

# INTERNZVALLEY

## MAJOR PROJECT (Web Development June Batch)

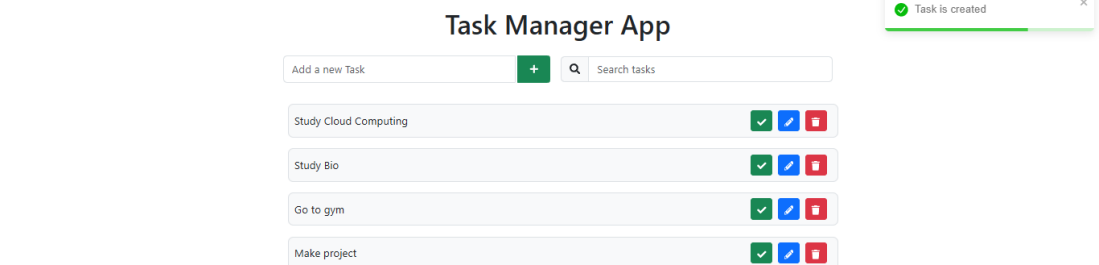
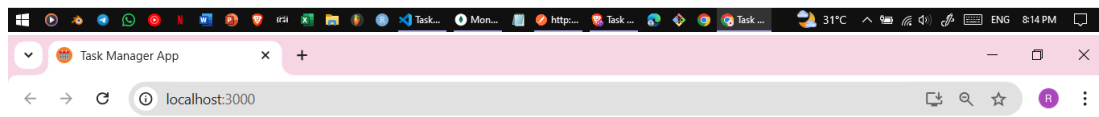
RISHI RAJ

(rishirajskpuram@gmail.com)

