API Document

API Summary List

This section provides a detailed list of all APIs offered by this project.

Backend server

API	Description
`POST /api/login`	Allows a user to log in using email and password. Returns a JWT token upon successful authentication.
`GET /api/token_status`	Returns the remaining expiration time (in seconds and minutes) for the current valid JWT token.
`POST /api/send_verification`	Sends a verification code to the user’ s email for purposes such as registration or password reset.
`POST /api/register`	Register a new user with email, password, and verification code.
`POST /api/logout`	Log out the currently authenticated user and invalidate the token.
POST /api/update_password	Update the password of the current user using a verification code.
`POST /api/chat`	Send a message to the LLM and stream its response, while also forwarding partial results to a video model. Messages are saved to the session.
`POST /api/sessionid`	Save a real-time communication (RTC) session ID in Redis for the current user.
`POST /api/chat/new`	Create a new chat session.
`GET /api/chat/history`	Get the list of all chat sessions of the current user.
`GET /api/message/list?session_id=<session_id>`	Retrieve all messages under a specific session.
`POST /api/chat/<chat_id>/favorite`	

	Set or unset a chat session as the user’ s favorite. Only one session can be marked as favorite at a time.
`DELETE /api/chat/<chat_id>`	Delete a chat session owned by the current user.
`POST /api/llm/activate`	Forward a model activation request to the model service. Used to activate a specific LLM backend.
`POST /api/upload`	Allows a user to upload a document to their personal knowledge base. Upload will be forwarded based on user role.
`DELETE /api/upload/<file_name>`	Delete an uploaded file by name from the user's knowledge base.
`GET /api/upload/users`	Allows a tutor to retrieve a list of all users.
`GET /api/user_files`	Fetch the current user’ s uploaded files.
`GET /api/public_files`	Fetch all publicly shared files. Only accessible by tutors.
`GET /api/user/profile`	Retrieve the current user's profile information.
`POST /api/user/profile`	Update the current user’ s profile fields such as username, full name, bio, etc. Changes are logged in the audit table.
`POST /api/user/avatar`	Upload and update the current user's avatar. The image is saved under `/static/avatars/user_<id>.jpg` .
`GET /api/user/notifications`	Get a paginated list of notifications for the current user.
`PUT /api/user/notifications/<notification_id>/read`	Mark a specific notification as read.
`DELETE /api/user/delete_account`	Delete the current user's account. Requires verification code for confirmation. The deletion is logged before execution.
`GET /api/admin/users`	Retrieve a paginated list of users. Can filter by `search` , `role` , and `status` .
`POST /api/admin/users`	Create a new user (by tutor).

`PUT /api/admin/users/<user_id>`	Update a user's role, status, or profile fields. Changes are logged.
`DELETE /api/admin/users/<user_id>`	Delete a user account and log the deletion reason.
`GET /api/admin/models`	Retrieve a list of models.
`GET /api/admin/knowledge`	Retrieve a list of knowledge base entries.
`GET /api/admin/user-action-logs`	Retrieve a paginated list of user action logs (e.g., updates, deletions).
`GET /api/avatar/list`	Fetch the list of available avatars from the avatar service.
`POST /api/avatar/preview`	Request and preview an avatar’ s image from the avatar service.
`POST /api/avatar/add`	Add a new avatar with optional face and voice prompts. Only accessible to tutors.
`POST /api/avatar/delete`	Delete an avatar by name via avatar service.
`POST /api/avatar/start`	Start an avatar by name.
`GET /api/tts/models`	Fetch available TTS (Text-To-Speech) models from the avatar service.

LLM

API	Description
`POST /chat/stream`	Main chat interface that provides streaming responses in real-time.
`GET /health`	Check the health status of the API server.
`POST /activate_model`	Switch between different Ollama models.

RAG

API	Description

`POST /user/upload`	This API allows a user to upload documents into their personal knowledge base.
`POST /user/delete`	This API allows a user to delete ingested documents from their personal knowledge base.
`POST /admin/upload`	This API allows an administrator to upload documents into the shared public knowledge base.
`POST /admin/delete`	This API allows an administrator to delete documents from the shared public knowledge base.
`POST /retriever`	This API performs retrieval for a question and returns the retrieved results.
`GET /api/users`	This API gets all users IDs who have personal knowledge base currently.
`GET /api/user_files`	This API gets the list of documents in the specific user's personal knowledge base.
`GET /api/public_files`	This API gets the list of documents in the public knowledge base.

Model management server

API	**Description**
`/avatar/get_avatars`	Get avatar info
`/avatar/preview`	Get avatar preview
`/avatar/add`	Create a new avatar
`/avatar/delete`	Delete specified avatar
`/avatar/start`	Start specified avatar

TTS Models

API	**Description**
`/tts/models`	Get TTS server configs
`/tts/start`	Start specified TTS server
`/tts/response`	Generate TTS Response

Lip-sync Models

API	Description
`/switch_avatar`	By specifying the avatar name, shut down the original lip-sync service and start the new backend service
`/create_avatar`	Create a temporary file required for inference based on the avatar name, the video path uploaded by the user, and whether it is blurred

Backend server APIs

User Login API

Usage Scenario

Used when a user logs into the system using their email and password.

Precautions

- Requires valid email and password.
- Any previously issued token will be revoked.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/login`
- **Authentication Required:** No

Request Parameters

Header

Name	Type	Required	Description
Content-Type	string	Yes	application/json

Body (JSON)

Name	Type	Required	Description
email	string	Yes	

			User's email address
password	string	Yes	User's login password
sessionid	string	No	Optional session ID

Response Parameters

Field	Type	Description
token	string	The JWT access token for the user

Notes

- The server revokes previous token sessions.
- Stores session ID in Redis if provided.

Request Example

代码块

```
POST /api/login HTTP/1.1
Content-Type: application/json
{
  "email": "user@example.com",
  "password": "securepassword"
}
```

Response Example

代码块

```
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUz..."
}
```

Status Codes

Code	Meaning

200	Login successful
400	Bad request (invalid data)
401	Unauthorized (invalid creds)
500	Server error

Token Status API

Usage Scenario

Used to check the expiration time of the currently authenticated JWT token.

Precautions

- Must be called with a valid access token.
- Token must still exist in Redis.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/token_status`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Parameters

Field	Type	Description
expires_in_seconds	int	Remaining token lifetime (seconds)
expires_in_minutes	float	Remaining token lifetime (minutes)

Request Example

```
代码块
1  GET /api/token_status HTTP/1.1
2  Authorization: Bearer eyJhbGciOi...
```

Response Example

```
代码块
1  {
2    "expires_in_seconds": 3599,
3    "expires_in_minutes": 59.98
4  }
```

Status Codes

Code	Meaning
200	Token valid, expiration returned
401	Unauthorized or expired token

Send Verification Code API

Usage Scenario

Sends a verification code to user’ s email for either registration or password reset.

Precautions

- Rate limited (e.g., 10 seconds between requests).
- If email is already registered and purpose is `register`, request will be rejected.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/send_verification`
- **Authentication Required:** No

Request Parameters

Header

Name	Type	Required	Description
Content-Type	string	Yes	application/json

Body (JSON)

Name	Type	Required	Description
email	string	Yes	Email to send verification code
purpose	string	Yes	register or reset

Response Parameters

Field	Type	Description
msg	string	Status message

Request Example

代码块

```
{
  "email": "user@example.com",
  "purpose": "register"
}
```

Response Example

代码块

```
{
  "msg": "Verification code sent."
}
```

Status Codes

Code	Meaning
200	Code sent successfully
400	Email missing
409	Email already registered
429	Rate limit exceeded

User Registration API

Usage Scenario

Registers a new user using email, password, and a verification code.

Precautions

- Verification code must be valid and not expired.
- Will fail if the email is already registered.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/register`
- **Authentication Required:** No

Request Parameters

Header

Name	Type	Required	Description
Content-Type	string	Yes	application/json

Body (JSON)

Name	Type	Required	Description
email	string	Yes	Email address to register
password	string	Yes	User password

code	string	Yes	Verification code
role	string	No	User role (default: student)

Response Parameters

Field	Type	Description
msg	string	Status message

Request Example

代码块

```
{
  "email": "user@example.com",
  "password": "securepass",
  "code": "123456",
  "role": "student"
}
```

Response Example

代码块

```
{
  "msg": "Registration successful"
}
```

Status Codes

Code	Meaning
201	User registered successfully
400	Missing required fields
401	Invalid or expired code
409	Email already registered

User Logout API

Usage Scenario

Logs out the current user and invalidates the token in Redis.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/logout`
- **Authentication Required:** Yes

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response

代码块

```
{
  "msg": "Logged out successfully."
}
```

Status Codes

Code	Meaning
200	Logout successful
401	Unauthorized or token error

Update Password API

Usage Scenario

Update a user’ s password using a verification code (without requiring current password).

