API / Schema DOC

This API is a conversational AI system with CRM and RAG support. It is Built using FastAPI all endpoint is built using RestFul Principles. This system store user and has a history of past chat which use had done and support dynamic document. we use swagger UI to interact and visualize RestAPI.

Endpoint 1 : Chat

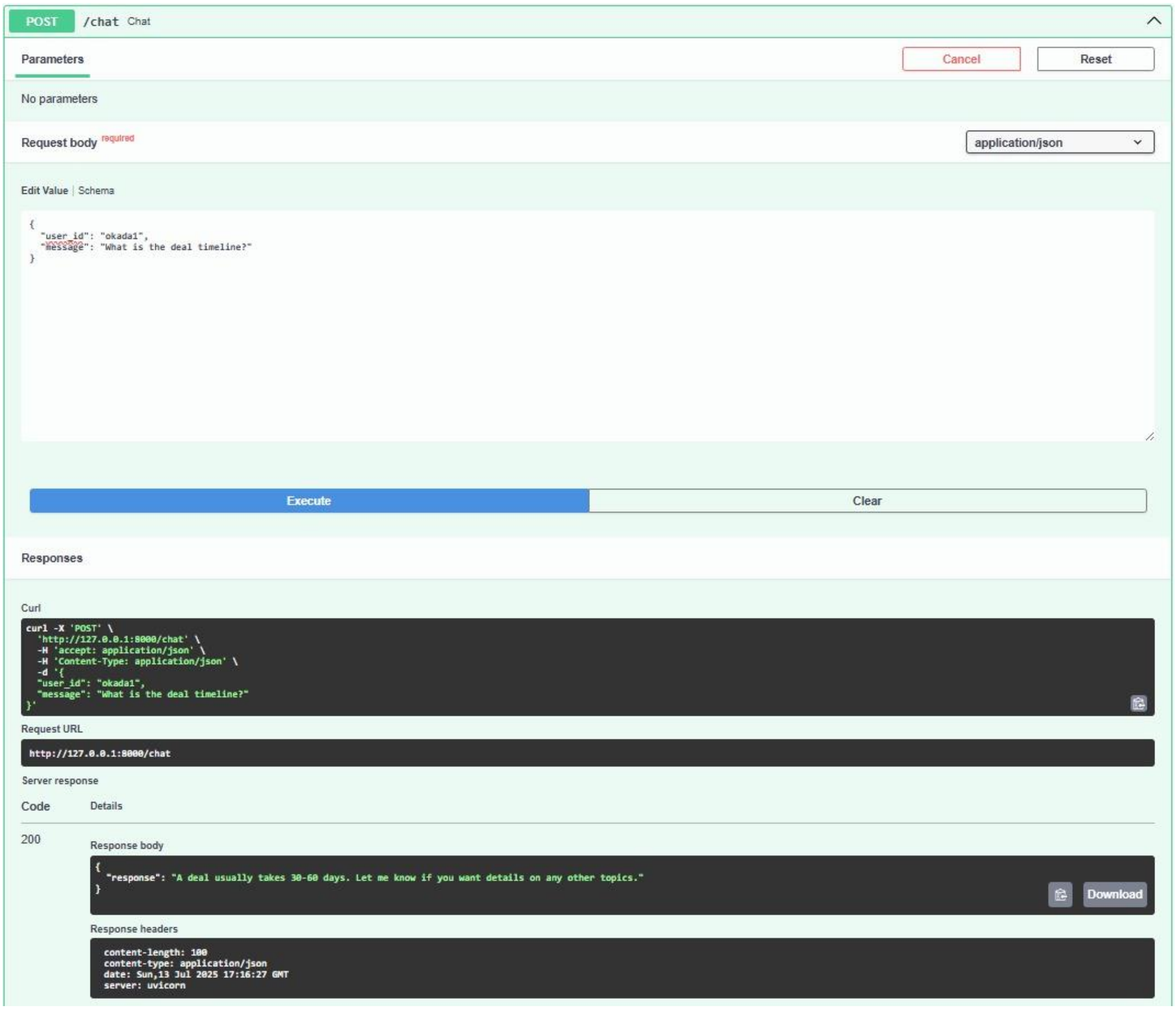
/chat

Here user can interact with chatbot with his/her user_id and message so chat can understand or track the chat and it's chat history using user_id.