### TASK MANAGER APP

#### 1. IMAGES

##### ADD A NEW TASK



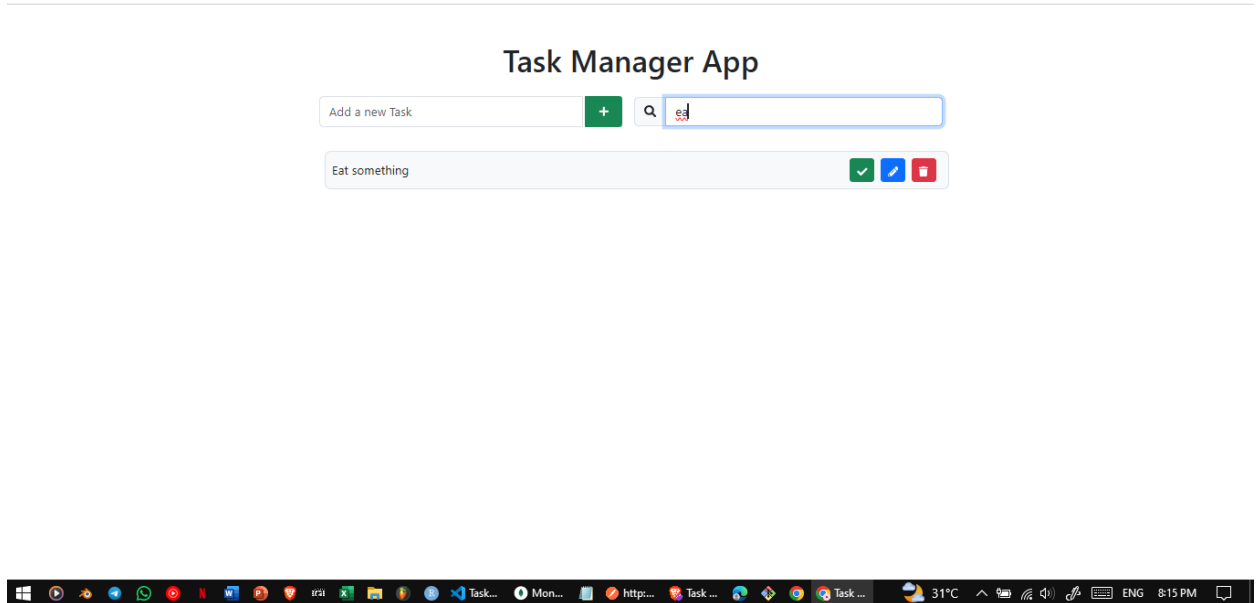
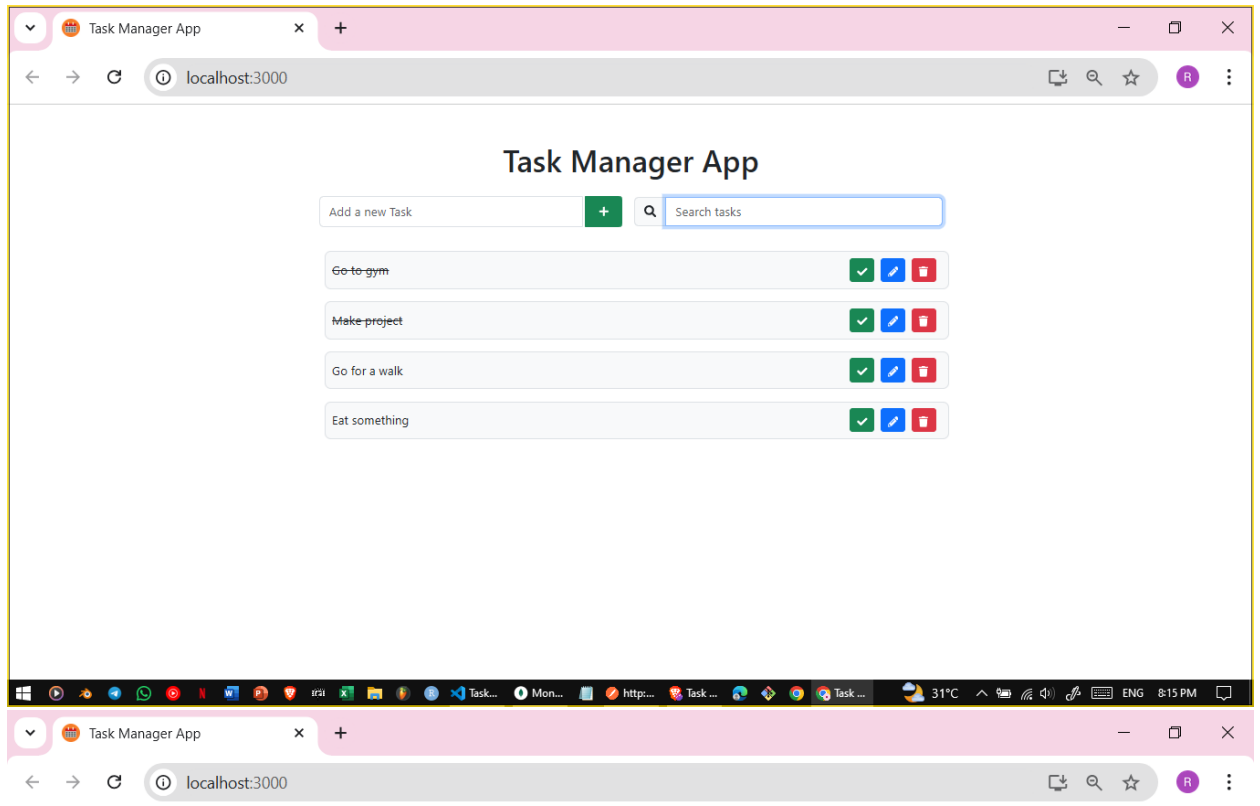
### CHECK ON THE COMPLETED TASKS

The screenshot shows a web browser window with the title 'Task Manager App' and the address bar displaying 'localhost:3000'. The app's header includes a green '+ Add a new Task' button and a search bar labeled 'Search tasks'. Below the header, a list of tasks is displayed, each with a green checkmark icon indicating completion. The tasks are: 'Study Cloud Computing', 'Study Bio', 'Go-to-gym', and 'Make-project'. Each task entry also features a blue pencil icon for editing and a red trash can icon for deletion. A green notification toast in the top right corner reads 'Task Updated'. The Windows taskbar at the bottom shows various application icons and system information, including the date 'Mon...', time '8:14 PM', and temperature '31°C'.