Request Explanation

- **Method:** `POST`
- **URL:** `/api/update_password`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (JSON)

Name	Type	Required	Description
code	string	Yes	Verification code
new_password	string	Yes	New password to set

Response

代码块

```
{
  "msg": "Password updated successfully"
}
```

Status Codes

Code	Meaning
200	Password updated
400	Missing fields or invalid code
401	Unauthorized

404	User not found

Create New Chat Session

Usage Scenario

Used to start a brand new chat session for the current user.

Precautions

- If `title` is not provided, the session will be titled as “Untitled”.
- Each user can have multiple sessions.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/chat/new`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (JSON)

Name	Type	Required	Description
title	string	No	Optional title of the session

Response Parameters

Field	Type	Description
session_id	int	ID of the created session

Request Example

```
代码块
1  {
2    "title": "Customer Support Inquiry"
3  }
```

Response Example

```
代码块
1  {
2    "session_id": 12
3  }
```

Status Codes

Code	Meaning
201	Session created successfully
401	Unauthorized

List All Chat Sessions

Usage Scenario

Fetch all chat sessions for the current user.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/chat/history`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Parameters

List of chat sessions, each with:

Field	Type	Description
id	int	Session ID
title	string	Session title
created_at	string	ISO timestamp
updated_at	string	ISO timestamp
message_count	int	Number of messages in session
is_favorite	bool	Whether this session is marked as favorite

Response Example

代码块

```
[
  {
    "id": 1,
    "title": "Math Tutor",
    "created_at": "2025-08-05T01:00:00Z",
    "updated_at": "2025-08-05T01:30:00Z",
    "message_count": 10,
    "is_favorite": false
  }
]
```

Status Codes

Code	Meaning
200	Success
401	Unauthorized

Get Message List in Session

Usage Scenario

Retrieve all messages (user + assistant) in a given chat session.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/message/list?session_id=<id>`
- **Authentication Required:** Yes

Request Parameters

Query

Name	Type	Required	Description
session_id	int	Yes	Target session ID

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Parameters

List of messages:

Field	Type	Description
role	string	user or assistant
content	string	Message text
created_at	string	ISO timestamp
file_type	string	Type of attached file (if any)

file_path	string	Path to attached file

Response Example

代码块

```
[
  {
    "role": "user",
    "content": "What's the capital of France?",
    "created_at": "2025-08-05T01:01:00Z",
    "file_type": null,
    "file_path": null
  },
  {
    "role": "assistant",
    "content": "Paris is the capital of France.",
    "created_at": "2025-08-05T01:01:02Z",
    "file_type": null,
    "file_path": null
  }
]
```

Status Codes

Code	Meaning
200	Messages returned
401	Unauthorized
404	Session not found or forbidden

Mark a Session as Favorite

Usage Scenario

Marks a specific chat session as the user's favorite. Only one session can be favorite at a time.

Request Explanation

- **Method:** `POST`

- **URL:** `/api/chat/<chat_id>/favorite`
- **Authentication Required:** Yes

Request Parameters

Path

Name	Type	Required	Description
chat_id	int	Yes	ID of the chat

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (JSON)

Name	Type	Required	Description
is_favorite	bool	No	Whether to mark as favorite

Response Example

代码块

```
{
  "msg": "Favorite status updated",
  "session_id": 1,
  "is_favorite": true
}
```

Status Codes

Code	Meaning
200	Successfully updated
401	Unauthorized

404	Session not found

Delete a Chat Session

Usage Scenario

Deletes a specific chat session owned by the current user.

Request Explanation

- **Method:** `DELETE`
- **URL:** `/api/chat/<chat_id>`
- **Authentication Required:** Yes

Request Parameters

Path

Name	Type	Required	Description
chat_id	int	Yes	ID of the session

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
{
  "msg": "Session 3 deleted"
}
```

Status Codes

Code	Meaning
200	Deleted successfully
401	Unauthorized
404	Session not found

Save RTC Session ID

Usage Scenario

Saves a session ID for real-time avatar or chat streaming, mapped to the user.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/sessionid`
- **Authentication Required:** Yes

Request Parameters

Body (JSON)

Name	Type	Required	Description
sessionid	string	Yes	RTC session ID

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response

Empty `200` response if successful

Status Codes

Code	Meaning
200	Saved successfully
400	Missing session ID
401	Unauthorized
404	User not found

Activate Specific LLM Model

Usage Scenario

Activates one of the available LLMs (e.g., LLaMA or GPT model) to respond to user queries.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/llm/activate`
- **Authentication Required:** Yes

Request Parameters

Body (JSON)

Name	Type	Required	Description
model	string	Yes	Model name to activate

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
{
  "message": "Model llama3.1:8b-instruct-q4_K_M has been activated, other models have been closed."
}
```

Status Codes

Code	Meaning
200	Model activated
400	Missing or invalid input
401	Unauthorized
500	Model backend service error

Upload a File to User Knowledge Base

Usage Scenario

Allows a user to upload a document (PDF, TXT, DOCX, etc.) into their personal knowledge base.

Precautions

- Requires login
- File is forwarded to internal file service
- Upload path depends on user role:
  - student → `/user/upload`
  - tutor → `/admin/upload`

Request Explanation

- **Method:** `POST`
- **URL:** `/api/upload`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>
Content-Type	string	Yes	multipart/form-data

Body (FormData)

Name	Type	Required	Description
file	file	Yes	Document to upload
...	any	No	Additional metadata

Response Parameters

Field	Type	Description
msg	string	Upload status message
...	any	Other fields from upload service

Request Example

代码块

```
POST /api/upload
Content-Type: multipart/form-data
Authorization: Bearer <token>

FormData:
file = example.pdf
user_id = 12
```

Response Example

代码块

```
{
  "msg": "Upload successful",
  "source_name": "example.pdf"
}
```

Status Codes

Code	Meaning
200	Upload successful
400	Missing file
403	Invalid user role
404	User not found
500	Upload service connection failed

Delete a User File

Usage Scenario

Allows a user to delete a file they previously uploaded to their knowledge base.

Precautions

- Role-sensitive behavior:
  - student → requires `user_id`
  - tutor → only needs `source_name`

Request Explanation

- **Method:** `DELETE`
- **URL:** `/api/upload/<file_name>`
- **Authentication Required:** Yes

Request Parameters

Path

Name	Type	Required	Description
file_name	string	Yes	Name of file to delete

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
{
  "msg": "File deleted successfully"
}
```

Status Codes

Code	Meaning
200	Deletion successful
403	Permission denied
404	User not found
500	Delete forwarding failed

Fetch All Users (Tutor Only)

Usage Scenario

Allows a tutor to retrieve the list of all registered users.

Precautions

- Tutor only
- Proxies request to `/api/users` on internal service

Request Explanation

- **Method:** `GET`

- **URL:** `/api/upload/users`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
[
  {
    "id": 12,
    "email": "user@example.com",
    ...}
]
```

Status Codes

Code	Meaning
200	User list fetched
403	Tutor permission required
500	Service call failed

Fetch Current User’ s Uploaded Files

Usage Scenario

Returns all files uploaded by the currently authenticated user.

Request Explanation

- **Method:** `GET`

- **URL:** `/api/user_files`
- **Authentication Required:** Yes

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
[
  {
    "source_name": "myfile.pdf",
    "chunk_count": 20
  },
  ...
]
```

Status Codes

Code	Meaning
200	File list returned
404	User not found
500	Service error

Fetch Public Files (Tutor Only)

Usage Scenario

Returns files that are publicly available to all tutors.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/public_files`
- **Authentication Required:** Yes

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
[
  {
    "source_name": "datasheet.txt",
    "chunk_count": 42
  },
  ...
]
```

Status Codes

Code	Meaning
200	Public files listed
403	Permission denied (not tutor)
500	File service unavailable

Get Current User Profile

Usage Scenario

Retrieves the currently authenticated user's profile, including email, username, name, avatar, and bio.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/user/profile`
- **Authentication Required:** Ye

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
{
  "email": "user@example.com",
  "username": "john",
  "full_name": "John Doe",
  "avatar_url": "/static/avatars/user_1.jpg",
  "bio": "I'm a developer.",
  "role": "student"
}
```

Status Codes

Code	Meaning
200	Profile retrieved successfully
404	User or profile not found

User Profile

Usage Scenario

Updates editable profile fields such as username, full name, phone number, and bio.

Request Explanation

- **Method:** `POST`

- **URL:** `/api/user/profile`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>
Content-Type	string	Yes	application/json

Body (JSON)

Name	Type	Required	Description
username	string	No	New username
full_name	string	No	New full name
phone	string	No	Phone number
bio	string	No	Short biography

Response Example

代码块

