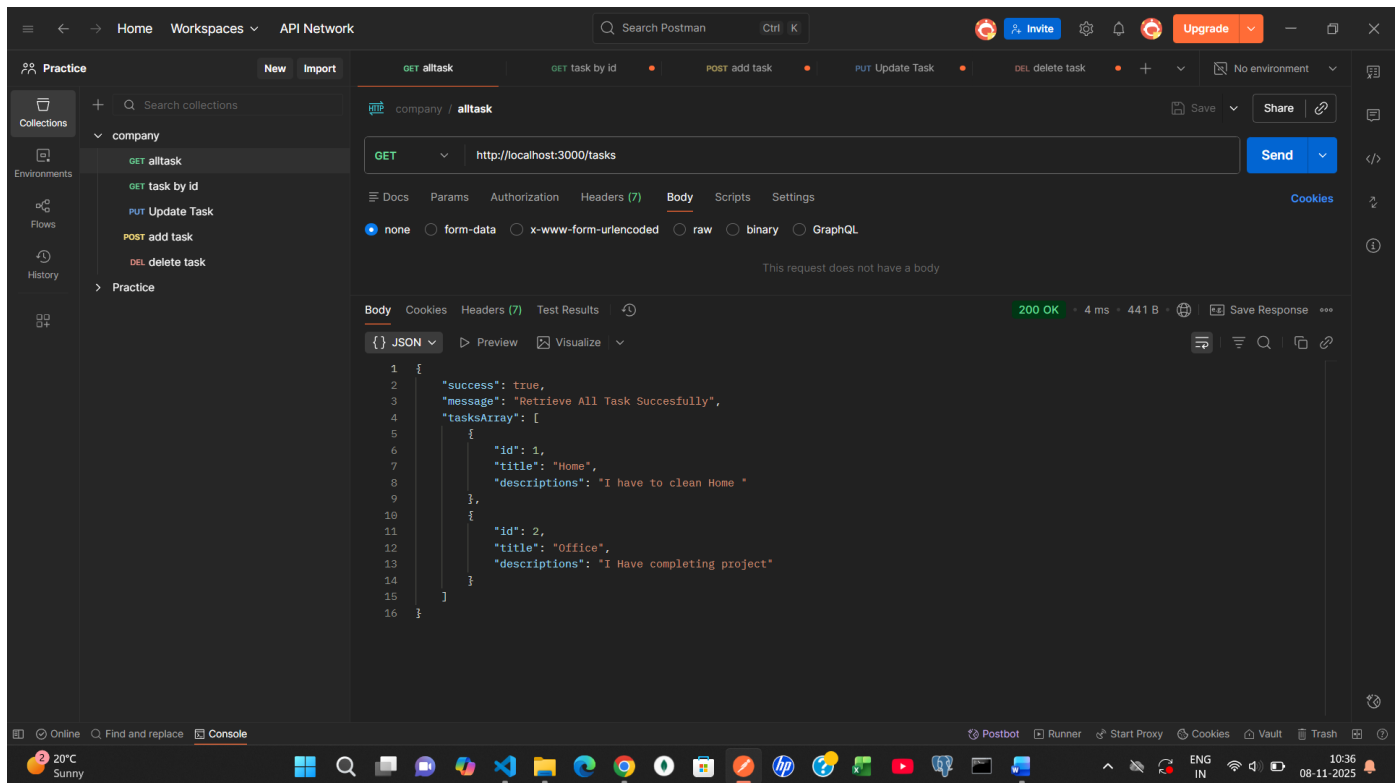


## CRUD OPERATION

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

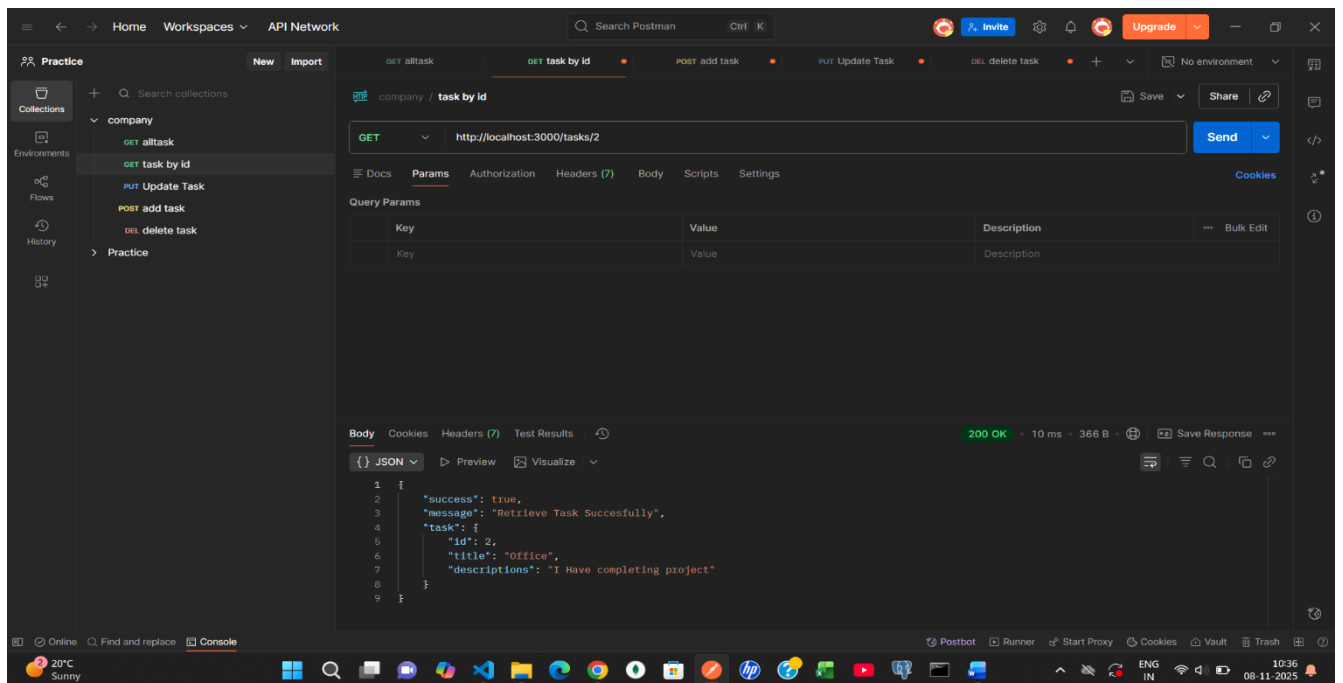
### 1.Retrieve a list of all tasks

```
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})
  }
};
