Project Structure Your project is organized as follows:

src/: Contains all React components, styles, and utility functions. components/: Contains individual component files for StockTable, StockChart, and StockDetail. App.js: The main application file that combines components. Component Breakdown

1. StockTable Component

```
import "../App.css";
import { useState, useMemo } from 'react';
import { useTable } from "react-table";

function StockTable(props) {
  return <Table data={props.stockData} />;
};
```

Imports: CSS for styling, useState and useMemo hooks from React for state management and performance optimization, and useTable from react-table for table functionalities. StockTable: A functional component that takes props and renders the Table component, passing stockData. Table Component

const Table = (props) => { const columns = useMemo(() => [ { Header: "Year", accessor: "year" }, //... other columns ], []);

Table Function: Defines the main table logic. Columns: Uses useMemo for performance optimization, defining the structure of the table with headers and accessors corresponding to data fields.

const { getTableProps, getTableBodyProps, headerGroups, rows, prepareRow } = useTable({ columns, data: props.data });

useTable: A hook that returns various properties and methods for table management, including props for the table and row preparation.

Rendering the Table

return (

{headerGroups.map(headerGroup => ( {headerGroup.headers.map(column => ( ))} ))} {rows.map(row => { prepareRow(row); return ( {row.cells.map(cell => ( ))} ); })}

| {column.render("Header")} |
|---|
| {cell.render("Cell")} |

);

Table Structure: Renders the table using JSX, mapping through header groups and rows. Props: Each element uses spread operator {...} to apply props returned by useTable. 2. StockChart Component This component uses react-chartjs-2 to render stock performance charts. Here’s a brief breakdown:

import { Line } from "react-chartjs-2"; import { Chart, registerables } from 'chart.js';

Chart.register(...registerables);

Imports: Importing the Line component from react-chartjs-2 for line charts and registering necessary components from Chart.js. Chart Data and Options

const datasets = props.stockDataList.map((item, index) => ({ label: `${item.symbol} : ${annualReturn}%`, //... other dataset properties }));

const datasets = props.stockDataList.map((item, index) => ({ label: `${item.symbol} : ${annualReturn}%`, //... other dataset properties }));

Data Mapping: Maps through stockDataList to create datasets for the chart, including labels and data points. Chart Rendering

<Line data={{ datasets }} options={{ /* options here */ }} />

Line Component: Renders the line chart using the constructed datasets and customizable options for aesthetics and behavior. 3. StockDetail Component This component likely displays in-depth information about a selected stock. function StockDetail(props) { // Detailed stock information rendering logic }

Props: Receives specific stock details to display relevant metrics and historical data. Styling Tailwind CSS: Applied throughout the project for responsive design and utility-based styling. Classes like flex, text-center, bg-white, etc., are used for layout and appearance. Project Functionality User Interactions: Users can hover over table rows to see detailed information, click on stocks to view charts, and analyze stock trends over time. Dynamic Data: The project handles dynamic stock data and renders it in a user-friendly format.