ThinkSync API Documentation

Base URL

`/api`

Authentication

All endpoints require a valid Microsoft Entra ID access token:

```
Authorization: Bearer <token>
```

Invalid or missing token:

```
Status: 401 Unauthorized
{
  "error": "Invalid token"
}
```

1 Endpoints

1.1 Create a Project

POST `/api/projects/create`

Creates a new project with its requirements.

Request Body

```
{
  "project": {
    "title": "AI for Healthcare",
    "description": "Exploring AI applications",
    "goals": "Improve diagnosis",
    "research_areas": "AI, Medicine",
    "start_date": "2025-05-01",
    "end_date": "2025-12-01",
    "funding_available": true
  },
  "requirements": [
    {
      "skill_required": "Python",
      "experience_level": "Intermediate",
      "role": "Data Scientist",
      "technical_requirements": "TensorFlow, Pandas"
    }
  ]
}
```

Success Response

```
Status: 201 Created
{
  "message": "Project created successfully",
  "project_ID": 42
}
```

Errors

- **400** - Missing or invalid data
- **401** - Invalid token
- **500** - Server error

1.2 Get Projects by Owner

GET `/api/projects/owner/:ownerId`
Returns all projects for the authenticated owner.

Success Response

```
Status: 200 OK
{
  "projects": [
    {
      "project_ID": 42,
      "owner_ID": "user-guid",
      "title": "AI for Healthcare",
      "description": "...",
      "goals": "...",
      "research_areas": "...",
      "start_date": "2025-05-01",
      "end_date": "2025-12-01",
      "funding_available": true,
      "created_at": "...",
      "review_status": "pending",
      "requirements": [
        {
          "requirement_ID": 7,
          "skill_required": "Python",
          "experience_level": "intermediate",
          "role": "Data Scientist",
          "technical_requirements": "TensorFlow, Pandas"
        }
      ]
    }
  ]
}
```

Errors

- **401** - Invalid token
- **403** - Unauthorized (wrong owner)

- **404** - No projects found
- **500** - Server error

1.3 Update a Project

PUT `/api/projects/update/:projectId`
Updates a project and replaces its requirements.

Request Body

Same as Create Project

Success Response

```
Status: 200 OK
{
  "message": "Project updated successfully"
}
```

Errors

- **400** - Invalid input
- **401** - Invalid token
- **403** - Unauthorized
- **404** - Project not found
- **500** - Server error

1.4 Delete a Project

DELETE `/api/projects/delete/:projectId`
Deletes a project and its requirements.

Success Response

```
Status: 200 OK
{
  "message": "Project deleted successfully"
}
```

Errors

- **401** - Invalid token
- **403** - Unauthorized
- **404** - Project not found
- **500** - Server error

1.5 Validation Rules

Project

- `title` (string, required, max 255)
- `description`, `goals`, `research_areas` (string, required)
- `start_date`, `end_date` (format YYYY-MM-DD)
- `funding_available` (boolean)

Note: `start_date` must be before `end_date`

Requirements

Each requirement object must contain:

- `skill_required` (string, max 255)
- `experience_level` (string: beginner, intermediate, professional)
- `role` (string, max 100)
- `technical_requirements` (string)

Notes

- Dates must be in `YYYY-MM-DD` format.
- All errors return a JSON object with an `error` key.
- Token must be valid and scoped for Microsoft Graph '/me'.

2 Collaboration Endpoints

2.1 Search for Potential Collaborators

POST `/api/collaborators/search`
Searches users based on name, skill, or position.

Headers

- `Authorization: Bearer <token>`

Request Body

```
{
  "searchTerm": "AI",
  "searchType": "skill"
}
```

Response (200 OK)

```
{
  "collaborators": [
    {
      "user_ID": 12,
      "fname": "Alice",
      "sname": "Wong",
      "department": "Computer Science",
      "acc_role": "Researcher",
      "res_area": "AI",
      "qualification": "PhD"
    }
  ]
}
```

Possible Errors

Status	Message
400	Search term and type are required
401	Access token is required
500	Internal server error

2.2 Send Invitation

POST `/api/collaborators/invite`

Send a collaboration invitation to another researcher.

Request Body

```
{
  "project_ID": 42,
  "recipient_ID": 17,
  "proposed_role": "ML Engineer"
}
```

Response (201 Created)

```
{
  "message": "Invitation sent successfully"
}
```

Possible Errors

Status	Message
400	Missing required fields or duplicate invitation
401	Access token is required
403	Only project owner can send invitations
404	Project not found
500	Internal server error

2.3 Get Received Invitations

GET `/api/collaborators/invitations/received`

Returns a list of collaboration invitations received by the user.

