

CAVITE STATE UNIVERSITY
Imus Campus
Cavite Civic Center Palico IV, Imus, Cavite
(046) 471-66-07 / (046) 471-67-70 / (046) 686- 23-49
www.cvsu.edu.ph

DEPARTMENT OF COMPUTER STUDIES
ITEC 116 – IT ELECTIVE 4 (SYSTEM INTEGRATION AND ARCHITECTURE 2)

ACTIVITY NO. 1 TO-DO LIST API + UI

Title of Activity

ConVINCE Task Manager

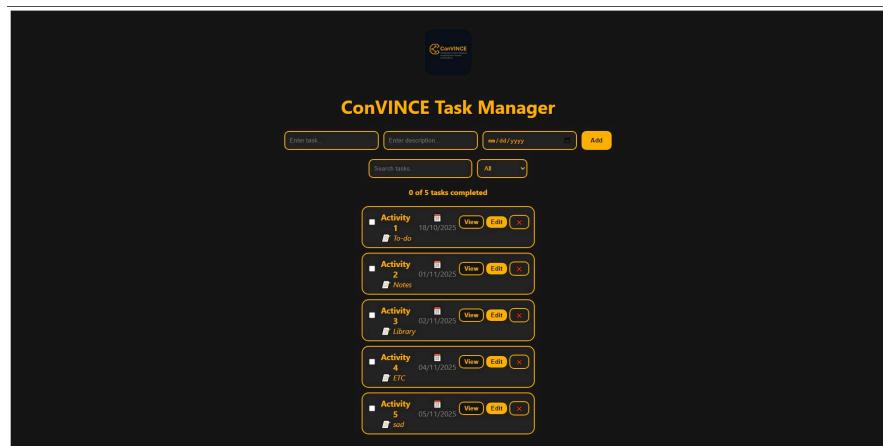
Short Description(What the app does)

The **ConVINCE Task Manager** is a full-stack To-Do List web application that helps users organize and track their daily tasks. It allows users to **add**, **view**, **edit**, **delete**, **search**, and **filter tasks** with deadlines and completion tracking. The **frontend** is developed using **React.js (Vite)**, while the **backend** is powered by **NestJS**, providing a structured **REST API** for task management and integrated **Swagger documentation** for API visualization.

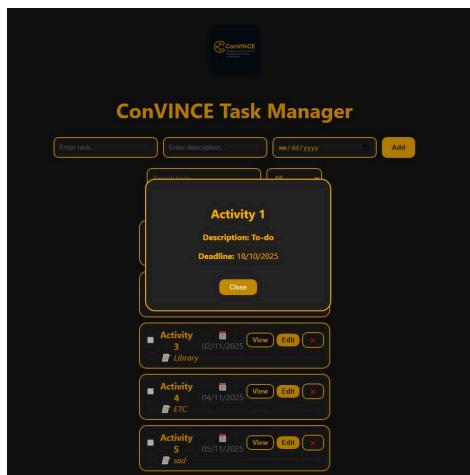
Screenshot(s) of working system (UI and API example)

User Interface (Frontend):

- **Task List (Main UI)**



- **Task View (Modal Popup)**



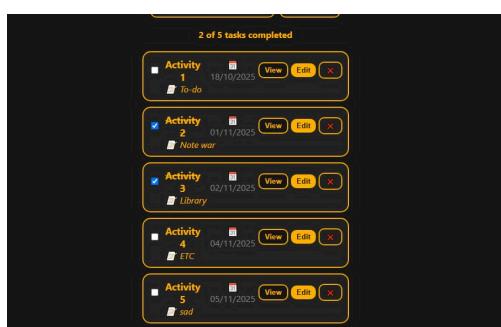
- **Task Edit Form**



- **Search and Filter Section**



- **Progress Tracker**



API Backend:

- API Endpoint (GET /api/tasks)

The screenshot shows the API documentation for the GET /tasks/{id} endpoint. It includes fields for entering parameters (id), executing the request, and viewing responses (curl command, request URL, server response, detailed headers, and detailed descriptions for status codes 200 and 404).

Responses

Request URL: `http://localhost:3001/tasks/1`

Server response:

Code **Details**

200 Response headers

```
access-control-allow-credentials: true
connection: keep-alive
content-length: 113
date: Wed, 19 Oct 2025 01:19:37 GMT
keep-alive: timeout=5
set-cookie: session=abc123
x-powered-by: Express
```

Responses

Code **Description**

200 Returns the task with the specified ID.

404 Task not found.

Request URL: `http://localhost:3001/tasks`

Server response:

Code **Details**

200 Response body

```
[{"id": 1, "title": "Activity 1", "description": "To-do", "completed": false, "deadline": "2025-10-18"}, {"id": 2, "title": "Activity 2", "description": "Note var", "completed": false, "deadline": "2025-11-01"}, {"id": 3, "title": "Activity 3", "description": "Meeting", "completed": true, "deadline": "2025-11-02"}, {"id": 4, "title": "Activity 4", "description": "Call", "completed": false, "deadline": "2025-11-04"}]
```

Response headers

```
access-control-allow-credentials: true
connection: keep-alive
content-length: 260
content-type: application/json; charset=utf-8
date: Wed, 19 Oct 2025 01:18:36 GMT
keep-alive: timeout=5
set-cookie: session=abc123
x-powered-by: Express
```

Responses

Code **Description**

200 Returns all tasks.

- PATCH /tasks/{id}

The screenshot shows the API documentation for the PATCH /tasks/{id} endpoint. It includes fields for entering parameters (id), executing the request, and viewing responses (curl command, request URL, server response, detailed headers, and detailed descriptions for status code 200).

Responses

Request URL: `http://localhost:3001/tasks/1`

Server response:

Code **Details**

200 Response headers

```
access-control-allow-credentials: true
connection: keep-alive
content-length: 113
date: Wed, 19 Oct 2025 01:40:07 GMT
keep-alive: timeout=5
set-cookie: session=abc123
x-powered-by: Express
```

Responses

Code **Description**

200 Task updated successfully.

- **DELETE /tasks/{id}**

The screenshot shows the Swagger UI interface for a DELETE operation. At the top, there are buttons for "Execute" and "Clear". Below that is a "Responses" section. Under "Responses", there's a "Carl" tab showing a curl command to delete a task. The "Request URL" is set to `http://localhost:3001/tasks/1`. The "Server response" section shows a status code of 200 with a JSON response body and various response headers. Below this, there's a table for "Responses" with rows for status codes 200 and 404, each with their respective descriptions and links.

Code	Description	Links
200	Task deleted successfully	No links
404	Task not found.	No links

Instruction on how to run the project

Requirements

- Node.js
- npm (comes with Node.js)
- Web Browser (Google Chrome, Edge, Firefox, etc.)
- Visual Studio Code
- NestJS
- React.js (Vite)
- TypeORM
- Swagger UI
- Git / GitHub

Steps to run the Project

1. Clone or download the project from GitHub.
2. Open the project folder in Visual Studio Code. (VScode)
3. Open a terminal and go to the backend folder:

```
cd backend
```

```
npm install
```

```
npm run start:dev
```

→ Runs the backend (NestJS) at <http://localhost:3000/>

→ Swagger UI: <http://localhost:3000/api>

4. Open another terminal and go to the frontend folder:

```
cd frontend
```

```
npm install
```

```
npm run dev
```

→ **Runs the frontend (React + Vite) at <http://localhost:5173/>**

5. Open the browser and go to the frontend URL to use the system.