I am attaching the compressed version of code of front end, backend, langflow exported app and the prompt we are using. Please refer to that parallalely

---

🩺 St Lukes | Doctor's App POC

A full-stack Proof of Concept (POC) named `doctor-app` (frontend) and `express-server` (backend), integrating a React-based rich text editor (Tiptap) with an Express.js proxy server that communicates with the Langflow API to extract and visualize medical tasks.

🚀 Getting Started

Prerequisites

- Node.js (v18+)
- npm or yarn
- Access to Langflow API (with token)

🖥️ Frontend Setup (`doctor-app`)

React app powered by Vite + MUI + Tiptap for doctor input interface.

Installation

```shell
cd doctor-app
npm install
```

Run Dev Server

```shell
npm run dev
```

Features

- Rich text editor (Tiptap)
- Automatic input processing (debounced)
- Sends input to Langflow via backend proxy
- Highlights medical terms (e.g. meds, dosages)
- Displays extracted tasks via `MedicalDataRenderer`

🌐 Backend Setup (`express-server`)

An Express.js server acting as a proxy to the Langflow API.

Installation

```shell
cd express-server
npm install
```

Start Server

```shell
node server.js
```

Endpoint

```unset
POST /api/proxy
```

- Accepts: `{ input_value: "..." }`
- Forwards to Langflow API
- Returns: processed output from Langflow

Notes

- CORS enabled
- Error handling for Langflow API failures
- Replace `Authorization` token and API URL with your own if needed

📡 Langflow API Details

```
Unset
POST
https://api.langflow.astra.datastax.com/lf/<workspace-id>/api/v1/
run/<flow-id>
```

Headers:

```
JSON
{
  "Authorization": "Bearer <your-token>",
  "Content-Type": "application/json"
}
```

Payload:

```
JSON
{
```

```
  "input_value": "doctor note text",
  "output_type": "chat",
  "input_type": "chat"
}
```

🔍 Highlights

- **Debounced Input:** Waits 2s after typing stops before API call
- **Dynamic Highlights:** Terms from Langflow response are highlighted in real-time
- **Extensible Design:** Easily swap Langflow with other LLMs (e.g. Ollama, OpenAI)

📌 Dev Flow

1. Start backend: `cd express-server && node server.js`
2. Start frontend: `cd doctor-app && npm run dev`
3. Navigate to `http://localhost:5173`
4. Type a doctor's note (e.g. “Patient prescribed 50mg Aspirin daily”)
5. See tasks extracted and highlights applied automatically

Langflow Cloud (Datastax Astra Integration)

🔗 Hosted Version: https://langflow.datastax.com

Steps:

1. **Sign in** with Datastax or GitHub
2. **Create or import a flow**
3. **Click "Deploy" or "Run via API"**
4. Use the provided HTTP endpoint with an API Key

Example API Request:

```
POST
https://api.langflow.astra.datastax.com/lf/<workspace-id>/api/v1/run/<
flow-id>
```

Headers:

```
{
  "Authorization": "Bearer <your-Astra-API-token>",
  "Content-Type": "application/json"
}
```

Body

```
{
  "input_value": "Patient has a fever and needs 500mg paracetamol.",
  "input_type": "chat",
  "output_type": "chat"
}
```