```
{
  "msg": "User profile updated successfully"
}
```

Status Codes

Code	Meaning
200	Profile updated successfully
404	User or profile not found

Upload User Avatar

Usage Scenario

Uploads a new profile avatar for the current user.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/user/avatar`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>
Content-Type	string	Yes	multipart/form-data

Body (FormData)

Name	Type	Required	Description
avatar	file	Yes	Image file (.jpg, .png, .jpeg)

Response Example

代码块

```
{
  "msg": "Avatar uploaded successfully.",
  "avatar_url": "/static/avatars/user_3.jpg"
}
```

Status Codes

Code	Meaning
200	Avatar uploaded
400	Invalid or missing file

Get Notifications

Usage Scenario

Retrieves a paginated list of system notifications for the current user.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/user/notifications`
- **Authentication Required:** Yes

Query Parameters

Name	Type	Required	Description
page	int	No	Page number (default: 1)
limit	int	No	Number per page (default: 10)
unread_only	bool	No	Only return unread (default: false)

Response Example

代码块

```
{
  "notifications": [
    {
      "id": 1,
      "title": "Welcome!",
      "message": "Thanks for joining us.",
      "type": "info",
      "is_read": false,
      "created_at": "2025-08-05T01:10:00Z"
    }
  ],
  "total": 1,
  "unread_count": 1,
  "page": 1,
  "limit": 10
}
```

Status Codes

Code	Meaning
200	Notifications retrieved
401	Unauthorized

Mark Notification as Read

Usage Scenario

Marks a specific notification as read.

Request Explanation

- **Method:** `PUT`
- **URL:** `/api/user/notifications/<notification_id>/read`
- **Authentication Required:** Yes

Request Parameters

Path

Name	Type	Required	Description
notification_id	int	Yes	ID of the notification

Response Example

代码块

```
{
  "msg": "Notification marked as read"
}
```

Status Codes

Code	Meaning
200	Marked successfully
401	Unauthorized
404	Notification not found

Delete Account

Usage Scenario

Deletes the current user account after verifying a verification code.

Precautions

- Requires verification code
- Deletes user and logs the action

Request Explanation

- **Method:** `DELETE`
- **URL:** `/api/user/delete_account`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (JSON)

Name	Type	Required	Description
code	string	Yes	Verification code

Response Example

代码块

```
{
  "msg": "Account deleted successfully"
}
```

Status Codes

Code	Meaning
200	Account deleted
400	Code missing or invalid
404	User not found

Get User List (Tutor Only)

Usage Scenario

Returns a paginated and filterable list of all users in the system. Accessible only by users with the `tutor` role.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/admin/users`
- **Authentication Required:** Yes (tutor)

Query Parameters

Name	Type	Required	Description
page	int	No	Page number (default: 1)
limit	int	No	Items per page (default: 20)
search	string	No	Search by email/username/fullname
role	string	No	Filter by user role
status	string	No	Filter by user status

Response Example

代码块

```
{
  "users": [
    {
      "id": "U001",
      "email": "user@example.com",
      "username": "john",
      "full_name": "John Smith",
      "role": "student",
      "status": "active",
```

```
      "created_at": "2025-08-01T00:00:00Z",
      "last_login": "2025-08-04T10:00:00Z"
    }
  ],
  "total": 50,
  "page": 1,
  "limit": 20
}
```

Status Codes

Code	Meaning
200	User list returned
403	Unauthorized (not tutor)

Create a New User (Tutor Only)

Usage Scenario

Allows a tutor to create a new user manually.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/admin/users`
- **Authentication Required:** Yes (tutor)

Body (JSON)

Name	Type	Required	Description
email	string	Yes	New user's email
password	string	Yes	Password
username	string	Yes	Display name
role	string	Yes	Role (e.g., student, tutor)
full_name	string	No	Full name
phone	string	No	Phone number
bio	string	No	Short bio
avatar_url	string	No	Avatar image path

Response Example

代码块

```
{
  "msg": "User created successfully",
  "user_id": "001"
}
```

Status Codes

Code	Meaning
201	User created
400	Missing required fields
403	Unauthorized (not tutor)
409	Email already exists

Update User Information (Tutor Only)

Usage Scenario

Allows tutors to update another user's role, status, or profile fields.

Request Explanation

- **Method:** `PUT`
- **URL:** `/api/admin/users/<user_id>`
- **Authentication Required:** Yes (tutor)

Path

Name	Type	Required	Description
user_id	int	Yes	ID of the user

Body (JSON)

Name	Type	Required	Description
email	string	No	Update email
role	string	No	Update user role
status	string	No	Update user status
username	string	No	Update username

Response Example

```
代码块
1  {
2    "msg": "User updated successfully"
3  }
```

Status Codes

Code	Meaning
200	Update successful
400	Missing or invalid input
403	Unauthorized (not tutor)
404	User not found

Delete User Account (Tutor Only)

Usage Scenario

Allows tutors to permanently delete a user and optionally log a reason.

Request Explanation

- **Method:** `DELETE`
- **URL:** `/api/admin/users/<user_id>`
- **Authentication Required:** Yes (tutor)

Path

Name	Type	Required	Description
user_id	int	Yes	ID of the user

Body (JSON)

Name	Type	Required	Description
reason	string	No	Reason for deletion

Response Example

代码块

```
{
  "msg": "User deleted successfully"
}
```

Status Codes

Code	Meaning
200	User deleted
403	Unauthorized (not tutor)
404	User not found

Get Model List

Usage Scenario

Returns a mock list of LLM models for display or management purposes.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/admin/models`
- **Authentication Required:** Yes (tutor)

Query Parameters

Name	Type	Required	Description
page	int	No	Page number (default 1)
limit	int	No	Items per page

Response Example

代码块

```
{
  "models": [
    {
      "id": "M001",
      "name": "GPT-3.5",
      "status": "Active",
      "type": "LLM",
      "owner": "OpenAI"
    }
  ],
  "total": 2,
  "page": 1,
  "limit": 20
}
```

Status Codes

Code	Meaning
200	Mock model list returned
403	Unauthorized (not tutor)

Get Knowledge Base List

Usage Scenario

Returns a mock list of knowledge bases.

Request Explanation

- **Method:** `GET`

- **URL:** `/api/admin/knowledge`
- **Authentication Required:** Yes (tutor)

Query Parameters

Name	Type	Required	Description
page	int	No	Page number (default 1)
limit	int	No	Items per page

Response Example

代码块

```
{
  "knowledge": [
    {
      "id": "K001",
      "name": "Python Basics",
      "status": "Active",
      "type": "Tutorial"
    }
  ],
  "total": 2,
  "page": 1,
  "limit": 20
}
```

Status Codes

Code	Meaning
200	Knowledge list returned
403	Unauthorized (not tutor)

Get User Action Logs

Usage Scenario

Returns a paginated list of user activity logs, including who modified or deleted what.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/admin/user-action-logs`
- **Authentication Required:** Yes (tutor)

Query Parameters

Name	Type	Required	Description
operator_email	string	No	Filter by operator
target_user_email	string	No	Filter by target user email
action	string	No	Filter by action (update, delete)
page	int	No	Page number (default: 1)
per_page	int	No	Items per page (default: 20)

Response Example

代码块

```
{
  "logs": [
    {
      "id": 1,
      "operator_email": "admin@example.com",
      "action": "update",
      "target_user_email": "user@example.com",
      "reason": "Permission update",
      "timestamp": "2025-08-05T10:00:00Z"
    }
  ],
  "total": 10,
  "page": 1,
  "per_page": 20
}
```

Status Codes

Code	Meaning
200	Logs returned
403	Unauthorized (not tutor)

Get Available Avatars

Usage Scenario

Retrieves a list of available avatars from the avatar service.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/avatar/list`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Response Example

代码块

```
[
  {
    "name": "Sophie",
    "description": "Professional female voice avatar",
    "model": "model-A",
    ...}
]
```

Status Codes

Code	Meaning
200	Avatar list retrieved
500	Avatar service connection failed

Preview an Avatar

Usage Scenario

Requests an avatar preview image from the avatar service and returns it as a binary stream.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/avatar/preview`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (FormData)

Name	Type	Required	Description
avatar_name	string	Yes	Name of the avatar

Response

Returns an image stream (typically `image/png`)

Status Codes

Code	Meaning
200	Avatar preview image returned
400	Missing avatar name
500	Avatar service error

Add a New Avatar (Tutor Only)

Usage Scenario

Allows a tutor to create a new avatar by uploading prompt voice and/or face images.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/avatar/add`
- **Authentication Required:** Yes (tutor only)

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (FormData)

Name	Type	Required	Description
name	string	Yes	Avatar name
avatar_blur	string	No	Blur level (optional)
support_clone	string	No	Whether avatar supports cloning
timbre	string	No	Voice timbre tag
tts_model	string	No	TTS model to use
avatar_model	string	No	Avatar model name
description	string	No	Description of the avatar
prompt_face	file	No	Optional image of the face
prompt_voice	file	No	Optional voice sample

Response Example

代码块

