Atlan Frontend Internship Task 2025

SQL Query Editor

Submitted By: Raghav Bansal

Submitted To: Atlan

Table of Contents

Project Overview

Purpose

The **SQL Query Editor** is a modern web-based application designed to provide an intuitive interface for writing, executing, and visualizing SQL queries. It integrates best practices in web development and ensures efficient data handling, making query execution seamless and interactive.

Key Objectives

- Offer a **professional** SQL query editing experience.
- Enable **efficient query execution** with optimized performance.
- Support **query history** management for easy reference.
- Provide **multiple data visualization formats** for insights.
- Ensure a **responsive, performant, and user-friendly interface**.

System Architecture

Component Structure

```
src/app/
├── components/
│   ├── QueryEditor.tsx       # SQL editor component
│   ├── ResultsTable.tsx      # Query results display
│   ├── QuerySelector.tsx     # Predefined queries
│   ├── QueryHistory.tsx      # Query history management
│   ├── Navbar.tsx            # Navigation and theme toggle
├── styles/
│   ├── theme.ts              # Theme configuration
│   ├── styled.d.ts           # Styled-components types
└── page.tsx                  # Main application entry
```

Data Flow

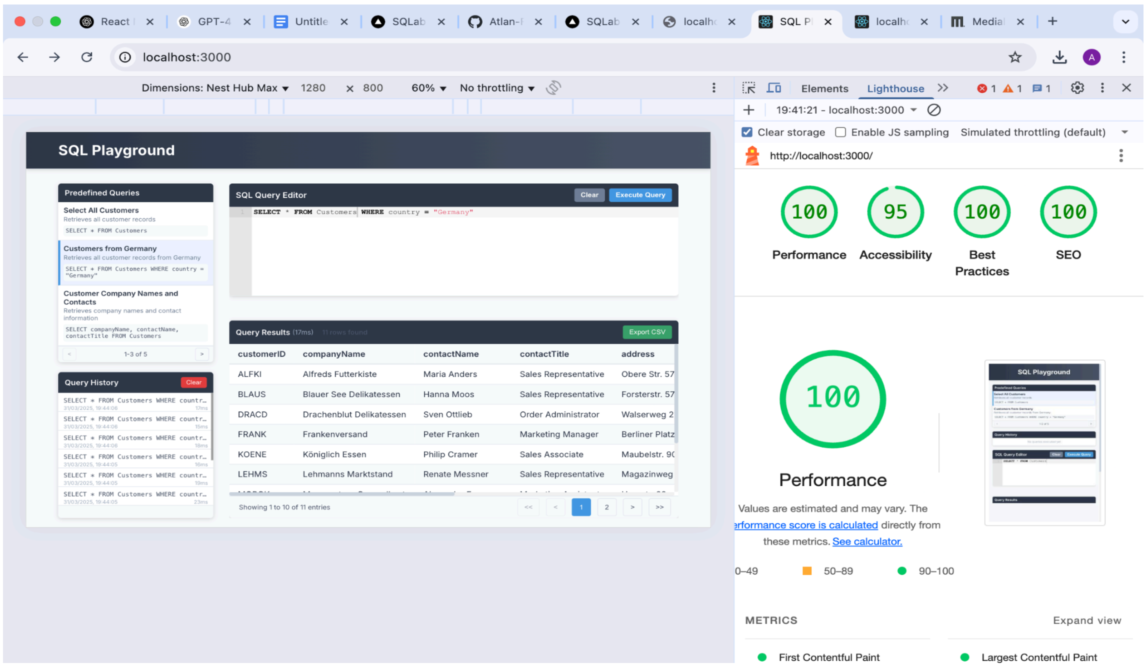
1. **User Input** → Query Editor
2. **Query Execution** → Results Processing
3. **Results Display** → Data Table
4. **History Update** → Local Storage

State Management

```
const [queryText, setQueryText] = useState(predefinedQueries[0].query);
const [results, setResults] = useState(predefinedQueries[0].data);
const [executionTime, setExecutionTime] = useState(0);
const [historyItems, setHistoryItems] = useState<HistoryItem[]>([]);
```

