

## **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

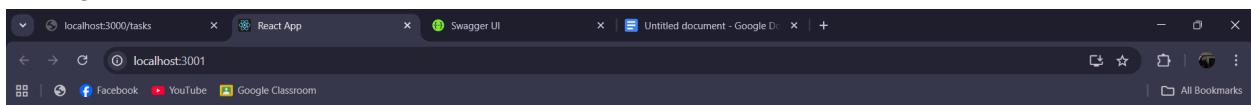
## Activity 1: Todo List App

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.

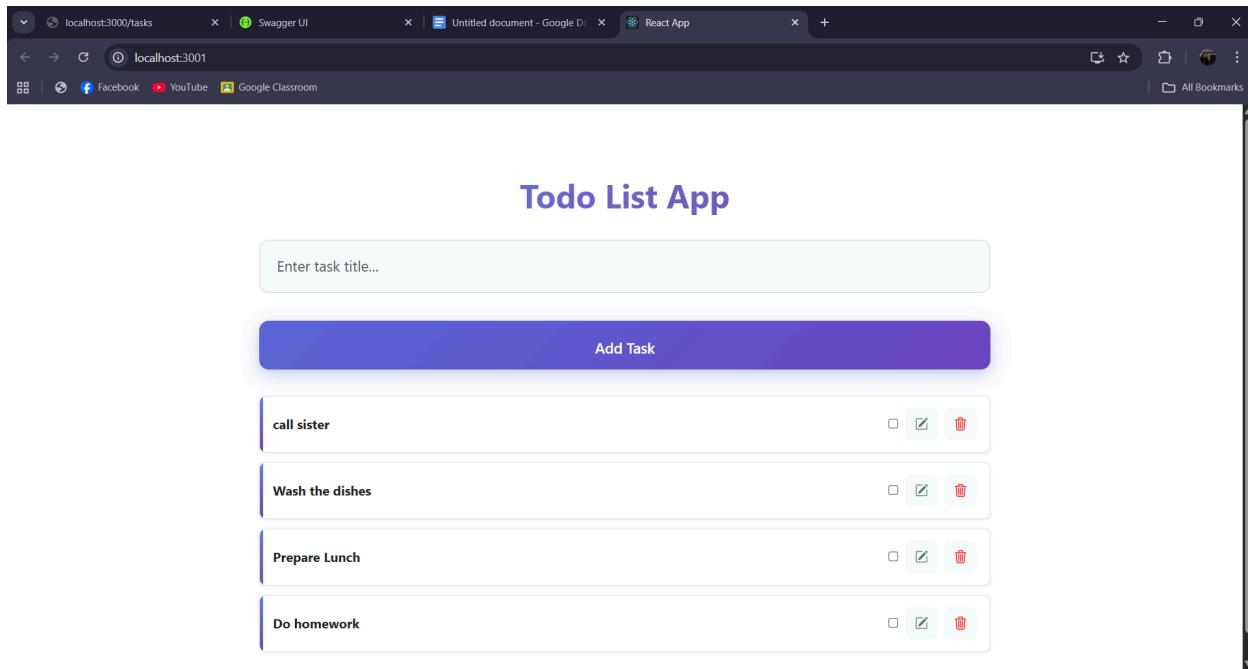
### Screenshot(s) of working System (UI + API example)