```
{
  "status": "success",
  "avatar_id": "avatar_123"
}
```

Status Codes

Code	Meaning
200	Avatar added successfully
400	Avatar creation failed
403	Unauthorized (not tutor)
500	Avatar service error or bad JSON

Delete an Avatar

Usage Scenario

Deletes an avatar by name via avatar backend service.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/avatar/delete`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (FormData)

Name	Type	Required	Description
name	string	Yes	Name of the avatar

Response Example

代码块

```
{
  "status": "success",
  "msg": "Avatar deleted"
}
```

Status Codes

Code	Meaning
200	Avatar deleted
400	Missing name or deletion failed
500	Avatar service error or bad JSON

Start an Avatar

Usage Scenario

Start a specific avatar for real-time interaction.

Request Explanation

- **Method:** `POST`
- **URL:** `/api/avatar/start`
- **Authentication Required:** Yes

Request Parameters

Header

Name	Type	Required	Description
Authorization	string	Yes	Bearer <access_token>

Body (FormData)

Name	Type	Required	Description
avatar_name	string	Yes	Name of the avatar

Response Example

代码块

```
{
  "status": "success",
  "msg": "Avatar started"
}
```

Status Codes

Code	Meaning
200	Avatar started
400	Missing avatar name or failed
500	Avatar service error or bad JSON

Get TTS Models

Usage Scenario

Fetches the list of available TTS models that avatars can use for voice generation.

Request Explanation

- **Method:** `GET`
- **URL:** `/api/tts/models`
- **Authentication Required:** Yes

Response Example

代码块

```
[
  {
    "model": "gtts-en",
    "language": "en",
    "description": "Standard English TTS"
  }
]
```

Status Codes

Code	Meaning
200	TTS model list returned
500	TTS service connection failed

LLM APIs

Overview

The RAG AI Assistant provides a sophisticated conversational AI system with streaming responses, intelligent query classification, and multi-source retrieval capabilities. The API is built with Flask and supports real-time streaming via Server-Sent Events (SSE).

Base URL

```
代码块
1  http://localhost:8100
```

Authentication

Currently, the API does not require authentication. All endpoints are publicly accessible.

Content Types

- **Request**: `application/json`
- **Response**: `application/json` or `text/event-stream` (for streaming endpoints)

Endpoints

1. Streaming Chat Interface

Endpoint:

`POST /chat/stream`

Description:

Main chat interface that provides streaming responses in real-time.

Request Body:

```
代码块
1  {
2    "user_id": "string",
3    "session_id": "string",
4    "input": "string"
5  }
```

Parameters:

- `user_id` (required): Unique identifier for the user
- `session_id` (required): Session identifier for conversation continuity
- `input` (required): User's message/query

Response: Server-Sent Events (SSE) stream

Stream Format:

代码块

```
data: {"chunk": "response text", "status": "streaming", "timestamp": "2024-01-01T00:00:00"}
data: {"chunk": "more text", "status": "streaming", "timestamp": "2024-01-01T00:00:01"}
data: {"status": "finished", "timestamp": "2024-01-01T00:00:02"}
```

Example Request:

代码块

```
curl -X POST http://localhost:8100/chat/stream \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": "user123",
    "session_id": "session456",
    "input": "What is the TCP/IP model?"
  }'
```

Example Response Stream:

代码块

```
data: {"chunk": "The TCP/IP model is a conceptual framework used to describe the functions of a networking system.", "status": "streaming", "timestamp": "2024-01-01T00:00:00"}
data: {"chunk": " It consists of four layers: Application, Transport, Internet, and Network Access.", "status": "streaming", "timestamp": "2024-01-01T00:00:01"}
data: {"status": "finished", "timestamp": "2024-01-01T00:00:02"}
```

Error Response:

```
代码块
{
    "error": "Missing required field: user_id"
}
```

2. Health Check

Endpoint: `GET /health`

Description: Check the health status of the API server.

Request: No parameters required

Response:

```
代码块
{
  "status": "healthy",
  "timestamp": "2024-01-01T00:00:00"
}
```

Example:

```
代码块
curl http://localhost:8100/health
```

3. Model Activation

Endpoint: `POST /activate_model`

Description: Switch between different Ollama models.

Request Body:

```
代码块
1  {
2    "model": "string"
3  }
```

Parameters:

- `model` (required): Model name to activate

Supported Models:

- `mistral-nemo:12b-instruct-2407-fp16` (default)
- `llama3.1:8b-instruct-q4_K_M`

Response:

```
代码块
1  {
2    "message": "Model mistral-nemo:12b-instruct-2407-fp16 has been activated,
   other models have been closed."
3  }
```

Error Response:

```
代码块
1  {
2    "error": "Please specify the model name to activate, e.g. {'model': 'mistral-
   nemo:12b-instruct-2407-fp16'}"
3  }
```

Example:

```
代码块
1  curl -X POST http://localhost:8100/activate_model \
```

```
    -H "Content-Type: application/json" \
    -d '{"model": "llama3.1:8b-instruct-q4_K_M"}'
```

Milvus API Integration

The system integrates with an external Milvus vector database service for document retrieval.

Milvus API Configuration

Base URL: `http://localhost:9090`

Query Endpoint: `/retriever`

Health Endpoint: `/retriever`

Milvus Query Parameters

When the system performs RAG retrieval, it sends the following parameters to the Milvus API:

代码块

```
{
  "question": "string",
  "user_id": "string",
  "personal_k": 3,
  "public_k": 3,
  "final_k": 4,
  "threshold": 0.3
}
```

Parameters:

- `question` : The search query
- `user_id` : User identifier for personal knowledge base

- `personal_k` : Number of results from personal knowledge base
- `public_k` : Number of results from public knowledge base
- `final_k` : Total number of final results
- `threshold` : Similarity threshold for results

Workflow Process

The API follows this workflow for each chat request:

1. **Guardrail Check**: Classifies content as normal/homework_request/harmful
2. **Query Rewrite**: Reformulates the query for better retrieval
3. **Query Classification**: Determines if RAG, web search, or direct response is needed
4. **Retrieval/Search**: Fetches relevant documents or web results
5. **Response Generation**: Generates streaming response with context

Error Handling

HTTP Status Codes

- `200` : Success
- `400` : Bad Request (missing required fields, invalid input)
- `500` : Internal Server Error

Error Response Format

代码块

```
{
  "error": "Error description"
}
```

Common Errors

6. **Missing Required Fields**

```
代码块
1  {
2    "error": "Missing required field: user_id"
3  }
```

7. **Connection Errors**

```
代码块
1  {
2    "error": "Connection failed - please ensure the API server is running"
3  }
```

8. **Model Activation Errors**

```
代码块
1  {
2    "error": "Please specify the model name to activate"
3  }
```

Rate Limiting

Currently, no rate limiting is implemented. Consider implementing rate limiting for production use.

CORS

The API includes CORS headers for cross-origin requests:

- `Access-Control-Allow-Origin: *`
- `Access-Control-Allow-Headers: Content-Type`

Session Management

The system uses SQLite-based checkpointing for session persistence:

- Session data is stored in `checkpoints.db`
- Each session maintains conversation history
- Sessions are identified by `session_id`

Content Safety

The system includes built-in content safety features:

Query Classification

- **normal**: General academic queries, course information
- **homework_request**: Requests for direct answers to assignments
- **harmful**: Inappropriate or dangerous content

Response Filtering

- Homework requests are blocked with educational responses
- Harmful content is rejected with policy reminders
- All responses are generated with academic integrity in mind

Testing

Use the provided test script to verify API functionality:

```
代码块
1   python rag/test_simple_api.py
```

The test script includes:

- Streaming API functionality test
- Health check validation
- Error handling verification

Dependencies

The API requires the following external services:

- **Ollama**: Local LLM service (port 11434)
- **Milvus API**: Vector database service (port 9090)
- **Tavily API**: Web search service (external)

Configuration

Environment variables can be used to configure the system:

代码块