Schema :

```
class ChatRequest(BaseModel):
    user_id: str = Field(..., example="okada1")
    message: str = Field(..., example="What is the deal timeline?")


class ChatResponse(BaseModel):
    response: str = Field(..., example="A deal usually takes 30-60 days.")
```



```
{
  "user_id": "okada1",
  "message": "What is the deal timeline?"
}
```

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/chat' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "user_id": "okada1",
  "message": "What is the deal timeline?"
}'
```

Request URL

```
http://127.0.0.1:8000/chat
```

Server response

Code | Details

200

Response body

```
{
  "response": "A deal usually takes 30-60 days. Let me know if you want details on any other topics."
}
```

Response headers

```
content-length: 100
content-type: application/json
date: Sun,13 Jul 2025 17:16:27 GMT
server: uvicorn
```

Endpoint 2: Upload CSV/JSON/PDF/TXT.

/upload_docs

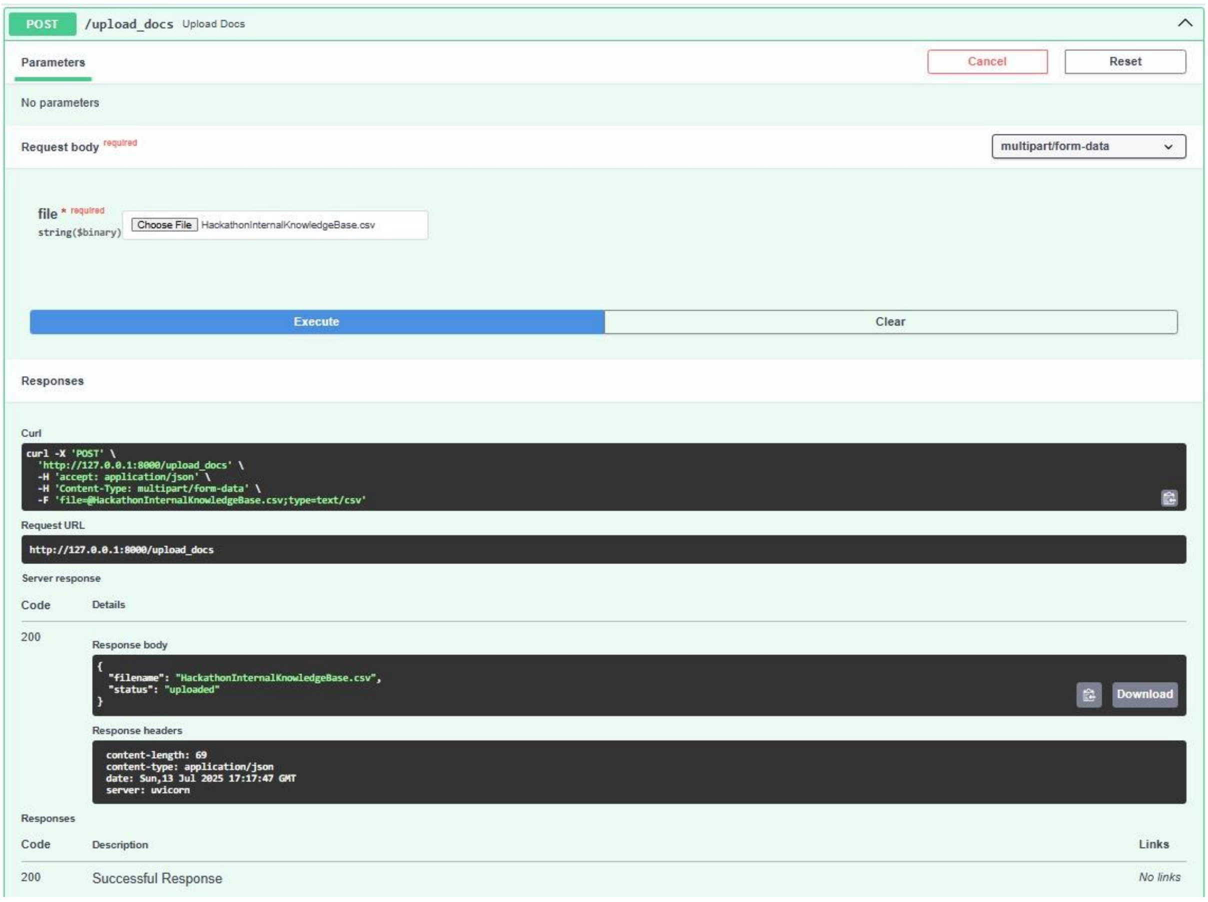
User can upload document in many formats like .csv,json,pdf and text to populate RAG knowledge base.

Schema :

class UploadDocsResponse(BaseModel):

  filename: str = Field(..., example="HackathonInternalKnowledgeBase.csv")

  status: str = Field(..., example="uploaded")


POST /upload_docs Upload Docs

Parameters

No parameters

Request body required — multipart/form-data

file * required string($binary) — HackathonInternalKnowledgeBase.csv

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/upload_docs' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@HackathonInternalKnowledgeBase.csv;type=text/csv'
```

