1. Final Demo Walkthrough

- The final demo is conducted to show the working of the Job Application Tracker project. It helps to prove that the system performs all the required operations correctly and that the objectives of the project are achieved.

the following points are covered:

1. Introduction:

- The project is introduced by mentioning its title and purpose. The Job Application Tracker is designed to help users keep a record of all their job applications in one place.

2. Technology Used:

- The project is developed using Node.js and Express for the backend, with JSON file storage used for saving data locally.

3. Functional Demonstration:

- ✓ Adding a new job entry using a POST request.
- ✓ Viewing all stored job applications using a GET request.
- ✓ Updating the details or status of a job application using a PUT request.
- ✓ Deleting a job entry using a DELETE request.

4. Explanation of Working:

- The flow of the application is explained briefly. When a user sends a request, it is processed by Express, and the data is updated in the JSON file which acts as a local database.
- connecting it with MongoDB or adding a frontend interface.

2. Project Report:

- The project report explains the complete development process of the Job Application Tracker. It contains details about the concept, objectives, system design, implementation, testing, challenges, and results.

The report includes the following sections:

- **Abstract:** A short description of what the project does and why it was created.

- **Introduction:** Explains the problem statement and the need for a job tracking system.

- **Objectives:**
    - To help users maintain job application records.
    - To practice CRUD operations using Node.js.
    - To design a simple and efficient backend.

- **System Design:** Explains how the system works internally. User requests are handled by Express routes and stored in a JSON file.

- **Implementation:** Details the tools, code structure, and main functions with screenshots.

- **Testing and Results:** Shows sample inputs and outputs tested using Postman.

- **Challenges and Solutions:** Lists common issues faced and how they were resolved.

- **Conclusion and Future Scope:** Summarizes what was learned and how the project can be improved further.

- **References:** Mentions all external tutorials and documentation used.

3. Screenshots and API Documentation

- Screenshots are included to visually represent the working of the project. They help to verify the correct execution of each function.

- The screenshots cover:

  - Server running in the terminal.

  - Folder structure of the project.

  - API calls in Postman showing the results of Add, View, Update, and Delete operations.

4. Challenges and Solutions

- This section lists the main difficulties faced during the development process and the methods used to solve them.

- Challenge Solution

- Data not saving properly in JSON file Implemented correct file handling using fs.writeFileSync.
- CORS error between frontend and backend Added and configured the CORS middleware in Express.
- Unique ID management for each job Used Date.now() to generate unique IDs for all job entries.
- Deployment issues Studied Render setup and corrected the start script in package.json.

5. GitHub Repository and README

A GitHub repository is created to maintain version control and make the project accessible online. The README file gives complete information about how to run and test the project.

README Content:

- Project Title: Job Application Tracker

- Overview: A simple Node.js project that allows users to track job applications.

- Features:
    - Add, view, update, and delete job details.
    - Data stored locally in JSON format.
    - Ready for future database integration.

- 4. Technologies Used:
    - Node.js, Express.js, File System, and CORS.
- 5. Installation Steps:
    - Clone the repository.
    - Run npm install.
    - Start the server using npm start.
    - Access at http://localhost:5000/api/jobs.
- 6. API Endpoints: Listed clearly with method and purpose.
- 7. Future Enhancements:
    - Integrate MongoDB.
    - Add login system.
    - Develop a React-based frontend.

6. Final Submission Checklist

- Before submitting the final project, the following items must be completed and verified:

  1. Source code uploaded to GitHub with proper README.

  2. Project report prepared in PDF format.

  3. Screenshots and API documentation included in a separate folder.

  4. Short demo video recorded or deployed link provided.

  5. All files organized neatly in a single folder.

Code:

data/jobs.json

```
[
  {
    "id": 1,
    "company": "Google",
    "position": "Software Engineer",
    "status": "Applied",
    "appliedDate": "2025-10-24",
    "notes": "Submitted via company portal"
  }
]
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Simple Job Tracker</title>
<style>
body { font-family: Arial; background:#f0f2f5; padding:20px; }
h1 { text-align:center; color:#2b5797; }
form { background:white; padding:15px; border-radius:10px; margin-
bottom:20px; box-shadow:0 2px 5px rgba(0,0,0,0.1); }
input, select, textarea { width:100%; padding:8px; margin:5px 0; border-
radius:5px; border:1px solid #ccc; box-sizing:border-box; }
button { padding:8px 12px; border:none; border-radius:5px;
background:#2b5797; color:white; cursor:pointer; margin-top:5px; }
button:hover { background:#1e3c72; }
.job { background:white; padding:10px; margin:10px 0; border-radius:8px;
box-shadow:0 2px 4px rgba(0,0,0,0.1); }
.job button { background:#555; margin-top:5px; padding:4px 8px; color:white;
border:none; border-radius:4px; cursor:pointer; }
.job button:hover { background:#333; }
</style>
</head>
<body>

<h1>Simple Job Tracker</h1>
```

```
<form id="jobForm">
  <input type="text" id="company" placeholder="Company" required>
  <input type="text" id="position" placeholder="Position" required>
  <select id="status">
    <option>Applied</option>
    <option>Interview</option>
    <option>Selected</option>
    <option>Rejected</option>
  </select>
  <input type="text" id="recruiter" placeholder="Recruiter / Email">
  <textarea id="notes" placeholder="Notes"></textarea>
  <button type="submit">Add Job</button>
</form>

<input type="text" id="searchInput" placeholder="Search by company...">

<div id="jobList"></div>

<script>
const company = document.getElementById("company");
const position = document.getElementById("position");
const status = document.getElementById("status");
const recruiter = document.getElementById("recruiter");
const notes = document.getElementById("notes");
const form = document.getElementById("jobForm");
const searchInput = document.getElementById("searchInput");
const jobList = document.getElementById("jobList");
```

```
// Initialize jobs
let jobs = JSON.parse(localStorage.getItem("jobs") || "[]");

// Save jobs to localStorage
function saveJobs() {
  localStorage.setItem("jobs", JSON.stringify(jobs));
}

// Render jobs
function renderJobs() {
  const search = searchInput.value.toLowerCase();
  jobList.innerHTML = "";
  jobs.filter(j => j.company.toLowerCase().includes(search))
      .forEach(j => {
        const div = document.createElement("div");
        div.className = "job";
        div.innerHTML = `
          <strong>${j.company}</strong> — ${j.position} <br>
          Status: ${j.status}<br>
          ${j.recruiter ? "Recruiter: " + j.recruiter + "<br>" : ""}
          ${j.notes ? "Notes: " + j.notes + "<br>" : ""}
        `;
        const editBtn = document.createElement("button");
        editBtn.textContent = "Edit";
        editBtn.addEventListener("click", () => editJob(j.id));
        const delBtn = document.createElement("button");
        delBtn.textContent = "Delete";
        delBtn.addEventListener("click", () => deleteJob(j.id));
        div.appendChild(editBtn);
        div.appendChild(delBtn);
        jobList.appendChild(div);
      });
}

// Add job
form.addEventListener("submit", e => {
  e.preventDefault();
  const newJob = {
    id: Date.now(),
    company: company.value,
    position: position.value,
    status: status.value,
    recruiter: recruiter.value,
    notes: notes.value
  };
  jobs.push(newJob);
  saveJobs();
  form.reset();
  renderJobs();
});

// Delete job
function deleteJob(id) {
  jobs = jobs.filter(j => j.id !== id);
  saveJobs();
  renderJobs();
```

```
}

// Edit job
function editJob(id) {
  const job = jobs.find(j => j.id === id);
  if (!job) return;
  company.value = job.company;
  position.value = job.position;
  status.value = job.status;
  recruiter.value = job.recruiter;
  notes.value = job.notes;
  jobs = jobs.filter(j => j.id !== id); // remove old entry
}

// Search filter
searchInput.addEventListener("input", renderJobs);

// Initial render
renderJobs();
</script>

</body>
</html>
```

