

Activity 1: Todo List App

GROUP MEMBERS:

Abelita, Czyrell Gwen R.

Aranzasu, Darryl Jedidiah A.

Aranzasu Darrylene Kaela A.

Gangan, Demrose Carla C.

Moral, Andrie B.

Relox, Nathaniel

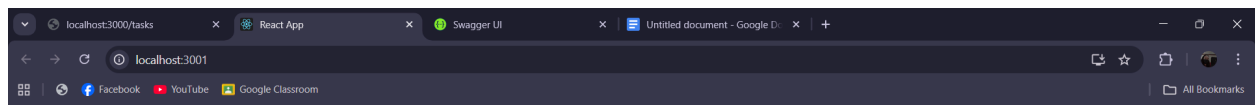
Activity 1: Todo List App

The todo list app allows users to manage their daily tasks. Users can create tasks, update the existing tasks and delete tasks once they're done doing it. It helps to organize work and keep track of things that they need to complete during the whole day. The interface is user-friendly so anyone can use it without difficulty.

Screenshot(s) of working System (UI + API example)

UI:

Adding a Task:



Todo List App

Add Task

call sister

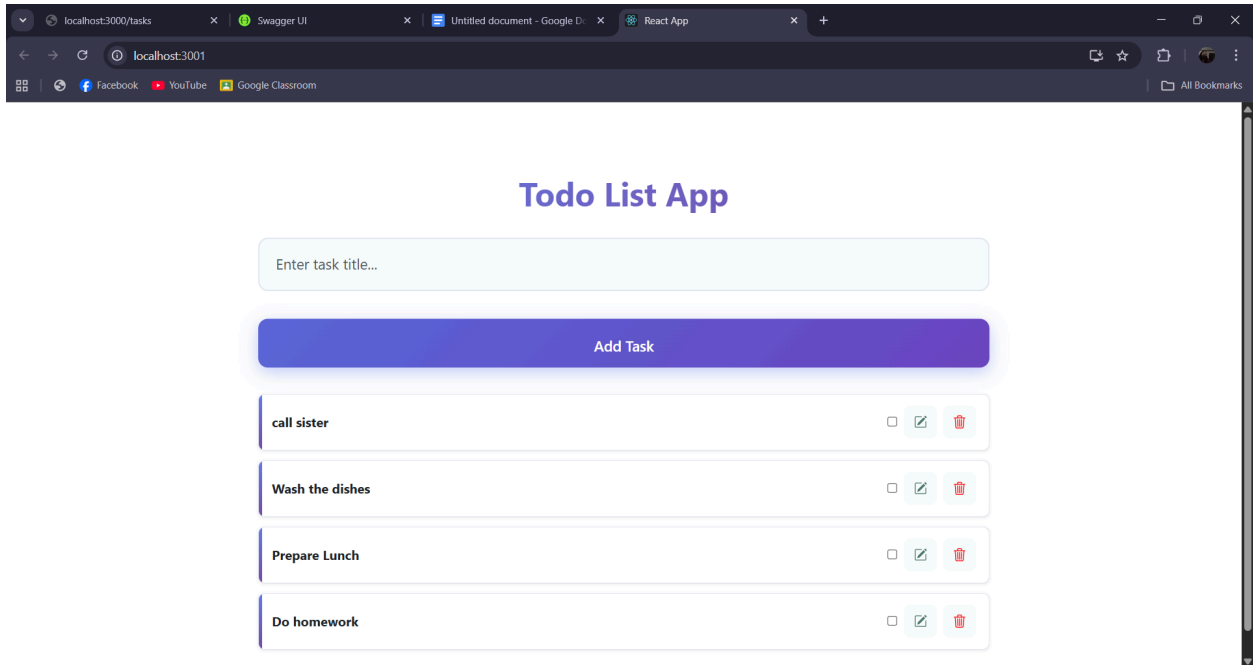