Performance Testing for the Website:

Page Load Times (Google Lighthouse used, Chrome Version 134.0.6998.166 based on Chromium V92)



localhost:3000

Dimensions: Nest Hub Max 1280 × 800 60% No throttling

SQL Playground

Predefined Queries

Select All Customers
Retrieves all customer records
`SELECT * FROM Customers`

Customers from Germany
Retrieves all customer records from Germany
`SELECT * FROM Customers WHERE country = "Germany"`

Customer Company Names and Contacts
Retrieves company names and contact information
`SELECT companyName, contactName, contactTitle FROM Customers`

1-3 of 5

Query History Clear

```
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:06                17ms
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:06                15ms
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:06                18ms
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:05                16ms
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:05                19ms
SELECT * FROM Customers WHERE countr…
31/03/2025, 19:44:05                23ms
```

SQL Query Editor Clear Execute Query

```sql
SELECT * FROM Customers WHERE country = "Germany"
```

Query Results (17ms) 11 rows found Export CSV

customerID	companyName	contactName	contactTitle	address
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	Walserweg 2
FRANK	Frankenversand	Peter Franken	Marketing Manager	Berliner Platz
KOENE	Königlich Essen	Philip Cramer	Sales Associate	Maubelstr. 90
LEHMS	Lehmanns Marktstand	Renate Messner	Sales Representative	Magazinweg

Showing 1 to 10 of 11 entries

Elements Lighthouse

19:41:21 - localhost:3000

☑ Clear storage ☐ Enable JS sampling Simulated throttling (default)

http://localhost:3000/

Performance	Accessibility	Best Practices	SEO
100	95	100	100

100
Performance

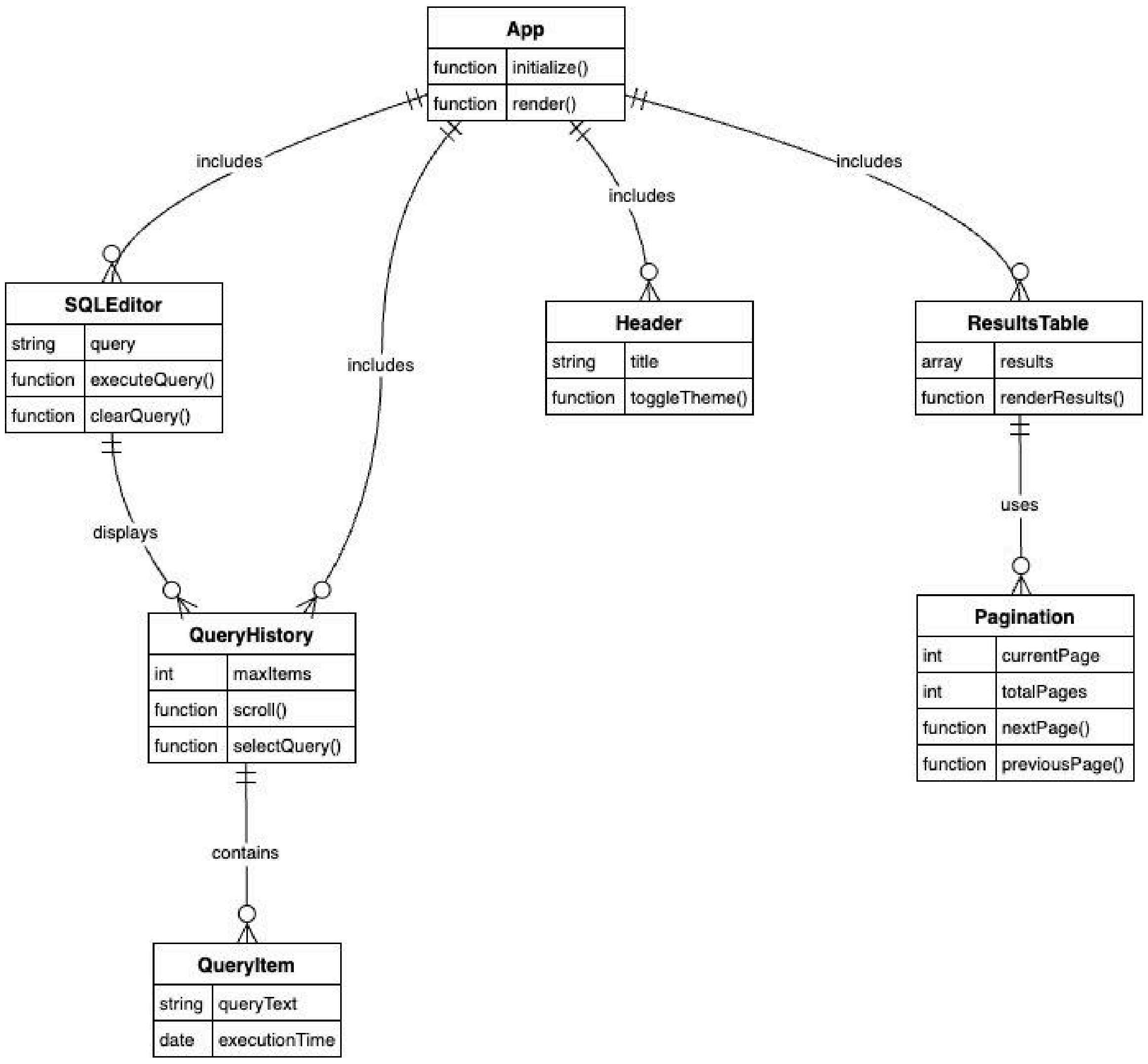
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