server.js

```
import express from "express";
import fs from "fs";
import cors from "cors";

const app = express();
app.use(cors());
app.use(express.json());

// File path
const FILE = "./jobs.json";

// Get all jobs
app.get("/jobs", (req, res) => {
  const data = JSON.parse(fs.readFileSync(FILE, "utf8"));
  res.json(data);
});

// Add a new job
app.post("/jobs", (req, res) => {
  const data = JSON.parse(fs.readFileSync(FILE, "utf8"));
  const newJob = { id: Date.now(), ...req.body };
  data.push(newJob);
  fs.writeFileSync(FILE, JSON.stringify(data, null, 2));
  res.json(newJob);
});

// Delete a job
app.delete("/jobs/:id", (req, res) => {
  let data = JSON.parse(fs.readFileSync(FILE, "utf8"));
  data = data.filter((j) => j.id != req.params.id);
  fs.writeFileSync(FILE, JSON.stringify(data, null, 2));
  res.json({ message: "Deleted" });
});

app.listen(5000, () => console.log("Server running on port 5000"));
```

JSON CRUD

controllers/jobController.js

```
import fs from "fs";
const filePath = "./data/jobs.json";

// Helper: read data
const readData = () => {
  const data = fs.readFileSync(filePath, "utf8");
  return JSON.parse(data);
};

// Helper: write data
const writeData = (data) => {
  fs.writeFileSync(filePath, JSON.stringify(data, null, 2));
};

// GET all jobs
export const getJobs = (req, res) => {
  const jobs = readData();
  res.json(jobs);
};

// POST new job
export const createJob = (req, res) => {
  const jobs = readData();
  const newJob = { id: Date.now(), ...req.body };
  jobs.push(newJob);
  writeData(jobs);
  res.json(newJob);
};
```

```
// PUT update job
export const updateJob = (req, res) => {
  const jobs = readData();
  const index = jobs.findIndex((j) => j.id == req.params.id);

  if (index === -1) return res.status(404).json({ message: "Job not
found" });

  jobs[index] = { ...jobs[index], ...req.body };
  writeData(jobs);
  res.json(jobs[index]);
};

// DELETE job
export const deleteJob = (req, res) => {
  const jobs = readData();
  const filtered = jobs.filter((j) => j.id != req.params.id);
  writeData(filtered);
  res.json({ message: "Job deleted" });
};
```

Run the Backend

```
npm start
```

Now open your browser or Postman and test:

GET http://localhost:5000/api/jobs

POST http://localhost:5000/api/jobs with JSON body:

```
{
  "company": "Amazon",
  "position": "Data Analyst",
  "status": "Applied",
  "notes": "Referred by senior"
}
```

PUT http://localhost:5000/api/jobs/:id

DELETE http://localhost:5000/api/jobs/:id

Repository Links:

https://github.com/subhashinichellachamy-byte

https://github.com/akilasasi1291-tech

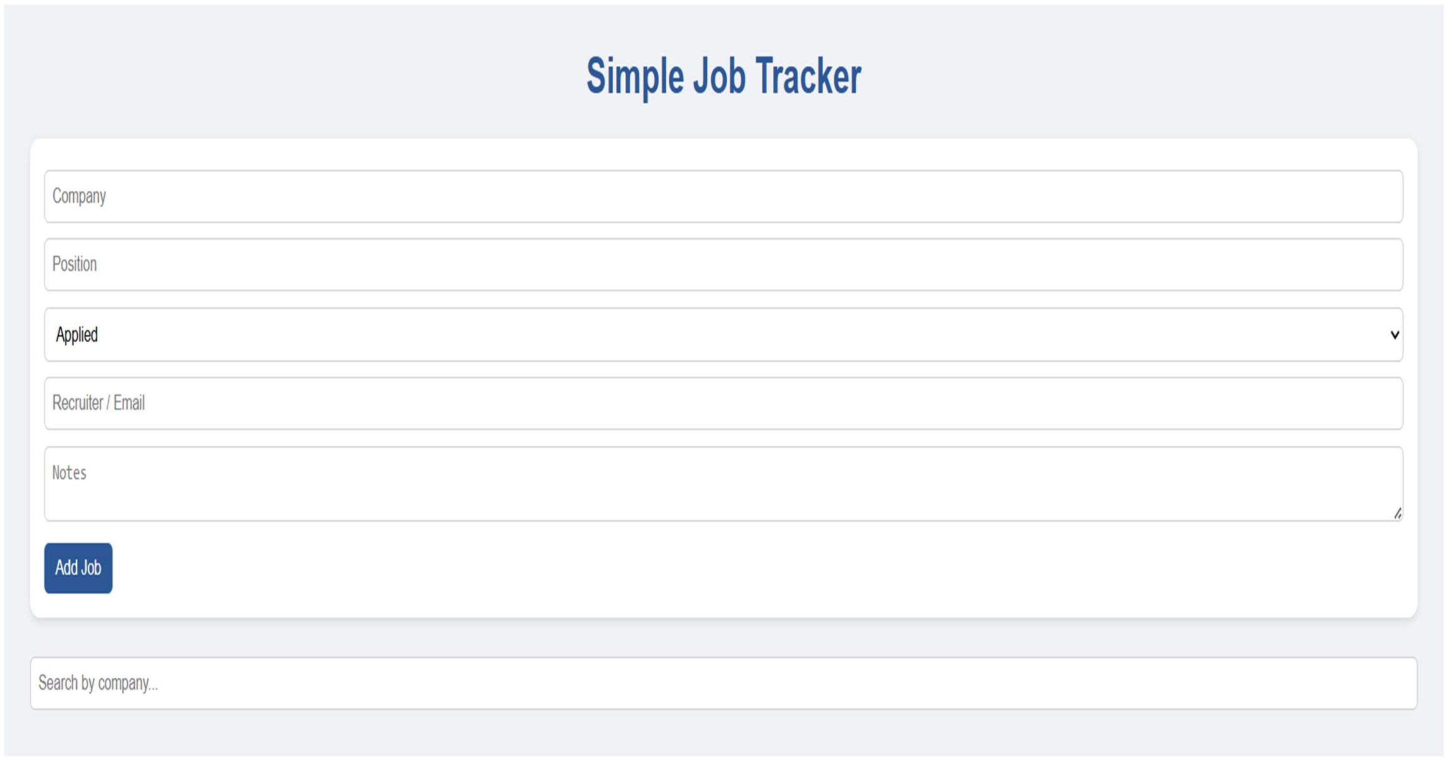
https://github.com/ayyajyoti2006-stack

https://github.com/GayathriBalamurgan7

https://github.com/alienshaaa

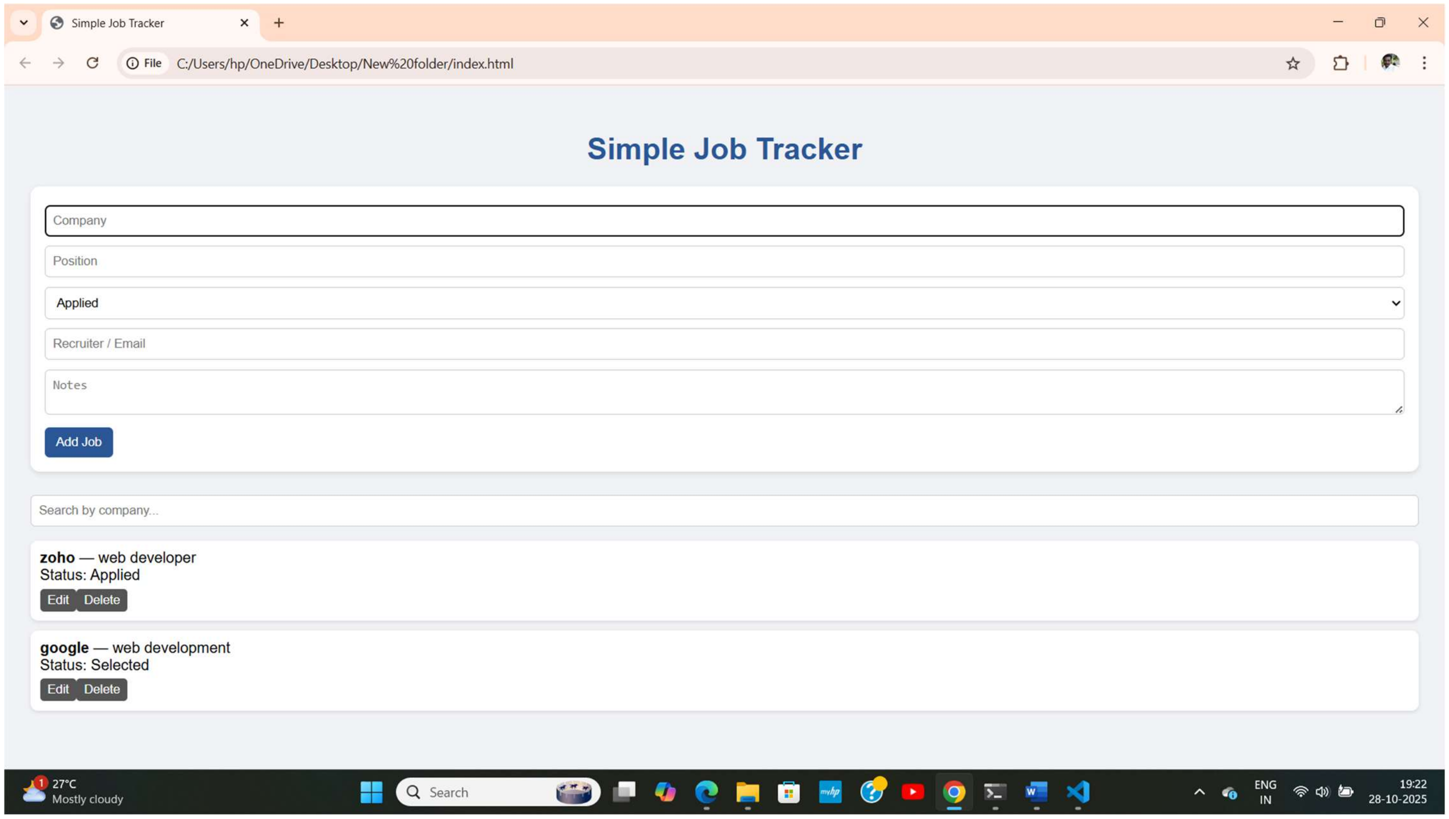
Output screenshot:

Home page:



Simple Job Tracker

Add job page :


Simple Job Tracker

C:/Users/hp/OneDrive/Desktop/New%20folder/index.html

Simple Job Tracker

Company

Position

Applied

Recruiter / Email

Notes

Add Job

Search by company...