Response (200 OK)

```
{
  "invitations": [
    {
      "invitation_ID": 1,
      "project_title": "AI for Cancer",
      "sender_fname": "John",
      "sender_sname": "Smith",
      "status": "pending",
      "current_status": "expired"
    }
  ]
}
```

2.4 Get Sent Invitations

GET `/api/collaborators/invitations/sent`

Returns a list of collaboration invitations sent by the user.

Response (200 OK)

```
{
  "invitations": [
    {
      "invitation_ID": 2,
      "recipient_fname": "Jane",
      "recipient_sname": "Doe",
      "project_title": "AI for Healthcare",
      "current_status": "pending"
    }
  ]
}
```

2.5 Update Invitation Status

PUT `/api/collaborators/invitation/:invitationId`

Updates the status of a specific invitation (accept, decline, cancel).

Request Body

```
{
  "status": "accepted"
}
```

Response (200 OK)

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

Possible Errors

Status	Message
400	Invalid status or non-pending invitation
401	Access token is required
403	Unauthorized action
404	Invitation not found
500	Internal server error

3 Microsoft Login and Role Registration Endpoints

3.1 Microsoft Login

POST `/api/auth/microsoft`

Authenticates a user via Microsoft OAuth and registers the user if they don't already exist.

Request Body

```
{
  "token": "eyJ0eXAi..."
}
```

Responses

- **201 Created**

```
{
  "message": "User registered successfully"
}
```

- **200 OK**

```
{
  "message": "User authenticated successfully"
}
```

- **400 Bad Request**

```
{
  "error": "Incomplete user data from Microsoft"
}
```

- **400 Bad Request**

```
{
  "error": "Please sign up using a University of Witwatersrand
     ↪ email domain"
}
```

- **401 Unauthorized**

```
{
  "error": "Access token missing"
}
```

3.2 Researcher Registration

POST `/api/auth/researcher`

Registers an authenticated user as a researcher and links their research details.

Request Body

```
{
  "token": "eyJ0eXAi...",
  "phone_number": "0123456789",
  "department": "Computer Science",
  "acc_role": "Researcher",
  "res_area": "AI",
  "qualification": "PhD",
  "current_proj": "AI in Healthcare"
}
```

Responses

- **201 Created**

```
{
  "message": "All input successful"
}
```

- **400 Bad Request**

```
{
  "error": "User already signed up as researcher"
}
```

- **400 Bad Request**

```
{
  "error": "Missing or invalid input fields for researcher"
}
```

- **500 Server Error**

```
{
  "error": "Server error"
}
```

3.3 Reviewer Registration

POST `/api/auth/reviewer`

Registers an authenticated user as a reviewer with their research details.

Request Body

Same as researcher endpoint.

Responses

Same as researcher endpoint, with "reviewer" replacing "researcher" in messages and errors.

3.4 Admin Registration

POST `/api/auth/admin`

Registers a user as an admin.

Request Body

```
{
  "token": "eyJ0eXAi...",
  "phone_number": "0123456789",
  "department": "Admin Dept",
  "acc_role": "Admin"
}
```

Responses

- **201 Created**

```
{
  "message": "All input successful"
}
```

- **400 Bad Request**

```
{
  "error": "User already enrolled as admin"
}
```

- **400 Bad Request**

```
{
  "error": "Missing or invalid input fields for admin"
}
```

- **500 Server Error**

```
{
  "error": "Server error"
}
```

4 Admin dashboard end points

Authentication

All endpoints require an Authorization header with a valid Bearer token.

4.1 GET /admin/users

Description: Fetch all users along with their roles.
Authorization: Admin only.

Response Example:


```
{
  "users": [
    {
      "user_ID": 1,
      "fname": "John",
      "sname": "Doe",
      "phone_number": "1234567890",
      "department": "Science",
      "acc_role": "researcher",
      "roles": "researcher"
    },
    {
      "user_ID": 2,
      "fname": "Jane",
      "sname": "Smith",
      "phone_number": "0987654321",
      "department": "Engineering",
      "acc_role": "reviewer",
      "roles": "reviewer, admin"
    }
  ]
}
```


4.2 PUT /admin/users/:userId/role

Description: Update the role of a user.
Authorization: Admin only.

Request Example:


```
{
  "newRole": "reviewer"
}
```


Successful Response Example:


```
{
  "message": "User role updated successfully"
}
```


4.3 GET /admin/projects/pending

Description: Fetch pending projects without any assigned reviewer.
Authorization: Admin only.

