**PROGRAMMING FOR AI**

**PROJECT REPORT**

**BS in Artificial Intelligence**



Department of Software Engineering

Faculty of Computer Science & Information Technology

The Superior University, Lahore

|  |  |  |
| --- | --- | --- |
| **Sr.#** | **Reg.#** | **Student Name** |
|  | SU92-BSAIM-F23-073 | Saad Waleed |

**Table of Contents**

[**1** **INTRODUCTION:** 5](#_Toc197283998)

[**2** **Supported Document Types:** 5](#_Toc197283999)

[**3** **Core Functionalities:** 5](#_Toc197284000)

[3.1 Optical Character Recognition (OCR): 5](#_Toc197284001)

[3.2 Text Extraction: 5](#_Toc197284002)

[3.3 Image Enhancement: 5](#_Toc197284003)

[3.4 Text Summarization: 5](#_Toc197284004)

[4 Technology Stack: 6](#_Toc197284005)

[4.1 Backend: 6](#_Toc197284006)

[4.2 OCR: 6](#_Toc197284007)

[4.3 PDF Processing: 6](#_Toc197284008)

[4.4 Document Processing: 6](#_Toc197284009)

[4.5 Summarization: 6](#_Toc197284010)

[4.6 Frontend: 6](#_Toc197284011)

[**5** **User Interface:** 6](#_Toc197284012)

[**6** **CODE IMPLEMENTATION of APP.py:** 6](#_Toc197284013)

[**6.1** **Setup** 6](#_Toc197284014)

[**6.2** **File Upload & Text Extraction** 7](#_Toc197284015)

[**6.3** **Summarization** 8](#_Toc197284016)

[**6.4** **Web Interface** 8](#_Toc197284017)

[6.5 Error Handling 9](#_Toc197284018)

[**7** **Index.Html:** 10](#_Toc197284019)

[**7.1** **Header:** 10](#_Toc197284020)

[**7.2** **Upload Form:** 10](#_Toc197284021)

[7.3 **Flash Messages:** 11](#_Toc197284022)

[**7.4** **Output Sections:** 11](#_Toc197284023)

[**7.5** **Additional Elements:** 12](#_Toc197284024)

[**7.6** **Scripts:** 12](#_Toc197284025)

[**8** **Mini.css:** 13](#_Toc197284026)

[**8.1** **Font Imports:** 13](#_Toc197284027)

[**8.2** **Header Styling (.fullbg):** 13](#_Toc197284028)

[**8.3** **Card Styling (.textcard, .seccard, .outputcard):** 14](#_Toc197284029)

[**8.4** **Buttons (.btn-gradient, .btn-pink-gradient):** 15](#_Toc197284030)

[**8.5** **Form Elements:** 16](#_Toc197284031)

[**8.6** **Other Elements:** 16](#_Toc197284032)

[**8.6.1** **.purplebg:** 16](#_Toc197284033)

[**8.6.2** **.thumbs-slider:** 16](#_Toc197284034)

[**8.6.3** **.logo, .qote, .fonttext, .handimage, .carcard:** 17](#_Toc197284035)

[**8.7** **Animations:** 17](#_Toc197284036)

[**9** **OUTPUT:** 18](#_Toc197284037)

Table of Figures

[Figure a app.py-1 7](#_Toc197284038)

[Figure b app.py-2 8](#_Toc197284039)

[Figure c app.py-3 9](#_Toc197284040)

[Figure d app.py-4 10](#_Toc197284041)

[Figure e Header output 11](#_Toc197284042)

[Figure f Flash messages – Output 12](#_Toc197284043)

[Figure g Scripts 13](#_Toc197284044)

[Figure h header-styling 14](#_Toc197284045)

[Figure i Card-styling 15](#_Toc197284046)

[Figure j Button-styling 16](#_Toc197284047)

[Figure k Thumb Slider 17](#_Toc197284048)

[Figure l output- text extraction 18](#_Toc197284049)

[Figure m output-summarization 18](#_Toc197284050)

Document Scanner and Summarizer

# **INTRODUCTION:**

* A web-based application that extracts text from various document formats and generates concise summaries automatically.

# **Supported Document Types:**

* Images: PNG, JPG, JPEG
* PDFs (both scanned images and embedded text)
* Word documents: DOCX and DOC

# **Core Functionalities:**

## Optical Character Recognition (OCR):

* Uses PaddleOCR to extract text from images and scanned PDFs.

## Text Extraction:

* Extracts embedded text from PDFs and DOCX files using PyMuPDF and python-docx libraries.

