AlphaX Entertainments

Release 1.1.1

Qadeer Ahmad

Sep 20, 2024

TABLE OF CONTENTS:

Welcome to the official AlphaX Entertainments documentation. This guide provides detailed instructions, overviews, and references for using and understanding the various components of our project.

CHAPTER

ONE

CONTENTS

1.1 AlphaX-Entertainments

1.1.1 app module

app.**load_css**(*file_name*)

Load a CSS file and inject it into the Streamlit app.

Parameters
file_name (*str*) – The path to the CSS file to load.

Return type
None

app.**main**()

Main function of the streamlit app. This function renders the main interface of the app.

The interface has a sidebar with two options: "Home" and "Try Now AlphaX". If the user selects "Home", a form is displayed to enter credentials. If the user selects "Try Now AlphaX", the app displays a subheader with the username and a menu with four options: Text, Image, Audio and Video. Depending on the user's choice, a different handler function is called to process the user's input.

Parameters
None

Returns
None

1.1.2 components package

Submodules (components)

components.display_components module

components.display_components.**display_home_content**() → None

Display the home content of the web app, including the title, description, and options for dialogues, images, audios, and videos.

The content is displayed in a grid layout with 4 columns. Each column contains a card with a title, description, and an image.

The user can interact with the cards by hovering over them, which will display a tooltip with the description.

This function is called when the user navigates to the home page of the web app.

components.display_components.**display_html_content**(*tag: str = 'footer'*) → None

Displays HTML content in the app based on the given tag. Possible values for the tag parameter are:

- 'credentials': Displays a message asking the user to enter their OpenAI API key.
- 'footer': Displays social media links at the bottom of the app.
- 'alert': Displays a warning message asking the user to enter their credentials.
- 'welcome': Displays a welcome message with the username.

Parameters
tag (*str*) – The tag to display the HTML content for. Defaults to 'footer'.

components.display_components.**display_recognition_result**(*data: Dict[str, str | List[str]]*) → None

Display the recognition result including title, poster, and details.

Parameters
data (*Dict[str, Union[str, List[str]]]*) – A dictionary containing movie details like title, genre, cast, etc.

components.display_components.**display_recommendation_result**(*data: List[Dict[str, str]]*) → None

Display a list of recommended movies with their titles and posters.

Parameters
data (*List[Dict[str, str]]*) – A list of dictionaries containing movie recommendations with title and IMDb URL.

components.input_components module

components.input_components.**get_audio_prompt**() → Tuple[str | None, str]

Get audio input from the user either by recording or uploading.

Returns
The recorded or uploaded audio and the input type.

Return type
Tuple[Union[str, None], str]

components.input_components.**get_choice**() → str

Display options for the user to choose from and return the selected option.

Returns
The selected option.

Return type
str

components.input_components.**get_credentials**() → bool

Get API credentials from the user and save them to a .env file.

Returns
True if credentials were successfully saved, False otherwise.

Return type
bool

components.input_components.**get_image_prompt**() → Tuple[str | None, str]

Get image URL or uploaded image from the user.

Returns
The image URL or uploaded image file and the input type.

Return type
Tuple[Union[str, None], str]

components.input_components.**get_page_choice**() → str

Renders a sidebar with options: "Home", "Credentials", and "Try AlphaX". Returns the selected option as a string.

components.input_components.**get_text_prompt**() → str

Get text input from the user.

Returns
The entered text.

Return type
str

components.input_components.**get_video_prompt**() → bytes | None

Get a video file uploaded by the user.

Returns
The uploaded video file or None if no file was uploaded.

Return type
Optional[bytes]

Module contents (components)

1.1.3 helpers package

Submodules (helpers)

helpers.save_credentials module

helpers.save_credentials.**save_credentials**(*api_key: str*) → bool

Save API credentials to a file.

This function writes the provided API key to a *.env* file, setting the key for OpenAI API access.

Parameters
api_key (*str*) – The API key/token to save.

Returns
True if the credentials were saved successfully, False otherwise.

Return type
bool

Example

```
>>> save_credentials("your_api_key_here")
True
```

helpers.utils module

helpers.utils.**download_image**(*url*)

Download an image from a URL and save it to the server.

This function downloads an image from the specified URL and saves it to a predefined folder. If the download fails, a default image is used.

Parameters
url (*str*) – The URL of the image to download.