### DELETE A TASK

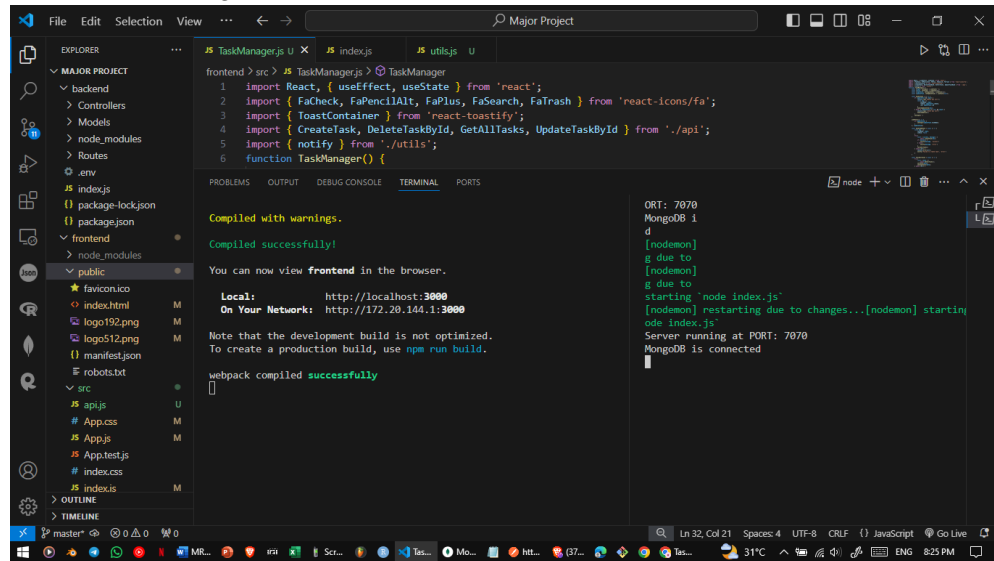
This screenshot shows the 'Task Manager App' after a task has been deleted. The browser window title and address bar remain the same. The task list now only contains 'Go-to-gym' and 'Make-project', both marked with green checkmarks. Two green notification toasts are visible in the top right corner, both reading 'Task is deleted'. The app's interface, including the '+ Add a new Task' button and the search bar, is consistent with the previous view. The Windows taskbar at the bottom shows the same set of application icons and system information, with the time now at '8:15 PM'.

## SEARCH FOR A TASK

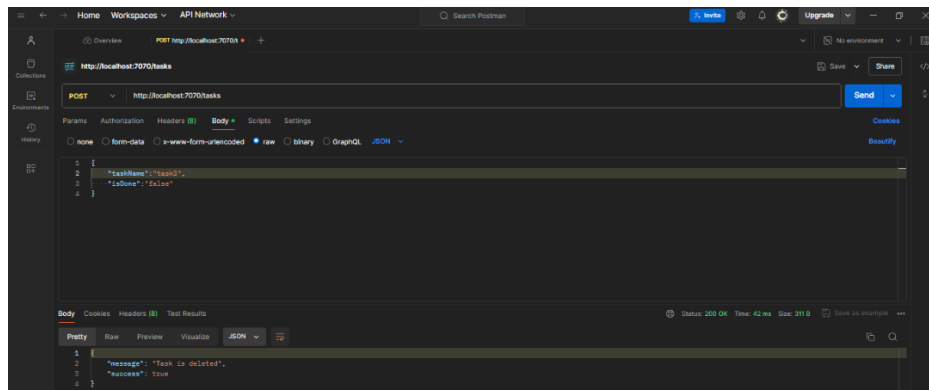
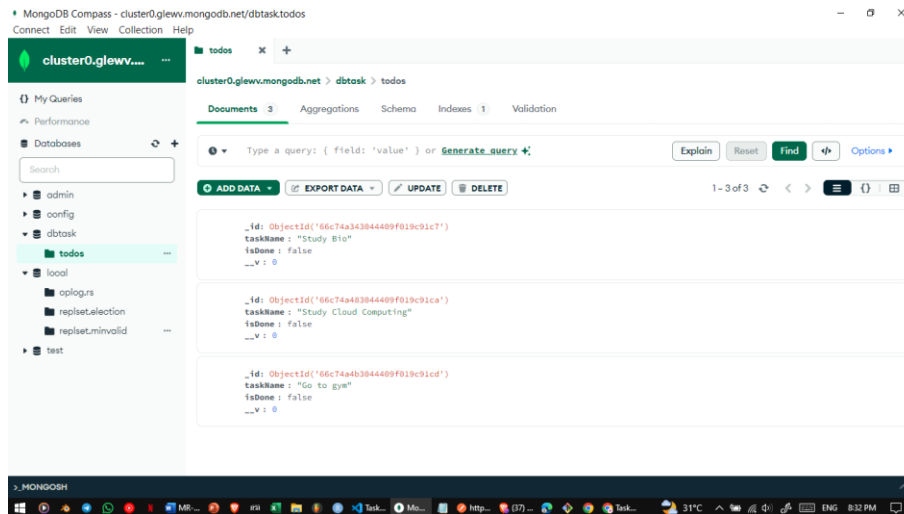