## Image Enhancement:

* Applies optional super-resolution (EDSR model) to improve OCR accuracy on low-quality images.

## Text Summarization:

* Utilizes a large language model (LLM) via LangChain and Groq API to generate clear, concise summaries of extracted text.

# Technology Stack:

## Backend:

* Python with Flask web framework

## OCR:

* PaddleOCR with OpenCV preprocessing

## PDF Processing:

* PyMuPDF (fitz) and PyPDF2

## Document Processing:

* python-docx

## Summarization:

* LangChain integration with Groq's Llama3-8b-8192 model

## Frontend:

* HTML, Bootstrap 4, custom CSS for UI/UX enhancements

# **User Interface:**

* Simple web interface allowing users to upload files, extract text, and view summaries.
* Responsive design with modern styling, animations, and user feedback via flash messages.

# **CODE IMPLEMENTATION of APP.py:**

## **Setup**

* Loads environment variables for API keys and secret keys.
* Initializes Flask app and configures an upload folder.
* Sets up PaddleOCR for text recognition and optionally loads a super-resolution model to enhance images.
* Configures a LangChain + Groq language model for text summarization.

## **File Upload & Text Extraction**

* User uploads an image, PDF, or Word document.
* The file is saved on the server.
* Depending on the file type:
* Images: Preprocessed (grayscale, blur, threshold), optionally enhanced, then OCR is performed.
* PDFs: Attempts to render pages as images and OCR them; if that fails, extracts embedded text.
* DOCX: Extracts text directly from document paragraphs.
* Extracted text is stored in the user session.

