LLM LAB – Frontend

Technical Documentation

This document provides a comprehensive technical overview of the frontend project, including setup instructions, architecture, technology stack, API strategy, component structure, UI/UX design rationale, quality metrics explanation, and deployment considerations.

# 1. Setup Instructions

Follow these steps to set up and run the project locally:

1. Clone the repository from Github - https://github.com/sitakanta007/llm-lab-frontend

2. Navigate to the project folder.

3. Run `npm install` to install dependencies.

4. Configure environment variables (such as API base URL).

5. Run `npm run dev` to start the development server.

6. Access the application in your browser at http://localhost:3000

**Recommended Node version: 18.x or above**

**Package Manager: npm**

# 2. Quick Start Guide

- Install dependencies and run the development server.  
- Open the application in a browser and navigate to the dashboard.  
- Run or load an experiment to view results.  
- Use comparison and visualization features to analyze data.  
- Export results if needed.

# 3. Folder Structure

The project directory is organized in a feature-driven structure for modularity and scalability.

llm-lab-frontend/  
 jsconfig.json  
 next.config.mjs  
 package.json  
 postcss.config.js  
 README.md  
 tailwind.config.js  
 hooks/  
 useMounted.js  
 store/  
 experimentSlice.js  
 index.js  
 styles/  
 globals.css  
 api/  
 axiosClient.js  
 experimentApi.js  
 app/  
 layout.jsx  
 page.jsx  
 experiments/  
 page.jsx  
 [id]/  
 page.jsx  
 components/  
 Providers.jsx  
 ResultCard.jsx  
 Charts/  
 ChartSkeleton.jsx  
 PremiumChart.jsx  
 Compare/  
 CompareFloatingBar.jsx  
 CompareModal.jsx  
 Home/  
 DashboardCard.jsx  
 PromptSection.jsx  
 Response/  
 ResponseCard.jsx  
 ResponseGrid.jsx  
 Utils/  
 ExportButton.jsx  
 InlineError.jsx  
 NavTabs.jsx  
 ParamCombos.jsx  
 ParamInput.jsx  
 ScrollToTopButton.jsx  
 SliderField.jsx  
 ThemeToggle.jsx  
 ToolTip.jsx  
 config/  
 paramsConfig.js

# 4. Technology Stack

The project uses the following technologies:

## Frontend Framework & Language

Next.js (React-based), JavaScript with JSX

## Styling

Tailwind CSS, Global styles managed via styles/globals.css, Utility-first styling approach

## State Management

Redux Toolkit, Centralized state handling through slices in the store directory.

## Build & Config

Next.js build pipeline, PostCSS, Tailwind configuration files, jsconfig.json for path alias.

# 5. Architectural Approach & Key Decisions

## 5.1 Data Flow

The application follows a unidirectional data flow pattern:

- Users interact with UI components.  
- Components dispatch actions.  
- Redux store updates the global state.  
- UI re-renders based on the updated state.  
- API layer remains decoupled for clean separation of concerns.

## 5.2 API Endpoints

All API calls are handled via Axios client defined in api/axiosClient.js.   
Experiment-related endpoints are managed in experimentApi.js.   
This modular API layer allows easy maintenance and reusability.

## 5.3 Component Structure

- Components are organized by feature folders.  
- Reusable components are kept inside Utils.  
- Page components live in the ‘app/’ folder following Next.js App Router conventions.  
- Charts, comparison modals, and dashboard cards are separated by logical grouping.

# 6. UI/UX Design Rationale

The UI/UX design emphasizes simplicity, responsiveness, and a clean interface:  
- Consistent use of Tailwind utility classes ensures responsive design.  
- Minimalist dashboards with clear visual hierarchy.  
- Interactive elements like sliders, tooltips, info chips, and modals enhance usability.  
- Dark/light theme toggle provides accessibility and personalization.

## User Journey

1. User lands on the home dashboard.  
2. User navigates to prompt section and creates experiments.  
3. System fetches and displays experiment results.  
4. User can compare results visually using charts, or through comparison modals.  
5. Export and customization options are available to the user.

# 7. Quality Metrics Visualization

The application includes visualization of quality metrics using chart components (PremiumChart, ChartSkeleton).  
These charts are rendered based on experiment results and model outputs.

## Metrics Displayed

- Temperature  
- Top\_p  
- Coherence Input  
- Frequency\_penalty

# 8. Deployment

The application is deployment on Vercel hosting platforms.  
Build and deployment steps:  
1. Run `npm run build` to create production build. (For Vercel deployments, configure Github account)  
2. Deploy the output folder to the hosting provider.  
3. Configure environment variables in production.  
4. Set up a custom subdomain via DNS using Amazon Route 53 if required.

# 9. Summary

- Modular architecture with reusable components.  
- Clean API integration and data flow.  
- Strong UI/UX design principles with responsive layouts.  
- Built-in visualization for metrics.