## 2. Source Code:

*Frontend and Backend integration:*



*MongoDB and Postman*



## Backend

Index.js

```
const express = require('express');
const app = express();
require('dotenv').config();
require('./Models/db');
const cors = require('cors');
app.use(cors());

const TaskRouter = require('./Routes/TaskRouter');
const bodyParser = require('body-parser');
const PORT = process.env.PORT || 7070;

app.get('/', (req,res)=>{
  res.send("Hello from rishi's server");
})
app.use(bodyParser.json());
app.use('/tasks',TaskRouter);

app.listen(PORT,()=>{
  console.log(`Server running at PORT: ${PORT}`);
})
```

TaskController.js

```
const express = require('express');
const app = express();
require('dotenv').config();
require('./Models/db');
const cors = require('cors');
app.use(cors());

const TaskRouter = require('./Routes/TaskRouter');
const bodyParser = require('body-parser');
const PORT = process.env.PORT || 7070;

app.get('/', (req,res)=>{
  res.send("Hello from rishi's server");
})
app.use(bodyParser.json());
app.use('/tasks',TaskRouter);

app.listen(PORT,()=>{
```

```
    console.log(`Server running at PORT: ${PORT}`);  
  })
```

Db.js

```
const mongoose = require('mongoose');  
  
const DB_URL = process.env.DB_URL;  
  
mongoose.connect(DB_URL)  
  .then(()=>{  
    console.log("MongoDB is connected");  
  })  
  .catch((err)=>{  
    console.log("MongoDB is not connected", err);  
  })
```

TaskModel.js

```
const mongoose = require('mongoose');  
const Schema = mongoose.Schema;  
  
const TaskSchema = new Schema({  
  taskName: {  
    type: String,  
    required: true  
  },  
  isDone: {  
    type: Boolean,  
    required: true  
  }  
});  
  
const TaskModel = mongoose.model('todos', TaskSchema);  
module.exports = TaskModel;
```

TaskRouter.js

```
const { createTask, fetchAllTasks, updateTaskById, deleteTaskById } =  
require('../Controllers/TaskController');  
  
const router = require('express').Router();  
  
router.get('/', fetchAllTasks);  
  
router.post('/', createTask);
```

```
router.put('/:id', updateTaskById);

router.delete('/:id', deleteTaskById);

module.exports = router;
```

## FRONTEND

Api.js

```
import { API_URL } from "../utils"

export const CreateTask = async (taskObj) => {
  const url = `${API_URL}/tasks`;
  console.log('url ', url)
  const options = {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(taskObj)
  };
  try {
    const result = await fetch(url, options);
    const data = await result.json();
    return data;
  } catch (err) {
    return err;
  }
}

export const GetAllTasks = async () => {
  const url = `${API_URL}/tasks`;
  console.log('url ', url)
  const options = {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json'
    }
  };
  try {
    const result = await fetch(url, options);
    const data = await result.json();
    return data;
  } catch (err) {
    return err;
  }
}
```

```

}

export const DeleteTaskById = async (id) => {
  const url = `${API_URL}/tasks/${id}`;
  console.log('url ', url)
  const options = {
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json'
    }
  };
  try {
    const result = await fetch(url, options);
    const data = await result.json();
    return data;
  } catch (err) {
    return err;
  }
}

export const UpdateTaskById = async (id, reqBody) => {
  const url = `${API_URL}/tasks/${id}`;
  console.log('url ', url)
  const options = {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(reqBody)
  };
  try {
    const result = await fetch(url, options);
    const data = await result.json();
    return data;
  } catch (err) {
    return err;
  }
}

```

App.js

```

import './App.css';
import TaskManager from './TaskManager';

```



```
function App() {
  return (
    <div className="App">
      <TaskManager/>
    </div>
  );
}

export default App;
```

utils.js

```
import { toast } from 'react-toastify';

export const notify = (message, type) => {
  toast[type](message);
}

export const API_URL = 'http://localhost:7070';
```