Request URL

```
http://127.0.0.1:8000/upload_docs
```

Server response

Code 200 — Response body

```
{
  "filename": "HackathonInternalKnowledgeBase.csv",
  "status": "uploaded"
}
```

Response headers

```
content-length: 69
content-type: application/json
date: Sun,13 Jul 2025 17:17:47 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	No links

Endpoint 3 : Create a user in CRM

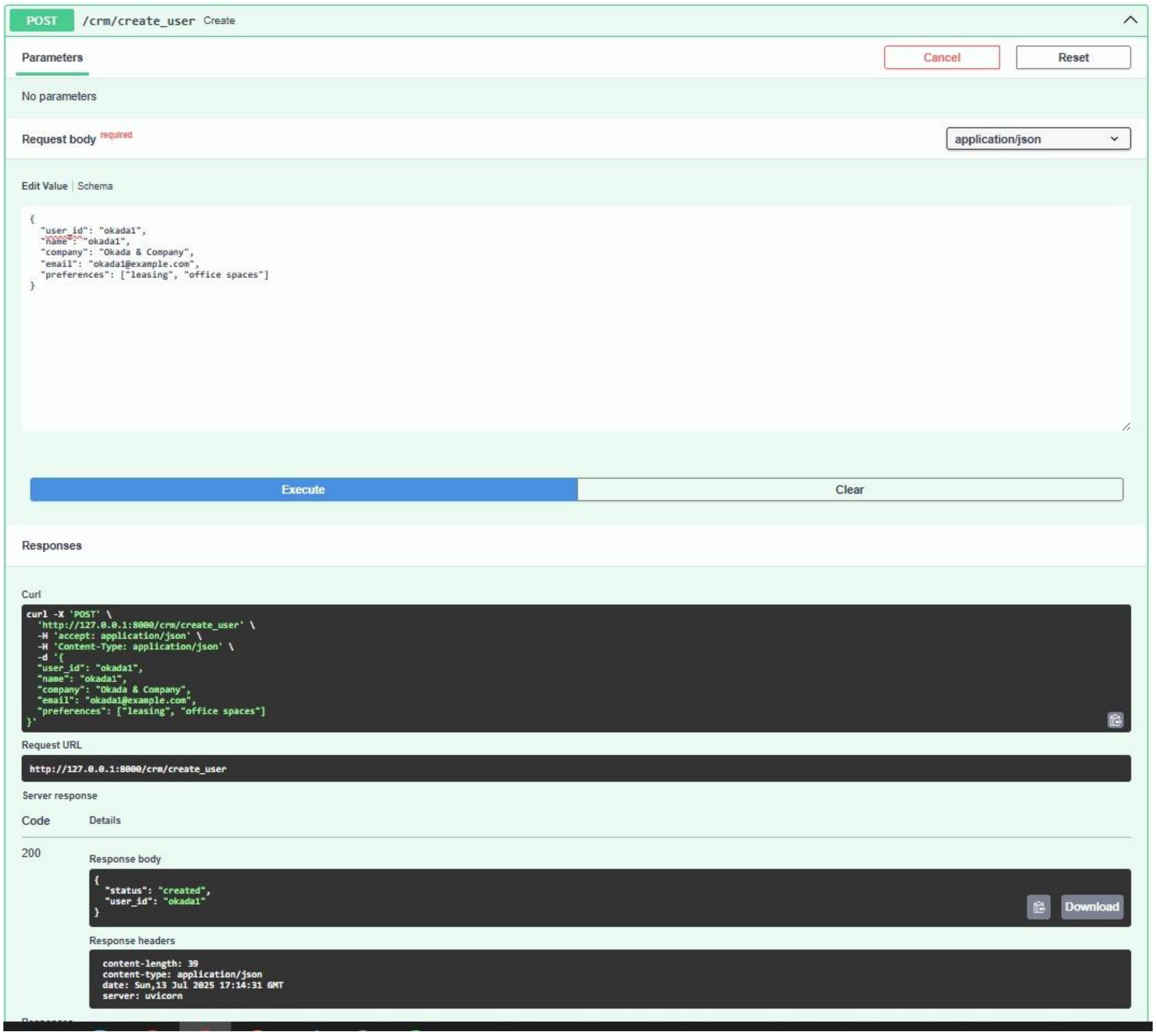
/crm/create_user

Here we are creating user and storing user detail into database and we used mongodb to storer user data.

Schema :

