High-Level Design (HLD) and Low-Level Design (LLD) for Drag-and-Drop Workflow UI for LLM Applications

High-Level Design (HLD)

Overview

The application is a web-based platform enabling users to create and deploy workflows involving Large Language Models (LLMs). Users can visually construct workflows with three primary nodes: Input, LLM, and Output. The workflows can be executed or deployed as standalone applications with an interactive chat interface.

Architecture Diagram

- User Interface (React) --> React Flow (Workflow Canvas)
- |
- v
- Node Components (Input, LLM, Output) --> State Management (React Context)
- |
- v
- OpenAI API Integration
- |
- v

Workflow Execution & Deployment

Functional Components

1. **Workflow UI**
    - **Drag-and-Drop Functionality**: Users can drag and drop nodes onto a canvas.
    - **Node Connections**: Allow connecting Input, LLM, and Output nodes in sequence.
2. **Node Functionality**
    - **Input Node**: Accepts user queries and validates inputs.
    - **LLM Node**: Configures LLM credentials and parameters.
    - **Output Node**: Displays the output from the LLM.
3. **Workflow Execution**
    - A "Run" button executes the workflow.

    - Errors during execution are handled gracefully.
4. **Deployment** (Optional Feature)
    - A "Deploy" button deploys the workflow.
    - A chat interface enables interaction with the deployed workflow.
    - "Undeploy" functionality for workflow removal.

Technology Stack

- **Frontend**: React with React Flow for the workflow interface.
- **API**: OpenAI API (or LLaMA) for LLM interaction.
- **State Management**: React Context.
- **Styling**: Tailwind CSS.
- **Deployment**: Optional integration with a backend or serverless architecture.

Key Features

- User-friendly drag-and-drop interface.
- Validation at each node to ensure proper connections and configurations.
- Error handling and informative feedback.
- Optional deployment with an interactive chat interface.

Low-Level Design (LLD)

Folder Structure

- openagi-workflow/
- ├── src/
- │ ├── components/
- │ │ ├── nodes/
- │ │ │ ├── InputNode.tsx
- │ │ │ ├── LLMNode.tsx
- │ │ │ └── OutputNode.tsx
- │ │ ├── ChatWindow.tsx
- │ │ └── Sidebar.tsx
- │ ├── hooks/
- │ │ └── useChat.ts
- │ ├── store/
- │ │ └── index.ts
- │ ├── types/
- │ │ └── index.ts
- │ ├── utils/
- │ │ └── openai.ts

- │   ├── App.tsx
- │   ├── main.tsx
- │   └── index.css
- ├── index.html
- ├── package.json
- ├── postcss.config.js
- ├── tailwind.config.js

└── tsconfig.json

Components

InputNode.tsx

- Accepts user input queries.
- Validates inputs to ensure compatibility with LLMs.

LLMNode.tsx

- Provides a configuration interface for specifying OpenAI credentials and parameters (e.g., model type, temperature, max tokens).
- Validates configurations before enabling connections to Output nodes.

OutputNode.tsx

- Displays the output generated by the LLM based on the input and configuration.

ChatWindow.tsx

- Interactive chat interface for deployed workflows.

Sidebar.tsx

- Contains drag-and-drop options for Input, LLM, and Output nodes.
- Provides workflow control buttons (Run, Deploy, Undeploy).

Hooks

useChat.ts

- Custom hook for managing chat state and interaction with the OpenAI API.

State Management

index.ts (React Context)

- Manages global state for:
    - Workflow nodes and connections.
    - Execution status.
    - Deployment status.

Utilities

openai.ts

- Wrapper for interacting with the OpenAI API.
- Handles request validation and error handling.

Main Application

App.tsx

- Entry point for the React application.
- Sets up React Flow and global state.

main.tsx

- Initializes the React app and renders the root component.

index.css

- Global styles using Tailwind CSS.

Workflow Execution Logic

1. **Node Validation**:
    - Ensure each node is correctly configured before execution.
2. **Data Flow**:
    - Input data flows to the LLM node, which processes it and sends the output to the Output node.
3. **Error Handling**:
    - Catch errors during execution and provide feedback via a notification system.

Deployment Workflow

1. **Deploy Button**:
    - Sends the workflow configuration to a backend for deployment.
2. **Chat Interface**:
    - Provides a chat window for interacting with the deployed workflow.
3. **Undeploy**:
    - Removes the deployed workflow from the server/backend.

Key Design Considerations

Scalability

- Use React Flow to manage complex node interactions and layouts efficiently.

User Experience

- Intuitive drag-and-drop functionality.
- Real-time validation and feedback.

Security

- Secure API key storage and usage for LLM configurations.
- Input sanitization to prevent vulnerabilities.

Extensibility

- Additional nodes (e.g., data processing nodes) can be added easily.
- Support for multiple LLMs by extending the LLM node functionality.

Performance

- Optimize node rendering and data flow to handle large workflows seamlessly.