Capstone: TDoc Digest Deployment Implementation

Table of content

TDoc Digest: An Interactive web application for generating abstractive summary of a 3GPP TDoc contribution

The TDoc Digest is an interactive web application tool to generate an abstractive summary of a given TDoc contribution. The tool takes the meeting id and TDoc number as inputs and processes the content of the TDoc, i.e., the long original text and produces an abstractive summary of the TDoc along with a semantic score.

Production ready code

The web API application is implemented using the Flask micro web framework. The code is available in the Github repository. The details of the Github repository and the project set up are provided in the Table below.

Github repository link for the Python code	https://github.com/sanjeewaherath1/Bootcamp/tree/2d42078a11effe83fe3631ef8efa300ca8a96bcc/capstone/TDocDigest
Environment variable	OPENAI_API_KEY environment variable should be configured with an open AI Secret Key. Details on how to find your API Key is available in "Where do I find my OpenAI API Key?"
Packages used (Python)	openai, flask, docx2txt, requests, os,shutil, logging, pyngrok, requests, zipfile, io, threading, datetime
Project set up details	An example in JETBRAIN PyCharm project set up is provided in Appendix A. It includes creating the project, setting up the OPENAI_API_KEY and package installation examples

Table 1: Production ready code and project set up example

The folder structure of TDoc Digest Flask application is shown below.