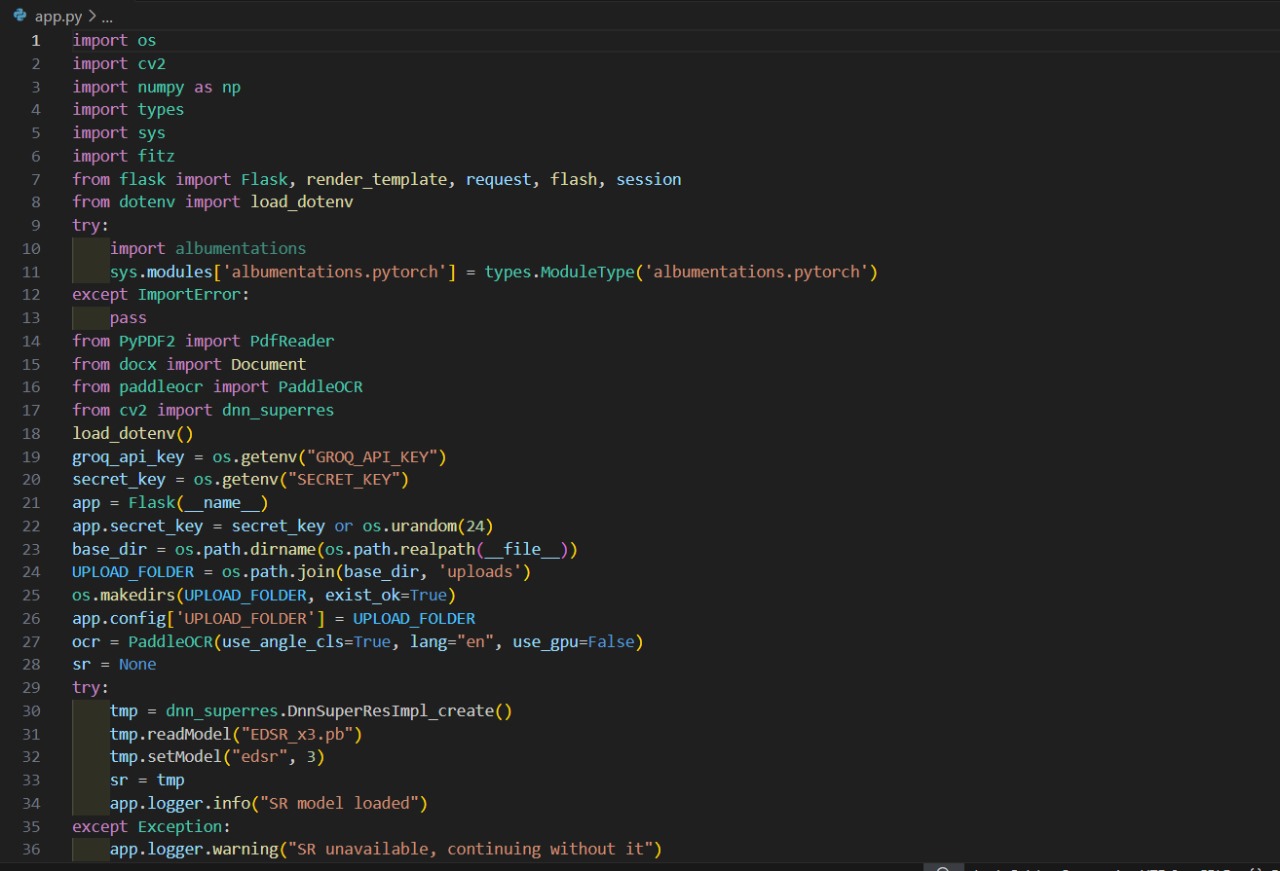


Figure a app.py-1

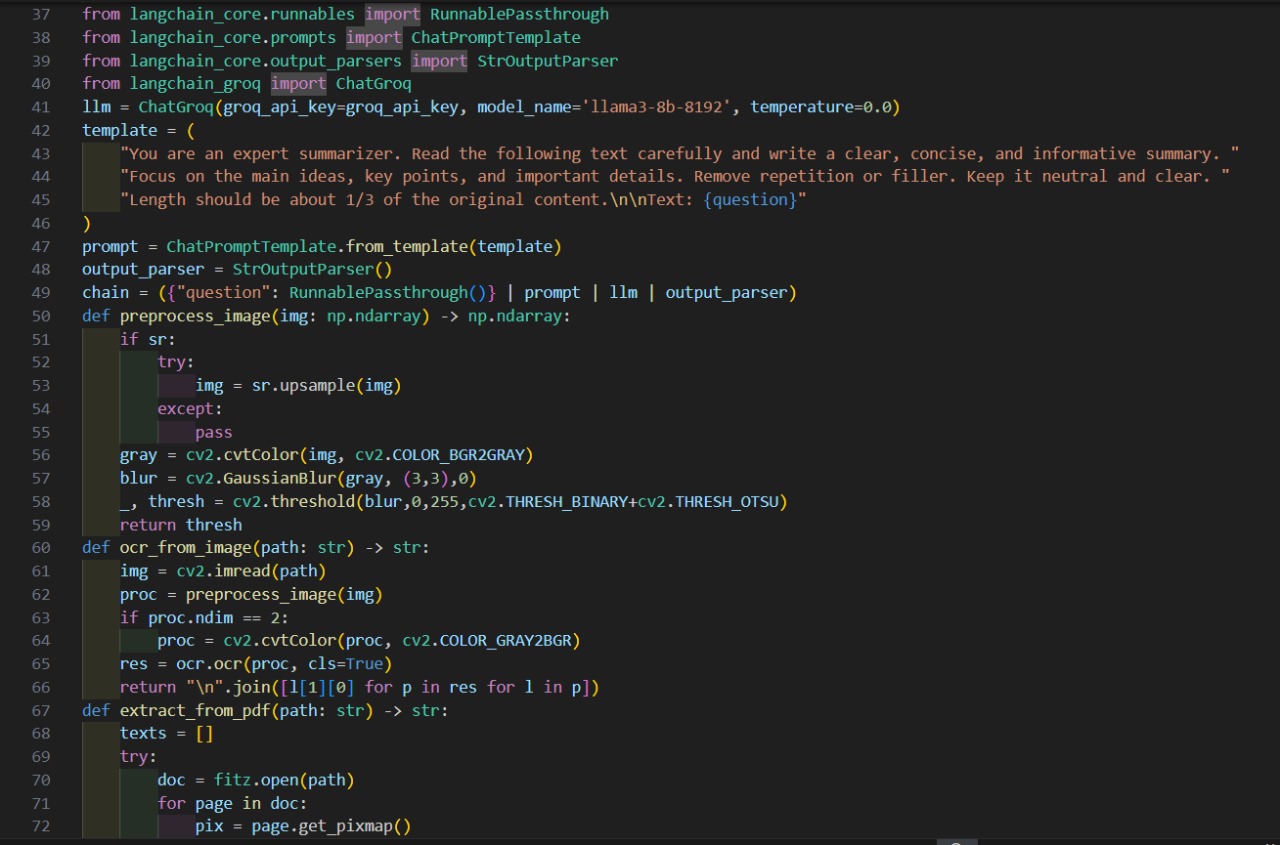


Figure b app.py-2

## **Summarization**

* When requested, the extracted text is sent to the Groq-powered LLM with a prompt to generate a clear and concise summary.
* The summary is saved in the session and displayed.

## **Web Interface**

* Single route (/) handles file uploads, extraction, and summarization.
* Uses Flask sessions to keep extracted text and summaries between requests.
* Provides user feedback via flash messages.