☐ ☒ ☐

Wash the dishes

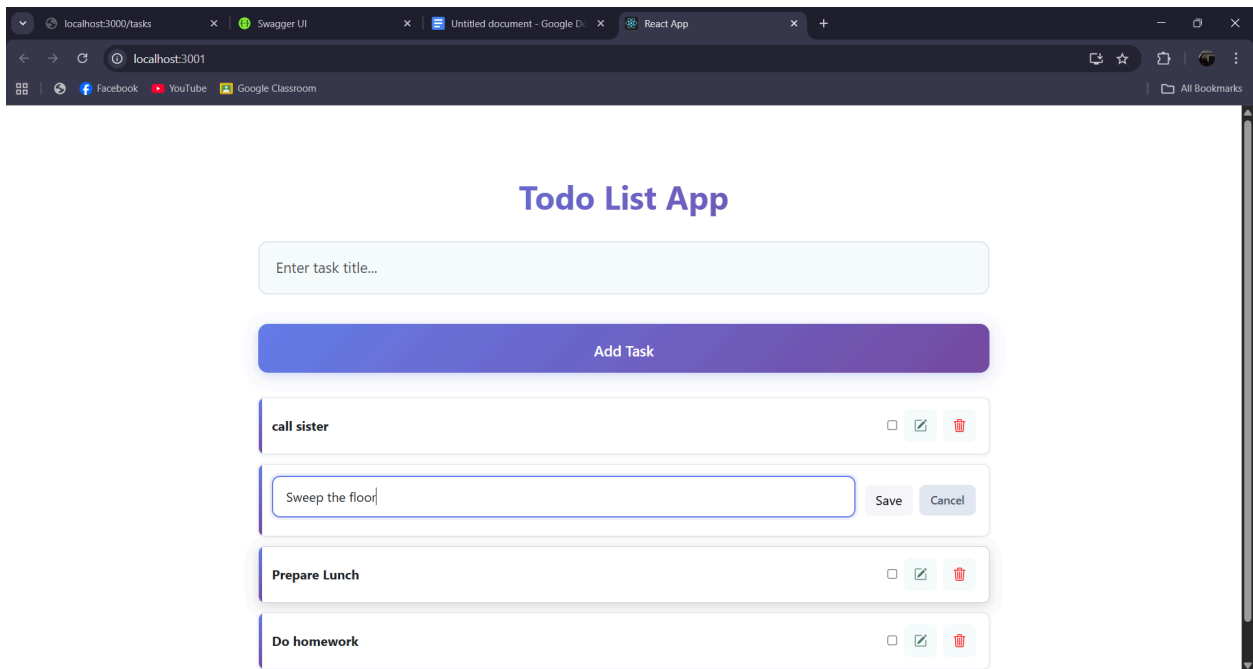
☐ ☒ ☐

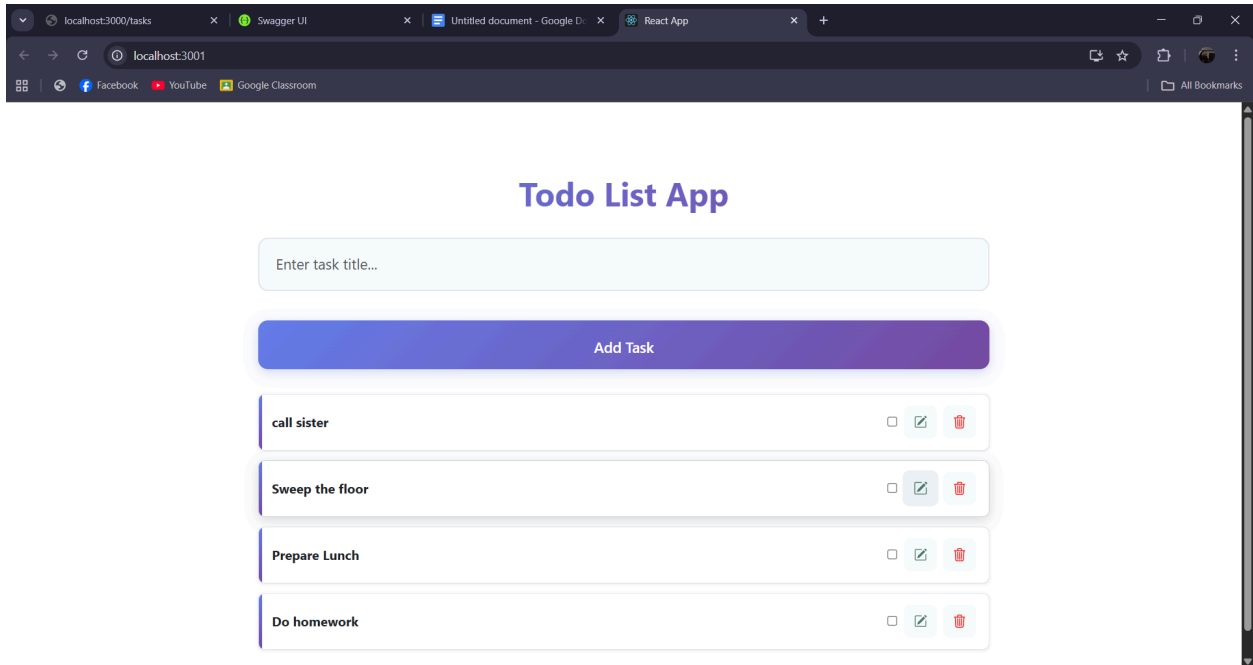
Prepare Lunch

☐ ☒ ☐

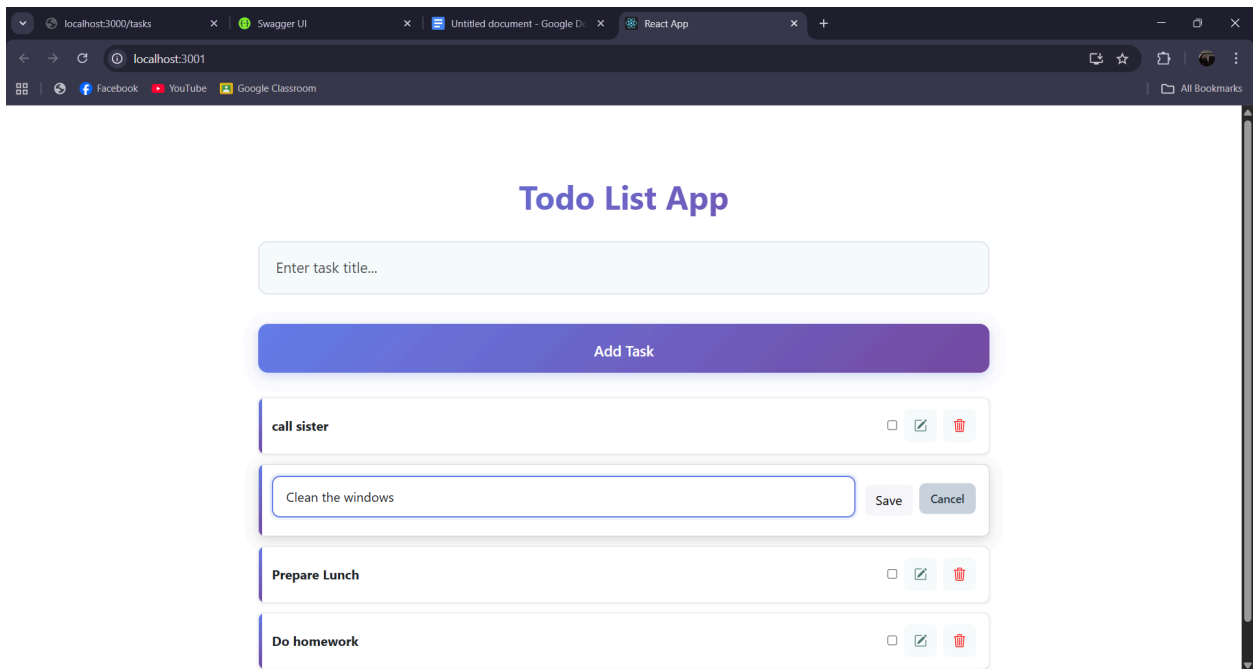


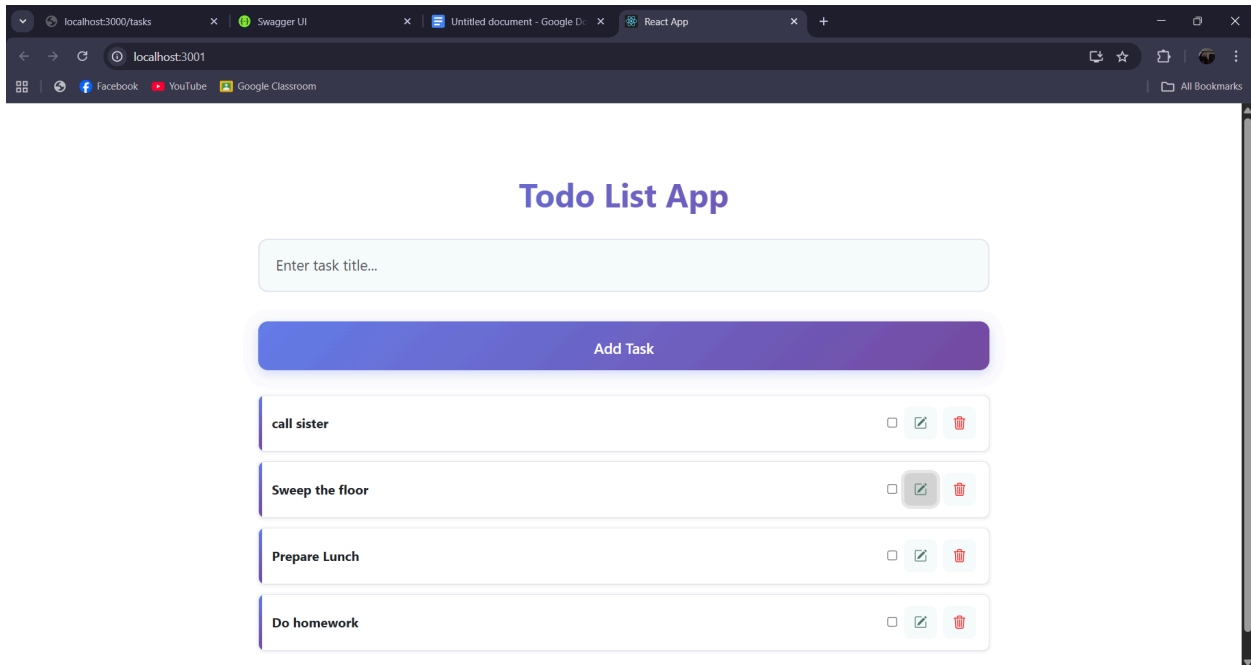
Updating the Task:



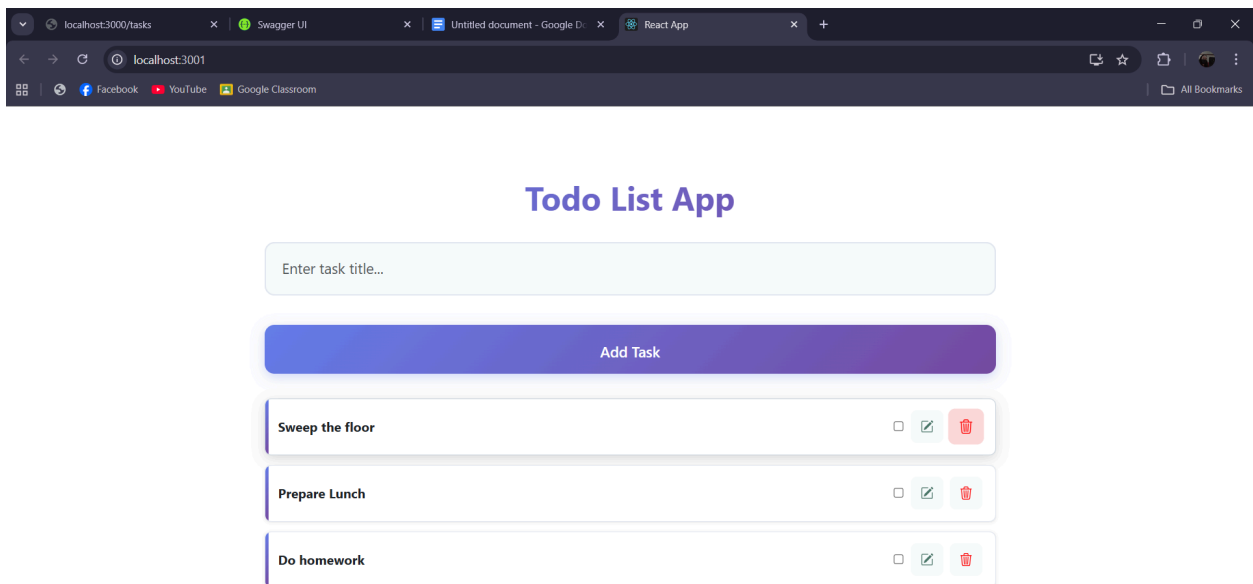


Cancelling a updating Task:



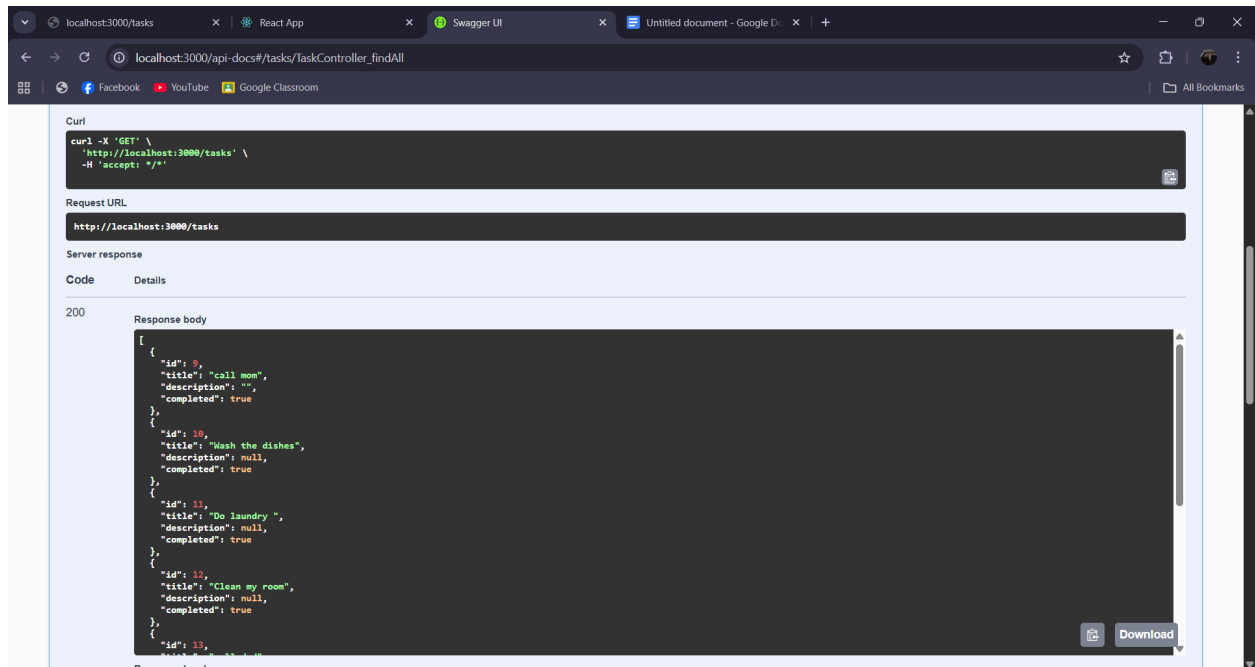


Deleting a task:



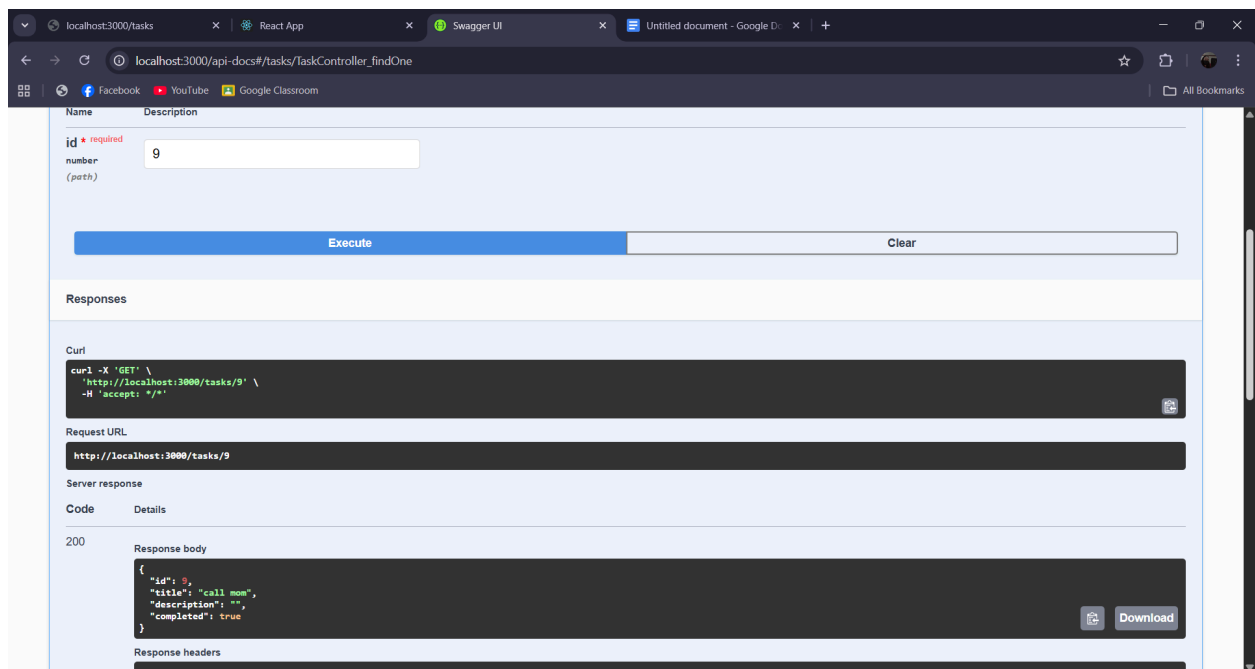
Swagger (API Docs)

Get All:



A screenshot of the Swagger UI interface in a web browser. The browser's address bar shows the URL `localhost:3000/api-docs#/tasks/taskController_findAll`. The interface displays the 'Curl' section with the command `curl -X 'GET' \ 'http://localhost:3000/tasks' \ -H 'accept: */*'`. Below this, the 'Request URL' is `http://localhost:3000/tasks`. The 'Server response' section shows a '200' status code. The 'Response body' is a JSON array of task objects, including tasks with IDs 9, 10, 11, 12, and 13. Each object contains 'id', 'title', 'description', and 'completed' fields. A 'Download' button is visible next to the response body.