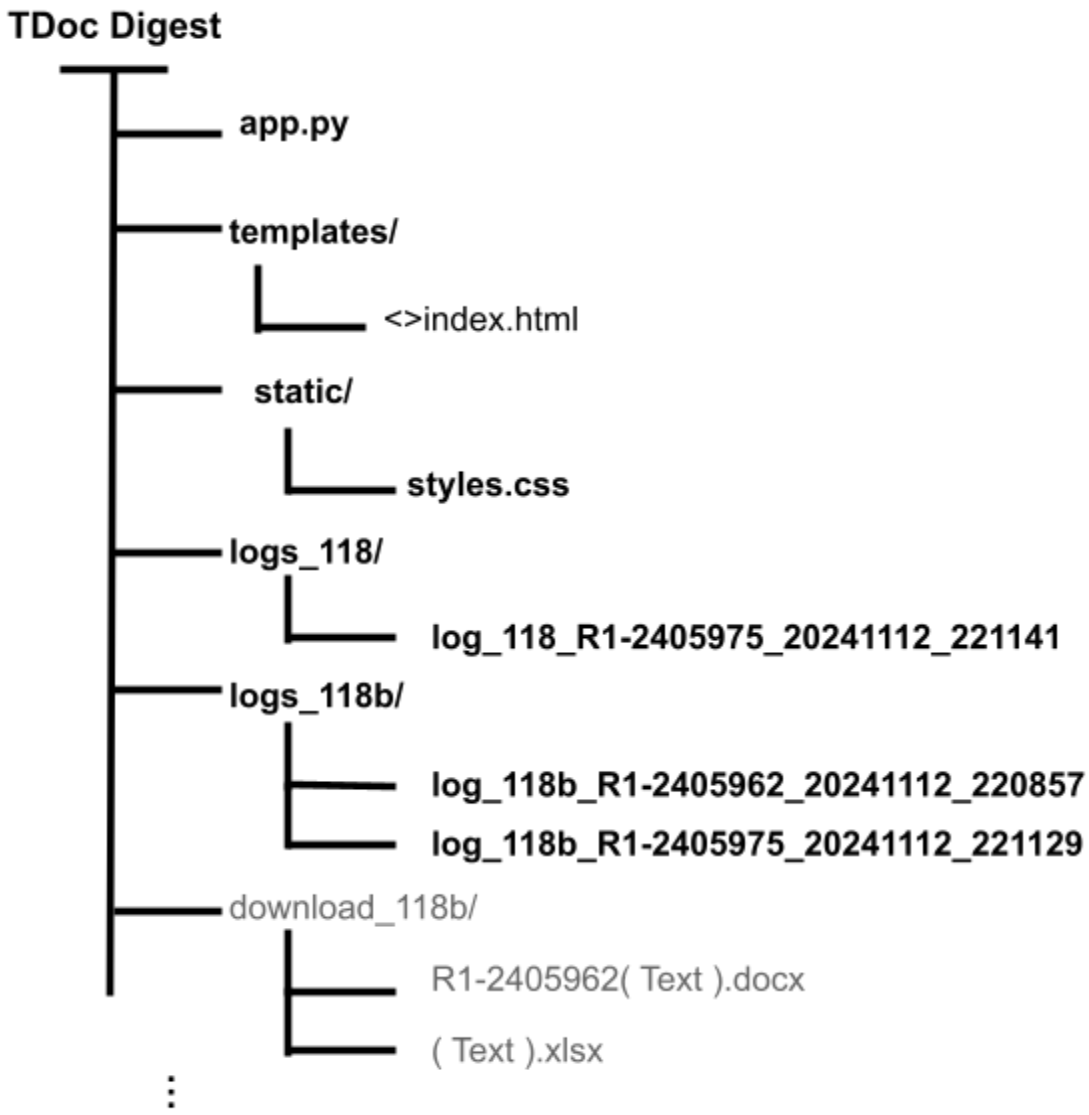


Fig. TDoc Digest Project Folder Structure

- **Main Application File (`app.py`):** This is the entry point of your Flask application. It contains the core configurations such as interpreting the meeting id/TDoc number, downloading the respective document from 3GPP FTP server, summary generation, log file handling, semantic score generation and web API (Flask).
- **Templates Folder (`templates/`):** This folder contains the index.html used by the web API.
- **Static Folder (`static/`):** The style css file used for web API.

- **log_<meeting id>:** This folder (eg. log_118b) contains the log file for each user request. The log file (eg. log_118b_R1-2405975_20241112_221129) are in the format of log_<meeting id>_<TDoc number>_<YearMonthDate>_<HourMinuteSeconds>.log. The file contains the debug information for the application developer.