Response Example:


```
{
  "projects": [
    {
      "project_ID": 1,
      "project_title": "AI Research",
      "researcher_fname": "John",
      "researcher_sname": "Doe"
```

```
    }
  ]
}
```

4.4 GET /admin/reviewers/search?research_area=AI

Description: Search for reviewers by research area.
Authorization: Admin only.
Response Example:

```
{
  "reviewers": [
    {
      "user_ID": 5,
      "fname": "Alice",
      "sname": "Brown",
      "res_area": "AI",
      "qualification": "PhD",
      "current_proj": "Machine Learning Models"
    }
  ]
}
```

4.5 POST /admin/projects/:projectId/assign-reviewer

Description: Assign a reviewer to a project.
Authorization: Admin only.
Request Example:

```
{
  "reviewerId": 5
}
```

Successful Response Example:

```
{
  "message": "Reviewer assigned successfully"
}
```

5 Reviewer API Endpoints

5.1 Middleware: `isReviewer`

Description: This middleware verifies that the authenticated user holds the `reviewer` role before proceeding. If authentication fails, or the user does not have the reviewer role, a `403 Unauthorized` or `401 Authentication Failed` error is returned.

5.2 GET /reviewer/proposals

Description: Retrieves all proposals assigned to the authenticated reviewer.
Request:

- Method: `GET`
- Headers: Must include a valid Authorization token.

Sample Response:


```
{
  "proposals": [
    {
      "project_ID": 1,
      "title": "AI in Healthcare",
      "description": "Using AI for diagnostics.",
      "goals": "Improve diagnostic accuracy.",
      "start_date": "2025-05-01",
      "skill_required": "Machine Learning",
      "experience_level": "Intermediate",
      "requirement_role": "Developer",
      "technical_requirements": "TensorFlow, Python",
      "researcher_ID": 12,
      "researcher_fname": "Alice",
      "researcher_sname": "Smith",
      "assignment_ID": 1001,
      "Assigned_at": "2025-04-20 10:15:00"
    }
  ]
}
```


Errors:

- `401 Authentication failed` if token is invalid.
- `500 Failed to fetch proposals` on server error.

5.3 POST /reviewer/proposals/:projectId/review

Description: Submits a review for an assigned proposal.

Request:

- Method: `POST`
- Headers: Must include a valid Authorization token.
- URL parameter: `projectId` (ID of the proposal to review)
- Body Parameters:
  - `comments` (string): Review comments.
  - `recommendation` (string): One of `approved`, `rejected`, or `revised`.

Sample Input:


```
{
  "comments": "The project is well structured and feasible.",
  "recommendation": "approved"
}
```


Sample Response:


```
{
  "message": "Review submitted successfully"
}
```


Errors:

- `400 Missing required fields` if `comments` or `recommendation` are missing.

- `400 Invalid recommendation` if recommendation is not one of the allowed values.
- `403 Project not assigned to this reviewer` if the reviewer is not assigned to the project.
- `500 Failed to submit review` on server error.

6 Messaging API

6.1 GET /messages

Description: Get all messages for the authenticated user.
Authentication: Required (Bearer Token)


```
[
  {
    "message_ID": 1,
    "sender_ID": 2,
    "receiver_ID": 3,
    "subject": "Project Update",
    "body": "Here's the latest update...",
    "sent_at": "2025-05-01T12:34:56.000Z",
    "is_read": false,
    "project_ID": 5,
    "project_title": "COVID Research",
    "sender_sname": "Doe",
    "sender_fname": "John",
    "receiver_sname": "Smith",
    "receiver_fname": "Anna",
    "attachments": [
      {
        "attachment_ID": 10,
        "file_name": "report.pdf",
        "file_url": "https://..."
      }
    ]
  }
]
```


6.2 GET /messages/unread

Description: Get all unread messages for the authenticated user.
Authentication: Required


```
[
  {
    "message_ID": 4,
    "sender_ID": 2,
    "subject": "Reminder",
    "sent_at": "2025-05-03T09:15:00.000Z",
    "sender_sname": "Lee",
    "sender_fname": "Grace",
    "attachment_count": 2
  }
]
```

6.3 POST /messages

Description: Send a new message with optional attachments.
Authentication: Required
Request Type: `multipart/form-data`
Fields:

- `receiver`$_ID$(*string*)`subject`(*string*)
- `body` (string)
- `project`$_ID$(*optional*, *stringornull*)`attachments`(*upto*5*files*, *optional*)

```
{
  "message": "Message sent successfully",
  "messageId": 123
}
```