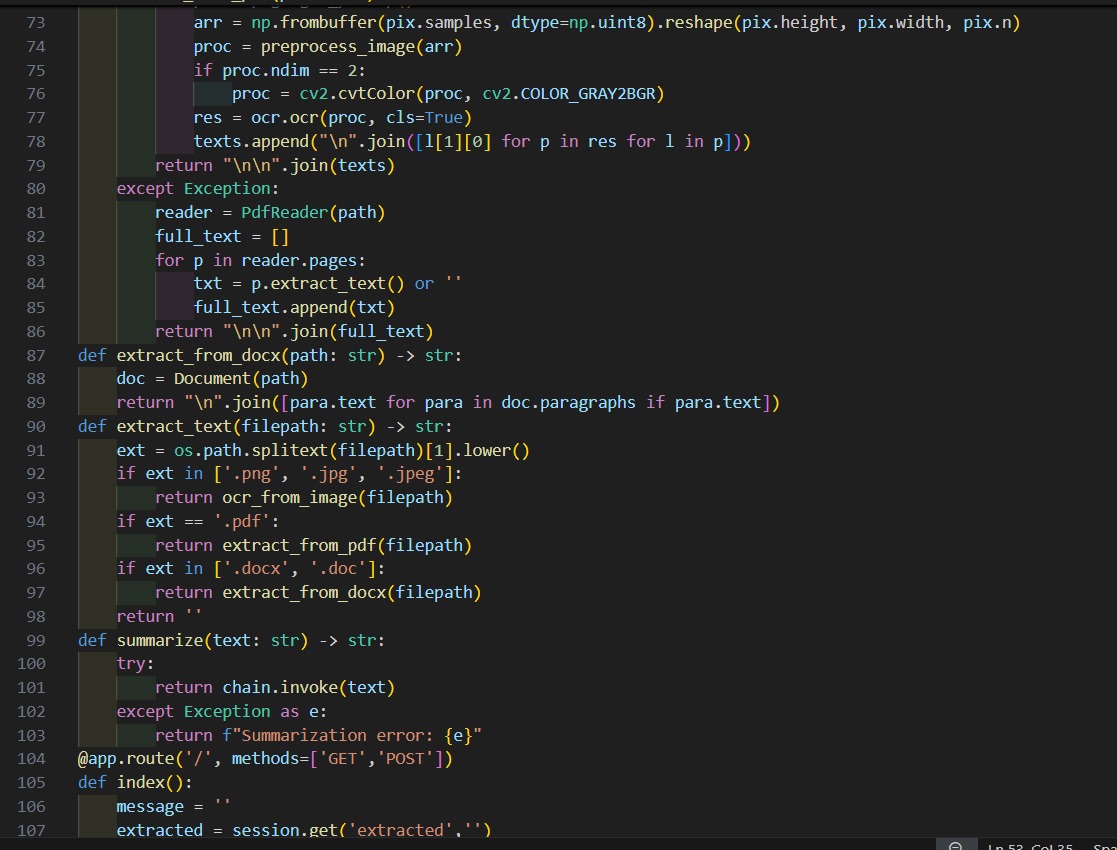


Figure c app.py-3

## Error Handling

* Gracefully handles missing super-resolution model or OCR failures.
* Provides fallback for PDF text extraction.
* Catches summarization errors and informs the user.