#### UI:

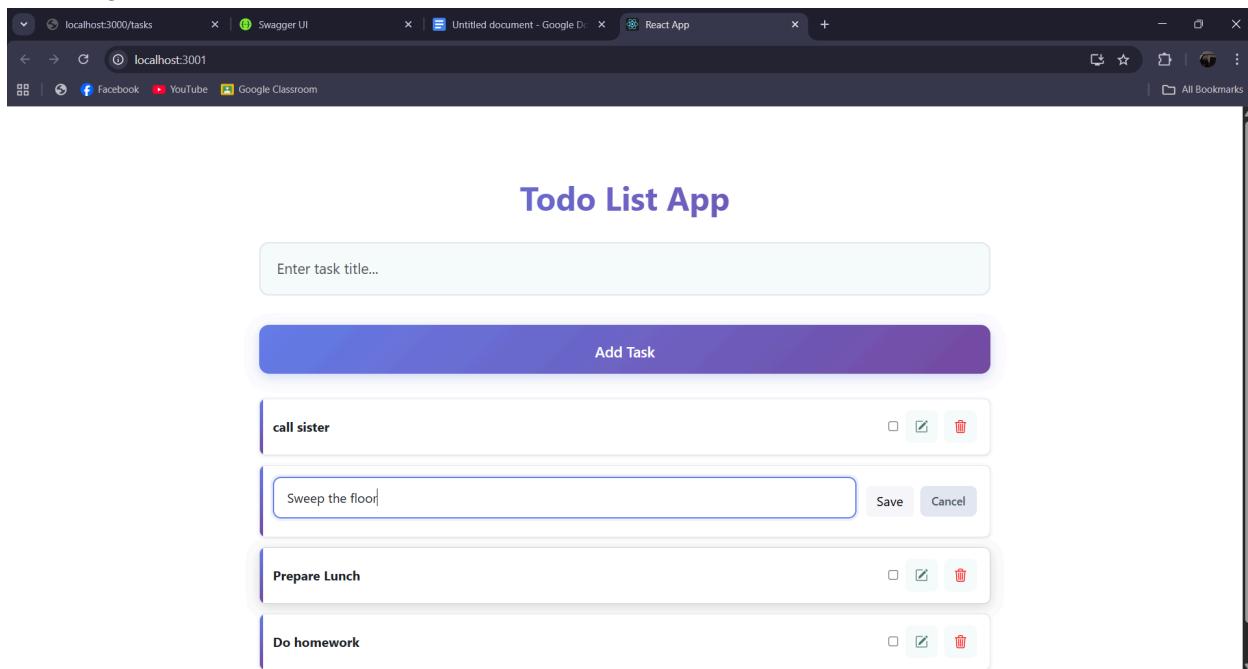
Adding a Task:

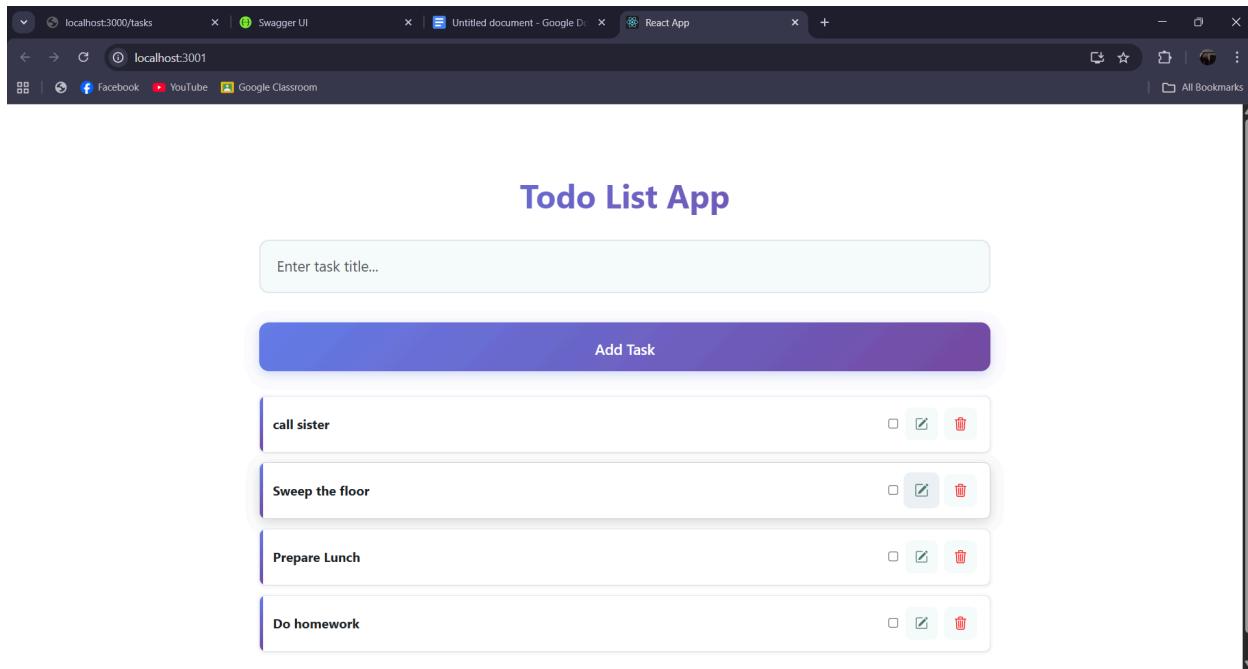


The screenshot shows the 'Todo List App' interface. At the top center is the title 'Todo List App'. Below it is a search bar containing the text 'Do homework'. A large purple button with the text 'Add Task' is centered below the search bar. Below this button are three task cards, each consisting of a text input field and a set of icons. The first card contains the text 'call sister' and has a checkbox, a checkmark icon, and a trash bin icon. The second card contains the text 'Wash the dishes' and has a checkbox, a checkmark icon, and a trash bin icon. The third card contains the text 'Prepare Lunch' and has a checkbox, a checkmark icon, and a trash bin icon.

