Full Monorepo Setup (Server + Client)

1. Organize your folder structure

/social-network-project
  /client   → React + Vite app
  /server   → Express + MongoDB app
  package.json → Root workspace controller

2. Root `package.json` setup

At the root (`/social-network-project/package.json`), set up like this:

```
{
  "name": "social-network-project",
  "private": true,
  "scripts": {
    "install-all": "npm --prefix server install && npm --prefix client install",
    "dev": "concurrently \"npm run server\" \"npm run client\"",
    "server": "npm --prefix server run dev",
    "client": "npm --prefix client run dev",
    "build": "npm --prefix client run build && npm --prefix server run build"
  },
  "devDependencies": {
    "concurrently": "^8.0.1"
  }
}
```

Notice:

- `"dev"` runs **both** the server and client in one terminal window!
- `"build"` builds **both** the React app and backend (if needed).
- `"concurrently"` lets you run multiple npm scripts side by side.

3. Install `concurrently` at the root

npm install -D concurrently

4. Client scripts (`/client/package.json`)

In `/client/package.json`, make sure you have:
"scripts": {

"dev": "vite",

"build": "vite build",

"preview": "vite preview"

}

5. Server scripts (`/server/package.json`)

In `/server/package.json`, make sure you have something like:
"scripts": {

"dev": "nodemon server.js",

"start": "node server.js",

"build": "echo 'no build step for server'"

}

Notice:

- `"dev"` uses `nodemon` for auto-reload.
- `"build"` is optional for now — your server probably doesn't need a build step unless you're bundling it.

Install nodemon if you haven't:

npm install -D nodemon

Or install globally:

npm install -g nodemon

6. How to Use It

From the **root directory**:

npm run install-all: Installs server and client deps

npm run dev: Starts both backend and frontend

npm run build: Builds client (and server if needed)

✔️ No more cd-ing into folders.

✔️ One simple root command to run the whole project.

⚡ Bonus Tip: Proxy Frontend Requests to Backend

To avoid CORS problems during development, you should **proxy API requests**.

In `/client/vite.config.js`:

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  server: {
    port: 5173,
    proxy: {
      '/api': 'http://localhost:3001'
    }
  }
});
```

This way:

- Frontend code like `axios.get('/api/users')` **automatically talks** to `localhost:3001`.
- No ugly CORS errors.
- No hardcoding `localhost:3001` into frontend code.

Final Workflow:

npm install

npm run install-all

npm run dev