TaskManager.js

```
import React, { useEffect, useState } from 'react';
import { FaCheck, FaPencilAlt, FaPlus, FaSearch, FaTrash } from 'react-icons/fa';
import { ToastContainer } from 'react-toastify';
import { CreateTask, DeleteTaskById, GetAllTasks, UpdateTaskById } from './api';
import { notify } from './utils';

function TaskManager() {
  const [input, setInput] = useState('');
  const [tasks, setTasks] = useState([]);
  const [copyTasks, setCopyTasks] = useState([]);
  const [updateTask, setUpdateTask] = useState(null);

  const handleTask = () => {
    if (updateTask && input) {
      console.log('update api call');
      const obj = {
        taskName: input,
        isDone: updateTask.isDone,
        _id: updateTask._id
      };
      handleUpdateItem(obj);
    } else if (updateTask === null && input) {
      console.log('create api call')
    }
  };
}
```

```

        handleAddTask();
    }
    setInput('')
}

useEffect(() => {
    if (updateTask) {
        setInput(updateTask.taskName);
    }
}, [updateTask])

const handleAddTask = async () => {
    const obj = {
        taskName: input,
        isDone: false
    }
    try {
        const { success, message } =
            await CreateTask(obj);
        if (success) {
            notify(message, 'success')
        } else {
            notify(message, 'error')
        }
        fetchAllTasks()
    } catch (err) {
        console.error(err);
        notify('Failed to create task', 'error')
    }
}

const fetchAllTasks = async () => {
    try {
        const { data } =
            await GetAllTasks();
        setTasks(data);
        setCopyTasks(data);
    } catch (err) {
        console.error(err);
        notify('Failed to create task', 'error')
    }
}

useEffect(() => {
    fetchAllTasks()
}, [])

```

```
const handleDeleteTask = async (id) => {
  try {
    const { success, message } = await DeleteTaskById(id);
    if (success) {
      notify(message, 'success')
    } else {
      notify(message, 'error')
    }
    fetchAllTasks()
  } catch (err) {
    console.error(err);
    notify('Failed to create task', 'error')
  }
}

const handleCheckAndUncheck = async (item) => {
  const { _id, isDone, taskName } = item;
  const obj = {
    taskName,
    isDone: !isDone
  }
  try {
    const { success, message } = await UpdateTaskById(_id, obj);
    if (success) {
      notify(message, 'success')
    } else {
      notify(message, 'error')
    }
    fetchAllTasks()
  } catch (err) {
    console.error(err);
    notify('Failed to create task', 'error')
  }
}

const handleUpdateItem = async (item) => {
  const { _id, isDone, taskName } = item;
  const obj = {
    taskName,
    isDone: isDone
  }
  try {
    const { success, message } = await UpdateTaskById(_id, obj);
```

```

        if (success) {
            notify(message, 'success')
        } else {
            notify(message, 'error')
        }
        fetchAllTasks()
    } catch (err) {
        console.error(err);
        notify('Failed to create task', 'error')
    }
}

const handleSearch = (e) => {
    const term = e.target.value.toLowerCase();
    const oldTasks = [...copyTasks];
    const results = oldTasks.filter((item) => item.taskName.toLowerCase().includes(term));
    setTasks(results);
}

return (
    <div className='d-flex flex-column align-items-center
w-50 m-auto mt-5'>
        <h1 className='mb-4'>Task Manager App</h1>
        <div className='d-flex justify-content-between
align-items-center mb-4 w-100'>
            <div className='input-group flex-grow-1 me-2'>
                <input type='text'
                    value={input}
                    onChange={
                        (e) => setInput(e.target.value)}
                    className='form-control me-1'
                    placeholder='Add a new Task'
                />
                <button
                    onClick={handleTask}
                    className='btn btn-success btn-sm me-2'
                >
                    <FaPlus className='m-2' />
                </button>
            </div>

            <div className='input-group flex-grow-1'>
                <span
                    className='input-group-text'
                >
                    <FaSearch />

```

```

        </span>
        <input
            onChange={handleSearch}
            className='form-control'
            type='text'
            placeholder='Search tasks'
        />
    </div>
</div>

<div className='d-flex flex-column w-100'>
    {
        tasks.map((item) => (
            <div key={item._id} className='m-2 p-2 border bg-light
w-100 rounded-3 d-flex justify-content-between
align-items-center'>
                <span
                    className={item.isDone ? 'text-decoration-line-through' : ''}
                >{item.taskName}
                </span>

                <div className=''>
                    <button
                        onClick={() => handleCheckAndUncheck(item)}
                        className='btn btn-success
btn-sm me-2'
                        type='button'>
                        <FaCheck />
                    </button>
                    <button
                        onClick={() => setUpdateTask(item)}
                        className='btn btn-primary
btn-sm me-2'
                        type='button'>
                        <FaPencilAlt />
                    </button>
                    <button
                        onClick={() => handleDeleteTask(item._id)}
                        className='btn btn-danger
btn-sm me-2'
                        type='button'>
                        <FaTrash />
                    </button>
                </div>
            </div>
        ))
    }
</div>

```

```
        ))
      }
    </div>

    <ToastContainer
      position='top-right'
      autoClose={3000}
      hideProgressBar={false}
    />
  </div>
)
}

export default TaskManager
```