Here's the updated API table with clear information about the parameters in the **body** and their **data types**, tailored for your **React.js developer**:

API Reference Table

Endpoint	Method	Request Parameters	Response Example	Description
`/user/register/`	POST	**Body:** `username` (string) - User's username `email` (string) - User's email `password` (string) - User's password	`{ "message": "User registered successfully", "user": { "username": "testuser", "email": "testuser@example.com", "password": "password123" } }`	Registers a new user.
`/user/login/`	POST	**Body:** `username` (string) - User's username `password` (string) - User's password	`{ "message": "Login successful" }` or `{ "detail": "Invalid username or password" }`	Logs in a user.
`/user/logout/`	POST	None	`{ "message": "Logout successful" }`	Logs out the currently authenticated user.
`/api/upload/`	POST	**Body:** `file_name` (string) - Name of the file `content` (string) - Content of the file	`{ "message": "File uploaded successfully", "file_name": "example.txt" }`	Simulates a file upload operation.

Endpoint	Method	Request Parameters	Response Example	Description
`/api/prompt/`	POST	**Body:** `prompt` (string) - The user's input prompt	`{ "response": "This is a static response", "conversation": { "messages": [{ "role": "user", "text": "..." }, { "role": "bot", "text": "..." }] } }`	Sends a prompt to the chatbot and gets a static response.
`/api/chat-history/get/`	GET	**Query Parameters:** `chat_id` (string) - ID of the chat	`{ "chat_id": "123e4567-e89b-12d3-a456-426614174000", "chat_name": "Health Chat", "conversation": { "messages": [...] } }` or `{ "detail": "Chat history not found" }`	Retrieves details of a specific chat by `chat_id`.
`/api/chat-history/create/`	POST	**Body:** `chat_name` (string) - Name of the chat `conversation` (object) - Conversation history data	`{ "message": "Chat history created", "chat_name": "New Chat" }`	Creates a new chat history record.
`/api/chat-history/update/`	PUT	**Body:** `chat_id` (string) - ID of the chat to update `chat_name` (string) - New name for the chat	`{ "message": "Chat history updated", "chat_id": "123e4567-e89b-12d3-a456-426614174000", "chat_name": "Updated Chat" }` or `{ "detail": "Chat history not found" }`	Updates the name of a chat by `chat_id`.
`/api/chat-history/delete/`	DELETE	**Body:** `chat_id` (string) - ID of the chat to delete	`{ "message": "Chat history deleted", "chat_id": "123e4567-e89b-12d3-a456-426614174000" }` or `{ "detail": "Chat history not found" }`	Deletes a chat history record by `chat_id`.

Endpoint	Method	Request Parameters	Response Example	Description
`/api/chat-history/list/`	GET	None	`[ { "chat_id": "123e4567-e89b-12d3-a456-426614174000", "chat_name": "Health Chat" }, { "chat_id": "123e4567-e89b-12d3-a456-426614174001", "chat_name": "Fitness Chat" } ]`	Retrieves a list of all chat IDs and their corresponding names.

Frontend Notes

1. **Data Types**:
   - Strings should be passed as plain text.
   - Objects (e.g., `conversation`) should be JSON-encoded.
2. **File Upload**:
   - Pass `file_name` and `content` as strings in the body.
3. **Chat Management**:
   - Use `GET /api/chat-history/list/` to populate the chat list.
   - Use `POST /api/chat-history/create/` to create new chats.
   - Use `PUT /api/chat-history/update/` to rename chats.
   - Use `DELETE /api/chat-history/delete/` to remove chats.
4. **Example Payloads**:
   - **Create Chat**:

```
{
  "chat_name": "New Chat",
  "conversation": {}
}
```

   - **Update Chat**:

```
{
  "chat_id": "123e4567-e89b-12d3-a456-426614174000",
  "chat_name": "Updated Chat"
}
```

   - **Delete Chat**:

```
{
  "chat_id": "123e4567-e89b-12d3-a456-426614174000"
}
```

This table provides a detailed explanation of the parameters and their data types to ensure your frontend developer has all the necessary information for integration. Let me know if you need further clarification or examples! 🚀