- **download_<meeting id>:** This folder (eg. download_118b) contains the files downloaded from the 3GPP FTP server. After producing the output, these folders and the content are deleted in order to save the disk space.

How to use the TDoc Digest web application?

API Input: A user interacts with a web API as shown in Fig. 1. The meeting id and TDoc number are entered as inputs and click the “Generate Summary” button in order to generate an abstractive summary of the specified TDoc contribution.

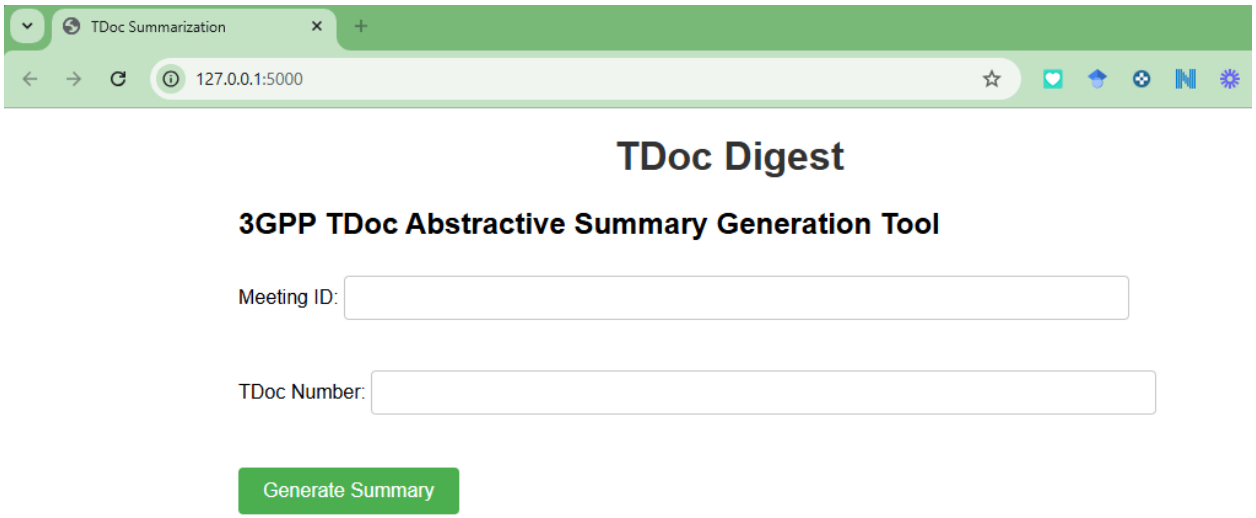


Fig. 1: TDoc Digest web API input

API Output (summary generation): The web API displays an abstractive summary of the TDoc contribution and the corresponding semantic score. An example output produced for TDoc number R1-2407616 in Meeting ID 118b is shown in Fig. 2 below.