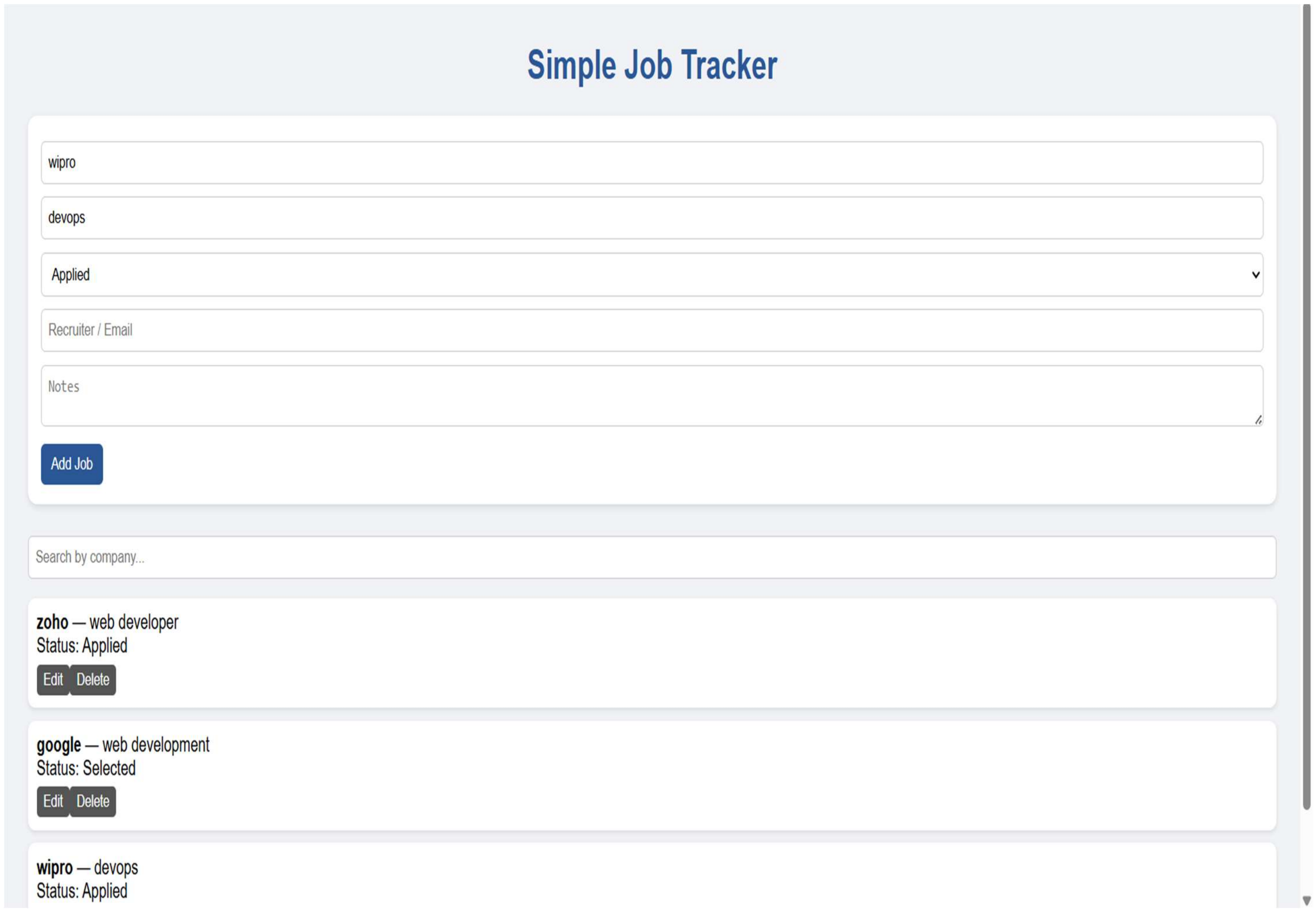
zoho — web developer
Status: Applied
Edit Delete

google — web development
Status: Selected
Edit Delete

27°C
Mostly cloudy

19:22
28-10-2025

Job list:



Simple Job Tracker

wipro

devops

Applied

Recruiter / Email

Notes

Add Job

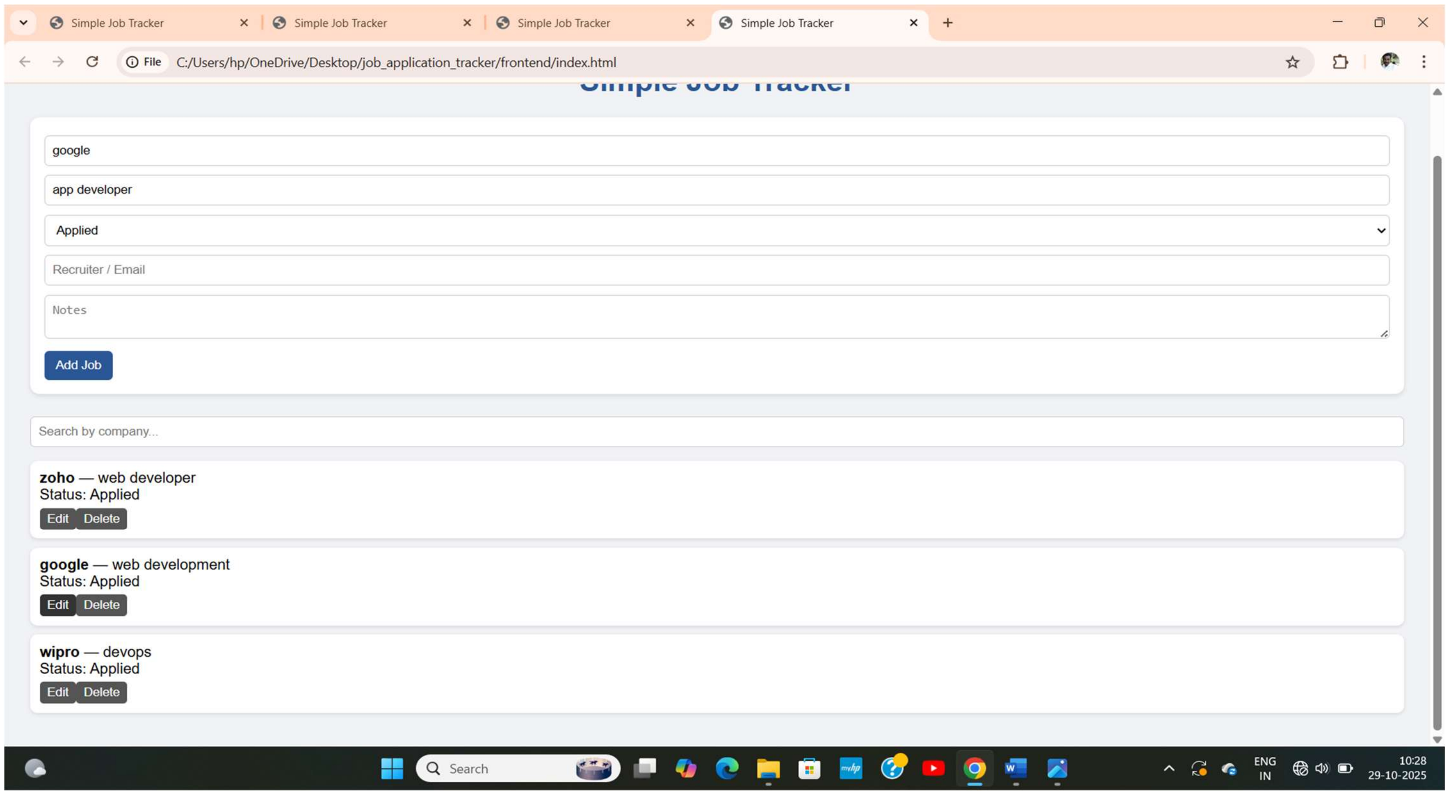
Search by company...

zoho — web developer
Status: Applied
Edit Delete

google — web development
Status: Selected
Edit Delete

wipro — devops
Status: Applied

Update job :



Simple Job Tracker
C:/Users/hp/OneDrive/Desktop/job_application_tracker/frontend/index.html

google

app developer

Applied

Recruiter / Email

Notes

Add Job

Search by company...

