**API Endpoints Documentation**

Table of Contents

[1. Overview & Auth 2](#_Toc195127988)

[2. Endpoints 2](#_Toc195127989)

[2.1. Contact Form 2](#_Toc195127990)

[2.2. Chatbot 3](#_Toc195127991)

[2.3. Dataset Requests 3](#_Toc195127992)

[2.3.1. POST /requests/make 3](#_Toc195127993)

[2.3.2. GET /requests/view 4](#_Toc195127994)

[2.3.3. GET /requests/view/pending 5](#_Toc195127995)

[2.3.4. GET /requests/action/<id> 5](#_Toc195127996)

[2.3.5. POST /requests/action/<id> 5](#_Toc195127997)

[2.3.6. GET /user\_requests 6](#_Toc195127998)

[2.4. GET /chat/list – (In code: UserConversationsView) 6](#_Toc195127999)

[2.4.2. GET /chat/messages/<conversation\_id> 7](#_Toc195128000)

[**2.4.3. WebSocket Connection for Sending Messages** 8](#_Toc195128001)

[3. Common Status Codes 8](#_Toc195128002)

# 1. Overview & Auth

* **Authentication**: Some endpoints require a specific user role, such as researcher, organization\_admin, contributor, or no specific role.
* **Roles**:
  + **researcher**: can request datasets and submit research findings.
  + **organization\_admin**: can view all requests for their organisation, approve/deny them, manage datasets, and invite contributors.
  + **contributor**: is an employee of the organisation and can upload datasets and process requests.
  + **none (AllowAny)**: endpoints open to the public.
* **Responses**: Generally return JSON, unless noted.

# 2. Endpoints

### 2.1. Contact Form

**POST /send\_contact\_email**

* **Access**: AllowAny
* **Purpose**: Receives contact form input and sends an email to the site’s admin.
* **Request Body**:

{

"email": "string, required",

"subject": "string, required",

"message": "string, required"

}

* **Successful Response** (200):

{

"success": "Email sent successfully"

}

* **Error Cases**:
  + **400** if any field is missing:

{"error" : "All fields are required"}

* + **500** if sending the email fails.

### 2.2. Chatbot

**POST /chat**

* **Access**: AllowAny
* **Purpose**: Receives user’s question and returns a chatbot’s textual response plus optional action links.
* **Request Body**:

{

"message": "string, can be empty or contain user’s text"

}

* **Successful Response** (200) example:

{

"text": "Hello! I’m AlacrityBot...",

"actions": [

{"label": "I’m an Organization", "href": "organization"},

{"label": "I’m a Researcher", "href": "researcher"}

]

}

* **Error Cases**: Usually 200 unless there’s a server error.

### 2.3. Dataset Requests

Researchers request access to datasets owned by organisations. Organisation admins or contributors can view and action these requests.

### 2.3.1. POST /requests/make

* **Role**: researcher
* **Purpose**: Creates a new dataset access request.
* **Request Body**:

{

"dataset\_id": "string, required",

"objective": "string, required"

}

* **Response**:
  + **201** on success:

{"message": "Request created successfully"}

* **Error Cases**:
  + **400** if required fields are missing or a pending request for that dataset already exists.
  + **404** if dataset doesn’t exist.

### 2.3.2. GET /requests/view

* **Role**: organization\_admin, contributor
* **Purpose**: Lists all requests relevant to the user’s organisation.
* **Response** (200) example:

[

{

"request\_id": "string",

"dataset\_id": "<uuid>",

"dataset\_id\_\_title": "string",

"researcher\_id": 2,

"request\_status": "pending|approved|denied|revoked",

"created\_at": "timestamp",

"updated\_at": "timestamp",

},

]

* **Error Cases**:
  + **500** on unhandled server errors.

### 2.3.3. GET /requests/view/pending

* **Role**: organization\_admin, contributor
* **Purpose**: Same as /requests/view but filtered for request\_status. This defaults to pending, but can be changed via a query param request\_status.
* **Response**: same structure as /requests/view.

### 2.3.4. GET /requests/action/<id>

* **Role**: organization\_admin, contributor
* **Purpose**: Retrieves details of one specific request by request\_id.
* **Response** (200) example:

{

"request\_id": 55,

"dataset\_id": "<uuid>",

"researcher\_id": 10,

"request\_status": "pending",

}

* **Error Cases**:
  + **404** if the request doesn’t exist.

### 2.3.5. POST /requests/action/<id>

* **Role**: organization\_admin, contributor
* **Purpose**: Accept, reject, or revoke a request.
* **Request Body**:

{

"action": "accept" | "reject" | "revoke"

}

* **Successful Response** (200):

{"message": "Request <action>ed successfully"}

* **Error Cases**:
  + **400** if invalid action or missing parameters.
  + **404** if request not found.