Returns
The path to the saved image, or the path to the default image if an error occurred.

Return type
str

Example

```
>>> download_image("https://example.com/image.png")
'data/images/downloaded.png'
```

helpers.utils.**ensure_directories**() → None

Ensure that the directories for saving images and videos exist.

This function makes sure that the directories specified in the constants IMAGE_SAVE_FOLDER and VIDEO_SAVE_FOLDER exist. If they do not exist, they are created.

Parameters
None

Returns

- *None*
- *Example* – >>> ensure_directories() # Directories are created if they don't exist

helpers.utils.**get_image_url**(*imdb_url*)

Retrieve the image URL from an IMDb page.

This function extracts the URL of the image from the IMDb page specified by the provided URL. If the extraction fails, a default image path is returned.

Parameters
imdb_url (*str*) – The URL of the IMDb page to retrieve the image from.

Returns
The URL of the image, or the path to the default image if an error occurred.

Return type
str

Example

```
>>> get_image_url("https://www.imdb.com/title/tt1234567/")
'https://example.com/image.png'
```

helpers.utils.**read_image**(*image_path*)

Read an image file and encode it in base64.

This function reads an image file from the specified path and encodes it as a base64 string.

Parameters
image_path (*str*) – The path to the image file.

Returns
The base64 encoded image data.

Return type
str

Raises

- **FileNotFoundError** – If the image file does not exist.
- **Exception** – For other errors encountered while reading the file.

Example

```
>>> read_image("data/images/sample.png")
'iVBORw0KGgoAAAANSUhEUgAAAAUA...'
```

helpers.utils.**save_image**(*uploaded_image*)

Save an uploaded image to the server.

This function saves the provided image file to a predefined folder.

Parameters
uploaded_image (*UploadedFile*) – The image file to be saved.

Returns
The path to the saved image.

Return type
str

Example

```
>>> save_image(uploaded_image)
'data/images/uploaded_image.png'
```

helpers.utils.**save_video**(*uploaded_file*)

Save an uploaded video to the server.

This function saves the provided video file to a predefined folder.

Parameters
uploaded_file (*UploadedFile*) – The video file to be saved.

Returns
The path to the saved video, or None if an error occurred.

Return type
str

Example

```
>>> save_video(uploaded_file)
'data/video/uploaded_video.mp4'
```

helpers.utils.**validate_audio_file**(*uploaded_file*) → bool

Validate the size of an uploaded audio file.

This function checks if the size of the uploaded audio file is within the allowed limit (4 MB).

Parameters
uploaded_file (*st.uploaded_file_manager.UploadedFile*) – The uploaded audio file.

Returns
True if the file size is less than or equal to 4 MB, False otherwise.

Return type
bool

Example

```
>>> validate_audio_file(uploaded_file)
True
```

Module contents (helpers)

1.1.4 prompts package

Submodules (prompts)

prompts.system_prompts module

Module contents (prompts)

1.1.5 src package

Submodules (src)

src.api_handlers module

class src.api_handlers.**LLMHandler**(*model: str*, *temperature: float = 0.7*, *max_tokens: int = 2000*, *max_retries: int = 3*, *timeout: int = 60*)

Bases: object

get_llm_instance() → ChatOpenAI | None

Create an instance of the Langchain OpenAI chat model with the provided parameters, or return None if an error occurs.

Parameters
None

Returns
The created LLM instance, or None if an error occurs.

Return type
Optional[ChatOpenAI]

handle_openai_error(*error: Exception*)

Handle errors from OpenAI.

This method handles errors that occur when invoking the LLM instance. It displays an error message to the user based on the type of error that occurred.

Parameters
error (*Exception*) – The error that occurred.

Raises
ValueError – If the LLM instance is not created.

handle_response(*messages: List[Tuple[str, str]]*) → str

Handle the response from the LLM instance, including any errors that may occur.

Parameters
messages (*List[Tuple[str, str]]*) – The messages to send to the LLM instance.

Returns
The response from the LLM instance, or an error message if an error occurs.

Return type
str

Raises

- **ValueError** – If the LLM instance is not created.
- **Exception** – If any other error occurs.

class src.api_handlers.**RecognitionHandler**(*model: str = 'gpt-4o'*, ***kwargs*)

Bases: *LLMHandler*

get_dialogue_response(*dialogue: str*) → str

Get the response from the LLM given a dialogue.

