

# SDE I/II - Take Home Assignment

## Context

We work with NGOs across India to help them track and report the impact of their work. Your task is to build a scalable and maintainable web app that lets NGOs **submit monthly reports** (individually or in bulk) and lets an admin **view a dashboard** that summarizes and tracks the data.

As we scale, we're particularly interested in **systems that can handle larger volumes of input**, partial failures, and background processing.

---

## What You'll Build

### Frontend

Use any modern frontend framework (React, Next.js, Vue, etc.)

Pages:

1. **Report Submission Form**
    - Fields: NGO ID, Month, People Helped, Events Conducted, Funds Utilized
    - Submits data to backend
  2. **Bulk Report Upload**
    - Upload a CSV file containing multiple monthly reports
    - Submits file to backend and returns a Job ID
    - After upload, shows a job progress UI (e.g. "Processed 35 of 50 rows")
  3. **Admin Dashboard**
    - View total numbers for a selected month:
      - Total NGOs Reporting
      - Total People Helped
      - Total Events Conducted
      - Total Funds Utilized
    - Optional: add month selector, filters, or region support
- 

### Backend

Use any backend stack (Node.js, Express, Flask, Django, etc.)

## Endpoints:

- `POST /report` – submit a single report
- `POST /reports/upload` – submit a CSV file of reports (processed asynchronously)
- `GET /job-status/{job_id}` – return job processing status
- `GET /dashboard?month=YYYY-MM` – returns aggregated data for the selected month

## Database:

- Use a persistent database (e.g., PostgreSQL, MongoDB, SQLite)
  - Ensure reports from the same NGO/month are not double-counted (idempotency)
- 

## Requirements

- Keep the UI simple and functional
  - Handle basic validation and errors
  - Do not process the entire CSV synchronously — the backend should process uploads in the background
  - Provide status updates (via polling or UI) during processing
  - Handle partial failures — not all rows may be valid
  - Include instructions in a README for setup and running
  - You're welcome to use AI tools (e.g., Loveable, Polymet, Cursor, GitHub Copilot) if it helps — feel free to mention where they were useful in your writeup
  - Deployed demo (Render, Vercel, Netlify)
- 

## Bonus (Optional)

- Retry logic for failed rows in CSV uploads
  - Filtering or pagination in the dashboard
  - Use of component libraries (e.g., MUI, Chakra)
  - Admin authentication for dashboard routes
  - OpenAPI/Swagger spec or Postman collection
  - Docker setup or CI/CD pipeline
  - Use of metrics, structured logging, or observability tools
- 

## Deliverables

1. Link to GitHub repo (or zip)

2. Short README with:
    - Tech stack
    - Setup instructions
    - API sample usage or screenshots of UI
    - Deployed link or demo recording of the application
  3. Short writeup or video:
    - Your approach and architectural decisions
    - Where you used AI tools (if applicable)
    - What you'd improve with more time or in a production-grade system
- 

## How to submit

Fill out the [Google Form Link](#) with the details.

---

## Time Expectation

This challenge is designed to be completed in a few focused sessions. Please don't worry about polish — we're more interested in your thinking and execution.