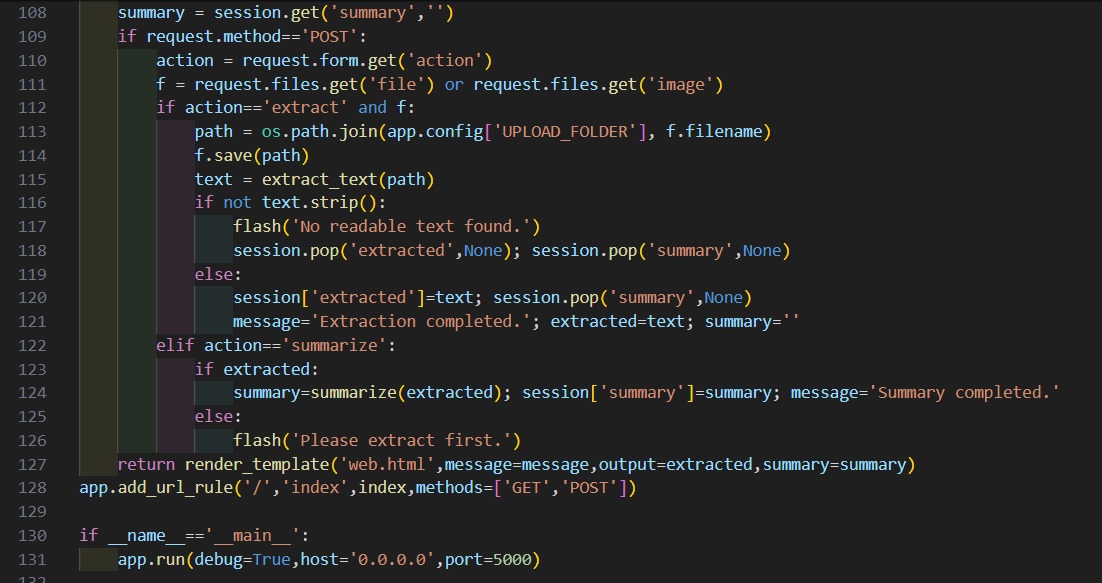


Figure d app.py-4

# **Index.Html:**

This is a Bootstrap-based web page for your Flask app that enables users to upload images for text extraction and summarization.

## **Header:**

Displays the app title centered with elegant typography.

## **Upload Form:**

Lets users select image files (.png, .jpg, .jpeg).

* Two buttons trigger either text extraction or summarization actions on the server.
* Uses multipart encoding for file upload.

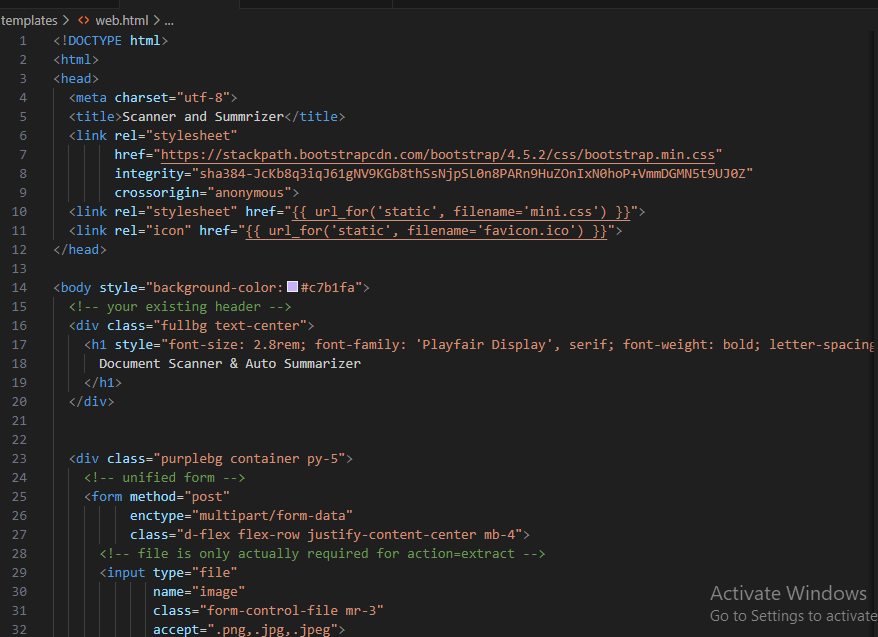


Figure e Header output

## **Flash Messages:**

* Shows server feedback (errors or status) in a styled alert.

## **Output Sections:**

* Displays extracted text and its summary in separate, formatted blocks.
* Uses <pre> tags to preserve text formatting.

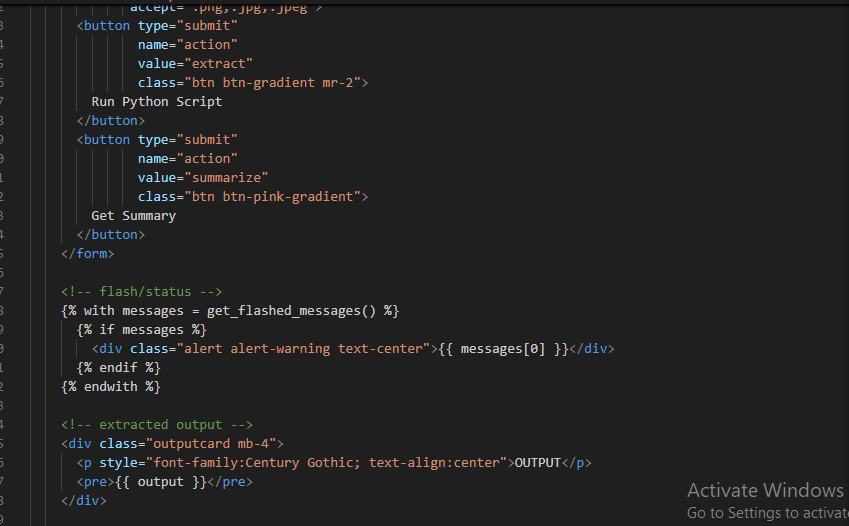


Figure f Flash messages – Output

## **Additional Elements:**

* Placeholder for thumbnails and a marquee footer with author credit.

