Trackit

API Documentation

Technical Writer

Autumn Han

Table of Contents

1. Introduction

Trackit API

A lightweight, personal task management API for users who want to stay organized across daily routines, work tasks, pet care, etc. The API enables you to group tasks by projects, set deadlines, receive notifications, and view all your schedules in an intuitive calendar-based overview. The API documentation and resources are available at github.com/awe-someautumn/trackit-api-docs.

Base URL

`https://698a9017-e4ab-4ee7-82d7-f605859ad87f.mock.pstmn.io/v1`

Authentication

- Most endpoints require a Bearer token.
- Token is retrieved via `POST /register` or `POST /login`
- Pass the token using: `Authorization: Bearer trackitMockToken`

Endpoints

- Authentication
    - Register a new user
    - Log in and receive an access token
- User
    - Get current user info
    - Update current user info
    - Upload a profile image
- Projects
    - Get all projects
    - Create a new project
    - Update a project
    - Delete a project
- Tasks

    - Get tasks within a project
    - Create a task within a project
    - Get a single task by ID
    - Update a task by ID
    - Delete a task by ID
- Overview
    - Get an overview of all tasks
- Notifications
    - Get user notifications
    - Mark a notification as read

2. Authentication

`POST /register` – Register a New User

This endpoint allows new users to create an account by providing an email, password, and name.

- A nickname may optionally be provided for personalization within the app.
- No authentication is required for this endpoint.

Parameters

Name	Type	In	Required	Description
email	string	body	Y	User's email (must be unique)
password	string	body	Y	User's password
name	string	body	Y	Full name of the user
nickname	string	body	N	Optional nickname for display in the app

Example Request Body

```
{
  "email": "johndoe@example.com",
  "password": "password123",
  "name": "John Doe",
  "nickname": "WriterJohn"
}
```

Responses

Status Code	Description
200 OK	The user has been successfully created. A unique user ID is returned.
400 Bad Request	It indicates when required fields (email, password, or name) are missing from the request.
409 Conflict	It indicates when the provided email is already registered.

Example Response Body

```
{
  "code": 201,
  "message": "User registered successfully.",
  "userId": "user_12345"
}
```

`POST /login` - Log in

This endpoint allows users to log into the system using their email and password.

- On successful login, the API returns a Bearer token that must be used to authenticate subsequent requests.
- No authentication is required for this endpoint.

Parameters

Name	Type	In	Required	Description
email	string	body	Y	User’s email (must be unique)
password	string	body	Y	User’s password

Example Request Body

```
{
  "email": "johndoe@example.com",
  "password": "password123"
}
```

Responses

Status Code	Description
200 OK	The credentials are valid and log in successfully.
400 Bad Request	It indicates when required fields (email or password) are missing from the request.
401 Unauthorized	It indicates when the provided credentials do not match any user.

Example Response Body

```
{
  "code": 200,
  "message": "Login successful.",
  "token": "Bearer trackitMockToken"
}
```

3. User

`GET /me – Get User Information`

This endpoint retrieves the profile information of the currently authenticated user.

- It is commonly used after login to display user data such as name, email, and optional nickname.
- Bearer token is required.

Responses

Status Code	Description
200 OK	It returns when the token is valid and the user's information is successfully retrieved.
401 Unauthorized	It indicates when the request does not contain a valid Bearer token.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "User info retrieved successfully."
  },
  "data": {
    "userId": "user_12345",
    "email": "johndoe@example.com",
    "name": "John Doe",
    "nickname": "WriterJohn",
    "profileImageUrl": null
  }
}
```

`PATCH /me` – Update User Information

This endpoint allows authenticated users to update their own profile.

- The user can modify their name or nickname.
- The request must include at least one field to update.
- Bearer token is required.

Parameters

Name	Type	In	Required	Description
name	string	body	N	New full name of the user
nickname	string	body	N	New nickname for display in the app

Example Request Body

```
{
  "nickname": "SuperWriter"
}
```

Responses

Status Code	Description
200 OK	It returns when the user’s profile is successfully updated.
400 Bad Request	It indicates when required fields (name or nickname) are missing from the request.
401 Unauthorized	It indicates when the request does not contain a valid Bearer

	token.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "User info updated successfully."
  },
  "data": {
    "userId": "user_12345",
    "email": "johndoe@example.com",
    "name": "John Doe",
    "nickname": "AwesomeWriter",
    "profileImageUrl": null
  }
}
```

`POST /me/profile-image – Upload a Profile Image`

This endpoint allows users to upload a new profile image.