Parameters
dialogue (*str*) – The dialogue to get the response for.

Returns
The response from the LLM.

Return type
str

get_image_response(*image_data: str*) → str

class src.api_handlers.**RecommendationHandler**(*model: str = 'gpt-4o'*, ***kwargs*)

Bases: *LLMHandler*

get_recommendation_response(*input_data: str*) → dict

Get a recommendation response for a given input.

Parameters
input_data (*str*) – The input data to generate recommendations for.

Returns
The recommendation response in JSON format.

Return type
dict

Raises
Exception – If there is an error parsing the recommendations response.

src.api_handlers.**get_recognition_response**(*input_data: str*, *input_type: str = 'dialogue'*) → str | None

Get a recognition response for a given input.

Parameters

- **input_data** (*str*) – The input data to generate recognition for.
- **input_type** (*str, optional*) – The type of input data. Defaults to "dialogue".

Returns
The recognition response in JSON format, or None if there is an error.

Return type
Optional[str]

Raises
Exception – If there is an error generating recognition response.

src.api_handlers.**get_recommendation_response**(*input_data: str*) → dict | None

Get a recommendation response for a given input.

Parameters
input_data (*str*) – The input data to generate recommendations for.

Returns
The recommendation response in JSON format, or None if there is an error.

Return type
Optional[dict]

Raises
Exception – If there is an error generating recommendations response.

src.audio_handlers module

src.audio_handlers.**audio_handler**() → None

Handles audio input (either via URL or uploaded file) for dialogue recognition and recommendation.

Displays a preview of the audio file and performs transcription, recognition, and recommendation processing.

src.audio_processing module

src.audio_processing.**audio_processing**(*uploaded_file*) → str | None

Process an uploaded audio file and return its path in WAV format.

Parameters
uploaded_file (*st.uploaded_file_manager.UploadedFile*) – The uploaded audio file.

Returns
Path to the processed WAV file or None if an error occurs.

Return type
Optional[str]

src.audio_processing.**convert_mp3_to_wav**(*mp3_path: str*) → str | None

Convert an MP3 file to WAV format.

Parameters
mp3_path (*str*) – Path to the MP3 file.

Returns
Path to the converted WAV file or an error message.

Return type
Optional[str]

src.audio_processing.**transcribe_audio_to_text**(*audio_path: str*) → str

Transcribe a WAV audio file to text using Google's Web Speech API.

Parameters
audio_path (*str*) – Path to the WAV audio file.

Returns
Transcribed text or error message if transcription fails.

Return type
str

src.image_handlers module

src.image_handlers.**image_handler**() → None

Handle image input, process it for recognition and recommendation, and display the results.

This function gets an image prompt from the user (either via URL or upload), processes the image, and sends it to the recognition and recommendation APIs to retrieve and display results.

It displays a preview of the image and handles any errors during the recognition and recommendation processes.

src.text_handlers module

src.text_handlers.**text_handler**() → None

Handles text input for dialogue recognition and recommendation.

It processes the dialogue entered by the user, sends it to recognition and recommendation APIs, and displays the results in a grid format.

src.video_handlers module

src.video_handlers.**video_handler**() → None

Handle video input, extract frames, and combine them into a grid.

This function gets a video file from the user, processes it to extract frames, and combines those frames into a grid layout. The combined image is then saved to the specified path.

src.video_processing module

src.video_processing.**combine_frames_in_grid**(*frames: List[Image]*, *output_path: str*, *rows: int = 5*, *cols: int = 5*) → str | None

Combine frames into a grid and save the combined image.

Parameters

- **frames** (*List[Image.Image]*) – List of frames to combine.
- **output_path** (*str*) – Path where the combined image will be saved.
- **rows** (*int*) – Number of rows in the grid.
- **cols** (*int*) – Number of columns in the grid.

Returns
Path to the saved combined image, or None if an error occurred.

Return type
Optional[str]

src.video_processing.**extract_frames**(*video_path: str*, *interval: int = 1*) → List[Image]

Extract frames from a video file at specified intervals.

Parameters

- **video_path** (*str*) – Path to the video file.
- **interval** (*int*) – Interval (in seconds) at which frames are extracted.

Returns
List of frames extracted from the video as PIL Images.

Return type
List[Image.Image]

Module contents (src)

PYTHON MODULE INDEX

INDEX

L

M

P

R

S

T

V