6.4 POST /messages/:messageId/attachments

Description: Upload an attachment to an existing message.
Authentication: Required
Request Type: `multipart/form-data` with one `file` field.

```
{
  "url": "https://..."
}
```

6.5 GET /messages/:messageId/attachments/:attachmentId

Description: Download a message attachment.
Authentication: Required
Response: File download stream

6.6 GET /messages/search-users?query=...

Description: Search for users by name.
Authentication: Required

```
[
  {
    "user_ID": 4,
    "sname": "Nguyen",
    "fname": "Kim"
  }
]
```

6.7 GET /messages/search-projects?query=...

Description: Search for projects by title.
Authentication: Required

```
[
  {
    "project_ID": 5,
    "title": "Genomics Study",
    "owner_fname": "Elena",
```

```
    "owner_sname": "Rodriguez"
  }
]
```

6.8 PUT /messages/mark-read

Description: Mark all messages from a given sender as read.
Authentication: Required

Request Body:

```
{
  "senderId": 2
}
```

```
{
  "message": "Messages marked as read"
}
```

Milestone Management API

Authorization

All endpoints require a valid Bearer token. Only users with the `researcher` role may access these routes.

GET /milestones

Description: Retrieve all projects owned by the authenticated researcher, including each project's milestones and collaborators. Also includes a milestone status summary.

Response Example:

```
{
  "projects": [
    {
      "project_ID": 1,
      "title": "AI Research",
      "owner_ID": "abc123",
      "milestones": [
        {
          "milestone_ID": 10,
          "title": "Initial Literature Review",
          "status": "In Progress",
          "expected_completion_date": "2025-06-01",
          "assigned_user_fname": "John",
          "assigned_user_sname": "Doe"
        }
      ],
      "collaborators": [
        {
          "user_ID": "abc123",
          "first_name": "Alice",
          "last_name": "Nguyen",
          "is_owner": true
        },
        {
```

```
          "user_ID": "xyz456",
          "first_name": "John",
          "last_name": "Doe",
          "is_owner": false
        }
      ]
    }
  ],
  "summary": [
    {
      "status": "Completed",
      "count": 2,
      "percentage": 40.0
    }
  ]
}
```


POST /milestones/:projectId

Description: Create a milestone for a given project.

Request Body:


```
{
  "title": "Write Paper Draft",
  "description": "First version of conference paper",
  "expected_completion_date": "2025-09-01",
  "assigned_user_ID": "xyz456",
  "status": "Not Started"
}
```


Success Response:


```
{
  "message": "Milestone created",
  "milestone_ID": 13
}
```


Validation Rules:

- `title` (required, non-empty string)
- `description` (optional string or null)
- `expected_completion_date` (optional, `YYYY-MM-DD`)
- `assigned_user_ID` (optional; must be owner or collaborator)
- `status` (optional; one of `"Not Started"`, `"In Progress"`, `"Completed"`)

GET /milestones/:milestoneId

Description: Retrieve details of a specific milestone and its potential collaborators.

Response Example:


```
{
  "milestone": {
    "milestone_ID": 10,
    "title": "Initial Literature Review",
    "status": "In Progress",
```

```
    "expected_completion_date": "2025-06-01",
    "project_title": "AI Research",
    "assigned_user_fname": "John",
    "assigned_user_sname": "Doe"
  },
  "collaborators": [
    {
      "user_ID": "abc123",
      "name": "Alice Nguyen"
    },
    {
      "user_ID": "xyz456",
      "name": "John Doe"
    }
  ]
}
```

PUT /milestones/:projectId/:milestoneId

Description: Update a milestone's information.

Request Body:

```
{
  "title": "Revise Literature Review",
  "description": "Add recent papers",
  "expected_completion_date": "2025-07-01",
  "assigned_user_ID": "xyz456",
  "status": "In Progress"
}
```

Success Response:

```
{
  "message": "Milestone updated"
}
```

DELETE /milestones/:projectId/:milestoneId

Description: Delete a milestone by ID.

Success Response:

```
{
  "message": "Milestone deleted"
}
```

GET /milestones/report/generate

Description: Generate a downloadable PDF report of all owned projects and their milestones. Includes pie charts and tables.

Response: PDF file with:

- Pie chart of milestone status distribution.
- List of projects with tables of collaborators and milestones.

Errors:

- `403 Unauthorized` if user is not a researcher.
- `400 Bad Request` on validation errors.
- `500 Server Error` on internal issues.

Funding Management API

Authentication

All routes require a valid Bearer token. Only users with the `researcher` role may access these endpoints.

POST /funding/:projectId

Description: Initialize funding details for a project.

