**Title: AI-Powered Content Summarizer Backend**

**Introduction**

This document provides a brief overview of the AI-powered content summarizer backend, which is a service that leverages AI to summarize content typed by the user. The service is built using Python and Flask as the backend framework, and it interacts with Elasticsearch for storing and retrieving content metadata and uses SQL database for user data management.

**Repository Structure**

***summarizer.py***:

* Contains the Flask application code, including the API endpoints and the integration with the AI model.
* Contains the SQL database models for managing user data.
* Contains unit tests for the API endpoints and the database models.

***requirements.txt***: Contains the required dependencies for the project.

***app.py***: Streamlit code for frontend UI.

***Chroma\_db*** : Vector Database, Contains user input text, embedding and metadata

***UserDb.db*** : Sqlite Database, Contains user details.

**Getting Started**

To get started with the project, follow these steps:

Clone the repository: Clone the repository to your local machine using the following command:

bash

git clone https://github.com/noviljohnson/TextSummarizer.git

Create a virtual environment: Create a virtual environment for the project using the following

Command: conda create -n envname python=3.10.9

Activate the virtual environment: Activate the virtual environment using the following

Command: conda activate envname

Install the dependencies: Install the required dependencies for the project using the following

Command: pip install -r requirements.txt

Run the application: Run the application using the following command:

> python summarizer.py # starts flask server

# in another cmd

> streamlit run app.py

**API Endpoints**

The following API endpoints are available in summarizer.py :

GET / login : Checks credentials of the user to login.

* Inputs : user email, password

GET / update\_user : Creates a new user.

* Inputs : user email, password, full name

[GET, POST ] / get\_summary :

* Summarizes the content provided in the request body and returns the summary.
* Also returns the answer if the user asks a question instead of text to summarize.
* Inputs : user email, input text

All inputs will be passed in json format

**AI Model Integration**

The AI models used in this project are the **LaMini-Flan-T5-248M** and **Intel/dynamic\_tinybert** model from Hugging Face. It is integrated into the application using the transformers library.

**Database Integration**

The application uses **Chroma** Vector Database for storing and retrieving content metadata and uses **SQLite3** database for user data management.

**Code Quality and Structure**

The code is structured in a way that makes it easy to read and understand. The code is organized into modules, and each module is responsible for a specific functionality. The code is also well-documented, with comments and docstrings that explain the code and its functionality.

LLM Framework : Langchain

Integration Framework : Flask

Summarizer.py contains

* Langchain pipelines for Text summarization and Q&A
* class - Summer()
  + Functions for creating databases (vector and SQL) : create\_db(), addDocs\_2\_vetordb()
  + Functions to add new user : add\_user()
  + Functions to check user credentials : get\_user\_details()
  + Functions to query databases (vector and SQL) : query\_VectorDb(), get\_user\_details()
  + Functions to update databases (vector and SQL) : create\_docs(), addDocs\_2\_vetordb()