```
export TAVILY_API_KEY="your_api_key"
export MILVUS_API_BASE_URL="http://localhost:9090"
export MILVUS_API_TIMEOUT="30"
```

Monitoring and Logging

The API includes comprehensive logging:

- Request/response logging
- Error tracking
- Performance monitoring
- Workflow state tracking

Logs are output to stdout with INFO level by default.

RAG APIs

Overview

This API document provides a detailed description of the Flask-based multi-user knowledge-base management and retrieval service backend interfaces, including user file upload and deletion, public file management, retrieval, and metadata queries.The database is based on Milvus.

Server Startup

- **Host:** `0.0.0.0`
- **Port:** `9090`
- **Temporary File Root Directory:** /home/jialu/workspace/jialu/tmp

User Load File

`POST /user/upload` allows a user to upload a file into the user’ s personal knowledge base. The service ensures the personal collection exists (creating it if missing), saves the file to a user-specific directory, and runs the ingestion pipeline with standard (non-admin) context. The response includes metadata about the ingestion (e.g., embedding counts, collection names, page/chunk statistics) along with `admin: false` and the `user_id` .

Usage Scenarios

- **Personal knowledge enrichment:** Users build a private knowledge base by uploading lecture notes, project documents, or papers, allowing an LLM to answer questions based on that content.
- **Updating or replacing previous uploads:** When a user wants to update an existing document, they first delete the outdated version’ s embeddings and then upload the new file so the personal knowledge base stays current and avoids duplication.

Request Description

Method	`POST`
Endpoint	`/user/upload`

Request Parameters

Header

代码块

```
Content-Type: multipart/form-data
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`user_id`	`string`	Yes	The unique identifier of the user.	`Alice`
`file`	`string`	Yes	The file path of the file to be uploaded.	`/data/kb/Alice/lecture_notes.pdf`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`admin`	`boolean`	false	Whether the upload was done with admin (`false` here).
`chunk_collection`	`string`	"kb_user_abc_chunk"	Identifier of the chunk/text knowledge collection, following the pattern `kb_{user_id}_chunk`.
`chunk_embs_num`	`integer`	43	Count of text chunks generated.
`ingest_file`	`string`	"file_ingested.pdf"	Original name of the uploaded file.
`page_collection`	`string`	"kb_user_abc_page"	Identifier of the page-level knowledge collection, following the pattern `kb_{user_id}_page`.
`page_count`	`integer`	34	Count of pages extracted from the document.
`page_embs_num`	`integer`	35054	Count of embeddings generated from the document’ s pages.
`user_id`	`string`	"user_abc"	The target user’ s identifier.

Error Responses

Missing parameters (HTTP 400)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Request must include both user_id and file"	human-readable reason.

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"File upload or processing failed, please try again later"	generic error summary.
`detail`	`string`	"Traceback (most recent call last): ..."	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{
 "admin":false,
 "chunk_collection":"kb_user_abc_chunk",
 "chunk_embs_num":43,
 "ingest_file":"Practice_Exam_Answers.pdf",
 "page_collection":"kb_user_abc_page",
 "page_count":34,
 "page_embs_num":35054,
 "user_id":"abc"
}
```

2. Missing parameters (HTTP 400)

```
代码块
1  {
2    "error": true,
3    "message": "Request must include both user_id and file"
4  }
```

3. Internal server error (HTTP 500)

```
代码块
1  {
2    "error": true,
3    "message": "File upload or processing failed, please try again later",
4    "detail": "Detailed exception message"
5  }
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	File saved and ingested successfully.
400	**Bad Request**	Missing `user_id` or `file` in request.
500	**Internal Server Error**	Knowledge Base collection failed or file ingestion failed.

User Delete File

`POST /user/delete` allows a user to remove a specific ingested document from their personal knowledge base. The service ensures the user's personal collection exists, then deletes all page-level and chunk-level embeddings from `source_name`. The response returns the counts of deleted page embeddings and chunk embeddings.

Usage Scenarios

- **File update/replacement:** Before uploading a revised version of a document, the user deletes the outdated source's embeddings to avoid duplication.
- **Undo mistaken upload:** A user accidentally ingests the wrong file or source and uses this endpoint to remove its embeddings before correcting the mistake.

Request Description

Method	`POST`
Endpoint	`/user/delete`

Request Parameters

Header

代码块

```
Content-Type: multipart/form-data
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`user_id`	`string`	Yes	The unique identifier of the user.	`Alice`
`source_name`	`string`	Yes	Original filename to be deleted(with suffix)	`lecture_notes.pdf`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`deleted_chunks`	`integer`	43	Number of chunk-level embeddings deleted for the specified source in the user’s personal knowledge base.
`deleted_page_embs`	`integer`	35054	Number of page-level embeddings deleted for the specified source in the user’s personal knowledge base.

Error Responses

Missing parameters (HTTP 400)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Request must include both user_id and source_name"	human-readable reason.

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Deletion failed, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{
 "deleted_chunks":43,
 "deleted_page_embs":35054
}
```

2. Missing parameters (HTTP 400)

代码块

```
{
  "error": true,
  "message": "Request must include both user_id and source_name"
}
```

3. Internal server error (HTTP 500)

```
代码块
1  {
2    "error": true,
3    "message": "Deletion failed, please try again later",
4    "detail": "Detailed exception message"
5
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	File deleted successfully.
400	**Bad Request**	Missing `user_id` or `source_name` in request.
500	**Internal Server Error**	File deletion failed.

Admin Load File

`POST /admin/upload` allows an administrator to upload a file into the shared public knowledge base. The service ensures the public collection exists, saves the file to a dedicated admin directory, and runs the ingestion pipeline with elevated `(is_admin=true)` context. The response includes metadata about the ingestion (e.g., embedding counts, collection names) along with an `admin: true` flag.

Usage Scenarios

- **Preloading shared content:** Administrators upload standard reference materials, templates, or guidelines so that all users can immediately leverage them in RAG-powered queries.
- **Updating or replacing public documents:** When a public document becomes outdated or needs revision (e.g., policy, curriculum), the old version’ s embeddings are deleted first and the new version can be uploaded to ensure the knowledge base reflects the latest content.
- **Providing common support resources:** Upload FAQs, user manuals, or training guides that serve many users without requiring per-user uploads.

Request Description

Method	`POST`
Endpoint	`/admin/upload`

Request Parameters

Header

代码块

```
Content-Type: multipart/form-data
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`file`	`string`	Yes	The file path of the file to be uploaded.	`/data/kb/Alice/lecture_notes.pdf`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`admin`	`boolean`	true	Whether the upload was done with admin (`true` here).
`chunk_collection`	`string`	"kb_admin_public_chunk"	Name of the text/chunk-level collection where chunk embeddings were stored (public knowledge base for admin).
`chunk_embs_num`	`integer`	43	Number of chunk-level embeddings generated from the ingested file.
`ingest_file`	`string`	"file_ingested.pdf"	Original name of the uploaded file.
`page_collection`	`string`	"kb_admin_public_page"	Name of the page-level collection used for page embeddings.

`page_count`	`integer`	34	Total number of pages extracted from the document.
`page_embs_num`	`integer`	35054	Count of embeddings generated from the document’ s pages.

Error Responses

Missing parameters (HTTP 400)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Request must include file"	human-readable reason.

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Public file upload or processing failed, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{
 "admin":true,
 "chunk_collection":"kb_admin_public_chunk",
 "chunk_embs_num":43,
 "ingest_file":"Practice_Exam_Answers.pdf",
 "page_collection":"kb_admin_public_page",
```

```
  "page_count":34,
  "page_embs_num":35054
}
```

2. Missing parameters (HTTP 400)

代码块

```
{
  "error": true,
  "message": "Request must include file"
}
```

3. Internal server error (HTTP 500)

代码块

```
{
  "error": true,
  "message": "Public file upload or processing failed, please try again later",
  "detail": "Detailed exception message"
}
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	File uploaded successfully.
400	**Bad Request**	Missing `file` in request.
500	**Internal Server Error**	File ingestion failed.

Admin Delete File

`POST /admin/delete` allows an administrator to remove a previously ingested public source from the shared knowledge base. The service ensures the public collection exists, then deletes all page-level and chunk-level embeddings associated with the given `source_name`. The response returns the counts of deleted page embeddings and chunk embeddings.

Usage Scenarios

- **Updating or replacing a public document:** Before uploading a revised version, the admin deletes the old source’ s embeddings to avoid stale or conflicting content.
- **Knowledge base management:** The administrator can prune low-quality, redundant, or stale public sources to keep the shared KB relevant.