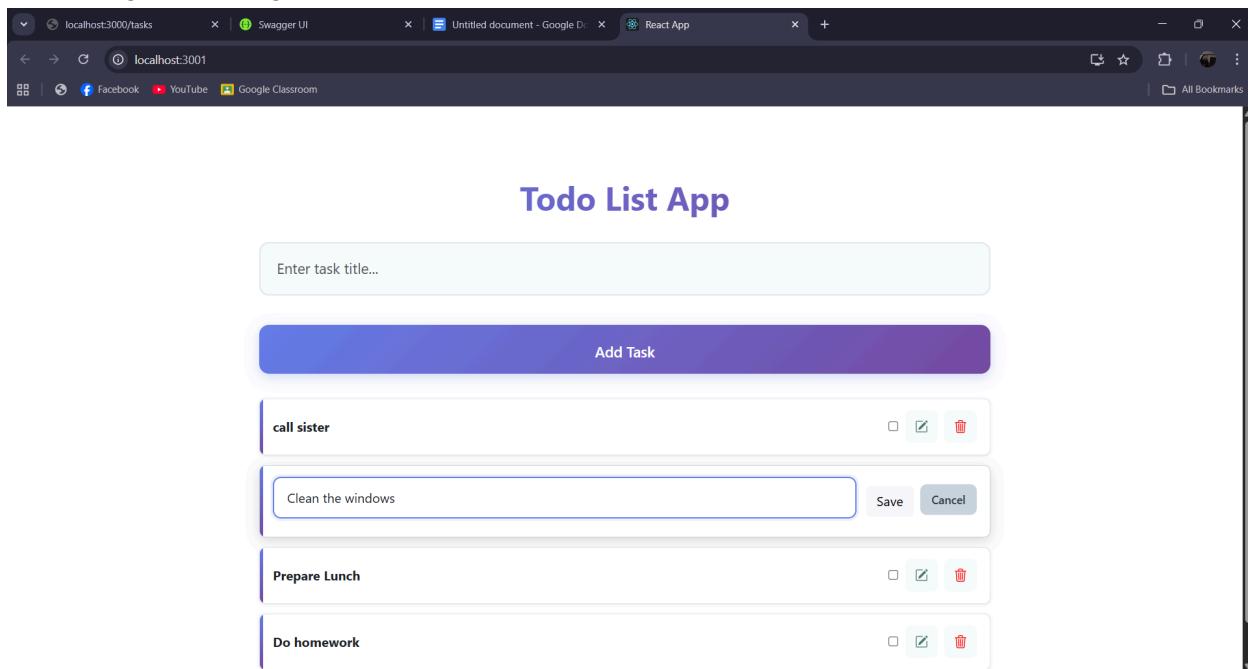


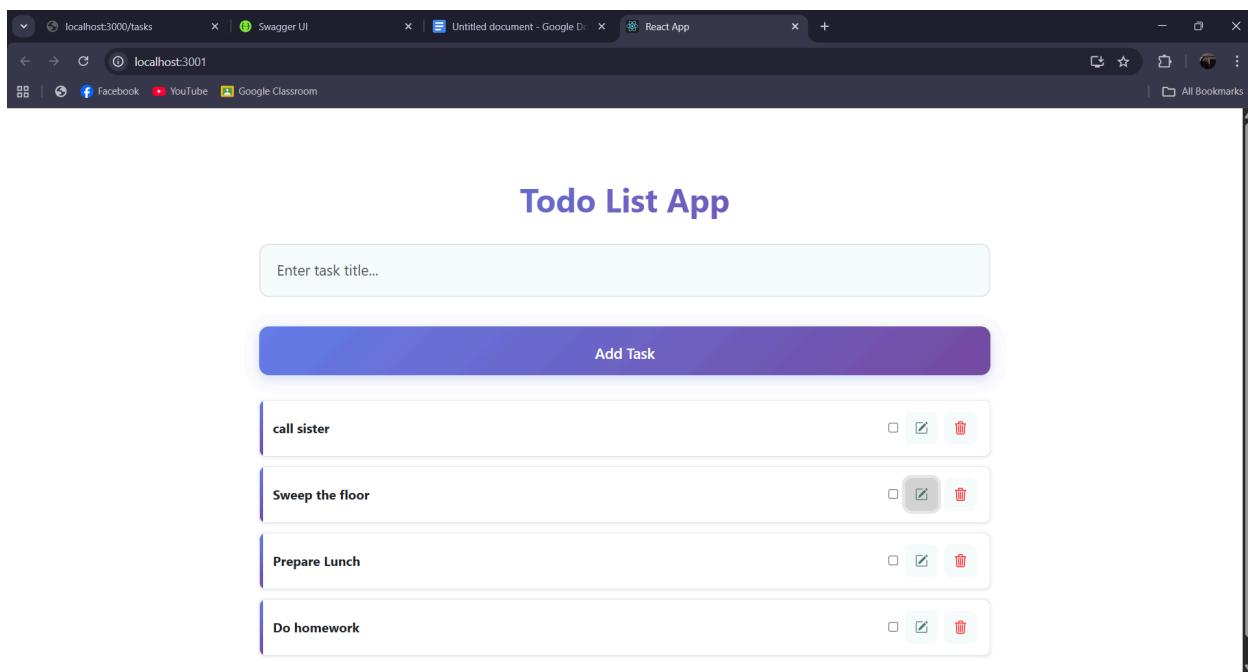
### Updating the Task:



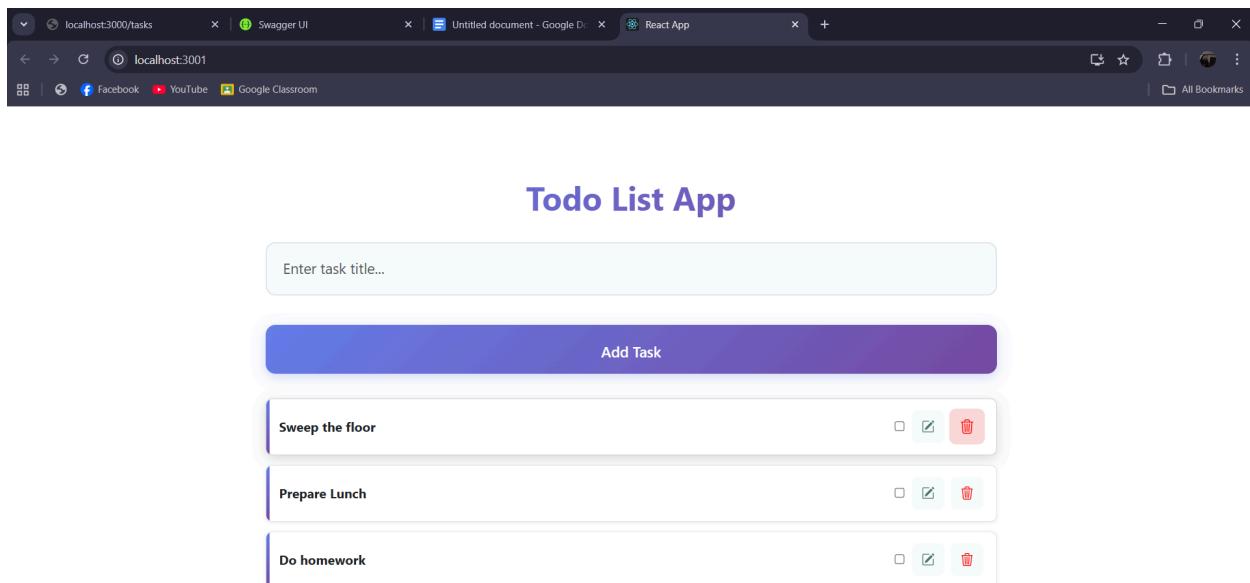


Cancelling a updating Task:





Deleting a task:



## Swagger (API Docs)

Get All:

The screenshot shows the Swagger UI interface for a 'tasks' endpoint. At the top, there's a 'Curl' section with a command line example:

```
curl -X 'GET' \
'http://localhost:3000/tasks' \
-H 'accept: */'
```

