementation**
  - 9. How to Use**
  - 10. Use Cases**
  - 11. Demo Video**
  - 12. Conclusion**
  - 13. Future Work**
  - 14. References**
-

## Introduction

The project aims to provide a user-friendly interface for summarizing text inputs using advanced natural language processing techniques. It utilizes Asyncflows for managing asynchronous actions and Gradio for creating an interactive UI, enabling easy text summarization for various applications.

---

## Problem Statement

Manual text summarization is time-consuming and prone to errors, especially with large volumes of text. Automated solutions are needed to efficiently condense textual content while preserving key information and context.

---

## Solution Overview

Our solution leverages Asyncflows for handling asynchronous tasks within the summarization pipeline. It integrates Gradio to offer an intuitive web-based interface where users can input text and receive concise summaries promptly.

---

## Target Audience

- **Researchers:** To quickly extract key findings from academic papers.
  - **Students:** To summarize study materials and lecture notes.
  - **Professionals:** To generate briefs and reports from extensive documents.
  - **Developers:** To integrate automated text summarization into other applications.
- 

## Project Setup

### Prerequisites

- Python 3.8+
- Asyncflows
- Gradio
- NLTK (Natural Language Toolkit)

- Other dependencies as specified in `requirements.txt`

## Installation

Provide step-by-step instructions for setting up the development environment and installing necessary dependencies.

Example:

```
# Clone the repository
git clone https://github.com/Kirai-Kevin/textsum.git
```

```
# Navigate to the project directory
cd textsum
```

```
# Install dependencies
pip install -r requirements.txt
```

## Running the Project

Explain how to run the project.

Example:

```
# Start the application
python main.py
```

---

## Flow Design

### Flow Diagram

Include your flow diagram here depicting the steps involved in the text summarization process using Asyncflows.

### Flow Description

1. **Input Handling:** Accepts input from users via a Gradio interface or text file upload.

2. **Text Processing:** Cleans and preprocesses the input text to remove noise and irrelevant content.
  3. **Asynchronous Summarization:** Utilizes Asyncflows to asynchronously summarize the preprocessed text using NLP models.
  4. **Output Generation:** Generates a summarized version of the input text suitable for quick review and understanding.
- 

## Custom Action

### Description

The custom action focuses on asynchronously summarizing text inputs using Asyncflows for efficient execution.

### Code

```
from asyncflows import action

from transformers import pipeline


@action
async def summarize_text_async(text):

    summarizer = pipeline("summarization")

    summary = summarizer(text, max_length=150, min_length=30,
do_sample=False)[0]['summary_text']

    return summary
```

### Integration

Integrate the custom action into the Asyncflows-based flow for text summarization:

```
from asyncflows import Flow

flow = Flow()
```

```
@flow.action
```

```
async def text_summarization_flow(input_text):
```

```
    cleaned_text = preprocess_text(input_text)
```

```
    summary = await summarize_text_async(cleaned_text)
```

```
    return summary
```

## Reusability

The `summarize_text_async` action can be easily reused in other projects or workflows that require text summarization capabilities. It abstracts the NLP model interaction and summarization logic, making it modular and adaptable.

---

## User Interface

### UI Design

Describe the design and layout of the user interface. Include screenshots if available.

### Gradio Implementation

Explain how Gradio was used to create the user interface for interacting with the project:

```
import gradio as gr
```

```
def summarize_text_interface(input_text):
```

```
    # Integration with Asyncflows-based main function
```

```
    summary = text_summarization_flow(input_text)
```

```
    return summary
```

```
iface = gr.Interface(
```

```
    fn=summarize_text_interface,
```

```
    inputs=gr.Textbox(lines=10, placeholder="Paste text here..."),
```

```
        outputs=gr.Textbox(placeholder="Summarized text will appear  
here..."),  
        title="Text Summarizer",  
        description="Paste text or upload a document to summarize its  
contents."  
    )  
  
iface.launch()
```

---

## How to Use

1. **Start the Application:** Run `python main.py` to launch the application.
  2. **Input Data:** Enter or paste text into the provided textbox.
  3. **Generate Output:** Click on the "Summarize" button to initiate the text summarization process.
  4. **View and Download Output:** The summarized text will be displayed. Optionally, download the output for further use.
- 

## Use Cases

1. **Academic Research:** Extract key findings and conclusions from research papers.
  2. **Educational Purposes:** Summarize study materials and lecture notes for better understanding.
  3. **Professional Reports:** Quickly generate executive summaries from lengthy documents.
- 

## Demo Video

Include a link to your demo video. The video should demonstrate the key features of your project and explain how it works.

[Demo Video Link](#)

---

## Conclusion

The Text Summarizer project demonstrates effective use of Asyncflows and Gradio to automate text summarization tasks, providing users with an efficient solution for information extraction from textual content.

---

## Future Work

- **Enhanced NLP Models:** Integrate advanced NLP models for improved summarization accuracy and context understanding.
  - **Multilingual Support:** Extend text summarization capabilities to handle documents in multiple languages.
  - **Customization Options:** Provide more options for user-defined summarization preferences and settings.
  - **Integration with External APIs:** Incorporate APIs for additional data enrichment and specialized summarization tasks.
- 

## References

Asyncflows Documentation: <https://asyncflow.readthedocs.io/en/latest/>

Gradio Documentation: <https://www.gradio.app/docs>

NLTK Documentation: <https://www.nltk.org/>

Transformers Documentation: <https://huggingface.co/docs/transformers/en/index>

---

Use this template to document your project thoroughly. Ensure that each section is detailed and provides a clear understanding of your work.