- The image must be sent as `multipart/form-data` with a single field named `image`.
- The uploaded image is associated with the current user's account.
- Bearer token is required.

Form Data Parameters

Name	Type	In	Required	Description
image	file	form	Y	Profile image to upload

Example Form-Data Upload

```
Key: image
Value: [Select Image File]
```

Responses

Status Code	Description
200 OK	The profile image is successfully uploaded and stored.

400 Bad Request	It indicates when the request field (image) is missing from the request.
401 Unauthorized	It indicates when the request does not contain a valid Bearer token.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Profile image uploaded successfully."
  },
  "data": {
    "profileImageUrl": "https://cdn.trackit.com/profiles/user_12345.jpg"
  }
}
```

4. Projects

`GET /projects - Get All Projects`

This endpoint retrieves a list of all projects belonging to the currently authenticated user.

- Each project includes metadata such as its title, color, deletability, and creation date.
- Bearer token is required.

Responses

Status Code	Description
200 OK	It returns all projects associated with the authenticated user.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Projects retrieved successfully."
  },
  "data": [
    {
      "projectId": "proj_001",
      "title": "Work",
      "color": "#FF9800",
      "isDeletable": false,
      "createdAt": "2025-05-17T09:00:00Z"
    },
    {
      "projectId": "proj_002",
      "title": "Daily Life",
      "color": "#4FC3F7",
      "isDeletable": true,
      "createdAt": "2025-05-17T10:00:00Z"
    }
  ]
}
```

POST /projects – Create a New Project

This endpoint allows authenticated users to create a new project.

- Each project must have a title and a color (HEX format).
- Projects can later be used to organize tasks.
- Bearer token is required.

Parameters

Name	Type	In	Required	Description
title	string	body	Y	The name of the project (must be unique)
color	string	body	Y	HEX color code (e.g., #FFFFFF, #000000)

Example Request Body

```
{
  "title": "Daily Life",
  "color": "#4FC3F7"
}
```

Responses

Status Code	Description

201 Created	The project was successfully created and assigned a unique project ID.
400 Bad Request	It indicates when required fields (title or color) are missing from the request.
401 Unauthorized	It indicates when the request does not contain a valid Bearer token.
409 Conflict	It indicates when the provided email is already registered.
422 Unprocessable Entity	It indicates when the color value is not in a valid HEX format.

Example Response Body

```
{
  "meta": {
    "code": 201,
    "message": "Project created successfully."
  },
  "data": {
    "projectId": "proj_002",
    "title": "Daily Life",
    "color": "#4FC3F7",
    "isDeletable": true,
    "createdAt": "2025-05-17T12:00:00Z"
  }
}
```

`PATCH /projects/:projectId – Update a Project`

This endpoint allows users to update the title, color, or deletable status of a specific project.

- The request must include at least one field to update.
- Only projects marked as `isDeletable: true` can have their deletable status changed.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
projectId	string	path	Y	ID of the project to update

Parameters (at least one is required)

Name	Type	In	Required	Description
title	string	body	N	New title for the project
color	string	body	N	HEX color code (e.g., #FFFFFF, #000000)
isDeletable	boolean	body	N	Whether the project is deletable

Example Request Body

```
{
  "title": "Home Routine",
  "color": "#FFB74D"
}
```

Responses

Status Code	Description
200 OK	The project was successfully updated with the provided fields.
400 Bad Request	It indicates when the request body is empty or contains invalid fields.
404 Not Found	It indicates when the specified project ID does not exist.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Project updated successfully."
  },
  "data": {
    "projectId": "proj_002",
    "title": "Home Routine",
    "color": "#FFB74D",
    "isDeletable": true,
    "createdAt": "2025-05-17T10:00:00Z"
  }
}
```

`DELETE /projects/:projectId` `- Delete a Project`

This endpoint allows users to delete a specific project by its ID.

- Only projects marked as `isDeletable: true` can be deleted.
  If `isDeletable: false`, the server will reject the deletion request.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
projectId	string	path	Y	ID of the project to delete

Responses

Status Code	Description
204 No Content	The project was successfully deleted.
400 Bad Request	It indicates when the request body is empty or contains invalid fields.
403 Forbidden	It indicates when attempting to delete a non-deletable project. (when `isDeletable: false`)
404 Not Found	It indicates when the specified project ID does not exist.

Example Response Body