Below it is a 'Request URL' field containing `http://localhost:3000/tasks`. The 'Server response' section is expanded, showing a 'Code' tab selected. Under 'Response body', the status code is 200, and the response content is a JSON array of task objects:

```
[  
  {  
    "id": 9,  
    "title": "call mom",  
    "description": "",  
    "completed": true  
  },  
  {  
    "id": 10,  
    "title": "Wash the dishes",  
    "description": null,  
    "completed": true  
  },  
  {  
    "id": 11,  
    "title": "Do laundry ",  
    "description": null,  
    "completed": true  
  },  
  {  
    "id": 12,  
    "title": "Clean my room",  
    "description": null,  
    "completed": true  
  },  
  {  
    "id": 13,  
    "title": "...',  
    "description": null,  
    "completed": true  
  }]
```

At the bottom right of the response area are 'Copy' and 'Download' buttons.

Get ID:

The screenshot shows the Swagger UI interface for a 'TaskController.findOne' endpoint. In the 'Parameters' section, there is a required parameter 'id' with the value '9' entered into the input field. Below the parameters is an 'Execute' button.

The 'Responses' section is expanded, showing a 'Code' tab selected. Under 'Response body', the status code is 200, and the response content is a JSON object representing a single task:

```
{  
  "id": 9,  
  "title": "call mom",  
  "description": "",  
  "completed": true  
}
```

At the bottom right of the response area are 'Copy' and 'Download' buttons.

## Post ID:

The screenshot shows the Swagger UI interface for a POST request to `/tasks/{id}`. The 'Parameters' section contains a single parameter `id` with a value of `18`. The 'Request body' section is set to `application/json` and contains the following JSON payload:

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

At the bottom, there are 'Execute' and 'Clear' buttons.

The screenshot shows the results of the POST request. The 'Request URL' is `http://localhost:3000/tasks/18`. The 'Server response' section shows a 201 status code. The 'Response body' is identical to the request body. The 'Response headers' include:

```
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-vY456640v6uKxGjeIzBscI9/fA
keep-alive: timeout=5
vary: Origin
x-powered-by: Express
```

The 'Responses' table shows a single entry for 201 status with no links.

## Put ID:

The screenshot shows the Swagger UI interface for a PUT request. In the 'Parameters' section, there is a single required parameter named 'id' with a value of '18'. The 'Request body' section is set to 'application/json' and contains the following JSON payload:

```
{
  "title": "Clean Room",
  "description": "",
  "completed": false
}
```

At the bottom, there are 'Execute' and 'Clear' buttons.

The screenshot shows the Swagger UI interface after executing the PUT request. It includes a 'Curl' section with the command:

```
curl -X 'PUT' \
'http://localhost:3000/tasks/18' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "title": "Clean Room",
  "description": "",
  "completed": false
}'
```

A 'Request URL' field shows the endpoint: `http://localhost:3000/tasks/18`. The 'Server response' section displays a status code of 200. Under 'Response body', the JSON response is shown:

```
{
  "id": 18,
  "title": "Clean Room",
  "description": "",
  "completed": false
}
```

Under 'Response headers', the following headers are listed:

```
access-control-allow-origin: http://localhost:3001
connection: keep-alive
content-length: 14
content-type: application/json; charset=utf-8
date: Fri, 17 Oct 2025 07:50:08 GMT
etag: W/"41-KgwTAAqNmwt7ACSL/siqQce+Mkk"
keep-alive: timeout=5
vary: Origin
x-powered-by: Express
```

## Delete ID:

The screenshot shows a Swagger UI interface for a REST API. The URL in the address bar is `localhost:3000/api-docs#/tasks/taskController_remove`. The main area displays a form for a `DELETE` request. A parameter `id` is required and has a value of `18`. Below the form are sections for `Responses`, `Curl` (containing the command `curl -X 'DELETE' '\n \'http://localhost:3000/tasks/18'\n -H 'accept: */'`), `Request URL` (`http://localhost:3000/tasks/18`), and `Server response`. The `Server response` section shows a `200` status code with the following headers:  
access-control-allow-origin: http://localhost:3001  
connection: keep-alive  
content-length: 0  
date: Fri,17 Oct 2025 07:52:04 GMT  
keep-alive: timeout=5  
vary: Origin  
x-powered-by: Express

## 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>