## **Scripts:**

* Includes jQuery, Popper.js, and Bootstrap JS for responsive UI and interactivity.

The template integrates seamlessly with Flask by rendering dynamic content (output, summary, and flash messages) passed from the backend.

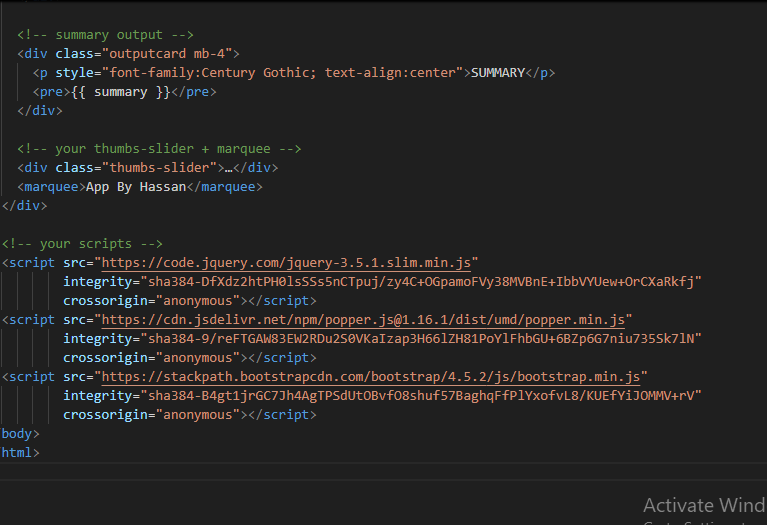


Figure g Scripts

# **Mini.css:**

## **Font Imports:**

Loads several Google Fonts (Bree Serif, Roboto, Lobster, Playfair Display, etc.) for stylish typography throughout the app.

## **Header Styling (.fullbg):**

Applies a purple-teal gradient, centers content with flexbox, adds a drop shadow, and animates the header to float in from above.

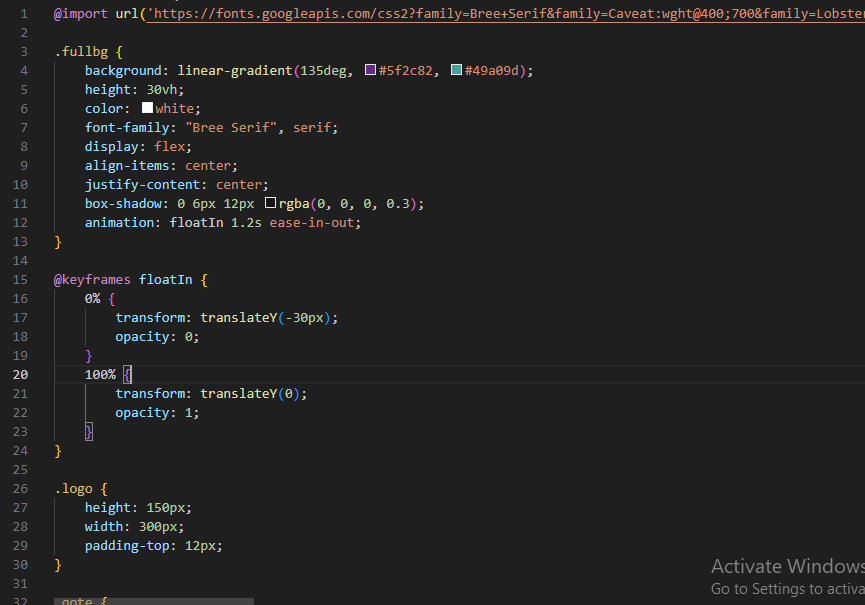


Figure h header-styling

## **Card Styling (.textcard, .seccard, .outputcard):**

Gives cards a semi-transparent, blurred background with rounded corners and drop shadows. Cards slightly lift and deepen their shadow on hover for a modern, interactive effect.