```
class CreateUserRequest(BaseModel):
    user_id: str
    name: str
    email: EmailStr
    company: Optional[str] = None
    preferences: Optional[List[str]] = []
```

```
class CreateUserResponse(BaseModel):
    message: str = Field(..., example="User created successfully.")
    user_id: str
```



```
{
  "user_id": "okada1",
  "name": "okada1",
  "company": "Okada & Company",
  "email": "okada1@example.com",
  "preferences": ["leasing", "office spaces"]
}
```

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/crm/create_user' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "user_id": "okada1",
  "name": "okada1",
  "company": "Okada & Company",
  "email": "okada1@example.com",
  "preferences": ["leasing", "office spaces"]
}'
```

Request URL

```
http://127.0.0.1:8000/crm/create_user
```

Server response

Code Details

200 Response body

```
{
  "status": "created",
  "user_id": "okada1"
}
```

Response headers

```
content-length: 39
content-type: application/json
date: Sun,13 Jul 2025 17:14:31 GMT
server: uvicorn
```

Endpoint 4: Update user

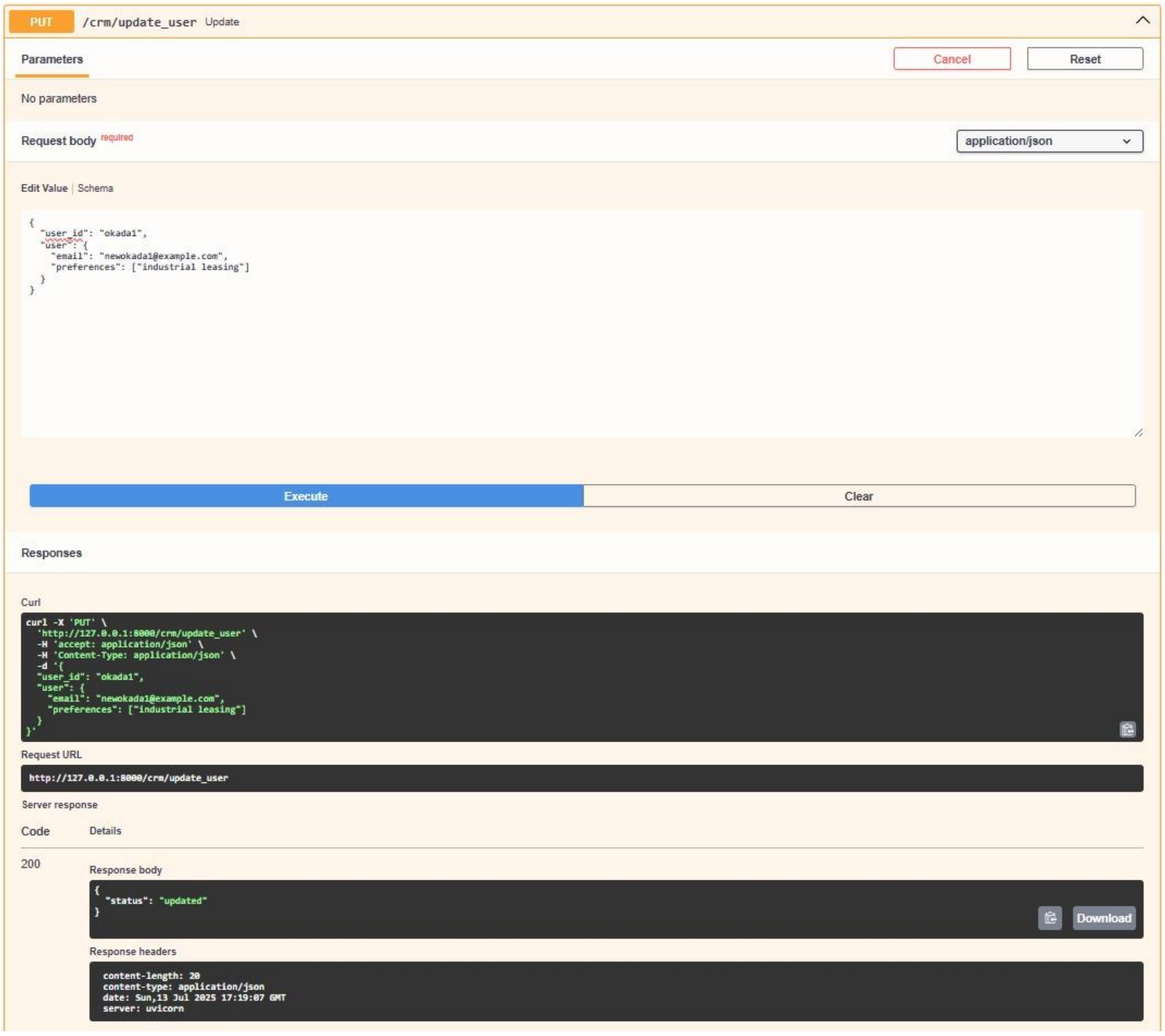
/crm/update_user

Updates existing CRM users and updates its details

Schema :

```
class UpdateUserRequest(BaseModel):
    user_id: str
    name: Optional[str]
    email: Optional[EmailStr]
    company: Optional[str]
    preferences: Optional[List[str]]


class UpdateUserResponse(BaseModel):
    message: str = Field(..., example="User updated successfully.")
```



```
{
  "user_id": "okada1",
  "user": {
    "email": "newokada1@example.com",
    "preferences": ["industrial leasing"]
  }
}
```

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/crm/update_user' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "user_id": "okada1",
  "user": {
    "email": "newokada1@example.com",
    "preferences": ["industrial leasing"]
  }
}'
```

Request URL

```
http://127.0.0.1:8000/crm/update_user
```

Server response

Code Details

200

Response body