Input (JSON):

```
{
  "total_awarded": 200000,
  "grant_status": "active",
  "grant_end_date": "2025-12-31"
}
```

Response:

```
{
  "message": "Funding initialized",
  "funding_ID": 42
}
```

Validation Rules:

- `total_awarded` (required): Non-negative number
- `grant_status` (optional): One of `active`, `completed`, `expired`, `cancelled`
- `grant_end_date` (optional): Date in `YYYY-MM-DD` format

—

GET /funding

Description: Retrieve a funding summary for all researcher-owned projects.

Response Example:

```
{
  "projects": [
    {
      "project_ID": 1,
      "title": "Cancer Research",
      "funding": {
        "total_awarded": 50000,
        "grant_status": "active",
        "grant_end_date": "2025-09-01",
        "amount_spent": 30000,
        "amount_remaining": 20000,
        "percentage": 60.0,
        "remainingPercentage": 40.0
```

```
      },
      "categories": [
        {
          "category_ID": 101,
          "category": "Equipment",
          "amount_spent": 15000,
          "percentage": 30.0
        }
      ],
      "funding_initialized": true
    }
  ]
}
```


—

PUT /funding/:projectId

Description: Update funding details for a project.

Input (JSON):


```
{
  "total_awarded": 300000,
  "grant_status": "completed",
  "grant_end_date": "2026-01-15"
}
```


Response:


```
{
  "message": "Funding updated"
}
```


—

DELETE /funding/:projectId

Description: Delete funding and all associated categories from a project.

Response:


```
{
  "message": "Funding deleted"
}
```


—

POST /funding/:projectId/categories

Description: Add a funding category to a project.

Input (JSON):


```
{
  "category": "Consumables",
  "description": "Chemicals and reagents",
  "amount_spent": 7500
}
```


Response:

```
{
  "message": "Funding category created",
  "category_ID": 56
}
```

Validation Rules:

- `category`: Required, non-empty string, max 255 chars
- `amount_spent`: Required, non-negative number

—

PUT /funding/:projectId/categories/:categoryId

Description: Update a specific funding category.

Input (JSON):

```
{
  "category": "Equipment",
  "description": "Upgraded lab microscope",
  "amount_spent": 10000
}
```

Response:

```
{
  "message": "Funding category updated"
}
```

—

DELETE /funding/:projectId/categories/:categoryId

Description: Delete a specific funding category.

Response:

```
{
  "message": "Funding category deleted"
}
```

—

GET /funding/:projectId/categories

Description: Fetch all categories associated with a project's funding.

Response Example:

```
{
  "categories": [
    {
      "category_ID": 101,
      "category": "Personnel",
      "description": "Research assistants",
      "amount_spent": 20000
    },
    {
      "category_ID": 102,
      "category": "Equipment",
```

```
      "description": "Lab kits",
      "amount_spent": 15000
    }
  ]
}
```

—

GET /funding/report

Description: Generates a downloadable PDF report of project funding, including pie charts and tables.

Response: PDF file containing:

- Project funding summaries
- Pie charts showing spending distribution
- Category breakdown tables

Dashboard Widgets API

Authentication

All endpoints require a valid Bearer token. Users can only access and manage their own widgets.

—

GET /widgets

Description: Retrieve all dashboard widgets for the authenticated user, sorted by position.

Response Example:

```
{
  "widgets": [
    {
      "widget_ID": 1,
      "user_ID": "abc123",
      "widget_type": "projects",
      "position_x": 0,
      "position_y": 1,
      "width": 2,
      "height": 2
    }
  ]
}
```

POST /widgets

Description: Add a new widget to the user's dashboard.

Request Example:

```
{
  "widget_type": "milestones",
  "position_x": 2,
  "position_y": 0,
  "width": 1,
  "height": 2
}
```

Valid widget types: `"projects"`, `"milestones"`, `"funding"`
Success Response:

```
{
  "message": "Widget added successfully",
  "widget_ID": 10
}
```

Errors:

- `400` – Invalid widget type
- `500` – Internal server error

—

PUT /widgets/:widgetId

Description: Update a widget's size and position.
Request Example:

```
{
  "position_x": 1,
  "position_y": 3,
  "width": 2,
  "height": 2
}
```

Success Response:

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

Errors:

- `404` – Widget not found (not owned by user or invalid ID)
- `500` – Internal server error

—

DELETE /widgets/:widgetId

Description: Delete a widget by its ID.
Success Response:

```
{
  "message": "Widget deleted successfully"
}
```

Errors:

- `404` – Widget not found
- `500` – Internal server error