0–49 🟧 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint ● Largest Contentful Paint

E-R Diagram :



Technical Stack

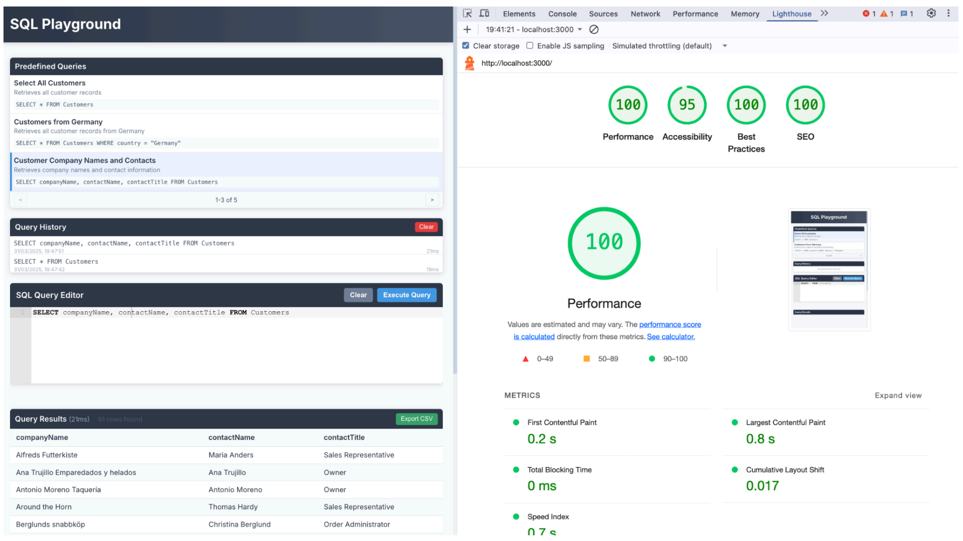
Core Technologies

- **Framework**: Next.js 15.2.4
- **Language**: TypeScript
- **UI Library**: React 19
- **Styling**: Styled-components

Key Dependencies

```
{
  "dependencies": {
    "@monaco-editor/react": "^4.7.0",
    "chart.js": "^4.4.8",
    "next": "15.2.4",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "react-icons": "^5.5.0",
    "styled-components": "^6.1.16"
  }
}
```

Responsive according to screen size



SQL Playground

Predefined Queries

Select All Customers
Retrieves all customer records
```
SELECT * FROM Customers
```

Customers from Germany
Retrieves all customer records from Germany
```
SELECT * FROM Customers WHERE country = "Germany"
```

