src\App.jsx

```jsx
import { PipelineToolbar } from './toolbar';
import { PipelineUI } from './ui';
import { SubmitButton } from './submit';

function App() {
  return (
    <div>
      <PipelineToolbar />
      <PipelineUI />
      <SubmitButton />
    </div>
  );
}

export default App;

```

src\toolbar.jsx

```jsx
// toolbar.js

import { DraggableNode } from './draggableNode';

export const PipelineToolbar = () => {

    return (
        <div style={{ padding: '10px' }}>
            <div style={{ marginTop: '20px', display: 'flex', flexWrap: 'wrap', gap: '10px' }}>
                <DraggableNode type='customInput' label='Input' />
                <DraggableNode type='llm' label='LLM' />
                <DraggableNode type='customOutput' label='Output' />
                <DraggableNode type='text' label='Text' />
            </div>
        </div>
    );
};

```

src\ui.jsx

```
// ui.js
// Displays the drag-and-drop UI
// --------------------------------------------------

import { useState, useRef, useCallback } from 'react';
import ReactFlow, { Controls, Background, MiniMap } from 'reactflow';
import { useStore } from './store';
import { shallow } from 'zustand/shallow';
import { InputNode } from './nodes/inputNode';
import { LLMNode } from './nodes/llmNode';
import { OutputNode } from './nodes/outputNode';
import { TextNode } from './nodes/textNode';

import 'reactflow/dist/style.css';

const gridSize = 20;
const proOptions = { hideAttribution: true };
const nodeTypes = {
  customInput: InputNode,
  llm: LLMNode,
  customOutput: OutputNode,
  text: TextNode,
};

const selector = (state) => ({
  nodes: state.nodes,
  edges: state.edges,
  getNodeID: state.getNodeID,
  addNode: state.addNode,
  onNodesChange: state.onNodesChange,
  onEdgesChange: state.onEdgesChange,
  onConnect: state.onConnect,
});

export const PipelineUI = () => {
    const reactFlowWrapper = useRef(null);
    const [reactFlowInstance, setReactFlowInstance] = useState(null);
    const {
      nodes,
      edges,
      getNodeID,
      addNode,
      onNodesChange,
      onEdgesChange,
      onConnect
    } = useStore(selector, shallow);

    const getInitNodeData = (nodeID, type) => {
      let nodeData = { id: nodeID, nodeType: `${type}` };
      return nodeData;
    }

    const onDrop = useCallback(
        (event) => {
          event.preventDefault();

          const reactFlowBounds = reactFlowWrapper.current.getBoundingClientRect();
          if (event?.dataTransfer?.getData('application/reactflow')) {
            const appData = JSON.parse(event.dataTransfer.getData('application/reactflow'));
            const type = appData?.nodeType;

            // check if the dropped element is valid
            if (typeof type === 'undefined' || !type) {
              return;
            }

            const position = reactFlowInstance.project({
              x: event.clientX - reactFlowBounds.left,
              y: event.clientY - reactFlowBounds.top,
            });

            const nodeID = getNodeID(type);
            const newNode = {
              id: nodeID,
              type,
              position,
              data: getInitNodeData(nodeID, type),
            };

            addNode(newNode);
          }
        },
        [reactFlowInstance]
    );

    const onDragOver = useCallback((event) => {
        event.preventDefault();
        event.dataTransfer.dropEffect = 'move';
    }, []);

    return (
        <>
        <div ref={reactFlowWrapper} style={{width: '100wv', height: '70vh'}}>
            <ReactFlow
                nodes={nodes}
                edges={edges}
                onNodesChange={onNodesChange}
                onEdgesChange={onEdgesChange}
                onConnect={onConnect}
                onDrop={onDrop}
                onDragOver={onDragOver}
                onInit={setReactFlowInstance}
                nodeTypes={nodeTypes}
                proOptions={proOptions}
                snapGrid={[gridSize, gridSize]}
                connectionLineType='smoothstep'
            >
                <Background color="#aaa" gap={gridSize} />
                <Controls />
                <MiniMap />
            </ReactFlow>
        </div>
        </>
    )
}

```

src\submit.jsx

```jsx
// submit.js

export const SubmitButton = () => {

    return (
        <div style={{display: 'flex', alignItems: 'center', justifyContent: 'center'}}>
            <button type="submit">Submit</button>
        </div>
    );
}