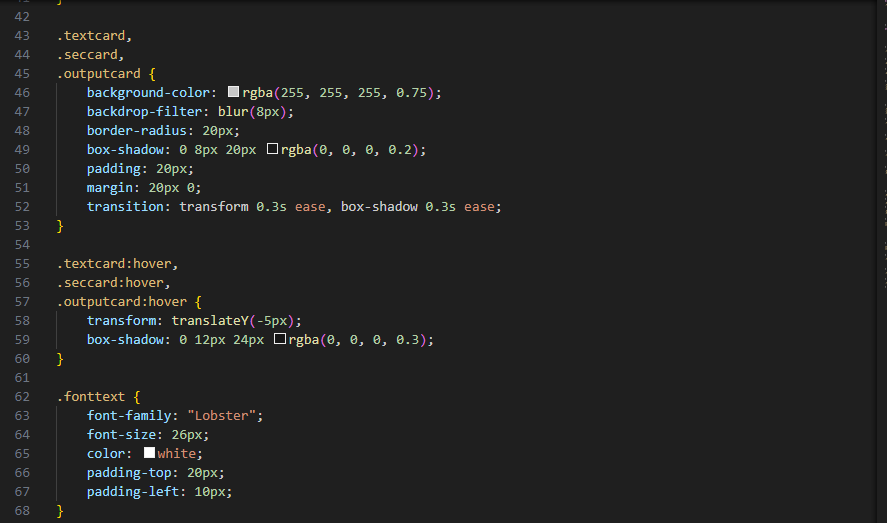


Figure i Card-styling

## **Buttons (.btn-gradient, .btn-pink-gradient):**

Stylish gradient backgrounds, white text, rounded corners, and a shadow. Buttons grow slightly and the shadow intensifies on hover for tactile feedback.

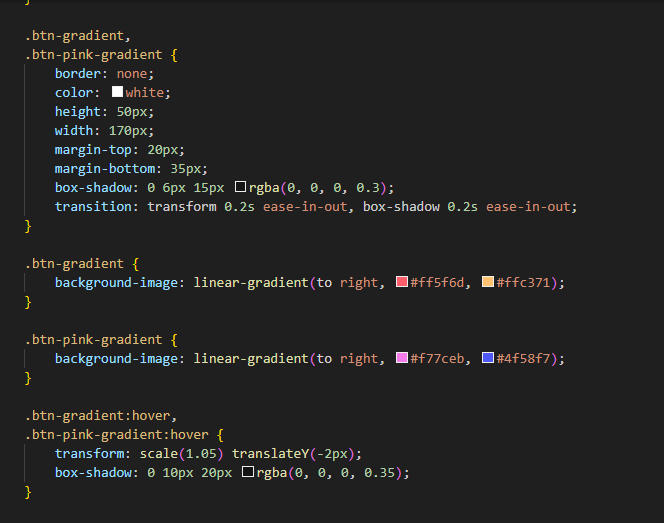


Figure j Button-styling

## **Form Elements:**

File inputs and submit buttons are styled with color, padding, rounded corners, and a subtle shadow, enhancing their appearance and usability.

## **Other Elements:**

### **.purplebg:**

Sets a soft purple background for the main content area.

### **.thumbs-slider:**

Arranges icons in a row for any slider/thumbs feature.

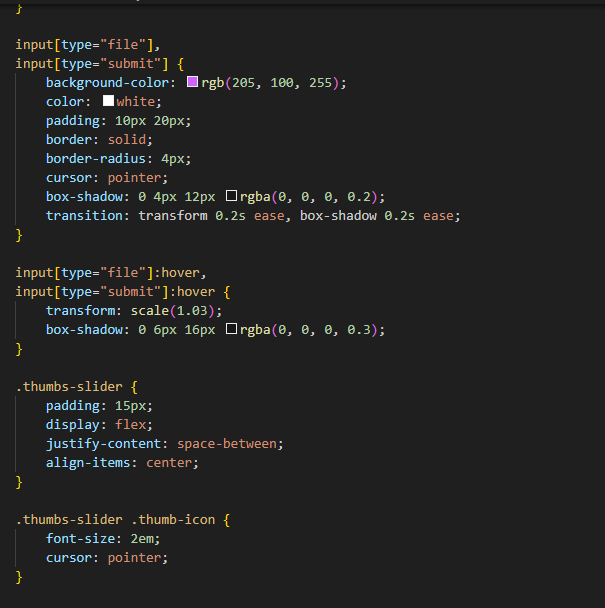


Figure k Thumb Slider

### **.logo, .qote, .fonttext, .handimage, .carcard:**

Custom styles for images, quotes, and decorative text/images.

## **Animations:**

The floatIn keyframes animate the header smoothly into view.

