

CRUD OPERATION

I have build crud Operation without using any database as per company assignment

1.Retrieve a list of all tasks

```
export const allTask = (req,res) => {
  try {
    if(!tasksArray){
      res.status(404).json({success: false,message: 'Tasks not found'})
    }
    res.status(200).json({
      success: true,
      message: 'Retrieve All Task Succesfully',
      tasksArray
    })
  } catch (error) {
    res.status(400).json({success: false,message:error.message})
  }
};
```

The screenshot shows the Postman interface with a successful API call. The collection is 'company' and the endpoint is 'GET alltask'. The response body is a JSON object:

```
{
  "success": true,
  "message": "Retrieve All Task Succesfully",
  "tasksArray": [
    {
      "id": 1,
      "title": "Home",
      "descriptions": "I have to clean Home"
    },
    {
      "id": 2,
      "title": "Office",
      "descriptions": "I Have completing project"
    }
  ]
}
```

2. Retrieve a specific task by ID

```
export const getTask = (req,res) => {
  try {
    const id = req.params.id;
    if(!id){
      return res.status(404).json({success: false,message: 'Required task Id'})
    }
    const task = tasksArray.find(task => task.id == id);
    if(!task){
      return res.status(404).json({success: false,message: 'This Task not found'})
    }
    res.status(200).json({
      success: true,
      message: 'Retrieve Task Succesfully',
      task
    })
  } catch (error) {
    res.status(400).json({success: false,message:error.message})
  }
};
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Practice' selected, showing collections like 'company' with methods: 'GET alltask', 'GET task by id', 'PUT Update Task', 'POST add task', and 'DEL delete task'. The main area displays a 'GET task by id' request. The URL is set to 'http://localhost:3000/tasks/2'. Below the URL, under 'Params', there is a table with one row: 'Key' (Value) and 'Description'. The response tab at the bottom shows a 200 OK status with a JSON body containing task details: { "success": true, "message": "Retrieve Task Succesfully", "task": { "id": 2, "title": "Office", "descriptions": "I Have completing project" } }.

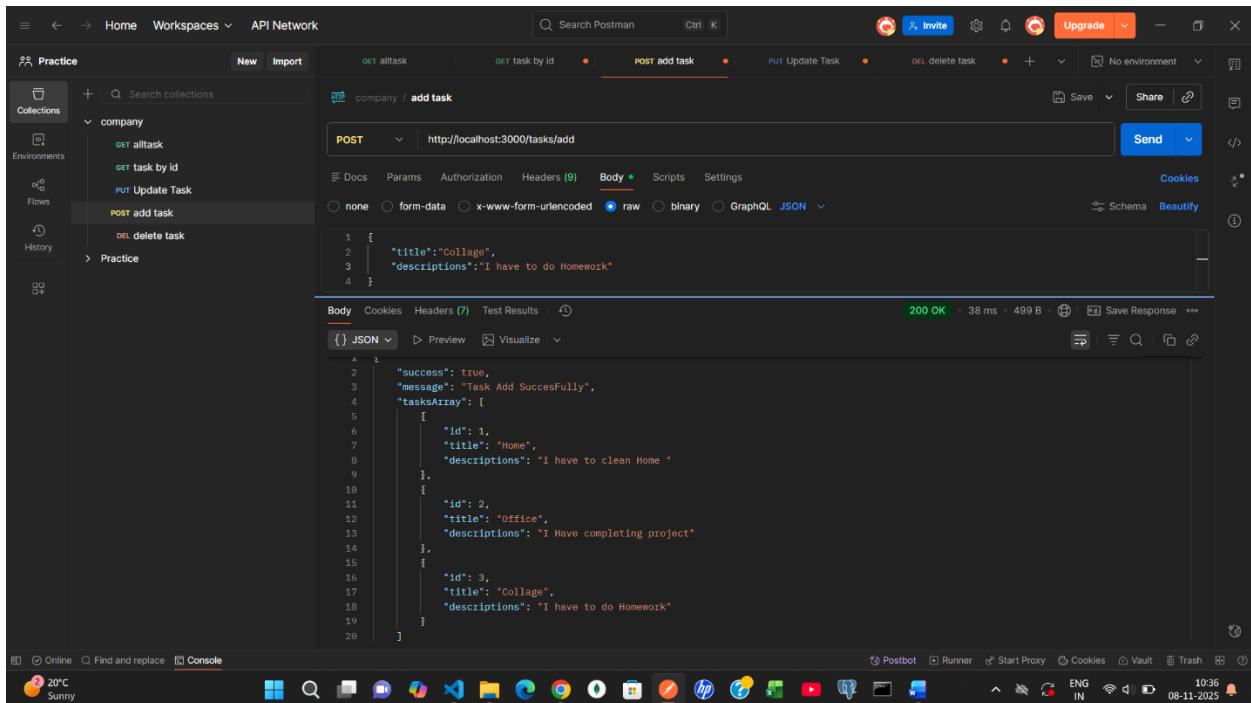
3. Create a new task

```
export const addTask = (req,res) => {
  try {
    const data = req.body;
```

```

if(!data){
    res.status(404).json({success: false,message: 'Task are Required'})
}
tasksArray.push({ id: tasksArray.length + 1, title:data.title, descriptions:data.descriptions });
res.status(200).json({
    success: true,
    message: 'Task Add SuccesFully',
    tasksArray
})
} catch (error) {
    res.status(400).json({
        success: false,
        message: error.message
    })
}
}
}

```



4. Update an existing task by ID

