API Contracts - Multi-Agent Conversational AI System

1. POST /chat/gemini

Input Format (application/json):

```
{
  "user_id": "string",
  "message": "What is the annual rent for 155 E 55th St?"
}
```

Response Schema:

```
{
  "user_id": "string",
  "guest_reference": "string | null",
  "conversation_id": "string | null",
  "response": "string",
  "used_context": "string",
  "tags": ["string"],
  "calendar_event_id": "string | null"
}
```

Usage Notes:

- Detects user intent and responds with either data insight or scheduling.
- Translates to/from English if needed.
- Handles greetings, unrelated queries, rent queries, and scheduling.

2. POST /upload_docs/

Form-data:

  file: (PDF, DOCX, TXT, CSV)

Response:

{
"message": "File uploaded and processed successfully"
}

Usage Notes:

- Extracts text, chunks it, embeds, and stores in FAISS index for RAG.

3. POST /calendar/create_event

Input Format:

{
"user_id": "string",
"title": "string",
"description": "string",
"datetime": "2025-07-13T06:34:19.667Z",
"conversation_id": "string"
}

Response:

200 OK or error message

Usage Notes:

- Used internally to log scheduling when detected from message.

4. GET /calendar/user_events/{user_id}

Returns all calendar events for a given user.

Sample Response:

[

```
  {
    "title": "Project meeting",
    "start_time": "...",
    "end_time": "...",
    "event_id": "..."
  }
]
```


5. PUT /conversation/{conversation_id}/update_tags

Input:


```
{
  "tags": ["string"],
  "status": "string"
}
```


Response:


```
{
  "message": "Conversation updated successfully"
}
```


Usage:

- Used to re-classify or mark status of past conversations.