```
{
  "code": 204,
  "message": "Project deleted successfully."
}
```

***Note**: The actual 204 response will return an empty body. This example is included for consistency in response format documentation.*

5. Tasks

`GET /projects/:projectId/tasks` `- Get Tasks in a Project`

This endpoint retrieves a list of all tasks belonging to a specific project.

- Each task includes details such as title, due date, status, priority, and creation date.
- Only tasks under the specified `projectId` will be returned.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
projectId	string	path	Y	ID of the project to retrieve tasks

Responses

Status Code	Description
200 OK	It returns all tasks under the given project ID.
400 Bad Request	It indicates when required fields are missing from the request.
401 Unauthorized	It indicates when the token is missing or invalid.
404 Not Found	It indicates when the specified project ID does not exist.
422 Unprocessable Entity	It indicates when `projectId` format does not meet the expected structure (format: proj_###).
500 Internal Server Error	Internal server error occurs.
503 Service Unavailable	Service is not available.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Tasks retrieved successfully."
  },
  "data": [
    {
      "taskId": "task_001",
      "title": "Write outline",
      "dueDate": "2025-05-20T08:00:00Z",
      "status": "not-started",
      "priority": "high",
      "projectId": "proj_001",
      "createdAt": "2025-04-17T09:00:00Z"
    },
    {
      "taskId": "task_002",
      "title": "Meet with publisher",
      "dueDate": "2025-05-21T10:30:00Z",
      "status": "in-progress",
      "priority": "medium",
      "projectId": "proj_001",
      "createdAt": "2025-04-17T09:10:00Z"
    }
  ]
}
```

POST /projects/:projectId/tasks – Create a Task

This endpoint allows users to create a new task under a specific project.

- Each task must include a title.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
projectId	string	path	Y	ID of the project

Parameters

Name	Type	In	Required	Description
title	string	body	Y	Title of the task
description	string	body	N	Detailed description of the task
dueDate	string	body	N	Due date in ISO 8601 format
status	string	body	N	Task status (e.g., `not-started`)
priority	string	body	N	Task priority (`low`, `medium`, `high`)

Example Request Body

```
{
  "title": "Buy groceries",
  "description": "Pick up eggs, milk, and fruit",
  "dueDate": "2025-05-30T10:00:00Z",
  "status": "not-started",
  "priority": "medium"
}
```

Responses

Status Code	Description
201 Created	The task was successfully created in the given project.
400 Bad Request	It indicates when required fields (title) are missing from the request.
404 Not Found	It indicates when the specified project ID does not exist.
422 Unprocessable Entity	It indicates when an enum-type field (status or priority) contains an invalid value.
500 Internal Server Error	Internal server error occurs.
503 Service Unavailable	Service is not available.

Example Response Body

```
{
  "meta": {
    "code": 201,
    "message": "Task created successfully."
  },
  "data": {
    "taskId": "task_021",
    "title": "Buy groceries",
    "description": "Pick up eggs, milk, and fruit",
    "dueDate": "2025-05-30T10:00:00Z",
    "status": "not-started",
    "priority": "medium",
    "projectId": "proj_002",
    "createdAt": "2025-05-28T09:00:00Z"
  }
}
```

`GET /tasks/:taskId` - Get a Task by ID

This endpoint retrieves the details of a single task using its unique `taskId`.

- It returns information such as title, description, due date, status, priority, and project association.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
taskId	string	path	Y	ID of the task to retrieve

Responses

Status Code	Description
200 OK	It returns the full details of the task matching the given task ID.
404 Not Found	It indicates when the specified task ID does not exist.
500 Internal Server Error	Internal server error occurs.
503 Service Unavailable	Service is not available.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Task retrieved successfully."
  },
  "data": {
    "taskId": "task_001",
    "title": "Write outline",
    "description": "Plan the structure of the upcoming article",
    "dueDate": "2025-05-20T08:00:00Z",
    "status": "not-started",
    "priority": "high",
    "projectId": "proj_001",
    "createdAt": "2025-04-17T09:00:00Z"
  }
}
```

`PATCH /tasks/:taskId` `- Update a Task by ID`

This endpoint allows users to update one or more fields of an existing task.

- The request must include at least one field to update.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description

taskId	string	path	Y	ID of the task to update

Parameters (at least one is required)

Name	Type	In	Required	Description
title	string	body	N	New title for the task
description	string	body	N	Detailed description of the task
dueDate	string	body	N	Due date in ISO 8601 format
status	string	body	N	Task status (e.g., `not-started`)
priority	string	body	N	Task priority (`low`, `medium`, `high`)

Example Request Body