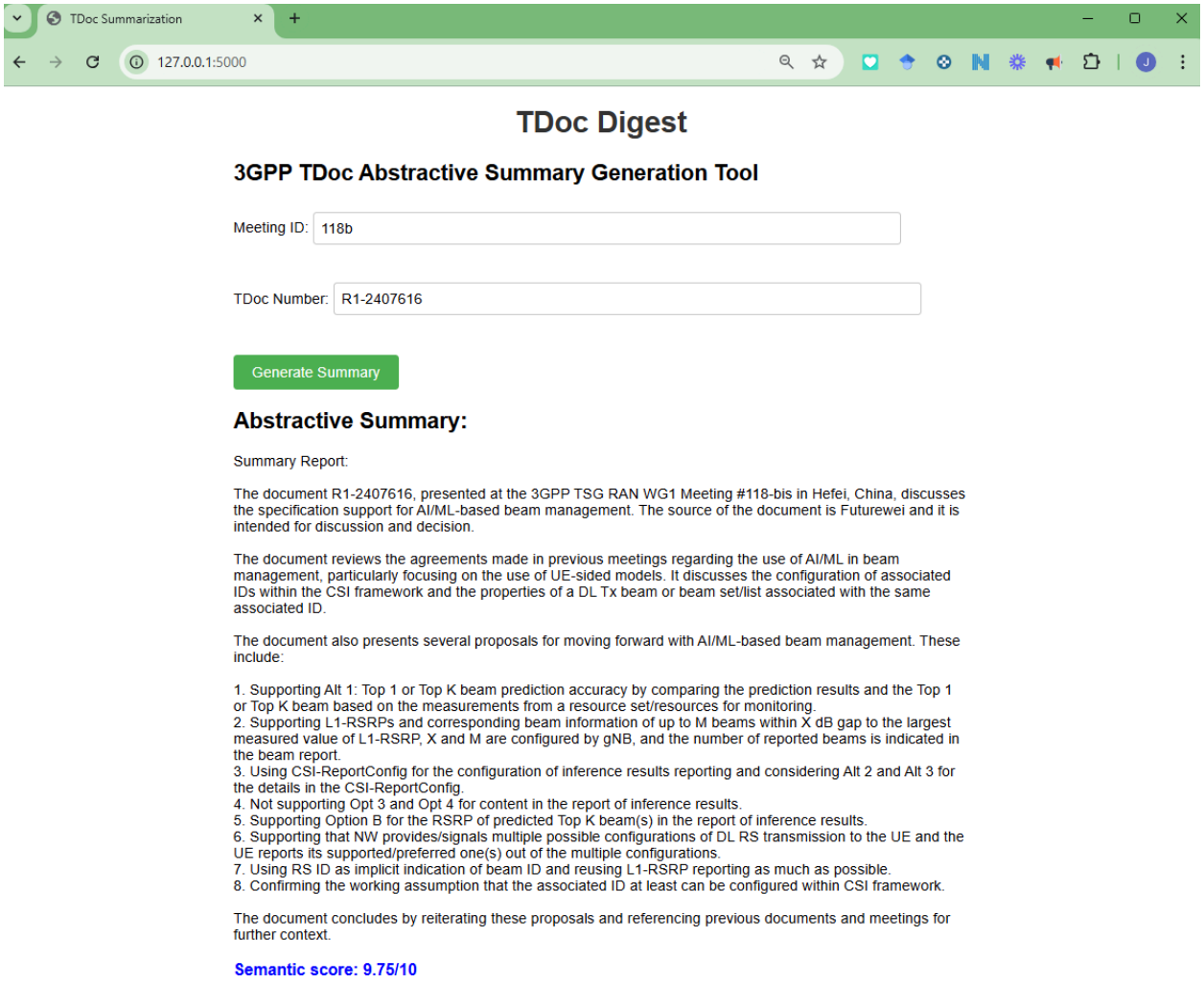


Fig. 2: TDoc Digest web API output (summary generation)

API Output (error generation): If the user inputs are erroneous, meaningful errors are produced by the application and displayed in the web API. In Fig. 3, an example is shown where the user enters the wrong TDoc number and the web API displays the error.