```

export const updateTask = (req,res) => {
    try {
        const task = tasksArray.find(task => task.id == req.params.id);
        if (!task) return res.status(404).json({ message: "task not found" });

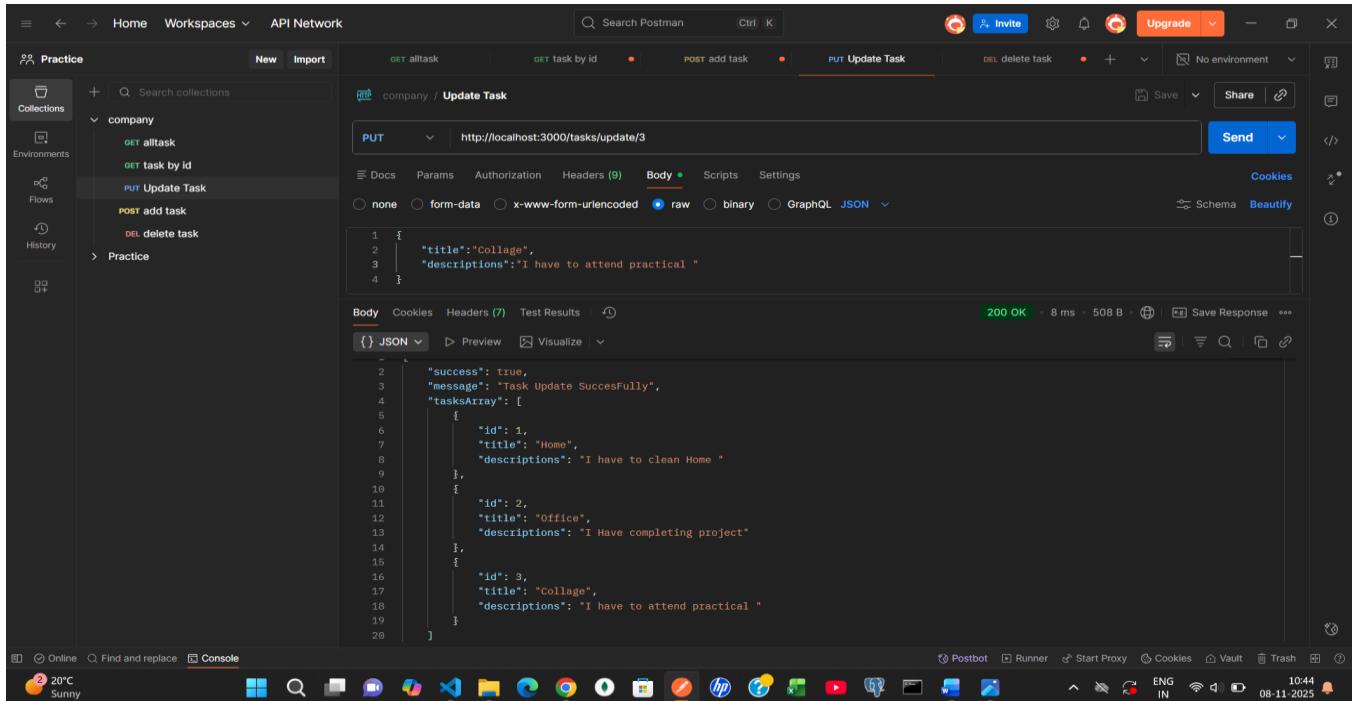
        task.title = req.body.title;
        task.descriptions = req.body.descriptions;
        res.status(200).json({
            success: true,
            message: 'Task Update SuccesFully',
        })
    }
}

```

```

        tasksArray
    })
} catch (error) {
    res.status(400).json({
        success: false,
        message: error.message
    })
}
}
}

```



5. Delete a task by ID

```

export const deleteTask = (req, res) => {
    try {
        const id = parseInt(req.params.id);
        if (!id) {
            return res.status(404).json({
                success: false,
                message: "Task ID is required",
            });
        }
        tasksArray = tasksArray.filter(task => task.id !== id);

        res.status(200).json({
            success: true,
            message: "Task deleted successfully",
            tasksArray,
        });
    }
}

```

```

} catch (error) {
  res.status(400).json({
    success: false,
    message: error.message,
  });
}
};

```

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar is visible with a 'company' collection expanded, showing five endpoints: GET alltask, GET task by id, PUT Update Task, POST add task, and DEL delete task. The 'DEL delete task' endpoint is selected.

In the main workspace, a DELETE request is configured with the URL `http://localhost:3000/tasks/2`. The 'Params' tab is selected, showing a single query parameter 'Key' with the value 'Value'. The 'Headers' tab shows a 'Content-Type' header set to 'application/json'.

At the bottom, the response is displayed in JSON format:

```

1  {
2    "success": true,
3    "message": "Task deleted successfully",
4    "tasksArray": [
5      {
6        "id": 1,
7        "title": "Home",
8        "descriptions": "I have to clean Home"
9      }
10 ]
11

```

The response status is 200 OK with a duration of 14 ms and a size of 368 B. The bottom right corner shows the system tray with various icons and the date/time: 08-11-2025 14:31.