Request Description

Method	`POST`
Endpoint	`/admin/delete`

Request Parameters

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`source_name`	`string`	Yes	Original filename to be deleted(with suffix)	`lecture_notes.pdf`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`deleted_chunks`	`integer`	43	Number of chunk-level embeddings deleted for the specified source in the public knowledge base.
	`integer`	35054	

`deleted_page_embs`			Number of page-level embeddings deleted for the specified source in the public knowledge base.

Error Responses

Missing parameters (HTTP 400)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Request must include source_name"	human-readable reason.

Internal server failure (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Deletion failed, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{
 "deleted_chunks":43,
 "deleted_page_embs":35054
}
```

2. Missing parameters (HTTP 400)

```
{
  "error": true,
  "message": "Request must include source_name"
}
```

3. Internal server failure (HTTP 500)

代码块

```
{
  "error": true,
  "message": "Deletion failed, please try again later",
  "detail": "Detailed exception message"

```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	File deleted successfully.
400	**Bad Request**	Missing `source_name` in request.
500	**Internal Server Error**	File deletion failed.

Retrieve

`POST /retriever` performs a hybrid retrieval over a user’s personal and the shared public knowledge bases to collect contextual hits for a given question. The request provides the `question`, the `user_id`, `personal_k`, `public_k`, `final_k`. Internally it runs page-level and chunk-level retrieval, merges them via a composite strategy, and returns the ranked hits along with an associated image path (if any) for downstream LLM.

Usage Scenarios

- **Personalized Q&A enhancement:** When a student asks a question about the uploaded file, the system can retrieve the relevant results to enhance the answers generated by downstream LLM.

Request Description

Method	`POST`

Endpoint	`/retriever`

Request Parameters

Header

代码块

```
Content-Type: application/json
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`question`	`string`	Yes	Questions the user wants to ask.	`"Who are Course Staff"`
`user_id`	`string`	Yes	The unique identifier of the user.	`"Alice"`
`personal_k`	`integer`	Yes	Specifies the number of top-ranked chunks to retrieve from the user’ s personal knowledge base.	`5`
`public_k`	`integer`	Yes	Specifies the number of top-ranked chunks to retrieve from the public knowledge base.	`5`
`final_k`	`integer`	Yes	Specifies the number of top-ranked chunks to retrieve from both public knowledge base and personal knowledge base.	`5`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`hits`	`array(dict)`	See below in **Note**	List of retrieval hits results.(List of results dicts) See below in **Note**

Error Responses

1. Missing parameters (HTTP 400)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Request must include both user_id and file"	human-readable reason.

2. Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"File upload or processing failed, please try again later"	generic error summary.
`detail`	`string`	"Traceback (most recent call last): ..."	exception/message for debugging

Note

> Provides a detailed explanation of the element and fields in the `hits` list.The element in `hits` list is `Dict`

Parameter Name	Type	Example	Description
`first_score`	`float`	`0.398`	Similarity score between this chunk and query in the first stage.

`second_score`	`float`	`-11.168`	Similarity score using cross-encoder reranker in the second stage.
`fused_score`	`float`	`0.615`	Combined/normalized score after merging `first_score` and `second_score`.
`page_num`	`integer`	`3`	Page number within the source document where this hit originated.
`source`	`string`	`"Course Outline & Logistics.pdf"`	Filename of the source document.
`text`	`string`	`"Course Staff - Lecturer-in-Charge: ..."`	Extracted text chunk providing context for the hit.
`knowledge_base`	`string`	`"personal"`	The text is from a personal knowledge base/public knowledge base.

Response Example

1. Success Response (HTTP 200)

代码块


```
{"hits":[
 {
  "text": "Assessment - Hands-on - 40% - Labs 20% - Assignment 20% - Assignment released in Week 4, due in Week 10 - Implement a networked application or protocol - We assume you are proficient in one of C/Java/Python - Concepts and theory - 60% - Mid-term test (20%): - Tuesday, 26th March 2024, 09:00 - 11:00 (Week 7) - Open-book online exam (you can attempt the exam from anywhere) - Final Exam (40%) - End of term, date TBA - Open-book online exam (you can attempt the exam from anywhere) - Hurdle - must score at least 40% to pass the course - Inspera platform, accessible through Moodle - https://www.student.unsw.edu.au/exams/inspera",
  "source": "Course Outline & Logistics.pdf",
  "page_num": 20,
  "first_score": 0.39844226837158203,
  "second_score": -11.168756484985352,
  "final_score": 0.6155794629227487,
  "knowledge_base": "personal"
  }
 ...]}
```

2. Missing parameters (HTTP 400)

```
代码块
1  {
2    "error": true,
3    "message":
4    "Request must include question, user_id, personal_k, public_k, and final_k"
5  }
```

3. Internal server error (HTTP 500)

```
代码块
1
2  {
3    "error": true,
4    "message": "Retrieval failed, please try again later",
5    "detail": "Detailed exception message"
6  }
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	Retrieval succeeded.
400	**Bad Request**	Missing one or more required fields.
500	**Internal Server Error**	An unexpected error occurred.

Get user list in knowledge base

`GET /api/users` get the list of all user IDs that currently have personal knowledge base collections.

Usage Scenarios

- **Admin management:** List all users with knowledge bases so an admin can perform maintenance.

Request Description

Method	`GET`
Endpoint	`/api/users`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`users`	`array(string)`	`["user_1", "user_2", ...]`	List of all user IDs.

Error Responses

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Failed to retrieve user list, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{
    "users": ["alice", "bob", "carol"]
}
```

2. Internal Server Error (HTTP 500)

```
代码块
    "error": true,
    "message": "Failed to retrieve user list, please try again later",
    "detail": "Detailed exception message"
}
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	Successfully retrieved the user list.
500	**Internal Server Error**	Retrieval failed.

Get user file lists

`GET /api/user_files` get the list of documents given a `user_id`. The list includes all files this specific user has ingested into their knowledge base

Request Description

Method	`GET`
Endpoint	`/api/user_files`

Request Parameters

Header

```
代码块
Content-Type: application/json
```

Body

Parameter Name	Type	Required(Yes/No)	Description	Example
`user_id`	`string`	Yes		`"Alice"`

			The unique identifier of the user.	

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`files`	`array(string)`	`["file_1", "file_2", ...]`	User `user_id`’s ingested file list

Error Responses

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.
`message`	`string`	"Failed to retrieve user files, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{"files":["Course Outline & Logistics.pdf"]}
```

2. Internal Server Error (HTTP 500)

代码块

```
{
```

```
    "error": true,
    "message": "Failed to retrieve user files, please try again later",
    "detail": "Detailed exception message"
}
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	Successfully retrieved the user file list.
500	**Internal Server Error**	Retrieval failed.

Get public file lists

`GET /api/public_files` get the list of documents in public knowledge base. The list includes all files the administrators have ingested into the knowledge base.

Request Description

Method	`GET`
Endpoint	`/api/public_files`

Response Parameters

Success Response (HTTP 200)

Parameter Name	Type	Example	Description
`files`	`array(string)`	`["file_1", "file_2", ...]`	All files in public knowledge base.

Error Responses

Internal server error (HTTP 500)

Parameter Name	Type	Example	Description
`error`	`boolean`	true	always `true` on failure.

`message`	`string`	"Failed to retrieve public files, please try again later"	generic error summary.
`detail`	`string`	"Detailed exception message"	exception/message for debugging

Response Example

1. Success Response (HTTP 200)

代码块

```
{"files":["Course Outline & Logistics.pdf"]}
```

2. Internal Server Error (HTTP 500)

代码块

```
{
    "error": true,
    "message": "Failed to retrieve public files, please try again later",
    "detail": "Detailed exception message"
}
```

Status Code

Status Code	Reason Phrase	Description
200	**Success**	Successfully retrieved the public file list.
500	**Internal Server Error**	Retrieval failed.

Avatar APIs

Get avatar info

Retrieve all avatar information currently available in the system.

Usage Scenarios

1. **Admin Perspective**: In the Avatar management interface, retrieve all stored Avatars in the system to perform operations like addition or deletion.
2. **User Perspective**: On the homepage, users can switch between supported virtual tutor avatars by retrieving the list from the system.

Request Description

Request Method	`GET`
Request URL	`/avatar/get_avatars`

Response Parameters

代码块

```
{
  "avatar1": {
        "description": "This is the first avatar in our collection.",
        "clone": false,
        "timbre": "default"/null,
        "tts_model": "sovits",
        "avatar_model": "MuseTalk"
    }, ...
}
```

Remarks

- Items with a return value of null should not be displayed on the profile card.
- When clone is true, timbre must be null.
- When clone is false, timbre must be a string.

Response Examples

Normal Response Example:

代码块

```
{
  "avatar1": {
        "description": "This is the first avatar in our collection.",
        "clone": false,
        "timbre": "default"/null,
        "tts_model": "sovits",
```