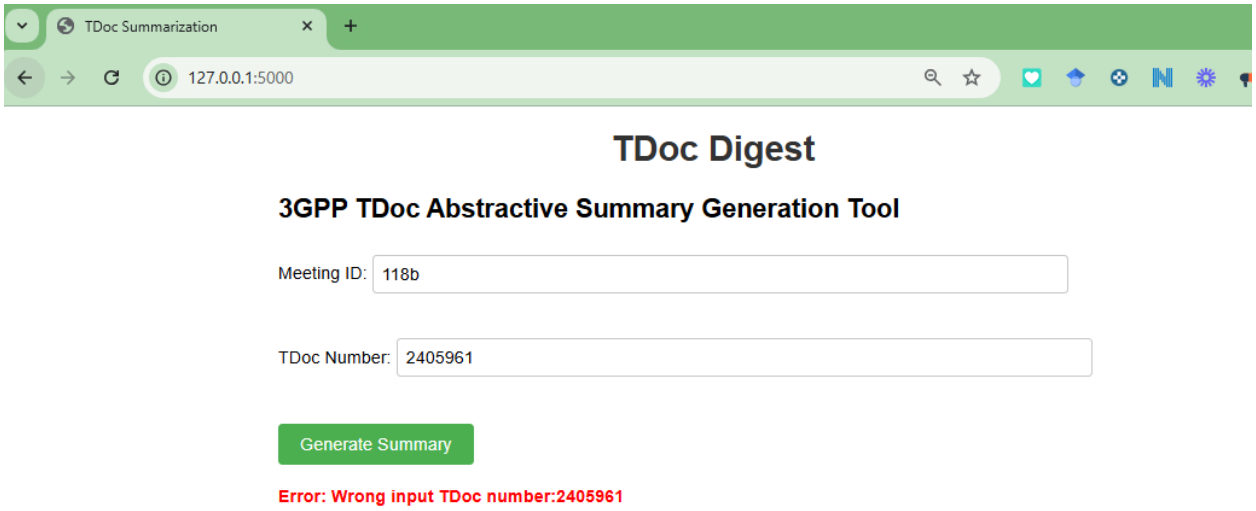
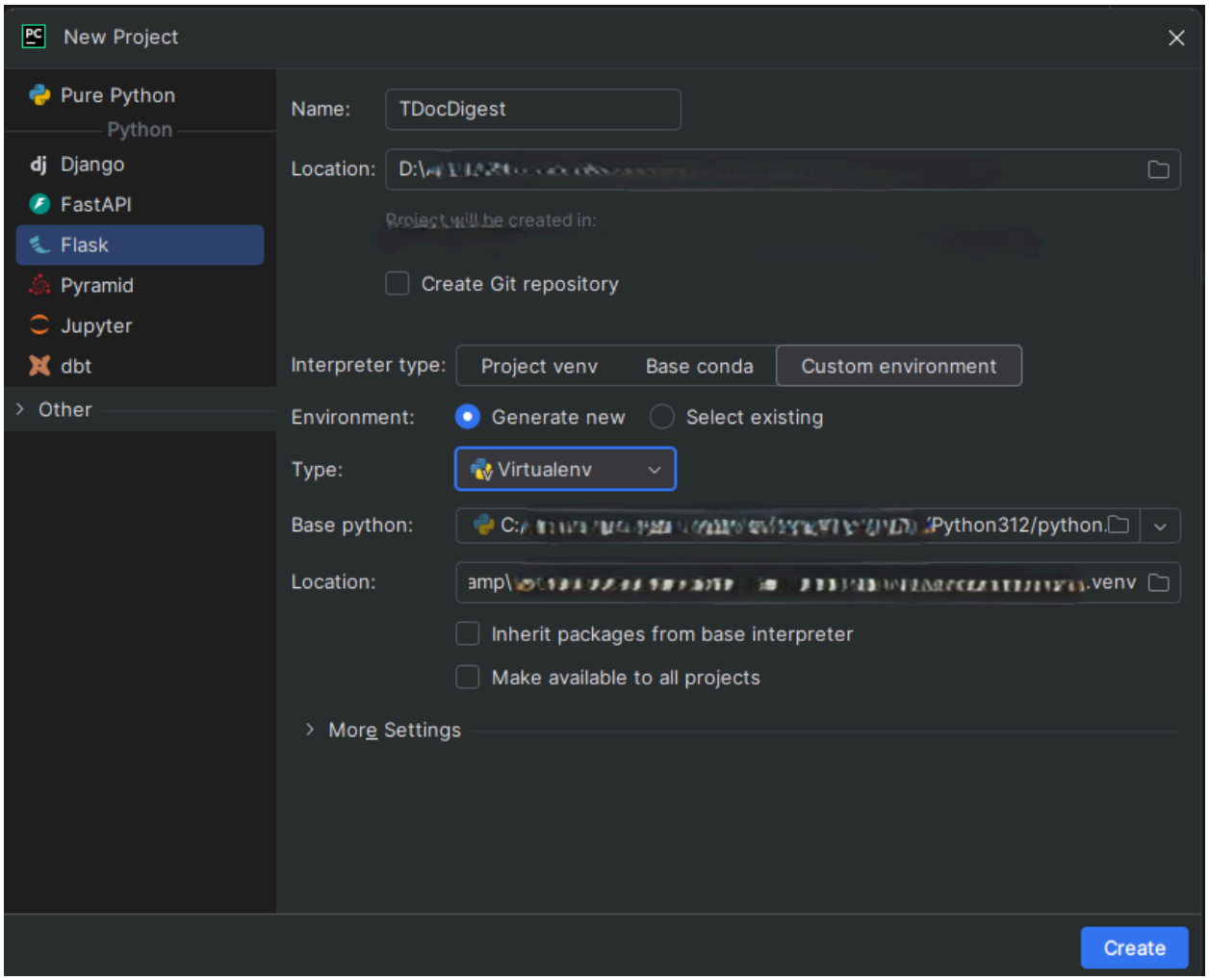