Get ID:



A screenshot of the Swagger UI interface for the 'Get ID' endpoint. The browser's address bar shows the URL `localhost:3000/api-docs#/tasks/taskController_findOne`. The 'Name' field is 'id' (required, number, path) and the 'Description' field is empty. A text input field contains the value '9'. Below the input fields are 'Execute' and 'Clear' buttons. The 'Responses' section shows a '200' status code. The 'Response body' is a JSON object representing a single task with ID 9, including 'id', 'title', 'description', and 'completed' fields. A 'Download' button is visible next to the response body.

Post ID:

The image shows the Swagger UI interface for a REST API. The URL is `localhost:3000/api-docs/`. The selected endpoint is `POST /tasks/{id}`. The **Parameters** section shows a required path parameter `id` with the value `18`. The **Request body** section is set to `application/json` and contains the following JSON:

```
{  "title": "Water change the aquarium",  "completed": false}
```

Buttons for **Execute** and **Clear** are at the bottom.

The image shows the Swagger UI interface displaying the server response for the `POST /tasks/18` request. The **Request URL** is `http://localhost:3000/tasks/18`. The **Server response** section shows a **Code** of `201` and a **Response body** containing the following JSON:

```
{  "title": "Water change the aquarium",  "completed": false,  "description": null,  "id": 18}
```

The **Response headers** section shows the following headers:

```
access-control-allow-origin: http://localhost:3001
connection: keep-alive
content-length: 82
content-type: application/json; charset=utf-8
date: Fri, 17 Oct 2025 07:51:13 GMT
etag: W/"52-vYAS6648V6uMxGJelBscL9/#A"
keep-alive: timeout=5
vary: Origin
x-powered-by: Express
```

The **Responses** section shows a table with the following data:

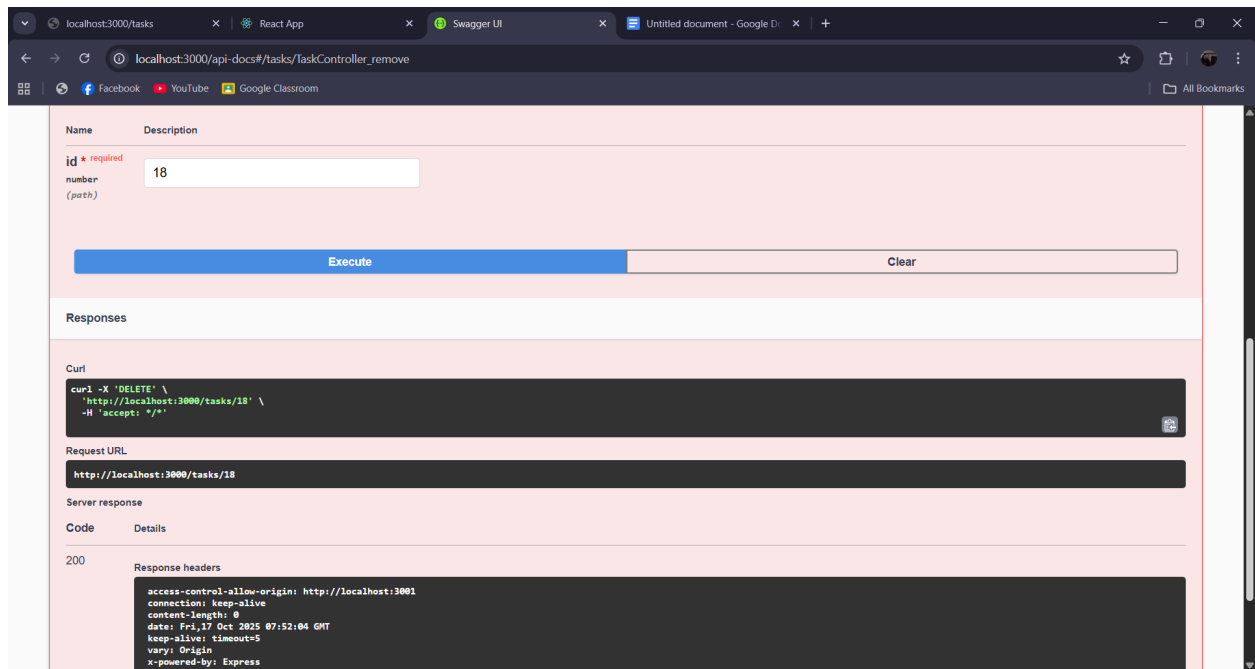
Code	Description	Links
201		No links

Put ID:

Swagger UI interface for a PUT request to `localhost:3000/tasks/18`. The interface includes a "Parameters" section with a required `id` parameter (number, path) set to `18`. The "Request body" section is set to `application/json` and contains a JSON object: `{ "title": "Clean Room", "description": "", "completed": false }`. The "Responses" section is currently empty. Buttons for "Cancel", "Reset", "Execute", and "Clear" are visible.

Swagger UI interface showing the result of the PUT request. The "Curl" section displays the command: `curl -X 'PUT' \ 'http://localhost:3000/tasks/18' \ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d '{ "title": "Clean Room", "description": "", "completed": false }'`. The "Request URL" is `http://localhost:3000/tasks/18`. The "Server response" section shows a `200` status code. The "Response body" is a JSON object: `{ "id": 18, "title": "Clean Room", "description": "", "completed": false }`. The "Response headers" include: `access-control-allow-origin: http://localhost:3001`, `connection: keep-alive`, `content-length: 65`, `content-type: application/json; charset=utf-8`, `date: Fri, 17 Oct 2025 07:58:08 GMT`, `etag: W/"41-ApeIhaQmet/7ACSL/sHgCmPKx"`, `keep-alive: timeout=5`, `vary: Origin`, and `x-powered-by: Express`. A "Download" button is present next to the response body.

Delete ID:



Instructions:

A. Download project from Google Classroom

Step 1: Click the attached zip or folder. Choose **Download**. If teacher gave a Git URL, copy it.

Step 2: Unzip the file and open the extracted folder in VS Code.

B. Prepare your machine

Step 3: Install Node.js (recommended LTS) from nodejs.org. Verify:

```
node -v
```

```
npm -v
```

Step 4: Install Yarn:

```
npm install -g yarn
```

C. Install dependencies

Step 5: Open terminal and go to backend folder:

```
cd path/to/project/backend/todo-backend
```

Step 6: Install packages:

```
npm install
```

or if using yarn

```
yarn
```

Step 7: In a new terminal tab, go to frontend folder:

```
cd path/to/project/frontend/todo-frontend  
npm install  
or yarn
```

D. Run backend and frontend

Step 8: Start backend (in backend folder):

```
npm run start:dev  
or npm run start
```

Step 9: Wait for output like **App is running at http://localhost:3000**

Step 10: Start frontend (in frontend folder):

```
npm start  
or npm run dev or yarn start
```

Step 11: Open browser: frontend usually **http://localhost:3001** or **http://localhost:3000** — check terminal output. If backend uses 3000, frontend will use another port like 3001 or 5173 (Vite)

Step 12: To view API docs (if using Swagger with NestJS): open

http://localhost:3000/api-docs

F. Test

15. Use the frontend UI to add/read/update/delete tasks. Or use Postman to call API endpoints.

Backend Link:

http://localhost:3000

Frontend Link:

http://localhost:3001

API DOCU Link (Swagger):

<http://localhost:3000/api-docs>