```

src\draggableNode.jsx

```
// draggableNode.js

export const DraggableNode = ({ type, label }) => {
    const onDragStart = (event, nodeType) => {
      const appData = { nodeType }
      event.target.style.cursor = 'grabbing';
      event.dataTransfer.setData('application/reactflow', JSON.stringify(appData));
      event.dataTransfer.effectAllowed = 'move';
    };

    return (
      <div
        className={type}
        onDragStart={(event) => onDragStart(event, type)}
        onDragEnd={(event) => (event.target.style.cursor = 'grab')}
        style={{
          cursor: 'grab',
          minWidth: '80px',
          height: '60px',
          display: 'flex',
          alignItems: 'center',
          borderRadius: '8px',
          backgroundColor: '#1C2536',
          justifyContent: 'center',
          flexDirection: 'column'
        }}
        draggable
      >
          <span style={{ color: '#fff' }}>{label}</span>
      </div>
    );
  };

```

src\nodes\inputNode.jsx

```
// inputNode.js

import { useState } from 'react';
import { Handle, Position } from 'reactflow';

export const InputNode = ({ id, data }) => {
  const [currName, setCurrName] = useState(data?.inputName || id.replace('customInput-', 'input_'));
  const [inputType, setInputType] = useState(data.inputType || 'Text');

  const handleNameChange = (e) => {
    setCurrName(e.target.value);
  };

  const handleTypeChange = (e) => {
    setInputType(e.target.value);
  };

  return (
    <div style={{width: 200, height: 80, border: '1px solid black'}}>
      <div>
        <span>Input</span>
      </div>
      <div>
        <label>
          Name:
          <input
            type="text"
            value={currName}
            onChange={handleNameChange}
          />
        </label>
        <label>
          Type:
          <select value={inputType} onChange={handleTypeChange}>
            <option value="Text">Text</option>
            <option value="File">File</option>
          </select>
        </label>
      </div>
      <Handle
        type="source"
        position={Position.Right}
        id={`${id}-value`}
      />
    </div>
  );
}

```

src\nodes\llmNode.jsx

```
// llmNode.js

import { Handle, Position } from 'reactflow';

export const LLMNode = ({ id, data }) => {

  return (
    <div style={{width: 200, height: 80, border: '1px solid black'}}>
      <Handle
        type="target"
        position={Position.Left}
        id={`${id}-system`}
        style={{top: `${100/3}%`}}
      />
      <Handle
        type="target"
        position={Position.Left}
        id={`${id}-prompt`}
        style={{top: `${200/3}%`}}
      />
      <div>
        <span>LLM</span>
      </div>
      <div>
        <span>This is a LLM.</span>
      </div>
      <Handle
        type="source"
        position={Position.Right}
        id={`${id}-response`}
      />
    </div>
  );
}
```

src\nodes\outputNode.jsx

```jsx
// outputNode.js

import { useState } from 'react';
import { Handle, Position } from 'reactflow';

export const OutputNode = ({ id, data }) => {
  const [currName, setCurrName] = useState(data?.outputName || id.replace('customOutput-', 'output_'));
  const [outputType, setOutputType] = useState(data.outputType || 'Text');

  const handleNameChange = (e) => {
    setCurrName(e.target.value);
  };

  const handleTypeChange = (e) => {
    setOutputType(e.target.value);
  };

  return (
    <div style={{width: 200, height: 80, border: '1px solid black'}}>
      <Handle
        type="target"
        position={Position.Left}
        id={`${id}-value`}
      />
      <div>
        <span>Output</span>
      </div>
      <div>
        <label>
          Name:
          <input
            type="text"
            value={currName}
            onChange={handleNameChange}
          />
        </label>
        <label>
          Type:
          <select value={outputType} onChange={handleTypeChange}>
            <option value="Text">Text</option>
            <option value="File">Image</option>
          </select>
        </label>
      </div>
    </div>
  );
}

```

src\nodes\textNode.jsx

```jsx
// textNode.js

import { useState } from 'react';
import { Handle, Position } from 'reactflow';

export const TextNode = ({ id, data }) => {
  const [currText, setCurrText] = useState(data?.text || '{{input}}');

  const handleTextChange = (e) => {
    setCurrText(e.target.value);
  };

  return (
    <div style={{width: 200, height: 80, border: '1px solid black'}}>
      <div>
        <span>Text</span>
      </div>
      <div>
        <label>
          Text:
          <input
            type="text"
            value={currText}
            onChange={handleTextChange}
          />
        </label>
      </div>
      <Handle
        type="source"
        position={Position.Right}
        id={`${id}-output`}
      />
    </div>
  );
}

```