Fig. 3: TDoc Digest web API output (error generation)

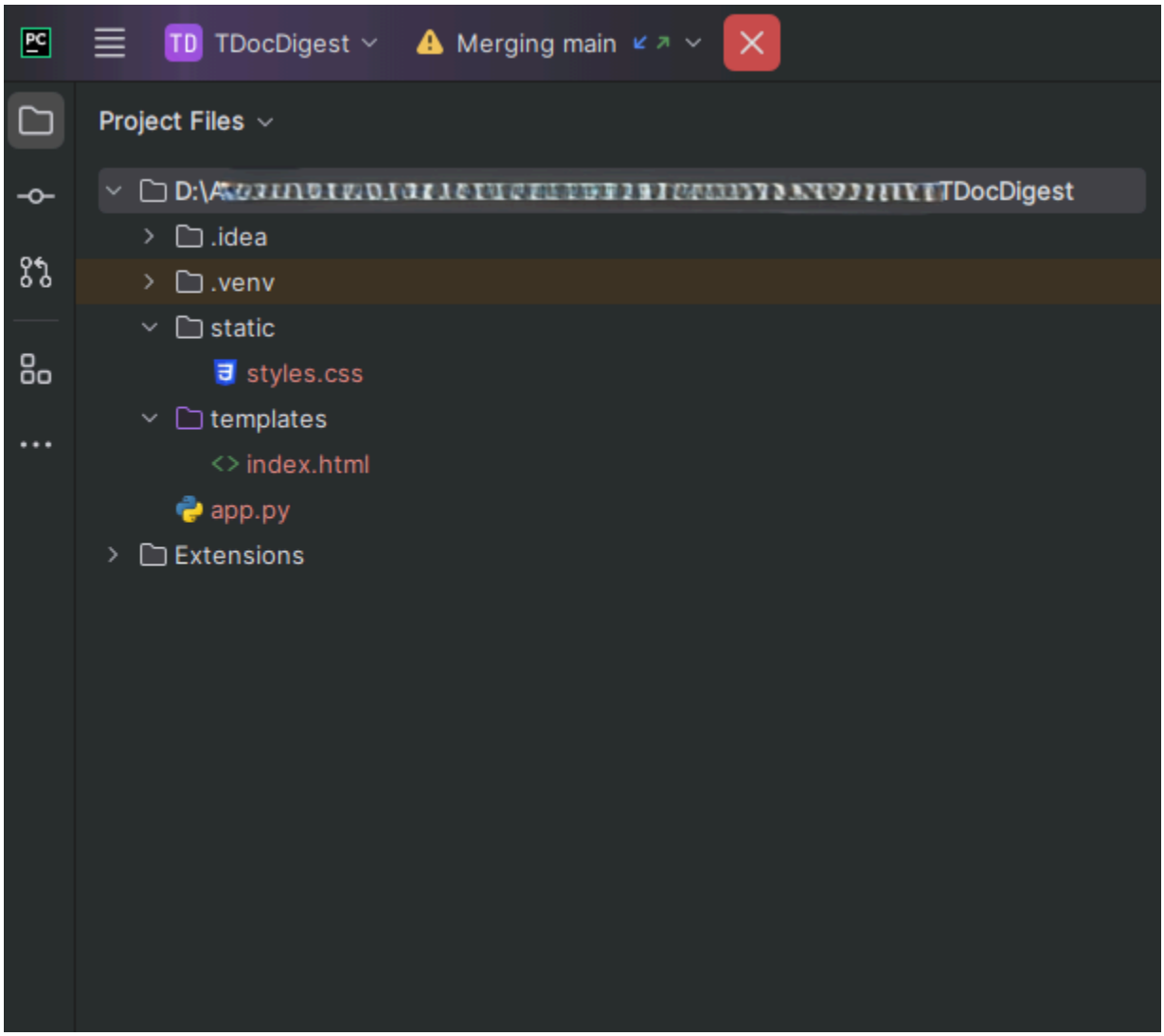
The web application produces various errors depending on the scenario.