Customer Company Names and Contacts
Retrieves company names and contact information
```
SELECT companyName, contactName, contactTitle FROM Customers
```

< 1-3 of 5 >

Query History — Clear

```
SELECT companyName, contactName, contactTitle FROM Customers
31/03/2025, 19:47:51                                   21ms
SELECT * FROM Customers
31/03/2025, 19:47:42                                   19ms
```

SQL Query Editor — Clear — Execute Query

```
SELECT companyName, contactName, contactTitle FROM Customers
```

Query Results (21ms) 91 rows found — Export CSV

companyName	contactName	contactTitle
Alfreds Futterkiste	Maria Anders	Sales Representative
Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
Antonio Moreno Taquería	Antonio Moreno	Owner
Around the Horn	Thomas Hardy	Sales Representative
Berglunds snabbköp	Christina Berglund	Order Administrator

☑ Clear storage ☐ Enable JS sampling Simulated throttling (default)

http://localhost:3000/

Performance	Accessibility	Best Practices	SEO
100	95	100	100

Performance 100

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS — Expand view

Metric	Value
First Contentful Paint	0.2 s
Largest Contentful Paint	0.8 s
Total Blocking Time	0 ms
Cumulative Layout Shift	0.017
Speed Index	0.7 s

Features

1. SQL Query Editor

- **Monaco Editor** integration for syntax highlighting and code formatting.
- **Real-time validation** for SQL syntax.
- **Autocomplete support** for SQL keywords.

2. Query Execution & Results Display

- **Efficient execution** with response time tracking.
- **Virtualized table rendering** for large datasets.
- **Sorting, pagination, and CSV export** support.

3. Query History Management

- **Stores past queries** for quick access.
- **Allows favoriting/starred queries** for frequent use.

4. Theming & UI Customization

- **Light/Dark mode** support.
- **Styled-components** for dynamic UI updates.

Implementation Details

Theme System

```
export const theme: DefaultTheme = {
  colors: {
    border: '#e2e8f0',
    background: '#ffffff',
    text: '#1a202c',
    primary: '#2563eb',
  },
  spacing: {
    xs: '4px',
    sm: '8px',
  }
};
```

Query Execution Logic

```
const executeQuery = useCallback(() => {
  const startTime = performance.now();
  setTimeout(() => {
    const endTime = performance.now();
    setExecutionTime(endTime - startTime);
    setResults(activeQuery.data);
    setHistoryItems(prev => [{ id: Date.now(), query: queryText, timestamp: new Date()
}, ...prev]);
  }, 500);
}, [activeQuery, queryText]);
```

Performance Optimization

1. Code Splitting

- Lazy loading **Monaco Editor**.
- Component-based code splitting.
- **Dynamic imports** for heavy components.

2. Rendering Optimization

```
const sortedData = useMemo(() => {
  return data.sort((a, b) => a[sortConfig.key] - b[sortConfig.key]);
}, [data, sortConfig]);
```

This improves rendering optimization by preventing unnecessary re-renders when dependencies remain unchanged.

3. State Management

- **Memoized state updates** to prevent unnecessary re-renders.
- **Efficient reactivity** with `useCallback` and `useMemo`.

Testing and Quality Assurance

1. Type Safety

- **TypeScript enforcement** for data structures.
- **Strict interface definitions** for API interactions.

2. Code Quality

- **ESLint & Prettier** for linting and formatting.
- **Consistent coding style** maintained across components.

Deployment

Local Development

```
# Installation
npm install

# Development server
npm run dev

# Production build
npm run build
npm start
```

Production Deployment

- **Vercel** deployment setup.

Future Enhancements

Planned Features

1. **Real database connectivity** for executing actual SQL queries.
2. **SQL query validation** for syntax error detection.
3. **Advanced data visualizations** (heatmaps, bar charts, etc.).
4. **User authentication & accounts** for saved preferences.
5. **AI-assisted SQL query suggestions**.
6. **Query execution plan visualization**.

Technical Improvements

1. **Error tracking** using **Sentry**.
2. **Performance monitoring** via **New Relic**.
3. **Enhanced accessibility & mobile responsiveness**.