```



## 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 Successfully',
      task
    })
  } catch (error) {
    res.status(400).json({success: false,message:error.message})
  }
};
```



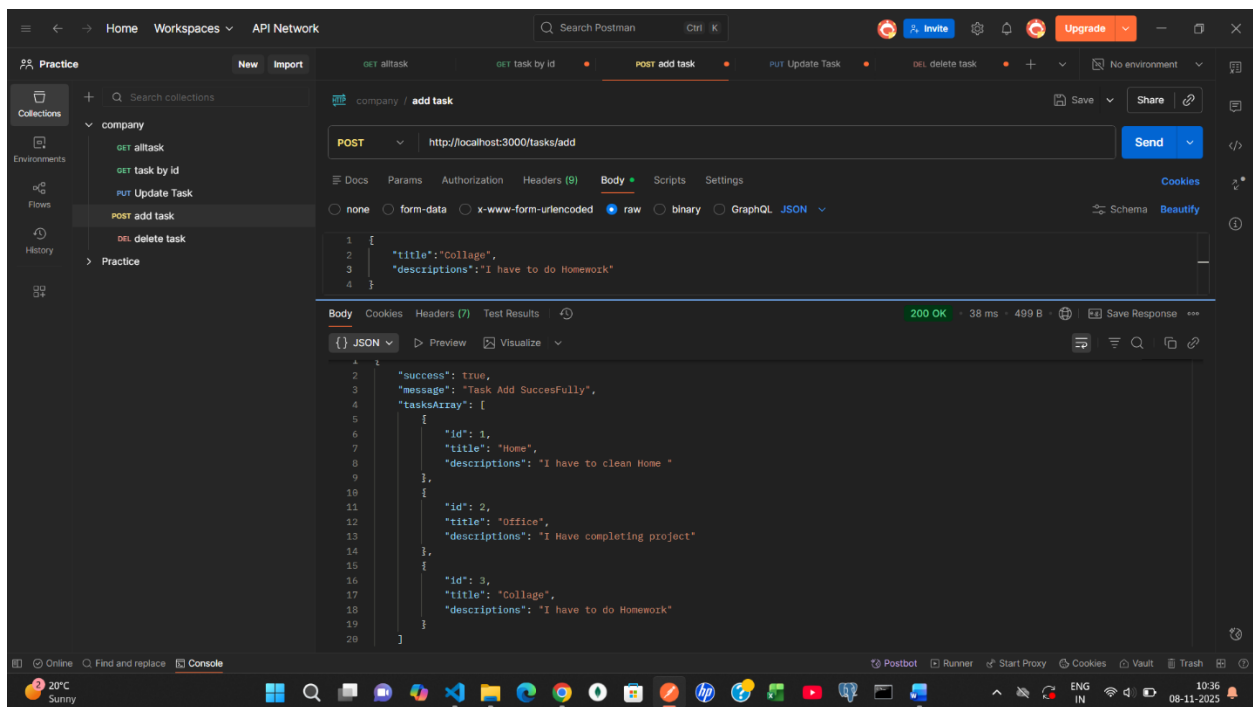
## 3.Create a new task