Appendix A: JETBRAIN PyCharm project set up

Step 1: Create a Flask project in PyCharm

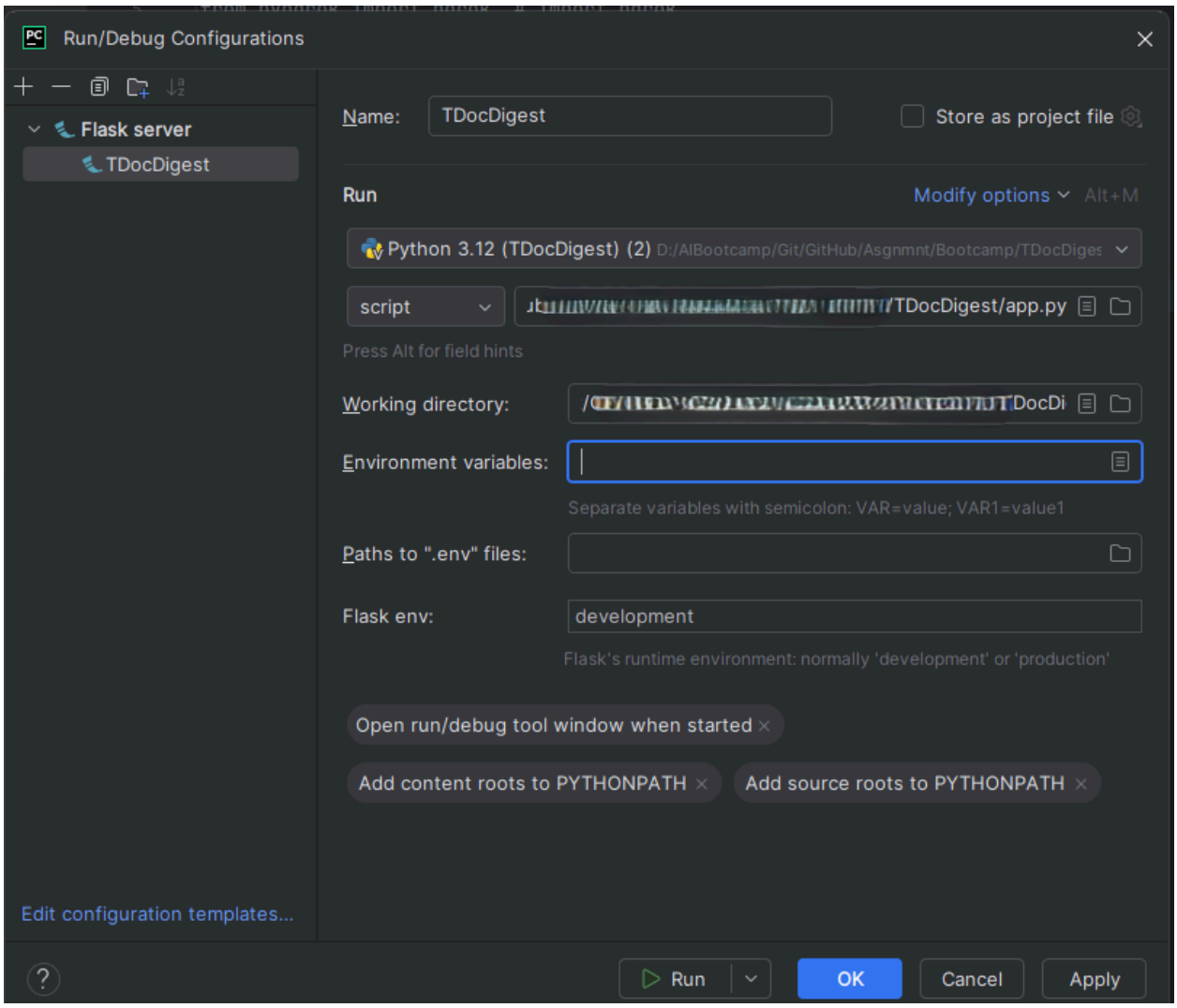


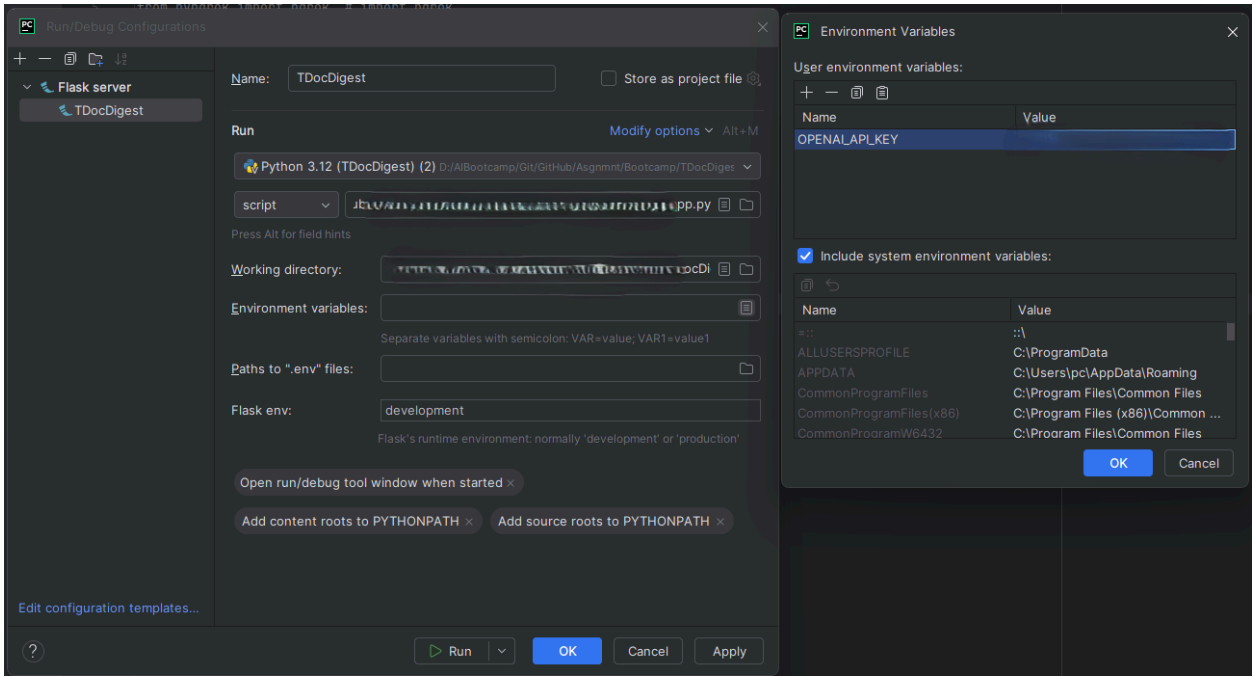
Step 2: Once the project is created, the folder structure (left top) is shown below.