```
{
  "status": "updated"
}
```

Response headers

```
content-length: 20
content-type: application/json
date: Sun,13 Jul 2025 17:19:07 GMT
server: uvicorn
```

.

Endpoint 4 : Fetch conversation history

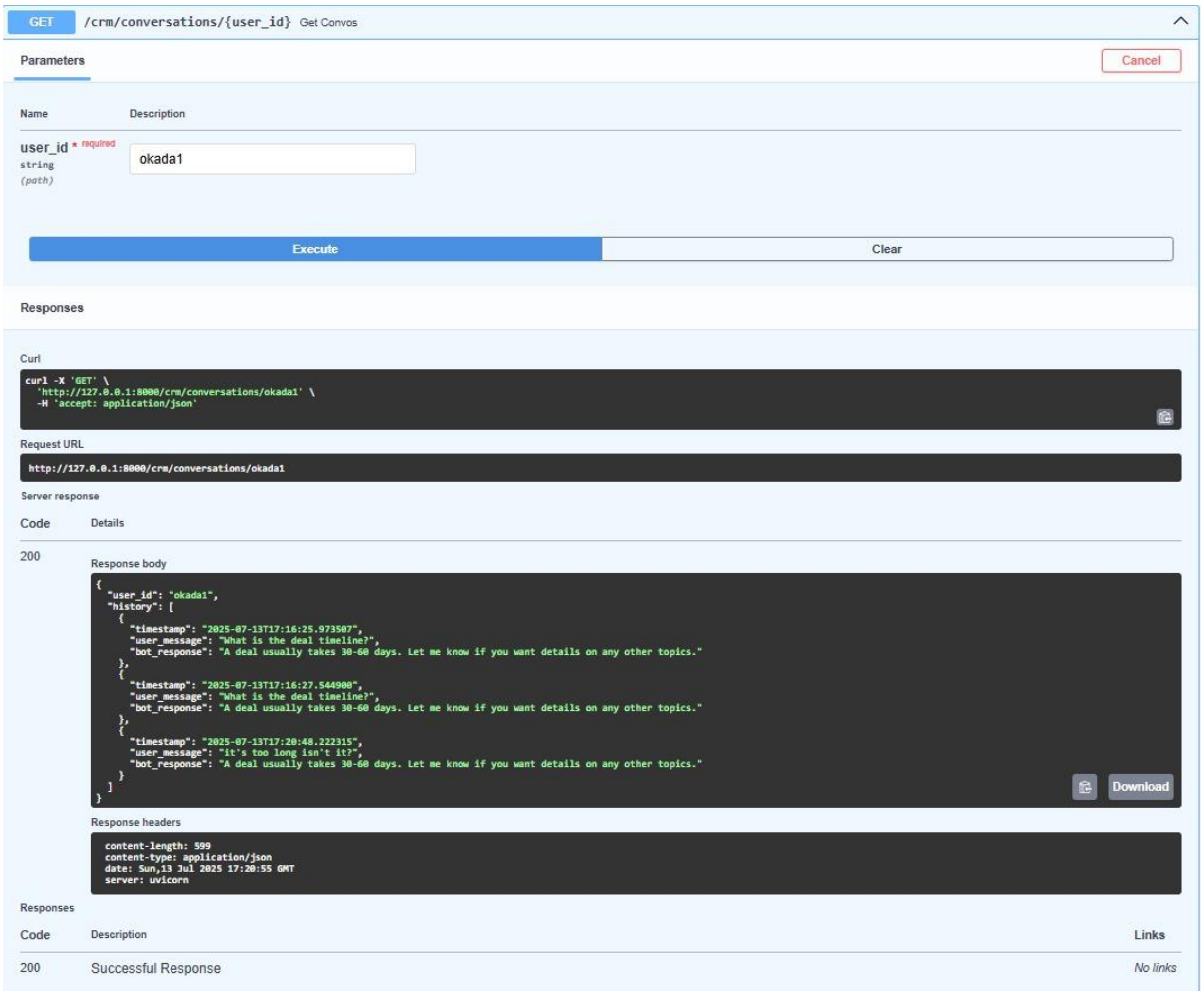
/crm/conversations/{user_id}

Using this end point we can fetch users entire chat history using user_id

Schema :

```
class Conversation(BaseModel):
    timestamp: datetime
    sender: str  # 'user' or 'assistant'
    message: str
    tags: Optional[List[str]] = []

class ConversationHistoryResponse(BaseModel):
    user_id: str
    conversation_history: List[Conversation]
```



Name | Description
user_id * required string (path) | okada1

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/crm/conversations/okada1' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/crm/conversations/okada1
```