```
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" });

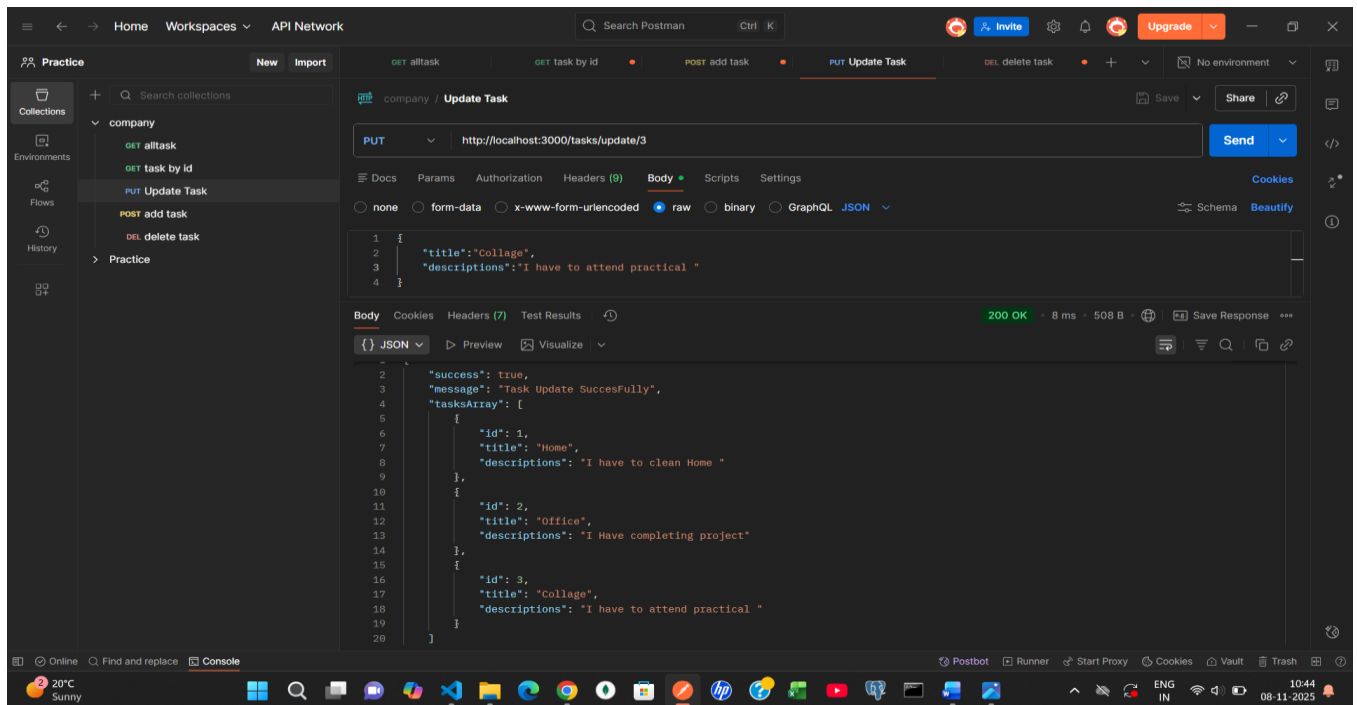
    task.title = req.body.title;
    task.descriptions = req.body.descriptions;
    res.status(200).json({
      success: true,
      message: 'Task Update SuccesFully',

```

```

        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 displays the Postman API client interface. The main workspace shows a **DELETE** request to `http://localhost:3000/tasks/2`. The **Params** tab is active, showing a table with headers **Key**, **Value**, and **Description**. The **Body** tab is also active, showing a **JSON** response:

```

{
  "success": true,
  "message": "Task deleted successfully",
  "tasksArray": [
    {
      "id": 1,
      "title": "Home",
      "descriptions": "I have to clean Home "
    }
  ]
}

```

The response status is **200 OK** with a response time of **14 ms** and a size of **368 B**. The interface includes a sidebar with **Collections** and **Environments**, and a bottom status bar showing system information like time and language.