Code Documentation

app.py

Function	Description
main()	The main function that sets up the Streamlit web application for an AI chatbot interface. It configures the page layout and initializes the login status. Depending on the user's login status, it displays different tabs for "About," "Login," and "Sign up," or the chat interface.

login_signup.py

Function	Description
send_email(recipient_email)	Sends an email with a new password to the specified recipient's email address. The new password is generated using the `generate_password()` function. The email includes instructions to reset the password after login.
forgot_password()	Renders the "Forgot Password" form where users can enter their email address. If the email exists in the database, an email with a new password is sent to the user's email address for password recovery.

load_robot_lottie()	Loads and returns a JSON representation of a Lottie animation (a type of animation file format).
is_valid_email(email)	Validates whether the provided email address follows a valid email pattern using regular expressions.
login()	Renders the login interface where users can enter their email and password for authentication. Handles user authentication and login status updates.
signup()	Renders the sign-up interface where new users can create an account by providing their email and password. Handles user registration and login status updates.
about()	Renders the "About Us" section, providing information about the application's purpose, features, and benefits. This function also includes animations and styling for the about section.

chat.py

Function	Description
load_robot_lottie()	Loads and returns a JSON representation of a Lottie animation (a type of animation file format). This function is used to load the animation that is displayed when the chatbot is initialized.
new_chat(user_id)	Generates a unique chat ID based on the current user's ID and the current timestamp. This ID is used to identify a chat session.
store_chat()	Stores the chat information into a binary file named "chats.bin". It updates the chat history and metadata for the current chat session.
clear()	Clears the user's input query, allowing them to input a new query.
chat_page()	Renders the main chat interface where users can interact with the AI chatbot. This function handles user inputs, chat history, file uploads, query processing, and menu navigation.

Sim.py

Function	Description
valid_link(link)	Checks if a given link is a valid YouTube video link.
save_uploaded_file(text, uploaded_file)	Saves an uploaded file to the specified path.
get_pdf_text(pdf)	Reads and extracts text content from a PDF file.
remove_punctuation(text)	Removes punctuation marks from the provided text.
remove_stopwords(query)	Removes stopwords (commonly used words that carry little meaning) from the input query.
fill_context(text)	Processes and encodes sentences using a pre-trained sentence embedding model for semantic similarity searching.
similarity_search(index, q, sentences, offset, threshold, p=None)	Performs semantic similarity search within a context using a sentence embedding model and Faiss (a similarity search library).
summary(text)	Generates a summary of a given text using the Hugging Face model "facebook/bart-large-cnn".

one_line(text, q)	Extracts a concise answer from the given context using the Hugging Face model "deepset/tinyroberta-squad2".
title_generator(text)	Generates a title for the provided text using the Hugging Face model "czearing/article-title-generator".
punctuate(text)	Adds appropriate punctuation to the provided text using the Hugging Face model "oliverguhr/fullstop-punctuation-multilang-large".
neural_coreference(text)	Resolves coreferences (e.g., pronouns referring to previous nouns) in the given text using neuralcoref.
get_transcript(url)	Retrieves and processes the transcript of a YouTube video specified by the URL.

audio.py

Function	Description
video_to_audio(input_video_path, output_audio_path)	Converts a video file into an audio file (MP3 format) and saves it to the specified output path. Uses the MoviePy library to extract the audio from the video.

mp3_to_wav(input_file, output_file)	Converts an MP3 audio file to WAV format and saves it to the specified output path. Uses the PyDub library for audio format conversion.
query(filename, API_URL, headers)	Sends a POST request to a specified API URL with an audio file's content and receives a response in JSON format. Used for making queries to a speech recognition model.
chunks(filename)	Divides a WAV audio file into smaller audio chunks of a specified size (in milliseconds). Returns the number of chunks created.
audio_to_text(filename)	Converts an audio file (MP3) to text using the Hugging Face model "openai/whisper-large-v2." Utilizes the query function to process audio chunks and retrieve text from each chunk. Manages the conversion of audio files, querying, and cleaning up temporary files.

database.py

Function	Description
generate_password(length=12)	Generates a random password of specified length using a

	combination of letters, digits, and punctuation characters.
hash_password_sha256(plain_password, salt=None)	Hashes a plain password using the SHA-256 algorithm along with an optional salt. Returns the hexadecimal representation of the password hash concatenated with the salt.
verify_password_sha256(plain_password, hashed_password)	Verifies if a plain password matches a given hashed password. Extracts the salt from the hashed password, hashes the plain password using the salt, and compares the result to the stored hash.
check_user(email)	Checks if a user with the specified email exists in the database. Connects to the MySQL database and executes a query to fetch user data based on the provided email. Returns a boolean indicating if the user exists.
get_password(email)	Retrieves the hashed password for a user with the specified email. Connects to the MySQL database and retrieves the hashed password associated with the provided email.

create_user(email, password)	Creates a new user in the database with the given email and hashed password. Generates a unique user ID, checks for duplicate emails, hashes the password, and inserts the user data into the Users table.
validate_login(email, password)	Validates user login credentials. Checks if the provided email exists and if the hashed password matches the stored hash.
reset_password(userid, old_password, new_password)	Resets the password for a user. Validates the old password, hashes the new password, and updates the password in the Users table for the specified user.
change_forgot_password(email, new_password)	Changes the password for a user using a forgot password flow. Hashes the new password and updates it in the Users table for the user with the provided email.
delete(userid)	Deletes a user from the database and associated chat history.