# **OUTPUT:**

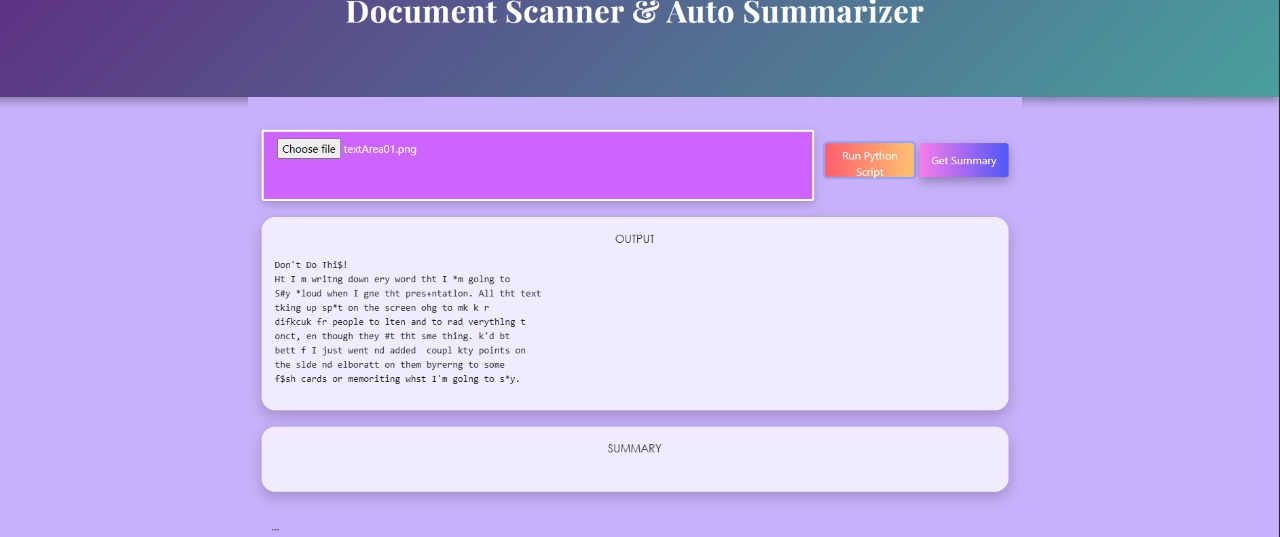


Figure l output- text extraction

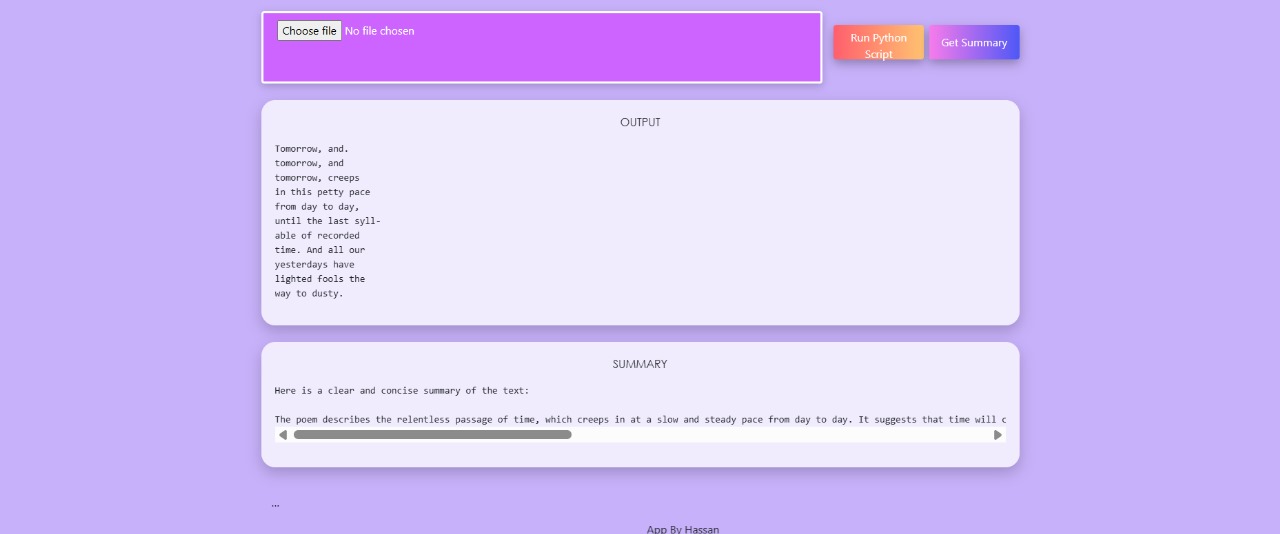


Figure m output-summarization

------------------------------------------------------------END----------------------------------------------------------------