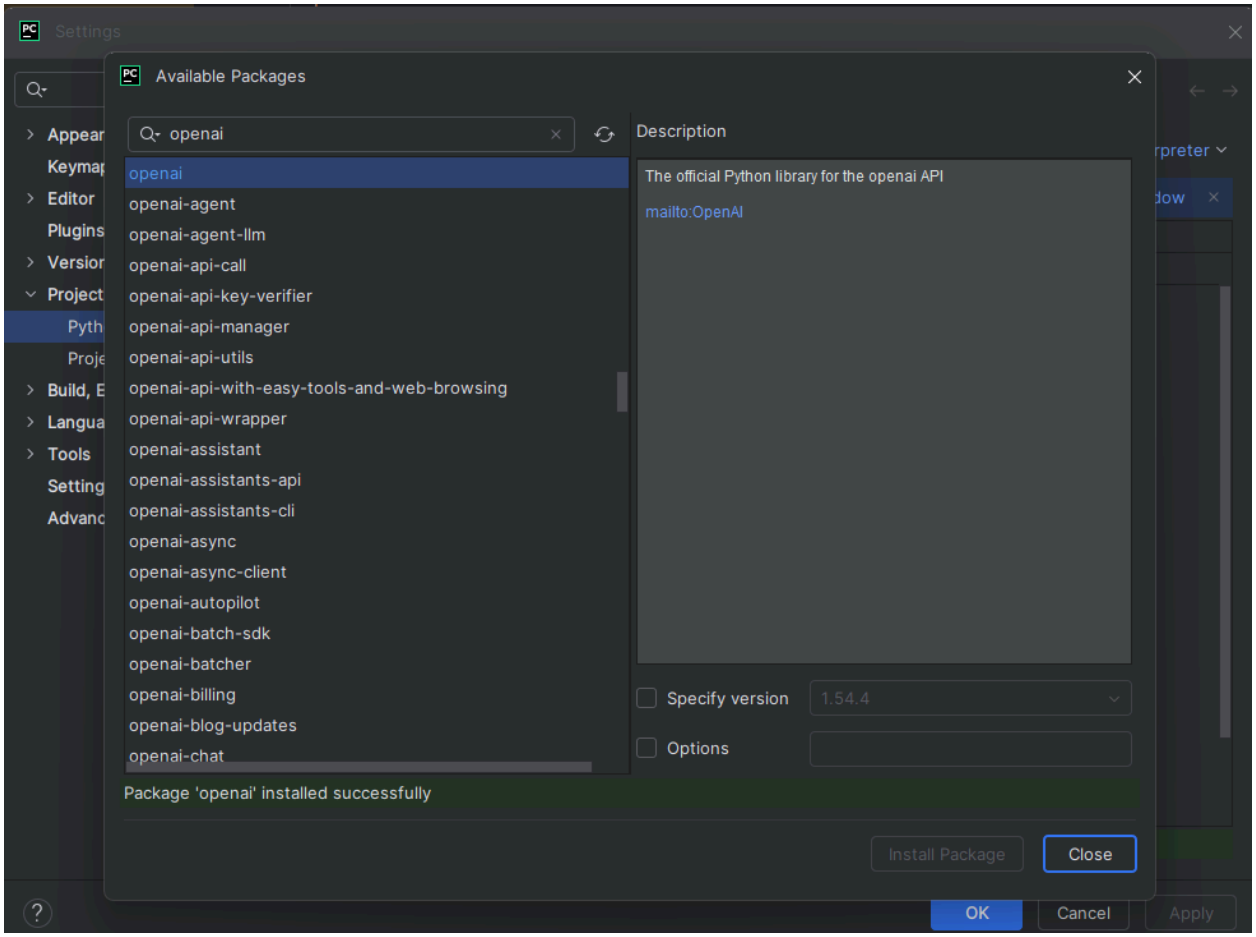
Step 3: Set environment variable OPENAI_API_KEY

Details on how to find your API Key is available in “Where do I find my OpenAI API Key?”




Run/Debug Configurations
Flask server
TDocDigest
Name: TDocDigest
Store as project file
Run
Modify options Alt+M
Python 3.12 (TDocDigest) (2)
script
Press Alt for field hints
Working directory:
Environment variables:
Separate variables with semicolon: VAR=value; VAR1=value1
Paths to ".env" files:
Flask env: development
Flask's runtime environment: normally 'development' or 'production'
Open run/debug tool window when started
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH
Edit configuration templates...
Run
OK
Cancel
Apply
Environment Variables
User environment variables:
Name
Value
OPENAI_API_KEY
Include system environment variables:
Name
Value
ALLUSERSPROFILE
C:\ProgramData
APPDATA
C:\Users\pc\AppData\Roaming
CommonProgramFiles
C:\Program Files\Common Files
CommonProgramFiles(x86)
C:\Program Files (x86)\Common ...
CommonProgramW6432
C:\Program Files\Common Files
OK
Cancel

Step 4: Install packages



Similarly, as necessary, packages such as openai, flask, docx2txt, requests, os, shutil, logging, pyngrok, requests, zipfile, io, threading, datetime can be installed