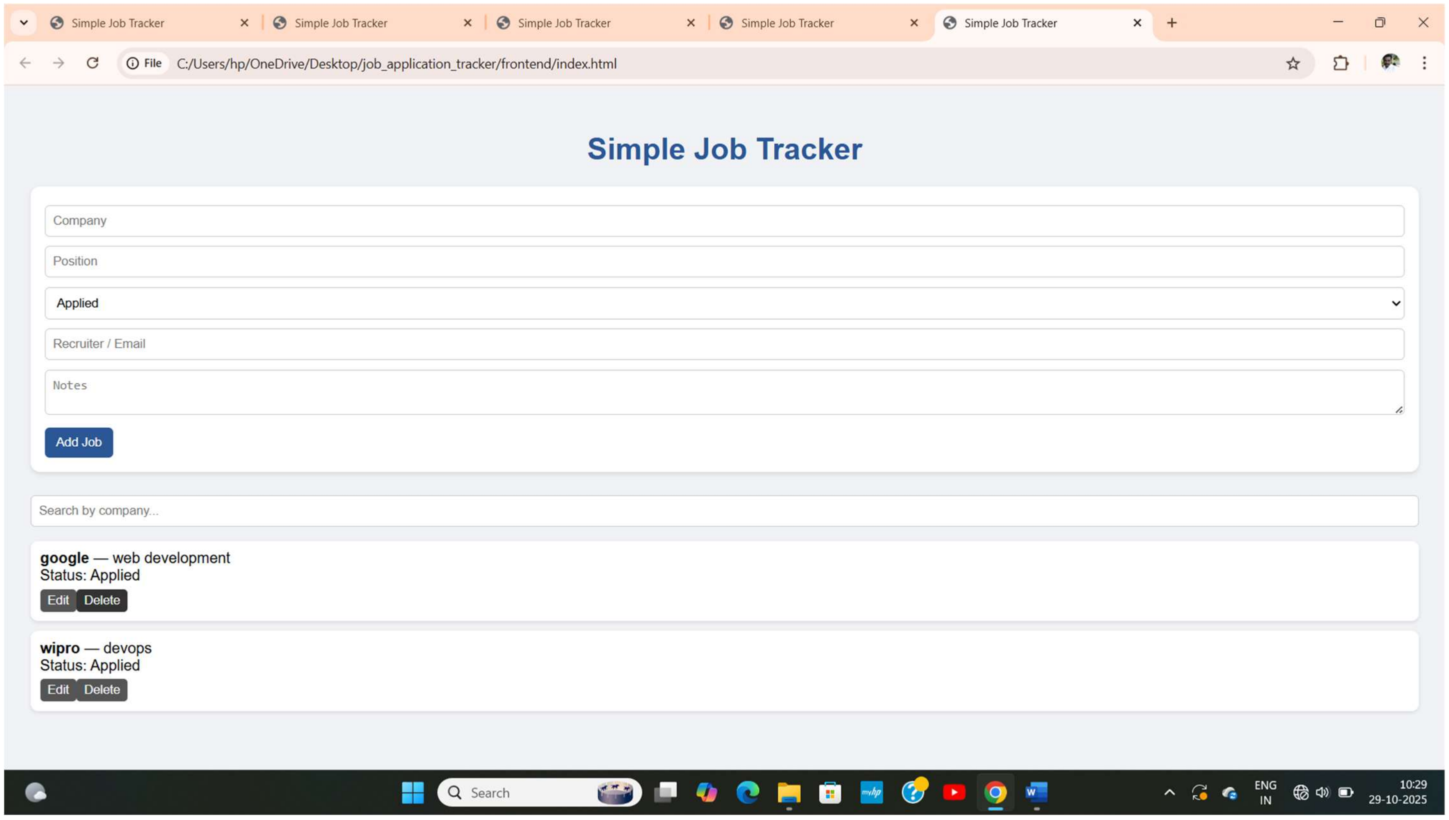
zoho — web developer
Status: Applied
Edit Delete

google — web development
Status: Applied
Edit Delete

wipro — devops
Status: Applied
Edit Delete

Search
ENG
IN
10:28
29-10-2025

Delete job:



Simple Job Tracker

google — web development
Status: Applied

wipro — devops
Status: Applied