```
{
  "status": "in-progress",
  "priority": "high"
}
```

Responses

Status Code	Description
200 OK	The task was successfully updated with the provided fields.
400 Bad Request	It indicates when the request body is empty or contains invalid fields.
404 Not Found	It indicates when the specified task ID does not exist.
422 Unprocessable Entity	It indicates when an enum-type field (status or priority) contains an invalid value.
500 Internal Server Error	Internal server error occurs.
503 Service Unavailable	Service is not available.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Task updated successfully."
  },
  "data": {
    "taskId": "task_003",
    "title": "Finalize research",
    "description": "Wrap up findings for the concept of new novel.",
    "dueDate": "2025-04-22T13:00:00Z",
    "status": "in-progress",
    "priority": "high",
    "projectId": "proj_001",
    "createdAt": "2025-04-17T09:20:00Z"
  }
}
```

`DELETE /tasks/:taskId - Delete a Task by ID`

This endpoint allows users to delete a specific task by its ID.

- Once deleted, the task cannot be recovered.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
taskId	string	path	Y	ID of the task to delete

Responses

Status Code	Description
204 No Content	The task was successfully deleted.
404 Not Found	It indicates when the specified task ID does not exist.
500 Internal Server Error	Internal server error occurs.

Example Response Body

```
{
  "code": 204,
  "message": "Project deleted successfully."
}
```

***Note**: The actual 204 response will return an empty body. This example is included for consistency in response format documentation.*

6. Overview

`GET /tasks/overview - Get an Overview of All Tasks`

This endpoint retrieves all tasks across all projects, optionally filtered by date range, project, or status.

- It is intended for calendar-style overview screens where users can see all their deadlines at a glance.
- Bearer token is required.

Parameters

Name	Type	In	Required	Description
startDate	string	query	N	ISO date string to filter tasks from
endDate	string	query	N	ISO date string to filter tasks to
status	string	query	N	Filter tasks by status (e.g., `in-progress`)
projectId	string	query	N	Filter tasks by a specific project ID

Responses

Status Code	Description
200 OK	The filtered list of tasks was successfully retrieved.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Tasks overview retrieved successfully."
  },
  "data": [
    {
      "taskId": "task_004",
      "title": "Draft writing",
      "dueDate": "2025-05-24T09:00:00Z",
      "status": "delayed",
      "priority": "medium",
      "projectId": "proj_001"
    },
    {
      "taskId": "task_006",
      "title": "Grocery shopping",
      "dueDate": "2025-05-20T11:00:00Z",
      "status": "not-started",
      "priority": "medium",
      "projectId": "proj_002"
    }
  ]
}
```

7. Notifications

`GET /notifications` – Get All Notifications

This endpoint retrieves a list of all notifications associated with the authenticated user.

- Notifications may include task reminders, deadline alerts, or status updates (e.g., delayed, due soon, completed).
- Bearer token is required.

Responses

Status Code	Description
200 OK	The list of notifications is successfully retrieved.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Notifications retrieved successfully."
  },
  "data": [
    {
      "notificationId": "n_001",
      "taskId": "task_004",
      "type": "due-soon",
      "message": "Task 'Draft writing' is due tomorrow.",
      "read": false,
      "createdAt": "2025-05-27T08:00:00Z"
    },
    {
      "notificationId": "n_002",
      "taskId": "task_005",
      "type": "delayed",
      "message": "Task 'Review references' is delayed.",
      "read": true,
      "createdAt": "2025-05-25T12:00:00Z"
    }
  ]
}
```

PATCH /notifications/:notificationId - Mark Notification as Read

This endpoint allows users to mark a specific notification as "read".

- It updates `read` status of the given notification.
- Bearer token is required.

Path Parameters

Name	Type	In	Required	Description
notificationId	string	path	Y	ID of the notification to update

Parameters

Name	Type	In	Required	Description
read	boolean	body	Y	Must be `true` to mark as read

Example Request Body

```
{
  "read": true
}
```

Responses

Status Code	Description
200 OK	The notification was successfully marked as read.
400 Bad Request	It indicates when the body is missing or does not contain a valid `read` field.
503 Service Unavailable	Service is not available.

Example Response Body

```
{
  "meta": {
    "code": 200,
    "message": "Notification marked as read."
  },
  "data": {
    "notificationId": "n_001",
    "taskId": "task_004",
    "read": true,
    "updatedAt": "2025-04-23T12:00:00Z"
  }
}
```