### 2.3.6. GET /user\_requests

* **Role**: researcher, contributor
* **Purpose**: Lists all dataset requests made by the authenticated user, including payment status.
* **Response** (200) example:

[

{

"request\_id": 999,

"dataset\_id\_id": "<uuid>",

"dataset\_id\_\_title": "My Dataset",

"request\_status": "approved|pending|...",

"has\_paid": false,

"dataset\_id\_\_price": 9.99,

"created\_at": ".04/04/25",

"updated\_at": "14/03/12"

},

]

* **Error Cases**:
  + 500 on server failure.

### 2.4. GET /chat/list – (In code: UserConversationsView)

* **Role**: organization\_admin, contributor, researcher
* **Purpose**: Lists all conversations in which the user is a participant. The user is either participant1 or participant2.
* **Endpoint** GET /users/conversations or similar.
* **Response** example:

[

{

"conversation\_id": 12,

"participant": {

"id": 5,

"first\_name": "John",

"last\_name": "Smith",

"profile\_picture": "http://..."

},

"last\_message": "This is a complaint",

"last\_timestamp": "…",

"unread\_count": 2

},

]

* **Error Cases**:
  + **403** if the role check fails.
  + **500** on server errors.

### 2.4.2. GET /chat/messages/<conversation\_id>

* **Role**: organization\_admin, contributor, researcher
* **Purpose**: Lists all messages in a particular conversation.
* **Response** example:

[

{

"message\_id": 1,

"sender\_id": 5,

"message": "Hello",

"timestamp": "…",

"sender\_first\_name": "Sarah",

"sender\_last\_name": "Lala",

"sender\_profile\_picture": "http://..."

},

]

* **Error Cases**:
  + **403** if user isn’t in that conversation.
  + **404** if conversation doesn’t exist.

### **2.4.3. WebSocket Connection for Sending Messages**

* In the code the ChatConsumer class uses a URL ws://.../ws/chat/<conversation\_id>?token=<JWT>.
* **Purpose**: Allows real-time message sending, receiving and typing indicators.
* **Example**:
  + Clients connect via WebSocket to ws://<host>/ws/chat/<conversation\_id>.
  + They can send a JSON message: {"message": "Hi there!"} to post.
  + On success, the updated message is broadcast to all participants in that conversation.

# 3. Additional Key Endpoints

In addition to the above, the codebase includes:

### User Auth & Profile

* + **POST /users/register** – sign up as researcher or organisation.
  + **POST /users/login** – login.
  + **POST /users/forgot\_password** / **POST /users/reset\_password** – password recovery.
  + **POST /users/change\_password** – change password when logged in.
  + **GET /users/logged\_in** – fetch details of the currently logged-in user.

### Organization

* + **POST /organisation/register** – register an organisation with an org\_admin user.
  + **POST /organisation/contributors/add** – invite a new contributor.
  + **GET/PUT /organisation/profile/<org\_id>** – read/update organisation data, including MinIO for images.
  + **POST /organisation/follow/<org\_id>** and **POST /organisation/unfollow/<org\_id>** – follow/unfollow organisations.

### Payments

* + **POST /payments/paypal/create/<dataset\_id>** – create a PayPal payment for a paid dataset.
  + **GET /payments/paypal/success** – PayPal success callback. Marks dataset as purchased.
  + **GET /payments/paypal/cancel** – PayPal cancel callback.
  + **GET /payments/history** – user’s purchase history.

### Research Submissions

* + **POST /research/saveSubmission** – create/update a draft or submit for approval.
  + **GET /research/pending** – organisation admins see submissions awaiting approval.
  + **POST /research/approveRejectSubmission/<id>** – organisation admin accepts or rejects.
  + **GET /research/viewSubmissions** – all published submissions which is available for the public.
  + **DELETE /research/deleteSubmission/<id>** – soft deletes submissions.
  + **POST /research/bookmark/<submission\_id>** – users can bookmark submissions, this is also counted so it acts as a like feature too.

### Notifications

* + **GET /notifications/count** – unread notification count.
  + **GET /notifications/list** – all user notifications.
  + **PATCH /notifications/markAllRead** – marks all as read.
  + **DELETE /notifications/<notif\_id>** – deletes single notification.

# 4. Common Status Codes

* **200 OK**: Request succeeded; response includes the relevant data.
* **201 Created**: Resource successfully created.
* **400 Bad Request**: Missing or invalid input.
* **403 Forbidden**: User is authenticated but not allowed due to insufficient permissions.
* **404 Not Found**: Resource not found.
* **500 Internal Server Error**: Unhandled exceptions, email failures, or other server-side errors.