```
        "avatar_model": "MuseTalk"
    }, ...
}
```

Error Response Example

代码块

```
{
  "detail": "Error description"
}
```

HTTP Status Code

HTTP Status Code	Description	Explanation	Example Message
200	Success	The request was successful	Avatar retrieved successfully
400	Failed to read avatar	The server failed to read the avatar	Unable to read avatar file
404	Avatar not found	The specified avatar does not exist	Avatar not found

---

Get avatar preview API

Get a preview image of the avatar with the specified name

Usage Scenarios

Admin Panel: Display Avatar Preview After Fetching Avatar Info

After retrieving avatar information from the admin interface, display the avatar preview image on the profile card list.

Notice

Future development may consider supporting the return of a preview video, which includes the tutor’ s appearance and the associated preview audio.

Request Description

Request Method	`POST`
Request URL	`/avatar/preview`

Response Examples

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Body

Parameter name	type	Required(Yes/No)	description	example
`name`	`String`	是	要获取preview的avatar名称	`Steven Jobs`

Response Examples

`image/png` image data stream

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Description	Explanation	Example Message
200	Deleted successfully	The avatar was successfully deleted	Avatar deleted successfully
400	Missing or malformed parameter	A required parameter is missing or invalid	Missing or invalid avatar_id
404	Avatar not found		Avatar not found

		The specified avatar does not exist	
500	Internal server error	The server encountered an unexpected error	Failed to delete avatar due to server error

---

Create a new Avatar

Create and save a new avatar profile into the system.

Usage Scenarios

In the Avatar Management interface, administrators can upload a reference video as the avatar’ s appearance, and select a TTS model to either choose a timbre or upload a reference audio file.

Request Description

Request Method	POST
Request URL	`/avatar/add`

Response Examples

Header

代码块

```
Content-Type: multipart/form-data
```

Body

Parameter Name	Type	Required (Yes/No)	Description	Example
`name`	string	Yes	Unique name of the Avatar (must not be duplicated)	Steven Jobs
`avatar_blur`	boolean	No	Whether to enable avatar blur effect	true

`support_clone`	boolean	Yes	Whether to support cloning functionality	true / false
`timbre`	string	Yes	Name of the timbre (must exist in the system)	Default
`tts_model`	string	Yes	TTS model to use, e.g., sovits	sovits
`avatar_model`	string	Yes	Avatar model to use, e.g., MuseTalk	MuseTalk
`prompt_face`	file (mp4)	Yes	Reference video file for the avatar appearance	video/file.mp4
`prompt_voice`	file (wav)	No	Reference audio file for the avatar voice	audio/file.wav
`description`	string	No	Description text for the avatar	“first avatar”

Response Examples

Parameter name	type	example	description
`status`	str	`success`	

Remarks

- It is recommended to refer to the UI: model_config.html
- The options for the dropdown searchable fields **[Model Selection]** and **[Timbre Selection]** should be obtained via the API GET /tts/models
- When support_clone is **false**:
  - timbre is **required**, and the selected timbre must exist in the timbre list corresponding to the selected model
  - prompt_voice and ref_text must be **null**
- When support_clone is **true**:
  - timbre must be **null**
  - prompt_voice and ref_text must be **provided**
- The reference text is hardcoded as:
  - "Hello, this is tutorNet speaking, what do you need? Do you want a cup of coffee?"

- The recorded audio must exactly match the above content.

Response Examples：

Normal Response Examples

```
代码块
1  {
2    "status": "success",
3  }
```

Error Response Examples：

```
代码块
1  {
2    "detail": "Error description"
3  }
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Description	Explanation	Example Message
200	Success, avatar added	The request was successful and the avatar was added	Avatar added successfully
400	Model or voice not found	The specified model or voice does not exist	Model abc123 not found
422	Invalid or unprocessable file format	The file format is incorrect or cannot be parsed (e.g., invalid mp4/wav)	Failed to parse uploaded file
500	Internal server error	The server encountered an unexpected error	Error saving avatar to database

Delete a specific avatar

Delete the avatar (including its image, character, or related resources) with the specified name.

Usage Scenarios

Administrators can delete a specified avatar to prevent users from using it.

Request Description

Request Method	`POST`
Request URL	`/avatar/delete`

Response Examples

Header

代码块

```
Content-Type: application/json
```

Body

Parameter name	type	Required(Yes/No)	description	example
name	string	true	Name of the Avatar to Delete	Steven Jobs

Response Examples

Parameter name	type	example
status	string	success

Response Examples：

Normal Response Examples

代码块

```
{
  "status": "success",
}
```

Error Response Examples：

代码块

```
{
  "detail": "Error description"
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Description	Explanation	Example Message
200 / 204	Deleted successfully	The avatar was successfully deleted	Avatar deleted successfully
400	Missing or malformed parameter	A required parameter is missing or invalid	Missing or invalid avatar_id
404	Avatar not found	The specified avatar does not exist	Avatar not found
500	Internal server error	The server encountered an unexpected error	Server error while deleting avatar

---

Start a Specified Avatar

Specify an avatar and launch it as the active background avatar.

Usage Scenarios

- After an administrator creates an avatar, they can designate it as the default running avatar and start it.

- When a user selects an avatar to use, the system will start the selected avatar and stop the currently running one.

Request Description

Request Method	POST
Request URL	`/avatar/start`

Response Examples

Header

代码块

```
Content-Type: application/json
```

Body

Parameter name	**type**	**Required(Yes/No)**	**description**	**example**
name	string	true	Name of the avatar to start	"steven"

Response Examples

Parameter name	**type**	**example**
status	string	success
port	int	5033

Remarks

- Use `GET /avatar/get_avatars` to get avatar list

Response Examples

Normal Response Examples：

```
{
  "status": "success",
  "port": 8205
}
```

Error Response Examples：

```
代码块
{
  "detail": "Error description"
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Description	Explanation	Example Message
200	Started successfully	The request was successful and processed	Request succeeded, avatar uploaded
400	Missing or malformed parameter	A required parameter is missing or invalid	Missing or invalid avatar_id
404	Avatar not found	The requested avatar resource does not exist	Avatar not found
500	Internal server error	The server encountered an unexpected error	Database error, avatar not saved

---

TTS Models APIs

Get TTS Server Information

Retrieve all available TTS (Text-to-Speech) models and their configuration details.

Usage Scenarios

- Fetch all available voice models for frontend display and user selection
- Check the current model status and license information
- Determine whether a model supports timbre (voice) customization

Notice

- **Request Frequency**: High-frequency calls are discouraged; consider caching the data on the client side for a certain period
- Some models are clone-type models (clone), and their timbre information may be null

Request Description

Request Method	GET
Request URL	/tts/models

Response Examples

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Response Examples

Parameter name	type	example	description
`model_name`	string	tts_model_1	Model identifier (key name)
`full_name`	string	Tacotron2	Full name of the model
`clone`	bool	false	Whether the model is a clone model
`status`	string	available	Current model status, e.g., available, down
`license`	string	MIT	Model’ s open-source license type
`timbres`	array	[“timbre1”]	List of available timbres (not present for clone models)
`cur_timbre`	string	timbre1	

			Currently used timbre (not applicable for clone models)

Remarks

The return value is a dictionary where the **key** is the model_name, and the **value** contains the corresponding parameters of that model.

Request Examples

代码块

```
GET /tts/models HTTP/1.1
Host: your-api-server.com
Content-Type: application/x-www-form-urlencoded
```

Response Examples

Normal Response Examples：

代码块

```
{
  "tacotron2": {
    "full_name": "Tacotron2",
    "clone": false,
    "status": "available",
    "license": "MIT",
    "timbres": ["timbre1", "timbre2"],
    "cur_timbre": "timbre1"
  },
  "voice_clone_1": {
    "full_name": "Custom Voice Clone",
    "clone": true,
    "status": "available",
    "license": "proprietary",
    "timbres": null,
    "cur_timbre": null
  },
  ...
}
```

