**Folder Structure**

bash

CopyEdit

time-tracker-app/

│

├── client/ # React Frontend

│ ├── public/

│ └── src/

│ ├── components/ # Reusable UI (Timer, Graph, TaskList)

│ ├── pages/ # Home, Login, Dashboard

│ ├── utils/ # API calls, Auth helpers

│ ├── App.jsx

│ └── index.js

│ ├── Dockerfile

│ └── .env

├── server/ # Node.js Backend

│ ├── models/ # Mongoose Models

│ ├── routes/ # Express Routes

│ ├── controllers/ # Route Logic

│ ├── middleware/ # Auth, Error handlers

│ ├── config/ # DB connection, JWT utils

│ ├── server.js

│ ├── Dockerfile

│ └── .env

├── docker-compose.yml # For local container orchestration

├── README.md

├── .gitignore

└── .github/

└── workflows/ # GitHub Actions CI/CD

├── frontend.yml

└── backend.yml

**📘 Project Report (Resume / README Ready)**

**🔹 Project Title:**

**Remote Work Time Tracker & Project Logger**

**🔹 Objective:**

To build a full-stack time tracking web application for remote workers and developers, enabling task management, productivity tracking, and automated weekly reporting with end-to-end DevOps integration.

**🔹 Features:**

* 🕒 Live time tracking per project (daily reset, cumulative sum)
* ✅ Optional task list for each project (e.g., "Build UI", "Setup DB")
* 📊 Weekly graphical productivity report (Bar/Line chart)
* 🧠 Per-project report on dashboard with time stats
* 🔐 Authentication (Register/Login)
* 📂 Clean UI (React), REST API backend (Node.js + Express + MongoDB)

**🔹 Tech Stack:**

* **Frontend**: React.js, Chart.js, Axios, TailwindCSS
* **Backend**: Node.js, Express, MongoDB, JWT Auth
* **DevOps**:
  + GitHub Actions (CI/CD)
  + Docker + Docker Compose
  + Deployment: Render (Backend), Vercel/Netlify (Frontend)
  + MongoDB Atlas (Cloud DB)

**🔹 DevOps Flow:**

1. Dockerized frontend and backend with docker-compose for local testing.
2. CI/CD pipeline via GitHub Actions for both frontend and backend:
   * Lint, Build, Test, and Deploy
3. Secrets managed via GitHub Actions
4. Free-tier deployment with live demo access

**🔹 Folder Structure:**

Organized as client/, server/, docker-compose.yml, and GitHub Actions workflows for clean development, collaboration, and testing.

**🔹 Deployment:**

* **Frontend**: [Vercel/Netlify URL]
* **Backend**: [Render/Railway URL]
* **Database**: MongoDB Atlas