Server response

Code | Details
200 | Response body

```
{
  "user_id": "okada1",
  "history": [
    {
      "timestamp": "2025-07-13T17:16:25.973507",
      "user_message": "What is the deal timeline?",
      "bot_response": "A deal usually takes 30-60 days. Let me know if you want details on any other topics."
    },
    {
      "timestamp": "2025-07-13T17:16:27.544900",
      "user_message": "What is the deal timeline?",
      "bot_response": "A deal usually takes 30-60 days. Let me know if you want details on any other topics."
    },
    {
      "timestamp": "2025-07-13T17:20:48.222315",
      "user_message": "it's too long isn't it?",
      "bot_response": "A deal usually takes 30-60 days. Let me know if you want details on any other topics."
    }
  ]
}
```

Response headers

```
content-length: 599
content-type: application/json
date: Sun,13 Jul 2025 17:20:55 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	No links

Endpoint 5 : Reset

/reset

This endpoint use to rest current conversation or remove conversation from history.

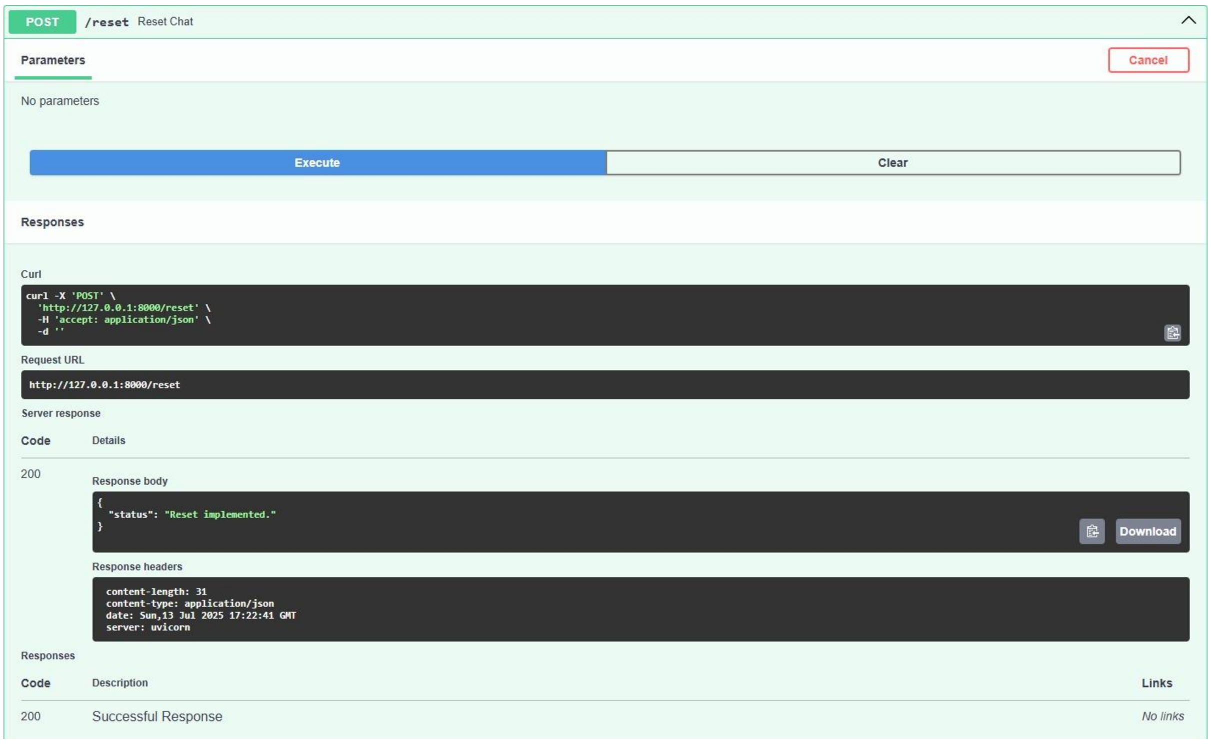
Schema :

class ResetRequest(BaseModel):

    user_id: str

class ResetResponse(BaseModel):

```
message: str = Field(..., example="Conversation memory reset.")
```



No parameters

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/reset' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://127.0.0.1:8000/reset
```

Server response

Code	Details
200	Response body

```
{
  "status": "Reset implemented."
}
```

Response headers

```
content-length: 31
content-type: application/json
date: Sun,13 Jul 2025 17:22:41 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*