Error Response Examples：

代码块

```
{
  "detail": "Model information not found."
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Description	Example
200	Success	Same as **Normal Response Examples**
500	Internal Server Error	"detail": "Model information not found."

启Start TTS Server

Start a specified TTS model service. If another service is already running, it will automatically stop before the new one starts.

Usage Scenarios

- Launch the TTS service to handle upcoming speech synthesis requests
- Restart the service when switching models or changing ports
- Toggle GPU mode on or off

Notice

- Any currently running TTS service will be terminated before the new one starts
- If the target port is already in use, an error will be returned
- The model must be in a valid state and have a valid file path
- The service must start within a specified timeout (60 seconds); otherwise, it will be terminated and an error will be returned
- Default port: 5033
- GPU is enabled by default

Request Description

Request Method	`POST`
Request URL	`/tts/start`

Response Examples

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Body

Parameter name	type	Required(Yes/No)	description	example
model_name	string	Yes	Name of the TTS model to start; must exist and be in an active state	tacotron2
port	int	No	Port number to start the service on (default: 5033)	5033
use_gpu	bool	No	Whether to use GPU to launch the model service (default: true)	true

Response Examples

Parameter name	type	example	description
status	string	“success”	Request status
message	string	TTS started successfully	TTS startup message
port	int	5033	Port the service is actually running on
model_name	string	tacotron2	Name of the model that was started
use_gpu	bool	true	Whether GPU is used

timbre	string/null	timbre1 / null	Current timbre (returns null if not available)

Request Examples

代码块

```
POST /tts/start HTTP/1.1
Host: your-api-server.com
Content-Type: application/x-www-form-urlencoded

model_name=tacotron2&port=5033&use_gpu=true
```

Response Examples

Normal Response Examples:

代码块

```
{
  "status": "success",
  "message": "TTS server 'tacotron2' started successfully.",
  "port": 5033,
  "model_name": "tacotron2",
  "use_gpu": true,
  "timbre": "timbre1"
}
```

Error Response Examples:

代码块

```
{
  "detail": "Model 'tacotron2' is not valid to use."
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP status code	描述	description	example

200	Success	Same as **Normal Response Examples**	同上Normal Response Examples
400	Bad Request	"detail": "Port 5033 is already in use."	"detail": "Port 5033 is already in use."
500	Server Error	"detail": "TTS model service startup timeout (over 60 seconds)"	"detail": "TTS model service startup timeout (over 60 seconds)"

Generate TTS Response (Audio Output)

Sends a text-to-speech (TTS) request to the currently running TTS server and returns the generated audio as a WAV file.

Usage Scenarios

- Convert user text into speech using the active TTS model;
- Customize voice output with text or audio prompts;
- Generate dynamic speech for chatbots, assistants, or announcements.

Notice

Requires a running TTS server started via `/tts/start`

- If the TTS server is not running, a 503 error will be returned;
- Depending on the TTS model (e.g. edgeTTS, tacotron, sovits), the prompt format varies internally;
- The actual TTS server is expected to expose a /generate endpoint;
- The response audio is returned as raw PCM bytes (audio/wav content converted to application/octet-stream).

Request Description

Request Method	POST
Request URL	/tts/response

Response Examples

Header

代码块

```
Content-Type: multipart/form-data
```

Body

Name	Type	Required	Description	Example
tts_text	string	Yes	Text to be synthesized into speech	"Hello, how can I help you?"
prompt_text	string	No	Text prompt (e.g., speaker ID or style reference)	"en-US-GuyNeural"
prompt_wav	UploadFile	No	Optional WAV file to guide voice/tone	audio/wav file

Response Examples

- **Content-Type**: application/octet-stream
- **Description**: Returns the raw PCM audio stream in WAV format.

Request Examples

代码块

```
curl -X POST http://localhost:8000/tts/response \
  -F "tts_text=Hello, welcome to our service." \
  -F "prompt_text=en-US-GuyNeural" \
  -F "prompt_wav=@/path/to/prompt.wav"
```

Response Examples

Normal Response Examples:

- binary content of WAV audio

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status	Description	Example Detail
503	No TTS server is running	"detail": "No TTS server is currently running."
500	TTS server did not return success	"detail": "Failed to get response from TTS server"

Lip-sync Models APIs

Add Tutor Face

This API is used to create a new avatar from a user-uploaded video file. The uploaded video is first processed, and then the MuseTalk project is used to generate avatar resources suitable for lip-sync services.

Usage Scenarios

- Automatically create intermediate avatar files after a user uploads a video, for use in later inference
- Allow system administrators to batch add new avatar resources

Notice

- **Input Format Limitation**: The input video must be in a supported format (.mp4, .avi, .mov, .mkv)
- **Processing Time**: Video processing may take some time (around 1 minute), so please wait patiently for the API response
- **File Path Requirement**: You must provide a valid path to an existing video file in the system
- **Blur Option**: Optionally enable blur processing for the generated avatar, useful in privacy-sensitive scenarios

Request Description

Request Method	POST
Request URL	`/create_avatar`

Response Examples

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Body

Parameter Name	Type	Required (Yes/No)	Description	Example
avatar_name	string	Yes	Name of the avatar, used as a unique identifier	avatar_1
video_path	string	Yes	Absolute path to the video file	/path/to/video.mp4
burr	boolean	No	Whether to apply blur processing (default: false)	true

Response Examples

Parameter Name	Type	Example	Description
status	string	success	Request status — success indicates success, error indicates failure
message	string	Avatar created successfully	Detailed message describing the result
image_path	string	/path/to/avatar_1.jpg	Path to the generated avatar image (returned on success)

Response Examples

Normal Response Examples：

代码块

```
{
  "status": "success",
```

```
  "message": "Avatar create succeed!",
  "image_path":
"/workspace/share/yuntao/LiveTalking/avatars/avatar_1/avatar_1.jpg"
}
```

Error Response Examples:

```
代码块
{
  "status": "error",
  "message": "Video file does not exist: /path/to/video.mp4"
}
{
  "status": "error",
  "message": "Unsupported video format. Please use .mp4, .avi, .mov, or .mkv
files"
}
{
  "status": "error",
  "message": "An error occurred while creating the avatar: Permission denied"
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP status code	Description	example
200	successful – API call was successful, but the actual result depends on the status field in the response	Normal JSON response
422	Parameter error – Missing or incorrectly formatted request parameters	Missing required parameter avatar_name or video_path
500	Internal server error – An unexpected error occurred while processing the request	System error or dependent service unavailable

Switch Avatar

This API is used to switch the currently running lip-sync service. It first stops the existing background lip-sync service, then starts a new one based on the specified avatar_name.

Usage Scenarios

- Used when an end user selects a different avatar and a switch is required.

Notice

- **Usage Limitation**: The avatar to be switched to must already exist in the avatar configuration file.
- **Request Frequency**: It is recommended to wait at least 30 seconds between requests to avoid service instability due to frequent switching.
- **Port Handling**: The API will automatically detect and terminate any existing process on port 8205.
- **Startup Time**: Starting a new avatar service takes some time. Please wait for the API to return a success response.

Request Description

Request Method	POST
Request URL	/switch_avatar

Response Examples

Header

代码块

```
Content-Type: application/x-www-form-urlencoded
```

Response Examples

Parameter Name	Type	Required (Yes/No)	Description	Example
avatar_id	string	Yes	Avatar ID, corresponding to the name of an already created avatar	avatar_1

ref_file	string	Yes	Absolute path to the reference audio file	/path/to/test.wav

Response Examples

Normal Response Examples:

代码块

```
{
  "status": "success",
  "message": "Successfully switched to avatar test_avatar_1. Service is now running on port 8205."
}
```

Error Response Examples:

代码块

```
{
  "status": "error",
  "message": "Script execution failed, process exited unexpectedly with return code: 1"
}
{
  "status": "error",
  "message": "An error occurred during execution: No such file or directory"
}
```

status code

The following lists only the status codes related to the business logic of this API. For additional statuses, please refer to the Common API Status Codes.

HTTP Status Code	Status	Description	Example
200	Success	API call succeeded, but check the status field in the response for the actual result	Normal JSON response
422	Parameter Error	Missing or incorrectly formatted parameters	Missing required parameter avatar_id or ref_file
500			

	Internal Server Error	Unexpected error occurred while processing the request	System error or dependent service unavailable

---

Public status code

Status Code	Description	Explanation	Example
200	OK	The request was successful and the requested data is returned.	Success
201	Created	The request was successful and a new resource was created.	Resource created successfully.
204	No Content	Commonly used for delete operations. The request was successful but returns no content.	—
400	Bad Request	The request format is incorrect, usually due to parameter or syntax errors.	Invalid request format.
401	Unauthorized	Authentication is required — the request was not authenticated.	Authentication is required.
403	Forbidden	The server understood the request but refuses to authorize it (insufficient permissions).	Access denied.
404	Not Found	The requested resource does not exist or the path is incorrect.	The requested resource was not found.
405	Method Not Allowed	The HTTP method used is not supported for this endpoint (e.g., unsupported POST/GET/PUT).	HTTP method not supported.

429	Too Many Requests	The request rate exceeds the allowed limit — throttling is needed.	Rate limit exceeded.
500	Internal Server Error	An unexpected error occurred on the server, usually due to code or system issues.	An unexpected error occurred on the server.